Ähnlichkeitsbasierte Modellierungsunterstützung für Geschäftsprozesse

Zur Erlangung des akademischen Grades eines
Doktors der Wirtschaftswissenschaften

(Dr. rer. pol.)

von der Fakultät für
Wirtschaftswissenschaften
der Universität Fridericiana zu Karlsruhe

genehmigte

DISSERTATION

von

Dipl.-Kffr. Agnes Koschmider

Tag der mündlichen Prüfung: 25.07.2007
Referent: Prof. Dr. Andreas Oberweis
Korreferent: Prof. Dr. Hagen Lindstädt

2007 Karlsruhe

Vorwort

Die Ergebnisse dieser Arbeit wurden während meiner Tätigkeit als wissenschaftliche Mitarbeiterin am Institut für Angewandte Informatik und Formale Beschreibungsverfahren (AIFB) an der Universität Karlsruhe (TH) erzielt.

Mein besonderer Dank gilt meinem Doktorvater Prof. Dr. Andreas Oberweis, der mir viel freien Raum für meine Forschung gelassen hat. Seine ständige Bereitschaft, auf meine jeweiligen Probleme einzugehen, war für mich eine große Bereicherung. Prof. Dr. Dr.h.c Wolffried Stucky danke ich für seine freundliche Unterstützung während meiner gesamten Zeit am Institut AIFB. Ebenso bedanke ich mich bei Prof. Dr. Hagen Lindstädt für die Übernahme des Korreferates sowie bei Prof. Dr. Rudi Studer und Prof. Dr. Karl-Heinz Waldmann für die Begleitung meiner mündlichen Prüfung.

Allen meinen Kollegen danke ich für ihre Hilfsbereitschaft und gute Zusammenarbeit in sehr angenehmer Atmosphäre. Stefanie Betz möchte ich danken, dass sie stets Zeit hatte, sich meine Ideen anzuhören und diese kritisch zu hinterfragen. Sehr danken möchte ich Dr. Marco Mevius, der mir mit seinen zahlreichen Hinweisen stets als wertvoller Mentor während meiner gesamten Zeit am Institut AIFB zur Seite stand. Ein besonderer Dank gilt Dr. Daniel Sommer für das zügige und gründliche Korrekturreferat und die konstruktiven Verbesserungsvorschläge. Dank gilt auch den zahlreichen Studien–, und Diplomarbeitern für ihre engagierte Mitarbeit.

Ich danke auch allen meinen Koautoren für die stets fruchtbare Zusammenarbeit: Stefanie Betz, Emmanuel Blanchard, Saartje Brockmans, Rebecca Bulander, Marc Ehrig, Thomas Hornung, Thomas Karle, Stefan Klink, Yu Li, Jan Mendling, Marco Mevius, Andreas Oberweis, Daniel Ried, Ute Rusnak, Daniel Sommer, Rudi Studer, Ralf Trunko und Markus Zaich.

Meinen Freundinnen Wasiliki Bouka, Cathleen Kassun, Daniela Kölbel, Silvia Naether, Silke Philipps, Dr. Tatjana Podgayetskaya, Bettina Sachse und Nadine Wellermann danke ich, dass sie mich immer zu meiner Arbeit motiviert haben. Vor allem möchte ich Heike Frankenberger danken, die mich in jeder Lebenssituation ihren Rückhalt hat spüren lassen.

Meinen Eltern danke ich für ihre uneingeschränkte Unterstützung, meine persönlichen Ziele jederzeit verfolgen zu können. Meiner Schwester Margarethe danke ich, dass sie stets jeden Schritt meiner Ausbildung gefördert hat. Meinem Bruder Thomas danke ich für die vielen Gespräche zu Metathemen und für seine Begeisterung, die er immer für meine Ideen aufgebracht hat.

Nicht zuletzt gilt mein ganz besonderer Dank meinem Freund Alexander Paar. Seine breite Allgemeinbildung und vor allem sein umfangreiches Wissen auf dem Gebiet der Informatik haben mir in den entsprechenden Situationen die Augen geöffnet. Er hat dazu beigetragen, dass ich nicht nur im Alltag sondern auch bei der wissenschaftlichen Arbeit Probleme sorgfältiger löse.

Karlsruhe, August 2007

Agnes Koschmider

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextauszüge

Definitionen

1 Einleitung

In diesem Kapitel werden zunächst nach Einführung in die bestehenden Problemstellungen bei der Modellierung von Geschäftsprozessen die Zielsetzung dieser Arbeit und der vorgeschlagene Lösungsweg vorgestellt. Anschließend wird der methodische Aufbau der Arbeit beschrieben und motiviert.

1.1 Problemstellung

Die präzise Modellierung von Geschäftsprozessen bildet die Grundlage für den Einsatz standardisierter Informationssysteme, die eine zielgerichtete Zuordnung von Daten- und Kommunikationsdiensten zu Anwendern unterstützen. Bei Veränderung von Anforderungen für diese Informationssysteme werden die entsprechenden Geschäftsprozesse häufig nicht angepasst, sondern neu modelliert, da eine Anpassung zeitaufwändiger wäre. Dabei könnte durch die Wiederverwendung von bereits modellierten Prozessteilen der Entwurfs- und Implementierungsaufwand von Informationssystemen (für die die Geschäftsprozesse die Grundlage bilden) erheblich reduziert werden. Ein Problem, das eine Anpassung der entsprechenden Prozessteile erschwert, besteht in der Schwierigkeit, zwei prinzipiell gleiche Geschäftsprozessmodelle zu identifizieren. Die meisten Modellierer vermischen die Abstraktionsgrade von Prozessaktivitäten: auf der gleichen Prozessebene werden einige Aktivitäten detaillierter und andere weniger detailliert dargestellt, obwohl ein gleicher Detaillierungsgrad für den konsistenten Entwurf eines Informationssystems wichtig ist. Das Ergebnis dieser unpräzisen Vorgehensweise sind Geschäftsprozessmodellvarianten, die nur mit viel Modellierungserfahrungen als solche erkannt werden können. Hinzu kommt, dass Modellierer ihre Geschäftsprozessmodelle in der Regel nicht mit demselben Vokabular beschreiben. Zur Modellierung von Prozessaktivitäten werden branchenspezifische Begrifflichkeiten verwendet, die auch bei Unternehmen gleicher Branche variieren können. Der Rohstofflieferant verwendet zur Beschreibung von Bestellungen den Begriff *commission*, der Erzeuger *order* und der Montagebetrieb

purchase order[1]. Auch das erschwert die Anpassung von Geschäftsprozessen, da die Erkennung unterschiedlicher Begriffe für gleiche Prozessobjekte entsprechendes Kontextwissen voraussetzt. In der Literatur wird deswegen ein „kontrolliertes" Vokabular zur Geschäftsprozessmodellierung gefordert.

In den am Markt verfügbaren Werkzeugen zur Geschäftsprozessmodellierung existiert bislang keine Funktion, die eine Anpassung von Geschäftsprozessmodellen unterstützt.

Ein weiteres Defizit der bestehenden Werkzeuge liegt in der unzureichenden Unterstützung des Modellierers, Unternehmensrestriktionen bei der Anpassung einzuhalten. Mit Hilfe der Geschäftsprozessanalyse können Prozessmodelle validiert, verifiziert und bewertet werden [vgl. exemplarisch Aals98, MBCD95]. Es kann überprüft werden, ob das Prozessmodell die Wirklichkeit richtig abbildet oder ob das Prozessmodell korrekt ist (Verklemmungsfreiheit, Lebendigkeit[2]) und einer bestimmten Leistungsfähigkeit entspricht. Allerdings kann nicht überprüft werden, ob der modellierte Realweltausschnitt den Restriktionen (Geschäftsregeln) des Unternehmens entspricht. Dabei ist die Fähigkeit, die Geschäftsregeln [Hall02] konsistent in das Informationssystem zu implementieren, ein kritischer Erfolgsfaktor von E-Business [KnEn04].

Im Rahmen dieser Arbeit wird erstmalig ein Unterstützungssystem für die Geschäftsprozessmodellierung vorgestellt, das eine Anpassung durch Wiederverwendung von bereits modellierten Geschäftsprozessteilen ermöglicht. Der Benutzer kann nach bereits modellierten Prozessteilen in einer Prozessbibliothek mit Hilfe einer in dieser Arbeit entwickelten Anfragesprache suchen. Zusätzlich wird in der Arbeit eine Empfehlungskomponente präsentiert, die dem Modellierer Prozessteile passend zu seinem gerade editierten Prozessmodell auf Basis von Geschäftsregeln aus der Prozessbibliothek vorschlägt. So werden die Prozessmodelle gleichzeitig an die jeweilige Umwelt angepasst. Die Empfehlungskomponente wird weiterhin dahingehend erweitert, dass auch das Abstraktionsniveau eines gerade editierten Geschäftsprozessmodells mit den empfohlenen Prozessteilen gleich bleibt. Somit sollen Modellierungsfehler weitgehend vermieden und ein konsistenter Entwurf des Informationssystems unterstützt werden. In der Arbeit wird auch die Einschränkung eines „kontrollierten" Vokabulars aufgehoben. Es werden stattdessen Ähnlichkeitsmaße definiert, durch die Prozessteile mit unterschiedlichem Vokabular zur Beschreibung gleicher Prozessobjekte wieder verwendet werden können.

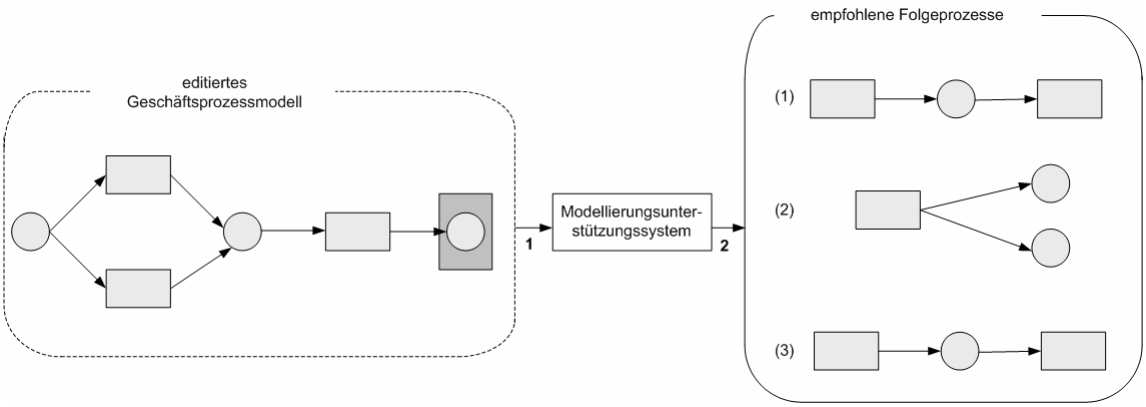
---

[1] Die in dieser Arbeit vorgestellten Methoden zur Modellierungsunterstützung wurden für Geschäftsprozesse aufgestellt, die in englischer Sprache modelliert wurden. Aus diesem Grund werden in der Problemstellung bereits englische Begriffe verwendet.

[2] Definition zu Lebendigkeit siehe [Reis86, S. 84].

Die Modellierungsunterstützung funktioniert wie in Abbildung 1 skizziert. Der Modellierer möchte ab einem bestimmten Prozesselement (in der Abbildung ist es eine Stelle, die durch ein dunkelgraues, umschließendes Rechteck hervorgehoben ist) Empfehlungen für passende Folgeprozesse bekommen (Schritt 1). Daraufhin schlägt ihm die Empfehlungskomponente drei passende Folgeprozesse vor (Schritt 2), aus denen er den für ihn relevanten Prozess auswählt und in seiner Modellierungsumgebung gegebenenfalls verändern kann. Alternativ dazu kann der Modellierer auch die Anfragesprache verwenden, die als Ergebnis ebenfalls passende Prozesselemente ausgibt.

Abbildung 1: Beispiel für eine Modellierungsunterstützung für Geschäftsprozesse



Ein besonderer Vorteil dieser Modellierungsunterstützung ist, dass die vorgeschlagenen Folgeprozesse das gleiche Abstraktionsniveau wie der gerade editierte Geschäftsprozesse haben und annotierten Geschäftsregeln entsprechen. Zusätzlich werden beim Vorschlag strukturelle Eigenschaften berücksichtigt, mit denen unerwünschte Deadlocks vermieden werden können.

Zur Realisierung einer Modellierungsunterstützung müssen zunächst Geschäftsprozesse modelliert werden. Zur Modellierung komplexer Geschäftsprozessmodelle und der prozessrelevanten Objekte eignen sich höhere Petri-Netze [GeLa81, DeOb96, Lenz03, Ober96]. Die formale Fundierung von Petri-Netzen ermöglicht eine korrekte syntaktische Verbindung eines editierten Geschäftsprozessmodells mit dem vorgeschlagenen Folgeprozess [Aals00, AaWe01]. Bestehende Analysemethoden für Petri-Netze [Pete77, SaOr00] können zur syntaktischen Überprüfung der verbundenen Geschäftsprozessmodelle verwendet werden.

Durch die Verwendung von Ontologien [Grub93a, Grub93b, StSt04] kann ein einheitliches Verständnis über Begriffe und die Beziehungen zwischen Begriffen von Geschäftsprozessmodellen definiert werden. Ontologien ermöglichen es, Missverständnisse in Form von Synonymen oder Homonymen aufzudecken und Abstraktionsgrade von Begriffen zu bestimmen. Die formale Beschreibung von Ontologien

erlaubt es zudem, Inferenzmechanismen [HuRy04] einzusetzen, um aus den annotierten Geschäftsregeln die passenden Folgeprozesse ermitteln zu können. Dadurch kann die Wiederverwendung von Geschäftsprozessmodellen erleichtert und die Interoperabilität in heterogenen Systemlandschaften verbessert werden.

1.2 Ziel der Arbeit

Hauptziel dieser Arbeit ist es, Methoden zu entwickeln, mit denen die Modellierung von Geschäftsprozessen benutzerindividuell unterstützt und somit die Fehlerrate von Geschäftsprozessmodellen weitgehend reduziert werden kann. Dabei sollte die Unterstützung von Benutzern während der Geschäftsprozessmodellierung entweder teilautomatisch und/oder automatisch durch ein Unterstützungssystem umgesetzt werden (abhängig von den Benutzerwünschen und dem Benutzertyp). Eine solche Unterstützung existiert bislang noch in keinem Geschäftsprozessmodellierungswerkzeug. Die Modellierungsunterstützung soll unabhängig von einem Modellierungswerkzeug funktionieren; Petri-Netz-basierte Geschäftsprozesse, die mit unterschiedlichen Werkzeugen modelliert wurden, sollen zwischen den Werkzeugen (möglichst ohne Informationsverlust) austauschbar sein. Hierfür soll ein allgemeingültiges XML-basiertes Speicher- und Austauschformat für einfache Petri-Netze erweitert werden.

Zusätzlich zu dem Austauschformat soll ein Beschreibungsformat für Petri-Netze entwickelt werden, welches den Einsatz von Inferenzmechanismen unterstützt und somit eine automatische Modellierungsunterstützung realisiert werden kann.

Eine besondere Schwierigkeit bei der Realisierung einer automatischen Modellierungsunterstützung ergibt sich dadurch, dass Modellierer während der Prozessmodellierung in der Regel Unternehmensrestriktionen in Form von Geschäftsregeln einhalten sollen. Die Nichteinhaltung dieser Regeln ist eine mögliche Fehlerquelle für Geschäftsprozessmodelle. Das Beschreibungsformat muss dahingehend erweiterbar sein, dass es erlaubt, die Allgemeingültigkeit von Unternehmensrestriktionen für bestimmte Geschäftsprozessmodelle zu überprüfen.

Eine weitere Fehlerquelle von Geschäftsprozessmodellen ist die Nichteinhaltung eines identischen Abstraktionsniveaus für Prozessaktivitäten auf der gleichen Prozessebene. Petri-Netze unterstützen die Systemdarstellung auf verschiedenen Abstraktionsebenen. Bislang existiert aber keine Software-gestützte Methode, die das Abstraktionsniveau von Prozessaktivitäten überprüft. Im Rahmen der Modellierungsunterstützung soll eine solche Methode entwickelt werden.

Bestimmte Geschäftsprozessmodelle oder nur Teile davon sind nicht für alle Mitarbeiter eines Unternehmens zugänglich (unternehmenskritische Geschäftsprozess-

modelle). Bislang wird ein Geschäftsprozess mit unternehmenskritischen Prozessaktivitäten vollständig gesperrt und somit sind auch die öffentlichen Teile für die Mitarbeiter nicht sichtbar. Bei der Realisierung der Modellierungsunterstützung sollen die öffentlichen Teile des Geschäftsprozesses trotzdem verfügbar sein und als passende Folgeprozesse genutzt werden können.

Eine solche Modellierungsunterstützung mit Zugang zu öffentlichen Teilen von Geschäftsprozessmodellen kann mit Hilfe einer Anfragesprache umgesetzt werden. Diese setzt aber voraus, dass nur die „privaten" Teile des Geschäftsprozesses gesperrt werden. Der Benutzer formuliert in seiner Anfrage Kriterien, nach denen passende Folgeprozesse gesucht werden sollen. Die Anfragesprache sollte auf einer formalen Grammatik basieren, damit die Gültigkeit von Anfragen überprüft werden kann. Zusätzlich sollte die Anfragesprache allgemeingültigen Anforderungen an Anfragesprachen [HeSc91] genügen.

Bei der Modellierung von Geschäftsprozessmodellen wird in der Literatur ein „kontrolliertes" Vokabular vorausgesetzt, damit keine begrifflichen Missverständnisse entstehen können. Eine solche Einschränkung ist allerdings in Unternehmen üblicherweise nicht realisierbar, vor allem nicht bei einer unternehmensübergreifenden Prozessmodellierung (jeder Unternehmenspartner verwendet sein eigenes Vokabular). Um diese Einschränkung aufzuheben, sollen verschiedene Ähnlichkeitsmaße definiert werden, mit denen begriffliche Missverständnisse in Geschäftsprozessmodellen möglichst automatisch erkannt werden können.

Eine Ursache für begriffliche Missverständnisse können synonym bzw. homonym verwendete Begriffe sein. Eine andere Fehlerquelle entsteht durch Begriffe mit unterschiedlichem Abstraktionsniveau, die synonym gebraucht werden (z.B. "Information" vs. "Mitteilung"). Bei der Modellierung von Geschäftsprozessen können zusätzlich noch leicht Editierfehler entstehen. Die Ähnlichkeitsmaße sollen auch diese Editierfehler identifizieren. Mit der Aufhebung der Restriktion eines „kontrollierten" Vokabulars werden durch die Empfehlungskomponente und die Anfragesprache zusätzlich passende Folgeprozesse vorgeschlagen, die mit einem unterschiedlichen Vokabular modelliert wurden.

Um Benutzer individuell bei der Modellierung von Geschäftsprozessen zu unterstützen, soll eine Benutzerbefragung durchgeführt werden. Aus den Ergebnissen der Befragung sollen sich unterschiedliche Hilfestellungen abhängig von der Zugehörigkeit einer Person zu einer bestimmten Benutzergruppe (Anfänger, Fortgeschritten, Experte) ableiten lassen. Eine solche Evaluation wurde bislang noch nicht durchgeführt.

Die entwickelten Methoden für die Modellierungsunterstützung sollen auch in einem Software-Werkzeug implementiert werden. Schwerpunkt der Umsetzung ist, einen Software-Prototyp für allgemein verwertbare Ähnlichkeitsmessungen, die Anfragesprache und die Empfehlungskomponente zu implementieren, allerdings ohne Berücksichtigung von Software-Qualitätsmerkmalen wie Ergonomie oder Performance des Systems.
Die Unterstützung während der Geschäftsprozessmodellierung kann mit unterschiedlicher Zielsetzung eingesetzt werden:

- Sie kann Anfänger bei der Modellierung von Geschäftsprozessen unterstützen. Dabei wird der zu editierende Geschäftsprozess mit Geschäftsprozessmodellen verglichen, die in einem Repository (Prozessbibliothek) [KeEi06] gespeichert wurden. Werden im Repository Geschäftsprozessteile gefunden, die sich als passende Folgeprozesse eignen, so werden diese zur automatischen Vervollständigung des Geschäftsprozessmodells vorgeschlagen.

- Unternehmen, die ihre Geschäftsprozessmodelle verbinden wollen, dabei allerdings ein unterschiedliches Vokabular verwenden, können sich (über die Anfragesprache) Schnittstellen ausgeben lassen, an denen die unternehmensübergreifenden Geschäftsprozessmodelle integriert werden können.

- Die Modellierung von Geschäftsprozessen orientiert sich häufig an Geschäftsregeln. Beispielsweise erfordert bei der Versicherung die Begutachtung von Schäden über einer Schadenssumme von 10 000 € einen externen Gutachter. Mit Hilfe von Inferenzmechanismen werden bei einer automatischen Vervollständigung von Geschäftsprozessmodellen nur solche Prozessteile zugelassen, die nicht gegen die Geschäftsregeln verstoßen.

- Veränderungen in den Anforderungen an Informationssysteme bedürfen auch Anpassungen von implementierten Funktionalitäten und den darunter liegenden Geschäftsprozessmodellen. Geschäftsprozessmodellvarianten sind das Ergebnis dieser Anpassungen. Das Modellierungsunterstützungssystem kann eingesetzt werden, um bereits modellierte Geschäftsprozessvarianten schneller als solche aufdecken zu können.

- Die Idee der Modellierungsunterstützung kann auch für methodisch unterschiedlich modellierte Geschäftsprozesse eingesetzt werden, z.B. wenn in einem Repository Petri-Netz- oder EPK-basierte [KeNS92] Geschäftsprozessmodelle hinterlegt wurden und der Benutzer dennoch die Modellierungsunterstützung nutzen möchte.

- Generell ist die Idee der Modellierungsunterstützung übertragbar z.B. auf Formalismen zur konzeptuellen Modellierung (Entity-Relationship-Diagramme) oder auf Sprachen zur Modellierung von Software (Unified Modeling Language).

1.3 Aufbau der Arbeit

In einem einführenden Kapitel (Kapitel 2) werden die Grundlagen für die Entwicklung einer Unterstützung während der Geschäftsprozessmodellierung gelegt, die eine methodische Fundierung für die Realisierung einer solchen Funktion bilden. Dazu werden Ontologien und Sprachen zur Modellierung von Ontologien [Beck04, BrGu04, KiLW95] erklärt. Es wird insbesondere auf die Web Ontology Language (OWL) [McHa03] eingegangen, die eine formale Modellierung von Aussagen über Daten in einem maschinell interpretierbaren Datenformat erlaubt. Anschließend werden Grundlagen der Geschäftsprozessmodellierung und verschiedene Modellierungssprachen [AABC05, KeNS92, WMC02] skizziert. Der Schwerpunkt liegt dabei auf der Beschreibung von elementaren und höheren Petri-Netzen [GeLa81, Jens92, Pete77, Reis86, ReRo98a, ReRo98b] zur Darstellung von Geschäftsprozessmodellen. Ferner werden die Anforderungen und Komponenten eines Modellierungsunterstützungssystems vorgestellt, die eine Grundlage für alle nachfolgenden Kapitel bilden. Der Schwerpunkt bei der Umsetzung der Modellierungsunterstützung liegt darauf, dem Benutzer möglichst wenige Einschränkungen vorzuschreiben, damit er die Modellierungsunterstützung nutzen kann. Deshalb werden in diesem Kapitel Anforderungen an ein Modellierungsunterstützungssystem skizziert.

Kapitel 3 widmet sich der Erweiterung eines bestehenden XML-basierten Austauschformats für elementare [Kind03, BCHW03] und höhere Petri-Netze. XML (eXtensible Markup Language) [W3C00] ist ein Austausch- und Beschreibungsformat für strukturierte und semi-strukturierte Dokumente. Anschließend wird das OWL-basierte Beschreibungsformat für elementare und höhere Petri-Netze entwickelt [KoOb05, KoRi05]. Dieses OWL-basierte Beschreibungsformat wird in einer formalen Darstellung vorgestellt.

Im folgenden Kapitel 4 werden Ähnlichkeitsmaße für das OWL-basierte Beschreibungsformat für Petri-Netze definiert [EhKO07, KoOb07b]. Nachdem sich das OWL-basierte Beschreibungsformat von Petri-Netzen nur durch eine andere Formatdarstellung unterscheidet, sind die Ähnlichkeitsmaße auf alle Petri-Netz-basierten Geschäftsprozessmodelle anwendbar. Zur Aufdeckung von Homonymen und Synonymen zwischen zwei Prozesselementnamen werden drei Ähnlichkeitsmaße eingeführt: *syntaktische*, *linguistische* und *strukturelle* Ähnlichkeitsmaße. Für das linguistische Ähnlichkeitsmaß ist eine Wissensbasis [SEHH03] in Form von Ontologien erforderlich. Als Wissensbasis wird WordNet [Fell98] verwendet. Die Aufdeckung von ähnlichen Begriffen mit einem unterschiedlichen Abstraktionsniveau (Generalisierung, Spezialisierung) erfordert die Berechnung der linguistischen Ausprägung von Elementnamen [Lin98, Resn99, WuPa94]. Hierfür wird eine Ähnlichkeitsmetrik aus der Literatur für ein viertes Ähnlichkeitsmaß erweitert. Die neu definierten Ähnlichkeitsmaße können zu einer Gesamtähnlichkeit kombiniert werden (kombiniertes Ähnlichkeitsmaß), die es erlaubt, den Ähnlichkeitsgrad zwischen zwei Geschäftsprozessen zu bestimmen. Mit Hilfe eines Ähnlichkeitsalgorithmus wird der Einsatz der Ähnlichkeitsmaße zur Berechnung der Ähnlichkeit zwischen Geschäftsprozessmodellen vorgestellt.

Um Prozessfragmente nach bestimmten Kriterien aus einer Prozessbibliothek extrahieren zu können, wird im Kapitel 5 eine spezielle OWL-basierte Anfragesprache definiert. Mit Hilfe der Anfragesprache kann auch nach Schnittstellen von Prozessmodellen mit einem bestimmten Ähnlichkeitswert gesucht werden. Es wird überprüft, in wieweit diese OWL-basierte Anfragesprache die allgemeingültigen Anforderungen für Anfragesprachen, wie sie in der Literatur existieren, erfüllt. Die Anfragesprache basiert auf einer formalen Grammatik und ermöglicht somit, die Gültigkeit von Anfragen zu überprüfen. Zuletzt wird auch die Ausdrucksmächtigkeit der Anfragesprache diskutiert. Die Anfragesprache bildet die Grundlage für eine teilautomatische Unterstützung der Geschäftsprozessmodellierung.

Eine automatische Modellierungsunterstützung auf Basis einer Empfehlungskomponente wird im Kapitel 6 ausführlich dargestellt. Zunächst wird nur eine Empfehlungskomponente vorgestellt, die beim Vorschlag von passenden Folgeprozessen ausschließlich Geschäftsregeln berücksichtigt. Im zweiten Schritt wird die Funktion dahingehend erweitert, dass auch das Abstraktionsniveau der gerade editierten Prozessaktivitäten beachtet wird. Zuletzt wird auf unterschiedliche Begrifflichkeiten für gleiche Prozesselementnamen in Geschäftsprozessen eingegangen. Folgeprozes-

se, die zum editierten Geschäftsprozessmodell sehr ähnliche Prozessaktivitäten verwenden und auch den Geschäftsregeln entsprechen, werden bei der Empfehlung ebenfalls angezeigt.

Im Rahmen einer Benutzerbefragung wird untersucht, welche Anzahl an Folgeprozessen dem Modellierer empfohlen werden soll. Insbesondere soll evaluiert werden, aus wie vielen Prozessaktivitäten ein Folgeprozess bestehen muss, damit der Modellierer die Prozesse schnell nachvollziehen und einen passenden Folgeprozess auswählen kann. Es werden Personen mit unterschiedlichen Modellierungserfahrungen befragt. Durch Auswertung der Befragung mit Hilfe von Verfahren des maschinellen Lernens werden im Kapitel 7 benutzerindividuelle Empfehlungen für Modellierungsunterstützungen abgeleitet.
Bei der Berechnung des kombinierten Ähnlichkeitsmaßes haben die vier Ähnlichkeitsmaße (syntaktisch, linguistisch, strukturell und abstraktionsniveaubasiert) eine unterschiedliche Gewichtung bezüglich der Gesamtähnlichkeit. Zudem kann die Gewichtung entweder manuell oder automatisch mit Hilfe von Verfahren des maschinellen Lernens zugewiesen werden, indem eine passende initiale Gewichtung gelernt wird. Diese Verfahren und die individuelle Wertigkeit von Ähnlichkeitsmaßen werden ebenfalls im Rahmen von Kapitel 7 vorgestellt.

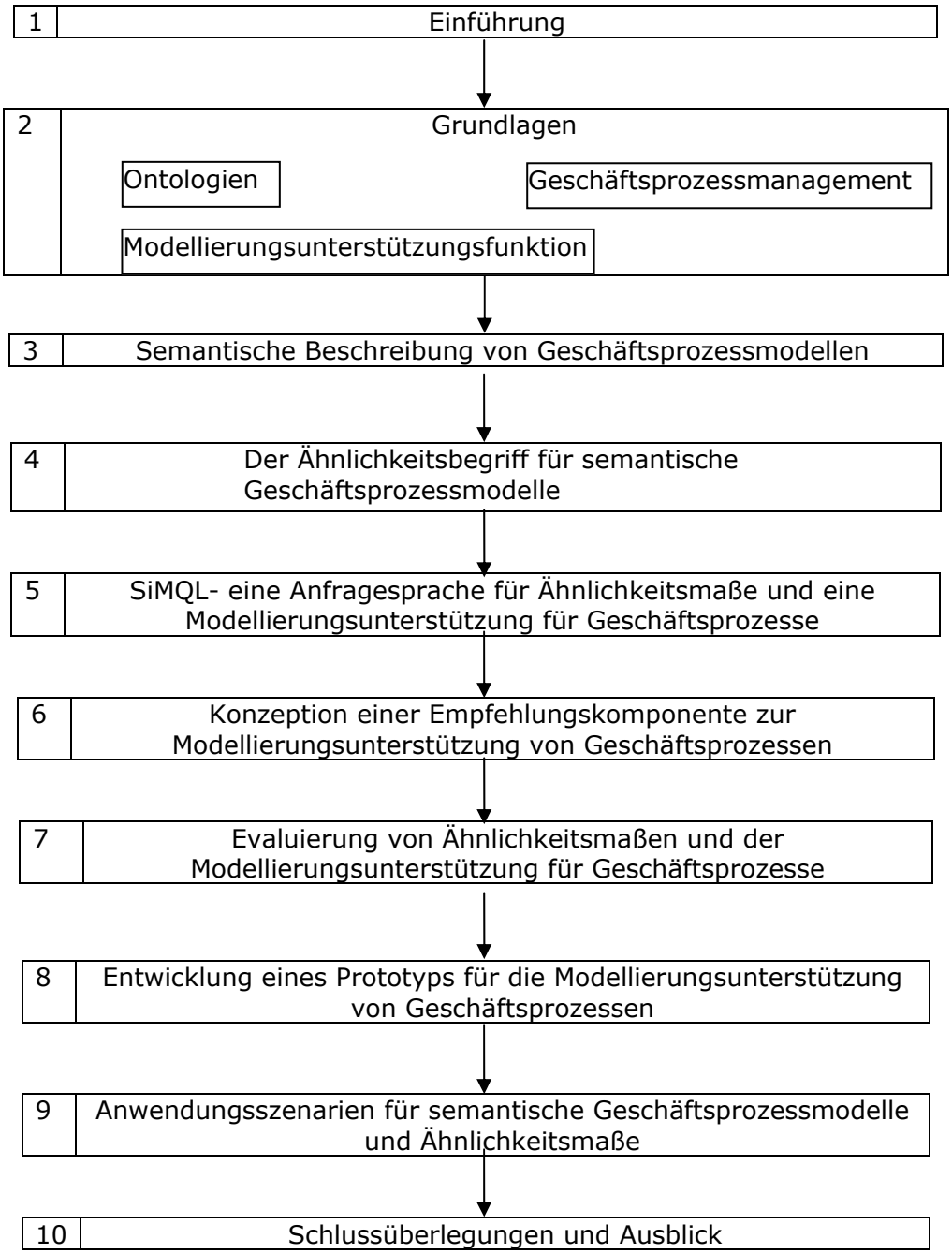
Die technische Umsetzung der teilautomatischen und automatischen Unterstützung während der Geschäftsprozessmodellierung wird im Kapitel 8 präsentiert.

Der Einsatz der Modellierungsunterstützung bzw. von Ähnlichkeitsmaßen für methodisch unterschiedlich modellierte Geschäftsprozesse wird im Kapitel 9 vorgestellt. Es wird speziell die Berechnung von semantischen Ähnlichkeiten zwischen EPK-basierten und Petri-Netz-basierten Geschäftsprozessmodellen betrachtet. Damit ist die Modellierungsunterstützung nicht nur auf Petri-Netze beschränkt, sondern kann auf weitere Geschäftsprozessmodellierungssprachen übertragen werden.

Am Ende der Arbeit folgen eine Zusammenfassung der wesentlichen Ergebnisse der Arbeit, die Diskussion der technischen und methodischen Grenzen des vorgeschlagenen Konzeptes und ein Ausblick auf weitere Forschungsmöglichkeiten bezüglich inhaltlicher und technischer Themen.

Abbildung 2 gibt einen Überblick über den Aufbau der Arbeit.

Abbildung 2: Aufbau der Arbeit



2 Grundlagen

In diesem Kapitel werden zunächst Grundbegriffe und Sprachen zur Modellierung von Ontologien und Geschäftsprozessen eingeführt, die im weiteren Verlauf der Arbeit benötigt werden.

2.1 Ontologie

Der Begriff Ontologie stammt ursprünglich aus der Philosophie und wird dort als die Lehre vom Sein verstanden. In der Informatik wird der Begriff seit den 90er Jahren verwendet und stellt die Wissensrepräsentation eines formal definierten Systems von Begriffen und Beziehungen dar.

Ontologie in der Philosophie

Der Begriff Ontologie reicht bis in die Antike zurück, stammt ursprünglich aus der Philosophie und wurde als Teil der Metaphysik (also: über Physik hinaus, nach der Natur kommend) verstanden. In der Philosophie ist der Begriff Ontologie eine Lehre, die sich (*primär) mit dem Sein, dem Seienden als solchem und mit den fundamentalen Typen von Entitäten beschäftigt* [Lehm23]. Mit einer Ontologie werden die Prinzipien des Seins beschrieben. Als Teil der Metaphysik beschreibt eine Ontologie eine sinnlich nicht mehr erfahrbare Welt und die hinter unseren Wahrnehmungen verborgenen (oder vermuteten) Tatbestände – Dinge, die existieren könnten.

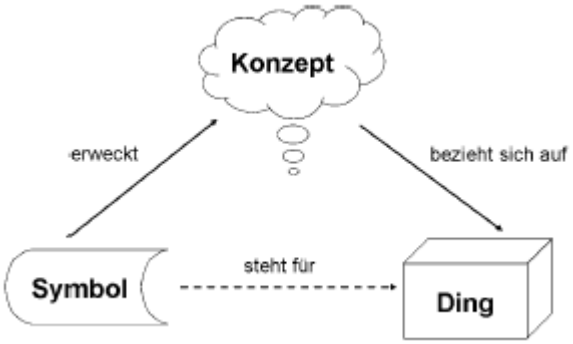
Semiotisches Dreieck in der Linguistik

Ontologien werden auch in der Linguistik verwendet. Unter Linguistik wird hier im weiteren Sinne der Gesamtbereich der Sprachwissenschaften subsumiert [Broc02]. In der Linguistik existieren Modelle, die die menschliche Sprache und somit auch Kommunikation beschreiben. Ein Bereich der Linguistik ist die Semiotik, die sich mit der Theorie der Zeichensysteme beschäftigt. Die Semiotik wird als Oberbegriff für die drei Teilgebiete Syntax, Semantik und Pragmatik verstanden. Die Entstehung der Ontologie in der Semiotik kann durch ein so genanntes semiotisches Dreieck

(Abbildung 3) veranschaulicht werden. Beim alltäglichen und intuitiven Gebrauch unserer Sprache werden die komplexen Zusammenhänge von Wörtern, Symbolen und deren Bedeutungen erst bei Kommunikationsproblemen wie sprachlichen Missverständnissen deutlich.

Die Semiotik behauptet, dass keine direkte Beziehung zwischen einem Wort (Symbol) und einem Gegenstand (Ding) besteht, was im semiotischen Dreieck durch eine gestrichelte Linie veranschaulicht wird. Diese Lehre geht davon aus, dass die Beziehung irgendwann willkürlich festgelegt worden sei, weshalb im semiotischen Dreieck ein Konzept eingeführt wird, das bei einem Wort erweckt wird und sich auf einen Gegenstand bezieht.

Abbildung 3: Das semiotische Dreieck



Beim Sprechen werden automatisch Wörter mit bestimmten Gegenständen verbunden. Die Wörter lösen in unseren Gedanken bestimmte Vorstellungen aus. Mit dem Wort Golf beispielsweise wird eine Sportart im Freien, der letzte Urlaub in einer Golfanlage oder eine Automarke verbunden (siehe Abbildung 4).

Abbildung 4: Begriffsbeispiel

Diese gedankliche Vorstellung verbinden wir auch direkt mit einem realen Bild, z.B. dem Sieger des Golfturniers, der in den gestrigen Sportnachrichten gezeigt wurde. Beim Reden verschmelzen Wörter, Vorstellungen und reale Bilder zu einem Ge-

samtbild und werden als gleichbedeutende Begriffe betrachtet. Damit steht ein Wort für ein bestimmtes Bild bzw. Objekt.

Die moderne Auffassung vom semiotischen Dreieck stammt von *C. K. Ogden* und *I. A. Richards* (1923) und geht davon aus, dass eine implizite Beziehung zwischen Symbolen und Objekten besteht. Symbole werden durch Wörter ausgedrückt, z.B. das geschriebene Wort „Golf" hat an sich noch keine Bedeutung. Erst wir selbst geben den Wörtern eine Bedeutung, die oft unterschiedlich sein kann. Ein Wort gewinnt erst an Bedeutung, wenn es mit einem Objekt, z.B. einem Auto der Marke Golf, das auf der Autobahn im Regen fährt, assoziiert werden. Oder man könnte an die Öltürme in der Region vom persischen Golf denken. All diese Vorstellungen können durch das Symbol „Golf" erweckt werden. Die gedanklichen Vorstellungen beziehen sich auf konkrete Objekte - und somit entsteht eine implizite Beziehung zwischen Symbolen und Objekten. Anderenfalls entstünden viele Missverständnisse in der Kommunikation.

Die menschliche Kommunikation, die zunächst intuitiv und einfach erscheint, ist ein komplexes Geflecht aus Beziehungen zwischen Objekten und Symbolen.

Ontologie in der Informatik

In der Informatik wurden Ontologien erstmals durch T. R. Gruber eingeführt. Gruber versteht unter einer Ontologie "*an explicit specification of a conceptualization*" bzw. "*a specification of a representational vocabulary for a shared domain of discourses – definitions of classes, relations, functions, and other objects*" [Grub93a, Grub93b]. Nach *Gruber* werden Ontologien zur expliziten konzeptuellen Modellierung eines gemeinsamen Wortschatzes für interagierende Parteien verwendet.

Im Bereich der Informatik dient u. a. das Entity-Relationship-Modell (ERM) [Chen76] als konzeptuelle Modellierungssprache für die Abbildung von Objekten (Entitäten) und Beziehungen. Die Semantik dieser konzeptuellen Modellierungssprache soll einerseits eine präzise Beziehung zur Realität sicherstellen und andererseits eine gezielte Manipulation der Daten durch darauf operierende Funktionen ermöglichen.

Im Allgemeinen beschreibt eine Ontologie (formal) Objekte, Objekteigenschaften und Zusammenhänge zwischen Objekten in Form einer hierarchischen Anordnung. Regeln in einer Ontologie werden als Axiome definiert. Beschreibungen von Konzepten werden meistens auf ein bestimmtes Wissensgebiet - auch Domäne genannt - beschränkt. Eine Domäne ist beispielsweise die Medizin, die Astronomie oder auch das Personalwesen eines Unternehmens.

2.1.1 Der Ontologiebegriff

In der Literatur existieren verschiedene Definitionen für den Begriff einer Ontologie. Eine Ontologie kann als ein gerichteter, azyklischer Graph dargestellt werden, wobei die Knoten und Kanten im Graph im Sinne der Graphentheorie verstanden werden[3]. Nachfolgend wird eine formale Definition für den Begriff Ontologie, wie er in dieser Arbeit verstanden wird, formuliert (angelehnt an [SEHH03]).

Definition 1: Ontologie

Eine Ontologie ist ein 5-Tupel $O:=(C, H_C, P_C, H_P, A)$ mit:

- C = Menge aller Konzepte C,
- H_C = Hierarchie, in der die Konzepte angeordnet sind,
- P_C = Menge aller Eigenschaften von Konzepten,
- H_P = Hierarchie, in der die Eigenschaften der Konzepte angeordnet sind,
- A = Menge der Axiome.

Ein Konzept ist ein Gegenstand, ein Symbol, ein Objekt bzw. eine Klasse.

Definition 2: Konzepthierarchie

Die Konzepthierarchie H_C (auch Taxonomie genannt) ist eine Teilmenge von C x C, und definiert Unter- und Oberkonzepte.

$H_C (c_1,c_2)$ mit $c_1,c_2 \in C$ bedeutet, dass c_2 Oberkonzept von c_1 ist.

Definition 3: Eigenschaft

P_C bezeichnet die Menge aller Eigenschaften von Konzepten. Über Eigenschaften können Definitions- und Wertebereiche von Konzepten modelliert werden. Formal werden sie als Untermenge eines kartesischen Produktes von *n* Mengen definiert:

$$P_1 \subset C \times C,\ P_2 \subset C \times C, \ldots,\ P_n \subset C \times C$$

$$P_c \subset P_1, P_2, \ldots, P_n$$

Definition 4: Eigenschaftshierarchie

Eigenschaften von Konzepten können in einer Hierarchie H_P angeordnet werden. H_P ist eine Teilmenge von P_C x P_C.

$H_P\ (p_{c_1}, p_{c_2})$ mit $p_{c_1}, p_{c_2} \in P_c$ bedeutet, dass p_{c_2} Oberbeziehung von p_{c_1} ist.

A ist eine Menge von Axiomen, mit denen implizites Wissen aus explizitem erschlossen werden kann. Mit Axiomen können Beschränkungen, beispielsweise in Form von Kardinalitäten und Regeln für Konzepte und Eigenschaften definiert werden.

[3] In der Graphentheorie gilt G=(E,V), wobei *E* die Menge der Knoten und *V* die Menge der Kanten ist. Nachfolgend werden alle Konzepte und die Instanzen der Konzepte unter E und die Eigenschaften der Konzepte unter V subsumiert.

Definition 5: Wissensbasis

Eine Wissensbasis ist ein 2-Tupel KB:= (O, I) mit:

- O = Ontologie entsprechend Definition 1
- I = Menge der Instanzen von Konzepten

Ein Konzept kann eine oder mehrere Ausprägungen haben, die als Instanzen (ausgedrückt durch *I*) bezeichnet wird.

Gegeben sei die folgende textuelle Beschreibung, aus der anschließend eine Ontologie gemäß Definition 1 abgeleitet wird:

Eine *Publikation* wird durch eine oder mehrere Personen (dem *Autor)*, durch einen *Titel* und durch ein Datum (*Erscheinungsdatum*) beschrieben. Der Autor *verfasst* bzw. *schreibt* eine Publikation. Im Jahr 2006 haben *M. Mustermann* und *M. Muster* ein Buch mit dem Titel „*Ontologien im E-Business*" geschrieben.

Aus den Definitionen 1, 2, 3, 4 und 5 resultiert:

C = {Publikation, Person, Autor, Titel, Datum, Erscheinungsdatum},

H_C = {(Autor,Person); (Erscheinungsdatum, Datum)},

$P_{schreiben}$= {(Autor, Publikation)},

H_P = {(schreiben, verfassen)},

I = {Buch, M. Mustermann, M. Muster, Ontologien im E-Business, 2006},

A = {∃ X (Autor), Y(Publikation): X [schreibt → Y]}

Aufbauend auf diesen Definitionen können Ontologien angelehnt an [NoMc01] wie folgt entworfen werden (diese Vorgehensweise wird im Kapitel 3 zum Entwurf einer Ontologie für Petri-Netze verwendet):

1. Definition des Vokabulars:
   a) Definition von allen existierenden Objekten der Anwendungsdomäne als Konzepte einer Ontologie,
   b) Definition aller Eigenschaften von Konzepten innerhalb der Anwendungsdomäne.

2. Einordnung der Konzepte und Eigenschaften der Anwendungsdomäne in eine Konzept- bzw. Eigenschaftshierarchie anhand taxonomischer Beziehungen.
3. Abbildung aller Regeln durch Einschränkung von Konzept- und Eigenschaftsdefinitionen.
4. Modellierung von Instanzen von Konzepten der Anwendungsdomäne.

Klassifizierung von Ontologien

Die Konzeptualisierung der Ontologie kann auf unterschiedlichen Ebenen stattfinden und somit unterschiedliches Abstraktionsniveau beschreiben; d.h. sie kann mehr oder weniger abstrakt oder detailliert sein. Je nach dem Grad der Allgemeingültigkeit unterscheidet Guarino [Guar98] vier Arten von Ontologien:

- *Top-Level-Ontologie*: beschreiben allgemeine Konzepte wie Zeit, Objekt, Ereignis. Diese Konzepte sind von Problem- und Anwendungsdomänen unabhängig und können somit für die Modellierung verschiedener Anwendungsdomänen verwendet werden. In der Top-Level Ontologie wird das höchste Abstraktionsniveau verwendet. Es wird ein Überblick über den zu modellierenden Ausschnitt der Realwelt gegeben.

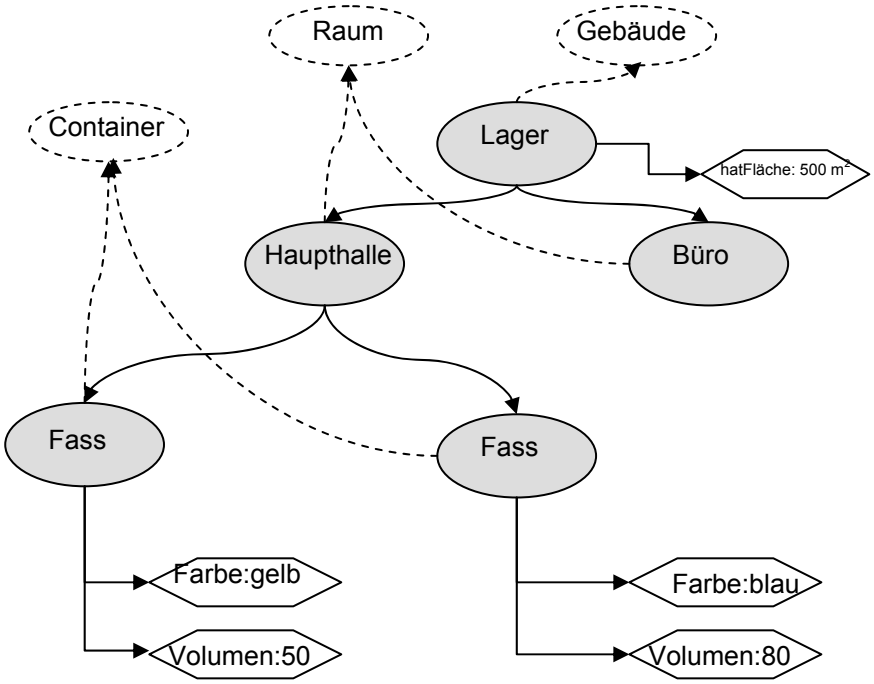
- *Domain-Ontologie:* im Gegensatz zur Top-Level Ontologie werden Domain- Ontologien zur Modellierung eines bestimmten Wissensgebietes wie beispielsweise der Vegetationszeit verwendet. Es wird nur ein begrenzter Ausschnitt der Realwelt beschrieben.

- *Task-Ontologie (auch Problem-Solving-Ontologien genannt):* kann als eine Verfeinerung/Spezialisierung der Top-Level Ontologie für ausgewählte Aufgaben und Lösungswege der Aufgaben (task) verstanden werden. Nach Guarino sollen die Anwendungsdomäne und die Applikation in der Ontologiemodellierung getrennt werden. Eine andere Sichtweise beschreibt Chandrasekaran in [ChJB98]: Eine „Früchteontologie" wird von einem Koch und einem Bauern aufgrund unterschiedlicher Zielsetzungen der Ontologie verschieden modelliert. Der Koch wird in der Ontologie viel detaillierter die Verarbeitung von Früchte beschreiben; während für den Bauern die richtige Auswahl und Verwendung von Pestiziden im Vordergrund stehen. Diese unterschiedliche Zielsetzung einer Ontologie bei gleicher Anwendungsdomäne erschwert eine Isolierung von Aufgaben, die die Ontologie beschreiben soll.

- *Application-Ontologie*: definiert den Wortschatz für eine bestimmte Anwendung (Application) und keine allgemeingültigen Konzepte. Der Fokus der Application-Ontologie liegt auf einem bestimmten Anwendungsgebiet und kann als eine Spezialisierung der Task und Domain Ontologie angesehen werden. Abbildung 5 zeigt ein Beispiel für eine Application-

Ontologie für Lagerräume. Ein Lagerraum hat eine Fläche von 500 m^2 und kann spezialisiert werden in die beiden Konzepte Büro und Haupthalle. Dabei werden Konzepte als Ovale dargestellt und die Eigenschaften der Konzepte als Sechsecke. Die Generalisierung von Konzepten zu Superkonzepten ist durch gestrichelte Ovale dargestellt.

Abbildung 5: Beispiel für eine Application-Ontologie



2.1.2 Sprachen zur Modellierung von Ontologien

Die explizite Modellierung einer Ontologie mit einer formalen Sprache ermöglicht zum einen eine präzise Mensch-zu-Mensch-, Mensch-und-Maschine und Maschine-zu-Maschine-Kommunikation. Des Weiteren unterstützt eine formale Beschreibung den Einsatz von Schlussfolgerungsverfahren, die es ermöglichen, Rückschlüsse aus den modellierten Zusammenhängen einer Ontologie zu ziehen (Inferenz).
Für die Darstellung der Ontologie in einem maschinenlesbaren und maschineninterpretierbaren Format wurden zwei Arten von formalen Sprachen vorgeschlagen: Beschreibungslogik-basierte und Frame-basierte Sprachen. Sprachen wie das Resource Description Framework Schema (RDFS) und die Web Ontology Language (OWL) basieren auf Beschreibungslogiken. Diese Sprachen haben sowohl eine graphische als auch eine XML-basierte textuelle Syntax. Frame[4]-basiere Sprachen bieten zur Ontologiemodellierung die Elemente *Frames* (vergleichbar einem Ontologiekonzept)

[4] Frames sind Varianten von semantischen Netzen. Alle Informationen, die für einen Begriff relevant sind, werden in einer Dateneinheit, genannt Frame gespeichert [Mins75].

und *Slots* (vergleichbar einer Ontologieeigenschaft). Frames bilden eine Klassenhierarchie und Slots können mit zusätzlichen Einschränkungen (Restriktionen) versehen werden.

In diesem Abschnitt werden zunächst Anforderungen an Ontologiesprachen definiert und im Anschluss Frame- und Beschreibungslogik-basierte Modellierungssprachen im Hinblick auf diese Anforderungen überprüft. Die Entscheidung für eine dieser Ontologiesprachen als Grundlage für die Realisierung einer Modellierungsunterstützung soll auf der Validierung dieser Anforderungskriterien basieren. Da alle Beschreibungslogik-basierten Sprachen die XML/RDF-Syntax als Austauschformat verwenden, werden auch XML und Basiskonstrukte von RDF eingeführt. Zudem werden Grundlagen zu XML für das im Kapitel 3 erweiterte Austauschformat für Petri-Netze benötigt.

Anforderungen an Modellierungssprachen für Ontologien:

Ausdrucksmächtigkeit:

a) Ein Konzept kann durch die Vereinigung zweier Konzepte entstehen (Super-/Subkonzept). Bei der Modellierung von Ontologien sollten komplex strukturierte Objekte der Realwelt adäquat beschreibbar sein (z.B. Durchschnitt, Vereinigung).
b) Ein Konzept wird durch Eigenschaften näher beschrieben. Die Eigenschaften müssen durch Einschränkungen näher spezifizierbar sein, um eindeutig den Definitions- und Wertebereich von Eigenschaften ausdrücken zu können (aber auch Einschränkungen wie Transitivität oder inverse Eigenschaften).
c) Eigenschaften von Konzepten können als Wertebereich Objekt- oder Datentypen haben. Die Ontologiemodellierung sollte einfache und komplexe Datentypdefinition unterstützen (XML Schema data types). Der Datentyp *string* beispielsweise definiert, dass ein Konzept eine Zeichenkette ist und damit bestimmte Operationen zulässt (nicht zugelassene Operationen wären die Addition oder die Division von Konzepten).
d) Dieselben Konzepte können in zwei Ontologien unterschiedlich benannt werden. Die unterschiedliche Benennung kann bei der Zusammenführung Schwierigkeiten bereiten. Es sollten Konstrukte vorhanden sein, mit denen die Gleichwertigkeit von Konzepten ausgedrückt werden kann (z.B. Auto = PKW).
e) Ein Konzept kann in einer konkreten Beziehung zu einem anderen Konzept stehen; d.h., es muss möglich sein, Beziehungen zwischen Konzepten in Form von Kardinalitäten angeben zu können (1:1 oder 1:n).

f) Ein maschinenlesbares und -interpretierbares Beschreibungsformat würden den Austausch von Ontologien bzw. Teile einer Ontologie unterstützt. Zudem unterstützt ein solches Format Inferenzen, die eine Konsistenzprüfung auf formaler Ebene und ein Auffinden von begrifflichen Entsprechungen ermöglichen.
g) Bei der Modellierung von Ontologien sollte es möglich sein, Konstrukte einer Ontologie wieder verwenden zu können, um Zeit und Kosten zu sparen[5]. Es soll möglich sein, Konstrukte in passenden Anwendungsdomänen erneut nutzen zu können.
h) Im Fall, dass nur einige Konzepte desselben Typs bestimmte Eigenschaften haben, müssen sich Regeln formulieren lassen, die diese Einschränkung beschreiben.
i) Ontologiekonstrukte müssen über eine eindeutige Kennung verfügen. Eine so genannte URI (Uniform Resource Identifier) [6] garantiert die eindeutige Identifizierung eines Ontologiekonstrukts. Die URI kann mit der ISBN (International Standard Book Number), die jedes Buch eindeutig identifizier, verglichen werden.
j) In einer Ontologie sollten sich nicht nur statische, sondern auch dynamische Zusammenhänge abbilden lassen können. Bei der Modellierung einer Autovermietung müssen auch zeitliche Aspekte berücksichtigt werden. Die Vermietung eines Autos hat einen definierten Anfang und ein definiertes Ende, das mit einem Zeitstempel ausgedrückt wird.
k) Konzepte können verschiedene Ausprägungen haben. Es soll eine Unterscheidung zwischen Konzept und Ausprägung des Konzepts möglich sein.

Formalisierungsgrad:

Automatische Schlussfolgerungen können nur gezogen werden, wenn die Ontologie in einer präzisen und formalen Notation modelliert wird. Um Inferenzen zu ermöglichen, sollte die Ontologie in einer eindeutigen Notation vorliegen.

Visualisierungsmöglichkeiten:

a) Die Modellierung und das Verstehen einer Ontologie sollten verschiedenen Benutzern möglich sein. Benutzer mit geringen Kenntnissen über Beschreibungslogiken sollte auch eine Ontologie modelliert und nachvollziehen können. Deswegen sollten graphische Komponenten die Erstellung einer Ontologie unabhängig

[5] In [BoNo05] wird gezeigt, das die Größe der Ontologie einer der signifikanten Kostenfaktoren bei der Erstellung von Ontologien ist.
[6] Uniform Resource Identifier wurde standardisiert in RFC 3986: http://www.ietf.org/rfc/rfc3986.txt

von Benutzerkenntnissen unterstützen und anschauliche, graphische Visualisierungsmöglichkeiten für das Verstehen von Ontologien bereitgestellt werden.

b) In einer Ontologie können sowohl Informationen, die jedem zugänglich sein sollen, als auch geheime Informationen modelliert werden können. Mit Sichten auf Ontologien kann bestimmten Benutzern Zugriff auf Teilmengen der Ontologie gewährt werden bzw. können Ontologien auf die Bedürfnisse von Benutzergruppen zugeschnitten werden.

Inferenzmöglichkeiten:

Es sollte möglich sein, anhand der Wissensbasis Schlussfolgerungen abzuleiten und eindeutige Interpretationen zuzulassen; z.B. durch Einsatz so genannter Inferenzmechanismen oder Reasoner [HuRy04].

F-Logik

Eine Frame-basierte Sprache zur Modellierung von Ontologien ist die F-Logik. Ein erster Formalismus, um Schlussfolgerungen auf Frame-basierten Daten vorzunehmen, wurde von [KiLW95] vorgeschlagen. Frames bedeutet *Anordnung* oder *Rahmen* und ermöglichen die Darstellung von Attributen und Beziehungen zwischen Konzepten. Mit Frames können beispielsweise Klassenhierarchien grafisch veranschaulicht werden. In F-Logik werden die Vorteile von zwei unterschiedlichen Modellierungsansätzen vereinigt.

Quelltext 1: Modellierung einer Ontologie mit F-Logik

```
// Begriffshierarchie
Autor::Person.
Person::#DEFAULT_ROOT_CONCEPT.
Document::#DEFAULT_ROOT_CONCEPT.

// Beziehungen
Autor["hat-geschrieben"=>>Document].
Person[Alter=>>a#Literal].
Document["hat-Autor"=>>Autor].

// Instanzen
Artikel:Document.
Alex:Autor.
Alex[Alter->>29.0; "hat-geschrieben"->>Artikel].
Peter:Person.
Peter[Alter->>35.0].
// Eigenschaften der Beziehungen
FORALL X,Y X["hat-Autor"->>Y] <-> Y["hat-geschrieben"->>X]
```

Einerseits ist F-Logik objektorientiert und unterstützt alle Modellierungskonstrukte (Klassen, Beziehungen, Klassenhierarchien, Vererbung), die das objektorientierte (OO) Paradigma anbietet. Andererseits besitzt die F-Logik die Ausdrucksmächtigkeit von deklarativen Sprachen[7]. Es können Regeln zwischen den Objekten als logische Formeln definiert werden. Das folgende Beispiel aus [MaMo03] zeigt die F-Logik-Syntax `Person` und `Document` sind Wurzelelemente, wobei `Person` die Unterklasse `Autor` hat. Zwischen den Klassen werden Beziehungen definiert: ein `Autor` hat ein `Document` geschrieben. Eine Instanz von `Autor` ist `Alex` und `Peter`.

Eine Implementierung der F-Logik findet sich im deduktiven Datenbanksystem Ontobroker[8]. In Ontobroker wurde eine Inferenzmaschine integriert, die unter anderem die in F-Logik beschriebenen Ontologien und die darin enthaltenen generischen Regeln lesen und verarbeiten kann. Somit kann implizites Wissen gewonnen werden.

Die F-Logik ist eine objektorientierte Sprache mit einer großen Ausdrucksmächtigkeit, die allerdings nicht entscheidbar ist. Es werden keine eindeutigen Interpretationen zugelassen. Darüber hinaus existieren für F-Logik keine adäquaten Visualisierungsmöglichkeiten als auch keine Serialisierung nach XML (keine Unterstützung von Datentypen). Die Semantik von ausdrucksstarken Logikprogrammen ist kompliziert.

eXtensible Markup Language

Die eXtensible Markup Language (XML) [W3C00] ist eine Teilmenge der umfangreichen Meta-Auszeichnungssprache Standard Generalized Markup Language (SGML). Der Vorteil von XML im Gegensatz zu SGML sind flexible Erweiterungsmöglichkeiten, indem beispielsweise Tags selbst definiert und neue Markup-Sprachen konzipiert werden können. Des Weiteren stellt XML Möglichkeiten bereit, neue Sprachen zu konzipieren[9]. Zur Beschreibung von XML-Inhalten kann entweder die Document Type Definition (DTD) oder XML-Schema[10] verwendet werden. Im Gegensatz zu XML-Schema eignet sich DTD nur eingeschränkt zur Beschreibung von XML-Dokumenten aufgrund mangelnder Typisierung und fehlender Namensräume. Namensräume erlauben die gemeinsame Nutzung verschiedener Schemata/globaler Typen in einem Dokument. XML-Schema kann die DTD vollständig ersetzen und

[7] Deklarative Sprachen basieren auf einer rechnerunabhängigen, mathematischen Theorie. Berechnungen erfolgen als Manipulationen von Werten und die Hauptkontrollstruktur bildet die Rekursion, insbesondere aus Effektivitätsgründen die repetitive Rekursion [Ullm88].
[8] http://www.ontoprise.de/content/
[9] Sprachen wie ebXML [ebXML] oder XHTML [Mint03] basieren auf XML.
[10] http://www.w3.org/2001/XMLSchema

besitzt umfangreichere Strukturbeschreibungen. Darüber hinaus ermöglicht es die Typisierung von Elementinhalten und Attributwerten und bietet verschiedene Modellierungsstile. Quelltext 2 zeigt ein XML Dokument mit dem Vaterknoten `Publikation` und seinen Kindnoten `Titel`, `Autor`, `Herausgeber`, `Ort`, `ISBN`, `Buchtitel`, `Seiten` und `Serie`. Das Element `Autor` wird noch durch seine zwei Kindnoten `Name` und `Vorname` beschrieben.

Quelltext 2: XML-Dokument zur Beschreibung von Büchern

```
<?xml version="1.0" encoding="UTF-8"?>
 <Publikation>
   <Titel>Semantische Erweiterung von XQuery</Titel>
   <Autor>
      <Name>Mustermann</Name>
      <Vorname>Tina</Vorname>
   </Autor>
   <Herausgeber>Springer</Herausgeber>
   <Ort>Berlin, Heidelberg</Ort>
   <ISBN>03053-12568-5</ISBN>
   <Buchtitel>Datenbanksysteme und XML</Buchtitel>
   <Seiten>22 - 37</Seiten>
   <Serie>Lecture Notes in Computer Science</Serie>
 </Publikation>
```

Mit XML kann die Struktur von Dokumenten unabhängig vom Inhalt des Dokuments definiert werden und es eignet sich als Austausch- und Beschreibungsformat für strukturierte und semi-strukturierte Dokumente. Deswegen wird im Kapitel 3 XML ein Austauschformat für Petri-Netze verwendet, damit die Unterstützung während der Geschäftsprozessmodellierung unabhängig vom Petri-Netz-Modell funktioniert.
Zur Modellierung einer Ontologie eignet sich XML aber nicht. Eine präzise Beziehung zwischen Symbolen und Objekten kann nicht dargestellt werden. Die Unzulänglichkeit von XML resultiert aus den mangelnden Vorgaben zur Beschreibung der Bedeutung von Elementen. Die nachfolgenden zwei Beispiele im Quelltext 3[11] sollen diese Unzulänglichkeit verdeutlichen. Sowohl das rechte als auch das linke XML-Dokument beschreiben Webdokumente mit einem Autor. Diese Informationen sind für den Menschen verständlich, weil wir aus den Namen der Tags die Bedeutung des Objekts ableiten können.

Quelltext 3: XML-Dokument mit gleicher semantischer Bedeutung

```
<document>                    ODER    <document href="page">
 <details>                             <author>Ora</author>
  <uri>href="page"</uri>              </document>
  <author>
```

[11] http://www.w3.org/DesignIssues/RDF-XML.html

```
    <name>Ora</name>
  </author>
 </details>
</document>
```

Ein Parser könnte überprüfen, ob es sich um syntaktisch korrekte Elemente handelt, aber keine Informationen über den Inhalt der Dokumente „schlussfolgern". Mit XML beschriebene Dokumente sind damit maschinenlesbar, aber nicht maschineninterpretierbar. Deutlicher wird die Unzulänglichkeit von XML-Dokumenten in Hinsicht auf die Modellierung von Ontologien, wenn keine Informationen aus der Bedeutung der Tags abgeleitet werden können.
Quelltext 4 zeigt ein für ein XML-Schema gültiges XML-Dokument[12]. Bekannt sei nur die Dokumentenstruktur. Der Betrachter kann nicht sagen, was der Elementinhalt `a="ppppp"` bedeutet, weil das umschließende Element `<y>` ohne weitere Informationen keine implizite Bedeutung hat.

Quelltext 4: XML-Dokument ohne implizite Bedeutung

```
<x>
 <y> a="ppppp"</y>
 <z>
  <w>qqqqq</w>
 </z>
</x>
```

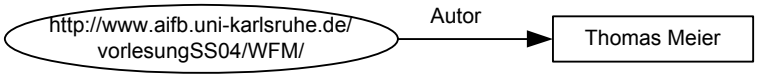
Den Inhalt von <y/> könnten wir erst erschließen, wenn Metadaten vorliegen würden. Die nachfolgenden Sprachen liefern Metadaten zu Elementen und unterstützen eine automatische Verarbeitung und Interpretation von Dokumenten durch Maschinen.

Resource Description Language

RDF ist ein formal fundiertes graphisches Modell [W3C04a], bestehend aus zwei Arten von Knoten – Ellipsen und Rechtecken – die über Kanten miteinander verbunden sind. Ein solches Tripel stellt in RDF eine Aussage dar [W3C04b]. Ellipsen repräsentieren Ressourcen und werden durch ein URI gekennzeichnet. Kanten beschreiben Eigenschaften von Ressourcen und bekommen durch Eigenschaftswerte (Rechtecke) eine spezifische Bedeutung. Abbildung 6 zeigt eine einfache RDF-Aussage – bestehend aus dem Tripel *#WFM*, *Autor* und *Thomas Meier*. Auf die Ressource *WFM* wird durch die URI *http://www.aifb.uni-karlsruhe.de/vorlesungenSS06/WFM/* verwiesen, *Autor* ist die Eigenschaft und *Thomas Meier* der Eigenschaftswert der zu beschreibenden Ressource.

[12] Ein XML-Dokument ist gültig für ein XML-Schema, wenn es den Definitionen in einem zugehörigen XML-Schema entspricht.

Abbildung 6: Einfache RDF-Aussage



Aussagen, Eigenschaften und Werte können ihrerseits wieder Ressourcen sein und somit untereinander kombiniert werden. RDF weist den Daten damit Bedeutung zu. RDF-Aussagen können äquivalent in einer RDF/XML-Syntax ausgedrückt werden [W3C04c]. Diese erlaubt maschinenlesbare und eindeutig verständliche Aussagen über Ressourcen. Im Quelltext 5 ist die RDF/XML-Syntax der RDF-Aussage aus Abbildung 6 veranschaulicht.

Quelltext 5: RDF/XML-Syntax zur RDF-Aussage in Abbildung 6

```
<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#>
  <rdf:Description rdf:about="http://..../vorlesungSS04/WFM/">
      <Autor>Thomas Meier</Autor>
  </rdf:Description>
</rdf:RDF>
```

Allerdings gibt RDF kein vordefiniertes Metadatenvokabular vor, sondern ermöglicht die Integration verschiedener Standards und die Definition unterschiedlicher Schemata, mit denen die Gültigkeit von RDF-Aussagen überprüft werden kann. Während ein XML-Schema nur eine Vorgabe für die syntaktische Struktur von wohlgeformten XML-Dokumenten macht, gehen RDF-Schemata (RDF-S) einen Schritt weiter: mit ihnen wird eine semantische Zuordnung von Begriffsbeziehungen erreicht, und den beschriebenen Eigenschaften wird eine semantische Ordnung gegeben, was eine automatische Weiterverarbeitung ermöglicht [W3C04d].
Für die Syntax der Web Ontology Language (OWL), wie sie im Anschluss an RDF-S erklärt wird, verwendet einige RDF-S-Elemente, die nachfolgend erklärt werden [Hjel01]:

- *rdfs:Resource* Ressourcen sind der allgemeinste Begriff für alle Elemente, die mit RDF-S beschrieben werden. Sämtliche modellierten Elemente sind Instanzen dieser Klasse.
- *rdfs:Class* Ressourcen können (wie im objektorientierten Sinne) in Klassen gruppiert werden. Der Typ einer Ressource wird über *rdf:type* definiert.

Im Quelltext 6 sind zwei verschiedene Arten der Definition von Klassen beschrieben. Der Name der Klasse kann entweder in einem `Description`-Element und einem `rdf:type`-Element geschachtelt oder nur als `rdfs:Class` deklariert werden.

Quelltext 6: Verschiedene Schachtelungen von rdfs:Class

```
<rdf:Description
  rdf:about="http://www.example.com/PersonSchema#Person">
   <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
                                    oder

<rdfs:Class rdf:ID="http://www.example.com/PersonSchema#Person"/>
```

- *rdfs:Literal* Literale sind Ressourcen wie Ganzzahlen oder Zeichenketten. Sie können einfach oder typisiert (rdfs:datatype) sein.
- *rdf:Property* Eigenschaften von Ressourcen und Beziehungen zwischen Ressourcen können durch Eigenschaften (properties) dargestellt werden. Eigenschaften sind nach der RDF-S-Syntax auch Ressourcen, werden aber nicht als Subjekt sondern als Prädikat in RDF-Aussagen ausgedrückt.

RDF-S definiert auch mehrere Eigenschaften, um hierarchische Beziehungen zwischen den Ressourcen zu beschreiben.

- *rdfs:type* Mit type kann die Zugehörigkeit einer Ressource zu einer Klasse festgelegt werden.
- *rdfs:subClassOf* Die Vererbung von Eigenschaften kann, wie bei objektorientierter Programmierung, mit Unterklassen erreicht *werden.* Jede Instanz der Unterklasse ist automatisch eine Instanz der Oberklasse.
- *rdfs:domain* und *rdfs:range* Mit diesen Konstrukten können Werte- und Definitionsbereiche für Eigenschaften formuliert werden.

Die Beschreibungslogik-basierten Sprachen sind eigenschaftsorientiert - d.h. Eigenschaften werden Ressourcen zugeordnet und können sogar ohne Klassenzugehörigkeit definiert werden. Die Syntax basiert weiterhin auf XML - somit ist die plattformunabhängige Verarbeitung von Dokumenten gewährleistet.

Die wichtigsten Einschränkungen von RDF-S im Hinblick auf die Modellierung von Ontologien sind:

- *keine lokale Definition von Definitions- und Wertebereichen (domain- und range) von Eigenschaften,*
- *nur unzureichende Unterstützung von Kardinalitäten,*
- *keine Definition von Axiomen wie Transitivität, Inverse oder Symmetrie,*

- *keine Unterstützung von Äquivalenz von Klassen,*
- keine Verwendung von booleschen Ausdrücken, um neue Konzepte aus vorhandenen Konzepten zu bilden.

Web Ontology Language

Die Web Ontology Language (OWL) [McHa03] geht über die Ausdrucksmächtigkeit von RDF-S hinaus und erweitert die Semantik des Modells dahin gehend, dass eine möglichst automatische Interpretation der Aussagen möglich wird. Die wichtigsten Eigenschaften, die OWL von den bisher beschriebenen Sprachen unterscheidet, sind:

- zwischen Konzepten können Mengenbeziehungen ausgedrückt werden,
- es wird ein umfangreiches Typsystem für Eigenschaften unterstützt,
- es können Merkmale und Einschränkungen von Eigenschaften definiert werden (Transitivität, Inverse, Symmetrie).

OWL wurde in drei Versionen entworfen, die sich in der Ausdrucksmächtigkeit unterscheiden: OWL Lite, OWL DL (Description Logic) und OWL Full. Die Wahl der OWL-Version ist vom benötigten Grad der Ausdrucksmächtigkeit (Komplexität) zur Modellierung der Ontologie abhängig. Diese Anforderungen ändern sich je nach Zielen, die eine zu implementierende Anwendung erfüllen soll.

- **OWL Lite:** verfügt im Gegensatz zu OWL DL und OWL Full über wenige formale Konstrukte und unterstützt somit hauptsächlich Benutzer, die einfache Klassifikationshierarchien ohne Konzepteinschränkungen modellieren wollen. Mit den OWL-Lite-Konstrukten sollen einfach und schnell Werkzeuge und Anwendungen implementiert werden können.

  Für die Deklaration von Konzepten wird in OWL ein eigenes Konstrukt verwendet: `<owl:Class>`. RDF-S bietet auch ein Konzeptkonstrukt `<rdfs:Class>` an, aber in der Deklaration eines RDF-S-Konzeptes können auch Metakonzepte (Konzepte, deren Instanzen wiederum Konzepte sind) eingeschlossen sein.

  Deklarierte Konzepte können in einer Konzepthierarchie organisiert werden und werden mit dem Element `<rdfs:subClassOf>` umgesetzt. Neben der Modellierung von Konzepten stehen dem Benutzer auch Konstrukte zur Modellierung von Eigenschaften zur Verfügung. OWL basiert auf dem Modell von RDF-S, weshalb Eigenschaften als selbständige Elemente modelliert und erst dann mit bestimmten Konzepten in Beziehung

gestellt werden. Eine Beziehung kann zwischen zwei Instanzen oder einer Instanz und einem Wert bestehen. Im ersten Fall handelt es sich um eine Eigenschaft mit dem Namen `<owl:ObjectProperty>`. Eine *ObjectProperty* hat als Wertebereich bestimmte Konzepte, die mit `<rdfs:range>` ausgedrückt werden kann. Der zweiten Fall (Beziehung zwischen Instanz und Wert) wird durch die Eigenschaft `<owl:DatatypeProperty>` ausgedrückt, die als Wertebereich keine Konzepte, sondern einen XML-Schema-Datentyp oder ein RDF Literal hat. Wie in RDF-S hat jede Eigenschaft neben dem Wertebereich auch einen Definitionsbereich. Mit `<rdf:domain>` wird der Definitionsbereich von Klassen über Eigenschaften deklariert.

Das wichtigste Ziel bei der Konzeption von OWL war eine verbesserte Unterstützung von automatischen Schlussfolgerungen. In OWL Lite ist es möglich, eine ObjectProperty als transitiv (`<owl:transitiveProperty>`) oder symmetrisch (`<owl:symmetricProperty>`) zu definieren. Es können auch inverse Eigenschaften `<owl:inverseOf>` definiert werden. Im Quelltext 7 wurde eine *ObjectProperty* hasChild mit einem Definitions- und Wertbereich Person und Child, die invers zur Eigenschaft hasParent ist, modelliert.

Quelltext 7: Modellierung der `inverseOf`-Einschränkung und ein für das Schema gültiges Instanzdokument

```
<owl:ObjectProperty rdf:ID="hasChild">
  <owl:inverseOf rdf:resource="#hasParent"/>
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="#Child"/>
</owl:ObjectProperty>

   Instanz:

<Person rdf:ID="Thomas">
 <hasChild rdf:ID="Jenny"/>
</Person >
```

Aufbauend auf dieser Modellierung von Konzepten mit Eigenschaften und deren Einschränkungen kann mit Hilfe von Inferenzmechanismen gefolgert werden, dass Thomas ein Elternteil von Jenny ist.

Ähnlich wie bei RDF-S, können die Eigenschaften mit `<rdfs:subPropertyOf>` in Hierarchien organisiert werden. Zusätzlich kann eine Beziehung das Prädikat `<owl:FunctionalProperty>` haben. Es besagt, dass diese Eigenschaft höchstens einen oder gar keinen Wert

annehmen kann. Es ist also eine Abkürzung für die Kardinalitäten min=0 und max=1. Eine noch speziellere Eigenschaft ist `<inverseFunctionalProperty>`.

Quelltext 8: Modellierung der `inverseFunctionalProperty`-Einschränkung

```
<owl:ObjectProperty rdf:ID="hasMaker" />
<owl:ObjectProperty rdf:ID="producesWine">
  <rdf:type rdf:resource="InverseFunctionalProperty"/>
   <owl:inverseOf rdf:resource="#hasMaker"/>
</owl:ObjectProperty>
```

Aus Quelltext 8 und dem Wissen, dass *WineABC* die Eigenschaft *hasMaker* mit dem Wertebereich *MakerXYZ* gilt und *Maker123* die Eigenschaft *producesWine* mit dem Wertebereich *WineABC* gilt (und WineABC keine weiteren Maker hat), kann über ein automatisches Schließen herausgefunden werden, dass *MakerXYZ* die gleiche Instanz wie *Maker123* ist.

Die Spezifikation von Kardinalitäten ist in OWL Lite eingeschränkt. Es können lediglich Minimum- und Maximum-Kardinalitäten von 0 bis 1 gesetzt werden. Das bedeutet, dass nur einfache Beziehungen definiert werden können (1:0, 1:1 oder 0:1).
Konzepte und Instanzen, die gleiche Dinge beschreiben, können mit so genannten Äquivalenzmechanismen zusammengeführt werden. Gleichwertige Konzepte können durch die Deklaration des Konstrukts `<owl:equivalentClass>` verschmolzen werden. Zwei Konzepte aus verschiedenen Ontologien, z.B. Auto und PKW, könnten hiermit als äquivalent gesetzt werden, womit alle Instanzen des Konzepts Auto gleichzeitig auch Instanzen des Konzepts PKW sind und alle Instanzen des Konzepts PKW auch Instanzen von Auto sind. Synonyme von Eigenschaften können analog mit `<owl:equivalentProperty>` gesetzt werden.
Allerdings werden diese beiden Konstrukte (owl:equivalentClass und owl:equivalentProperty) selten in der Ontologiemodellierung verwendet, weshalb in Kapitel 4 Verfahren zur Auffindung von Synonymen in Ontologien vorgestellt werden.
Zwei Ressourcen, die verschiedene Namen haben, können auf dieselbe Instanz verweisen. Dieser Zusammenhang wird mit dem Konstrukt `<owl:sameAs>` ausgedrückt.
Die Namenskonventionen erlauben es nicht, dass zwei Instanzen als unterschiedlich beschrieben werden, indem sie unterschiedlich benannt

werden. Wenn man die Eigenschaft *hatAugenfarb*e als funktional definiert und dann mit den zwei Werten *blau* und *grün* belegen würde, so würde über das automatische Schließen gefolgert werden, dass blau und grün die gleiche Farbe sind. Wenn dagegen die Farben als `<owl:differentFrom>` definiert werden, dann wird über das automatische Schließen dieser Widerspruch (blau ≠ grün) erkannt werden. Bei mehreren Instanzen, die verschieden voneinander sind, kann `<owl:Alldifferent>` verwendet werden.

Ein weiterer Vorteil von OWL im Gegensatz zu RDF-S ist die Möglichkeit der Definition von lokalen Beschränkungen von Eigenschaften. Bei RDF-S gelten definierte Einschränkungen von Eigenschaften global, d.h. Einschränkungen (domain und range) gelten für alle Klassen, die mit der beschränkenden Eigenschaft verbunden sind. Weiterhin können unterschiedliche Einschränkungen für verschiedene Klassen nicht definiert werden. Die Eigenschaft *hatHersteller* mit dem Wertebereich *Produzent* könnte z.B. nicht für Weine auf Weingut verfeinert, also lokal eingeschränkt werden. In OWL Lite (im Gegensatz zu OWL DL) können Einschränkungen aber nur begrenzt angewendet werden. Als einzige Einschränkung kann eine Schnittmenge aus Instanzen von einer nicht anonymen Klasse und einer Eigenschaftsbeschränkung gebildet werden.
Die Syntax von `<owl:intersectionOf>` wird wie im Quelltext 9 veranschaulicht, modelliert:

Quelltext 9: Modellierung der intersectionOf-Einschränkung

```
<owl:Class rdf:ID="SportlicheStudenten">
 <owl:intersectionOf rdf:parseType="Collection">
  <owl:Class rdf:about="#Student"/>
   <owl:Restriction>
    <owl:onProperty rdf:resource="#hasHobby"/>
     <owl:someValuesFrom>
      <owl:Class rdf:about="#Sport"/>
     </owl:someValuesFrom>
   </owl:Restriction>
  </owl:Class>
 </owl:intersectionOf>
</owl:Class>
```

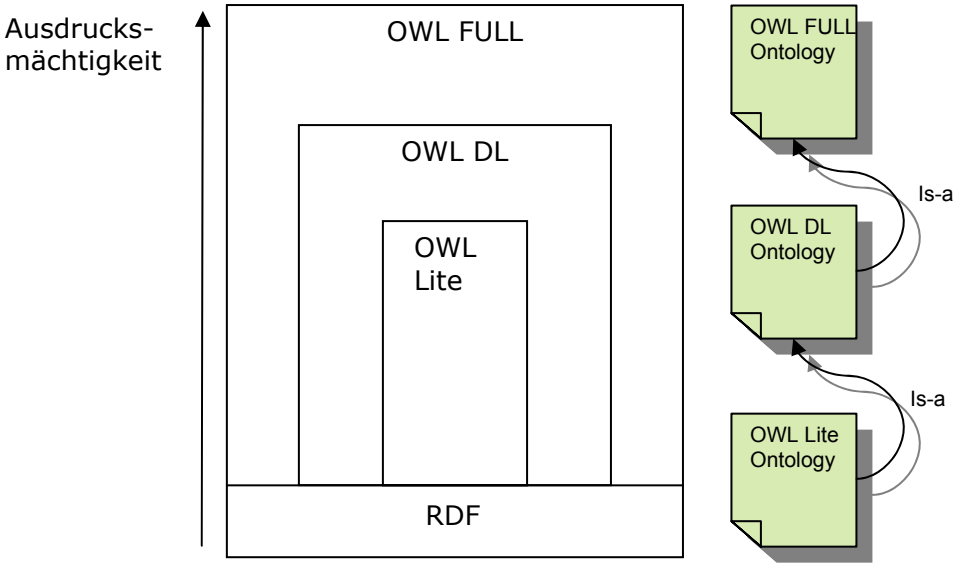
Etwa 20 neue Konstrukte schränken syntaktisch die RDF-S-Syntax ein. Damit ist jedes OWL Lite Dokument auch ein gültiges RDF-S-Dokument, aber nicht umgekehrt.

- **OWL DL:** Die Sprachkonstrukte von OWL Lite sind, wie gerade beschrieben, nur für einfache Modellierung von Ontologien ausreichend. Die höhere Ausdrucksmächtigkeit von OWL als Ontologiemodellierungssprache wird bei OWL DL deutlicher. OWL DLhat zwei wichtige Eigenschaften: hohe Ausdrucksmächtigkeit und Entscheidbarkeit. Mit automatischem Schließen kann impliziertes Wissen in endlicher Zeit berechnet und gewonnen werden. Automatisches Schließen oder Reasoning, die Arten von Inferenzmechanismen sind, unterstützen Konsistenzprüfungen von Wissensbasen, Strukturierung von Ontologien durch Berechnung der Unterklassen und Äquivalenz- und Disjunktion-Bestimmungen zwischen Konzepten [HuRy04]. Durch das automatische Schließen auf Instanzebene kann die Zugehörigkeit der Instanzen zu Konzepten inferiert werden. Die Ausdruckmächtigkeit von OWL DL drückt sich vor allem in der unterschiedlichen Art und Weise aus, wie Konzepte deklariert werden können; durch eine Aufzählung der Instanzen oder als beliebige boolesche Verknüpfungen von Konzepten und Eigenschaften mit `<owl:unionOf>`, `<owl:complementOf>` oder `<owl:intersectionOf>`. Zusätzlich kann mit dem Element `<owl:disjointWith>` eingeschränkt werden, dass zwei Konzepte keine gemeinsamen Instanzen haben dürfen. Die Semantik von Beziehungen zwischen Klassen kann bei OWL DL exakt spezifiziert werden. Die Kardinalitäten von Eigenschaften können beliebig definiert werden und sind nicht mehr auf 0 und 1 beschränkt.

- **OWL Full** ist die komplexeste, ausdrucksmächtigste, allerdings auch eine nicht entscheidbare, Variante von OWL. OWL- und RDF-S-Sprachelemente können beliebig kombiniert werden. OWL Full stützt sich auf die gleichen Sprachkonstrukte wie OWL DL, wobei die Anwendbarkeit der Konstrukte mehr Freiraum bietet. Ähnlich wie bei RDF-S wird in OWL Full nicht zwischen Instanzen und Klassen unterschieden. Instanzen können selbst wieder Klassen mit Eigenschaften sein, was Schwierigkeiten bei Inferenzen bereitet.

Ein Zusammenhang zwischen RDF und den drei OWL-Varianten ist in Abbildung 7 veranschaulicht (angelehnt an [Lacy05]). OWL Full ist die ausdrucksmächtigste Sprache, sie ist allerdings nicht entscheidbar. In OWL DL gibt es mehr Einschränkungen von Elementen, dafür ist sie entscheidbar.

Abbildung 7: Zusammenhang zwischen Beschreibungslogik-basierten Sprachen



2.1 Geschäftsprozesse

Im vorhergehenden Kapitel wurden der Ontologie-Begriff und Sprachen zur Modellierung von Ontologien detailliert beschrieben. Zur Umsetzung einer Unterstützung während der Geschäftsprozessmodellierung werden in diesem Kapitel der Zweck von und Sprachen zur Modellierung von Geschäftsprozessen erklärt. Zusätzlich werden noch Analysemethoden für Geschäftsprozesse eingeführt.

Nach [HaCh93] ist ein Geschäftsprozess *„a collection of activities that takes one or more kinds of input and creates an output that is of value for the customer*". In [Dave93] wird ein Geschäftsprozess definiert als *„a structured, measured set of activities designed to produce a specified output for a particular customer or market. It implies a strong emphasis on how work is done within an organisation, in contrast to a product focus´s emphasis on what."*

In dieser Arbeit wird auf die Definition von Geschäftsprozessen nach A. Oberweis [Ober96] zurückgegriffen. Er versteht unter einem Geschäftsprozess eine Menge von manuellen, teil-automatisierten oder automatisierten betrieblichen Aktivitäten, die nach bestimmten Regeln, auf ein bestimmtes Ziel hin ausgeführt werden.

Ein zusammenhängender, rechnergestützter Teil eines Geschäftsprozesses wird Workflow genannt. Der Unterschied zwischen Geschäftsprozessen und Workflows liegt in der Automatisierbarkeit der Prozesse. Die Struktur realer Prozesse wird durch so genannte Prozessmodelle beschrieben. Dazu müssen alle Aktivitäten, die auszuführen sind, erkannt werden. Außerdem müssen alle Pfade entlang des Pro-

zesses und alle Regeln, die für die Wahl der Pfade entscheidend sind, z.B. mit Hilfe von Interviews mit Wissensträgern aufgenommen werden. Prozessmodelle von Geschäftsprozessen werden im Unternehmen Geschäftsprozessmodelle genannt. Prozessmodelle von Workflows werden Workflow-Modelle genannt. Das Prozess-Management ermöglicht, dass der Arbeitsfluss so organisiert wird, dass die zu leistende Arbeit zur richtigen Zeit und von der richtigen Ressource ausgeführt wird [AaDO00].
Wichtige Instrumente, damit Geschäftsprozesse dem technischen und wirtschaftlichen Änderungen gerecht und Umwelteinflüsse des Unternehmens adaptiert werden können, sind die Modellierung, Analyse und kontinuierliche Verbesserung von Prozessen.

2.2.1 Zweck der Geschäftsprozessmodellierung

Bevor Systementwickler oder Programmierer lauffähige Programme erzeugen oder Systemanalytiker Anpassungen oder Verbesserungen der Prozesse vornehmen können, benötigen sie als Grundlage ihrer Arbeit ein Prozessmodell, welches die relevanten Arbeitsschritte, die dafür verwendeten Materialien und die verantwortlichen Personen beschreibt. Prozessmodelle liefern Transparenz über alle ablaufenden Prozesse, benötigten Prozessobjekte und beschreiben verwendete Ressourcen. Ein Prozessmodell wird von einem Modellierer erstellt. Dieser muss sich zunächst detaillierte Informationen über die zu modellierenden Geschäftsprozesse besorgen. Die Informationsbeschaffung kann durch Interviews mit Wissensträgern, Analyse von Unterlagen und eigene Beobachtung gewonnen werden [StHa02]. Wichtig ist, dass bei der Wissensgewinnung folgende Fragen beantwortet werden:

- Was ist relevant für die Modellbildung (z.B. welche Ressourcen, Objekte)?
- Welche Konzepte und welche Beziehungen existieren?
- Wie fein muss das resultierende Prozess-Modell sein?

Diese Fragen müssen in Kontext der Geschäftsprozessmodellierung beantwortet werden, d.h. Ziele, die mit der Modellierung der Geschäftsprozesse verfolgt werden, und der Ausschnitt des Prozessmodells aus der Realwelt tragen entscheidend zur Modellgestaltung bei. Verschiedene Kontexte liefern unterschiedliche Modelle; das bedeutet, dass ein Prozessmodell niemals eindeutig ist und derselbe Geschäftsprozess unterschiedlich modelliert werden kann.

Im Folgenden werden mögliche Ziele einer Prozess-Modellierung nach [Ober96] aufgelistet:

- Zur Erleichterung der Kommunikation zwischen verschiedenen Personen
- Zum Zweck der Dokumentation
- Zum Zweck der Analyse für eine nachfolgende Verbesserung und Reorganisation
- Zu Entwurfszwecken
- Zur Planung des Ressourcen-Einsatzes
- Als Grundlage für die Unterstützung durch ein Workflow-Management-System bei der Ablaufplanung
- Als Grundlage der Überwachung und Steuerung von Abläufen

2.2.2 Sprachen zur Modellierung von Geschäftsprozessen

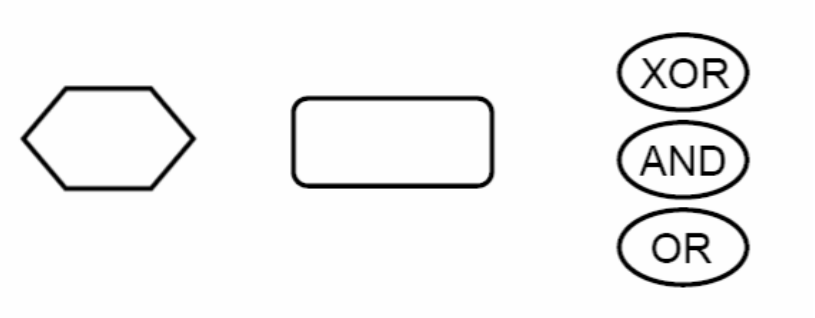
Zur Modellierung von Prozessen können verschiedene Sprachen eingesetzt werden, unter anderen Ereignisgesteuerte Prozessketten [KeNS92] oder (höhere) Petri-Netze [Aals98], [Ober96]. Vor der Wahl einer konkreten Modellierungssprache muss sich der Modellierer über die Anforderungen im Klaren sein, die die gewählte Methode erfüllen soll.
Zur Orchestrierung[13] und Choreographie[14] von Geschäftsprozessen wurden die Business Process Execution Language [AABC05] bzw. die XML Process Definition Language [WMC05] als so genannte ausführbare Sprachen entwickelt. Zusammenhänge zwischen den Modellierungssprachen werden ebenfalls in diesem Kapitel angerissen.

Ereignisgesteuerte Prozessketten

Ereignisgesteuerte Prozessketten (EPK) [KeNS92] verfügen aufgrund ihrer graphischen Darstellung über eine hohe Anschaulichkeit, lassen aber nur eine statische Sicht auf Prozess-Strukturen zu. Die Hauptkonstrukte einer EPK sind Ereignisse, Funktionen und drei Verknüpfungsoperatoren (XOR, AND, OR), wie in Abbildung 8 veranschaulicht. Neben Knoten haben EPKs Kanten, die Ereignisse mit Funktionen und Funktionen mit Ereignissen verbinden. Ereignisse sind passive Elemente, die Auslöser für Funktionen sind. Funktionen sind aktive Elemente, die etwas durchführen.

[13] Gibt Regeln an, nach denen Web-Services zu einem komplexen Dienst aggregiert werden können.
[14] Beschreibt Regeln, nach denen Web-Services interagieren können.

Abbildung 8: Konstrukte einer EPK



Verzweigungen und Synchronisationen von Ereignissen/Funktionen werden nicht implizit in EPKs modelliert, sondern über Verknüpfungsoperatoren ausgedrückt. Eine Verzweigung einer Funktion in mehrere Ereignisse muss mit einem der drei Verknüpfungsoperatoren modelliert werden (siehe Abbildung 8).
Eine Funktion hat die Entscheidungskompetenz über den weiteren Ablauf; Ereignisse haben keine solche Entscheidungskompetenz. Deswegen ist es nicht erlaubt, dass ein vorwärtsverzweigtes Ereignis mit zwei Funktionen über XOR- und OR-Verknüpfungsoperatoren verbunden ist. In erweiterten EPKs (eEPK) ist es möglich, Funktionen mit Organisationseinheiten zu verbinden und ein- und ausgehende Datenflüsse zu modellieren. Damit können Datenflüsse und Organisationseinheiten über mehrere Prozessschritte verfolgt werden.
EPKs können den Anforderungen der Simulation von Geschäftsprozessen ohne syntaktisch-semantische Erweiterungen nicht gerecht werden. Die Syntax von EPKs ist zwar semi-formal, es existieren allerdings einige Ansätze in der Literatur, die die Syntax von EPKs formalisieren [AaDK02]. Insgesamt besitzen EPKs nur hinreichende Regeln für die Modellausführung.

Business Process Execution Language

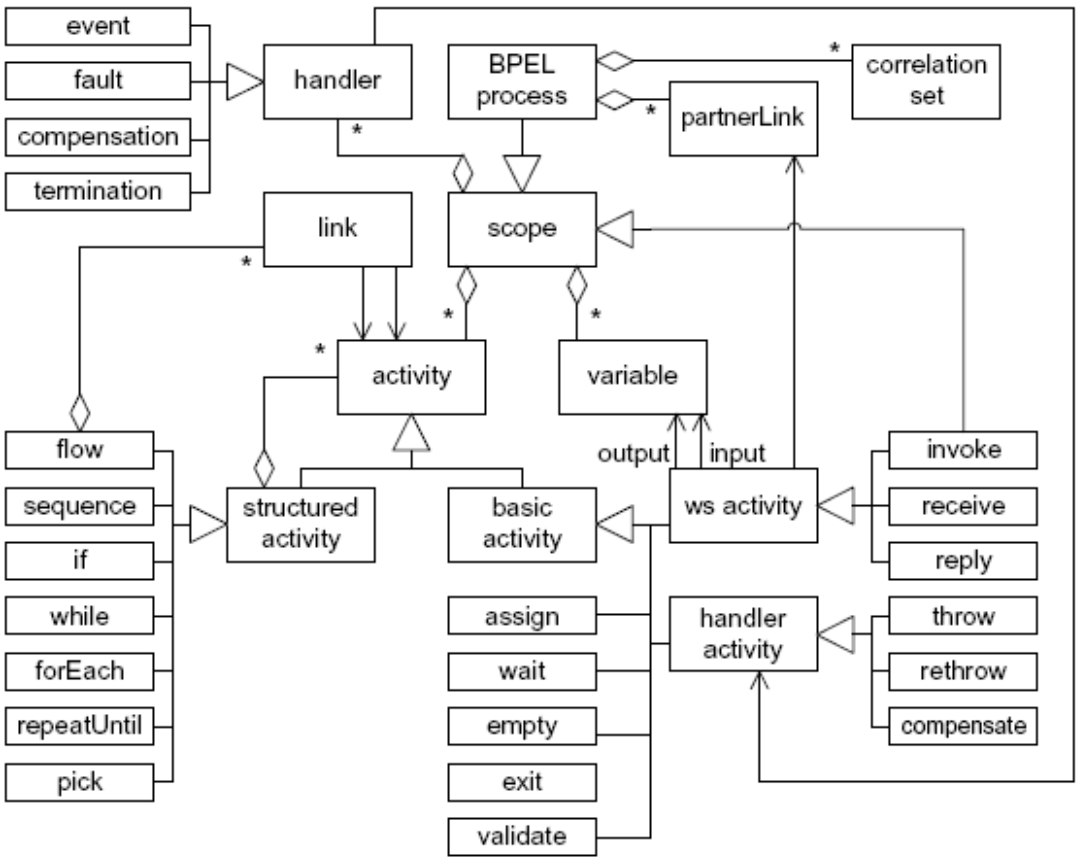
Die Business Process Execution Language (BPEL) ist eine textuelle Sprache, die es ermöglicht, komplexe Geschäftsprozesse, die als Web-Services[15] definiert werden, zu orchestrieren[16]. Ende des Jahres 2005 wurde BPEL in der zweiten Version von

[15] Zu Definitionen für Web-Services siehe [UDDI01] oder [W3C02]. Nach [W3C02] ist ein Web-Service a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artefacts. A Web Service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols.

[16] In der BPEL-Spezifikation wird zwischen „executable" und „abstract" BPEL-Prozessen unterschieden. Im Folgenden werden nur ausführbare Prozesse, die die Prozess-Orchestrierung unterstützen, vorgestellt. Abstrakte BPEL-Prozesse werden zur Modellierung von öffentlichen Eigenschaften von Geschäftsprotokollen verwendet.

der Standardisierungsorganisation OASIS[17] vorgelegt. Die Hauptkonzepte von BPEL sind *basic* und *structured activities*, *variables*, *partner links*, und *handler*s. In Abbildung 9 sind die Konzepte von BPEL in einem UML-Klassendiagramm dargestellt [HKKR05]. Ein minimaler BPEL-Prozess definiert *partner links*, *variables* und *activities*. Mit *partner links*, auf die über *basic activities* verwiesen wird, wird der Nachrichtenaustausch zwischen zwei Parteien repräsentiert. Über den *partner link* kann eine Referenz zu einem *partner link type* definiert werden, in dem gegenseitig notwendige Endpunkte des Nachrichtenaustausches festgelegt werden: die beiden Attribute *myRole* und *partnerRole* geben jeweils die Rollen der Partner an. Variablen werden verwendet, um sowohl Prozess-Daten als auch Eingangs- und Ausgangsnachrichten, die über Web-Service-Aktivitäten mittels *partner links* ausgetauscht werden, zu speichern. Über die basic activities *assign*, *throw* und *rethrow* können Variablenwerte verändert werden.

Abbildung 9: Metamodel von BPEL



[17] http://www.oasis-open.org/home/index.php

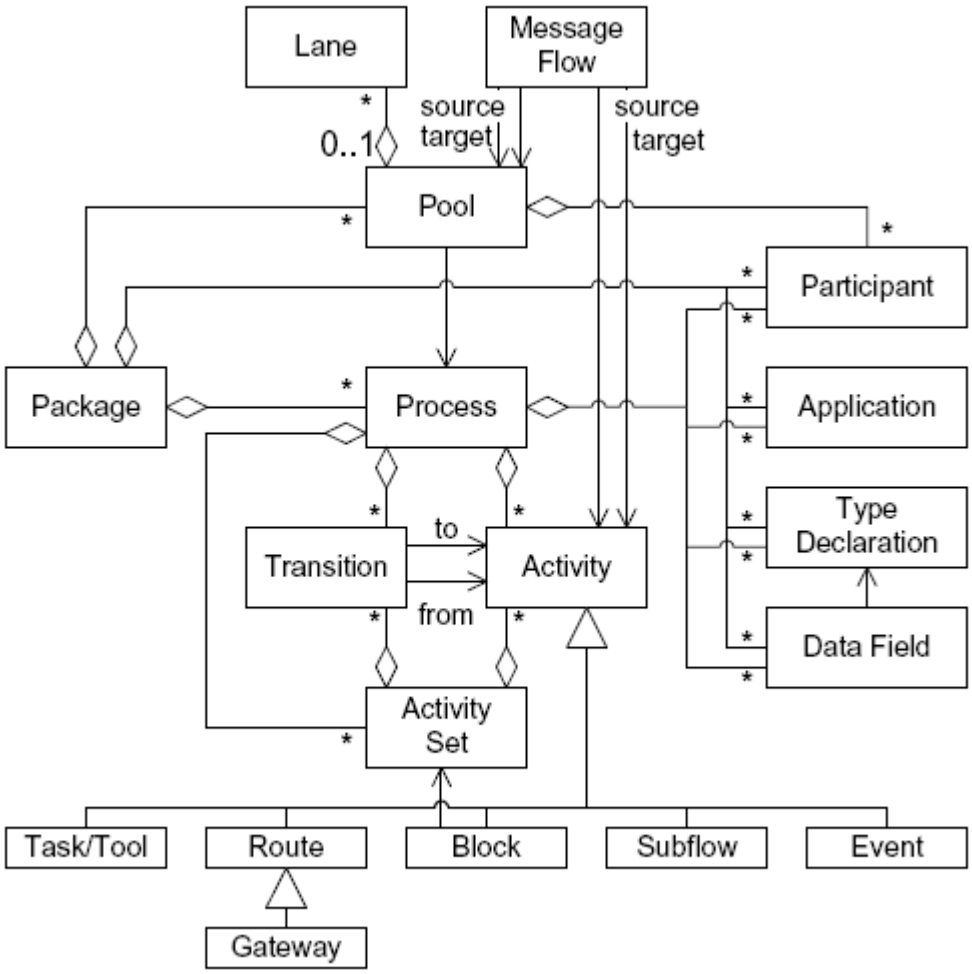
BPEL ist eine blockorientierte Sprache[18] und erlaubt es, bei der Definition von lokalen Umgebungen (*Scopes*) lokale Variablen einzufügen. Mit den Scopes können außerdem Fehlerbehandlungen (*Fault Handler*), Kompensationsbehandlungen (*Compensation Handler*) und Ereignisbehandlungen (*Event Handler*) beschrieben werden. Durch die Schachtelung von *structured activities* wird der Kontrollfluss in BPEL ausgedrückt. BPEL bietet eine Serialisierung nach XML an. Schleifen in Prozessen können durch die Aktivitäten *while*, *forEach* und *repeatUntil* ausgedrückt werden, eine sequentielle Abarbeitung durch *sequence*, eine parallele durch *flow* und Bedingungen durch *if*.
Zusätzliche Synchronisationseinschränkungen werden durch *links* definiert. Basic activities sind atomare Aktivitäten, die nicht aus anderen Aktivitäten zusammengesetzt sind. Die *invoke*-Aktivität wird zum synchronen oder asynchronen Aufruf eines Web-Service verwendet. Mit *receive* und *request* werden Schnittstellen zum Empfangen (receive) von und zum Antworten (request) auf Nachrichten definiert. Informationen von Variablen können mit *assign*-Aktivitäten verändert werden. Damit Aktivitäten auf bestimmten Input warten, kann die *wait*-Aktivität eingefügt werden. *Empty* und *exit* sind Aktivitäten zum „Nichtstun" bzw. zur Beendigung von Prozessen. Einfache Aktivitäten haben als Input und Output Nachrichten-Variablen und verweisen auf *partner links*.

XML Process Definition Language

Ursprünglich wurde die XML Process Definition Language (XPDL) als ein XML-basiertes Austauschformat für Interfaces des Workflow Reference Models vorgeschlagen. XPDL sollte ein minimales Meta-Model beschreiben, das gemeinsam verwendete Konstrukte in einer Prozessdefinition identifizieren sollte [WMC02]. Ende 2005 wurde eine zweite Version von XPDL veröffentlicht. Die Intention dieser XPDL-Version ist vorrangig, ein XML-basiertes Austauschformat für die Business Process Modeling Notation [Whit04] anzubieten. Die Umsetzung von XPDL 2.0 erforderte neue Konzepte, die aus BPMN übernommen wurden, wie *pools*, *gateways* oder *events*. Die Hauptkonzepte von XPDL sind in der Abbildung 10 in einem UML-Klassendiagramm veranschaulicht.

[18] Die Ausführung von Anweisungen erfolgt blockweise. Dabei können einzelne Blöcke beliebig ineinander verschachteln werden.

Abbildung 10: Metamodell von XPDL



Ein package ist das abstrakteste Konzept und beinhaltet alle Informationen, die mit einer Prozessdefinition (inklusive *pools*, *processes*, *participants*, *applications*, *type*, *declaration* und *data fields*) verbunden sind. Ein Prozess (bzw. Workflow) definiert die auszuführenden Aktivitäten und ihre Reihenfolge. Seit der zweiten Version können in XPDL auch *partner links* (vergleichbar mit *partner links* in BPEL) definiert werden. Der Kontrollfluss ergibt sich in XPDL durch Transitionskanten (*transition arcs*) zwischen Aktivitäten. Dabei können Transitionen Bedingungen zugeordnet werden, die das Aktivieren von Transitionen einschränken. Operationen von Aktivitäten können *participants*, *applications* und *data field* definieren. In XPDL werden verschiedene Typen von Aktivitäten unterschieden. Die *task/tool*-Aktivität beschreibt eine Aktivität, die automatisch ohne menschliche Interaktion ausgeführt werden kann. Join und Split- Bedingungen werden durch *route*-Aktivitäten spezifiziert. Diese Eigenschaften von *route*-Aktivitäten unterstützen die Formulierung

komplexer Bedingungen des Kontrollflusses. Über die *block activity* wird ein eingebetteter Unterprozess ausgedrückt, der durch Aktivitäten (*activity set*) ausgelöst wird. Aus der BPMN-Spezifikation wurden spezielle Typen von Aktivitäten, so genannte *events*, übernommen. Im Gegensatz zur BPEL unterstützt XPDL statische Informationen von Aktivitäten mittels zeitlicher Beschränkungen (*deadline* und *limit*) und Eigenschaftsattribute, die für Simulationsengines nützlich sind. Des Weiteren können in BPEL noch keine Unterprozesse definiert werden. Bislang wurde eine Erweiterung von BPEL um Unterprozesse nur in einem ersten Entwurf vorgeschlagen [KMLP05]. XPDL soll ausdrücklich interoperabel zu Applikationen wie Enterprise Java Beans (EJBs)[19] oder XSLT[20] sein. BPEL unterstützt nur die Modellierung und Ausführung von Web-Services.

Petri-Netze

Petri-Netze [Reis86, Pete77] kombinieren Vorteile der graphischen Darstellung von Geschäftsprozessen mit einer formalen Semantik des beschreibenden Systemverhaltens. Aufgrund ihrer umfangreichen mathematischen Fundierung können Petri-Netz-basierte Geschäftsprozesse modelliert, analysiert und mittels einer Workflowengine ausgeführt werden. Nach [Ober96] erfüllen Petri-Netze sämtliche Anforderungen an eine Modellierungssprache für Geschäftsprozesse: formale Syntax und Semantik, graphische Darstellung, hohe Ausdrucksmächtigkeit, Werkzeugunterstützung, Herstellerunabhängigkeit sowie explizite Darstellung von Zuständen und Ereignissen.

Petri-Netze sind eine anschauliche und einfach zu erlernende Modellierungsmethode für diskrete dynamische Systeme, mit der sich sowohl strukturelle als auch dynamische Eigenschaften abbilden lassen. Die allgemeine Definition eines Netzes [Petr62, BrRR87] gilt für alle Petri-Netz-Typen.

Definition 6: Netz

Ein Tripel N= (S, T, F) wird als Netz bezeichnet, falls gilt:

(i) S, T sind endliche Mengen

(ii) $S \cap T = \varnothing$

(iii) $S \cup T \neq \varnothing$

(iv) $F \subseteq (S \times T) \cup (T \times S)$

[19] http://java.sun.com/products/ejb/
[20] http://www.w3.org/TR/xslt

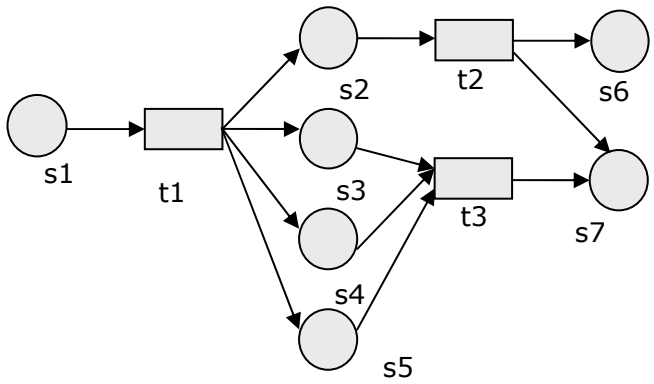
Die Menge S beschreibt Stellen[21]. Transitionen[22] sind die Elemente der Menge T. Die Flussrelation F des Netzes N wird durch gerichtete Kanten zwischen den Stellen und Transitionen gebildet.

Als Petri-Netz wird ein gerichteter, bipartiter Graph mit zwei disjunkten Knotenmengen und einer Menge von Kanten zwischen diesen Knoten bezeichnet. Die disjunkten Knotenmengen sind Stellen (dargestellt als Kreise) und Transitionen (dargestellt als Vierecke). Stellen sind die passiven und Transitionen die aktiven Komponenten eines Ablaufs. Abbildung 11 zeigt die graphische Darstellung eines Netzes N=(S,T,F). Es besteht aus folgenden Mengen:

S={s1,s2,s3,s4,s5,s6,s7}, T={t1,t2,t3}

F={(s1,t1),(t1,s2),(t1,s3),(t1,s4),(t1,s5),(s2,t2),(s3,t3),
(s4,t3),(s5,t3),(t2,s6),(t2,s7),(t3,s7)}

Abbildung 11: Ein Petri-Netz



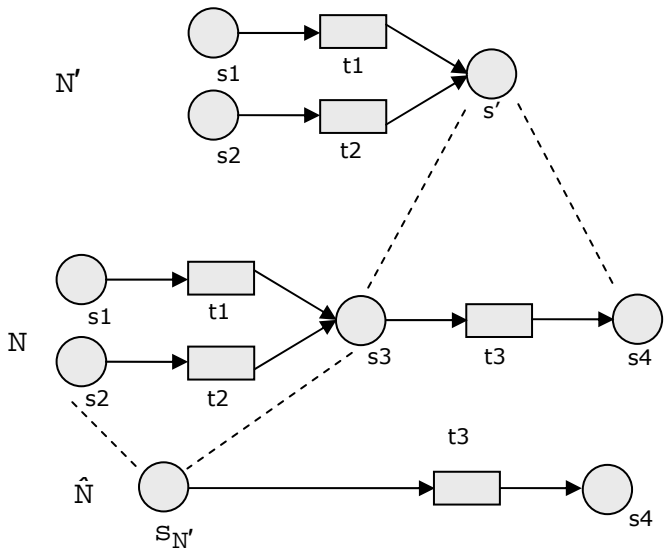
Für unterschiedliche Anwendungsgebiete wurden verschiedene Petri-Netz-Typen vorgeschlagen. Zu den elementaren Petri-Netzen zählen unter anderem Bedingungs/Ereignis-Netze und Stellen/Transitionen-Netze [Reis85]. Als höhere Petri-Netze wurden unter anderem Prädikate/Transitionen-Netze, XML-Netze oder gefärbte Petri-Netze vorgeschlagen. Zur Beschreibung eines Workflows werden Aufgaben und Bedingungen, die Aufgaben anstoßen durch ein spezielles Petri-Netz-Modell, das Workflow-Netz nach [Aals98], beschrieben.

Geschäftsprozessmodelle haben unterschiedliche Abstraktionsgrade. Derselbe Geschäftsprozess kann mit unterschiedlichem Detaillierungsgrad modelliert werden. Es ist deshalb notwendig, bei der Modellierungsunterstützung nur solche Teilstücke von Geschäftsprozessen vorzuschlagen, die den gleichen Abstraktionsgrad wie das editierte Geschäftsprozessmodell haben. In Petri-Netzen können Unterprozesse durch Verfeinerungen/Vergröberungen von Transitionen bzw. Stellen zu einem

[21] Bedingungen für die Aktionen oder Handlungen.
[22] Aufgaben, Aktionen oder Handlungen.

Subnetz/Supernetz modelliert werden. Die Verfeinerung von Prozessen reduziert die Komplexität und erleichtert eine Wiederverwendung von Geschäftsprozessmodellen. In Abbildung 12 wird die Stelle s' im Netz N durch die Elemente s3, t3 und s4 näher beschrieben. Im untersten Netz werden die Elemente s1, s2, t1, t2 und s3 durch $s_{N'}$ zusammengefasst. Die Umkehrung einer Vergröberung ist eine Verfeinerung.

Abbildung 12: Beispiel einer Vergröberung von N zu $\hat{N}$ und N'.

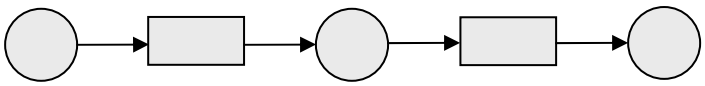


Im Rahmen der (semi-) automatischen Unterstützung für Benutzer während der Geschäftsprozessmodellierung wird auf Ablaufmuster zurückgegriffen. In Kapitel 6 werden Ablaufmuster zur Bestimmung des Abstraktionsgrades von Begriffen benötigt und zur Bestimmung von korrekten Folgeprozessen, weshalb sie nachfolgend vorgestellt werden.

Ablaufmuster nach [Aals98]:

Abbildung 13 zeigt eine Sequenz, d.h. die Aufgaben werden seriell in der aufgezeigten Reihenfolge ausgeführt.

Abbildung 13: Sequenzieller Ablauf



Die Parallelität eines Ablaufes ist in Abbildung 14 dargestellt. Nach einer Und-Teilung (AND-Split) laufen die Prozesse parallel ab. Die Aufgaben können demnach ohne gegenseitige Beeinflussung parallel, nacheinander oder teilweise nacheinander bearbeitet werden. Parallele Prozesse können durch Synchronisation (Und-Zusammenführung bzw. AND-Join) vereinigt werden, was bedeutet, dass zwei oder mehr Stellen als Eingangsstellen einer Transition zusammengeführt werden.

Abbildung 14: Paralleler Ablauf

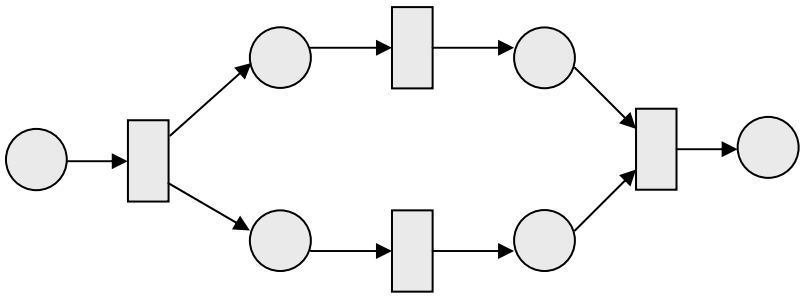
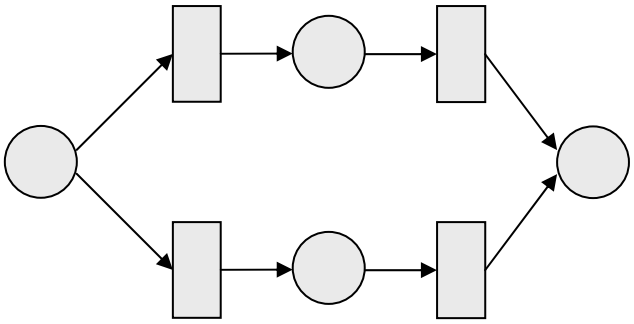


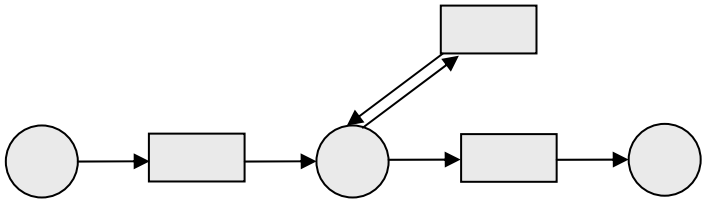
Abbildung 15 zeigt die Modellierung einer Konfliktsituation. Eine Oder-Teilung (OR-Split) ermöglicht die Wahl zwischen mindestens zwei Transitionen. Nach dem alternativen Ablauf können zwei oder mehr Transitionen wieder zusammengeführt (OR-Join) werden.

Abbildung 15: Konfliktsituation



Die Wiederholung eines Ablaufs ist in Abbildung 16 dargestellt.

Abbildung 16: Iteration

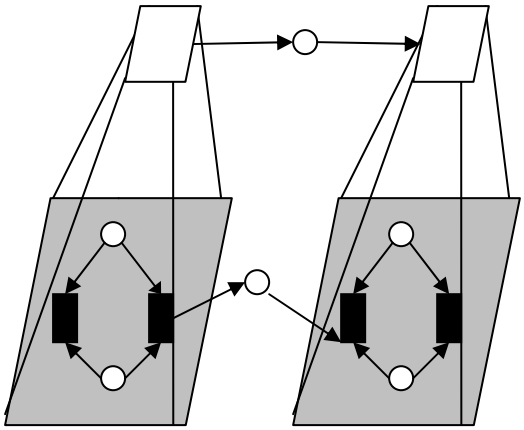


Im Gegensatz zu elementaren Petri-Netzen können in höheren Petri-Netzen Marken Eigenschaften zugeordnet werden. Komplexe Modelle können kompakter dargestellt werden, und somit wird der Einsatz effizienter Algorithmen möglich [Baum96]. Farbige Petri-Netze bieten beispielsweise durch die Erweiterung um Farbe die Möglichkeit, beliebige unterscheidbare Individuentupel zu erzeugen.

Hierarchische Petri-Netze

Hierarchische Petri-Netze [Fehl92] sind keine Erweiterung des entsprechenden Netz-Modells, sie erlauben nur eine Darstellung eines unterschiedlichen Abstraktionsgrades und die Strukturierung komplexer Netze. Um Petri-Netze hierarchisch strukturieren zu können, wird ein (Unter-)Prozess als ein neues Element eingeführt. Abbildung 17 zeigt ein Petri-Netz mit zwei Verfeinerungen von Transitionen.

Abbildung 17: Hierarchisches Petri-Netz mit verfeinerten Transitionen



Prädikate/Transitionen-Netze

Prädikate/Transitionen-Netze (Pr/T-Netze) [GeLa81, Genr86] ermöglichen durch ihre hohe Ausdrucksmöglichkeit die Modellierung komplexer Vorgänge. Ein Prädikat ist eine Eigenschaft, die für ein Objekt gilt oder nicht. Im Gegensatz zu elementaren Petri-Netzen, deren Marken nicht unterscheidbare Objekte sind, haben Pr/T-Netze individuelle Marken. Diese Marken haben genauso viele Komponenten, wie die Stellenzahl des Prädikates, auf dem sie liegen, beträgt.
Ein Pr/T-Netz wird wie folgt formal definiert:

Definition 7: Prädikate/Transitionen-Netz

Ein striktes[23] Prädikate/Transitionen-Netz ist ein Tupel $N = (S, T, F, \Psi, KB, TI, M^0)$, für welches gilt:

(i) (S, T, F) ist ein Netz. S wird als Menge von Prädikaten mit veränderlichen Ausprägungen interpretiert.

(ii) $\Psi = (D, FT, PR)$ ist eine Struktur bestehend aus einer Individuenmenge D, einer auf D definierten Menge von Funktionen FT und einer Menge PR von auf D definierten Prädikaten mit unveränderlichen Ausprägungen.

(iii) Die Kantenbeschriftung KB weist jeder Kante aus F eine Menge von Variablentupeln mit der Stelligkeit des adjazenten Prädikats zu.

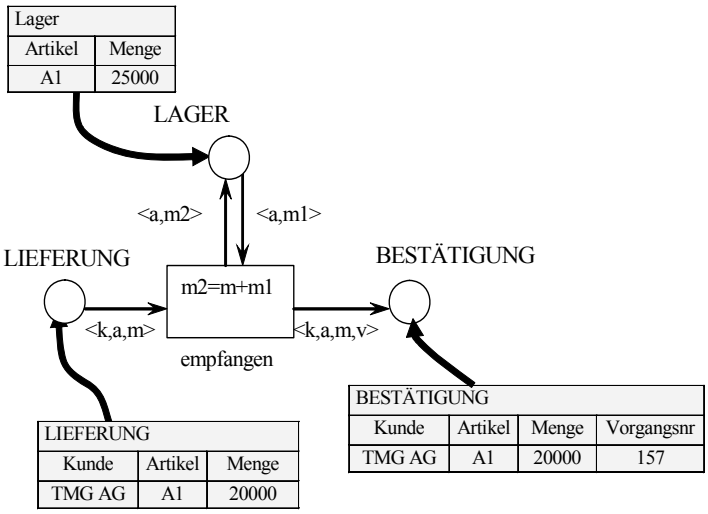
(iv) TI weist jeder Transition aus T eine Transitionsinschrift in Form eines über Ψ und der Menge der an den adjazenten Kanten vorkommenden Variablen gebildeten prädikatenlogischen Ausdrucks zu.

(v) M markiert die Prädikate mit Mengen von konstanten Individuentupeln mit der Stelligkeit des entsprechenden Prädikats. M0 ist die Startmarkierung.

In Abbildung 18 ist ein Ausschnitt eines Geschäftsprozesses zur Bestätigung von Lieferungen mit Pr/T-Netzen modelliert. Im Prädikat Lager befindet sich ein Tupel <A1, 25000>, das die Funktionen Artikel und Menge beschreibt. Die Kante von *Lage*r nach *empfangen* hat die Stelligkeit des Prädikats *Lager*. Mit dem Aktivieren der Transition *empfangen* wird eine neue Menge *m2* bestehend aus der Menge *m* und der Menge *m1* berechnet.

[23] Ein Pr/T-Netz heißt strikt, falls Prädikate nicht mit zwei identischen Individuentupeln markiert werden dürfen. Bei nicht strikten Pr/T-Netzen kann die Markierung den Prädikaten Multimengen zuweisen. Unter Multimengen versteht man eine Zusammenfassung von Elementen mit einer Vielfachheit größer oder gleich eins [Genr86]. Im weiteren Verlauf werden nur strikte Pr/T-Netze betrachtet, sofern nicht explizit auf die Verwendung nicht-strikter Pr/T-Netze hingewiesen wird.

Abbildung 18: Modellierung eines Geschäftsprozesses mit einem Pr/T-Netz



In strikten Pr/T-Netzen repräsentieren Prädikate ($s \in S$) Relationstypen mit n paarweise verschiedenen Attributen ($A_s^1,....,A_s^n$) und jeweiliger Domäne D_s^i für jedes A_s^i mit $i \in \{1,...,n\}$. Die Markierungen der Stellen sind als Relationen des entsprechenden Typs gegeben. Eine Relation R_s wird als eine Menge von Tupeln verstanden ($R_s \subseteq D_s^1 \times .... \times D_s^n$). Transitionen stellen Operationen auf den Eingangs- bzw. Ausgangsrelationen dar. Die Transitionsinschrift einer Transition $t \in T$ eines Pr/T-Netzes setzt sich aus Schaltbedingungen und Schaltoperationen zusammen, die in einer prädikatenlogischen Formel vereinigt werden.

Definition 8: Term

Sei *X* eine Menge von Variablen und *FT* eine Menge von Funktionen definiert auf X^n. Die Menge *T* der Terme über *X* wird folgendermaßen gebildet:

(i) Jede Variable $x \in X$ ist ein Term.

(ii) Seien $f \in FT$ und $t_1,..., t_n$ Terme, dann ist $f(t_1,..., t_n)$ ein Term.

Definition 9: Prädikatenlogischer Ausdruck

Für die Menge P_Ψ der prädikatenlogischen Ausdrücke zu einer Struktur $\Psi = (D, FT, PR)$ gilt:

(i) $wahr \in P_\Psi$.

(ii) $P \in PR$ n-stelliges Prädikat, $t_1,..., t_n$ Terme $\Rightarrow P(t_1,..., t_n) \in P_\Psi$.

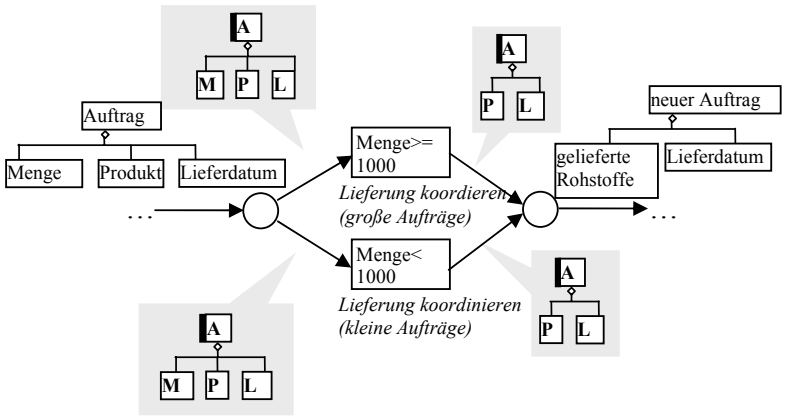
(iii) $p_1, p_2 \in P_\Psi \Rightarrow \begin{cases} p_1 \wedge p_2 \in P_\Psi \\ p_1 \vee p_2 \in P_\Psi \\ \neg p_1 \in P_\Psi \\ p_1 \rightarrow p_2 \in P_\Psi \end{cases}$.

XML-Netze

Wie bereits in Kapitel 2.2 beschrieben, wird XML als ein Dokumentenaustauschformat zwischen mehreren Parteien und zur Beschreibung von Dokumenten in einem für Menschen und Maschinen lesbaren Format verwendet. Komplexe hierarchische Strukturen von XML-Objekten und deren Manipulationen in Geschäftsprozessen können durch XML-Netze [Lenz03] modelliert werden. In XML-Netzen werden Marken als XML-Dokumente verstanden. Stellen werden als Behälter für Dokumente interpretiert, die bezüglich eines XML-Schematas gültig sind. Über Kanten werden Lese- und Manipulationsoperationen von ganzen Dokumenten und Dokumententeilen vorgenommen. In XML-Netzen können ebenfalls Nebenläufigkeiten von Operationen auf demselben Dokument modelliert werden, allerdings nur, solange disjunkte Dokumententeile bearbeitet werden. Der Fluss von XML-Dokumenten wird durch das Schalten von Transitionen definiert. Den adjazenten Kanten werden Filterschemata zugeordnet, die relevante Dokumente für die Transition beschreiben.

In Abbildung 19 ist ein Ausschnitt eines XML-Netzes zur Koordinierung von Aufträgen modelliert. Ein Auftrag wird durch die Elemente Menge, Produkt und Lieferdatum beschrieben. Die Filterschemata an den beiden Kanten von Auftrag bis Lieferung koordinieren (große Aufträge und kleine Aufträge) haben die gleiche Struktur wie das XML-Dokument Auftrag.

Abbildung 19: Modellierung eines Geschäftsprozesses mit XML-Netzen



Wie bereits erwähnt, stellen hierarchische Petri-Netze keine Erweiterung des entsprechenden Netz-Modells dar. Sie erlauben nur eine Darstellung eines unterschiedlichen Abstraktionsgrades und die Strukturierung komplexer Netze. Somit kann die Hierarchisierung von Prozessen in allen Petri-Netz-Varianten verwendet werden. In Lenz [Lenz03] wird beschrieben, dass jedes Pr/T-Netz als XML-Netz angesehen werden kann, da jedes Relationenschema in ein XML-Schema übertragen werden kann.
Die im weiteren Verlauf der Arbeit vorgestellten Methoden für eine Unterstützung während der Geschäftsprozessmodellierung lassen sich damit mit wenigen Änderungsoperationen auf alle Varianten von Petri-Netzen anwenden.

Geschäftsprozessmodellanalyse
Hauptziele der Geschäftsprozessanalyse sind die Validierung, Verifikation und Leistungsbewertung des Prozessmodells. Bei der Validierung wird überprüft, ob das Modell die Wirklichkeit richtig abbildet – durch Prüfung der Struktur und des Verhaltens der einzelnen Prozesse. Die Korrektheit des Prozessmodells (Verklemmungsfreiheit, Lebendigkeit) wird bei der Verifikation geprüft. Bei der Leistungsbewertung erfolgt die Auswertung der Leistungsfähigkeit des Prozesses, indem einzelne Parameter, wie beispielsweise die Durchlaufzeit, die Kosten oder die Ressourcenauslastung überprüft werden.
Im Folgenden werden einige Analysemethoden für Petri-Netze wie die Strukturanalyse, die Verhaltensanalyse, die Erreichbarkeitsanalyse, die Linear-algebraische Analyse und die Simulation erläutert.

Bei der **Strukturanalyse** wird u.a. überprüft, ob das zu analysierende Netz zusammenhängend oder stark zusammenhängend ist, ob das Netz ein Free-Choice-Netz [Reis86] darstellt und ob es die Eigenschaften eines Workflow-Netzes [Aals98] besitzt. Nicht zusammenhängende Petri-Netze können beispielsweise nicht getrennt analysiert werden.

Bei der **Verhaltensanalyse** wird das Schaltverhalten eines Netzes auf Grundlage der modellierten Marken und Markierungen überprüft. Folgende dynamische Eigenschaften sollten bei der Modellierung eines Prozesses berücksichtigt werden [Baum96]:

- Sicherheit: Der Prozess soll in endlicher Zeit ausführbar sein, d.h. beschränkt sein.

- Lebendigkeit: Der Prozessverlauf soll lebendig sein. Es muss ein verklemmungsfreier Ablauf gewährleistet werden.

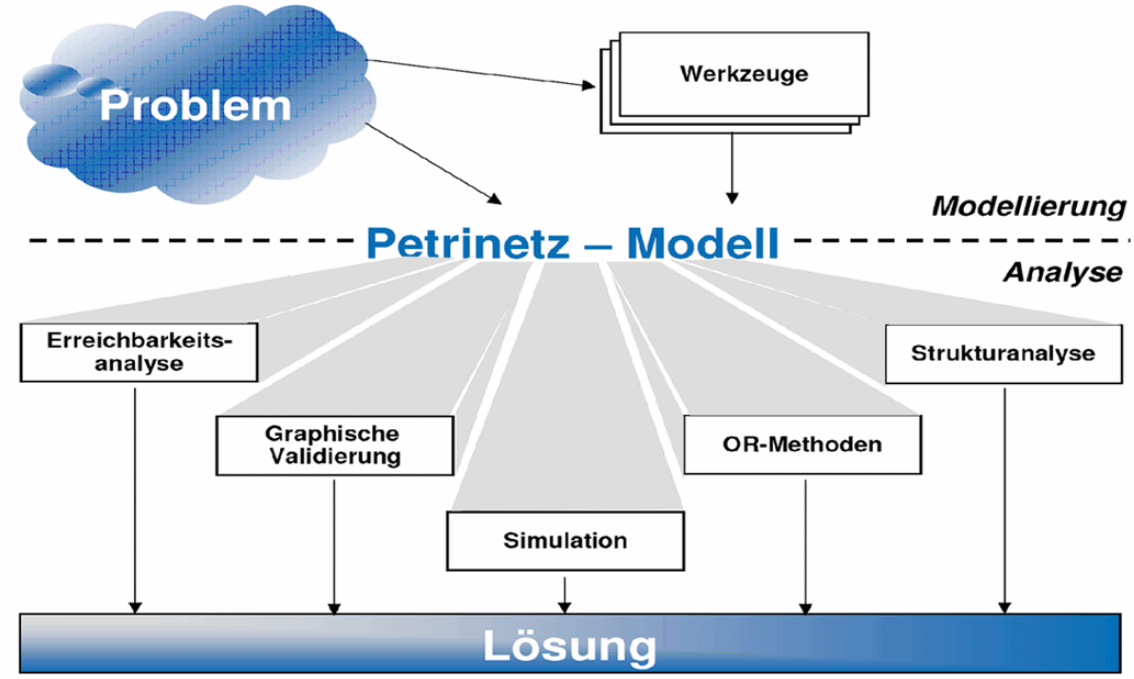
Mit der **Erreichbarkeitsanalyse** sollen alle erreichbaren Zustände eines Systems untersucht werden. Es wird untersucht, ob erwartete oder unzulässige Zustände (z.B. Deadlocks) eintreten können. Zur Überprüfung der Zustände wird ein Markierungs- und/ oder Überdeckungsgraph aufgestellt.

Bei der **Linear-algebraischen Analyse** werden Matrixrepräsentationen von Petri-Netzen aufgestellt, um Aussagen über das Verhalten eines markierten Netzmodells gewinnen zu können. Stellen bzw. Transitionen werden in der Matrixrepräsentation eines Petri-Netzes durch Zeilen bzw. Spalten beschrieben.

Geschäftsprozesse können auch durch **Simulation** analysiert werden. Durch die Simulation können Prozesse validiert [Aals98], Leistungsbewertungen vorgenommen [MBCD95] und der Modellierungsprozess von Abläufen unterstützt werden, indem die Korrektheit und die Vollständigkeit der Ablauf- und Objektbeschreibungen überprüft werden [Ober96].

In Abbildung 20 sind die beschriebenen Analysemethoden für Petri-Netze veranschaulicht (Abbildung ist aus [RiSt04] entnommen).

Abbildung 20: Analysemethoden für Geschäftsprozesse, insbesondere für Petri-Netze



2.2.3 Vergleich von Sprachen zur Modellierung von Geschäftsprozessen

In vorhergehendem Unterkapitel wurden BPEL, XPDL, EPKs und Petri-Netze beschrieben. BPEL und XPDL adressieren eine andere Ebene der Geschäftsprozessmodellierung als EPKs und Petri-Netze. Mit Modellierungssprachen wie EPK oder Petri-Netzen werden Aspekte der Realwelt implementationsunabhängig modelliert. Wobei mit Petri-Netzen die Komposition der einzelnen Web-Services modelliert werden kann. Sprachen wie BPEL oder XPDL sind XML-basierte Sprachen zur Beschreibung von Geschäftsprozessen. Dabei werden die einzelnen Aktivitäten als Web Services implementiert. Somit beschreiben BPEL und XPDL technische Informationen, die die Ausführung von Prozessen als Web-Service unterstützen.
In [KoMe05] findet sich ein Einsatzbeispiel zur Modellierung von Aktivitäten als Web Services mit XML-Netzen und die Serialisierung in die XML-Syntax von BPEL. Ein Ansatz zur Integration der Schemata von BPEL und XPDL findet sich in [HoKM06].

2.3 Modellierungsunterstützung

In diesem Abschnitt werden die Idee und die Komponenten eines Unterstützungssystems während der Geschäftsprozessmodellierung vorgestellt, die die Grundlage für alle nachfolgenden Kapitel darstellen. Es wird auch der Bezug zwischen Ontologien und Geschäftsprozessen für diese Arbeit hergestellt.

2.3.1 Einführung

Generell soll die Modellierungsunterstützung Benutzer bei der Erstellung von Geschäftsprozessmodellen helfen, indem passende und korrekte Folgeprozesse zu dem gerade editierten Geschäftsprozess aus einer Prozessbibliothek vorgeschlagen werden. Die Korrektheit der vorgeschlagenen Folgeprozesse zu einem gerade editierten Geschäftsprozess wird anhand zweier Kriterien überprüft:

1. gültige syntaktische Verbindung:
Dieses Kriterium muss aufgrund der Bipartitheit von Petri-Netzen eingehalten werden, d.h. wenn der Benutzer eine Stelle ausgewählt hat, zu der er Folgeprozesse sucht, dann müssen die Folgeprozesse mit einer Transition beginnen. Analoges gilt für Transitionen, die ausgewählt wurden. Auch hier muss es eine syntaktisch gültige Verbindung geben. Bei Petri-Netzen sind Verbindungen syntaktisch gültig, wenn sie zwischen einer Stelle zu einer Transition bestehen ($F \subseteq (S \times T) \cup (T \times S)$) und sie

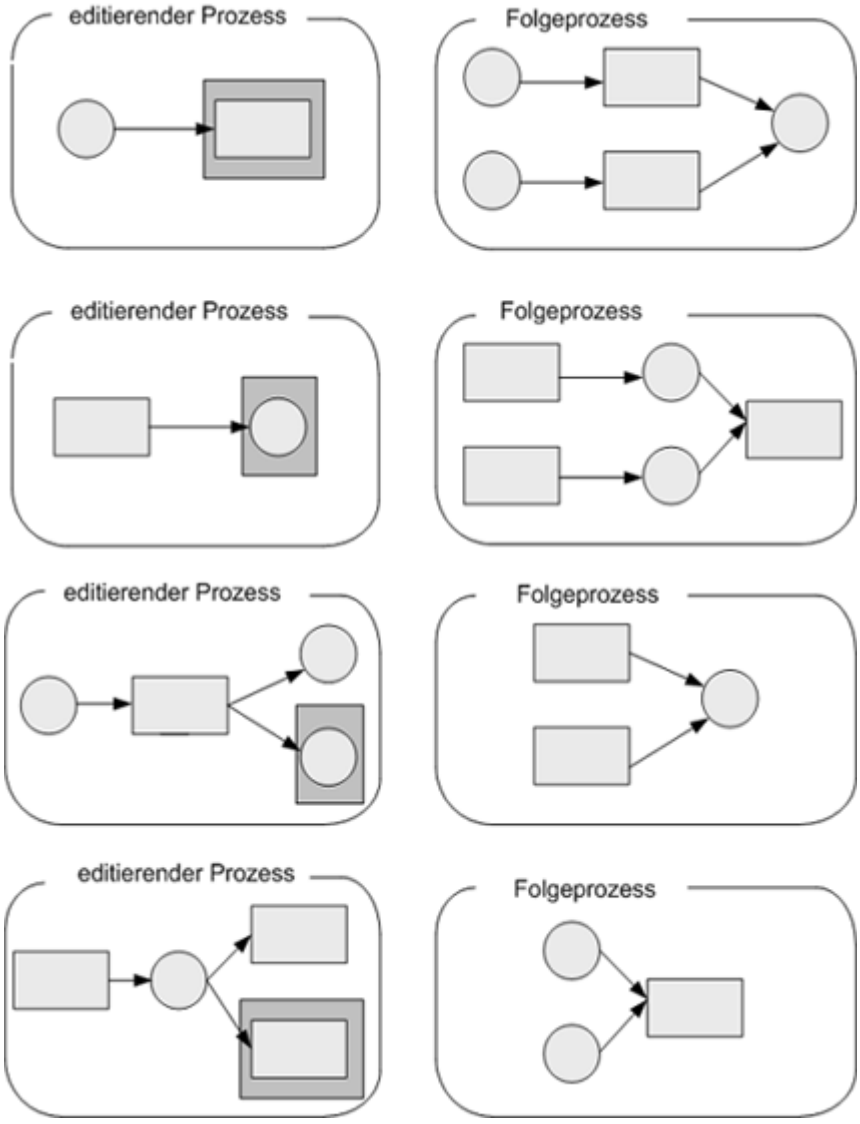
sind syntaktisch ungültig, wenn eine Stelle mit einer Stelle bzw. eine Transition mit einer Transition verbunden wird ($F_n \subseteq (S \times S) \cup (T \times T)$).

2. well-structureness:

Das Konzept der well-structureness ist bei [Aals98] formalisiert. Ein Petri-Netz PN ist well-handled, wenn für je zwei Knoten x,y mit entweder ($x \in T$ und $y \in S$) oder ($x \in S$ und $y \in T$) keine zwei Pfade von x nach y existieren, die nur x und y gemeinsam haben. Ein Workflow-Netz ist well-structured, wenn das erweiterte $\overline{PN}$ well-handled ist. Well-handleness kann in polynominaler Zeit durch Verwendung einer modifizierten Version des max-flow min-cut Technik (beschrieben in [AaHV97]) bestimmt werden.

Die Einhaltung dieses Kriterium hilft, unerwünschte Verklemmungen in Form von Deadlocks zu vermeiden. Bei der Modellierung von Geschäftsprozessen werden ein sequentieller, ein paralleler, ein iterativer Ablauf und ein Ablauf mit Konfliktsituation unterschieden [Aals98]. Das im gerade editierten Prozess modellierte Muster beeinflusst die Suche nach Folgeprozessen. Abbildung 22 zeigt „schlechte" Empfehlungen von Folgeprozessen unter der Berücksichtigung von Ablaufmustern. Bei der Realisierung der Modellierungsunterstützung sollten deswegen Ablaufmuster berücksichtigt werden, damit die Prozessintegrität gewahrt und damit weitgehend Fehler vermieden werden können.

Abbildung 21: „schlechte" Empfehlungen von Folgeprozessen



Die Eigenschaft „passend" kann über semantische bzw. linguistische Analysen überprüft werden und wird durch die folgenden zwei Kriterien bestimmt:

1. „inhaltlich" sinnvolle Folgeprozesse:
Die vorgeschlagenen Folgeprozesse müssen inhaltlich sinnvoll zu dem gerade editierten Geschäftsprozess sein. Dieses Kriterium wird anhand der im Kapitel 4 definierten Ähnlichkeitsmaße überprüft.

2. Identisches Abstraktionsniveau:
Die meisten Modellierer vermischen die Abstraktionsgrade von Prozessaktivitäten, indem sie auf der gleichen Prozessebene einige Aktivitäten detaillierter und andere weniger detailliert darstellen. Für den konsistenten Entwurf eines Informationssystems ist allerdings ein gleicher Detaillierungsgrad wichtig. Petri-Netze unterstützen

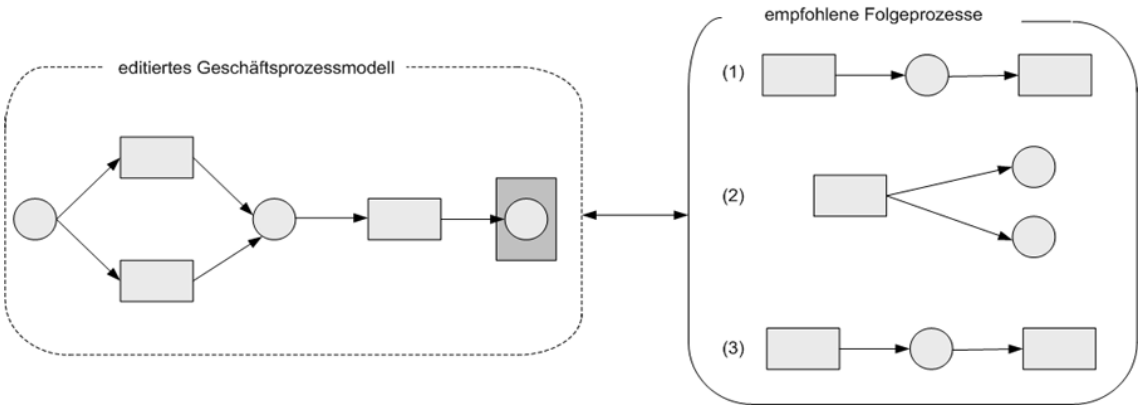
die Systemdarstellung auf verschiedenen Abstraktionsebenen. Generell können Prozesshierarchien durch Verfeinerung von Transitionen zu Subprozessen beliebig gebildet werden. Es sollte aber darauf geachtet werden, dass die Aktivitäten in einem Diagramm das gleiche Abstraktionsniveau besitzen. Für die in dieser Arbeit vorgestellte Modellierungsunterstützung werden fünf Hierarchieebenen unterschieden:

1) Prozesslandkarte (Abstrakte Darstellung der Kernprozesse)
2) Kontextdiagramm (Austausch von Ergebnissen zwischen Kernprozessen)
3) Prozessebene 1 (Konkrete Modellierung eines Geschäftsprozesses)
4) Prozessebene 2 (Verfeinerung einer Transition aus Prozessebene 1)
5) Prozessebene 3 (Verfeinerung einer Transition aus Prozessebene 2)

Der Benutzer kann die Modellierungsunterstützung auf jeder Hierarchieebene aufrufen.

Abbildung 22 zeigt den Vorschlag von korrekten Folgeprozessen zu einem gerade editierten Geschäftsprozess. Zunächst muss der Anwender ein Prozesselement auswählen, zu dem er Folgeprozesse sucht (in Abbildung 22 ist das die dritte Stelle). Der Geschäftsprozess wurde als Petri-Netz modelliert und die Folgeprozesse sollen auch als Petri-Netz angezeigt werden

Abbildung 22: Idee einer Unterstützung während der Geschäftsprozessmodellierung



Die Visualisierung von passenden und korrekten Folgeprozessen kann auf zwei unterschiedliche Weisen erfolgen**:**

- semi-automatisch: der Benutzer sucht mit Hilfe einer Anfragesprache nach entsprechenden Knoten und fügt diese selbst in sein Prozessmodell ein,

- automatisch: eine Empfehlungskomponente schlägt aufgrund bestimmter Kriterien, die bei der Prozessmodellierung eingehalten werden sollen, die entsprechenden Folgeprozesse vor.

Durch die Verwendung einer Anfragesprache können Elemente von Prozessmodellen nach bestimmten Eigenschaften gefiltert werden, indem beispielsweise nur alle Transitionen eines Prozesses angezeigt werden sollen. Das hat den Vorteil, dass der Benutzer zielgerichtet die für ihn relevanten Knoten auswählen und Ablaufmuster verändern kann (Abläufe, die vorher sequentiell waren, werden nun mit Hilfe der ausgewählten Transitionen parallel modelliert).
Beim automatischen Vorschlag hat der Benutzer keinen direkten Einfluss auf die vorgeschlagenen Folgeprozesse. Er kann aber beispielsweise Geschäftsregeln angeben, die für den zu modellierenden Geschäftsprozess gelten müssen und somit die Auswahl an passenden Folgeprozessen einschränken. Diese Art der Visualisierung eignet sich vor allem für Modellierungsanfänger, da zum einen nur korrekte Folgeprozesse angezeigt werden und die Folgeprozesse bestimmten Geschäftsregeln entsprechen. Diese beiden Punkte helfen Modellierungsfehler weitgehend zu vermeiden. Zum anderen muss der Benutzer über keine Informationen von bestehenden Geschäftsprozessen verfügen, da passende Folgeprozesse automatisch ausgewählt werden.

2.3.2 Anforderungen

Nachfolgend werden geforderte Eigenschaften einer Modellierungsunterstützung skizziert. Eigenschaften wie effizienter Zugriff auf Folgeprozesse oder die Skalierbarkeit beim Wachstum der Menge an Prozessfragmenten sollen direkt vom Datenbankmanagementsystem gewährleistet werden. Es werden nur Eigenschaften des Modellierungsunterstützungssystems beschrieben.

- *Übersichtlichkeit*

Übersichtlichkeit von vorgeschlagenen Folgeprozessen hängt mit der Anzahl an Elementen eines Folgeprozesses ab. Eine „gute" Anzahl an Elementen, die praktisch erprobt wurde, sind 15 Elemente. Diese Anzahl an Gesamtelementen sollte auch beim Vorschlag von passenden Folgeprozessen berücksichtigt werden. D.h., wenn der Benutzer bereits 5 Elemente modelliert hat, dann sollte ihm ein Folgeprozess mit maximal 10 Elementen vorgeschlagen werden. Generell gilt, je kürzer die Folgeprozesse, desto schneller kann der Benutzer das für ihn passendere Prozess-

stück auswählen. Wobei die Folgeprozesse mindestens vier (eher sechs) Elemente enthalten sollten, damit die Modellierungsunterstützung dem Benutzer einen Nutzen stiftet.

- *Berücksichtigung von Benutzertypen*

Bei der Modellierungsunterstützung sollten unterschiedliche Benutzertypen (Anfänger, Fortgeschrittene, Experten) durch eine nutzerindividuelle Empfehlung von Folgeprozessen unterschieden werden können. Experten können aufgrund ihrer zahlreichen Modellierungserfahrungen schneller Prozessmodelle überblicken und als passender bzw. unpassender einschätzen. Somit können Experten mehr Folgeprozesse als Anfängern vorgeschlagen werden, ohne dass es dabei zu einer signifikanten Verzögerung bei der Prozessmodellierung kommt. Bei der Speicherung von Geschäftsprozessen in der Prozessbibliothek sollte deswegen auch der Benutzertyp des modellierten Geschäftsprozesses annotiert werden. Über eine entsprechende Annotation könnten Aussagen über die Qualität der Folgeprozesse gemacht werden. Geschäftsprozesse, die von Experten modelliert wurden, werden vermutlich qualitativ hochwertiger sein als die eines Modellierungsanfängers.

- *Reihenfolge der Modellierung*:

Petri-Netze werden in der Regel von links nach rechts modelliert. Denkbar ist auch eine Modellierung von oben nach unten. Dabei stellt das erste Element (auf der linken Seite) den Input bzw. den Prozessauslöser und das letzte Element (auf der rechten Seite) den Output des Geschäftsprozesses bzw. das Prozessergebnis dar. Der Geschäftsprozess kann sowohl vorwärts ausgehend vom Auslöser als auch rückwärts ausgehend von den Ergebnissen modelliert werden. Die Modellierungsunterstützung sollte unabhängig von der Reihenfolge der Modellierung verfügbar sein; d.h. sowohl wenn beginnend beim Prozessauslöser als auch wenn beginnend beim Prozessergebnis modelliert wird.

- *Eigenschaften der Prozessfragmente:*

Prozessfragmente, die in der Prozessbibliothek gespeichert werden, müssen bestimmte strukturelle Eigenschaften wie beispielsweise die Free-Choiceness erfüllen (es ist nicht erlaubt, die Ablaufmuster Alternative und Synchronisation zu kombinieren). A-priori Analysen reduzieren die Suchzeit nach korrekten Folgeprozessen, weil die gespeicherten Prozessfragmente bereits korrekt sind. Die syntaktische Korrekt-

heit muss lediglich bei der Zusammenführung des gerade editierten Geschäftsprozesses und der möglichen Folgeprozesse überprüft werden.

- *Allgemeingültige Nutzung der Modellierungsunterstützung*

Die Modellierungsunterstützung soll in jedem Petri-Netz-Werkzeug genutzt werden können. Ein allgemeingültiges Speicherformat für Petri-Netz-Werkzeuge unterstützt – im Gegensatz zu einem proprietäre Speicherformat – einen verlustfreien Austausch von Petri-Netz-basierten Geschäftsprozessmodellen. Damit kann ein gerade editiertes Prozessmodell in ein anderes Petri-Netz-Werkzeug importiert werden ohne dass dabei Informationen über die Darstellung oder Graphik der Elemente verloren gehen. Ein weit verbreitetes Speicher- und Austauschformat für Petri-Netze ist die Petri Net Markup Language [Webe02].

2.3.3 Namenskonventionen

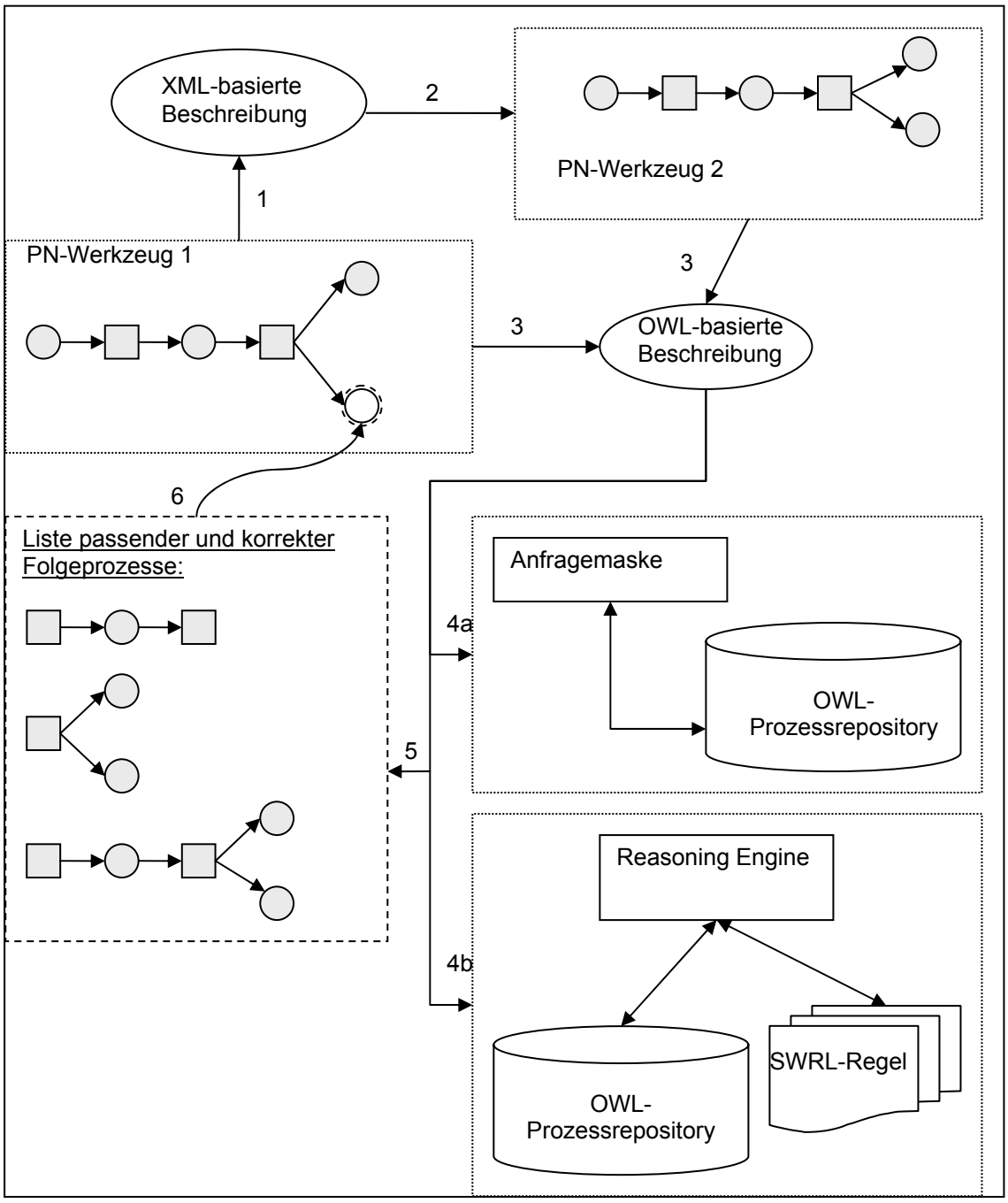
Damit Fehler bei der Modellierung von Geschäftsprozessen weitgehend reduziert werden können, müssen Namenskonventionen für Prozesselemente eingehalten werden. Für die Namensgebung von Transitionen sollten Verben zusammen mit Substantiven verwendet werden. Es werden keine substantivierten Verben verwendet. Verben wie verwalten, aktualisieren, anlegen, ändern deuten häufig auf eine funktionsorientierte Denkweise hin, in diesem Fall sollte nochmals überprüft werden, ob prozessorientiert dokumentiert wird. Für die Namensgebung von Stellen werden Substantive im Plural verwendet; zumeist mit einem Zusatz, der den aktuellen Status beschreibt, z.B.: erfasste Aufträge, geprüfte Aufträge, freigegebene Fertigungsaufträge. Die Einhaltung von Namenskonventionen ist vor allem bei der Berechnung von semantischen Ähnlichkeiten zwischen den Prozessobjekten des editierten Prozesses und des Folgeprozesses notwendig. Eine unpräzise Benennung von Prozessobjekten kann zu unzureichenden Informationen bei der Ähnlichkeitsberechnung auf Basis einer Ontologie führen.
Bei der Realisierung der Modellierungsunterstützung wird die Einhaltung der Namenskonventionen vorausgesetzt.

2.3.4 Komponenten

Die Idee einer Modellierungsunterstützung soll in den nächsten Kapiteln wie in der Abbildung 23 gezeigt, realisiert werden. Ausgehend von Petri-Netz-basierten Geschäftsprozessmodellen (PN-Werkzeug 1) soll es möglich sein, Geschäftsprozess-

modelle zwischen Werkzeugen austauschen zu können und die Modellierungsunterstützung mit den erweiterten Komponenten in jedem Petri-Netz-Werkzeug nutzen zu können. Hierfür wird eine XML-basierte Darstellung von Petri-Netzen benötigt (1), die durch Import in ein anderes PN-Werkzeug (2) das identische im PN-Werkzeug 1 exportierte Geschäftsprozessmodell anzeigt.
Zur Nutzung einer Empfehlungskomponente muss der Geschäftsprozess in ein Format transformiert werden, das Inferenzmechanismen unterstützt (im Kapitel 2.1. wurde ausführlich die Fähigkeit von OWL zum Einsatz von Inferenzmechanismen erläutert). Zunächst wird der gerade editierte Geschäftsprozess automatisch in ein OWL-basiertes Beschreibungsformat übersetzt (3). Die Modellierungsunterstützung kann dann auf zwei unterschiedliche Weisen aufgerufen werden. Einmal mit Hilfe einer Anfragesprache, indem deren Syntax in einer entsprechenden Maske formuliert wird (4a) oder automatisch über eine Empfehlungskomponente (4b). Zur Realisierung der Modellierungsunterstützung wird es ein sogenanntes Prozessrepository geben. In dieser Abbildung sind zwei Prozessrepositories eingezeichnet, um den unabhängigen Zugriff über die Anfragesprache und die Empfehlungskomponenten zu verdeutlichen. Nachdem sich Anwender gewöhnlich bei der Modellierung von Geschäftsprozessen an interne Unternehmensrestriktionen halten müssen, wird das OWL-basierte Beschreibungsformat dahingehend erweitert, dass auch Geschäftsregeln in dem Format integriert und somit verarbeitet werden können. Zur Erweiterung wird die Semantic Web Rule Language (SWRL) verwendet. Mit Hilfe eines Inferenzmechanismus (oder über die Anfragesprache) werden Folgeprozesse aus einer Prozessbibliothek ermittelt und dem Benutzer zur Vervollständigung seines Geschäftsprozesses vorgeschlagen (5). Der Benutzer kann den Folgeprozess dann gegebenenfalls in seiner Modellierungsumgebung verändern. In der Abbildung wird zweimal ein OWL-Prozessrepository dargestellt, was allerdings nicht eine redundante Datenhaltung implizieren soll. Diese doppelte Darstellung des Repositories soll verdeutlichen, dass das Feature „ähnlichkeitsbasierte Anfragesprache" und „regelbasierte Empfehlungskomponenten" zwei unabhängige Features sind, die aber auf demselben Prozessrepository arbeiten.

Abbildung 23: Komponenten eines Modellierungsunterstützungssystems für Geschäftsprozesse



Im Kapitel 2 wurden Grundlagen zu Ontologien und Geschäftsprozessen beschrieben, auf denen in den nächsten Kapiteln aufgebaut wird. So bilden Geschäftsprozesse die Ausgangslage für die Modellierungsunterstützung. Durch die Verwendung von Ontologien kann ein einheitliches Verständnis über Begriffe und die Beziehungen zwischen Begriffen von Geschäftsprozessmodellen definiert werden.

Das nächste dritte Kapitel beschreibt das XML- und OWL-basierte Beschreibungsformat für Geschäftsprozessmodelle. Im Kapitel 4 werden Ähnlichkeitsmaße defi-

niert, mit denen ähnliche Prozesselemente und synonym verwendete Begriffe in Geschäftsregeln und Prozessen gefunden werden können. Mit Hilfe der Anfragesprache (Kapitel 5) lassen sich passende Folgeprozesse mit bestimmten Eigenschaften (z.B. einem bestimmten Ähnlichkeitswert) ermitteln. Die automatische Empfehlungskomponente wird im Kapitel 6 beschrieben.

3 Semantische Beschreibung von Geschäftsprozessmodellen

Aufbauend auf den Grundlagen zu Ontologien und Geschäftsprozessen werden in diesem Kapitel zunächst ein XML-basiertes Austauschformat und anschließend ein OWL-basiertes Beschreibungsformat für elementare und höhere Petri-Netze (insbesondere Pr/T-Netze) beschrieben. Das im nächsten Kapitel beschriebene XML-basierte Austauschformat kann als ein allgemeingültiges Speicherformat für Petri-Netz-Werkzeuge verwendet werden und ersetzt proprietäre Speicherformate. Für die automatische Modellierungsunterstützung wird aber ein Format benötigt, das Inferenzen unterstützt. Im Kapitel 2 wurde bereits auf die semantische Unzulänglichkeit von XML für dieses Anwendungsszenario hingewiesen. Deshalb wird im Kapitel 3.2. ein OWL-basiertes Beschreibungsformat für Petri-Netze vorgestellt [KoOb05, KoOb07a]. Dieses Kapitel endet mit einer Zusammenfassung bestehender Ansätze zur semantischen Beschreibung von Petri-Netzen und Web-Services und einer Abgrenzung dieser Ansätze zu dem in diesem Kapitel vorgestellten OWL-basierten Beschreibungsformat für Petri-Netze.

3.1 XML-basiertes Austauschformat für Petri-Netze

Ein erster Vorschlag für ein allgemeingültiges Dateiformat für Petri-Netze war die *Abstract Petri Net Notation* (APNN) [BaKK95]. Dabei wird unter einem allgemeingültigen Dateiformat die einfache Erweiterung eines maschinenlesbaren Formats zur Repräsentation verschiedener Petri-Netz-Typen oder graphischer Informationen verstanden. Allerdings müssen die für APNN existierenden Parser für jeden neuen Petri-Netz-Typ konfiguriert werden. Eine einfache Implementierung neuer Petri-Netz-Typen und eine werkzeugunabhängige Repräsentation von Petri-Netzen wurde mit der Petri Net Markup Language (PNML) [Webe02] vorgeschlagen[24].

[24] Die Implementierung neuer Petri-Netz-Typen ist mit dem Petri-Netz-Kern [WeKi03] möglich.

3.1.1 Austauschformat für elementare Petri-Netze

PNML ist ein XML-basiertes Dateiformat für Petri-Netz-Werkzeuge und ein Austauschformat für Petri-Netz-Modelle. Die Verwendung von XML als Speicherformat löst die bis dahin proprietären Speicherformate[25] ab. Als Anforderungen an PNML als Austauschformat wurden formuliert:

- *Lesbarkeit*: Das Format soll für den Menschen leicht lesbar sein und Änderungen unterstützen,
- *Allgemeinheit*: Das Format soll verschiedene Typen von Petri-Netzen unterstützen. Es soll möglich sein, das Dateiformat um weitere Petri-Netz-Typen zu erweitern,
- *Graphische Repräsentation:* Alle Elemente und die graphischen Zusammenhänge der Elemente eines Petri-Netzes müssen in dem Format definiert werden können.

Zur Beschreibung eines allgemeinen Austauschformats für Petri-Netze wird bei PNML zwischen typspezifischen und netztypunabhängigen Elementen unterschieden. Netztypunabhängige Elemente sind Elemente, die alle Petri-Netz-Typen gemeinsam haben. Typspezifische Elemente sind Elemente, die nur in einem bestimmten Petri-Netz-Typ vorkommen.

Zu den netztypunabhängigen Elementen zählen:

- **Netzelemente:** Jedes Petri-Netz besteht, wie bereits in Kapitel 2.2 angesprochen, aus den drei Elementen Stellen, Transitionen und Kanten. Petri-Netze, die zerlegt werden müssen, weil sie nicht auf einer Seite modelliert werden können, werden in PNML mit den beiden Konzepten Seite und Modul realisiert. Seiten erlauben, dass ein Netz auf verschiedenen Seiten gezeichnet werden kann. Mittels einer Annotation bei Referenzstellen/Referenztransitionen wird auf die entsprechende zerlegte Seite verwiesen. In einem Modul wird eine Schnittstelle des Petri-Netzes definiert. Im Gegensatz zu Seiten kann in Modulen nur über die Schnittstelle Einfluss auf das Petri-Netz genommen werden.
- **Identifikator:** Der Identifikator eines Petri-Netzes oder eines Netz-Elements wird durch das *id*-Attribut angegeben. Der Wert dieses Attributes muss die Anforderungen an den Attributtyp *ID* von XML erfüllen[26].

[25] Alle Petri-Netz-Werkzeuge hatten bis dahin eigene Speicherformate.
[26] Das bedeutet, dass der Wert eine Zeichenkette sein muss, die mit einem Buchstaben oder dem Unterstrich-Zeichen beginnt.

Über den Identifikator eines Elements kann auf Elemente (Stellen/Transitionen) im Nachbereich verwiesen werden.

- **Label:** Jedes Netzelement wird durch einen Label erweitert. Zulässige Label und mögliche Kombinationen hängen vom jeweiligen Petri-Netz-Typ ab. Es werden zwei Arten von Labels unterschieden: *Annotation* und *Attribute*. Der Name eines PNML-Elements, die Markierung einer Stelle oder die Kanteninschrift sind Beispiele für Annotationen. Typischerweise ist eine Annotation ein Label mit einem unbeschränkten Wertebereich. Ein Attribut ist dagegen ein Label mit beschränktem Wertebereich. Ein Beispiel dafür ist der Kantentyp, der das Attribut einer Kante beschreibt und einen der Werte *normal, read, inhibitor* oder *write* besitzt. Die Richtung der Kante hängt von diesem Attribut ab (der Kantentyp von einer Stelle zu einer Transition hat den Wert *read*; der Kantentyp von einer Transition zu einer Stelle den Wert *write*). Der grundlegendste Unterschied zwischen Annotationen und Attributen ist, dass Annotationen als Text eines Netzelements angezeigt werden; ein Attribut hat dagegen eine Auswirkung auf die Form des Netzelements selbst. Außerdem hat eine Annotation eine eigene graphische Information (s. u.), während ein Attribut diese nicht hat.

  Beispiele für Labels sind <initialMarking>, <name> (das sind Annotationen) oder <arc typ="read"> (ein Attribute).
- **Graphische Informationen:** Jedes Netzelement und jedes Label ist mit Informationen verbunden, die die jeweilige graphische Erscheinung bestimmen. Für einen Knoten eines Netzes ist das beispielsweise die Position in einem (x,y)-Koordinatensystem; für eine Kante ist es eine Liste von Zwischenpunkten und der Kantenverlauf; und für Labels sind es die relativen Positionen zu ihrem Netzelement (Abstände). Absolute und relative Positionen beziehen sich auf den Referenzpunkt eines Netzelementes oder einer Annotation. Der Referenzpunkt ist die Mitte der graphischen Repräsentation für ein Netzelement (für eine bestimmte Stelle ist der Referenzpunkt die Mitte des Kreises). Der Referenzpunkt einer Kante ist die Mitte des ersten Segments einer Kante. Alle Positionen beziehen sich hierbei auf ein kartesisches Koordinatensystem. Positionen werden in PNML mit dem `<position>`-Element eingeleitet. Abstände von Elementen zueinander mit dem `<offset>`-Element.

  In PNML können noch graphische Informationen wie *Dimension*, *Linie* und *Font* für Netzelemente angegeben werden. Tabelle 1 zeigt alle mögli-

chen Unterelemente des <graphics>-Elements und deren Attribute angegeben in [Webe02].

Tabelle 1: Kindelemente des <graphics>-Elements

XML element	Attribute	Domain
<position>	x	decimal
	y	decimal
<offset>	x	decimal
	y	decimal
<dimension>	x	nonNegativeDecimal
	y	nonNegativeDecimal
<fill>	color	CSS2-color
	image	anyURI
	gradient-color	CSS2-color
	gradient-rotation	{vertical, horizontal, diagonal}
<line>	shape	{line, curve}
	color	CSS2-color
	width	nonNegativeDecimal
	style	{solid, dash, dot}
<font>	family	CSS2-font-family
	style	CSS2-font-style
	weight	CSS2-font-weight
	size	CSS2-font-size
	decoration	{underline, overline, line-through}
	align	{left, center, right}
	rotation	decimal

- **Werkzeugspezifische Informationen:** Bei der Speicherung von Petri-Netz-Modellen werden zusätzliche werkzeugspezifische Informationen hinterlegt. Diese Informationen sind für andere Werkzeuge nicht relevant, sondern haben nur Einfluss auf das Petri-Netz in Bezug auf das spezielle Werkzeug. Für das innere Format dieser werkzeugspezifischen Informationen gibt es keine Vorgaben. Es muss lediglich die werkzeugspezifische Information markiert (über das Element <toolspecific/>) und dem speziellen Werkzeug zugewiesen werden.

Tabelle 2 veranschaulicht alle netztypunabhängigen PNML-Elemente nach [Webe02]. **PetriNet**, **Place**, **Transition**, **Arc**, **Page**, **RefPlace** und **RefTrans** sind Netzelemente. Jedes dieser Netzelemente hat einen Identifikator, der über ein eindeutiges XML-Attribut (id) zugewiesen wird. Graphische Informationen werden über

die Elemente `<graphics>` und werkzeugspezifische Informationen über die Elemente `<toolspecific>` definiert.

Tabelle 2: Netztypunabhängige PNML-Elemente

Class	XML element	XML Attributes
PetriNetDoc	<pnml>	
PetriNet	<net>	id: ID type: anyURI
Place	<place>	id: ID
Transition	<transition>	id: ID
Arc	<arc>	id: ID source: IDRef (Node) target: IDRef (Node)
Page	<page>	id: ID
RefPlace	<referencePlace>	id: ID ref: IDRef (Place or RefPlace)
RefTrans	<referenceTransition>	id: ID ref: IDRef (Transition or RefTrans)
ToolInfo	<toolspecific>	tool: string version: string
Graphics	<graphics>	

Typspezifische Elemente

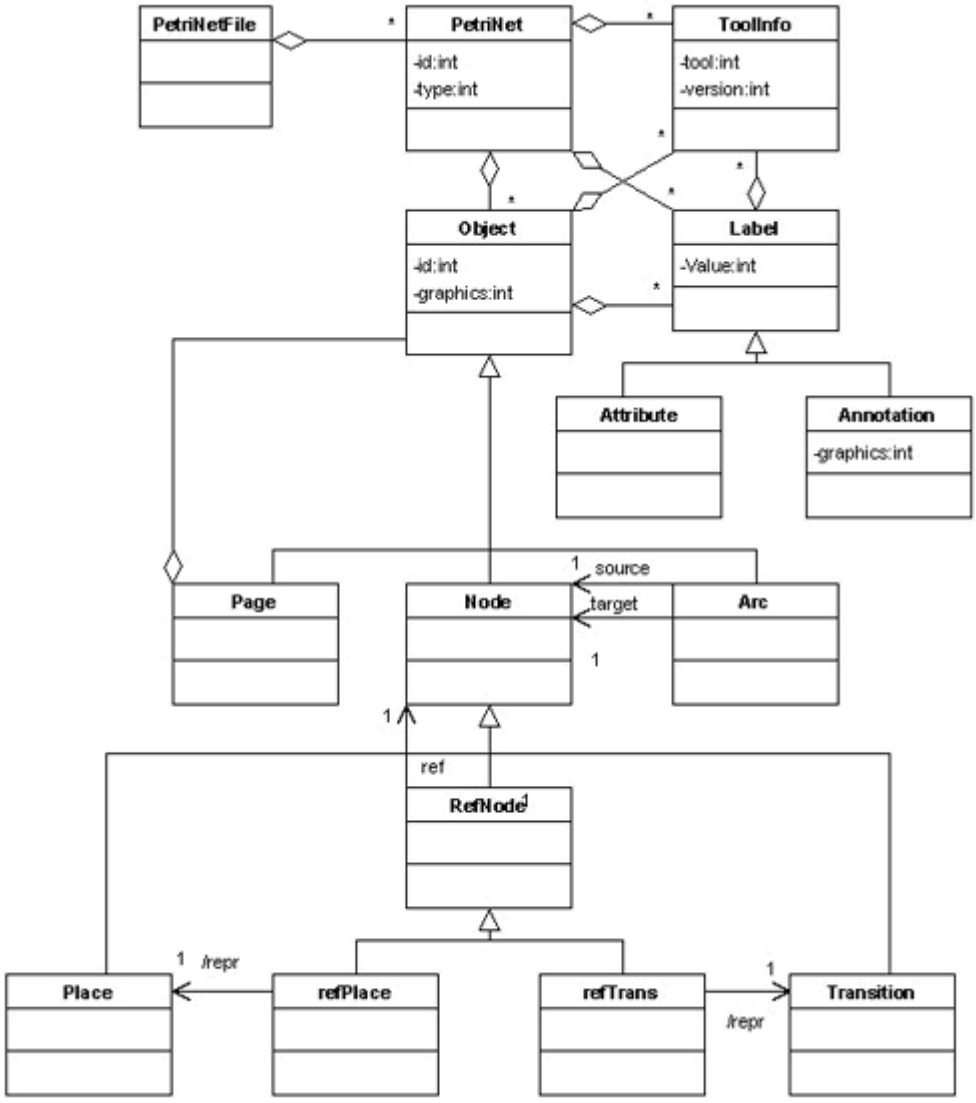
Bei typspezifischen Elementen werden Labels entsprechend dem Petri-Netz-Typ definiert. Die Art, die Anzahl der Labels sowie ihre zulässige Kombination in einem Netz werden im Petri-Netz-Typ festgelegt. Für PNML ist ein Petri-Netz-Typ eine Ansammlung von Labeldefinitionen, die als Petri-Netz-Typdefinition (PNTD) bezeichnet wird. Jede Labeldefinition legt den Wertebereich des Labels und die Art des Netzelementes fest, mit dem das Label verbunden ist. Aus einem Vorrat von PNTD-Dokumenten wird dasjenige zur Definition einer speziellen PNML für Netze eines entsprechenden Typs ausgesucht, das das PNML-Dokument instanziiert. Das bedeutet, dass ein PNML-Dokument für spezielle Petri-Netz-Typen aus dem entsprechenden PNTD-Dokument erzeugt wird.

Abbildung 24 zeigt alle Elemente von PNML in einer UML-basierten Notation. Beispielsweise steht **PetriNetFile** für ein XML-Dokument und **PetriNet** für alle möglichen Petri-Netze mit den XML-Attributen id und type. Das Element **PetriNet** kann[27] durch die Elemente **ToolInfo** (werkzeugspezifische Informationen), **Label** und **Ob-**

[27] In UML-Klassendiagrammen [HKKR05] werden vier Beziehungstypen unterschieden: Generalisierung (Subsumptionsbeziehung; modelliert als dreieckige Spitze), Assoziation (wird durch die Zusammenfassung von Objektbeziehungen zwischen Objekten aus denselben Klassen gebildet), Aggregation (drückt die „besteht aus"-Beziehung aus und wird als Linie mit einem weißen Diamanten modelliert) und Komposition (eine spezielle Form der Aggregation, die mit einem schwarzen Diamanten modelliert wird).

ject beschrieben werden. Dabei ist das Element **Node** immer ein **Object** und **Attribute** und **Annotation** sind immer ein **Label.**

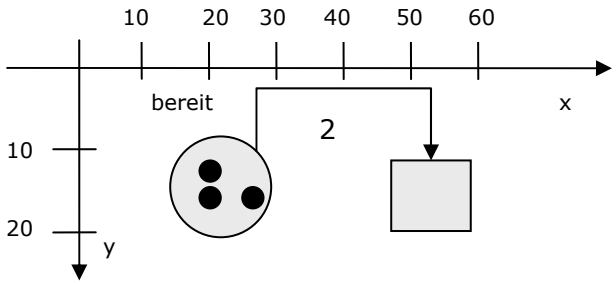
Abbildung 24: UML-basierte Darstellung von PNML



Technische Realisierung des Austauschformats

Am Beispiel des Petri-Netzes in Abbildung 25 wird die Umsetzung des Netzes nach PNML erklärt. Das Petri-Netz besteht aus einer Stelle und einer Transition, die über eine Kante miteinander verbunden sind. Die Stelle hat drei Marken und wurde mit *bereit* benannt.

Abbildung 25: Ein Beispiel-Petri-Netz



Jedes PNML-Dokument ist ein XML-Dokument mit dem Wurzelelement `<pnml>`, das beliebig viele `<net>`-Elemente enthalten kann. Zur Beschreibung des Netzes werden im `<net>`-Element die XML-Attribute `id` (zur eindeutigen Identifikation des Netzes) und `type` (zur eindeutigen Definition des Petri-Netz-Typs) verwendet. Jedes Netz hat einen Namen, der mit dem `<name>`-Element eingeleitet wird. Netzelemente sind in PNML nach ihren Konzepten benannt, z.B. `<place>` für eine Stelle, `<arc>` für eine Kante oder `<referenceTransition>` für eine Referenztransition. Dabei hat jedes Netzelement einen eindeutigen Identifikator. Der Wert des Identifikators wird als Wert des XML-Attributs id angegeben.
Der Quelltext 10 zeigt die Umsetzung der Stelle aus Abbildung 25 in PNML-Syntax.

Quelltext 10: PNML-Syntax der Stelle aus Abbildung 25

```
<place id="pl">
      <graphics>
            <position x="22" y="14"/>
      </graphics>
      <name>
            <text>bereit</text>
            <graphics>
                  <offset x="-6" y="-10"/>
            </graphics>
      </name>
      <initialMarking>
            <text>3</text>
            <toolspecific tool="PN4all" version="0.1">
                  <tokenposition x="-2" y="-2"/>
                  <tokenposition x="-2" y="2"/>
                  <tokenposition x="-5" y="2"/>
            </toolspecific>
      </initialMarking>
</place>
```

Der Stelle *bereit* mit den Koordinaten (22, 14) wurde der Identifikator p1 zugewiesen. Der Name der Stelle wird in dem Element `<name>` und den Kindelementen `<text>` und `<offset>` beschrieben. Das Label *bereit* hat einen Abstand von (-8, -9)

zu den Koordinaten der Stelle. Die initiale Markierung der Stelle ist 3 (es liegen drei Marken in der Stelle). Die individuellen Positionen der Marken zur den Koordinaten der Stelle werden im Kindelement <tokenposition> hinterlegt. Der Quelltext 11 zeigt die PNML-Umsetzung der Transition für das in Abbildung 25 modellierte Petri-Netz.

Quelltext 11: PNML-Syntax der Transition aus Abbildung 25

```
<transition id="t1">
      <graphics>
            <position x="53" y="15"/>
      </graphics>
</transition>
```

Das <transition>-Element hat (genauso wie das <place>-Element) einen eindeutigen Identifikator, der den Wert t1 hat. Die Position der Transition wird über das Kindelement <position> des <graphics>-Elements angegeben.

Neben Stellen und Transitionen gibt es noch Kanten. Eine Kante, die mit dem XML-Element <arc> definiert wird, hat eine Quelle (source) und ein Ziel (target). Auch eine Kante hat einen eindeutigen Identifikator, damit Kanten zwischen gleichen Knoten unterschieden werden können. Die graphische Information einer Kante wird mit dem <graphics>-Element beschrieben, das <position>-Kindelemente hat. Kanteninschriften werden durch die Annotation <inscription> eingeleitet, die ebenfalls ein <graphics>-Element enthalten. Das Element <offset> (Kindelement von <graphics>) gibt den relativen Abstand der Kanteninschrift zum Referenzpunkt der Kante an. Quelltext 12 veranschaulicht die Umsetzung der Kante des Petri-Netzes aus Abbildung 25 in PNML-Syntax.

Quelltext 12: PNML-Syntax der Kante aus Abbildung 25

```
<arc id="a1" source="p1" target="t1">
      <graphics>
            <position x="28" y="10"/>
            <position x="53" y="10"/>
      </graphics>
      <inscription>
            <text>2</text>
            <graphics>
                  <offset x="15" y="-6"/>
            </graphics>
      </inscription>
</arc>
```

Der eindeutige Identifikator der Kante von p1 nach t1 lautet a1. Dabei hat der Startpunkt der Kante die Koordinaten (28, 10) und der Endpunkt liegt bei (53, 10). Die Inschrift der Kante und der Abstand der Kanteninschrift werden in den Kindelementen des Elements `<inscription>` beschrieben.

3.1.2 Austauschformat für höhere Petri-Netze

Die allgemeine Verwendbarkeit von PNML wird nachfolgend genutzt, um ein XML-basiertes Austauschformat für höhere Petri-Netze zu definieren – insbesondere für strikte Pr/T-Netze, die im zweiten Kapitel eingeführt wurden. Das Austauschformat Pr/TML (Pr/T Markup Language) besteht aus netztypunabhängigen Elementen wie in PNML definiert (Netzelemente, Identifikator, Labels, graphische Informationen und werkzeugspezifische Informationen) und typspezifischen Elementen.
Zu den typspezifischen Elementen zählen die folgenden Labels:

- **Condition** und **Operation** für die Transitionsinschrift in Form eines über Ψ und der Menge der an den adjazenten Kanten vorkommenden Variablen gebildeten prädikatenlogischen Ausdrucks,
- **Relation** für die Menge *PR* von auf der Individuenmenge *D* definierten Prädikaten mit unveränderlichen Ausprägungen,
- **Tuple** für die Elemente der Individuenmenge *D*,
- **Attribute** für eine auf *D* definierte Menge von Funktionen *FT*

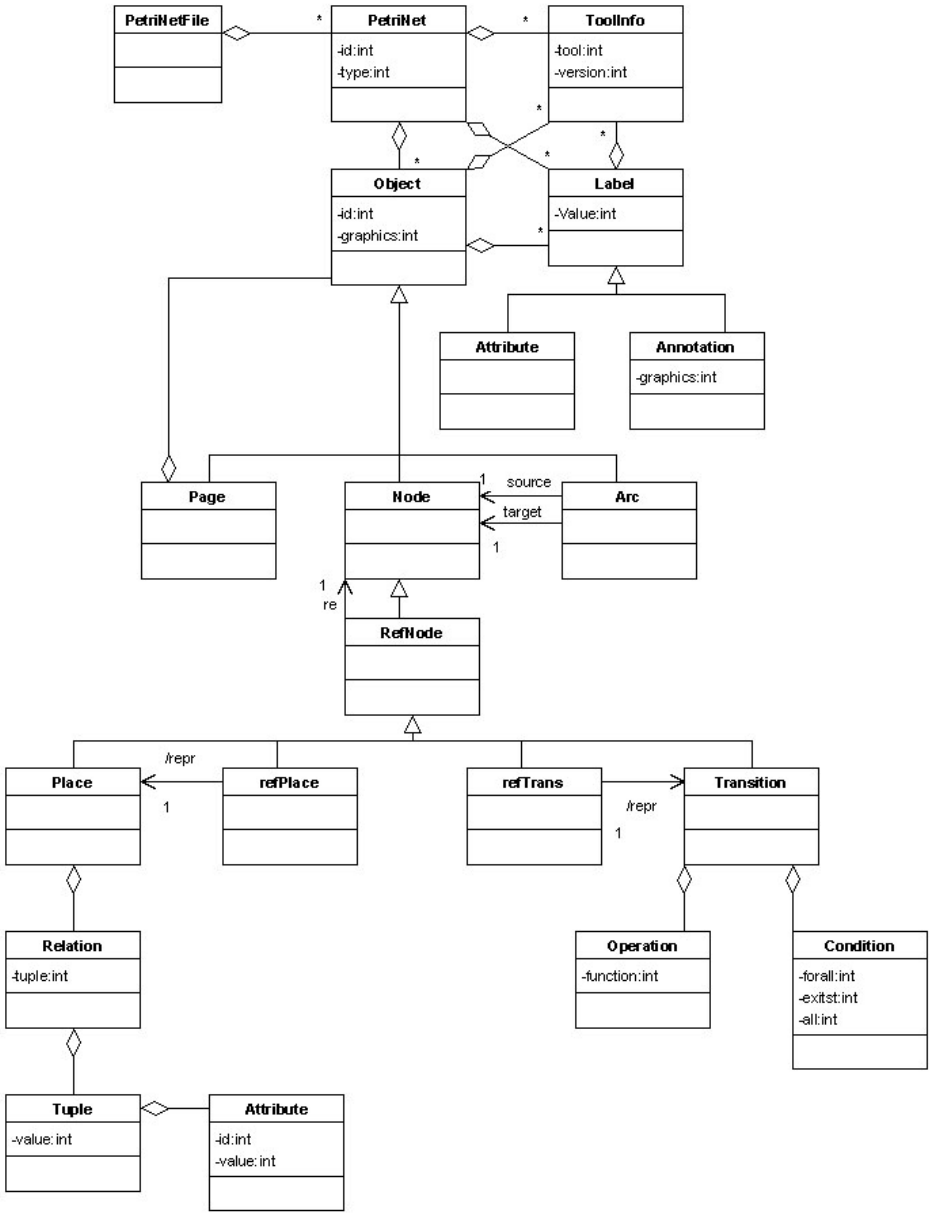
Tabelle 3 zeigt die typspezifischen Elemente des Pr/TML-Austauschformats.

Tabelle 3: Typspezifische Pr/TML-Elemente

Class	XML element	XML attributes
Condition	<conditon>	forall: string exists: string and: string
Operation	<operation>	function: string
Relation	<relation>	tuple: Tuple
Tuple	<tuple>	attribute: Attribute
Attribute	<attribute>	id = ID value: string

Abbildung 26 zeigt die Erweiterung der UML-basierten Notation aus Abbildung 24 um die typspezifischen Pr/TML-Elemente. Dabei kann eine Transition als Inschriften eine Bedingung und/oder den Namen einer Operation haben.

Abbildung 26: UML-basierte Notation der Pr/TML-Elemente

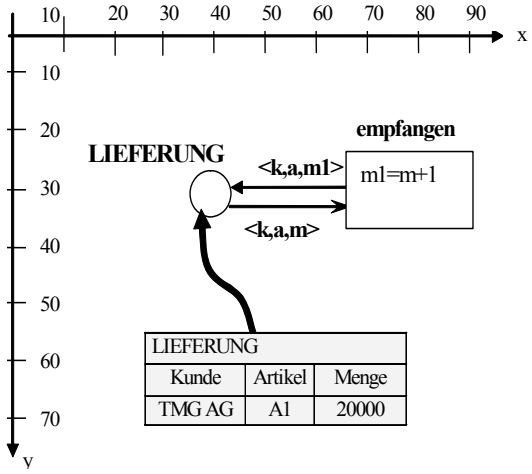


Technische Realisierung des Austauschformats

Am Beispiel des Petri-Netzes in Abbildung 27 wird die Realisierung in Pr/TML erklärt. Das Pr/T-Netz besteht aus einer Stelle *Lieferung* und einer Transition *empfangen* mit der Transitionsinschrift m1=m+1. Die Lieferung wird durch die Relation *Lieferung* mit den Attributen *Kunde*, *Artikel*, *Menge* und den Attributwerten *TMG*

AG, *A1* und *20000* identifiziert. Die Stelle *Lieferung* und die Transition *empfangen* sind über eine Lösch- und eine Einfügekante verbunden. Die Löschkante hat die Inschrift **<**k,a,m**>** und die Einfügekante die Inschrift **<**k,a,m1**>**.

Abbildung 27: Beispiel Pr/T-Netz



Jedes Pr/TML-Dokument wird durch das Wurzelelement `<prtml>` eingeleitet und kann beliebig viele `<net>`-Elemente haben. Zur Beschreibung des Pr/T-Netzes werden alle netztypunabhängigen Elemente entsprechend den PNML-Vorgaben verwendet. Der Netzknoten `<place>` bekommt zusätzlich zu den Kindelementen `<graphics>`, `<name>` und `<initialMarking>` den Kindknoten `<relation>`. Das Element `<tuple>` besteht aus den beiden Kindelementen `<attribute>` und `<value>`, die die Markierung der Stelle beschreiben[28]. Quelltext 13 zeigt die Pr/TML-Umsetzung der Stelle aus Abbildung 27.

Quelltext 13: Umsetzung der Stelle aus Abbildung 27 in Pr/TML-Syntax.

```
<place id="p1">
      <graphics>
            <position x="40" y="30"/>
      </graphics>
      <name>
            <text>Lieferung</text>
            <graphics>
                  <offset x="-10" y="-8"/>
            </graphics>
      </name>
      <initialMarking>
            <relation>
```

[28] Das Element *relation* hat keinen graphics-Kindnoten, da der Benutzer die Markierung der Stelle interaktiv mit Hilfe eines halbautomatischen Assistenzsystems (Wizard) im Editor eingibt.

```
                  <tuple>
                        <attribute id="a1">
                              <name>Kunde</name>
                              <value>
                                    <name>TMG AG</name>
                              </value>
                        </attribute>
                        <attribute id="a2">
                              <name>Artikel</name>
                              <value>
                                    <name>A1</name>
                              </value>
                        </attribute>
                        <attribute id="a4">
                              <name>Menge</name>
                              <value>
                                    <name>20000</name>
                              </value>
                        </attribute>
                  </tuple>
            </relation>
      </initialMarking>
</place>
```

Die Koordinaten der Stelle `Lieferung` mit dem Identifier p1 sind (20, 20) und der Name der Stelle hat einen Abstand von (-10, -8) zum Referenzpunkt der Stelle. Die initiale Markierung der Stelle wird durch die Attribute `Kunde`, `Artikel` und `Menge`, die entsprechende Identifikatoren haben, beschrieben.

Als Notation für mathematische Formeln, wie sie in der Transitionsinschrift benutzt werden, kann MathML [CIMP03] verwendet werden. MathML (Mathematical Markup Language) ist eine XML-basierte Empfehlung des W3C zur Darstellung von Elementen mit mathematischem Inhalt. Die semantische Sicht[29] von MathML hat die folgenden Tags[30]:

- *apply*: umschließendes Element für die auszuführenden Teile der Formel,
- *ci:* Beschreibung von Bezeichnern,
- *cn:* Beschreibung von numerischen Angaben,
- *not :* unärer Operator, der genau ein Kindelement haben kann,
- *minus:* das zweite Kindelement wird vom ersten Kindelement subtrahiert,
- *plus:* die nachfolgenden (beliebig vielen) Kindelemente werden addiert,
- *eq:* Definition einer Gleichung.

[29] In MathML wird auch eine präsentative Sicht definiert, die insbesondere für die Darstellung mathematischer Formeln in HTML verwendet wird.
[30] Es werden nur Tags erklärt, die auch im weiteren Verlauf des Kapitels benötigt werden.

Dabei sind Operatoren <minus/>, <plus/> oder <eq/> Elemente, die kein Ende-Tag haben und inhaltsleer definiert werden. Die Umsetzung der Transition *empfangen* in PrTML zeigt Quelltext 14.

Quelltext 14: PrTML-Syntax für die Transition aus Abbildung 27

```
<transition id="t1">
      <graphics>
            <position x="78" y="30"/>
      </graphics>
      <name>
            <text>empfangen</text>
            <graphics>
                  <offset x="0" y="10"/>
            </graphics>
      </name>
      <operation>
            <apply>
                  <eq/>
                  <apply>
                        <plus/>
                        <ci>m</ci>
                        <cn>1</cn>
                  </apply>
                 <times/>
                 <ci>m</ci>
                 <cn>1</cn>
            </apply>
            <graphics>
                  <offset x="-5" y="-2"/>
            </graphics>
      </operation>
</transition>
```

Die Transition *empfangen* wird bei den Koordinaten (78, 30) modelliert. Die Inschrift der Transition wird entsprechend den MathML-Vorgaben definiert. Bei der Inschrift m+1=m1 handelt es sich um einen logischen Ausdruck, dessen Inschrift um das `<operation>`-Element umschlossen wird.

Kanten werden wie bei PNML mit dem XML-Element `<arc>` und den Attributen id, source und target definiert. Die Inschrift der Kanten wird über das `<inscription>`-Element mit einem entsprechenden `<graphics>`-Element beschrieben.

Quelltext 15: PrTML-Syntax für die Kante von *Lieferung* nach *empfangen*

```
<arc id="a1" source="p1" target="t1">
      <graphics>
            <position x="42" y="32"/>
            <position x="65" y="32"/>
      </graphics>
      <inscription>
            <text>k,a,m</text>
```

```
            <graphics>
                  <offset x="5" y="-2"/>
            </graphics>
      </inscription>
</arc>
```

Analog zur Löschkante wird die Einfügekante in PrTML umgesetzt.

Quelltext 16: PrTML-Syntax für die Kante von *empfangen* nach *Lieferung*

```
<arc id="a2" source="t1" target="p1">
      <graphics>
            <position x="62" y="29"/>
            <position x="42" y="29"/>
      </graphics>
      <inscription>
            <text>k,a,m1</text>
            <graphics>
                  <offset x="-2" y="-2"/>
            </graphics>
      </inscription>
</arc>
```

Im nächsten Kapitel wird ein OWL-basiertes Beschreibungsformat für elementare und höhere Petri-Netze vorgestellt. Während das XML-basierte Austauschformat als Speicherformat für die werkzeuggestützte Realisierung einer Modellierungsunterstützung für Geschäftsprozesse genutzt werden kann, eignet sich das OWL-basierte Beschreibungsformat zur (semi-) automatischen Verarbeitung von Petri-Netz-Modellen und der Unterstützung von Inferenzen.

3.2 OWL-basiertes Beschreibungsformat für Petri-Netze

Angelehnt an die Vorgehensweise für die Erstellung von Ontologien aus Abschnitt 2.1.1. kann ein OWL-basiertes Beschreibungsformat wie folgt erstellt werden:

- Definition von Konzepten und Eigenschaften der Anwendungsdomäne (Abschnitt 3.2.1.)
- Einordnung der Konzepte und Eigenschaften in eine Konzept- bzw. Eigenschaftshierarchie (Abschnitt 3.2.2.)
- Erweiterung der Ontologie um Instanzen aus der Anwendungsdomäne (Abschnitt 3.2.3.)

In Kapitel 2 wurden bereits die Begriffe Ontologien, Geschäftsprozesse und Geschäftsprozessmodelle eingeführt. Nachfolgend wird die OWL-basierte Beschreibung von Geschäftsprozessmodellen als semantische Geschäftsprozessmodelle verstanden und wie folgt definiert:

Definition 10: Semantische Geschäftsprozessmodelle

Ein semantisches Geschäftsprozessmodell beschreibt ein Geschäftsprozessmodell in einer präzisen maschinenlesbaren und -interpretierbaren Notation, die gezielte (semi-) automatische Manipulationen des Geschäftsprozessmodells durch darauf operierende Funktionen ermöglicht. Ein Format ist maschinenlesbar, wenn Rechner die Struktur der Dokumente erkennen können. Bei einem maschineninterpretierbaren Format können Rechner die Struktur „verstehen" und es können Folgerungen aus dem Inhalt eines Dokuments geschlossen werden.

Semantische Geschäftsprozessmodelle (die Grundlage für die Realisierung einer Modellierungsunterstützung für Geschäftsprozesse bilden) können für Petri-Netze beschrieben werden, die:

(i) einen definierten Anfang haben und

(ii) nicht stark zusammenhängend sind.

Definition 11: Zusammenhängend / Stark zusammenhängend

Ein Petri-Netz $N=(S, T, F)$ ist *zusammenhängend*, wenn keine Zerlegung der Knotenmengen in X_1 und X_2 existiert, so dass gilt:

$$X_1,\ X_2 \neq \emptyset$$
$$X_1 \cup X_2 = S \cup T$$
$$X_1 \cap X_2 = \emptyset$$
$$F \subseteq (X_1 \times X_1) \cup (X_2 \times X_2)$$

$N=(S, T, F)$ heißt *stark zusammenhängend*, wenn für zwei Elemente $x, y \in \{S, T\}$ mit $x \neq y$ eine Sequenz $(z_1, z_2),(z_2, z_3),\ldots(z_{n-1}, z_n) \in F$ mit $(n \geq 2)$ existiert, so dass $x = z_1$ und $y = z_n$.

Um aus einem semantischen Geschäftsprozessmodell[31] (semi-) automatisch die ursprüngliche operationale Semantik eines Petri-Netzes wiederherzustellen, darf das Petri-Netz nicht stark zusammenhängend sein. Aus der Definition 11 kann nur entnommen werden, dass $x=z_1$ und $y=z_n$, allerdings nicht ob x vor y modelliert wird. Bei der Verbindung von stark zusammenhängenden semantischen Geschäftsprozessmodellen hätte das System Probleme passende Schnittstellen zu finden, an denen die Prozesse verbunden werden könnten. Um diese Einschränkung für ein

[31] Im weitern Verlauf der Arbeit werden nur semantische Geschäftsprozessmodelle für Petri-Netz-basierte Geschäftsprozessmodelle betrachtet.

semantisches Geschäftsprozessmodell zu vernachlässigen, könnte ein Konzept eingefügt werden, das das Startelement eines Petri-Netzes festlegt. Allerdings wird die Eigenschaft (stark zusammenhängend) nach Verbindung eines editierten Geschäftsprozesses und eines stark zusammenhängenden Folgeprozesses nicht mehr erfüllt sein. Damit die Eigenschaft trotzdem erfüllt werden kann, werden weitere (formale) Kriterien vorausgesetzt. Grundsätzlich fraglich ist, ob überhaupt stark zusammenhängende Geschäftsprozesse in realen Ausschnitten modelliert werden.

3.2.1 Definition von Konzepten und Eigenschaften

Laut Definition 6 ist ein Petri-Netz ein Tripel $N = (S, T, F)$. Aus der Flussrelation $F \subseteq (S \times T) \cup (T \times S)$ folgen die Bipartitheit des Netzgraphen sowie die Richtung der Kanten. Es werden die zwei disjunkten Teilmengen von Kanten unterschieden: $F_r \subseteq S \times T$ und $F_w \subseteq T \times S$.

Daraus ergeben sich als netztypunabhängige Konzepte der Ontologie für Petri-Netze:

- *PetriNet* für alle möglichen Petri-Netze *N = (S, T, F)*,
- *Node* für alle Knoten (S und T)*,*
- *Arc* für alle Kanten aus *F,*
- *Place* für alle Stellen aus *S*,
- *Transition für alle Transitionen aus T,*
- *FromPlace für alle Kanten aus* F_r*,*
- *ToPlace* für alle Kanten aus F_w.

In Definition 6 wird ausgedrückt, dass ein Petri-Netz *N* aus einer Knotenmenge $S \cup T$ und einer Kantenmenge *F* besteht*.* Dieser Zusammenhang wird in der Ontologie für Petri-Netze ausgedrückt, indem dem Konzept *PetriNet* Objekteigenschaften zugeordnet werden:

- *hasNode* für die Domäne *PetriNet* und die Wertebereiche *Place* und *Transition*,
- *hasArc* für die Domäne *PetriNet* und die Wertebereiche *FromPlace* und *ToPlace*.

Nachdem laut Definition 6 $S \cup T \neq \varnothing$ gilt, besitzt *hasNode* die Kardinalität [1..*], da ein Petri-Netz aus mindestens einem Knoten (Stelle oder Transition) besteht. Eine

Kante kann erst definiert werden, wenn mindestens zwei unterschiedliche Knoten bestehen; *hasArc* hat deswegen die Kardinalität [*].
Um auszudrücken, dass Kanten eine Quelle (source) und ein Ziel (target) haben, erben die Konzepte *FromPlace* und *ToPlace* vom Konzept *Arc* die Objekteigenschaft *hasNode*, wobei *hasNode*.*FromPlace* als Wertebereich *Place* und *hasNode*.*ToPlace* als Wertebereich *Transition* hat.
Abhängig vom Petri-Netz-Typ beinhalten Stellen Marken[32]. Bei elementaren Petri-Netzen sind die Marken noch anonym. In der Ontologie werden Stellen Markierungen über die Objekteigenschaft

- *hasMarking* für die Domäne *Place*

zugewiesen.

Petri-Netze folgen einer operationalen Semantik; das bedeutet, dass die Stelle s1 zeitlich vor der Stelle s2 oder s3 „erfüllt" wird (wenn zwischen s1 und s2 die Transition t1 existiert). Das Gleiche gilt für Transitionen. Kanten verlaufen immer zwischen Stellen und Transitionen (bzw. zwischen Transitionen und Stellen); es gilt $\bullet s \cup s\bullet \subseteq T$ bzw. $\bullet t \cap t\bullet \subseteq S$. Dieser Zusammenhang wird in der Ontologie für Petri-Netze ausgedrückt, indem Stellen und Transitionen Objekteigenschaften zugeordnet werden, die als Wertebereich den entsprechenden nachgelagerten Knotentyp (Transitionen bzw. Stellen) haben. Auf diese Eigenschaften wird insbesondere bei der Ähnlichkeitsberechnung im Kapitel 4 – genauer bei der Berechnung der strukturellen Ähnlichkeit von zwei Prozesselementnamen – zurückgegriffen:

- *placeRef* für die Domäne *Transition* und den Wertebereich *Place*,
- *transRef* für die Domäne *Place* und den Wertebereich *Transition*.

Um ein striktes Pr/T-Netz in OWL auszudrücken, erweitern wir die oben definierte Ontologie um netztypabhängige Konzepte und Eigenschaften:

- *LogicalConcept* für alle Transitionsinschriften aus *TI*,
- *IndividualDataItem* für alle Relationen R_s,
- *Attribute* für alle Attribute A_s^i,
- *Value* für Teilmengen aus $D_s^1 \times .... \times D_s^n$.

Dem Konzept *Transition* wird die Objekteigenschaft

[32] In S/T-Netzen weist *K* jeder Stelle eine Anzahl von Marken zu, die nicht überschritten werden darf. Diese Eigenschaft könnte in der Ontologie ausgedrückt werden, indem eine Datentypeigenschaft *hasCapacity* mit dem Wertebereich *xsd:nonNegativeInteger* eingefügt wird.

- *hasLogicalConcept* mit der Domäne *Transition* und dem Wertebereich *LogicalConcept*

zugewiesen.

Laut Definition 7 wird eine Kante $f \in F$ eines Pr/T-Netzes als Menge von Variablentupeln beschriftet. Die Kantenbeschriftung entspricht den Konzepten:

- *Delete* für die Inschriften von F_r und
- *Insert* für die Inschriften von F_w.

Die Konzepte *FromPlace* und *ToPlace* werden durch die Objekteigenschaft

- *hasInscription* mit der Domäne *FromPlace bzw. ToPlace* und dem Wertebereich *Delete bzw. Insert*

erweitert.

Die gleiche Stelligkeit der Lösch- bzw. Einfügekanten wie das Prädikat werden in der Ontologie durch die Objekteigenschaft

- *hasAttribute* mit der Domäne *Delete* bzw. *Insert* und dem Wertebereich *IndividualDataItem*

modelliert.

Für die Klassen *LogicalConcept* und *IndividualDataItem* wird die Objekteigenschaft

- *hasAttribute* mit der Domäne *LogicalConcept* bzw*. IndividualDataItem* und dem Wertebereich *Attribute*

zugewiesen, und für das Konzept *Attribute* wird die Objekteigenschaft

- *hasValue* mit dem Wertebereich *Value*

definiert. Für die Klasse *Value* gibt es die folgende Objekteigenschaft:

- *hasRef* mit der Domäne und dem Wertebereich *Value*.

Die Annotation einer Transition $t \in T$ eines Pr/T-Netzes setzt sich aus Schaltbedingungen und Operationen zusammen, die in einer prädikatenlogischen Formel vereinigt werden. Bedingungen und Operationen werden in separaten Ontologiekonzepten definiert. Die Schaltbedingung Con_t kann durch freie Variablen, die im Variablentupel der adjazenten Kanten enthalten sein müssen, durch gebundene Variablen sowie durch auf diesen Variablen definierte Prädikate beschrieben werden. In der Schaltoperation Op_t können Funktionen definiert werden, die auf Variablentupeln der schreibenden Kanten definiert sind und diesen einen Wert aus einer

entsprechenden Domäne zuweisen. In der Ontologie werden Bedingungen und Operationen durch

- *Condition* für die Schaltbedingungen Con_t aller t∈ Tund
- *Operation* für die Operationen Op_t aller t∈ T

modelliert. Diesen Konzepten werden die Objekteigenschaften wie folgt zugewiesen:

- *hasOperation* mit der Domäne *LogicalConcept* und dem Wertebereich *Operation*,
- *hasCondition* mit der Domäne *LogicalConcept* und dem Wertebereich *Condition*.

Um eine prädikatenlogische Formel in der Pr/T-Netz-Ontologie als Schaltbedingung einer Transition darstellen zu können, muss diese in Pränex-Normalform[33] vorliegen. Besitzt die Formel des Weiteren eine Matrix in KNF[34], so kann diese durch die folgenden drei charakteristischen Datentypeigenschaften dargestellt werden:

- *forall*, *exists* und *and* mit der Domäne *Condition* und dem Wertebereich `xsd:string`.

Die Darstellung einer Operation als Menge beliebig vieler Funktionen erfolgt über die Datentypeigenschaft:

- *function* mit der Domäne *Operation* und dem Wertebereich `xsd:string`.

3.2.2 Konzeption einer Ontologie für Petri-Netze

Die im vorhergehenden Abschnitt definierten Konzepte und Eigenschaften wurden nur informell dargestellt. Die Zusammenhänge zwischen den Konzepten und den Eigenschaften werden entsprechend Definition 1 in einer Ontologie formuliert. Zunächst werden noch die wichtigsten Elemente von OWL in formaler Syntax eingeführt, die im weiteren Verlauf dieses Abschnittes benötigt werden.

[33] Eine Aussage in der Prädikatenlogik erster Stufe liegt in der Pränex-Normalform vor, wenn alle Quantoren außen bzw. links vor der eigentlichen Formel stehen. Eine Formel liegt in Pränex-Normalform vor, wenn sie von der Form $Q_1 y_1,...,Q_n y_n L$ ist, mit $n \geq 0$, $Q_1,...,Q_n \in \{\exists, \forall\}$ und L eine quantorenfreie Formel ist. $Q_1 y_1,...,Q_n y_n$ heißt Pränex und L ist die Matrix [HeWe90].

[34] Eine Formel (in der Aussagenlogik) liegt in konjunktiver Normalform (KNF) vor, wenn sie eine Konjunktion von Disjunktionstermen ist, d. h. sie ist von der Form $\bigwedge_i \bigvee_j (\neg) x_{ij}$ [HeWe90].

Tabelle 4: Formale Syntax von OWL DL

Element	**DL Syntax**
intersectionOf	$A_1 \sqcap \ldots \sqcap A_2$
unionOf	$A_1 \sqcup \ldots \sqcup A_2$
allValuesFrom	$\forall$ P.A
someValuesFrom	$\exists$ P.A
maxCardinality	$\leq$ nP
minCardinality	$\geq$ nP

Eine Ontologie für elementare Petri-Netze ist ein 5-Tupel O:= (C,H_c,P,H_p,A) mit

1. C = {PetriNet, Node, Arc, Place, Transition, FromPlace, ToPlace}
2. H_c = {(Place, Node); (Transition, Node); (ToPlace, Arc); (FromPlace, Arc)}
3. $P_{hasNode}$ = {(PetriNet, Node)}, $P_{hasNode}$ = {(Arc, Place)};
   $P_{hasNode}$ = {(Arc, Transition)}, P_{hasArc} = {(PetriNet, Arc)},
   $P_{placeRef}$ = {(Transition, Place)}, $P_{transRef}$ = {(Place, Transition)}[35]
4. H_p = {$\varnothing$}
5. A_1 {PetriNet $\equiv \geq 1$hasNode.Node $\sqcap$ hasArc.Arc}
   A_2 {Arc $\equiv \exists$ hasNode.(Transition $\sqcup$ Place}
   A_3 {Transition $\equiv \geq 1$placeRef.Place}
   A_4 {Place $\equiv \geq 1$transRef.Transition}

Eine Ontologie für strikte Pr/T-Netze ergibt sich aus der Erweiterung der Ontologie für elementare Petri-Netze um typabhängige Konzepte, Eigenschaften und Axiome. Sie wird definiert durch:

1. C = {PetriNet, Node, Arc, Place, Transition, FromPlace, ToPlace, Delete, Insert, LogicalConcept, IndividualDataItem, Attribute, Value, Operation, Condition}

[35] Der Wertebereich der Objekteigenschaft *hasMarking* wird im nächsten Kapitel eingeführt. Die Objekteigenschaft wird im nächsten Abschnitt berücksichtigt.

2. H_c = {(Place, Node); (Transition, Node); (ToPlace, Arc); (FromPlace, Arc)}

3. $P_{hasNode}$= {(PetriNet, Node)}, $P_{hasNode}$={(Arc, Place)};
$P_{hasNode}$= {(Arc, Transition)}, P_{hasArc}={(PetriNet, Arc)},
$P_{placeRef}$= {(Transition, Place)}, $P_{transRef}$={(Place, Transition)},
$P_{hasMarking}$= {(Place, IndividualDataItem)}, $P_{hasInscription}$={(FromPlace, Delete)},
$P_{hasInscription}$ ={(ToPlace, Insert)}, $P_{hasCondition}$={(LogicalConcept, Condition)},
$P_{hasCondition}$ ={(LogicalConcept, Operation)},
$P_{hasAttribute}$= {(LogicalConcept,Attribute)},
$P_{hasAttribute}$ ={(IndividualDataItem, Attribute)},
$P_{hasAttribute}$ ={(Delete, IndividualDataItem)},
$P_{hasAttribute}$ ={(Insert, IndividualDataItem)},
$P_{hasValue}$ ={(Attribute, Value)}, P_{hasRef}={(Value, Value)},
$P_{function}$ ={(Operation, xsd: string)}, P_{forall}={(Condition, xsd: string)},
P_{exists} ={(Condition, xsd: string)}, P_{and}={(Condition, xsd: string)}

4. H_p = {∅}

5. A_1 {PetriNet ≡ ≥1hasNode.Node ⊓ hasArc.Arc}
A_2 {Arc ≡ ∃hasNode.(Transition ⊔ Place}
A_3 {Transition ≡ ≥1placeRef.Place} ⊓
=1hasLogicalConcept.LogicalConcept}
A_4 {Place ≡ ≥1transRef.Transition} ⊓
=1hasMarking.IndividualDataItem}
A_5 {FromPlace ≡ ≥1hasInscription.Delete ⊓ ∃hasNode.Place}
A_6 {ToPlace ≡ ≥1hasInscription.Insert ⊓ ∃has-
A_7 Node.Transition}
{LogicalConcept ≡ (=1hasCondition.Condition ⊔
A_8 =1hasOperation.Operation) ⊓ ∃hasAttribute.IndividualDataItem}
A_9 {IndividualDataItem ≡ ≥1hasAttribute.Attribute}
A_{10} {Delete ≡ ∀hasAttribute.IndividualDataItem}
A_{11} {Insert ≡ ∃hasAttribute.IndividualDataItem}
A_{12} {Attribute ≡ ≥hasValue.Value}
A_{13} {Value ≡ hasRef.Value}
A_{14} {Condition ≡ forall(string) ⊔ exists(string) ⊔ and(string)}
{Operation ≡ function(string)}

Laut Definition 6 gilt $S \cap T = \varnothing$. Um auszudrücken, dass Instanzen des Place-Konzepts nicht gleichzeitig Instanzen des Transition-Konzepts sein können (Disjunktion), wird in der elementaren und in der Pr/T-Netz-Ontologie eine entsprechende Einschränkung mit owl:disjointWith eingeführt:

$$A_0 \{ \text{Place} \cap \text{Transition} = \varnothing \}$$

Die abgebildeten Ontologiekonzepte sind direkt oder indirekt vom allgemeinsten OWL-Konzept *owl:Thing* abgeleitet. Dieser Sachverhalt muss allerdings nicht in der Ontologie explizit dargestellt werden, da alle Konzepte vom Typ *owl:Class* von *owl:Thing* abstammen.

Alle Elemente der elementaren und der Pr/T-Netz-Ontologie können entweder in der SHOIN(D)[36]-Beschreibungslogik oder in Form von XML/RDF-Syntax ausgedrückt werden. OWL DL ist eine syntaktische Variante der SHOIN(D)-Beschreibungslogik, die trotz ihres hohen Grads an Ausdrucksmächtigkeit entscheidbar ist [Horr05]. In Schlussfolgerungssystemen wie Racer[37] und FaCT++[38] wird die Beschreibungslogik SHIQ(D) [HaST99] verwendet. Sie unterscheidet sich von SHOIN(D) vor allem dadurch, dass keine Nominale unterstützt werden[39]. Um eine skalierte Entscheidbarkeit in SHIQ(D) zu erreichen, wird OWL DL um Regeln, die Beschreibungslogik-basiert sind, erweitert [HuMS04]. Abfragen für diese Erweiterung von SHIQ(D) mit Beschreibungslogik-basierten Regeln werden durch einen Algorithmus erreicht, der SHIQ(D) zu disjunktiven funktionsfreien logischen Programmen (Datalog-Programmen[40]), welche wiederum auf einem Teil der Logik erster Ordnung aufgebaut sind, reduziert.
Quelltext 17 zeigt einen Ausschnitt[41] der Pr/T-Netz-Ontologie in XML/RDF-Syntax. Alle Ontologiekonzepte werden als owl:Class definiert. Die Disjunktion von Place und Transition wird durch owl:disjointWith beschrieben. Die Verwandtschaftsbeziehungen werden durch rdfs:subClassOf definiert. Alle Eigenschaften eines Konzepts werden als owl:ObjectProperty (bzw. owl:DatatypeProperty) definiert. Die Domänen und der Wertebereich einer Eigenschaft werden innerhalb der Konzeptdefinition durch die Merkmale rdfs:domain und rdfs:range festgelegt. Die Zugehörigkeit mehrerer Konzepte zu beispielsweise demselben Wertebereich wird durch owl:unionOf erreicht.

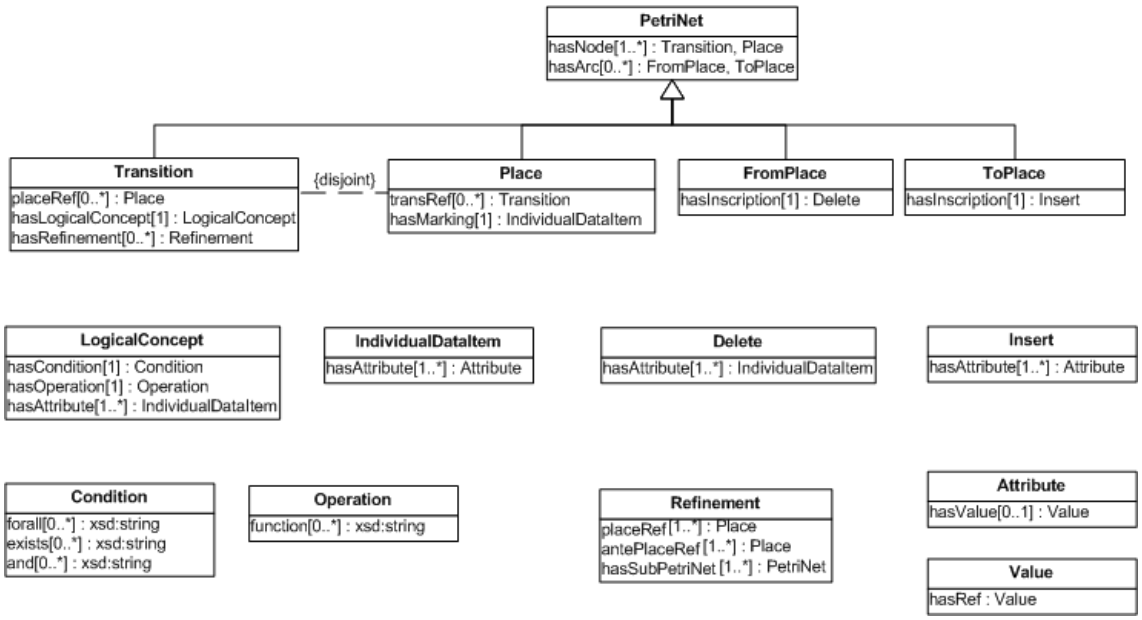
[36] *O* steht für nominals, *N* für unqualified number restrictions und (*D*) für datatypes
[37] http://www.racer-systems.com/products/tools/protege.phtml
[38] http://owl.man.ac.uk/factplusplus/
[39] Nominale in der OWL DL-Syntax werden durch `owl:oneOf` oder `owl:hasValue` ausgedrückt.
[40] Datalog ist eine Datenbank-Programmiersprache, die Prolog syntaktisch und semantisch ähnelt.
[41] Die vollständige Pr/T-Netz-Ontologie findet sich im Anhang A.

Quelltext 17: Ausschnitt der Pr/T-Netz-Ontologie in XML/RDF-Syntax

```
<rdf:RDF...
 <owl:Class rdf:about="#Transition">
  <owl:disjointWith rdf:resource="#Place"/>
    <rdfs:subClassOf rdf:resource="#PetriNet"/>
 </owl:Class>
 <owl:ObjectProperty rdf:about="#hasNode">
   <rdfs:domain rdf:resource="#PetriNet"/>
   <rdfs:range>
     <owl:Class>
       <owl:unionOf rdf:parseType="Collection">
         <owl:Class rdf:about="#Transition"/>
         <owl:Class rdf:about="#Place"/>
       </owl:unionOf>
     </owl:Class>
   </rdfs:range>
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:about="#hasCondition">
    <rdfs:domain rdf:resource="#LogicalConcept"/>
    <rdfs:range rdf:resource="#Condition"/>
 </owl:ObjectProperty>
</rdf:RDF>
```

Alle Konzepte und Eigenschaften der Pr/T-Netz-Ontologie zeigt Abbildung 28 in einem UML-Klassendiagramm. Konzepte bilden den Namen der Klasse und Eigenschaften sind als Attribute der Klasse modelliert. Dort wo keine Verbindung zwischen Klassen modelliert wurde, besteht eine Aggregationsverbindung. Die Verbindung mit einer Raute am Ende stellt eine Generalisierung dar.

Abbildung 28: Graphische Darstellung der Pr/T-Netz-Ontologie



3.2.3 Instanzmodellierung

Der Verweis von Instanzen auf Konzepte und Eigenschaften der Pr/T-Netz-Ontologie kann durch einen Namensraumpräfix realisiert werden. In Quelltext 18 wird der Namensraumpräfix petri: deklariert. Bei der Benennung (bzw. Instanziierung) von Kanten werden die beiden verschiedenen Knoten (Place und Transition) immer mit einem doppelten Unterstrich (__) voneinander getrennt (Lieferung__empfangen und empfangen__Lieferung). Besteht der Name eines Knotens aus mehreren Wörtern, dann werden diese nur mit einem einfachen Unterstrich getrennt (z.B. Bestätigung_verschickt__Brief_weiterleiten). Konzepte, Eigenschaften und Instanzen müssen in OWL immer eine eindeutige URI haben; deswegen wird die Inschrift von Löschkanten von der Inschrift von Einfügekanten durch einen zusätzlichen Unterstrich unterschieden (k_a_m *vs*. k_a_m_). Die Markierung einer Stelle wird mit R_ und dann dem Namen der Stelle eingeleitet (R_Lieferung). Bedingungen und Operationen in Transitionsschriften werden mit Con_ bzw. Op_ und dem Namen der Transition definiert (z.B. Con_empfangen). Die Semantik von Instanzen von semantischen Geschäftsprozessmodellen ist für unser Anwendungsszenario eindeutig durch die Namen der Stellen und Transitionen gegeben.

Quelltext 18: Instanzmodellierung für das Pr/T-Netz in Abbildung 27 (Ausschnitt)

```
<rdf:RDF...
  <petri:FromPlace rdf:about="#Lieferung__empfangen">
    <petri:hasInscription rdf:resource="#k_a_m"/>
    <petri:hasNode rdf:resource="#Lieferung"/>
  </petri:FromPlace>
  <petri:ToPlace rdf:ID="empfangen__Lieferung">
    <petri:hasInscription rdf:resource="#k_a_m1_"/>
    <petri:hasNode rdf:resource="#empfangen"/>
  </petri:ToPlace>
  <petri:Transition rdf:ID="empfangen">
    <petri:placeRef>
      <petri:Place rdf:ID="Lieferung">
        <petri:hasMarking>
          <petri:IndividualDataItem rdf:ID="R_Lieferung">
            <petri:hasAttribute rdf:resource="#Kunde"/>
            <petri:hasAttribute rdf:resource="#Artikel"/>
            <petri:hasAttribute rdf:resource="#Menge"/>
          </petri:IndividualDataItem>
        </petri:hasMarking>
      </petri:Place>
    </petri:placeRef>
  </petri:Transition>
</rdf:RDF>
```

3.2.4 Eigenschaften der Pr/T-Netz-Ontologie

Die vorgestellte Ontologie für Petri-Netze bzw. Pr/T-Netze erfüllt alle Anforderungen, wie sie an eine Modellierungssprache für Ontologien im Kapitel 2.1. formuliert wurden.

- *Ausdrucksmächtigkeit*: die Ausdrucksmächtigkeit der Ontologie ist auf die Bildung von Klassen/Subklassen sowie Eigenschaften (Objekt- und Datentypeigenschaften) mit einem Wertebereich und einer Domäne beschränkt. Zusätzlich können binäre Eigenschaften und Instanzen von Klassen gebildet werden. Die Ontologie kann auf beliebige Konstrukte (siehe auch Kapitel 6.2.1) erweitert werden. Die Modellierung von zeitlichen Aspekten war nicht erforderlich, weshalb solche Konstrukte nicht in der Ontologie nicht verwendet werden.
- *Formalisierungsgrad*: die Konstrukte der Pr/T-Netz-Ontologie wurden in einer präzisen und formalen Notation formuliert.
- *Visualisierungsmöglichkeiten*: Eine (benutzerfreundliche) graphische Oberfläche zur Veränderung oder Erweiterung der Ontologie bietet das Werkzeug Protégé[42] von der Stanford University.
- *Inferenzmöglichkeiten*: Der Einsatz von Inferenzmechanismen ist ebenfalls über Protégé oder die KAON API möglich (siehe Kapitel 8.1).

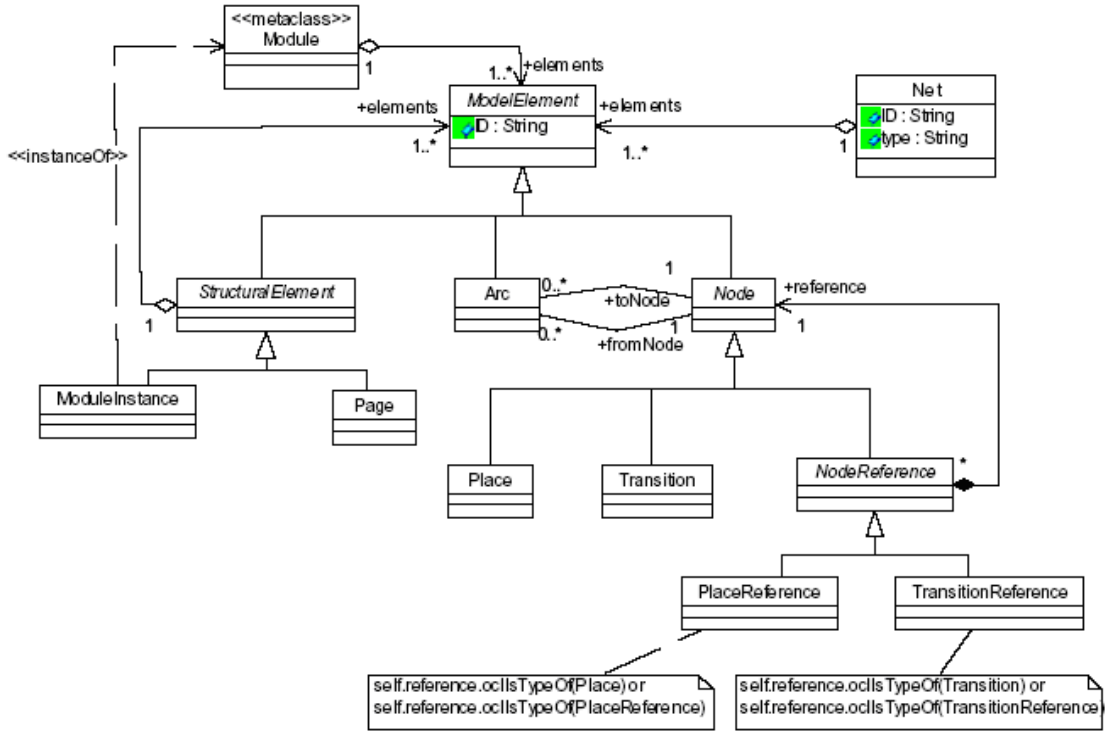
3.3 Untersuchung bestehender Ansätze zur semantischen Beschreibung von Geschäftsprozessmodellen

In der Literatur finden sich einige Ansätze, die Ontologien und Web-Services kombinieren, um eine automatische Verarbeitung - beispielsweise Komposition - von Web-Services zu unterstützen. Wie in Kapitel 2.2.3 skizziert, könnte die Zusammenfügung von Web-Services zur Komposition von Web-Services zu einem höherwertigen Dienst als Geschäftsprozessmodelle verstanden werden.
In diesem Abschnitt werden zunächst zwei Ansätze für semantische Beschreibungen von Petri-Netzen vorgestellt. Anschließend wird die OWL-basierte Web-Service-Ontologie OWL-S und die W3C Member Submission der so genannten Web Service Modeling Ontology (WSMO) beschrieben, um eine Abgrenzung zwischen Ansätzen zur semantischen Beschreibung von Petri-Netzen und Web-Services zu verdeutlichen.

[42] http://protege.stanford.edu/

Gasevic, Jovanovic und Devedzic [GaJD05] haben eine Ontologie für Petri-Netze konzipiert, um die Wiederverwendung von Petri-Netzen im semantischen Web zu unterstützen. Die konzipierte Ontologie ist kompatibel zu PNML; es werden in der Petri-Netz-Ontologie graphische Informationen und Koordinaten der Elemente angegeben. Die Darstellung der Hauptelemente dieser Petri-Netz-Ontologie erfolgt in UML-Notation, in der Axiome der Ontologie nicht vollständig definiert werden können. Der Fokus dieses Ansatzes ist nicht, die Vorteile von Ontologien auszunutzen, indem die Semantik von Petri-Netz-basierten Geschäftsprozessen interpretiert wird, sondern ein Transformationsskript von PNML nach OWL anzubieten. Die Hauptelemente der Petri-Netz-Ontologie nach [GaJD05] sind in Abbildung 29 veranschaulicht. Im Unterschied zu der in dieser Arbeit vorgestellten Ontologie für Petri-Netze verwenden [GaJD05] alle Elemente von PNML in ihrer Ontologie. Ontologien (z.B. modelliert in OWL) dienen zur Wissensrepräsentation und modellieren im Vergleich zu Petri-Netzen jedoch eher statische Aspekte (bekannte Tatsachen). Deswegen besteht die Ontologie für Petri-Netze, wie sie im vorhergehenden Kapitel erklärt wurde, nur aus statischen Elementen.

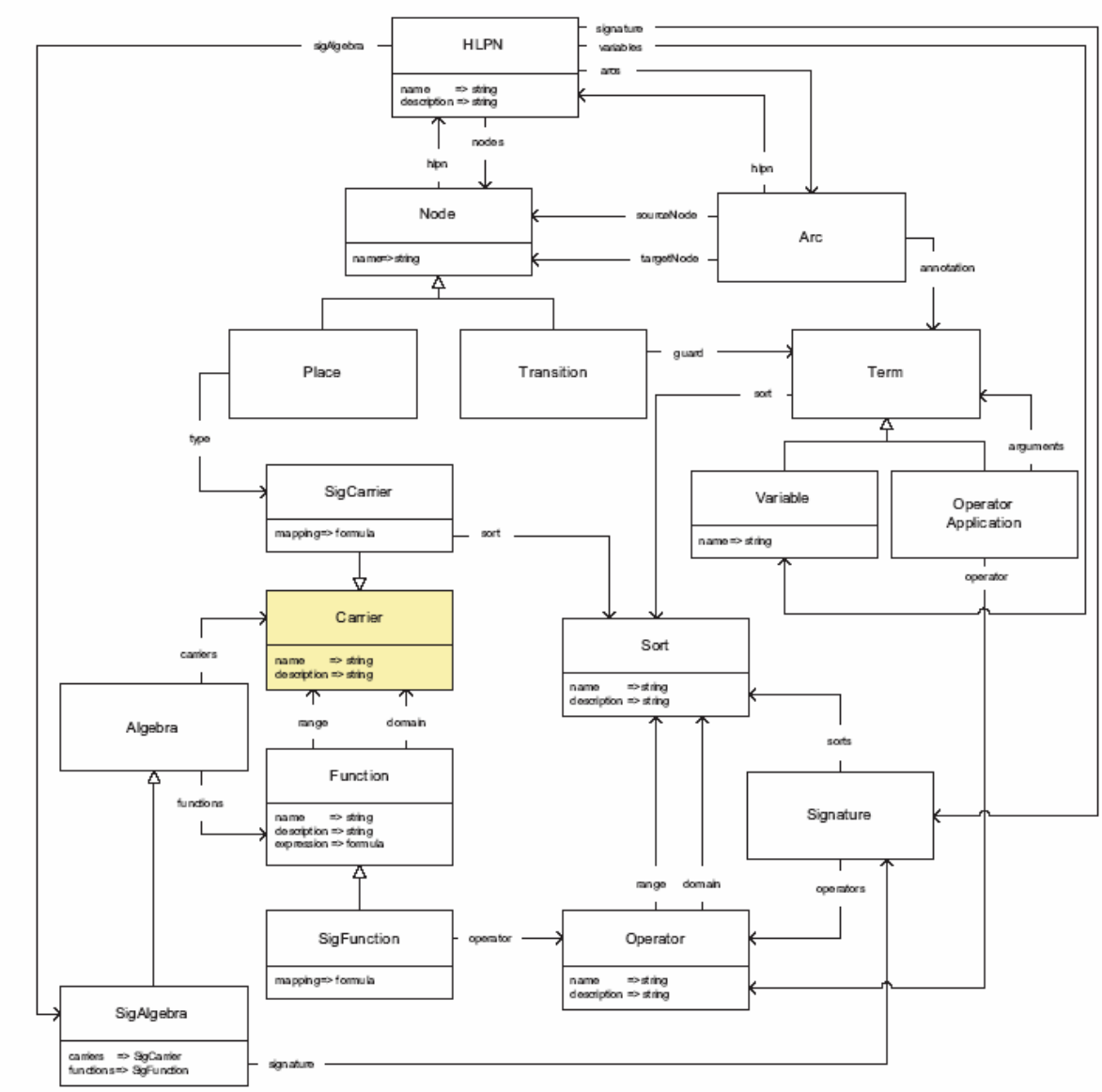
Abbildung 29: Hauptelemente der Petri-Netz-Ontologie nach [GaJD05]



Die High-level-Petri-Net-Ontologie (HLPN-Ontologie) nach [ViLB06] unterstützt eine semantische Verifikation von PNML-Modellen. In PNML wird nur vorgegeben, dass eine Kante eine Quelle (source) und ein Ziel (target) haben muss, die vom Typ

IDREF sein müssen. Mit Hilfe der HLPN-Ontologie wird überprüft, ob die Quelle wirklich eine Stelle (bzw. Transition) und die Senke wirklich eine Transition (bzw. Stelle) ist. Die Verifikation erfordert, dass die PNML-Modelle zunächst in F-Logik übersetzt werden und dass dann mit einem Reasoner (z.B. FLORA2[43]) logische Schlussfolgerungen abgeleitet werden können. Der Veröffentlichung von [ViLB06] ist allerdings nicht zu entnehmen, ob die Unentscheidbarkeit der F-Logik Schwierigkeiten bei der Verifikation bereitet. Die Elemente der HLPN-Ontologie sind in Abbildung 30 dargestellt.

Abbildung 30: Statische Elemente der High-Level-Petri-Net-Ontologie nach [ViLB06]



[43] http://flora.sourceforge.net/

Im Gegensatz zu [GaJD05] werden in der HLPN-Ontologie keine graphischen Informationen benutzt. Im Vergleich zu der in dieser Arbeit vorgeschlagenen Ontologie für Petri-Netze und der HLPN-Ontologie ergibt sich eine unterschiedliche Zielsetzung: [ViLB06] unterstützen eine semantische Verifikation von PNML-Modellen und nicht eine gezielte Manipulation von Petri-Netz-basierten Geschäftsprozessmodellen durch darauf operierende Funktionen.

OWL-S

OWL-S (OWL for Services; ehemals DAML-S)[44] soll das automatische Auffinden, Aufrufen und Zusammensetzen von Diensten im Internet ermöglichen [OWLS04]. Im März 2006 wurde das Pre-Release der Version 1.2 vorgestellt. OWL-S ist eine auf OWL DL basierende Ontologie zur Beschreibung von Web-Diensten, insbesondere Web-Services.

In der OWL-S-Spezifikation finden sich Überschneidungen zur Web-Service Description Language (WSDL). WSDL stellt eine abstrakte Sicht auf einen Dienst dar, der von konkreten Netzwerk- und Nachrichtenprotokollen abstrahiert. Die Kernfunktionalität unterscheidet WSDL von OWL-S aber deutlich: in einer WSDL-Datei werden Informationen über einen Web-Service-Aufruf beschrieben; in OWL-S findet sich hauptsächlich die Beschreibung wieder, was ein Dienst anbieten soll. Es soll also Wissen über einen Web-Service beschrieben werden, der das Auffinden, den Aufruf und die Zusammensetzung von Web Services automatisch erleichtern soll. Dabei können die Input/Output-Parameter als DL-basierte Typen definiert werden. Die Informationen, die WSDL und OWL-S enthalten, können sich auch ergänzen [MBLP03]. In WSDL wird der Typ eines Parameters beispielsweise als xsd:string deklariert und OWL-S beschreibt den Typ als Nachname.

OWL-S bietet mehrere Klassen zur Beschreibung von Web-Services, wobei der Beschreibung drei Basisklassen zugrunde liegen:

1. **Service Profile** - Was bietet der Dienst an?

Sowohl der Anbieter als auch der Abnehmer haben die Möglichkeit, die Funktionalitäten eines Dienstes zu spezifizieren, den sie anbieten oder anfordern wollen (provider advertising bzw. consumer request).

Mit dem Service Profile können Anfragen mit den Spezifikationen abgeglichen und passende Dienste vorgeschlagen werden. Das Service Profile beschreibt die folgenden Informationen:

- Informationen über den Anbieter des Dienstes,

[44] http://www.ai.sri.com/daml/services/owl-s/1.2/

- funktionelle Informationen über die Ein- und Ausgabeparameter und die Vor- und Nachbedingungen, die für diesen Dienst erfüllt werden müssen,
- zusätzliche Eigenschaften wie Klassifizierung (z.B. nach UNSPSC)[45] oder Qualitätsinformationen. Anhand dieser Informationen kann z.B. ein Agent einen passenden Dienst finden. Zwei weitere Klassen beinhalten Informationen, die das Aufrufen des Dienstes durch den Agenten ermöglichen sollen.

2. **Service Model** - Wie arbeitet ein Dienst?
Ein Dienst kann entweder für gegebenen Input bestimmte Informationen liefern oder auch bestimmte Veränderungen verursachen. Dies wird durch die Vorbedingungen und Auswirkungen des Prozesses im Service Model dokumentiert. Das Aufrufen eines Dienstes, die Komposition von Services und das Monitoring werden durch das Service Model erleichtert. Beispielsweise beschreibt ein Service Profile von Amazon[46], dass Bücher über diesen Dienst bestellt werden können. Um allerdings ein Buch zu bestellen, muss der Benutzer den Titel des Buches und Zahlungsangaben machen; diese Informationen sind im Service Model hinterlegt. Zu einer Benutzeranfrage können die Service Profiles von mehreren Diensten passen. Der Requester kann sich anhand der Service-Model-Beschreibungen den am besten passenden Dienst aussuchen.

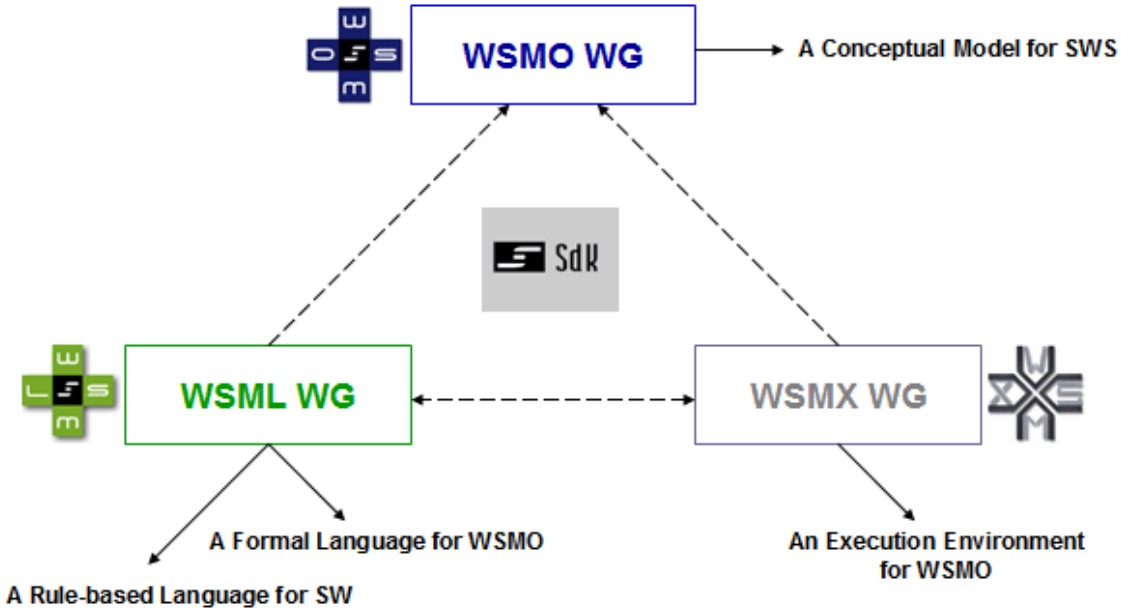
3. **Service Grounding** - Wie kann man den Dienst ansprechen?
Im Service Grounding werden technische Informationen des Service-Aufrufs wie Protokolle, Nachrichtenformate und Adressierung spezifiziert.

WSMO

Als Gegenstück zur OWL-S wurde WSMO (Web Service Modeling Ontology) [BDFL05] vorgeschlagen. Die WSMO-Spezifikation wurde innerhalb der SDK Cluster Arbeitsgruppe entwickelt, zu der auch WSML (Web Service Modeling Language) und WSMX (Web Service Execution Language) zählen. WSMO bietet ein konzeptionelles Modell für Semantische Web-Services (SWS) [DoJK05] und stellt die abstrakten Konzepte mittels Ontologien bereit. Zur Beschreibung der SWS in WSMO bietet WSML eine auf F-Logik aufbauende regelbasierte und formale Sprache. WSMX bildet die Ausführungsumgebung für die WSMO-Services. Abbildung 31 veranschaulicht die Zusammenhänge zwischen den Spezifikationen [RLKB06].

[45] siehe United Nations Standard Products and Services Code
[46] http://www.amazon.com

Abbildung 31: Drei Spezifikationen der SDK Cluster Arbeitsgruppe



Im Gegensatz zu OWL-S gibt es in WSMO eine klare Trennung zwischen den einzelnen Anwendungen, denn diese sollen so unabhängig wie möglich sein. Jede Komponente wird für sich isoliert definiert. Zudem ist WSMO konform mit der URI-Spezifikation und unterstützt XML-Serialisierungen. Zu den Hauptelementen zählen:

- **Ontologies**: bieten ein präzises Vokabular für alle Komponenten. Die wesentlichen Elemente der WSMO Ontologies sind *Concepts*, *Attributes*, *Instances*, *Relations*, *Functions* und *Axioms*.
- **Goals**: beschreiben die Zielsetzung eines Clients zum Aufruf eines Web-Services. Es wird die Funktionalität und das Kommunikationsverhalten des Web-Service definiert, die zur Zielerreichung notwendig sind.
- **Mediators**: werden als Intermediäre zwischen Komponenten verwendet; es werden vier verschiedene Mediators unterschieden:
  a) *OO Mediators* verknüpfen Ontologien mit Ontologien,
  b) *GG Mediators* verknüpfen Ziele mit Zielen,
  c) *WG Mediators* verknüpfen Web-Services mit Zielen (diese Vermittler sind für die Dienstauffindung von elementarer Bedeutung),
  d) *WW Mediators* verknüpfen Web-Services mit Web-Services
- **Web Services**: umfassen die semantische Beschreibung des Web-Services (Ontologies, Mediators, Capabilities, Interfaces). Dabei werden über Capability die Funktionalität des Web-Services und über Interfaces die Schnittstellen definiert.

Quelltext 19 zeigt die Beschreibung eines Web-Services zur Kreditkartenabrechnung, der mit WSMO[47] modelliert wurde. Zunächst werden Namensräume deklariert, um Elemente, die in der Beschreibung verwendet werden, eindeutig zu benennen. Der Web-Service wird durch zahlreiche nicht funktionale Eigenschaften wie den Titel, das Datum oder den Typ des Web-Services beschrieben. Die Elemente dieses Web-Services wurden in der Ontologie #purchaseOntology modelliert.

Quelltext 19: Beschreibung eines Web-Services in WSMO

```
namespace {_"http://example.org/CreditCardCharging#",
 dc _"http://purl.org/dc/elements/1.1#",
 po _"http://example.org/purchaseOntology#",
 foaf _"http://xmlns.com/foaf/0.1/",
 wsml _"http://www.wsmo.org/wsml/wsml-syntax#",
 ccci _"http://www.example.org/CCChargingInterfaceOntology#"}

webService _"http://example.org/CreditCardChargingWebService"
  nonFunctionalProperties
    dc:title hasValue „Credit Card Charging Web Service"
    dc:creator hasValue „Association of all Credit Card Companies"
    dc:description hasValue "web service for charging a credit
       card with a given amount and creating a remittance order
       for a given recipient"
    dc:publisher hasValue " Association of all Credit Card
       Companies"
    dc:date hasValue "2006-01-12"
    dc:type hasValue <<http://www.wsmo.org/2004/d2/#webservice>>
    dc:format hasValue "text/html"
    version hasValue "$Revision: 1.5 $"
  endNonFunctionalProperties

importsOntology _"http://example.org/purchaseOntology"
```

An einigen Stellen ist WSMO noch nicht vollständig und bedarf an Forschungsarbeit [RoSc05].

Mit der Beschreibung von OWL-S und WSMO soll die unterschiedliche Zielsetzung dieser Spezifikationen mit der Petri-Netz-Ontologie, die im vorhergehenden Abschnitt vorgestellt wurde, deutlich werden. Web-Services können miteinander verbunden werden, wenn Schnittstellen zueinander passen. Dabei müssen technische als auch funktionale Informationen abgeglichen werden und evtl. manuelle Anpassungen vorgenommen werden. Bei der Ähnlichkeitsmessung von Web-Services werden die Services als Black Boxes betrachtet; die Ähnlichkeit wird nur anhand der Schnittstellen-Informationen beurteilt. Die Petri-Netz-Ontologie hingegen unter-

[47] http://www.wsmo.org/2004/d3/d3.3/v0.1/

stützt eine ganzheitliche (semi-) automatische Ähnlichkeitsmessung von Petri-Netzen (bzw. Geschäftsprozessen, die mit Petri-Netzen modelliert wurden) und die Verbindung von Petri-Netz-Ontologien ist nicht von technischen Informationen abhängig.

4 Der Ähnlichkeitsbegriff für semantische Geschäftsprozessmodelle

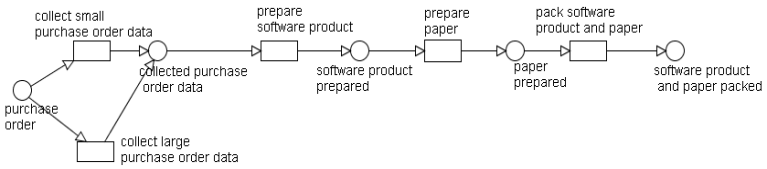
Bei der Modellierung von Geschäftsprozessmodellen wird in der Literatur ein „kontrolliertes" Vokabular vorausgesetzt, damit keine begrifflichen Missverständnisse bei der Verbindung von Prozessen entstehen können. Eine solche Einschränkung wird in diesem Kapitel aufgehoben. Zur Bestimmung der Ähnlichkeit zwischen Namen von Konzeptinstanz semantischer Geschäftsprozessmodelle werden die Ähnlichkeitsmaße: *syntaktische*, *linguistische*, *strukturelle*, *abstraktionsniveaubasierte, und kombinierte Ähnlichkeit* eingeführt [EhKO07, KoOb07b]. Als Ergebnis der Ähnlichkeitsberechnung resultiert ein numerischer Wert, der die entsprechende Ähnlichkeit zwischen zwei Namen angibt. Für das linguistische Ähnlichkeitsmaß wird Hintergrundwissen in Form von Ontologien eingesetzt, welches ebenfalls in diesem Kapitel erläutert wird. Zudem werden auch bestehende Ansätze aus der Literatur zur Ähnlichkeitsberechnung zu den in dieser Arbeit vorgestellten Ähnlichkeitsmaßen in Beziehung gesetzt.

4.1 Einführende Beispiele

Am Beispiel der beiden Geschäftsprozessmodelle in Abbildung 32 und 33 wird die Notwendigkeit spezieller Ähnlichkeitsmaße für semantische Geschäftsprozessmodelle erklärt. Beide Geschäftsprozesse wurden mit Pr/T-Netzen modelliert, allerdings wurden Stellenmarkierungen und Transitionsinschriften aus Übersichtsgründen nicht in den Abbildungen angegeben. In der rechten Spalte beider Abbildungen ist ein Ausschnitt des entsprechenden semantischen Geschäftsprozessmodells abgebildet. In beiden Geschäftsprozessen werden Bestellungen abgewickelt. Allerdings haben die Prozesse einen unterschiedlichen Ablauf. Während im oberen Prozess die Aktivitäten sequentiell abgearbeitet werden, gibt es im unteren Prozess eine parallele Verarbeitung; d.h., die die Struktur der Prozesse ist unterschiedlich; die Se-

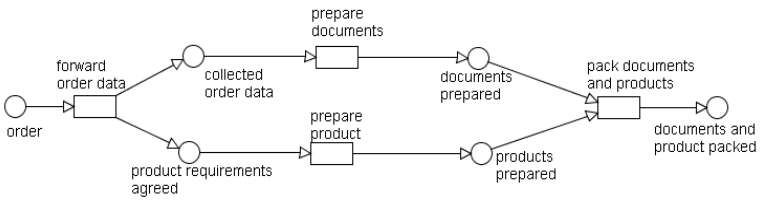
mantik der Elementnamen dafür ähnlich (*prepare software product* ähnelt dem Elementnamen *prepare product* oder *purchase order* ähnelt *order*).

Abbildung 32: Semantisches Geschäftsprozessmodell „sending purchase order"




```
<petri:Place rdf:ID="prepare_software-product">
   <petri:hasMarking>
      <petri:IndividualDataItem rdf:ID="R_prepare_software-product">
        <petri:hasAttribute rdf:resource="#Number"/>
        <petri:hasAttribute rdf:resource="#Name"/>
        <petri:hasAttribute rdf:resource="#Fabricator"/>
      </petri:IndividualDataItem>
    </petri:hasMarking>
  <petri:transRef rdf:resource="#software-product_prepared"/>
   ....
</petri:Place>
<petri:Transition rdf:ID="prepare_paper">
  <petri:placeRef>
    <petri:Place rdf:ID="paper_prepared/">
  </petri:placeRef>
</petri:Transition>
   .......
....
```

Abbildung 33: Semantisches Geschäftsprozessmodell „sending order"




```
<petri:Place rdf:ID="forward_order-data">
  <petri:transRef rdf:resource="#collected_order-data"/>
  <petri:transRef rdf:resource="#product-requirements_agreed"/>
    <petri:placeRef>
      <petri:Place rdf:ID="paper_product">
        <petri:hasMarking>
          <petri:IndividualDataItem rdf:ID="R_prepare_product">
            <petri:hasAttribute rdf:resource="#identification_number"/>
            <petri:hasAttribute rdf:resource="#name"/>
            <petri:hasAttribute rdf:resource="#producer"/>
          </petri:IndividualDataItem>
        </petri:hasMarking>
      </petri:Place>
    </petri:placeRef>
</petri:Place>
   .......
....
```

Bei den Ähnlichkeitsberechnungen zwischen Elementnamen, die in diesem Kapitel vorgestellt wird, werden nur indirekt die Struktur und der Kontrollfluss der Geschäftsprozesse berücksichtigt. Die Grundlage für die Ähnlichkeitsberechnungen bilden die Namen der Prozesselemente. Mit Hilfe eines Ähnlichkeitsmaßes ist es möglich, eine bestimmte Schwelle anzugeben, so dass die Menge an passenden und korrekten Folgeprozessen für die Modellierungsunterstützung aus einem Prozessrepository gefiltert werden kann. Beispielsweise wäre es sinnvoll, alle Folgeprozesse

mit einer Ähnlichkeit von kleiner als 0,3 auszusortieren, um die Ergebnismenge an relevanten Prozessen einzugrenzen.

4.2 Definition Ähnlichkeit

Nachfolgend wird eine Definition für ein allgemeines Ähnlichkeitsmaß gegeben, da dieser Begriff im weiteren Verlauf der Arbeit benutzt wird. Außerdem werden die beiden Begriffe Ähnlichkeitsmaß und Ähnlichkeitsmetrik voneinander abgegrenzt.
Mit einem Ähnlichkeitsmaß wird der Grad an Übereinstimmungen zwischen zwei Objekten bestimmt.

Definition 12: Ähnlichkeitsmaß
Eine Abbildung sim: $S \times S \rightarrow [0,1]$ über einer Menge S von semantischen Geschäftsprozessmodellen heißt Ähnlichkeitsmaß, falls $\forall$ x, y $\in$ S gilt [Rich92]:

(i)	sim(x,y)	=	sim(y,x)	(Symmetrie)
(ii)	sim(x,x)	=	1	(Reflexivität, Selbstähnlichkeit)

Im Falle von sim(x, y) = 1 $\rightarrow$ x = y wird diese Eigenschaft als *strenge Reflexivität* bezeichnet.

In der Mathematik wird als Metrik eine Funktion verstanden, die je zwei Punkte eines n-dimensionalen Raums einen reellen Wert zuordnet, welcher als Abstand der beiden Punkte voneinander aufgefasst wird [Carm92].
Eine Ähnlichkeitsmetrik hat neben den Symmetrie- und Reflexivitäts-Eigenschaften als weitere Eigenschaft die Dreiecksungleichung:

(iii)	sim(x,y)+sim(y,z) $\geq$ sim(x,z)	(erfüllte Dreiecksungleichung)

Der Abstand von zwei Objekten gibt deren Ähnlichkeit an; je näher die Objekte aneinander sind (je geringer der Abstand), desto ähnlicher sind sie.

4.3 Bestehende Ansätze

Ursprünglich stammt der Ähnlichkeitsbegriff aus der Philosophie (erstmals beim griechischen Philosophen Heraklit beschrieben) [Mall22]; er wurde kritisch von Tversky [Tver77] diskutiert. In der Informatik hat sich die Berechnung von Ähnlichkeiten durch die so genannte *Levenshtein-Distanz* [Leve66], ein formales Ähnlich-

keitsmaß zur Untersuchung der Ähnlichkeit zwischen Zeichenketten, etabliert[48]. Im Bereich von Datenbeständen haben Vorwerk [Vorw77], Veenker und Vorwerk [VeVo75] formale Ähnlichkeitsverfahren zum Vergleich von Zeichenketten definiert. Beim Zeichenkettenvergleich werden entweder nur einzelne Buchstaben von Wörtern verglichen oder so genannte n-Gramme[49]. Linguistische bzw. informationsbasierte Verfahren benutzen Hintergrundwissen (z.B. in Form von Ontologien), bei denen Begriffe und ihre Beziehungen untereinander in lexikalisch-semantischen Beziehungen vorgegeben sind. In den folgenden Unterkapiteln werden bestehende Ansätze aus der Literatur zur Ähnlichkeitsmessung vorgestellt.

4.3.1 Eigenschaftsbasierte Ähnlichkeit

Ein bedeutender Ansatz zur Ähnlichkeitsbestimmung von Objekten (Gegenständen) in der Psychologie stammt von Amos Tversky [Tver77]. Seine Theorie ist eine kritische Betrachtung der bis zu diesem Zeitpunkt angewandten Methoden, die sich hauptsächlich mit metrischen Distanzen beschäftigt haben. Seine Beobachtungen hat er auf das menschliche Empfinden von Beziehungen zwischen Objekten gestützt. Dabei hat er zwei fundamentale Annahmen getroffen [Blou01]:

- Sind die Darstellungen von Objekten als Punkte im Raum geeignet?
- Werden die Axiome der metrischen Distanz tatsächlich erfüllt?

Tversky hat beobachtet, dass nur wenige der Objekte im Raum darstellbar sind und viel häufiger bei der Wahrnehmung der Ähnlichkeit das Vorhandensein oder die nicht Existenz von bestimmten Eigenschaften eines Objektes entscheidend ist. Somit entspricht eine quantitative Darstellung von Objekten als Punkte im Koordinatensystem viel weniger unserem Empfinden als eine qualitative Betrachtung aller Eigenschaften von Objekten.
Sein zweiter Kritikpunkt galt der Erfüllung der metrischen Axiome (siehe [BaWN82]):

1. $d(x,y) = 0$ für $x = y$ und $d(x,y) > 0$ für $x \neq y$
2. $d(y,x) = d(x,y)$
3. $d(x,y) + d(y,z) \geq d(x,z)$

Tversky wies darauf hin, dass die bis dahin getroffene Annahme der Philosophen und Psychologen, die Ähnlichkeit sei immer eine symmetrische Beziehung, falsch sei. Er belegte diesen Kritikpunkt mit einer empirischen Studie, in der er beobachtet

[48] Eine detaillierte Übersicht über Verfahren zur Berechnung von Zeichenkettenähnlichkeiten findet sich unter http://www.dcs.shef.ac.uk/~sam/stringmetrics.html
[49] n-Zeichenketten

hatte, dass die Aussagen über die Ähnlichkeit tendenziell gerichtet sind. Bei der Aussage *A ist wie B* wurde bereits *B* als Referenzobjekt ausgewählt, da evtl. mehr Wissen über das Objekt *B* als über *A* vorliegt. Andere Beispiele für asymmetrische Ähnlichkeiten wären *das Porträt ähnelt der Person XY* oder *eine Ellipse ist wie ein Kreis*.

Diese Erkenntnisse können beim Ähnlichkeitsalgorithmus von Tversky berücksichtigt werden. In diesem Fall ist dann die Anfrage nach der Ähnlichkeit gerichtet, und dies soll in die Berechnung der Ähnlichkeit einbezogen werden. Damit hat er gezeigt, dass die Forderung von Symmetrie als Eigenschaft einer Ähnlichkeitsmetrik nicht immer zutrifft. Für die Verletzung der Dreiecksungleichung (3. Axiom) hat Tversky ebenfalls Gegenbeispiele genannt: d(*Uhr, Armbanduhr*) + d(*Armbanduhr, Armband*) ≤ d(*Uhr, Armband*).

In vielen Fällen ist die Dreiecksungleichung erfüllt, aber das obige Beispiel zeigt, dass man es nicht als Axiom hinnehmen kann. Ähnlich ist es mit der Minimalität. In vielen Fällen empfindet man, dass komplexere Objekte sich selbst ähnlicher sind, als es bei einfachen Objekten der Fall ist: *sim*(*Twin , Twin*) ≥ *sim*(*circle , circle*).

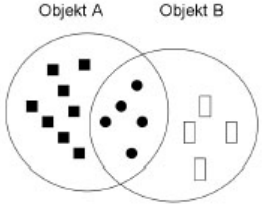
Mit diesen Erkenntnissen hat Tversky sein Contrast-Model vorgestellt, das die Ergebnisse aus seiner empirischen Studie berücksichtigt. In seinem Modell werden Objekte als Mengen von Eigenschaften - *sets of features*- dargestellt. Ein Bleistift kann als die Menge der Eigenschaften *ausHolz*, *länglich* und *Schreibmaterial* beschrieben werden:

Bleistift {*ausHolz, länglich, Schreibmaterial*}

Kreide {*weiss, Schreibmaterial, kurz*}

Beim Vergleich von zwei Objekten werden ihre Eigenschaftsmengen miteinander in Beziehung gesetzt. Jede Eigenschaft von Objekt A wird mit den Eigenschaften des Objektes B verglichen, um Übereinstimmungen zu finden. Daher ist die Methode auch als **Features Matching Process** bekannt. Bei Tversky wird Ähnlichkeit als eine lineare Kombination von gemeinsamen und unterscheidbaren Eigenschaften zwischen zwei Objekten definiert (siehe Abbildung 34).

Abbildung 34: Objekte als Feature Sets



Dabei ist die Berücksichtigung der unterscheidbaren Eigenschaften eine Besonderheit, die erst von Tversky vorgestellt wurde. Sie basiert auf der Beobachtung, dass im menschlichen Wahrnehmen Objekte durch Eigenschaften unterscheidbar sind und dies die Ähnlichkeit beeinflusst. Zur Berechnung der Ähnlichkeit von Objekten nach Tversky werden die gemeinsamen Eigenschaften und die Unterschiede zwischen den Objekten ermittelt. Dabei wird auch der Einfluss der *hervorstechenden* Eigenschaften bestimmt. Als *hervorstechend* werden Eigenschaften mit der höchsten Intensität und dem bedeutendsten Kontext bestimmt, wie in einem Beispiel bei [Heit97] angegeben: ein großer, roter Kamin auf dem Dach ist bestimmt mehr *hervorstechend* als ein kleiner grauer. Zudem ist ein Kamin, für jemanden der Kamine repariert, besonders *hervorstechend*.
Auf den Grundgedanken von Tversky, dass Eigenschaften Objekte beeinflussen, wird bei der Berechnung einer so genannten strukturellen Ähnlichkeit im Kapitel 4.6.3. zurückgegriffen, in dem der Kontext von Elementen bei der Berechnung der Ähnlichkeit berücksichtigt wird.

4.3.2 Syntaktische Ähnlichkeit

Die Levenshtein-Distanz, häufig auch Editierdistanz genannt, gibt die Anzahl an Änderungsoperationen an, die notwendig sind um eine Zeichenkette in eine andere zu ändern. Zu den Änderungsoperationen zählen das Einfügen, Löschen, Ersetzen oder Verschieben von einzelnen Zeichenketten. Die Levenshtein-Distanz hat sich vor allem bei der Aufdeckung von Tippfehlern bewährt. Das zu vergleichende Wortpaar (*Oerson* vs. *Person*) ergibt eine Levenshtein-Distanz von 1, da eine Änderungsoperation - nämlich das Ersetzen von *O* durch *P* - zu identischen Zeichenketten führt.
Die Levenshtein-Distanz zur Berechnung der syntaktischen Ähnlichkeit ist wie folgt definiert:

$$sim_{LevDist}(c_1, c_2) = 1 - \frac{LevDist(c_1, c_2)}{\max Length(c_1, c_2)} \qquad (4.1)$$

Dabei ist $LevDist(c_1,c_2)$ die Levenshtein-Distanz zwischen zwei zu vergleichenden Wortpaaren (c_1,c_2), die durch die Länge der längeren Zeichenkette von beiden Wörtern dividiert wird. Für das obige Beispiel ergibt sich beispielsweise eine $sim_{LevDist}$(Oerson, Person) von 0.83.

4.3.3 Informationsbasierte Ähnlichkeit

Resnik und Lin haben Verfahren vorgestellt, bei denen der Informationsgehalt von Konzepten mit Hilfe eines Textkorpus (Ansammlung von Dokumenten, die den Wortschatz für bestimmte Anwendungsgebiete angeben) bestimmt wird.
Resnik hat in [Resn99] ein neues Maß für die Ähnlichkeit in Taxonomien vorgestellt: jedem Konzept in der Taxonomie wird eine Zahl zugeordnet. Die Zahl beschreibt die Häufigkeit, mit der das Wort in einem Textkorpus vorkommt. Dabei wird auch die Taxomomie der Begriffe berücksichtigt, d.h. bei einem Vorkommen des Wortes *Student* wird es auch als Vorkommen von *Person*, *Lebenswesen* und *Objekt* gezählt. Solche Vorgängerknoten werden *subsumer* genannt, ihre Menge wird mit *words(c)* bezeichnet.

$$freq(c) = \sum_{n \in words(c)} count(n) \tag{4.2}$$

Gegeben die Häufigkeiten für alle Konzepte aus der Taxonomie definiert Resnik eine Funktion *p(c)*, die für jedes Konzept *c* eine Wahrscheinlichkeit *p* berechnet, mit der das Wort *c* samt subsumers *n* (∈ *words(c)*) im gegebenen Korpus vorkommt.

$$p(c) = \frac{freq(c)}{N} \tag{4.3}$$

Dabei ist *N* die gesamte Anzahl an beobachteten Wörtern (auch jene, die nicht von einer WordNet-Klasse subsumiert werden). Für das Wurzelelement gilt p(root)=1, weil das Wurzelelement alle Wörter subsumiert und somit ist die Wahrscheinlichkeit 1, dass im Korpus auf das Wurzelelement samt subsumers getroffen wird.
Diese Funktion ist monoton steigend, wenn man sich vom untersten Taxonomieknoten in der Ontologie zum Wurzelknoten bewegt. Um zu überprüfen, ob oberste Konzepte in der Ontologie weniger informativ sind als speziellere, die tiefer in der Taxonomie sind (*Zebra* vs. *Tier*), beschreibt Resnik den Informationsgehalt (information content) eines Konzeptes als eine aus der Informationstheorie [CoTh01] bekannte Gleichung. In der Gleichung ist der Informationsgehalt einer Aussage gleich dem negativen Logarithmus der Wahrscheinlichkeit dieser Aussage:

$$InformationsGehalt(C) = -\log(P(c)) \tag{4.4}$$

Das entspricht der intuitiven Vorstellung, dass der Informationsgehalt sinkt, je abstrakter das Konzept ist. Für das Wurzelelement ist er gleich 0. Die andere Überlegung von Resnik ist, dass die Ähnlichkeit von zwei Konzepten durch ihren am meisten informativen subsumer beschrieben werden kann. Je mehr Informationen zwei Konzepte gemeinsam haben, desto mehr Gemeinsamkeiten haben sie und desto informativer müsste der letzte gemeinsame Vorgänger der beiden Konzepte in der Taxonomie sein. Mit diesen beiden Überlegungen kann die Ähnlichkeit der Konzepte c_1 und c_2 als Informationsgehalt von dem *most informative subsumer* von c_1 und c_2 beschrieben werden.

$$sim_{IC_{resnik}}(c_1, c_2) = \max_{c \in S(c_1, c_2)} \left[-\log p(c) \right] \qquad (4.5)$$

Dabei ist $S(c_1, c_2)$ die Menge an Konzepten, die c_1 und c_2 subsumieren.

Lin hat in [Lin98] ein Modell vorgestellt, das dem von Resnik sehr ähnlich ist. Jedoch hat er theoretisch begründet, dass die Ähnlichkeit zwischen zwei Konzepten, nicht wie bei Resnik lediglich von deren Gemeinsamkeiten abhängt (also von dem Informationsgehalt des most informative subsumer (MIS)), sondern auch von den zu vergleichenden Konzepten beeinflusst wird.
Die gemeinsamen Eigenschaften werden durch den Informationsgehalt des MIS mit c_0 repräsentiert. Lin stellte auch fest, dass die Wahrscheinlichkeiten $P(c_1)$ und $P(c_2)$ unabhängig sind (die Auswahl der zwei Konzepte ist unabhängig voneinander) und stellt die Wahrscheinlichkeiten als Summe dar.

$$sim_{Lin}(c_1, c_2) = \frac{2 \times logP(c_0)}{\log P(c_1) + \log P(c_2)} \qquad (4.6)$$

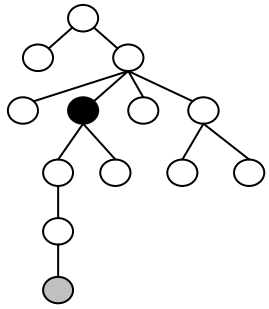
Zur Bestimmung der Ähnlichkeit zwischen Konzeptinstanzen in semantischen Geschäftsprozessmodellen steht allerdings kein Textkorpus zur Verfügung, weshalb diese beiden Verfahren nicht zur Bestimmung der linguistischen Ähnlichkeit zwischen Konzeptinstanzen verwendet werden. Diese Verfahren der Ähnlichkeitsmessung wurden vorgestellt, um die Definition eigener Ähnlichkeitsmaße für semantische Geschäftsprozessmodelle begründen zu können.

4.3.4 Ähnlichkeitsmetriken für Abstraktionsgrade

Um den Abstraktionsgrad zweier Konzepte in einer Taxonomie zu bestimmen, muss die relative Entfernung der Konzepte zur Wurzel berücksichtigt werden.

Eine Ontologie ist ein Graph, in dem Konzepte als Knoten repräsentiert werden. Der schwarze Knoten in Abbildung 35 ist näher an der Wurzel (oberster Knoten im Graphen) als der graue Knoten; somit hat der graue Knoten einen höheren Detaillierungsgrad als der schwarze Knoten.

Abbildung 35: Darstellung einer Ontologie als Graph



Die Bestimmung der Knotenentfernung in Graphen wurde bereits in der Graphentheorie untersucht (eine ausführliche Zusammenfassung findet sich in [ChGR94]). Zahlreiche Algorithmen - mit Unterstützung von Optimierungsmöglichkeiten und unter Berücksichtigung von Kantengewichtungen - können die kürzesten und längsten Pfade in Graphen bestimmen. Die Algorithmen lassen sich gut auf Graphen anwenden, in denen alle Knoten Konzepte auf gleicher Abstraktionsebene repräsentieren. Die Ergebnisse sind aber nicht zufrieden stellend, wenn Entfernungen zwischen zwei Knoten in einer Ontologie berechnet werden sollen. Die Ontologien besitzt hierarchische Strukturen, in der die Anzahl der Kanten zwischen zwei Konzepten oft unterschiedliche Bedeutung haben. Beispielsweise haben die beiden Konzepte *horse* und *zebra* eine Graphdistanz von zwei, da sie einen gemeinsamen Vaterknoten *equine* haben. Eine Graphdistanz von zwei haben auch *plant* und *animal*, weil sie unter dem gemeinsamen Vaterknoten *organism* subsumiert werden. Es ist aber intuitiv nachvollziehbar, dass ein *Pferd* einem *Zebra* ähnlicher ist als eine *Pflanze* einem *Tier*.
Die Anzahl der Kanten ist also nicht aussagekräftig genug, um den Abstraktionsgrad zweier Ontologiekonzepte zu bestimmen.

Wu & Palmer 1994

Wu & Palmer [WuPa94] haben eine Metrik zur Bestimmung der konzeptuellen Entfernung definiert, allerdings unter Berücksichtigung der Betrachtung der Spezialisierung der Konzepte, also ihrer Tiefe in der Taxonomie. Die Ähnlichkeitsmetrik wird als ein Quotient der Gemeinsamkeiten und der Unterschiede zwischen zwei Konzepten aufgefasst.

Die Ähnlichkeitsmetrik nach Wu & Palmer wird wie folgt definiert:

$$sim_{Wu\&Palmer}(c_1, c_2) = \frac{2 \times N3}{N1 + N2 + 2 \times N3} \quad (4.7)$$

Zunächst muss der nächste gemeinsame Vaterknoten von c_1 und c_2 (zwei Begriffe) gefunden werden, der c_3 genannt wird. Dann ist N1 die Anzahl an Knoten von c_1 nach c_3 und N2 die Anzahl an Knoten von c_2 nach c_3. N1 ergibt sich durch die Anzahl der Knoten von c_3 zum Wurzelelement der Taxonomie.
Somit wird die Tiefe der Konzepte in der Taxonomie berücksichtigt, nicht aber die Dichte der Taxonomie. Die beiden Konzepte *paper* vs. *document* haben als den nächsten gemeinsamen Elternknoten *instance writing*, der fünf Knoten vom *root*-Element in WordNet (siehe Kapitel 4.5) entfernt ist. *Paper* und *document* haben ein N1 von 1 und N2 von 2. Daraus ergibt sich eine Ähnlichkeit $sim_{Wu\&Palmer}$(paper, document) von 0.77.

Li, Bandar, McLean 2003

Zur Berechnung der konzeptuellen Distanz von zwei Begriffen haben Li, Bandar und McLean [LiBM03] neben der Pfadlänge zwischen zwei Konzepten auch die Dichte und Tiefe in der Hierarchie berücksichtigt. Li, Bandar und McLean haben die Ähnlichkeit *s* zwischen zwei Konzepten $c_1$und c_2 als Funktion der drei Parameter (*l* Länge, *h* Höhe, *d* Dichte) definiert mit:

$$sim(c_1, c_2) = f(l, h, d) \quad (4.8)$$

Im Gegensatz zum Kürzesten-Weg-Algorithmus aus der Graphentheorie wird bei Li, Bandar und McLean die hierarchische Struktur eines semantischen Netzes[50] berücksichtigt. Zwei Faktoren α und β gewichten den Einfluss der einfachen Entfernung im Netz und der Tiefe vom ersten gemeinsamen Elternknoten von c_1 und c_2. Das bedeutet, dass bei gleicher Entfernung im Graph die konzeptuelle Entfernung trotzdem unterschiedlich sein kann. Zwei abstraktere Konzepte, deren Entfernung vom Wurzelknoten gleich ist, haben weniger gemeinsam (die konzeptuelle Distanz ist größer), als zwei spezialisierte Konzepte, die tiefer in der Hierarchie sind.
Diese Lösung von Li, Bandar und McLean kann für Taxonomien und semantische Netze angewendet werden. Es wird aber vorausgesetzt, dass alle Beziehungen im Graphen vom Typ „IST-EIN" sind, was für Ontologien nicht zutrifft. Gerade die Viel-

[50] Zur Einführung in semantische Netze siehe [Sowa91].

falt an Beziehungen in Ontologien ist eine wichtige Eigenschaft der Ontologiemodellierung.

4.3.5 Zusammenfassung der Ähnlichkeitsmetriken

Die vorgestellten Verfahren zur Ähnlichkeitsmessung werden anhand eines Referenzverfahrens aus der Literatur verglichen. In einem Experiment haben *Miller* und *Charles* [MiCh91] Ähnlichkeitsmetriken anhand von englischen Begriffen verglichen[51]. In dem Experiment hat eine Gruppe von Studierenden die Ähnlichkeit zwischen bestimmten Wortpaaren nach dem menschlichen Empfinden bestimmen müssen. Dabei wurde 0 als untere Grenze gesetzt (überhaupt nicht ähnlich) und 4 als obere Grenze (identische Bedeutung bzw. Synonym-Beziehung) gewählt.
Die starke Korrelation mit anderen durchgeführten Versuchen verstärkt die Aussagekraft des Versuchs von Miller und Charles. Zu einem von *Rubenstein* und *Goodenough* [RuGo65] früher durchgeführten Versuch beträgt die Korrelation 0.97. Resnik hat das Experiment mit den gleichen Wortpaaren wiederholt und seine Ergebnisse [Resn99] waren mit den Ergebnissen von *Rubenstein* und *Goodenough* mit 0.96 korreliert.
Das menschliche Empfinden über die linguistische Distanz von Wörtern und den Sinn eines Wortes scheint eine gute Richtlinie zu sein, um die vorgestellten Verfahren miteinander zu vergleichen. Daraus soll die am besten geeignete Ähnlichkeitsmetrik für die Berechnung der abstraktionsniveaubasierten Ähnlichkeit zwischen Konzeptinstanzen von semantischen Geschäftsprozessmodellen bestimmt werden.
Wie von Resnik vorgeschlagen, wird als Referenzpunkt für alle Aussagen über die Ähnlichkeit zweier Objekte das menschliche Empfinden gewählt. Ein Ausschnitt der Ergebnisse der Studie ist in Tabelle 5 zusammengefasst sowie bei [Lin98] zu finden. Starke Korrelation mit der Studie von [RuGo65] war das Kriterium des Vergleichs.
Die Methode von Resnik [Resn99] liegt sehr nahe an den Ergebnissen von Miller und Charles (korreliert mit 0.795). Allerdings ist der Aufwand, den Informationsgehalt von Begriffen zu bestimmen, sehr groß und erfordert einen Textkorpus. Die gleichen Schwierigkeiten gibt es bei der Methode von Lin [Lin98], die sogar eine noch höhere Korrelation (0.834) mit dem Referenzverfahren hat. Als guter Kompromiss zwischen aufwendigen Berechnungen und zufrieden stellender Korrelation resultiert das Verfahren von Wu & Palmer. Die Idee, nicht nur Gemeinsamkeiten, sondern auch die Unterschiede bei der Ähnlichkeitsberechnung miteinzubeziehen, ist auch in dem Ansatz von Tversky wieder zu finden. Zudem liegen die Resultate

[51] Für den Vergleich der Effizienz von Ähnlichkeitsmetriken für die deutsche Sprache gibt es keine vergleichbaren Evaluationen.

nach Wu & Palmer sehr nahe an den Ergebnissen beim Versuch mit menschlicher Beurteilung, nämlich bei 0.803.

Tabelle 5: Vergleich verschiedener Ähnlichkeitsmetriken

Wort 1	Wort 2	Miller&Charles	Resnik	Wu&Palmer	Lin
car	Automobile	3.92	11.630	1.00	1.00
gem	Jewel	3.84	15.634	1.00	1.00
journey	Voyage	3.84	11.806	.91	.89
boy	Lab	3.76	7.003	.90	.85
coast	Shore	3.70	9.375	.90	.93
asylum	Madhouse	3.61	13.517	.93	.97
magician	Wizard	3.50	8.744	1.00	1.00
midday	Noon	3.42	11.773	1.00	1.00
food	Fruit	3.08	1.703	.33	.24
bird	Cock	3.05	8.202	.91	.83
bird	Crane	2.97	8.202	.78	.67
tool	Implement	2.95	6.136	.90	.80
brother	Monk	2.82	1.722	.50	.16
lad	Brother	1.66	1.722	.55	.20
journey	Car	1.16	0	0	0
food	Rooster	0.89	.538	.7	.04
coast	Hill	0.87	6.329	.63	.58
forest	Graveyard	0.84	0	0	0
monk	Slave	0.55	1.722	.55	.18
coast	Forest	0.42	1.703	.33	.16
Correlation with	**Miller and Charles**	**1.00**	**0.795**	**0.803**	**0.834**

Die Methode von Wu&Palmer wird im Rahmen dieser Arbeit verwendet, um Subklassen/Superklassen bzw. unterschiedliche Abstraktionsgrade von Prozesselementnamen zu bestimmen.

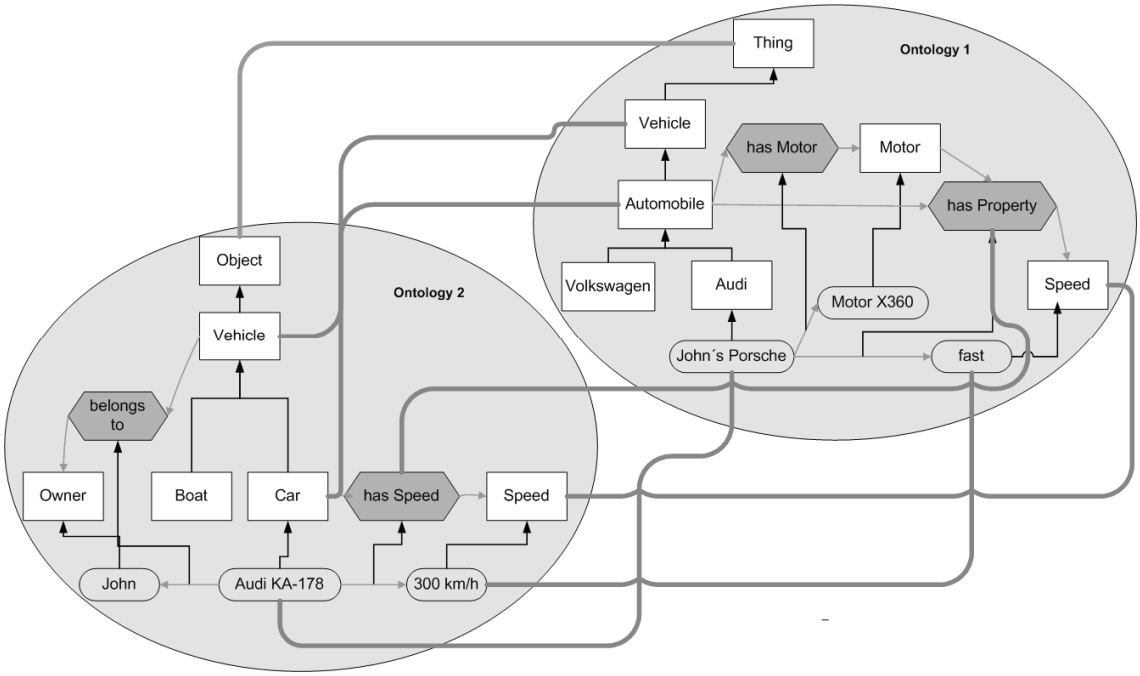
4.4 Unterschiede zwischen Ähnlichkeitsmaßen für Ontologien, Textdokumente und Geschäftsprozessmodelle

Aus den Unterschieden zwischen Ontologien, Textdokumenten und Geschäftsprozessen lassen sich Bedürfnisse für Ähnlichkeitsmaßen für Geschäftsprozesse ablei-

ten. Zudem soll die Wiederverwendung von in der Literatur bestehenden Ähnlichkeitsmaße für Ontologien und Textdokumente geprüft werden.

Eine Ontologie besteht aus Konzepten, Instanzen von Konzepten, Beziehungen zwischen Konzepten bzw. Instanzen und Axiomen. In der Abbildung 36 sind Konzepten als Rechtecke, Eigenschaften als Sechsecke und Instanzen als Ovale modelliert worden. Bei der Berechnung von Ähnlichkeiten werden Konzepte mit Konzepten, Eigenschaften mit Eigenschaften und Instanzen mit Instanzen verglichen. Die breiteren grauen Verbindungslinien zeigen die Korrespondenzen von Konzepten, Eigenschaften und Instanzen zu deren Entsprechungen in einer zweiten Ontologie.

Abbildung 36: Beispiel-Ontologie



Generell besteht der Name von Konzepten und Instanzen aus einem Begriff (z.B. Vehicle, Owner, Speed *vs.* fast, John), wobei die Namen von Instanzen entweder Hauptwörter oder Adjektive sein können. Bei der Benennung werden auch keine Abkürzungen benutzt, da eine Ontologie ansonsten unpräzise wäre (was auch im Widerspruch zur Idee einer Ontologie stehen würde). Weiterhin finden sich in der Ontologie auch keine Stoppwörter. Die Spezialisierung von Konzepten wird über Subklassen ausgedrückt. Ferner können Abhängigkeiten zwischen Konzepten/Eigenschaften durch spezielle Konstrukte ausgedrückt werden (z.B. *owl:inverseOf*, *owl:equivalentProperty*, *owl:sameClass*).

Textdokumente bestehen aus aneinander gereihten Sätzen, die in Zusammenhang miteinander stehen. Ein vollständiger Satz besteht mindestens aus einem Subjekt

und einem Verb, die sich sprachlich und logisch aufeinander beziehen. In Textdokumenten wird ein Satz durch weitere Satzbausteine (Objekte, Subjekte, Prädikate) erweitert, die miteinander zusammenhängen. Verben können in unterschiedlicher grammatikalischer Form benutzt werden und durch Adverbien näher beschrieben werden. Generell werden in Textdokumenten viele Stoppwörter verwendet (der, die, um, weil, dafür, weiteres, usw.).

Namen von Aktivitäten in Geschäftsprozessen werden durch ein Subjekt und ein Prädikat beschrieben und in der Regel nicht durch weitere Satzbausteine (wie bei Textdokumenten) erweitert. Damit werden Namen von Aktivitäten durch mehr Elemente als Ontologien und in der Regel weniger Elemente als Textdokumente beschrieben. Bei der Modellierung von Geschäftsprozessen für einen bestimmten Anwendungskontext wird ein Subjekt (zur Beschreibung von Prozessobjekten) häufig wieder verwendet; z.B. die *Bestellung*, die bearbeitet, die verschickt, die abgelehnt wird. Im Gegensatz zu Ontologien, bei denen keine redundanten Namen auftreten. In Textdokumenten kann die Wiederverwendung von Namen oft durch *der*, *die*, *diese* oder *sie* ersetzt werden. Darüber hinaus werden bei Prozesselementnamen - im Gegensatz zu Textdokumenten - viel weniger Stoppwörter und Abkürzungen benutzt (im direkten Vergleich zu Ontologien sind es viel mehr Stoppwörter und Abkürzungen). Eine unterschiedliche Abstraktion wird durch Verfeinerung von Aktivitäten durch Unterprozesse modelliert. Bei Ontologien wird eine Hierarchisierung durch Subklassen erzielt.

Zusammenfassend lassen sich einige Unterschiede zwischen Beschreibungen von Textdokumenten, Ontologien und Geschäftsprozessen feststellen, so dass die in der Literatur vorgeschlagenen Ähnlichkeitsmaße für Textdokumente und Ontologien nur bedingt bzw. gar nicht für Geschäftsprozesse wieder verwendet werden können. Dies macht eine Erweiterung von bestehenden Ähnlichkeitsmaßen aus der Literatur speziell für Ähnlichkeitsberechnungen zwischen Geschäftsprozessen notwendig.

4.5 Hintergrundwissen

Zur Berechnung der linguistischen Ähnlichkeit wird im weiteren Verlauf der Arbeit auf WordNet [Fell98] zurückgegriffen. WordNet ist ein lexikalisches Online-Referenzsystem – eine Datenbank mit über 200.000 Wörtern in englischer Sprache. Das Projekt WordNet ist mittlerweile 20 Jahre alt und stammt von dem Cognitive Science Laboratory an der Princeton University. Die Begriffe sind in Form eines semantischen Netzes in Mengen so genannter Synsets als Synonyme, Hyperonyme/Hypernyme und Meronyme aufgebaut. Jedes Synset repräsentiert ein lexikali-

sches Konzept. Das semantische Netz entsteht durch semantische Beziehungen zwischen verschiedenen Synsets. Die wichtigsten lexikalischen Beziehungen sind nach [Fell98]:

- **Synonyme:** Zwei Ausdrücke sind Synonyme, wenn das Austauschen dieser gegeneinander in einem beliebigen Satz den Wahrheitswert des Satzes nicht verändert. In der Literatur existiert auch eine abgeschwächte Definition für Synonyme: Zwei Wörter sind synonym in einem linguistischen Kontext *K*, wenn die Ersetzung des einen durch das andere den Wahrheitswert im Kontext *K* nicht verändert.
- **Antonyme**: Dichotomie der Bedeutung von zwei Wörtern (ist Gegenteil von).
- **Hyponyme/Hypernyme:** IST-EIN-Beziehung. Unter-/Oberkonzept-Beziehung. Hiermit kann die Generalisierung/Spezialisierung modelliert werden, z.B.: Baum IST-EINE Pflanze.
- **Meronymie**: IST-TEIL-VON Beziehung, z.B. Glasscheibe IST-TEIL-VON Fenster.
- **Morphologie:** Zu Beginn seiner Entwicklung sollte WordNet lediglich semantische Beziehungen enthalten. Schnell hat sich aber herausgestellt, dass auch die Morphologie (Wortstruktur) eines Wortes nicht außer Acht gelassen werden kann. Ein Benutzer, der nach dem Wort *Bäume* sucht, soll auch einen Treffer erzielen, wenn nur das Wort *Baum* in der Datenbasis enthalten ist.

Es existieren viele umfangreiche Online-Wörterbücher. Allerdings ist dort die Suche sehr zeitaufwendig und die Ergebnisse der Anfrage schwer nachvollziehbar, weil sie nicht geordnet sind. Die von *Miller* stammende Idee war es, den Benutzer nicht bei einer alphabetischen Suche, sondern einer Suche auf der konzeptuellen Ebene zu unterstützen.

4.6 Ähnlichkeitsmaße für semantische Geschäftsprozessmodelle

Mit Hilfe bestimmter Ähnlichkeitsmaße soll in den folgenden Unterabschnitten der Ähnlichkeitsgrad zwischen semantischen Geschäftsprozessmodellen bestimmt werden. Zur Aufdeckung von Tippfehlern oder grammatikalisch unterschiedlich benutzten Prozesselementnamen wird eine syntaktische Ähnlichkeit verwendet. Die linguistische Ähnlichkeit zwischen zwei semantischen Geschäftsprozessmodellen

korreliert positiv mit der Anzahl an verwendeten Synonymen, Homonymen und Hyponymen bzw. Hypernymen in semantischen Geschäftsprozessmodellen. Zur Bestimmung dieser linguistischen Mittel werden spezielle Ähnlichkeitsmaße vorgestellt.

4.6.1 Syntaktische Ähnlichkeit

Maedche und *Staab* [MaSt01] verwenden die Editierdistanz nach Levenshtein, um ein so genanntes String-Matching (SM) zwischen zwei Ontologiekonzepten zu berechnen. Dabei wird die Anzahl an Änderungsoperationen gegen die kürzeste Zeichenkette zweier Ontologiekonzepte gewichtet. Das Ergebnis der Ähnlichkeitsberechnung liegt zwischen 0 und 1; dabei gibt 1 an, dass zwei Ontologiekonzepte identisch sind. Das String-Matching nach Maedche und Staab ist invers zur Editierdistanz und wird wie folgt definiert:

$$sim_{syn}(c_1, c_2) = \max(0, \frac{\min(|c_1|, |c_2|) - ed(c_1, c_2)}{\min(|c_1|, |c_2|)}) \quad (4.9)$$

Das syntaktische Ähnlichkeitsmaß sim_{syn} ist 0, wenn die kleinste Zeichenkettenlänge von zwei Elementnamen kleiner ist als *ed*. Im Falle zweier identischer Zeichenketten (*request* vs. *request*) ist sim_{syn} immer gleich 1. Für die Instanznamen (*send verification* vs. *send confirmation*) wird sim_{syn} von 0.625 berechnet; dabei ist $\min(|c_1|, |c_2|)$ gleich 16 und $ed(c_1, c_2)$ gleich 6.

Im Falle von ganzen Sätzen als Elementnamen sollten Stemming[52] [Port97] und Stoppwort-Eliminierung verwendet werden, um sinnvolle Ähnlichkeitswerte zu berechnen. Der Elementname *send letter to human resources and staff council* wird auf *send_letter_human-resources_staff-council* reduziert. Dabei werden Wörter mit geringer eigener Bedeutung wie *and*, *or*, *not*, *to*, *with* oder *near* aus dem Elementnamen entfernt.

Allerdings reicht die Berechnung einer syntaktischen Ähnlichkeit in unserem Anwendungsszenario nicht aus, um Ähnlichkeitsbeziehungen zwischen Prozesselementnamen zu bestimmen. Beispielsweise ergibt sich für das Wortpaar *Waffe* vs. *Waffel* ein String-Matching von 0.83, da die Wörter orthographisch sehr ähnlich sind. Dieser Ähnlichkeitswert erscheint aber zu hoch, denn die Wörter haben eine unterschiedliche linguistische Bedeutung.

[52] Beim Stemming nach Porter wird beispielsweise –s als Endung bei Plural Nomen, Adjektiven und Verben zweite Person Singular entfernt.

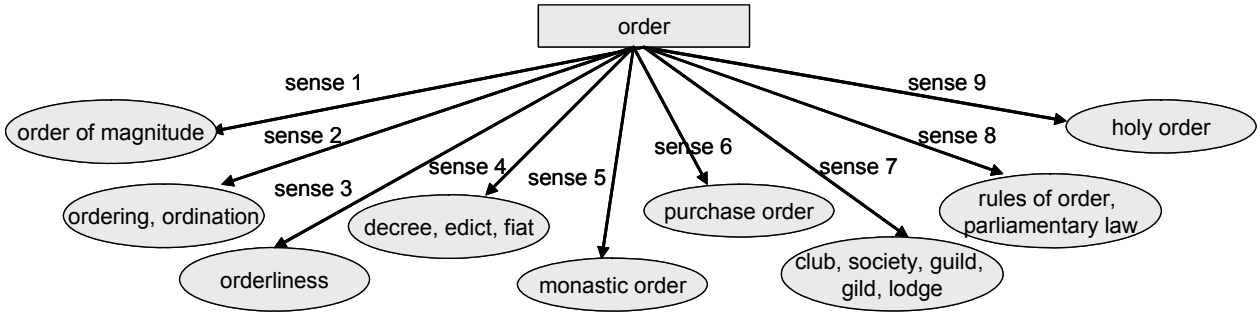
4.6.2 Linguistische Ähnlichkeit

Prozesselementnamen, die orthographisch sehr unterschiedlich sind, sollen eine syntaktische Ähnlichkeit von 0 haben. Die Berechnung einer linguistischen Ähnlichkeit könnte aber für dieses unterschiedliche Wortpaar einen Wert von 1 ergeben, wenn es sich bei den Wörtern um Synonyme handelt.

Zur Bestimmung von Synonymen zwischen Konzeptinstanzen von semantischen Geschäftsprozessmodellen wird auf Synsets von WordNet zurückgegriffen. Abbildung 37 zeigt neun Synonyme für den Begriff *order*, die von WordNet vorgegeben werden. Dabei werden Synonyme in WordNet über *senses* definiert. Die Beschreibung für das erste Synonym (sense 1) von *order* lautet:

> *a degree in a continuum of size or quantity; "it was on the order of a mile"; "an explosion of a low order of magnitude".*

Abbildung 37: Synonyme für den Begriff „order"



Analog zu dieser Beschreibung des ersten Synonyms gibt es für die weiteren *senses* von *order* Beschreibungen. Über die Beschreibung zu den einzelnen *senses* lässt sich die zutreffende Bedeutung im Prozessmodell erschließen. Allerdings sollen die Ähnlichkeitsmaße eine (semi-)automatische Berechnung von Ähnlichkeiten unterstützen. Die Erschließung der passenden Bedeutung erweist sich für den Computer ohne Kontextinformationen als unmöglich. Bei der (semi-)automatischen Berechnung der linguistischen Ähnlichkeit kann die Maschine nicht den zutreffenden *sense* filtern, weshalb alle Synonyme eines Begriffes, die in WordNet vorgegeben sind, in der Berechnung der linguistischen Ähnlichkeit berücksichtigt werden. Je mehr Synonyme ein Begriff hat, desto geringer wird die linguistische Ähnlichkeit sein.

Nicht immer benennen Modellierer Prozesselemente mit Hauptwörtern und Verben wie in den Modellierungsrichtlinien nach Holland[53] vorgeschlagen. Um Vergleichbarkeit von Ähnlichkeitswerten zwischen Prozesselementen, die nur mit Verben be-

[53] Bill Holland: Processing Modelling Guide; http://www.for.gov.bc.ca/his/datadmin/s47.htm

schrieben sind, zu erreichen, werden Verben substantiviert. *Find* wird zu *finding* oder:

- assign → assignment
- collect → collection

Laut Definition 12 ist der Ähnlichkeitswert von zwei Begriffen symmetrisch. Der sich ergebende numerische Wert ist für beide Begriffe gleich. Eine linguistische Ähnlichkeit kann nur zwischen zwei Konzeptinstanzen berechnet werden, wenn die Begriffe in der jeweiligen Menge der Synonyme zu finden sind. Zur Berechnung der linguistischen Ähnlichkeit werden über die Funktion $\eta(c)$ zunächst alle WordNet *senses*, die in Synonym-Beziehung zueinander stehen, abgefragt. $S = \eta(c_1) \cap \eta(c_2)$ ist die Schnittmenge an gemeinsamen Synonymen der beiden zu vergleichenden Namen von Konzeptinstanzen c_1 und c_2. Die Kardinalität *f(S)* definiert sich wie folgt:

$$f(S) = \begin{cases} 1 & \text{iff } \eta(c_1) \cap \eta(c_2) \neq \varnothing \\ 0 & \text{iff } \eta(c_1) \cap \eta(c_2) = \varnothing \end{cases} \tag{4.10}$$

Die beiden zu vergleichenden Begriffe haben eine unterschiedliche Anzahl an Synonymen. Bei der Berechnung der linguistischen Ähnlichkeit wird die maximale Anzahl an Synonymen von beiden Konzeptinstanzen verwendet. Ansonsten würde eine Berechnung der linguistischen Ähnlichkeit mit der minimalen Anzahl an Synonymen nicht alle Synonyme berücksichtigen. Dabei ist $\max = (\eta(c_1), \eta(c_2))$ das Maximum der Kardinalitäten der beiden Mengen $\eta(c_1)$ und $\eta(c_1)$.

$$sim_{ling}(c_1, c_2) = \frac{f(S)}{\max(|\eta(c_1)|, |\eta(c_2)|)} \tag{4.11}$$

Die linguistische Ähnlichkeit sim_{ling} ist gleich 1, wenn c_1 und c_2 Synonyme voneinander sind und beide keine weiteren Synonyme mehr haben. In diesem Fall ist der Zähler und Nenner von sim_{ling} gleich 1.0. Wenn c_1 und c_2 keine Synonyme sind, dann ist sim_{ling} immer gleich 0.0. Für das Instanzpaar (*send order* vs. *send purchase order*) resultiert eine sim_{ling} von 0.125; bei diesem Paar wird nur eine sim_{ling} von *order* vs. *purchase order* berechnet, weil die Zeichenkette *send* gleich *send* ist. *Order* und *purchase order* stehen in einer Synonym-Beziehung zueinander, wobei order die meisten Synonyme hat; nämlich 8.
Mit der linguistischen Ähnlichkeit können allerdings nicht Homonyme aufgedeckt werden. Für das Wortpaar (*order* vs. *order*) ergeben sich eine linguistische (zu-

nächst ohne Betrachtung der senses) und eine syntaktische Ähnlichkeit von 1. Ein Modellierer meint aber mit *order* einen *Auftrag* (*"a commercial document used to request someone to supply something in return for payment and providing specifications and quantities")*; jemand anderes subsumiert unter *order* eine *Anordnung* ("*a command given by a superior*"). Zur Aufdeckung von Homonymen bedarf es zusätzlicher Kontextinformationen.

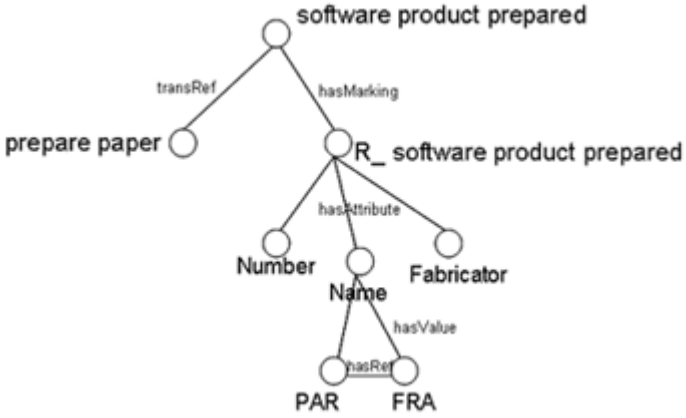
4.6.3 Strukturelle Ähnlichkeit

Als Kontext einer Konzeptinstanz werden alle Informationen subsumiert, die die linguistische Ausprägung des Prozesselementnamens beeinflussen. Dabei haben Stellen in der Pr/T-Netz-Ontologie einen anderen *Kontext* als Attribute und Transitionen. Der *Kontext* von Stellen wird als ein Tupel con_p mit <a, v, sV, tR> definiert, bei dem:

- a = alle Attribute der ausgewählten Stelle,
- v = alle Werte für jedes Attribut,
- sV = alle Nachbarwerte eines einzelnen Wertes und
- tR = alle nachfolgenden Transitionen, die über die Eigenschaft *transRef* mit der ausgewählten Stelle verbunden sind.

Abbildung 38 veranschaulicht den *Kontext* con_P für die Konzeptinstanz software product prepared aus Abbildung 32. Dabei wird die Markierung durch die Attribute Number, Name (das die Werte PAR für Paris und FRA für Frankfurt hat) und Fabricator beschrieben. Die nachfolgende Transition lautet prepare paper. In diesem Beispiel ist a=<Number, Name, Fabricator>; v=<PAR, FRA>; sV_{PAR}=<FRA> und sV_{FRA}=<PAR>; tR =<prepare paper>.

Abbildung 38: Ein Beispielkontext



In Abbildung 38 ist zu sehen, dass Attributnamen durch Werte näher beschrieben werden; Paris und Frankfurt sind eher *Name* als *Number*, was in diesem konkreten Beispiel auch modelliert wurde. Der Einfluss von Werten auf Attributnamen wird durch den *Kontext* con_A mit dem Tupel <sA, v, sV> definiert:

- sA = alle Nachbarattribute des ausgewählten Attributs,
- v = alle Werte des ausgewählten Attributs und
- sV = alle Nachbarwerte.

Transitionsnamen haben Einfluss auf nachfolgende Stellennamen. Deswegen wird der *Kontext* von Transitionen als ein Tupel con_T = <pR> mit:

- pR = allen nachfolgenden Stellen, die über die Eigenschaft placeRef mit der ausgewählten Stelle verbunden sind,

definiert.

Allerdings haben die Tupelelemente einen unterschiedlichen Einfluss auf die Elementnamen. Werte wie *125* oder *Meier* sind weniger aussagekräftig als Attribute, die einen höheren Einfluss auf beispielsweise den Stellennamen haben. Die Daten *Autor*, *Titel*, *Herausgeber* und *Verlag* deuten eher auf ein Buch als die Tupelelemente <*Hans Meier*, *XML* und *Johnson*>, die beispielsweise Daten in einem Ausweis, einer E-Mail oder einer Folie sein könnten.

Ein höherer bzw. geringerer Einfluss von Elementen wird durch die Gewichtung *w* berücksichtigt. Analog zur Gewichtung von statistischen Proben [FrPP97] hängt die Höhe der Gewichtung der Kontextelemente vom Einflussfaktor des Tupelelements und der Prozessgröße ab. Dabei ist die Gesamtsumme der Gewichte eines *Kontextes* immer 1.0. Elemente, die einen signifikant höheren Einfluss haben, werden höher gewichtet.

Bei großen Prozessmodellen (über ca. 100 Elemente) spielen Gewichte eine geringere Rolle, da ab einer bestimmten Größe Gewichte keine Auswirkungen auf das Ähnlichkeitsergebnis mehr haben. Im Rahmen der Evaluation der Modellierungsunterstützung wird im Kapitel 7 auf die Abhängigkeit der Höhe der Gewichtung bei der Berechnung von Ähnlichkeiten auf Prozessmodelle eingegangen.

Neben der Gewichtung wird der Ähnlichkeitswert von Kontextelementen durch das verwendete Ähnlichkeitsmaß beeinflusst. Die Berechnung der linguistischen Ähnlichkeit für Werte würde zu keinen sinnvollen Ergebnissen führen, denn Werte wie *einhundertdreizehn* oder *Meier* haben keine Synonyme. Bei Werten mit dem Wertebereich integer, double oder date wird keine arithmetische Differenz zwischen den

beiden Werten berechnet wie bei *Bergmann* [Berg02] vorgeschlagen. Eine syntaktische Ähnlichkeit wird nur berechnet im Falle von identischen ganzzahligen Werten. Tabelle 6 veranschaulicht den *Kontext* von Place, Attribute, Value und Transition und gibt eine Beispielgewichtung an.

Tabelle 6: Kontext und Gewichtungen von Konzeptinstanzen

Element	Kontext	Ähnlichkeitsmaß	Gewichtung
Stellen	Attribute Werte Nachfolger (transRef)	ling Ähn. synt Ähn. synt/ling Ähn.	0.5 0.1 0.4
Attribute	Nachbar-Attribute Werte	ling Ähn. synt Ähn.	0.6 0.4
Werte	Attribute Nachbar-Werte	ling Ähn. ling Ähn.	0.5 0,5
Transition	Nachfolger (placeRef)	synt Ähn.	1.0

Ist der Kontext für die einzelnen Elemente gegeben, ergibt sich die strukturelle Ähnlichkeit durch Berechnung der entsprechenden Ähnlichkeitsmaße für die Kontextelemente, die jeweils mit ihrer Gewichtung multipliziert werden. Dabei ist c_1 ein ausgewählter Instanzname eines semantischen Geschäftsprozessmodells und c_{2_j} eine Menge an Instanznamen aus einem anderen semantischen Geschäftsprozessmodell. Dann gibt $\max_{j\in 1..m}(\mathrm{sim}_k(c_1, c_{2_j}))$ das Maximum an syntaktischer und/oder linguistischer Ähnlichkeit (abhängig vom Kontextelement) zwischen den Kontextelementen der Instanzen c_1 und der Menge an Instanzen c_{2_j} an:

$$sim_{str}(c_1, c_2) := \frac{\sum_{i=1}^{n} \max_{j\in 1..m}\left(w_{k_i} * sim_{k_i}\left(c_{1_i}, c_{2_j}\right)\right)}{\sum_{i=1}^{n} w_{k_i}} \qquad (4.12)$$

Dieses Ähnlichkeitsmaß ist 1.0, wenn die linguistische und/oder syntaktische Ähnlichkeit für die Kontextelemente 1.0 ist. Für den in Abbildung 38 angegebenen Kontext des Namens *software product prepared* und einen anderen Beispielkontext für den Namen *software product prepared* mit *prepare report* (transition), *Number,*

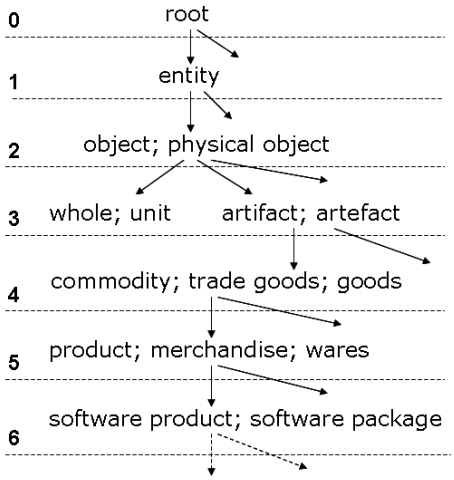
Name, Fabricator (Attribute) und *PAR* und *FRA* (Value) ergibt sich eine strukturelle Ähnlichkeit von

$$\text{sim}_{\text{str}_p} = \frac{0.5*\frac{1}{11}+0.5*\frac{1}{6}+0.5*1+0.1*1+0.1*1+0.4*\frac{1}{7}}{2.1} = 0.43.$$

4.6.4 Abstraktionsniveaubasierte Ähnlichkeit

Mit der Berechnung der abstraktionsniveaubasierten Ähnlichkeit sollen auch Konzeptinstanzen gefunden werden, die eine gleiche linguistische Bedeutung haben, aber ein unterschiedliches Abstraktionsniveau. Für das Instanzpaar (*product* vs. *software product*) wird eine syntaktische Ähnlichkeit von 0.4 und eine linguistische Ähnlichkeit von 0.0 berechnet, weil *software product* kein Synonym von *product* ist. Durch die Bestimmung der Hypernyme von *software product* mit WordNet wird anhand der Abbildung 39 deutlich, das *software product* ein Unterkonzept von *software* ist (Synonyme der einzelnen Wörter werden mit einem Strichpunkt getrennt) und eine linguistische Ähnlichkeit größer 0 hat.

Abbildung 39: Tiefe in der Taxonomie



Zur Berechnung der Abstraktionstiefe zweier Konzeptinstanzen wird die Ähnlichkeitsmetrik nach Wu&Palmer verwendet:

$$sim_{dep}(c_1, c_2) = \frac{2 \times N3}{N1 + N2 + 2 \times N3} \qquad (4.12)$$

Zunächst wird der nächste gemeinsame Elternknoten gesucht. Für das Wortpaar (*product* vs. *software product*) ist es product. Daraus ergibt sich ein N1 von 0, N2 von 1 und N3 von 5 und eine $sim_{Wu\&Palmer}$ von 0.91.
Die Ähnlichkeitsmetrik nach Wu&Palmer ist gleich 1.0, wenn N1 und N2 0 sind und N3 gleich 1. In diesem Fall handelt es sich um ein gleiches Paar, das dem Wurzelelement entspricht (*root* vs. *root*). Falls N1 und N2 keinen gemeinsamen Vaterknoten haben, dann ist sim_{dep} gleich 0.0.
Allerdings ist es möglich, dass ein Modellierer unter dem Namen *product* nicht einen *software product* subsumiert, sondern einen *development product*. Aus diesem Grund wird die Ähnlichkeitsmetrik nach Wu&Palmer um das strukturelle Ähnlichkeitsmaß erweitert. Die abstraktionsniveaubasierten Ähnlichkeit lautet:

$$sim_{abs}(c_1, c_2) := \frac{sim_{dep}(c_1, c_2) + sim_{str}(c_1, c_2)}{2} \tag{4.13}$$

Die abstraktionsniveaubasierten Ähnlichkeit ist 1.0, wenn sim_{dep} und sim_{str} 1.0 sind.

4.6.5 Kombinierte Ähnlichkeit

Durch Kombination der vier Ähnlichkeitsmaße ergibt sich eine kombinierte Ähnlichkeit, die als Ergebnis einen numerischen Wert für den Ähnlichkeitsgrad von Namen von Instanzpaaren liefert:

$$sim_{com}(c_1, c_2) :=$$

$$\frac{w_{c_{syn}} * sim_{syn}(c_1, c_2) + w_{c_{ling}} * sim_{ling}(c_1, c_2) + w_{c_{str}} * sim_{str}(c_1, c_2) + w_{c_{abs}} * sim_{abs}(c_1, c_2)}{w_{c_{syn}} + w_{c_{ling}} + w_{c_{str}} + w_{c_{abs}}}$$

Zur Bestimmung der kombinierten Ähnlichkeit für das Paar (prepare software product vs. prepare product) wird zunächst die syntaktische Ähnlichkeit der Namen berechnet und gewichtet. Anschließend wird die linguistische Ähnlichkeit der Namen bestimmt und mit einer individuellen Gewichtung multipliziert. Durch Addition der strukturellen und Abstraktionsniveau-Ähnlichkeit zu diesen beiden Ähnlichkeitsmaßen und Division durch die einzelnen Gewichte ergibt sich das kombinierte Ähnlichkeitsmaß.

4.6.6 Gesamtähnlichkeit

Die fünf Ähnlichkeitsmaße ermöglichen es, Ähnlichkeiten zwischen Namen von Konzeptinstanzen von semantischen Geschäftsprozessmodellen zu berechnen. Mit Hilfe des kombinierten Ähnlichkeitsmaßes kann eine Ähnlichkeit zwischen semantischen Geschäftsprozessmodellen ermittelt werden. Auf den ersten Blick sind sich die beiden Geschäftsprozessmodelle in Abbildung 32 und 33 inhaltlich ähnlich. Zur Ermittlung des Ähnlichkeitsgrades zwischen den beiden Geschäftsprozessmodellen wird die kombinierte Ähnlichkeit für alle Konzeptinstanzen beider Modelle berechnet. Dabei werden alle Konzeptinstanzen des einen Prozessmodells mit allen Konzeptinstanzen des anderen Prozessmodells verglichen. Es wird nur der höchste Ähnlichkeitswert zwischen zwei Instanznamen betrachtet. Dabei sei c_1 eine Konzeptinstanz von $SBPM_1$ und c_{2_j}, mit $j \in 1..m,$ eine Menge an Konzeptinstanz aus $SBPM_2$. Dann ist $\max_{j \in 1...m}\left(sim_{com}\left(c_1, c_{2_j}\right)\right)$ die maximale kombinierte Ähnlichkeit zwischen c_1 und Konzeptinstanzen c_{2_j} aus $SBPM_2$.

$$sim_{SBPM}\left(SBPM_1, SBPM_2\right) := \frac{1}{n}\sum_{i=1}^{n}\left(\max_{j \in 1...m}\left(sim_{com}\left(c_{1_i}, c_{2_j}\right)\right)\right) \in [0,1] \quad (4.14)$$

Für die Geschäftsprozessmodelle aus den Abbildungen 32 und 33 ergibt sich eine sim_{SBPM} von 0.75, die sich aus der Aggregation der einzelnen Ähnlichkeitswerte ergibt, von denen einige in der Tabelle 7 angegeben sind. Dieser numerische Wert der Ähnlichkeitsberechnung gibt den Ähnlichkeitsgrad zwischen zwei Geschäftsprozessmodellen an. Je näher der Wert bei 1 liegt, desto ähnlicher sind sich die Prozessmodelle.

Tabelle 7: Beispiel-Ähnlichkeitswerte

Instanzname	sim_{com}	sim_{syn}	sim_{ling}	$sim_{str\ P}$	$sim_{str\ A}$	$sim_{str\ V}$	$sim_{str\ T}$	sim_{abs}
#purchase order, #order	0.7	0.1	0.67	1.0	0.0	0.0	1.0	0.0
#collected purchase order data, #collected order data	0.7	0.1	0.67	1.0	0.90	0.0	0.0	0.0
#prepare software product, #prepare product	0.6	0.4	0.0	0.43	0.0	0.0	0.0	1.0

#paper prepared, #document prepared	0.75	0.39	0.0	0.95	0.0	0.0	0.0	1.0
#pack software product and paper, #pack document and product	0.65	0.1	0.0	0.85	0.0	0.0	0.0	0.95

Gleichheit, Ungleichheit und Durchschnitt zwischen den semantischen Geschäftsprozessmodellen gilt, wenn:

$$sim(SBPM_1, SBPM_2) = 1 \quad \text{iff } C_1 = C_2 \quad \text{(Gleichheit)}$$

$$sim(SBPM_1, SBPM_2) = 0 \quad \text{iff } C_1 \cap C_2 = \varnothing \quad \text{(Ungleichheit)}$$

$$sim(SBPM_1, SBPM_2) \in]0...1[\quad \text{iff } C_1 \cap C_2 = \{x | (x \in C_1) \wedge (x \in C_2)\} \quad \text{(Durchschnitt)}$$

Ein Ähnlichkeitswert von 0.75 bedeutet, dass 75% der Elemente des einen Prozessmodells den Elementen des anderen Prozessmodells gleich sind.

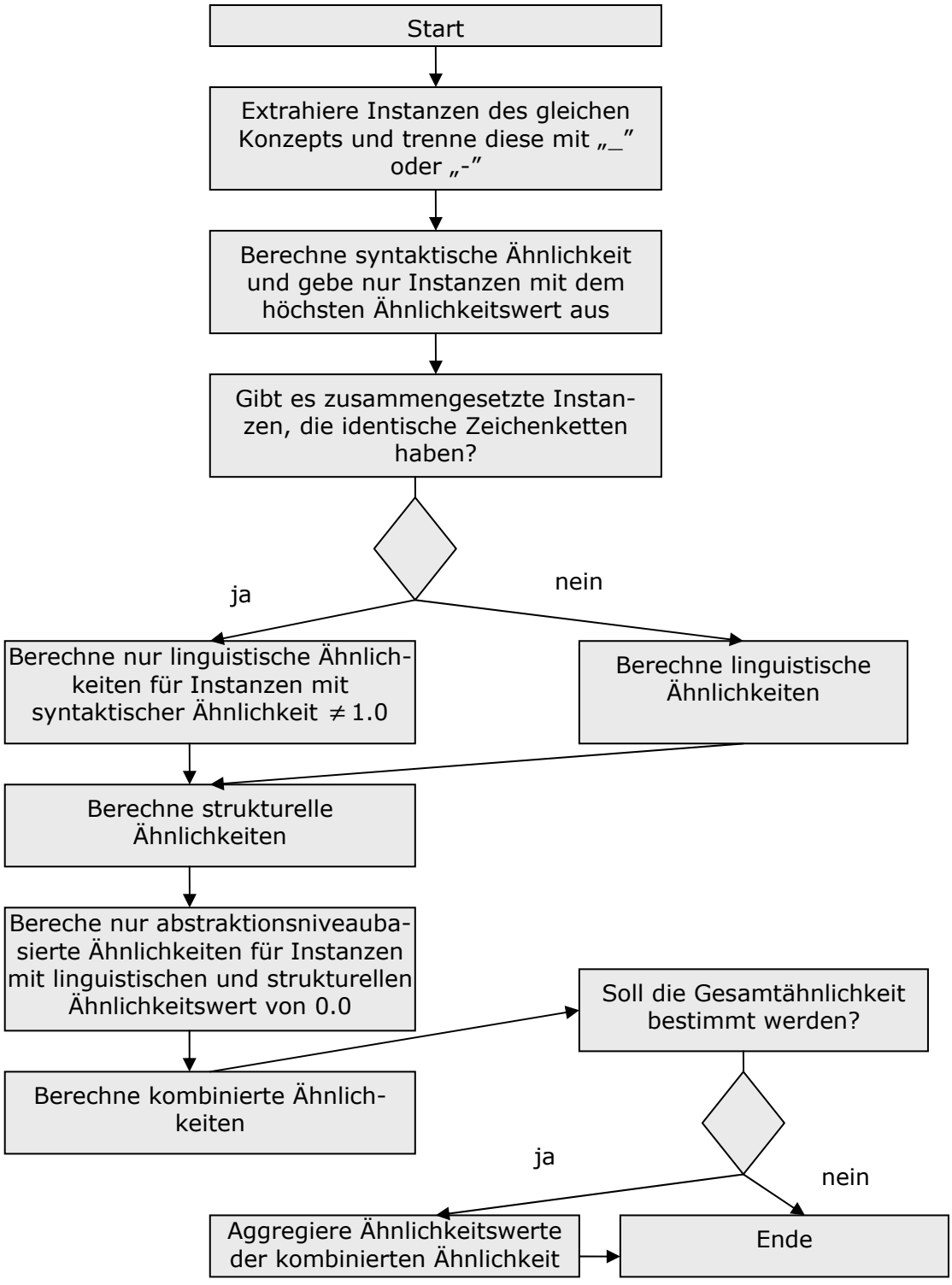
4.6.7 Ähnlichkeitsalgorithmus

Abschließend wird der verwendetet Algorithmus beschrieben, der Ähnlichkeitswerte zwischen Namen von Konzeptinstanzen und semantischen Geschäftsprozessmodellen berechnet.

Im ersten Schritt werden alle Konzeptinstanzen der zu vergleichenden Prozessmodelle extrahiert. Dabei werden nur Instanzen vom gleichen Konzept verglichen: Instanzen von *Place* nur mit Instanzen von *Place* und Instanzen von *Transition* nur mit Instanzen von *Transition*. Zusammengesetzte Instanzen werden mit einem „_" oder „-", wie in Abschnitt 3.2.3 erklärt, getrennt. Dann wird die syntaktische Ähnlichkeit berechnet. Beispielsweise wird der Name *prepare software product* mit allen Instanzen des Konzepts Transition aus dem anderen Prozessmodell verglichen, und es wird die syntaktische Ähnlichkeit bestimmt. Für die Paare mit der höchsten syntaktischen Ähnlichkeit wird die linguistische Ähnlichkeit berechnet – allerdings nur für Begriffe, die keine syntaktische Ähnlichkeit von 1.0 haben. Darauf folgend werden die strukturellen Ähnlichkeiten bestimmt. Im Falle einer linguistischen oder syntaktischen Ähnlichkeit von 0.0 wird die abstraktionsniveaubasierte Ähnlichkeit berechnet. Ansonsten werden nur linguistische, syntaktische und strukturelle Ähnlichkeitsmaße in der Berechnung berücksichtigt. Mit den Ergebnissen aus diesen Ähnlichkeitsmaßen ergeben sich die kombinierten Ähnlichkeiten und die Gesamt-

ähnlichkeit. Der nachfolgende Algorithmus veranschaulicht die Ähnlichkeitsberechnung. Als Input hat der Ähnlichkeitsalgorithmus mindestens zwei semantische Geschäftsprozessmodelle.

Abbildung 40: Ähnlichkeitsalgorithmus



5 SiMQL- eine Anfragesprache für Ähnlichkeitsmaße und eine Modellierungsunterstützung für Geschäftsprozesse

Anfragesprachen dienen dazu, Konstrukte bereit zu stellen, mit denen aus einem Informationsbestand die Informationen extrahiert werden können, die für den Anfragenden von Interesse sind. Neben der häufig genutzten Anfragesprache SQL[54] für relationale Datenbanksysteme wurden weitere Anfragesprachen vorgeschlagen, die aus speziellen Anforderungen entstanden sind. In diesem Kapitel wird eine Anfragesprache vorgestellt, mit der Anfragen an semantische Geschäftsprozessmodelle formuliert werden können. Es sollen Prozessteile nach bestimmten Kriterien aus einem Prozessrepository extrahiert werden können. Mit Hilfe der Anfragesprache soll auch nach Schnittstellen von Prozessen mit einem bestimmten Ähnlichkeitswert gesucht werden. Die Anfragesprache wird als eine (semi-) automatische Modellierungsunterstützung für Geschäftsprozesse verstanden.

5.1 Bestehende Anfragesprache für OWL

Als Anfragesprache für RDF wurde in der Literatur SPARQL vorgeschlagen, die zum aktuellen Zeitpunkt als Working Draft beim W3C vorliegt [W3C06a]. Für die Formulierung von Anfrage an OWL-Formate existiert beispielsweise OWL-QL[55]. Nachdem das OWL-basierte Beschreibungsformat von semantischen Geschäftsprozessmodellen über die RDF-Syntax hinausgeht, wird nachfolgend nur die vorgeschlagene Anfragesprache für OWL vorgestellt.

OWL-QL

OWL-QL wurde an der Stanford University entwickelt und ist eine deklarative Anfragesprache, die auf die Anfragesprache DAML-QL zurückgeht. Die Zielsetzung

[54] Die erste SQL-Norm wurde 1986 von der ANSI-Kommission verabschiedet. Im Jahre 1992 entstand das stark revidierte SQL-92.
[55] http://www.ksl.stanford.edu/projects/owl-ql/

von OWL-QL ist die Realisierung von Anfragen und Antworten zwischen OWL-Datenbasen über das Internet. Dabei agiert eine Kommunikationsseite als fragende und die andere Kommunikationsseite als antwortende Instanz. OWL-QL bietet eine formale Beschreibung der Anfragesprache durch semantische Beschreibungen der Beziehungen zwischen Anfragen, Antworten und einer Wissensbasis. Eine OWL-QL-Anfrage besteht aus einer Menge von OWL-Vorlagen. Eine solche Menge von Vorlagen wird als „Query Pattern" bezeichnet. Eine der wesentlichen Funktionalitäten der Sprache ist die Bindung von Elementen an Variablen. Dadurch wird eine Quantifizierung der Antwort garantiert. Es gibt drei verschiedene Bindungstypen, *must-bind var*, *may-bind var* und *dont-bind var*. Eine korrekte Antwort muss alle Bindungen für alle *must-bind var*- sowie *dont-bind var*-Bindungen berücksichtigen. Alle Bindungen von *may-bind var*-Bindungen können berücksichtigt werden. Im Query Pattern wird die Anfrage formuliert; also die Beziehungen und Eigenschaften von Objekten, die Ergebnisse der Anfrage liefern sollen. Der folgende Ausschnitt ist ein Referenzbeispiel, das aus der OWL-QL-Internetseite entnommen wurde. Zur Bearbeitung der Anfrage wird eine Datenbasis mit Personen, Fahrzeugen und Eigenschaften von Fahrzeugen (z.B. Farbe, Typ) benötigt. In der Anfrage wird nach Eigentümern von roten Autos gesucht. Dazu werden die Variablen `c` (car) und `p` (person) verwendet. Hierbei wird `c` durch die Bedingung `type` und `has-color` eingeschränkt. Da die Anfrage die Besitzer eines roten Autos ermitteln soll, ist die Person als zwingende Bindung anzugeben.

Quelltext 20: Anfrage mit OWL-QL

QueryPattern:

```
Query: ("Who owns a red car?")
Query Pattern: {(owns ?p ?c)(type ?c car)(has-color ?c Red)}
Must-Bind Variables List: (?p)
May-Bind Variables List: ()
Don't-Bind Variables List: ()
Answer Pattern: {}
Answer KB Pattern: …
```

Die Anfrage:

```
<owl-ql:query xmlns:owl-ql="http://www.w3.org/2003/10/owl-ql-syntax#"
      xmlns:var="http://www.w3.org/2003/10/owl-ql-variables#">
 <owl-ql:queryPattern>
  <rdf:RDF>
   <rdf:Description rdf:about="http://.../../owl-ql-variables#p">
    <owns rdf:resource="http://..../../owl-ql-variables#c"/>
   </rdf:Description>
     <Car rdf:ID="http://www.w3.org/2003/10/owl-ql-variables#c">
      <has-color rdf:resource="#Red"/>
     </Car>
```

```
  </rdf:RDF>
 </owl-ql:queryPattern>
 <owl-ql:mustBindVars>
   <var:p/>
 </owl-ql:mustBindVars>
 <owl-ql:answerKBPattern>
   <owl-ql:kbRef rdf:resource="http://joedata/joe.owl"/>
 </owl-ql:answerKBPattern>
 <owl-ql:answerSizeBound>5</owl-ql:answerSizeBound>
</owl-ql:query>
```

Als Antwort wird zurückgegeben:

```
Answer Pattern Instance: {(Joe owns a red car)}
Query:
Server: …
```

OWL-QL ist eine ausdrucksstarke Anfragesprache. Allerdings ist ein hoher Schreibaufwand notwendig, um selbst einfache Anfragen zu formulieren. Zudem erzeugt eine Anfrageauswertung eine große Menge an Daten, wodurch komplexe Anfrageauswertungen schnell ineffizient beziehungsweise nicht performant werden können.

Für die Berechnung der Ähnlichkeit von Elementnamen von semantischen Geschäftsprozessmodellen, werden spezialisierte Anfragen benötigt, die mit Anfragesprachen wie SQL oder OWL-QL nicht realisiert werden können (zumindest nicht ohne signifikante Erweiterungen der Sprache). Zudem werden die berechneten Ähnlichkeitswerte nicht in Relationen, sondern in einem eigenen Format, abgelegt. Aus diesem Grund wird in den nächsten Unterkapiteln SiMQL (Similarity Measures Query Language), eine eigene OWL-basierte Anfragesprache für Ähnlichkeitswerte, vorgestellt.

5.2 OWL-basierte Anfragesprache für Ähnlichkeitswerte

Die Anfragesprache für Ähnlichkeitswerte orientiert sich an der SQL-Anfragesyntax, die im einfachsten Fall aus dem Rahmenkonstrukt

- Auswahlklausel (SELECT; in SiMQL mit QUERY realisiert),
- Quellenklausel (FROM) und
- Bedingungsklausel (WHERE bzw. ONCON[56])

besteht:

```
SELECT <auswahl>        →    QUERY <auswahl>
FROM   <quelle>         →    FROM  <owl quelle>
WHERE  <bedingung>      →    ONCON <bedingung>
```

[56] Die Abkürzung ONCON steht für ON CONDITION

In SiMQL wurden aufbauend auf einer SQL-ähnlichen Syntax drei verschiedene Grundtypen von Anfragen definiert.

5.2.1 Anfragetypen

Im Kapitel 5.3. werden allgemeine und spezielle Anforderungen an Anfragesprachen beschrieben. Die speziellen Anforderungen haben insbesondere einen direkten Einfluss auf die Definition der Anfragesprache. Deswegen existieren drei verschiedene Typen, um Anfragen für Ähnlichkeitswerte von semantischen Geschäftsprozessmodellen zu formulieren. Eine Anfrage ermittelt immer eine Ergebnismenge, wobei eine leere Menge ebenfalls zulässig ist. Eine leere Menge bedeutet, dass kein Element die angefragte Bedingung erfüllt.
Die drei Anfragetypen könnten auch in einem Anfragetyp zusammengefasst werden, was allerdings die Bedienbarkeit der Anfragemaske/die Formulierung der Anfrage erschweren würde. Im Folgenden werden die drei Anfragetypen mit Beispielen erklärt:

Typ 1: QUERY_SELECTION

QUERY_SELECTION selektiert eine Untermenge aus einem oder mehreren ausgewählten semantischen Geschäftsprozessmodellen. Das können mehrere zusammenhängende Prozesselemente sein. Dabei wird ein Pfad zu einem eindeutigen Knoten oder zu einer Knotenebene angegeben. Die Angabe von mehreren Pfadangaben ist ebenfalls zulässig. Dieser Anfragetyp ist zunächst unabhängig von der Ähnlichkeitsberechnung, da nur danach gefragt wird, ob bestimmte Knoten (Stellen oder Transitionen) existieren (ohne Berücksichtigung eines Ähnlichkeitswertes).

QUERY_SELECTION beantwortet die folgenden Fragen:

- *Welche Elemente existieren in den semantischen Geschäftsprozessmodellen*?
- *Existiert ein spezielles Element in semantischen Geschäftsprozessmodellen?*

In einer Beispiel-OWL-Datei `travelBooking.owl` soll nach allen Transitionen (in SiMQL mit *Trans* abgekürzt) mit beliebigen Namen und Ähnlichkeitswert gesucht werden. Die Anfrage hierfür lautet:

Quelltext 21: Anfrage in SiMQL (QUERY_SELECTION)

```
QUERY
   OWL('travelBooking.owl')#TRANS()
FROM
   travelBooking.owl
```

Die Anfrage liefert zwei Transitionen mit dem Namen `send_rejection` und `check_request`:

```
TRANSITION:http://www.aifb.uni-.../travelBooking.owl#send_rejection
TRANSITION:http://www.aifb.uni-.../travelBooking.owl#check_request
```

Typ 2: QUERY_CONDITION

QUERY_CONDITION führt eine Ähnlichkeitsberechnung über genau zwei ausgewählte semantische Geschäftsprozessmodelle durch[57]. Die Ergebnismenge dieses Anfragetyps wird durch die Bedingung, die die Ähnlichkeitswerte erfüllen müssen, eingeschränkt. Dabei arbeitet dieser Anfragetyp auf einem angegebenen Pfad zu einem Knoten oder einer Knotenebene, der oder die als Ausgangspunkt für die Ergebnisermittlung dienen.

QUERY_CONDITION beantwortet die folgenden Fragen:

- *Welche Elemente von zwei semantischen Geschäftsprozessmodellen erfüllen nach Berechnung der Ähnlichkeit einen bestimmten Ähnlichkeitswert?*
- *Erfüllt ein bestimmtes Element nach Berechnung der Ähnlichkeit eine bestimmte Ähnlichkeit?*

Aus den beiden Beispiel semantischen Geschäftsprozessmodellen `travelBooking.owl` und `reservation.owl` sollen alle Stellen auf der ersten Ebene bestimmt werden, die eine kombinierte Ähnlichkeit größer 0.2 haben. Der Sternoperator steht für *all*e Elemente einer bestimmten Ebene, nach denen gesucht werden soll.

Quelltext 22: Anfrage in SiMQL (QUERY_CONDITION)

```
QUERY
   *
FROM
   travelBooking.owl,reservation.owl
ONCON
   OWL('travelBooking.owl')#PLACE().ALIGN('sim_com')>0.2
```

In einer weiteren Anfrage soll für alle Stellen auf der ersten Ebene der Durchschnittswert von Ähnlichkeitswerten ausgegeben werden. In der aktuellen Implementierung von SiMQL muss in der Operationsklausel ein *function operator value* angegeben werden, weshalb ein logischer Vergleich > 0.0 eingefügt werden muss.

[57] In der aktuellen Implementierung der Anfragesprache können nur zwei semantische Geschäftsprozessmodelle miteinander verglichen werden. Denkbar ist auch der Vergleich zwischen mehreren Prozessmodellen.

Quelltext 23: Weitere Anfrage in SiMQL (QUERY_CONDITION)

```
QUERY
   *
FROM
   travelBooking.owl,reservation.owl
ONCON
   OWL('travelBooking.owl')#PLACE().AVG(ALIGN('sim_com'))>0.0
```

Ohne den Sternoperator könnte die Anfrage aber auch verkürzt werden, indem die QUERY-Klausel weggelassen wird:

Quelltext 24: Abgekürzte Anfrage von Quelltext 23 mit SiMQL

```
FROM
   travelBooking.owl,reservation.owl
ONCON
   OWL('travelBooking.owl')#PLACE().AVG(ALIGN('sim_com'))>0.0
```

Die Anfrage aus Quelltext 23 bzw. 24 liefert als Ergebnis drei Stellen nämlich `client-data`, `confirmation` und `rejection` mit einem Ähnlichkeitswert von 0.27092.

```
Result for Condition:
      PLACE: [0.2709200803255186]
             http://www.aifb.uni-.../travelBooking.owl#client-data
      PLACE: [0.2709200803255186]
             http://www.aifb.uni-.../travelBooking.owl#confirmation
      PLACE: [0.2709200803255186]
             http://www.aifb.uni-.../travelBooking.owl#rejection
```

Typ 3: QUERY_ALIGNSELECTION

QUERY_ALIGNSELECTION führt ebenfalls eine Ähnlichkeitsberechnung über genau zwei semantische Geschäftsprozessmodelle durch. Im Gegensatz zu QUERY_CONDITION arbeitet QUERY_ALIGNSELECTION nicht auf einem Pfad, sondern auf der gesamten OWL-basierten Darstellung von Petri-Netzen und gibt für alle angegebenen Bedingungen jeweils die Teilmenge zurück, die diese Bedingung erfüllt.

QUERY_ALIGNSELECTION beantwortet die folgende Frage:
Welche Elemente erfüllen nach Berechnung eines Ähnlichkeitswertes x und einer eventuellen Berechnung einer Aggregatfunktion die Bedingung, die in dem logischen Vergleich mit einem Wert y geprüft wird?

Aus den beiden semantischen Geschäftsprozessmodellen `travelBooking.owl` und `reservation.owl` sollen der Durchschnitt und die Summe der kombinierten Ähnlichkeit und der Ähnlichkeit der strukturellen Ähnlichkeit von Stellen berechnet werden. Zudem soll das Maximum und Minimum der strukturellen Ähnlichkeit ausgegeben werden:

Quelltext 25: Anfrage mit SiMQL (QUERY_ALIGNSELECTION)

```
QUERY
  AVG(ALIGN('sim_com'),ALIGN('sim_str_P')),
  SUM(ALIGN('sim_str_P'),ALIGN('sim_str_A')),
  MAX(ALIGN('sim_com')), MIN(ALIGN('sim_com')),
  ALIGN('sim_com')
FROM travelBooking.owl,reservation.owl
```

Die Anfrage liefert das folgende Ergebnis:

```
Result for: AVG(ALIGN('sim_com'),ALIGN('sim_str_P'))
      PLACE:[0.7953733766233766]
            http://www.aifb.uni-.../reservation.owl#request
      PLACE:[0.7953733766233766]
            http://www.aifb.uni-.../travelBooking.owl#request
      TRANS:[0.24675324675324672]
            http://www.aifb.uni-.../reservation.owl#reject
      PLACE:[0.5066187021683674]
            http://www.aifb.uni-.../contact
      PLACE:[0.5066187021683674]
            http://www.aifb.uni-.../travelBooking.owl#booking
      ...

Result for: SUM(ALIGN('sim_str_P'),ALIGN('sim_str_A'))
      PLACE:[1.1814935064935066]
            http://www.aifb.uni-.../reservation.owl#request
      PLACE:[0.02647480867346939]
            http://www.aifb.uni-.../contact
      PLACE:[0.02647480867346939]
            http://www.aifb.uni-.../travelBooking.owl#booking
      ...
Result for: MAX(ALIGN('sim_com'))
      PLACE:[0.5907467532467533]
            http://www.aifb.uni-.../reservation.owl#request
      PLACE:[0.5907467532467533]
            http://www.aifb.uni-.../travelBooking.owl#request
      ...
Result for: MIN(ALIGN('sim_com'))
      PLACE:[0.013237404336734695]
            http://www.aifb.uni-.../contact
      PLACE:[0.013237404336734695]
            http://www.aifb.uni-.../travelBooking.owl#booking
      ...
Result for: ALIGN('sim_com')
      TRANS:[0.49350649350649345]
            http://www.aifb.uni-.../reservation.owl#reject
```

```
PLACE:[0.013237404336734695]
      http://www.aifb.uni-.../contact
      ...
```

5.2.2 Grammatik

In diesem Kapitel wird die Grammatik von SiMQL erklärt. Eine Grammatik gibt die formale Beschreibung einer Sprache exakt und ohne die Einschränkungen durch die Ungenauigkeit natürlicher Sprachen wieder. Eine der bekanntesten Notationen zur Darstellung von Grammatiken ist die Backus-Naur-Form (BNF) [Knut64] beziehungsweise deren erweiterte Form EBNF[58]. Die große Verbreitung von (E)BNF hat dazu geführt, dass zahlreiche Werkzeuge entwickelt wurden, die die Formulierung von (E)BNF-Notation unterstützen.

BNF und EBNF

Die Backus-Naur-Form ist eine Notation zur Definition von Grammatiken. BNF ist von John Backus und Peter Naur während der Entwicklung von Algol-60 entwickelt worden, um erstmals die Syntax einer Programmiersprache exakt definieren zu können. Mit der BNF können nach der Chomsky-Hierarchie [Chom59] Typ-2- bzw. kontextfreie Grammatiken beschrieben werden. BNF ist eine mächtige und kompakte Notation. Diese Eigenschaften haben zur großen Bekanntheit und weiten Verbreitung von BNF in verschiedensten Themengebieten geführt. Eine Menge von BNF-Regeln über einem Alphabet A heißt Grammatik.

BNF unterscheidet zwischen Terminal- und Nichtterminalsymbolen. Terminalsymbole sind Konstanten (Zeichen oder Zeichenketten), Nichtterminalsymbole sind Variablen (Platzhalter für syntaktische Elemente). Die Definition von Nichtterminalsymbolen wird als Produktion bezeichnet. Die Zeichenkette „::=" weist einem mit „< >" umschlossenen Nichtterminalsymbol ein Nichtterminal- oder Terminalsymbol zu. Alternativen bei der Zuweisung werden mit „|" dargestellt, Optionen mit „[ ]" geklammert und Wiederholungen durch Rekursionen erzeugt:

Alternative: `<Ziffer> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9`

Option:
```
<Kennzeichen> ::= <Buchstabe> [<Buchstabe>] [<Buchstabe>] -
<Buchstabe> [<Buchstabe>] Leerzeichen <Ziffer> [<Ziffer>]
[<Ziffer>] [<Ziffer>]
```

[58] http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf

Auf der linken Seite einer BNF-Regel steht immer eine syntaktisch Variable, die mit < > umschlossen ist. Die rechte Seite beschreibt syntaktische Variablen und Terminalzeichen, die durch Operatoren kombiniert werden können. Dabei gilt Klammer (Wiederholung, Option) vor Verkettung vor Auswahl.

EBNF

EBNF ist die bekannteste Erweiterung von BNF. EBNF ist durch den Standard ISO 14977 definiert und wird unter anderem vom W3C zur Definition von XML verwendet. Folgende Symbole stehen zur Notation von erweiterten BNF-Regeln zur Verfügung:

Tabelle 8: Syntax von EBNF

Definition	:=
Logisches Oder	|
Optionale Wiederholung	{..}
Terminalsymbole	”
Logisches Und	Explizit mit Leerzeichen getrennt
Endezeichen	;
Optional	[..]
Gruppierung	(..)
Wiederholung	*

Obwohl EBNF von den Konstrukten her nicht ausdrucksmächtiger als BNF ist, ist EBNF besser für die Beschreibung von Grammatiken geeignet, da komplexe Sachverhalte einfacher und kompakter zu beschreiben sind. Beispielsweise kann auf Rekursionen verzichtet werden, da bei EBNF das Symbol * für Wiederholungen eingeführt wurde.

```
Buchstabe   = 'a' | 'b' | 'c' | ... | 'x' | 'y' | 'z'
Wort        = ' ' | (Buchstabe)*
```

Die Grammatik für SiMQL wurde in EBNF notiert. Grundlage für die Grammatik ist die Definition der Symbole aller zulässigen Basistypen. Diese sind `char`, `digit` und `sign` sowie darauf basierend `double` und `string`. Das Symbol `operator` definiert alle zulässigen logischen Vergleichsoperatoren.

Abbildung 41: Basistypen von SiMQL

```
CHAR            =     A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H'
                      | 'I' | 'J' | 'K' | 'L' | 'M' | 'N' | 'O' |
                      'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' | 'W'
                      | 'X' | 'Y' | 'Z | 'a' | 'b' | 'c' | 'd' | 'e'
                      | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' | 'l' |
                      'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't'
                      | 'u' | 'v' | 'w' | 'x' | 'y' | 'z' ;
DIGIT           =     '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7'
                      | '8' | '9' ;
SIGN            =     '_' | '-' ;
STRING          =     (CHAR | DIGIT | SIGN) {CHAR | DIGIT | SIGN} ;
DOUBLE          =     DIGIT ['.'] {DIGIT} ;
OPERATOR        =     '<' | '>' | '<=' | '>=' | '=' | '!=' ;
```

Da alle Anfragen in SiMQL als Quelle ein oder mehrere semantische Geschäftsprozessmodelle verwenden, wird im Folgenden die Notation zur Angabe eines semantischen Geschäftsprozessmodells definiert. Dabei ist zu berücksichtigen, dass es sich nicht um eine vollständige Pfadangabe zu einem semantischen Geschäftsprozessmodell handelt, sondern nur um relative Pfade zu einem definierten Ordner, der durch (den Benutzer) an das Anfragesystem vorgegeben wurde.

```
OWL_FILE        =     STRING '.owl' ;
```

Um auf einem semantischen Geschäftsprozessmodell einen Pfad zu einem Knoten oder zu einer Knotenebene anzugeben, sind folgende Notationen zu verwenden: ein Pfad beginnt mit der Angabe des semantischen Geschäftsprozessmodells und der Angabe, ob auf `Trans` (Transitionen) oder `Place` (Stellen) angefragt wird. Um auf einen bestimmten Pfad zu navigieren, muss die Angabe `Ref()` verwendet werden; dabei muss `Ref()` mindestens einmal angegeben werden. Beispielsweise verweist die Formulierung OWL`(''<name>.owl'')#TRANS():Ref()` auf alle Stellenelemente auf der ersten Ebene. Bei wiederholter Angabe von Ref() wechselt der Typ der Knotenebene immer zwischen Trans und Place, abhängig vom initialen Typ.

Abbildung 42: Pfadangaben in SiMQL

```
OWL_PATH                =     OWL OWL_INDIVIDUAL {':' OWL_REFERENCE};
OWL                     =     'OWL("' STRING '.owl")#' ;
OWL_INDIVIDUAL          =     ('TRANS' | 'PLACE') OWL_INDIVIDUAL_TYPE ;
OWL_INDIVIDUAL_TYPE     =     '()' | '("' STRING '")' | '(' DIGIT ')' |
                              '(' DIGIT ',"' STRING '")' ;
OWL_REFERENCE           =     'REF' OWL_REFERENCE_TYPE ;
OWL_REFERENCE_TYPE      =     '()' | '("' STRING '")' |
                              '(' [OPERATOR] DIGIT ')' ;
```

Eine konkrete Anfrage wird in SiMQL mit der folgenden Syntax formuliert. Dabei werden die in Abschnitt 5.3.1. beschriebenen Anfragetypen verwendet, die mit dem Terminalsymbol `QUERY` eingeleitet werden.

```
QUERY                   =     'QUERY'(
                                      QUERY_SELECTION |
                                      QUERY_CONDITION |
                                      QUERY_ALIGNSELECTION
                                    ) ;
```

Im Einzelnen wird die Syntax der Anfragetypen wie folgt definiert:

Abbildung 43: Syntax der Anfragetypen von SiMQL

```
QUERY_SELECTION         =     OWL_PATH (',' OWL_PATH)* 'FROM'  OWL_FILE
                              (',' OWL_FILE)* ;
QUERY_CONDITION         =     OWL_FILE ',' OWL_FILE 'ONCON' CONDITION ;
QUERY_ALIGNSELECTION    =     (FUNCTION | MAX | MIN)
                              (',' (FUNCTION | MAX | MIN))*
                              'FROM'
                              OWL_FILE ',' OWL_FILE ;
```

Um eine Bedingung und eine Funktion zu definieren, wird `ALIGN` als Platzhalter für eines der folgenden Ähnlichkeitsmaße verwendet: `sim_com` (für kombinierte Ähnlichkeit), `sim_str` (strukturelle Ähnlichkeit), `sim_ling` (linguistische Ähnlichkeit), `sim_str_P` (strukturelle Ähnlichkeit von Stellen), `sim_str_T` (strukturelle Ähnlichkeit von Transitionen) oder `sim_str_A` (strukturelle Ähnlichkeit von Attributen). Um eine spätere Erweiterung der Implementierung von SiMQL zu vereinfachen, sind diese textuellen Repräsentationen der Ähnlichkeitsmaße nicht Teil der Grammatik.

Abbildung 44: Bedingungen und Funktionen von SiMQL

```
CONDITION          =    OPERATION ;
OPERATION          =    OWL_PATH '.' FUNCTION OPERATOR VALUE ;
FUNCTION           =    SUM | AVG | ALIGN ;
MAX                =    'MAX(' ALIGN ')' ;
MIN                =    'MIN(' ALIGN ')' ;
SUM                =    'SUM(' ALIGN ',' ALIGN {',' ALIGN} ')' ;
AVG                =    'AVG(' ALIGN ',' ALIGN {',' ALIGN} ')' ;
ALIGN              =    'ALIGN("' STRING '")' ;
VALUE              =    DOUBLE ;
```

Die Implementierung von SiMQL und die Integration in SemPeT werden im Kapitel 7.2.2 erklärt.

5.3 Anforderungen an die OWL-basierte Anfragesprache

Anfragesprachen lassen sich in *prozedurale* und *deklarative* Anfragesprachen unterscheiden. Bei deklarativen Anfragesprachen definiert der Benutzer, welche Daten für ihn interessant sind. Bei prozeduralen Anfragesprachen wird hingegen zusätzlich definiert, wie die Auswertung der Daten vorgenommen werden soll. Zur Konzeption einer deklarativen Anfragesprache für Ähnlichkeitswerte sind neben den allgemeinen Anforderungen an eine Anfragesprache auch spezielle systembedingte Anforderungen zu berücksichtigen. Diese speziellen Anforderungen begründen sich durch das verwendete technische Verfahren, das zur Berechnung von Ähnlichkeitswerten benutzt wird, und sind im Besonderen durch die zu Grunde liegende OWL-basierte Darstellung von Geschäftsprozessen (semantische Geschäftsprozessmodelle) bedingt.
Im Folgenden wird auf die von Heuer und Scholl [HeSc91] als allgemeingültig definierten Kriterien für deklarative Anfragesprachen näher eingegangen. Dabei wird die Einhaltung/Nichteinhaltung der Anforderungen für die neu definierte OWL-basierte Anfragesprache SiMQL überprüft.

5.3.1 Allgemeine Anforderungen

Als allgemeine Anforderungen werden definiert:

Ad-Hoc-Formulierung

Anfragen müssen direkt formuliert werden können, ohne dass der Benutzer ein vollständiges Programm schreiben muss.

Die Anfragesprache muss eine Syntax bereitstellen, die leicht verständlich ist und intuitiv eine Anfrageformulierung unterstützt. Diese Anforderung wird in der OWL-Anfragesprache durch eine SQL-ähnliche Syntax erreicht.

Deskriptivität

Anfragen sollen definieren, was man haben will und nicht wie das Ergebnis ermittelt werden soll.

Diese Anforderung gilt uneingeschränkt für die entwickelte OWL-basierte Anfragesprache, die eine deklarative und keine prozedurale Anfragesprache darstellt. Bei einer Anfrage muss der Benutzer keine Berechnungsalgorithmen definieren, sondern nur die für ihn relevanten Suchkriterien angeben.

Mengenorientiertheit

Eine Operation soll auf Mengen von Daten arbeiten und nicht auf einzelnen Elementen.

Diese Forderung wird auch von SiMQL erfüllt. Allerdings trifft der klassische Mengenbegriff nur teilweise zu, da zwar alle in einem Geschäftsprozessmodell vorkommenden Prozesselementnamen als Menge vorliegen, die einzelnen Elemente allerdings wechselseitig in Beziehung stehen (wie auch aus relationalen Datenbanksystemen bekannt). Das Anfrageergebnis ist jedoch wie gefordert eine nicht sortierte Menge, die ebenfalls durch Teilergebnisse erzeugt werden kann, die über Mengenoperationen verknüpft werden können.

Abgeschlossenheit

Ein Ergebnis einer Anfrage kann wieder als Eingabe für eine Anfrage verwendet werden.

Diese Forderung wird aufgrund der Struktur des Ähnlichkeitsergebnisses nicht erfüllt. Ein Anfrageergebnis beinhaltet neben den Elementnamen aus den zu Grunde liegenden OWL-basierten Darstellungen von zwei Geschäftsprozessmodellen immer auch die Metadaten, die sich durch die Ähnlichkeitsmessung ergeben (Ähnlichkeitswert, Elementname, Namensraum). Möglich wäre, diese Metadaten nur beim endgültigen Ergebnis einer Anfrage mit einzubeziehen, sodass geschachtelte Abfragen möglich wären.

Sicherheit

Jede syntaktisch korrekte Anfrage muss terminieren, darf nicht in eine Endlosschleife geraten und muss ein endliches Ergebnis liefern.

Die Forderung nach Sicherheit einer Anfrage ist erfüllt. Syntaktisch nicht korrekte Anfragen werden automatisch erkannt und nicht verarbeitet. Die Anfrage nach Transitionen als erstes Prozesselement hat einen leeren Rückgabewert, wenn der Prozess mit einer Stelle anfängt. Als Fehlermeldung bekommt der Benutzer ein „Error" und keine weiteren Informationen zur Fehlerentstehung.

Eingeschränktheit

Unter Maßgabe der Optimierbarkeit, Sicherheit und Effizienz darf die Anfragesprache keine komplette Programmiersprache sein.

Die Eingeschränktheit gilt für SiMQL, da die Anfragesprache keine komplette Programmiersprache ist.

Vollständigkeit

Die Anfragesprache muss mindestens so mächtig sein wie eine Standardsprache. (Eine Anfrage einer relationalen Anfragesprache muss zum Beispiel mindestens so mächtig sein wie die ihr zu Grunde liegende Relationenalgebra.)

Diese Forderung bezieht sich direkt auf das Datenmodell der Anfragesprache. Das verwendete Datenmodell der Anfragesprache ist OWL DL. Die OWL-basierte Darstellung von Petri-Netzen ist ein spezielles OWL-Schema, das zwar auch in einer Beschreibungslogik (SHOIN-D) modelliert werden kann, allerdings speziell für Petri-Netze entwickelt wurde. Bei diesem Format handelt es sich nicht um ein klassisches Datenmodell, wie es durch die Relationenalgebra für relationale Datenbanken vorgegeben wird. Die Anforderung der Vollständigkeit kann nicht direkt gewährleistet werden; formale Vergleiche zwischen dem OWL-Schema und der Anfragesprache wären notwendig, um diese Anforderung zu erfüllen.

Adäquatheit

Alle Konstrukte, die im Datenmodell verwendet werden, müssen unterstützt werden.

Neben dem Kriterium Vollständigkeit kann auch die Adäquatheit nicht gewährleistet werden, da nur ein Teil des zu Grunde liegenden OWL-Modells für die Ähnlichkeitsmessung verwendet wird. So sind in der OWL-basierten Darstellung von Geschäftsprozessen Einschränkungen und Klassen definiert, die nicht für die Ähnlichkeitsberechnung benötigt werden (z.B. `owl:disjointWith`).

Orthogonalität

Sofern eine Typübereinstimmung gegeben ist, können Sprachkonstrukte gleich verwendet werden und sind beliebig kombinierbar.

Funktionen und Operationen, die einen Rückgabewert vom gleichen Typ haben, können nicht in jedem Fall orthogonal verwendet werden. So können zwar zum Beispiel Summen- und Durchschnittsfunktion äquivalent verwendet werden, jedoch nicht die Summen- und Maximalfunktion. Bei der Summen- und Durchschnittsfunktion werden Operationen über zwei angegebene Ähnlichkeitsmaße gebildet. Bei der Maximalfunktion wird der Maximalwert eines Ähnlichkeitswerts bestimmt. Eine Orthogonalität ist allerdings bezüglich der Operations- und Funktionsparameter gewährleistet.

Erweiterbarkeit

Die Anfragesprache soll bei einer Erweiterung des Datenmodells ebenfalls erweiterbar sein.

Die Anfragesprache wurde so konzipiert, dass spätere Erweiterungen keine (oder nur minimale) Überarbeitungen der bestehenden Strukturen bedingen. Das Kriterium der Erweiterbarkeit ist demzufolge gewährleistet, da SiMQL auf einer Grammatik basiert, die mit Standardmethodik (JAVA, BNF) entwickelt wurde und eine leichte Erweiterung unterstützt.

Optimierbarkeit

Die Anfragesprache besteht aus wenigen optimierbaren Operationen.

Da die Deskriptivität gewährleistet ist, können alle Operationen unabhängig optimiert werden. Obwohl bei der prototypischen Implementierung der Anfragesprache nicht die Optimierung im Vordergrund stand, ist eine hinreichende Optimierung im Zuge der Implementierung erfolgt (beim Zugriff auf semantische Geschäftsprozessmodelle und das Outputformat der Ähnlichkeitsberechnung).

Effizienz

Operationen sollen effizient ausführbar sein.

Die Effizienz kann nicht im herkömmlichen Sinne ermittelt werden. Es kann keine Komplexität angegeben werden, da die Berechnung der Ähnlichkeitswerte - die direkt das Anfrageergebnis bestimmen - implementierungsbedingt unterschiedliche Effizienzgrade hat. Im besten Fall dauert die Ergebnisberechnung 1 Minute und im schlimmsten Fall 30 Minuten. Die Ähnlichkeitsmessung bestimmt während des gesamten Verarbeitungsvorgangs dynamisch weitere Berechnungen, die eventuell einer erneuten Anfrage des Benutzers bedürfen. Die Berechnungsdauer der

Ähnlichkeitswerte steigt mit zunehmender Größe der semantischen Geschäftsprozessmodelle.

Eine Zusammenfassung der Kriterien und ihr Erfüllungsgrad in SiMQL sind in Tabelle 9 dargestellt.

Tabelle 9: Erfüllung allgemeiner Anforderungen von SiMQL

Anforderung	Erfüllt	Teilweise erfüllt	Nicht erfüllt
Ad-Hoc-Formulierung	x		
Deskriptivität	x		
Mengenorientiertheit		x	
Abgeschlossenheit			x
Sicherheit	x		
Eingeschränktheit	x		
Vollständigkeit			x
Adäquatheit			x
Orthogonalität		x	
Erweiterbarkeit	x		
Optimierbarkeit		x	
Effizienz		x	

5.3.2 Spezielle Anforderungen

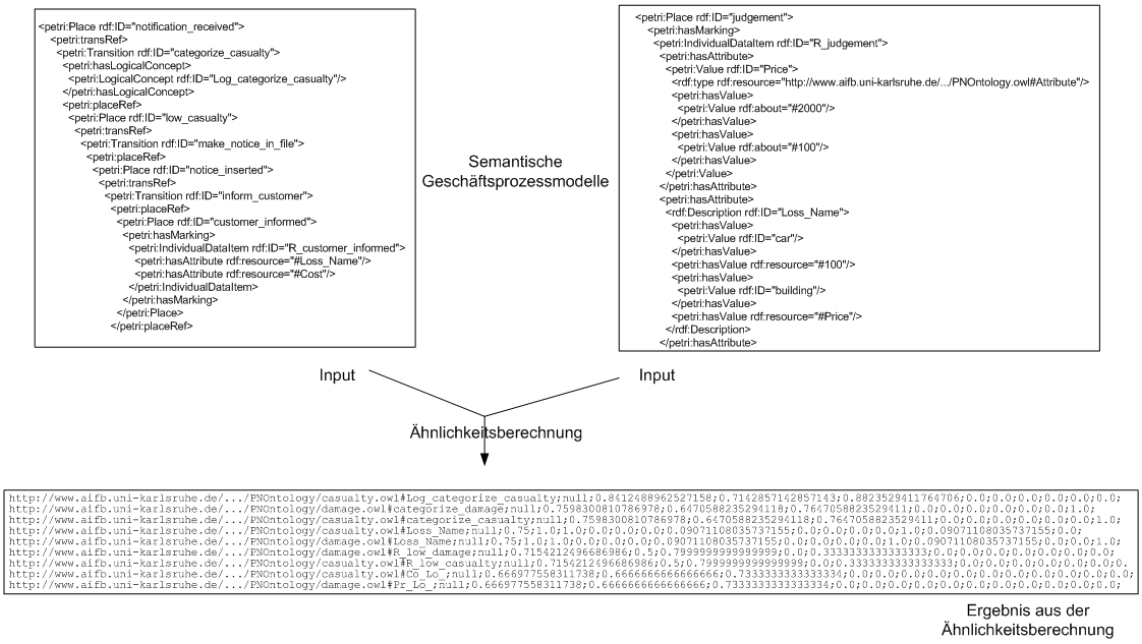
Bei der prototypischen Entwicklung von SiMQL sind neben den allgemeinen Kriterien für eine Anfragesprache auch spezielle Anforderungen zu berücksichtigen. Diese Anforderungen ergeben sich einerseits aus technischen Gegebenheiten und andererseits aus dem Erfordernis, dass die Anfragesprache und die darauf aufbauende grafische Oberfläche in eine Werkzeugumgebung zur Modellierung von Geschäftsprozessen, zur XML- und OWL-Serialisierung und zur Berechnung von Ähnlichkeitsmaßen (das Modellierungswerkzeugs SemPeT wird im Kapitel 8 erklärt) eingebunden werden soll.

Integration und Technik

Bei der Integration der Anfragesprache in SemPeT ergeben sich technische Herausforderungen, die berücksichtigt werden müssen. Eine Herausforderung ergibt sich direkt aus der Anfrage von Ähnlichkeitswerten. Die Berechnung der Ähnlichkeits-

werte wird temporär für jede Anfrage vorgenommen und das Ergebnis wird in einem eigenen Datenformat abgelegt. Die Syntax für die Anfragesprache muss somit zwei verschiedene Formate kennen und eine Mischung aus beiden Formaten interpretieren können. Abbildung 45 zeigt die unterschiedlichen Formate von semantischen Geschäftsprozessmodellen und vom Ergebnis der Ähnlichkeitsberechnung.

Abbildung 45: Input- und Outputformate der Ähnlichkeitsberechnung



Soll eine Anfrage nicht über die gesamten semantischen Geschäftsprozessmodelle erfolgen, muss das OWL-Modell, das die OWL-basierte Darstellung von Petri-Netzen repräsentiert, eingeschränkt werden. Eine Einschränkung ist in diesem Sinne als eine Filterung auf eine Teilstruktur der OWL-basierten Darstellung zu verstehen. Diese Einschränkung erfolgt durch die direkte Auswahl eines Knotens in der OWL-basierten Darstellung beziehungsweise durch Auswahl einer Knotenebene. Eine Knotenebene kann durch zusätzliche Angabe eines Attributwertes nochmals gefiltert werden. Durch Angabe des Attributes für die eindeutige Kennzeichnung eines Knotens ist es so möglich, einen speziellen Knoten für eine Anfrage zu definieren. Im Vergleich dazu sind beispielsweise bei SQL alle Einschränkungen auf Teilmengen der ursprünglichen Datenbasis entweder durch geschachtelte Anfragen oder durch entsprechende Angaben in der Bedingungsklausel vorzunehmen.

Bedienbarkeit

Die Anfragesprache SiMQL wurde in SemPeT über eine graphische Oberfläche integriert. Dabei stand bei der Implementierung von SiMQL die Benutzerfreundlichkeit der Anfrageformulierung im Vordergrund, weshalb für die Anfrageklauseln „Kurzschreibweisen" definiert wurden. Nicht benötigte Angaben, die sich direkt aus dem Kontext ergeben, sind bei der entwickelten Anfragesprache nicht anzugeben.

Um alle allgemeinen und speziellen Anforderungen erfüllen zu können, wären Erweiterungen der Grammatik notwendig. Diese Erweiterungen werden im Kapitel 10.4 (methodische Grenzen des Forschungsansatzes) skizziert.

5.4 Ausdrucksmächtigkeit

Die Entwicklung eines Datenmodells verlangt Kompromisse. So korreliert die Ausdrucksmächtigkeit einer Sprache mit der Effizienz der Implementierung.
Vereinfacht betrachtet ist die Ausdrucksmächtigkeit von SiMQL beschränkt auf die Anfrage eines Ähnlichkeitswerts von Stellen und Transitionen. Der Ähnlichkeitswert ist ein numerischer Wert, der zwischen 0 und 1 liegt und aus der Ähnlichkeitsberechnung zwischen zwei Knoten (Stellen oder Transitionen) resultiert.
Die primitivste Anfrage in SiMQL ist eine Suche nach allen existierenden Knoten in einem bestimmten Geschäftsprozess; d.h., SiMQL unterstützt den All-Operator (*). Die Anfrage kann konkretisiert werden, indem nur alle existierenden Stellen (Place()) bzw. Transitionen (Trans()) eines Geschäftsprozesses ausgegeben werden sollen. Aus der Menge aller vorhandenen Knoten kann dann nach Knoten mit bestimmten Eigenschaften, insbesondere einem Ähnlichkeitswert gesucht werden (z.B. sim > 0.3). Wobei SiMQL alle Ähnlichkeitsmaße, wie sie im Kapitel 4.6 definiert wurden, unterstützt; d.h., über SiMQL können Knoten ermittelt werden, die eine bestimmte syntaktische Ähnlichkeit haben. Darüber hinaus unterstützt SiMQL alle aggregierten Funktionen (Min, Max, Sum, Avg). Durch geringe Veränderung können Ergebnisse mehrerer QUERY-Anweisungen mittels der Mengenoperationen wie z.B. Vereinigung verknüpft werden. Die Ausdrucksmächtigkeit von SiMQL beruht auch auf der Möglichkeit weitere QUERY-Anweisungen innerhalb des ONCO-Teils formulieren zu können.
Bei Ähnlichkeitswerten handelt es sich um numerische Werte; somit können mit SiMQL keine string-Operationen und Funktionen formuliert werden wie beispielsweise ein Substring-Matching (starts-with, ends-with). Ferner ist auch kein Casting von Datentypen möglich.

SiMQL arbeitet auf semantischen Geschäftsprozessmodellen und dem Outputformat der Ähnlichkeitsberechnung. Der Benutzer kann über SiMQL nur Knoten anfragen, diese aber nicht löschen (Delete), ersetzen (Replace) oder neue Knoten einfügen (Insert). Diese Operationen wurden in SiMQL nicht realisiert, da die Anfragesprache als Hilfestellung für die Integration von Geschäftsprozessen dienen soll und Benutzern die Resultate (bei der Anfragesprache Knoten bzw. Prozessfragmente) benutzerindividuell verändern können/sollen und nicht die Knoten von Geschäftsprozessen in einem Prozessrepository manipulieren sollen.
Die Anfrageformulierung in SiMQL bietet bislang keine Möglichkeiten, um Datentypen, Operationen und Funktionen selbstzudefinieren, damit die Lesbarkeit von Anfragen und die Wiederverwendung verbessern werden. Bei SiMQL handelt es sich um eine deklarative Anfragesprache. Somit hat der Benutzer bei der Formulierung seiner Anfrage mit SiMQL keinen Einfluss auf die Auswertungsreihenfolge von Funktionen und Operationen und damit auf eine effiziente Auswertung.
SiMQL bietet bislang noch keine Möglichkeit, um Ablaufmuster von Geschäftsprozessen (siehe Kapitel 2.2) innerhalb der Anfrage formulieren zu können. Es kann nur nach Knoten gesucht werden, die einem bestimmten Ähnlichkeitswert entsprechen und nicht, ob sie parallel zu einander sind.

6 Konzeption einer Empfehlungskomponente zur Modellierungsunterstützung von Geschäftsprozessen

Beim Entwurf von Geschäftsprozessen müssen sich Modellierer an Restriktionen des Unternehmens orientieren. Die Geschäftsprozesse müssen den vom Unternehmen definierten Geschäftsregeln entsprechen. In diesem Kapitel wird eine weitere Unterstützung während der Geschäftsprozessmodellierung auf Basis einer Empfehlungskomponente vorgestellt. Zunächst berücksichtigt die Empfehlungskomponente beim Vorschlag von passenden Folgeprozessen ausschließlich Geschäftsregeln. Im zweiten Schritt wird die Komponente dahingehen erweitert, dass auch unterschiedliche Begrifflichkeiten für gleiche Prozessobjekte und Geschäftsregeln berücksichtigt werden. Folgeprozesse, die zum editierten Geschäftsprozess sehr ähnliche Prozessaktivitäten verwenden und auch Geschäftsregeln entsprechen, werden bei der Empfehlung ebenfalls angezeigt. Zuletzt wird das Abstraktionsniveau der editierten Prozessaktivitäten beim Vorschlag passender Folgeprozesse beachtet.

6.1 Geschäftsregeln

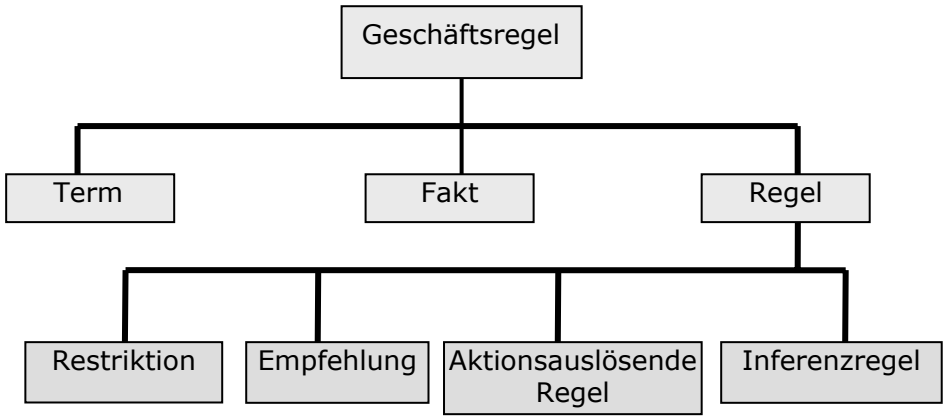
Geschäftsregeln repräsentieren die Unternehmenspolitik, unternehmerisches Wissen und Fachkenntnisse. Die Business Rule Group [BRG] definiert eine Geschäftsregel "*as a statement that defines or constrains some aspect of the business, and it is intended to assert business structures or to control or influence the behaviour of the business*". Gewöhnlich werden Geschäftsprozesse entsprechend bestimmter Geschäftsregeln modelliert, z.B. *falls ein Kunde fünf Schadensfälle gemeldet hat oder die Schadenssumme einen Wert von 2000 Euro in einem Jahr überschreitet, dann wird sein Vertrag gekündigt.*

6.1.1 Klassifikation von Geschäftsregeln

In der Literatur finden sich verschiedene Klassifikationsschemata für Geschäftsregeln. Von Halle [Hall02] beschreibt ein abstraktes Klassifikationsschema für Geschäftsregeln, das aus einem Term, einem Fakt und einer Regel besteht (siehe Ab-

bildung 46). Ein Term kann ein Konzept (Kunde), eine Eigenschaft (treuer Kunde) oder ein Wert (weiblich) eines Konzepts sein. Fakten sind Aussagen, die Terme durch Präpositionen und Verben verbinden. Beispiele für Fakten sind *Kunde kann Bestellung vornehmen* oder *Kunde kann Kredit aufnehmen*. Eine Regel ist eine deklarative Aussage, die eine Berechnung von Informationseinheiten unterstützt und kann klassifiziert werden in eine Regel, die Restriktionen (*Wenn der Kunde unter 18 Jahre alt ist, dann darf er keine Bestellung abgeben*) oder Empfehlungen vorgibt (*Ein Kunde sollte zu einem Zeitpunkt nicht mehr als 10 offene Bestellungen haben*), Aktionen oder (*Wenn die Bestellung eingetroffen ist, dann kann die Bestellung verarbeitet werden*) Inferenzen auslöst (*Wenn zwei Vollzahler eine Reise gebucht haben, dann zahlt die dritte Person nur die Hälfte*).

Abbildung 46: Klassifikationsschema für Geschäftsregeln



Unter regelbasierter Modellierung können „Ereignis-Bedingung-Aktion"-Regeln (Event-Condition-Action) für Datenbanksysteme [GaDi93] oder logikbasierte Sprachen wie Prolog [ClMe87] und Ansätze für Regelfunktionseinheiten (rule engines) wie Jess [Golb04] subsumiert werden. Die meisten Regelsprachen unterstützen die logischen Operatoren AND ($\wedge$) und OR ($\vee$). Manche Regelsprachen bieten auch eine Negation (NOT).

Für unser Anwendungsszenario einer Unterstützung während der Prozessmodellierung werden die vier Arten von Geschäftsregeln zu drei Regeltypen subsumiert:

- *Einschränkende Regeln* (äquivalent zu Restriktionen): es werden zwei Arten von einschränkenden Regeln unterschieden. Zum einen kann sich dieser Regeltyp auf die Integrität von (semantischen) Geschäftsprozessmodellen beziehen (siehe Kapitel 2.3.1) und gilt für jedes (semantische) Geschäftsprozessmodell. Beispielsweise muss auf eine Stelle immer eine Transition folgen und umgekehrt. Die Gültigkeit

dieses Regeltyps ist erfüllt, wenn der modellierte Prozess mit den vorgegebenen Einschränkungen übereinstimmt. Anderenfalls wird diese Regel verletzt. Einschränkende Regeln gewährleisten die Integrität von Geschäftsprozessen bei einer Modellierungsunterstützung.
Zum anderen können mit diesem Regeltyp Restriktionen beschrieben werden: *Kunden, die unter 18 sind, dürfen keine Online-Reservierung vornehmen*.

- *(Ereignis-)Bedingung-Aktion-Regeln* (äquivalent zu Aktionsauslösenden Regeln): bei diesem Regeltyp müssen bestimmte Bedingungen erfüllt sein, bevor Aktionen ausgeführt werden können. Dabei kann die Überprüfung der Gültigkeit von Bedingungen von Ereignissen abhängen. Diese Regeln werden vor der Geschäftsprozessmodellierung definiert und in einem Repository für eine spätere Anwendung gespeichert.
  Die Syntax einer Ereignis-Bedingung-Aktion-Regel[59] sieht wie folgt aus:

```
INITIATION <event 1> OR <event 2> OR .. OR <event n>
IF <condition 1> AND60 <condition 2> AND .. AND <condition n>
THEN <action 1> AND <action 2> AND .. AND <action 3>
```

  Der Ereignisteil, der aus *1* bis *n* Ereignissen bestehen kann, beschreibt den Auslöser der Regel und ist durch OR-Verknüpfungen verbunden; d.h. ein Ereignis reicht aus, um die Regel auszulösen. Der Bedingungsteil spezifiziert *1* bis *n* Bedingungen, die erfüllt sein müssen, bevor eine Aktion ausgeführt werden kann. Im Fall von mehreren Bedingungen werden Bedingungen durch den logischen AND-Operator verbunden. Die Erfüllung von Bedingungen löst eine oder mehrere Aktionen von Folgeprozessen aus, die durch AND-Verknüpfungen verbunden sind. Aktionen können wiederum neue Bedingung-Aktion-Regeln auslösen.
  Die INITIATION- und IF-Teile sind für den aktuell zu modellierenden Geschäftsprozess relevant und der THEN-Teil beschreibt Aktionen, die Folgeprozesse (die zur Vervollständigung vorgeschlagen werden) beinhalten müssen. Der THEN-Teil schränkt die Menge an passenden Prozessfragmenten ein.
  Dabei ist der INITIATION-Teil optional, da die Regel

```
INITIATION <event 1> OR[61] <event 2> OR … OR <event n>
IF <condition 1> …
```

[59] Eine regelbasierte automatische Anwenderunterstützung und deren Implementierung wurden nur für Regeln in der englischen Sprache vorgenommen, weshalb die Syntax in Englisch erklärt wird.
[60] Bedingungen mit OR-Verbindungen (exklusives Oder) können über IF a OR b THEN c ≡IF a THEN c, IF b THEN c ausgedrückt werden.
[61] Das OR ist ein exklusives Oder (XOR).

umgeschrieben werden kann zu:

```
a) IF <event 1> AND <condition 1> …
b) IF <event 2> AND <condition 1> …
c) …
d) IF <event n> AND <condition 1> …
```

- *Dynamische Regeln* (äquivalent zu Empfehlungen und Inferenzregeln): Regeln gelten üblicherweise nur für einen bestimmten Kontext und müssen für einen anderen Kontext umformuliert bzw. angepasst werden. Einige Regeln benötigen sogar Informationen über Daten eines bestimmten Prozesses. Mit Hilfe von so genannten dynamischen Regeln werden Regeln als Frage umformuliert und an den Benutzer gestellt, der den Realweltausschnitt modelliert. Denn der Modellierer weiß am besten, ob bestimmte Kriterien zur Laufzeit erfüllt werden können. Die Regel *wenn ein hoher Schaden vorliegt, dann muss ein Gutachter konsultiert werden* wird zu der Frage umformuliert *liegt hier ein hoher Schaden vor?* oder *was ist ein hoher Schaden?*
  Die Syntax von dynamischen Regeln sieht wie folgt aus:

```
IF QUERY natural language query
THEN <action 1> AND <action 2> AND … AND <action n>
```

Der THEN-Teil schränkt wiederum die Menge an passenden Folgeprozessen ein.

Am Beispiel der folgenden drei Regeln wird in den nächsten Unterabschnitten die Konzeption und Realisierung einer ähnlichkeitsbasierten Empfehlungskomponente für Geschäftsprozesse erklärt:

```
1) IF customer order is checked
   THEN manufacture item AND send article
2) IF material ordered AND
   QUERY Is the material price greater than 100 Euro?
   THEN approve material
3) IF QUERY Is inventory of article X below its minimal inventory?
   THEN reorder article X from supplier
```

Im Falle der Gültigkeit dieser drei Regeln, würde das Unterstützungssystem Prozessfragmente vorschlagen, die mit der Herstellung und Versendung von Artikeln (Regel 1) und der Frage an den Benutzer, ob der Materialpreis höher als 1000 Euro

ist (Regel 2), zu tun haben. Regel 3 ist eine Kombination einer ECA und einer dynamischen Regel.

6.1.2 Konsistenz von Geschäftsregeln

Konsistente Geschäftsregeln sind gültige Regel. Bei der Realisierung der Empfehlungskomponente muss allerdings beachtet werden, dass Geschäftsregeln auch inkonsistent sein können. Inkonsistenzen können in Geschäftsregeln auf zweierlei Weise entstehen [Hall02]:

- zwei Regeln haben denselben IF-Teil, führen aber zu einem unterschiedlichen THEN-Teil
- zwei Regeln haben einen unterschiedlichen IF-Teil, führen aber zu demselben THEN-Teil

Zum Auffinden von inkonsistenten Geschäftsregeln könnte eine Tabelle mit der Menge aller für bestimmte Geschäftsprozesse annotierter Geschäftsregel erstellt werden. Tabelle 10 zeigt, dass Regel R5 und R10 denselben IF-Teil haben, aber einen unterschiedlichen THEN-Teil. Bei Regel 19 und Regel 23 ist es umgekehrt. Damit handelt es sich um inkonsistente Regeln. Regel R16 ist konsistent, da es keine weitere Geschäftsregel gibt, die einen gleichen IF- oder THEN-Teil als R16 hat.

Tabelle 10: Annotierte Geschäftsregeln für Geschäftsprozessmodelle

Regel-ID	**IF-Teil**	**THEN-Teil**	
R5	*wenn ein hoher Schaden vorliegt*	*dann muss ein Gutachter konsultiert werden*	inkonsistent
R10	*wenn ein hoher Schaden vorliegt*	*dann muss ein Gutachter konsultiert und ein Gutachten erstellt werden*	
R19	*Wenn die Kreditwürdigkeit eines Kunden positiv geprüft wurde*	*dann darf der Kunde eine Versicherung abschließen*	inkonsistent
R23	*Wenn die Kreditwürdigkeit eines Kunden positiv geprüft wurde und er in Deutschland einen ständigen Wohnsitz hat*	*dann darf der Kunde eine Versicherung abschließen*	
R16	*Wenn der Kunde die letzten zwei Jahre keine Versicherungsleistungen in Anspruch genommen hat*	*dann sollen die Versicherungsprozente um 10 % gesenkt werden*	konsistent

Für ein System ist es schwierig zu entscheiden, welche Regel gültig ist. Denkbar wäre, dass der Benutzer bei Annotation von Geschäftsregeln an Prozessmodelle eine Prioritätenliste angibt. Mit Hilfe der Benutzerprioritäten könnten die Geschäftsregeln abgearbeitet werden und passende Folgeprozesse von der Empfehlungskomponente vorgeschlagen werden:

Priorität	**RegelID**
1	R10
2	R23
3	R16

6.2 Konzeption

Die Realisierung einer Empfehlungskomponente erfordert die Berücksichtigung von Geschäftsregeln, die für bestimmte Geschäftsprozessmodelle formuliert wurden. Die Idee der Realisierung dieser Komponente wurden bereits im Kapitel 6.1 erklärt. Das OWL-basierte Beschreibungsformat von Petri-Netzen muss allerdings um entsprechende Regeln erweitert werden, die die Integrität von Petri-Netz basierten Geschäftsprozessen garantieren. Diese Regeln werden in der Semantic Web Rule Language beschrieben.

6.2.1 Semantic Web Rule Language

Die Semantic Web Rule Language (SWRL)[62] [HPBT04] ist eine Kombination aus OWL DL und Unary/Binary Datalog RuleML [BoTW01] und wurde beispielsweise in Jess [Frie01] und in KAON2[63] umgesetzt. In der SWRL-Spezifikation werden Regeln wie folgt definiert: *antecedent* $\rightarrow$ *consequent*, wobei *antecedent* und *consequent* aus 0 bis n Atomen bestehen. Jedes Atom wird durch die Form *C(x)*, *P(x,y)*, *sameAs(x,y)* oder *differentFrom(x,y)* beschrieben, wobei *C* ein OWL-Konzept ist, *P* eine OWL-Eigenschaft und *x*,*y* Variablen oder OWL-Instanzen sein können. Wenn der *antecedent*-Teil eingehalten wird, dann wird auch der *consequent*-Teil eingehalten. Wenn der *antecedent*-Teil leer ist (d.h. immer wahr), dann wird der *consequent*-Teil eingehalten. Wenn der *consequent*-Teil leer ist (d.h. immer falsch), dann darf der *antecedent*-Teil nicht eingehalten werden; dies ermöglicht, Einschränkungen zu formulieren.

[62] http://www.daml.org/2003/11/swrl/
[63] http://kaon2.semanticweb.org/

Zusätzlich gibt es in der SWRL-Spezifikation noch so genannte *customer built-ins*[64] (anwenderdefinierte Ausdrücke), von denen einige aus der XQuery und XPath Spezifikation übernommen wurden. Das built-in swrlb:stringLength kann verwendet werden, um die Länge eines Strings zu berechnen oder swrlb:add, um zwei Zahlen zu addieren. In der nachfolgenden Regel wird das built-in swrlb:greaterThanOrEqual benutzt, welches ausdrückt, dass eine Person mit 18 oder mehr Jahren ein Erwachsener ist:

```
Person(?p) ∧ hasAge(?p, ?age) ∧ swrlb:greaterThanOrEqual(?age, 18)
→ Adult(?p)
```

Der nachfolgende Quelltext zeigt die Einbindung einer SWRL-Regel in das OWL-basierte Beschreibungsformat von Petri-Netzen. Eingeleitet werden SWRL-Regeln mit dem Namensraumpräfix `<swrl:>`. Jeder Regel wird eine ID zugeordnet, die eindeutig sein muss. Die Bestandteile einer SWRL-Regel – in diesem Fall `<first>` – werden im nächsten Abschnitt erklärt.

Quelltext 26: Kombination von semantischen Geschäftsprozessmodellen mit SWRL-Regeln

```
<rdf:RDF ....
 <owl:Class rdf:ID="Transition">
  <rdfs:subClassOf rdf:resource="#PetriNet"/>
   <owl:ObjectProperty rdf:ID="placeRef">
   ....
  <swrl:Imp rdf:ID="Rule-3">
    <swrl:body>
      <swrl:AtomList>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:first>
              <swrl:ClassAtom>
                <swrl:classPredicate rdf:resource="#Place"/>
                <swrl:argument1 rdf:resource="#predecessor"/>
              </swrl:ClassAtom>
            </rdf:first>
            <rdf:rest>
              <swrl:AtomList>
                <rdf:first>
                  <swrl:IndividualPropertyAtom>
                   <swrl:propertyPredicate rdf:resource="#transRef"/>
                    <swrl:argument2 rdf:resource="#origin"/>
                    <swrl:argument1 rdf:resource="#predecessor"/>
                  </swrl:IndividualPropertyAtom>
                </rdf:first>
                <rdf:rest rdf:resource="#nil"/>
              </swrl:AtomList>
            </rdf:rest>
```

[64] http://www.daml.org/2004/04/swrl/builtins

```
          </swrl:AtomList>
        </rdf:rest>

        ....
</rdf:RDF>
```

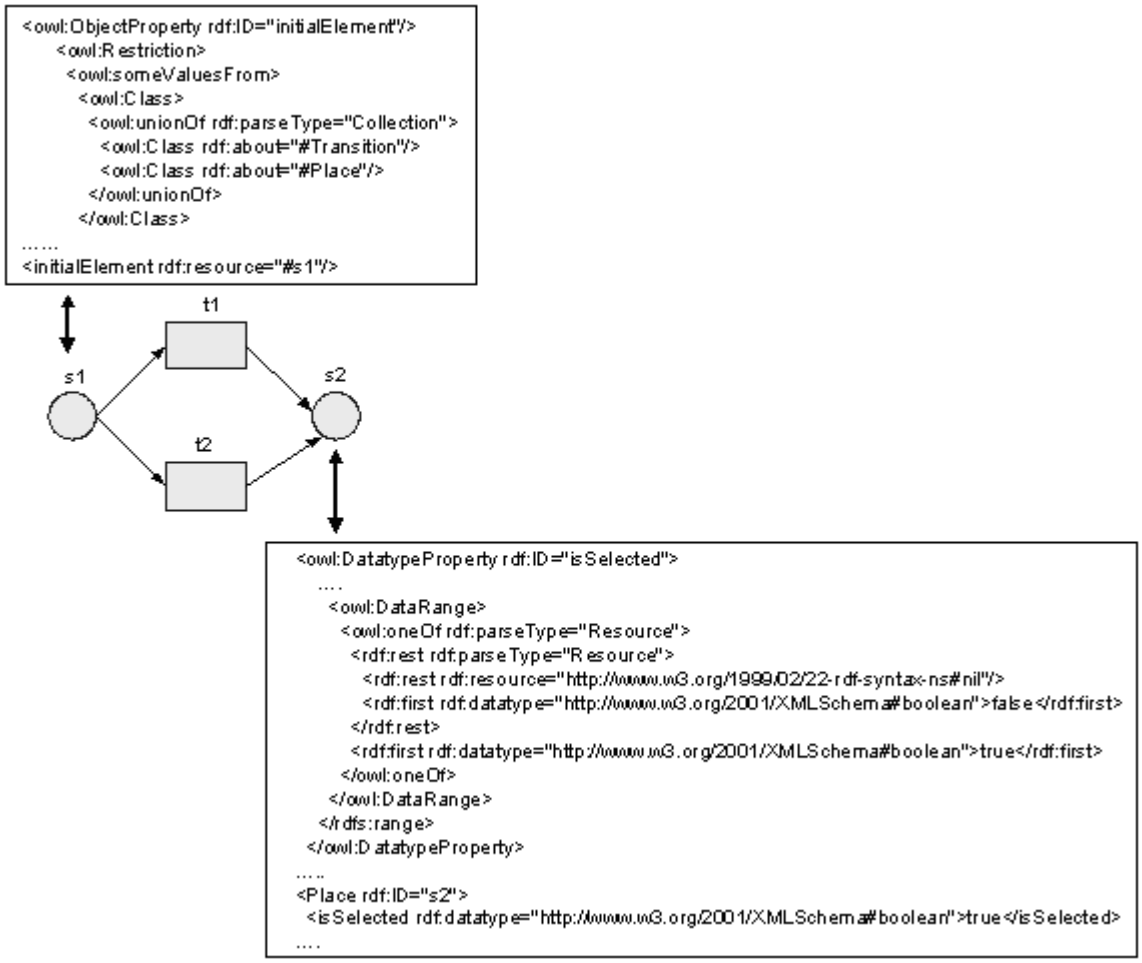
6.2.2 Automatische Geschäftsprozessergänzung

Die drei Regeltypen, wie sie im Abschnitt 6.1.1 erklärt wurden, werden wie folgt realisieren:

1. Einschränkende Regeln → IF THEN Ausdrücke
2. ECA Regeln → IF THEN Ausdrücke
3. Dynamische Regeln → natürliche Sprachanfrage

Bevor allerdings das OWL-basierte Beschreibungsformat von Petri-Netzen als maschineninterpretierbares Format für die Empfehlungskomponente genutzt werden kann, muss es um zwei weitere Eigenschaften erweitert werden [HoKO07]. Die Datentypeigenschaft *isSelected* mit der Domäne *Place* oder *Transition* und dem Wertebereich *boolean* (true or false) soll das Element markieren, ab dem die Empfehlungskomponente aufgerufen werden soll. Die Objekteigenschaft *initialElement* mit der Domäne *PetriNet* und dem Wertebereich *Place* oder *Transition* gibt das Startelement des Geschäftsprozesses an (siehe Abbildung 47; falls ab der Stelle *s2* die Empfehlungskomponente aufgerufen werden würde). Diese Eigenschaften erleichtern ein automatisches Auffinden des markierten Elements. Zusätzlich wird die Klasse *Query* mit den beiden Unterklassen *AskQuery* und *QuerySatisfied* eingeführt, um dynamische Regeln realisieren zu können.

Abbildung 47: Zuordnung von initialElement und isSelected



Für die Realisierung einer Empfehlungskomponente werden die folgenden anwenderdefinierten Ausdrücke definiert. Dabei ist *node* ein Element und *net* ein Folgeprozess. Die inkrementelle Definition von initialen SWRL-Regeln, auf denen die nachfolgenden Prädikate aufbauen, ist im Anhang B angegeben.

1. *beforeOrFirst(?node1*[65]*,?node2)*: Wird eingehalten, wenn *node2* ein (transitiver) Nachfolger von *node1* ist oder *node1* das erste Element eines Folgeprozesses ist,
2. *beforeOrLast(?node1,?node2)*: Wird eingehalten, wenn *node2* ein (transitiver) Nachfolger von *node1* ist oder *node1* das ausgewählte Element im gerade editierten Geschäftsprozess ist,
3. *afterOrFirst(?node1,?node2)*: Wird eingehalten, wenn *node2* ein (transitiver) Vorgänger von *node1* ist oder *node1* das erste Element eines Folgeprozesses ist,

[65] Variablen wird ein Fragezeichen vorangesetzt, z.B. ?x.

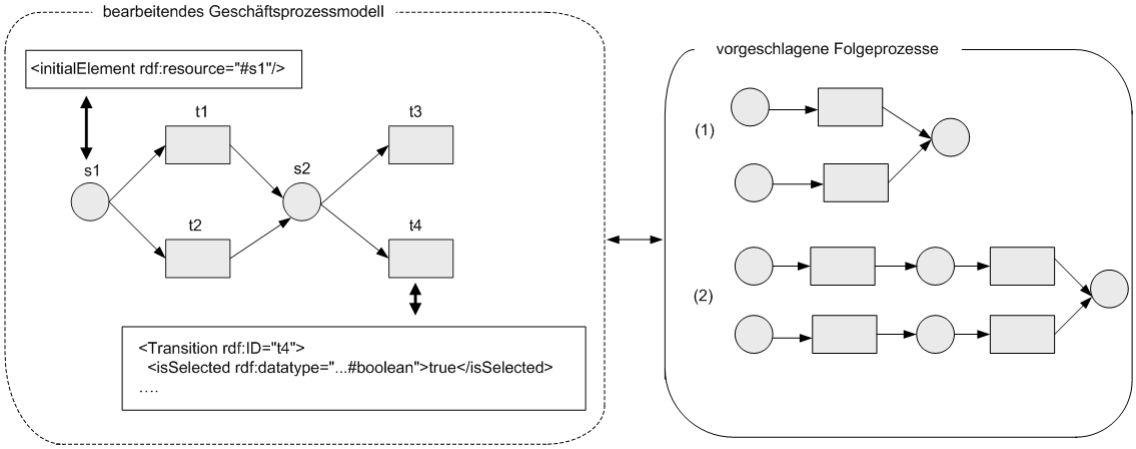
4. *afterOrLast(?node1, ?node2)*: Wird eingehalten, wenn *node2* ein (transitiver) Vorgänger von *node1* ist oder *node1* das ausgewählte Element im gerade editierten Geschäftsprozess ist,

Zusätzlich müssen noch SWRL-Prädikate definiert werden, die Ablaufmuster des editierten und des vorzuschlagenden Prozesses berücksichtigen:

1. *arcPossible(?node,?net)*: Wird eingehalten, wenn es eine syntaktische gültige Verbindung[66] zwischen dem ausgewählten Element (*node*) und dem Folgeprozess (*net*) gibt. D.h., wenn *isSelected* bei einer Stelle ausgewählt wurde, dann wird eine Transition vorgeschlagen und wenn *isSelected* eine Transition ist, dann wird eine Stelle vorgeschlagen.
2. *branching(?node,?net)*: Wird eingehalten, wenn das ausgewählte Element *node* ein Element eines vorwärtsverzweigten Knotens ist und der Folgeprozess *net* eine Rückwärtsverzweigung beinhaltet. D.h., der zusammengesetzte Prozess muss well-structured sein.

Damit kann der Benutzer ein Element auswählen, dass zu einem vorwärtsverzweigten Knoten gehört. So können auch Folgeprozesse vorgeschlagen werden, die nicht nur mit einem Element beginnen (siehe Abbildung 48).

Abbildung 48: Vorschlag von korrekten Folgeprozessen



Algorithmus 1 illustriert, wie eine ECA-Regel (ohne dynamische Regel) bzw. die zweite Art von einschränkenden Regeln in eine SWRL-Regel umgewandelt werden kann. Der INITIATION-Teil kann wie oben beschrieben zu `IF… AND` Klauseln umformuliert werden.

[66] Gültige Verbindungen ergeben sich aus $F \subseteq (S \times T) \cup (T \times S)$ und ungültige aus $F_n \subseteq (S \times S) \cup (T \times T)$

Algorithmus 1 Umwandlung einer ECA Regel (ohne QUERY-Teil) in eine SWRL-Regel

```
1: swrlRule⇐„isSelected(?node1,true)∧initialElement(?net,
             ?node2)∧ arcPossible(?node1,?net)∧
             branching(?node1,?net)"
2: for all conditions cond_i (i∈1..n) that appear in the IF
   part do
3:   swrlRule.append(" ∧ beforeOrLast(cond_i,?node2)")
4: end for
5: for all actions act_i (i∈1..n) that appear in the THEN part
       do
6:   swrlRule.append("∧ afterOrFirst(act_i,?node2)")
7: end for
8: swrlRule.append("→ isPossible(?node1,?net)")
```

Der Algorithmus 1 funktioniert wie folgt:

`1:`

Zunächst muss an einem Knoten (?node1) die Empfehlungskomponente aufgerufen werden (Wert ist *true*). Dann muss es zwischen dem ausgewählten Knoten (?node1) und dem vorzuschlagenden Folgeprozess eine gültige Verbindung geben (arcPossible(?node1,?net)), die well-structured ist (branching(?node1,?net)).

`2 bis 4:`

Wenn diese Regel erfüllt ist, dann werden für alle Bedingungen im IF-Teil das Prädikat *beforeOrLast* eingefügt und die einzelnen Bedingungen mit dem AND-Operator verknüpft.

`5 bis 7:`

Alle Aktionen im THEN-Teil werden durch den AND-Operator verknüpft und das Prädikat *afterOrFirst* wird eingefügt.

`8:`

Dem *antecedent*-Teil wird der *consequent*-Teil *isPossible(?node1,?net)* angehängt.

Die Geschäftsregel 1 aus Kapitel 6.1.1 wird zu der folgenden SWRL-Regel umgewandelt:

isSelected(?node1,true) ∧ *initialElement(?net,?node2)* ∧ *arcPossible(?node1,?net)* ∧ *branching(?node1,?net)* ∧ *beforeOrLast (customer_order_checked,?node2)* ∧ *afterOrFirst(manufacture_item,?node2)* ∧ *afterOrFirst(send_article,?node2)* → *isPossible(?node1,?net)*

Die Kombination von ECA- und dynamischen Regeln ist im Algorithmus 2 abgebildet. Hier müssen unterschiedliche SWRL-Regeln generiert werden; zunächst müssen die Fragen getriggert werden, wenn zusätzliche Bedingungen ebenfalls erfüllt werden müssen (im Algorithmus als swrlRule[0] modelliert). Dabei werden alle Anfragen als Individuen der Klasse *Query* beschrieben. Wenn ein bestimmtes (kontextabhängiges) Kriterium erfüllt ist, werden die Individuen, die die Anfrage repräsentieren, auch dem Typ *AskQuery* zugeordnet. Die vom Typ *AskQuery* stammenden Individuen können durchsucht und als Fragen an den Benutzer weitergeleitet werden. Alle Fragen, die vom Benutzer bejaht werden, werden auch als entsprechende Individuen vom Typ *QuerySatisfied* ausgedrückt. Die Realisierung über diese Klassen ermöglicht, die Umsetzung von ECA-Regeln bzw. des zweiten Typs von einschränkenden Regeln mit dynamischem Teil.

Algorithmus 2 Umwandlung einer ECA-Regel in eine SWRL-Regel

```
1: swrlRule ⇐ newString[2]
2: swrlRule[0] ⇐ „isSelected(?node1,true) ∧
                initialElement(?net,?node2) ∧ arcPossi-
                ble(?node1,?net) ∧ branching(?node1,?net)"
3: swrlRule[1] ⇐ „isSelected(?node1,true) ∧
                initialElement(?net,?node2) ∧ arcPossi-
                ble(?node1,?net) ∧ branching(?node1,?net)"
4: for all conditions cond_i (i∈1..n) that appear in the IF
   part do
5:    if cond_i is a QUERY then
6:      swrlRule[1].append("∧QuerySatisfied(natural_language
                          _query)")
7:    else
8:      swrlRule[0].append("∧ beforeOrLast(cond_i ,?node2)")
9: end if
10: end for
11:for all actions act_i (i∈1..n) that appear in the THEN part
       do
12:   swrlRule[0].append("∧ afterOrFirst(act_i ,?node2)")
13:   swrlRule[1].append("∧ afterOrFirst(act_i ,?node2)")
13: end for
14: swrlRule[0].append("→AskQuery(natural_language_query)")
15: swrlRule[1].append("→ isPossible(?node1,?net)")
```

1:

Beim Algorithmus 2 werden zwei Regeltypen unterschieden ([0] und [1]).

2:

Für Regeltyp 0 muss zunächst an einem Knoten (?node1) die Empfehlungskomponente aufgerufen werden (Wert ist *true*). Dann muss es zwischen dem ausgewählten Knoten (?node1) und dem vorzuschlagenden Folgeprozess eine gültige Verbindung geben (arcPossible(?node1,?net)), die well-structured ist (branching(?node1,?net)).

3:

Für Regeltyp 1 muss zunächst an einem Knoten (?node1) die Empfehlungskomponente aufgerufen werden (Wert ist *true*). Dann muss es zwischen dem ausgewählten Knoten (?node1) und dem vorzuschlagenden Folgeprozess eine gültige Verbindung geben (arcPossible(?node1,?net)), die well-structured ist (branching(?node1,?net)).

4 bis 10:

Der IF-Teil kann nun aus einem Query-Teil oder einer „statischen" Bedingung bestehen. Im Falle eines Query-Teils wird die Anfrage dem Prädikat *QuerySatisfied* hinzugefügt. Ansonsten wird wie im Algorithmus 1 verfahren.

11 bis 13:

Aktionen wird sowohl für den Regeltyp 0 als auch für den Regeltyp 1 das Prädikat *afterOrFirst* hinzugefügt. Bedingungen werden mit dem Prädikat *beforeOrLast* umschlossen.

14:

Für den Regeltyp 0 gibt es einen consequent-Teil *AskQuery(natural_language_query)*

15:

Für den Regeltyp 1 gibt es einen consequent-Teil *isPossible(?node1,?net)*

Die Geschäftsregel 3 im Kapitel 6.1.1 wird zur folgenden SWRL-Regel transformiert:

1. *isSelected(?node1,true)* $\wedge$ *initialElement(?net,?node2)* $\wedge$ *arcPossible (?node1,?net)* $\wedge$ *branching(?node1,?net)* $\wedge$ *afterOrFirst (reorder_article_supplier,?node2)*
   $\rightarrow$ *AskQuery(Is_inventory_of_article_X_below_its_minimal_inventory)*
2. *isSelected(?node1,true)* $\wedge$ *initialElement(?net,?node2)* $\wedge$ *arcPossible (?node1,?net)* $\wedge$ *branching(?node1,?net)* $\wedge$ *QuerySatisfied (Is_inventory_of_article_X_below_its_minimal_inventory)* $\wedge$ *afterOrFirst(reorder_article_supplier,?node2)*
   $\rightarrow$ *isPossible(?node1,?net)*

Eine sinnvolle (semi-) automatische Übersetzung von natürlichsprachlich beschriebenen Regeln nach SWRL setzt voraus, dass sich Modellierer bei der Regelformulierung an die IF…THEN Syntax halten.

6.3 Ähnlichkeitsbasierte Empfehlungskomponente

Aufbauend auf Algorithmus 2 und 3 wird in den nächsten Unterabschnitten die Berücksichtigung der Semantik von Prozesselementnamen und Geschäftsregeln und des Abstraktionsniveaus bei der Realisierung einer Empfehlungskomponente beschrieben.

6.3.1 Berücksichtigung von Geschäftsregeln

Bei der Realisierung der Empfehlungskomponente muss neben Geschäftsregeln berücksichtigt werden, dass kein kontrolliertes Vokabular zur Beschreibung von Geschäftsregeln und Prozessen vorausgesetzt werden kann. Die Empfehlungskomponente sollte synonym gebrauchte Begriffe aufdecken und bei der Vervollständigung berücksichtigen.
Eine semantisch ähnliche Geschäftsregel könnte syntaktisch mit unterschiedlichen Begriffen ausgedrückt werden:

1. `IF purchase order was handled`
   `THEN initiate payment`
2. `IF order was managed`
   `THEN pay invoice`

Des Weiteren könnte das im Geschäftsprozess verwendete Vokabular ebenfalls syntaktisch unterschiedlich sein. Die Ähnlichkeitsmaße für Prozesselementnamen, wie sie im Kapitel 4.6 definiert wurden, können verwendet werden, um Synonyme, Homonyme und Begriffe mit unterschiedlichem Abstraktionsniveau in Regelbeschreibungen und Geschäftsprozessen zu finden. Diese semantische Ähnlichkeit wird im Reasoningprozess durch den anwenderdefinierten Ausdruck *swrl:semSimilarity (?node1, ?node2, ?arg)* berücksichtigt. Dieses Prädikat wird als wahr evaluiert, wenn die semantische (bzw. kombinierte) Ähnlichkeit von *node1* und *node2* über einer vordefinierten Schwelle (z.B. Ähnlichkeitswert >= 0.4) liegt. Algorithmus 3 zeigt die Umwandlung von ECA-Regeln (ohne dynamische Regel) bzw. des zweiten Typs von einschränkenden Regeln in eine SWRL-Regel unter Berücksichtigung von

Ähnlichkeitsmaßen[67]. Für die Berechnung der semantischen Ähnlichkeit zwischen Begriffen für Geschäftsregeln und Geschäftsprozessen wird der Fakten-Teil von Regeln herangezogen.

Algorithmus 3 Umwandlung einer ECA Regel (ohne QUERY-Teil) in eine SWRL-Regel

```
1: swrlRule⇐„isSelected(?node1,true)∧initialElement(?net,
             ?node2)∧ arcPossible(?node1,?net)∧ branching
             (?node1,?net)"
2: for all conditions cond_i (i∈1..n) that appear in the IF
   part do
3:  swrlRule.append("∧ beforeOrLast(?node_condition_i ,?node1)")
4:  swrlRule.append("∧ semSimilarity(?node_condition_i , cond_i ,0.4)")
5: end for
6: for all actions act_i (i∈1..n) that appear in the THEN
    part do
7:   swrlRule.append("∧ afterOrFirst(?node_action_i ,?node2)")
8:   swrlRule.append("∧ semSimilarity(?node_action_i , act_i , 0.4)")
9: end for
10: swrlRule.append("→ isPossible(?node1,?net)")
```

1:

Zunächst muss an einem Knoten (?node1) die Empfehlungskomponente aufgerufen werden (Wert ist *true*). Dann muss es zwischen dem ausgewählten Knoten (?node1) und dem vorzuschlagenden Folgeprozess eine gültige Verbindung geben (arcPossible(?node1,?net)), die well-structured ist (branching(?node1,?net)).

2 bis 5:

Wenn diese Regel erfüllt ist, dann werden für alle Bedingungen im IF-Teil das Prädikat *beforeOrLast* und nach Bedingungen, deren Ähnlichkeit >=0.4 ist, gesucht. Falls eine solche Bedingung gefunden wird, dann wird der anwenderdefinierte Ausdruck *semSimilarity* eingefügt und die einzelnen Bedingungen mit dem AND-Operator verknüpft.

6 bis 9:

Wenn Aktionen im THEN-Teil gefunden werden, deren Ähnlichkeitswert >=0.4 ist, dann werden die Aktionen durch den AND-Operator verknüpft und die Prädikate *afterOrFirst* und *semSimilarity* werden eingefügt.

10:

[67] Die Suche nach synonym gebrauchten Bedingungen und Aktionen kann auch für Geschäftsregeln mit dynamischem Teil (siehe Algorithmus 2) angewendet werden. Der Algorithmus 2 kann analog zum Algorithmus 3 erweitert werden.

Dem *antecedent*-Teil wird der *consequent*-Teil *isPossible(?node1,?net)* angehängt.

Angenommen, für den zu modellierenden Prozess gilt die Geschäftsregel `IF purchase order was handled then initiate payment`. Der aktuell editierte Geschäftsprozess besteht aus vier Elementen: *order received*, *manage order, order managed* und *inform customer*. Beim Element *inform customer* ruft der Benutzer die Empfehlungskomponente auf. Dann wird mit Hilfe des Ähnlichkeitsalgorithmus die kombinierte Ähnlichkeit zwischen *purchase_order_handled* und den vier Elementen berechnet. In der Ergebnisliste wird *purchase_order_handled* und *order_managed* mit einem Ähnlichkeitswert von 0.8 angegeben (der größer als die vordefinierte Schwelle von 0.4 ist). Damit ist der Bedingungsteil der Regel erfüllt. Anschließend wird nach Regeln gesucht, die den Aktionsteil erfüllen.
Ausgehend von Algorithmus 3 ergibt sich die folgende SWRL-Regel für das obige Szenario:

$$isSelected(?node1, true) \wedge initialElement(?net,?node2) \wedge arcPossible\ (?node1, ?net) \wedge branching(?node1,?net) \wedge beforeOrLast\ (?\text{purchase\_order\_handled}, node1) \wedge afterOrFirst(?\text{initiate\_payment}, ?node2) \wedge semSimilarity(?\text{node}_{\text{condition}_i}, purchase\_order\_handled, 0.4) \wedge semSimilarity(?\text{node}_{\text{action}_i}, initiate\_payment, 0.4) \rightarrow isPossible(?node1,?net)$$

Damit wird bei der Überprüfung des gerade editierten Geschäftsprozessmodells nach semantisch ähnlichen Begriffen zu *purchase_order_handled* gesucht und als Folgeprozesse werden Teile vorgeschlagen, die eine semantische Ähnlichkeit (größer 0.4) zu *initiate_payment* haben.
Der Vorteil von Algorithmus 3 (der ebenfalls auf Geschäftsregeln mit dynamischem Regelteil angewendet werden kann) ist, dass nicht nur nach syntaktisch gleichen Begriffen gesucht wird, sondern auch nach semantisch ähnlichen. Damit ist der Modellierer nicht an ein kontrolliertes Vokabular gebunden. Zudem wird auch Zeit gespart, die notwendig wäre, um inhaltlich ähnliche Geschäftsregel zu finden, die mit einer unterschiedlichen Syntax beschrieben wurden.
Ein weiterer Vorteil von Algorithmus 3 ist, dass mit dem Ähnlichkeitswert gleichzeitig eine Art Metrik definiert wird, so dass nur zum editierten Geschäftsprozess korrekte und passende Folgeprozesse vorgeschlagen werden.

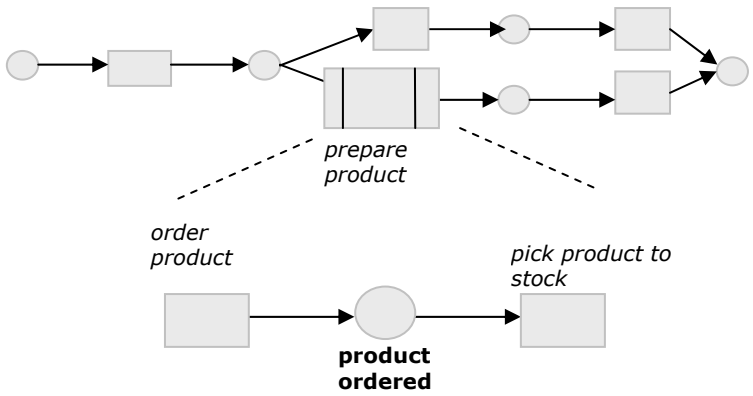
6.3.2 Berücksichtigung des Abstraktionsgrades

Als weiteres sollte bei der Realisierung einer Empfehlungskomponente die Einhaltung eines gleichen Abstraktionsniveaus berücksichtigt werden [KoBl07]. Die von der Empfehlungskomponente vorgeschlagenen Folgeprozesse sollten den gleichen Abstraktionsgrad haben, wie das gerade editierte Geschäftsprozessmodell, um die Konsistenz und Modularität des Geschäftsprozessmodells zu gewährleisten.

Geschäftsprozesse können entweder top-down oder bottom-up modelliert werden. Die Vorgehensweise hängt von Informationen ab, die bei der Modellierung des Realweltausschnitts vorliegen. Falls der Modellierer bereits über sehr viele Informationen verfügt, dann eignet sich das bottom-up Verfahren, bei dem im Detail modellierte Geschäftsprozesse zu abstrakteren Aktivitäten vergröbert werden. Beim top-down Verfahren beginnt die Modellierung auf einem abstrakten Niveau. Geschäftsprozesse können detaillierter durch Verfeinerung von Prozessaktivitäten (Subprozesse) modelliert werden. Die Verfeinerung von Aktivitäten sollte allerdings nicht bis auf Beschreibung von Aktivitäten wie Kundennummer 123 heruntergebrochen werden (eine Auftragsnummer ist keine sinnvolle Aufgabe, die erledigt werden soll). Durch Verfeinerung von Aktivitäten können Prozessaktivitäten im Subprozess auf drei unterschiedliche Weisen beschrieben werden, die nachfolgend als Verfeinerungsmuster eingeführt werden:

(1) Die Prozesselementnamen haben gleiche Substantive[68]. In Abbildung 49 wird das Substantiv *product* für die Beschreibung aller Prozesselemente im Subprozess wieder verwendet)

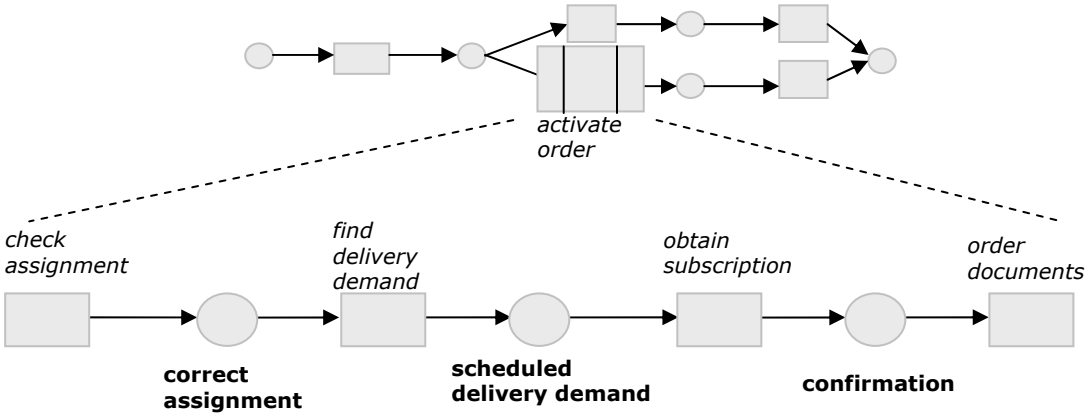
Abbildung 49: Vefeinerungsmuster „Gleiches Substantiv"



(2) Die Prozesselementnamen im Subprozess haben unterschiedliche Substantive im Gegensatz zur Prozessaktivität, die verfeinert wird:

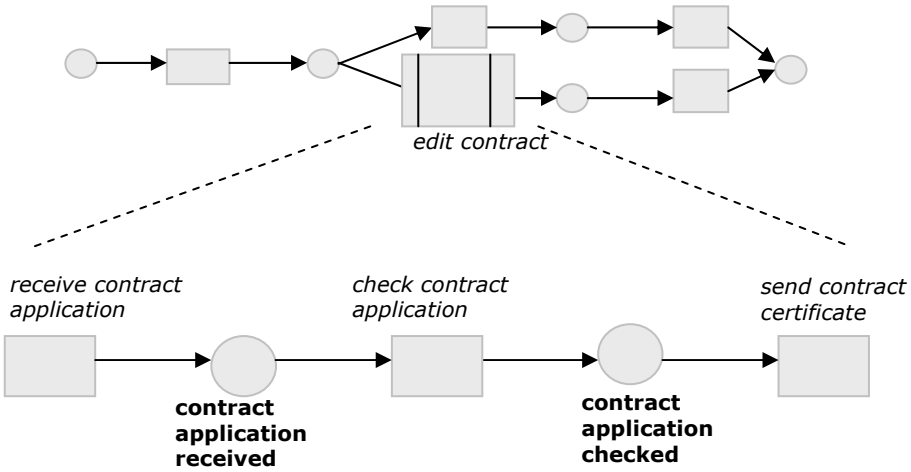
[68] Substantive beschreiben Objekte, die im Prozessmodell modelliert werden. Das können Datenressourcen oder Dokumente sein, die das Ziel oder die Strategie des Prozessmodells wiedergeben.

Abbildung 50: Verfeinerungsmuster „Ungleiche Substantive"



(3) Die Prozesselementnamen im Subprozess sind eine Spezialisierung der zu verfeinernden Prozessaktivität. In Abbildung 51 ist der Begriff *contract application* und *contract certificate* eine Spezialisierung des Begriffs *contract*.

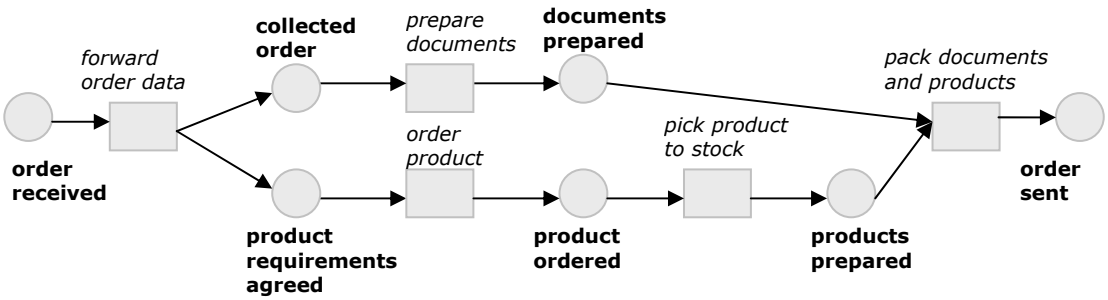
Abbildung 51: Verfeinerungsmuster „Spezialisierung von Substantiven"



Zur Berechnung des Abstraktionsgrades von Prozesselementnamen werden der editierte Geschäftsprozess und ein entsprechender Folgeprozess zunächst zu einem Geschäftsprozess integriert. Der Integrationsprozess lehnt sich an die Vorgehensweise aus [Laus88] an.

Im nächsten Schritt werden die Ablaufmuster (siehe Kapitel 2.2) des integrierten Geschäftsprozesses berücksichtigt, die sequentiell, parallel, alternativ oder iterativ sein können. Der Algorithmus wird eine Verzweigung als einen Anhaltepunkt (Breakpoint) betrachten. Falls der integrierte Geschäftsprozess ausschließlich aus sequentiellen Ablaufmustern besteht, dann gibt es keinen Anhaltepunkt. Bei einer Verzweigung wird im oberen und im unteren Pfad nach Verfeinerungsmustern gesucht. Dann wird der entsprechende Pfad (branch) B ($B \in B_1,..,B_n$) als ein Zweig betrachtet, wenn er aus mehr als vier Elementen $e_1,\ldots,e_n$ aus E_B besteht. Das gleiche gilt bei nur einem sequentiellen Ablaufmuster. Der Geschäftsprozess muss aus mindestens vier Elementen bzw. zwei Transitionen bestehen, damit das Abstraktionsniveau analysiert werden kann. Abbildung 52 zeigt einen integrierten Geschäftsprozess. Der Modellierer hat an der Stelle *forward order data* die Empfehlungskomponente aufgerufen. Daraufhin wurde als Folgeprozess das hintere Stück des Geschäftsprozesses gefunden. Zur Analyse des Abstraktionsgrades fängt der Algorithmus bei der Stelle *order received* an und hält bei der Verzweigung der Transition *forward order data* an. Allerdings besteht die Sequenz nur aus zwei Elementen, weshalb der Algorithmus weiter traversiert.

Abbildung 52: Integrierter Geschäftsprozess (Aufdeckung Verfeinerungsmuster 1)



Nun wird der Zweig *collected order data* bis *documents prepared* mit den beiden Elementen *order received* und *forward order data* isoliert und es wird nach Verfeinerungsmustern gesucht. Als Referenz zur Aufdeckung von Verfeinerungsmustern wird die linguistische Ausprägung der Elemente berechnet. Die im Kapitel 4 vorgestellten Ähnlichkeitsmetriken und -maße für Geschäftsprozessmodelle können nicht verwendet werden, da nicht eine relative Ähnlichkeit, sondern eine linguistische Ausprägung für einen einzelnen Prozesselementnamen berechnet werden soll:

$$\text{specificity}(c_i) = \frac{-\log P(c_i)}{\max_{c_x \in C}\{-\log P(c_x)\}}$$

$$\text{wobei } P(c_i) = \begin{cases} 1 & \text{wenn } c_i = c_o \\ \dfrac{P(c_j)}{\left|\{c_x, c_x \subseteq c_j\}\right|} & \text{wenn } c_i \subseteq c_j \end{cases}$$

Die Ergebnisse aus der Berechnung der linguistischen Ausprägung zeigt Tabelle 11. Der erste Zweig fängt bei *order received* an und endet bei *documents packed* und der zweite Zweig beginnt bei *order received* und hört bei *products prepared* auf.

Tabelle 11: Linguistische Ausprägung für das Geschäftsprozessmodell aus Abbildung 52

Prozesselementname	**Substantiv**	**Zusammengesetzte Substantive**	**Verb**
order received	0.352	-	0.392
forward order data	-	0.352; 0.235	0.407
collected order data	-	0.352; 0.235	0.204
prepare documents	0.406	-	0.260
documents packed	0.406	-	0.260
order received	0.352	-	0.392
forward order data	-	0.352; 0.235	0.407
product requirements agreed	-	0.362; 0.347	0.444
order product	0.362	-	0.352
product ordered	0.362	-	0.352
pick product to stock	0.362; 0.423	-	0.203
products prepared	0.362	-	0.359

Zunächst wird nach Verfeinerungsmuster 1 gesucht. Wenn in beiden Zweigen ein Verfeinerungsmuster gefunden wird, dann hat der gesamte Geschäftsprozess einen gleichen Detaillierungsgrad. Wenn allerdings nur in einem Zweig ein Verfeinerungsmuster gefunden wird, dann gibt es einen unterschiedlichen Abstraktionsgrad. Nachfolgend ist der Algorithmus zur Aufdeckung des Verfeinerungsmusters 1 gezeigt:

Algorithmus 1 zum Auffinden von Verfeinerungsmuster 1

Input: $B_1,\ldots,B_n$ seien eine Menge von Sequenzen mit individuellen Elementmengen E_{B_i} ; $i \in 1\ldots n$.

$\sigma(e_1),\ldots, \sigma(e_n)$ sei die linguistische Ausprägung für alle Elemente in E_{B_i} bei denen das erste Element einer Sequenz den Vorbereich e_{l_p} in einer Menge von Elementen E_B hat.

Output: Elemente, die zum Verfeinerungsmuster 1 gehören

Method: Führe folgende Schritte aus:

for all $E \in E_{B_1,\ldots,} E_{B_n}$ **do**
  if $|E| \geq 4 \wedge \sigma(e_1) = \ldots = \sigma(e_n),\ e_1,\ldots,\ e_n \in E_B$
    gehe solange zurück bis $\sigma(e_1) \neq \sigma(e_{l_p})$ und hebe die inkonsistenten Namen hervor
  else
    fahre fort
  end if
end for

Nach Berechnung der linguistischen Ausprägung für jedes Prozesselement wird nach zusammenhängenden Prozesselementen gesucht, die eine höhere linguistische Ausprägung als der Vor- und Nachbereich der zusammenhängenden Elementgruppe aufweist. Falls die Elementgruppe eine niedrigere linguistische Ausprägung als der Vor- und Nachbereich hat, dann könnte die zusammenhängende Elementgruppe auch vergröbert werden.

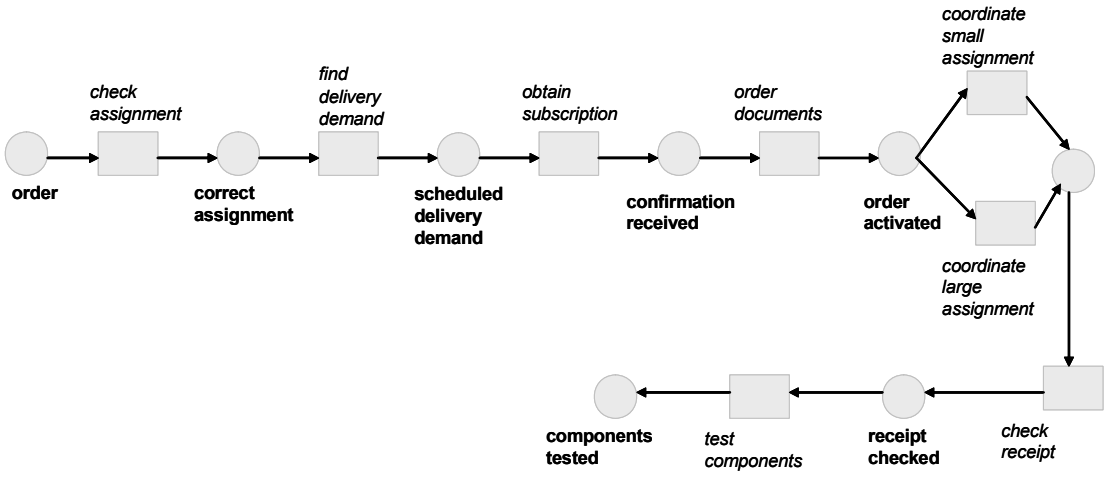
Betrachtet man den oberen Zweig für den Geschäftsprozess aus Abbildung 52, so gibt es fünf Elemente. Zwei Elementgruppen (prepare documents; documents prepared) und (forward order data; collected order data) haben die gleiche linguistische Ausprägung. Allerdings handelt es sich bei *forward order data* und *collected order data* um zusammengesetzte Substantive, die detaillierter sind als ein einzelnes Substantiv und deswegen diese beiden Elementgruppen nicht als eine zusammenhängende Prozesselementgruppe betrachtet werden. Im unteren Zweig hingegen gibt es eine zusammenhängende Elementgruppe bestehend aus *order product*, *product ordered*, *pick product to stock und products prepared*, da die Elemente die gleiche linguistische Ausprägung von 0.362 besitzen und diese kleiner als 0.352 ist.

Unter Berücksichtigung der beiden Elemente *pack documents and products* und *order sent* folgt das gleiche Ergebnis, nämlich, dass im unteren Zweig das Verfeinerungsmuster 1 gefunden wird und somit der integrierte Geschäftsprozess nicht konsistent wäre, wenn dieser Folgeprozess vorgeschlagen würde.

Die Aufdeckung des Verfeinerungsmusters 2 benötigt mehr Berechnungsaufwand als für Verfeinerungsmuster 1. Der integrierte Geschäftsprozess in Abbildung 53

wird nach der Stelle *order activated* verzweigt. Allerdings besteht jeder Ast dieser Verzweigung aus einem Element, weshalb dieses nicht als ein Zweig, sondern die Verzweigung nur als Anhaltepunkt betrachtet wird.

Abbildung 53: Integrierter Geschäftsprozess (Verfeinerungsmuster 2)



Angenommen, der Benutzer hat ab dem Element *scheduled delivery demand* die Modellierungsunterstützung aufgerufen. Dann wird zunächst die linguistische Ausprägung der Prozesselemente von *order* bis *order activated* bestimmt (Ergebnisse siehe Tabelle 12).

Tabelle 12: Linguistische Ausprägung für das Geschäftsprozessmodell aus Abbildung 53

Prozesselementname	Substantiv	Zusammengesetzte Substantive	Verb
order	-	0.515; 0.360	-
check assignment	0.560	-	0.350
correct assignment	0.560	-	0.430
find delivery demand	-	0.307; 0.225	0.444
Scheduled delivery demand	-	0.307; 0.225	0.444
obtain subscription	0.498	-	0.392
confirmation received	0.432	-	0.392
order documents	0.501	-	0.515
order activated	0.515	-	0.515
coordinate small assigment	0.560	-	0.337
coordinate big assigment	0.560	-	0.337
delivery	0.307		0.337
check receipt	0	-	0
receipt checked		-	0
test components	0.261	-	0.342
components tested	0.342		0.261

Zur Aufdeckung des Verfeinerungsmusters 2 wird nur die linguistische Ausprägung von Transitionen betrachtet, da Transitionsnamen die (aktiven) Aufgaben ausdrücken und Stellennamen passive Element beschreiben, die bereits erledigt wurden. Die Aufdeckung des Verfeinerungsmuster 2 hängt von zwei Faktoren ab: dem Durchschnitt der linguistischen Ausprägungen von Transitionen μ und wie beim Verfeinerungsmuster 1 einer höheren/niedrigeren linguistischen Ausprägung von zusammenhängenden Elementgruppen $e_1,\ldots,e_n$ aus E_B als der Vor- und Nachbereich. Der Algorithmus ist nachfolgend gezeigt:

Algorithmus 2 zum Auffinden von Verfeinerungsmuster 2

Input: $B_1,\ldots,B_n$ seien eine Menge von Sequenzen mit individuellen Elementmengen E_{B_i} ; $i \in$ 1…n.

$\sigma(e_1),\ldots, \sigma(e_n)$ sei die linguistische Ausprägung für alle Elemente in E_{B_i} ; e_{l_P} and $e_{l_{po}}$ seien der Vor- und Nachbereich des ersten und letzten Elements in einer Elementmenge in E_B.
$\mu(\sigma(e_{t_i}))$ sei die durchschnittliche linguistische Ausprägung aller Transitionen in $E_{T_i} \in E_{B_i}$; $i \in$ 1…n.
Output: Elemente, die zu Verfeinerungsmuster 2 gehören
Method: Führe die folgenden Schritte aus:

for all $E \in E_{B_i,\ldots,} E_{B_n}$ **do**
  if $|E| \geq 4$ und $(\sigma(e_{t_i}) > \mu(\sigma(e_{t_i})) \vee \sigma(e_{t_i}) > (\sigma(e_{l_P})) \vee \sigma(e_{n_{po}}))$
    then
      hebe die inkonsistenten Namen hervor
  else
    fahre fort
  end if
end for

Der Durchschnitt der linguistischen Ausprägungen der Transitionen in Abbildung 53 beträgt 0.434. Eine höhere linguistische Ausprägung haben die zusammenhängenden Elemente *check assignment*, *find delivery demand*, *obtain subscription* und *order documents* mit 0.455, 0.488, 0.445 und 0.508. Ansonsten gibt es keine weiteren zusammenhängenden Elementgruppen mit einer höheren linguistischen Ausprägung als die durchschnittliche linguistische Ausprägung aller Transitionen. Damit haben nur die beiden Elemente *obtain subscription* und *confirmation received* des Folgeprozesses den gleichen Abstraktionsgrad wie die Elemente des editierten Geschäftsprozesses. Die übrigen Elemente des Folgeprozesses haben einen anderen Abstraktionsgrad weshalb dieser Folgeprozess auch nicht vorgeschlagen werden sollte.
Das Verfeinerungsmuster 3 kann aufgrund einer höheren linguistischen Ausprägung als der Vor- und Nachbereich von zusammenhängenden Prozesselementgruppen gefunden werden.

Die vorgestellten Algorithmen zur Aufdeckung von Verfeinerungsmustern in integrierten Geschäftsprozessen können über einen zusätzlichen selbstdefinierten SWRL-Ausdruck berücksichtigt werden:

6. *semAbstraction(?node1, ?net):* Wird eingehalten, wenn der Abstraktionsgrad zwischen dem ausgewählten Knoten (an dem die Modellierungsunterstützung aufgerufen wurde) und dem ausgewählten Folgeprozess *net* gleich ist.
Der antecedent-Teil wird damit erweitert um

```
1: swrlRule⇐„isSelected(?node1,true)∧initialElement(?net,
                ?node2)∧ arcPossible(?node1,?net)∧
                branching(?node1,?net)∧
                semAbstraction(?node1,?net)”
```

Algorithmus 3 kann analog dazu erweitert werden.
Somit werden bei der Modellierungsunterstützung folgende Kriterien berücksichtigt:

- Geschäftsregeln; die Geschäftsprozessmodelle werden damit gleichzeitig an die jeweilige Umwelt angepasst.
- Semantisch ähnliche Begriffe für gleiche Prozessobjekte und Geschäftsregeln. Es werden auch Folgeprozesse vorgeschlagen, die nicht mit einem gleichen Vokabular modelliert wurden. Somit wird die Einschränkung des kontrollierten Vokabulars aus der Literatur aufgehoben.
- Abstraktionsniveau. Es werden nur Folgeprozesse mit gleichen Abstraktionsgrad wie das editierende Geschäftsprozessmodell vorgeschlagen. Somit können Modellierungsfehler weitgehend vermieden werden.

Diese drei Funktionalitäten der Empfehlungskomponente helfen Benutzern, Geschäftsprozessmodelle entsprechend den Unternehmensrestriktionen konsistent zu modellieren.
Im nächsten Kapitel werden die Ergebnisse einer Benutzerbefragung zu den im Kapitel 4.6 definierten Ähnlichkeitsmaßen und der Modellierungsunterstützung vorgestellt.

7 Evaluierung von Ähnlichkeitsmaßen und der Modellierungsunterstützung für Geschäftsprozesse

Ausgehend von einer Klassifikation gemäß der benutzten Lernstrategie (Kapitel 7.1) werden in den darauf folgenden Unterabschnitten zwei Verfahren des maschinellen Lernens ausführlich beschrieben. Mit Hilfe dieser Verfahren soll eine passende Gewichtung der Variable *w* für das kombinierte Ähnlichkeitsmaß (siehe Kapitel 4.6.5) und eine Einteilung von Modellieren in Gruppen bestimmt werden können, damit daraus benutzerindividuelle Modellierungsunterstützungen für Geschäftsprozesse abgeleitet werden können. Eine „korrekte" initiale Gewichtung des Parameters *w* soll den vom Menschen eingeschätzten Ähnlichkeitswert approximieren (siehe Diskussion im Kapitel 4.3.5). Als Ziel des maschinellen Lernens formulieren Michalski und Kodratoff [MiKo90] *„Research in machine learning has been concerned with building computer programs able to construct new knowledge or to improve already possessed knowledge by using input information*."

7.1 Klassifikation von Verfahren des maschinellen Lernens

Verfahren des maschinellen Lernens können abhängig vom Anwendungsgebiet unterschiedlich verwendet werden. Carbonell, Michalski und Mitchell [CaMM83] klassifizieren Verfahren des maschinellen Lernens aus drei verschiedenen Sichtweisen:

Klassifikation gemäß der benutzten Lernstrategie

Das Erlernen von Wissen kann durch verschiedene Lernstrategien erreicht werden. Vergleichbar mit menschlichen Lernstrategien können Maschinen durch Anweisungen lernen, indem aufbereitetes Wissen vorgegeben wird. Sie können aber auch durch Deduktion, Analogie oder aus Beispielen lernen. Beim Lernen aus Beispielen soll anhand einer Gegenüberstellung von „guten" und „schlechten" Beispielen gelernt werden.

Klassifikation gemäß dem gelernten Typ von Wissen

Maschinelles Lernen kann verwendet werden, um unterschiedliche Arten von Wissen zu erlernen. Das zu erlernende Wissen können Parameter in algebraischen Ausdrücken, Entscheidungsbäume, formale Grammatiken, Regeln oder Begriffshierarchien sein. Möglich ist auch, dass Graphen und Netzwerke oder Schemata gelernt werden.

Klassifikation gemäß dem Anwendungsbereich

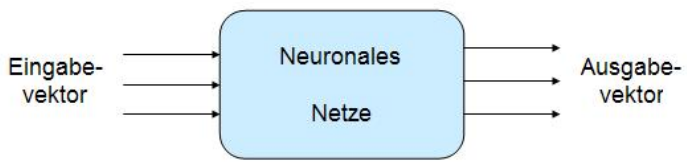
Die Einsatzfelder von Verfahren des maschinellen Lernens sind breit gestreut. Im medizinischen Bereich hilft maschinelles Lernen bei der Assistenz von Operationen oder chirurgischen Eingriffen. In der Bioinformatik wird maschinelles Lernen zur bioinformatischen Datenverarbeitung verwendet. Im Marketing unterstützt maschinelles Lernen die Einteilung des Kundenkonsums zu bestimmten Kundengruppen.

Ausgehend von einer Klassifikation gemäß der benutzten Lernstrategie werden in den nachfolgenden Unterabschnitten zwei Verfahren ausführlich beschrieben: Neuronale Netze und Clusteranalyse. Mit der Clusteranalyse sollen Modellierer in Benutzergruppen eingeteilt werden. Mit neuronalen Netzen soll die initiale Gewichtung *w* gelernt werden.

7.1.1 Neuronale Netze

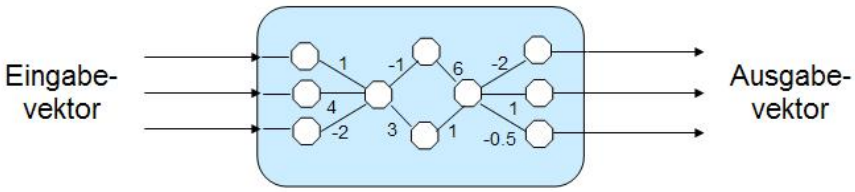
Neuronale Netze stellen vereinfachte Modelle des zentralen Nervensystems dar [Sche97]. Ein neuronales Netz kann als Abbildungsvorschrift verstanden werden, das eine Menge von Eingaben und Ausgaben beschreibt und das in nur zwei Zustände versetzt werden kann: entweder ist es aktiv (Zuordnung eines binären Wertes von 1) oder passiv (binär kodiert mit 0). Eingaben werden durch spezifische Eigenschaften, so genannte Eingabevektoren, kodiert und können die Eingabewerte 0 oder 1 haben. Die Ausgabe wird ebenfalls durch einen Vektor beschrieben (siehe Abbildung 54). Dabei können die Ein- und Ausgabevektoren aus unterschiedlichen Datenräumen entnommen werden.

Abbildung 54: Neuronales Netz als Abbildungsvorschrift



Neuronale Netze sind parallele Verarbeitungseinheiten, die aus vielen miteinander verbundenen, einfachen Prozessoren bestehen [Call03]. Diese Prozessoren sind stark vereinfacht und können nur einfache Rechnungen durchführen. Jeder Prozessor innerhalb eines Netzes kennt nur die folgenden Signale: jenes, das er in regelmäßigen Abständen empfängt, und jenes, das er regelmäßig an andere Prozessoren überträgt. Einfache lokale Prozessoren sind in der Lage, komplexe Aufgaben durchzuführen, wenn sie innerhalb eines großen Netzwerks koordiniert zusammenarbeiten. Die Verbindung zweier Prozessoren wird durch ein Gewicht bewertet. Durch die Modifikation dieser Gewichtung kann das Ein- und Ausgabeverhalten des Netzes in die gewünschte Form (Zielwert) verändert werden. Abbildung 55 veranschaulicht eine unterschiedliche Gewichtung von Verbindungen zwischen Eingabe- und Ausgabeströmen.

Abbildung 55: Neuronale Netze als Verbindungsstruktur einfacher Prozessoren

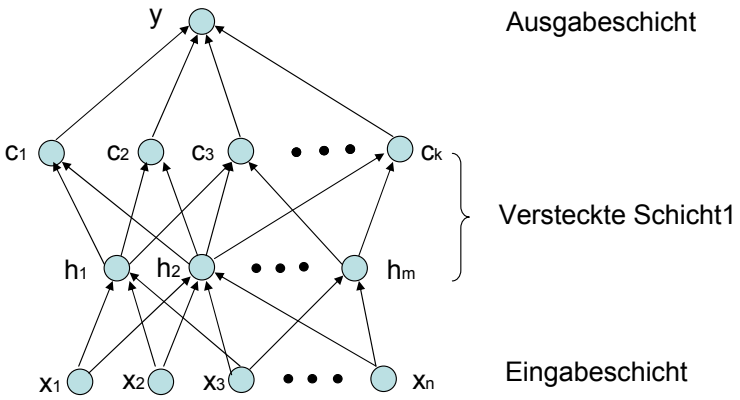


Ist die Summe über das Produkt aller Eingaben mit den dazugehörigen Gewichten größer als ein gewisser Schwellwert, so wird die Ausgabe auf 1 (aktiv), anderenfalls auf 0 (passiv) gesetzt.

Eine besondere Eigenschaft einiger Typen von neuronaler Netzen ist die Fähigkeit, beliebige Funktionen aus einer Menge von Übungsbeispielen zu erlernen (Lernstrategien). Mit dem mehrschichtigen, vorwärtsgerichteten, neuronalen Netz (Multilayer Perceptron) [Brau97] soll die initiale Gewichtung für Kontextelemente, die beim kombinierten Ähnlichkeitsmaß benötigt werden, gelernt werden.

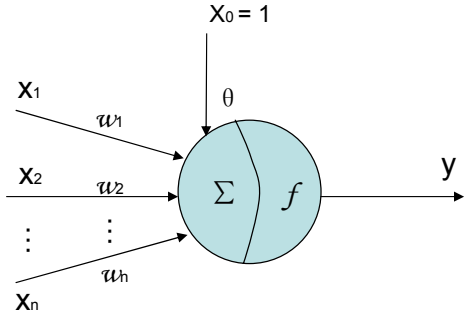
Ein vorwärtsgerichtetes neuronales Netz ist ein gerichtetes, azyklisches Netz und besteht aus einer Eingabeschicht, einer Ausgabeschicht und mindestens einer verborgenen Schicht, wie Abbildung 56 zeigt. In diesem Netz wandert die Aktivierung von der Eingabeschicht zur Ausgabeschicht, und die Einheiten (Perzeptronen) in einer Schicht sind mit jeder anderen Einheit in der nächst höheren Schicht verbunden. Die gesamte mittlere Schicht wird als versteckte Schicht bezeichnet, da sie nicht direkt Eingaben von der Umgebung bekommt und auch keine Ausgaben an die Umgebung abgibt.

Abbildung 56: Mehrschichtiges vorwärtsgerichtetes neuronales Netz



Jede Einheit in einem mehrschichtigen, vorwärtsgerichteten, neuronalen Netz wird durch ein Perzeptron [Rose58] beschrieben. Die Ausgabe des Perzeptron wird durch die Eingaben ($x_1,x_2,\ldots,x_n$), Gewichte ($w_1,w_2,\ldots,w_n$) und den Schwellwert (θ) bestimmt. Die Ausgabe y wird durch eine (logistische) Aktivierungsfunktion *f* (Erklärung siehe weiter unten) abgeleitet.

Abbildung 57: Einzel-Perzeptron



Aufgabe des vorwärtsgerichteten neuronalen Netzes ist es eine gegebene mehrdimensionale Funktion möglichst gut zu approximieren. Die Parameter des Netzes (Gewichte und Schwellen) sind so zu wählen, dass zum einen die Stützstellen möglichst genau geschätzt werden und zum anderen diese Einstellung der Parameter auf den übrigen Eingabenraum generalisiert werden kann. Eine höhere Anzahl von Übungsbeispielen verbessert das Generalisierungsverhalten. Allerdings sind viele unpassende Beispiele nicht so lehrreich wie wenige gut gewählte.

Ein mehrstufiges, vorwärtsgerichtetes, neuronales Netz ist in Abbildung 56 dargestellt. $x_1,\ldots,x_n$ sind die Eingabewerte, die die Werte $h_1,h_2,\ldots,h_m$ berechnen. Diese Werte sind wiederum die Eingabewerte für $c_1,c_2,\ldots,c_k$. Ausgabe des Netzes bildet y.

Die Gewichte der versteckten Schicht werden mit Hilfe der Ableitung der Fehlerfunktion nach dem entsprechenden Gewicht verändert. Die Funktion lautet

$$E = \sum (y_i - z_i)^2$$

Die y-Werte bilden die Ausgabe des Netzes und die z-Werte sind vorgegebene Werte. Je kleiner E ist, desto besser ist das Netz. Bei einem E=0 ist das Netz exakt (der automatisch berechnete Wert entspricht dem Zielwert).

Bei der Berechnung der Ableitung nach einem Gewicht in einer versteckten Schicht werden lediglich Zwischenergebnisse der nachfolgenden Schicht benötigt. Ziel ist es, eine Menge von Netzgewichten für das zugrunde liegende Problem zu finden.

In den 70er Jahren entwickelte Paul Werbos ein Verfahren zum Anpassen von Gewichten, das auf ein mehrschichtiges vorwärtsgerichtetes neuronales Netz angewendet werden kann. Das Verfahren ist unter dem Namen Backpropagation bekannt. Hinton, Rumelhart und Williams [RuHW86] haben erstmals einen Algorithmus zum Trainieren für mehrstufige Netze vorgestellt. Ein mehrschichtiges vorwärtsgerichtetes neuronales Netz wird durch die folgenden Gleichungen beschreiben:

$$h_m = f\left(\sum_{i=1}^{n} w_{im} x_i - \theta_m\right) \quad m = 1, 2, \ldots, n_2$$

$$c_k = f\left(\sum_{i=1}^{n_2} v_{jk} h_j - \theta_k\right) \quad k = 1, 2, \ldots, n_1$$

$$y = f\left(\sum_{k=1}^{n_1} w_k c_k - \theta\right)$$

Dabei ist h_m die Ausgabe der ersten versteckten Schicht; w_{im} ist das Gewicht zwischen dem *i*-Neuron der Eingabeschicht und dem *m*-Neuron der ersten versteckten Schicht; θ_m ist der Schwellwert des *j*-Neurons in der ersten Schicht. c_k ist die Ausgabe des *k*-Neurons in der zweiten versteckten Schicht. Das Gewicht zwischen dem *j*-Neuron der ersten versteckten Schicht und dem *k*-Neuron der zweiten versteckten Schicht wird durch v_{jk} repräsentiert und θ_k ist der entsprechende Schwellwert. w_k ist das Gewicht zwischen der zweiten versteckten Schicht und der Ausgabeschicht und θ ist der Schwellwert. *f* ist die binäre Transferfunktion.

Auf die oberste Schicht kann wieder eine Schicht aufgesetzt werden, die als Eingabe die Ausgabe der darunter liegenden Schicht hat.
Angenommen ein Ausgabewert y sei zu berechnen.
Dann ist:
$h_1=f(2*x_1-2*x_2-1)$
$h_2=f(-2*x_1+2*x_2-1)$
und daraus ergibt sich eine Ausgabe
$y=f(-3*h_1+0*h_2+2)$

Durch Nachrechnen kommt man auf eine in der Tabelle gezeigte Funktion:

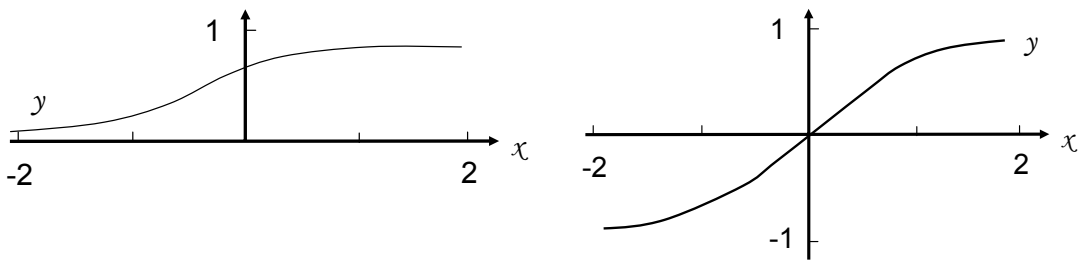
x1	**x2**	**h1**	**h2**	**y**
0	0	0	0	1
0	1	0	1	1
1	0	1	0	0
1	1	0	0	1

Im einen mehrschichtigen vorwärtsgerichteten neuronalen Netz werden die Ausgangssignale durch eine Aktivierungsfunktion berechnet. Für ein Backpropagation-Netz wären beispielsweise sigmoide Funktionen geeignet.

Sigmoide Funktionen

Das Ergebnis einer sigmoiden Funktion liegt zwischen 0 und 1. Dabei können die Steigung und der Wertebereich der sigmoiden Funktion variieren. Zwei Beispielfunktionen für sigmoide Funktionen sind die logistische Aktivierungsfunktion, die mit $f_{\log}(x)=\frac{1}{1+e^{-x/g}}$ und der Tangens hyperbolicus, der mit $f_{\tanh}(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}}$ berechnet werden.

Abbildung 58: Tangens hyperbolicus (links) und logistische Funktion mit g=0.5(rechts)



Der Backpropagation-Algorithmus besteht aus fünf Schritten:

(1) Initialisierung der Gewichte, in der ein stochastischer Wert im Intervall [-1,1] jeweils an w_{im}, w_{mt}, θ_m, γ_m vergeben wird,

(2) Auswahl von stochastischen Eingabe- und Zielexemplaren $\mathrm{P}_k = (a_1^k, a_2^k, ..., a_n^k)$ und $\mathrm{T}_k = (y_1^k, y_2^k, ..., y_n^k)$ und berechne die Belegung von h-Werten in der versteckten Schicht für das Netz,

(3) Korrektur von Gewichten für Eingabe- und Zielwerte,

(4) Wähle ein paar weitere Lernexemplare[69] für dieses Netz und gehe zu Schritt 2 bis alle Lernexemplare durchtrainiert wurden,

(5) Testen des trainierten Netzes mit Testexemplaren. Wenn die Differenz kleiner als ein erwünschter Wert ist, dann ist das Netz konvergent. D. h., das Netz wird erfolgreich trainiert. Wenn das Netz keinen erwünschten Wert oder kleineren Wert als einen erwünschten Wert erreicht, dann ist das Netz nicht konvergierbar.

dabei seien

$\mathrm{P}_k = (a_1, a_2, ..., a_n)$ die Eingabevektoren des Netzes,

$\mathrm{T}_k = (y_1, y_2, ..., y_n)$ die Zielvektoren des Netzes,

w_{im}, $i = 1, 2, ..., n$, $m = 1, 2, ..., p$ die Gewichte zwischen Eingabe- und Mittelschicht,

w_{mt}, $m = 1, 2, ..., n$, $t = 1$ die Gewichte zwischen der Mittel- und der Ausgabeschicht,

θ_m, $\mathrm{m} = 1, 2, ..., p$ die Schwellwerte für die Mittelschicht,

γ_m, $\mathrm{m} = 1, 2, ..., p$ die Schwellwerte für die Ausgabeschicht,

$k = 1, 2, ..., m$ Parameter**.**

7.1.2 Clusteranalyse

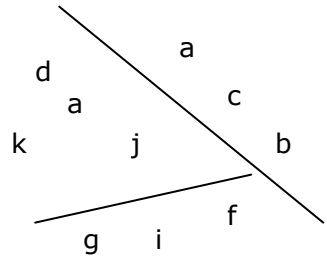
Ein Cluster ist eine Gruppe von Objekten mit ähnlichen Eigenschaften. In der Regel sind die Objekte eines Clusters verschieden von den Objekten eines anderen Clusters. Die Herausforderung ist dabei, Objekte nach geeigneten Eigenschaften aufzuteilen und passende Cluster zu bilden. Die Aufteilung von Personen in Cluster

[69] Beim Lernen von Gewichten werden Lern- und Testexemplare verwendet. Lernexemplare werden benutzt, um das Netz zu trainieren. Mit Testexemplaren wird das trainierte Netz getestet, ob die entsprechende Gewichte und Schwellwerte angemessen sind.

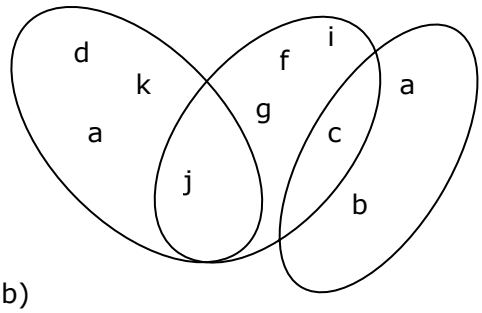
könnte z.B. nach der Eigenschaft weiblich/männlich erfolgen, so dass alle Männer einem Cluster und alle Frauen zu einem anderen Cluster subsumiert werden.
Mit Hilfe von Softwareprogrammen kann anhand festgelegter Benutzereigenschaften eine Clusteranalyse durchgeführt werden. Die Ähnlichkeit von betrachteten Objekten wird mittels mehrerer Eigenschaften (Variablen) gemessen. Die Statistiksoftware SPSS[70] unterstützen eine rechnergestützte Clusteranalyse ebenso wie die Software WinStat[71], ClusCorr98[72], CLUSTAN[73] oder SoftGuide[74].
Der Output einer Clusteranalyse kann in unterschiedlichen Diagrammformen dargestellt werden [WiFr05] (siehe Abbildung 59). Die Zugehörigkeit von Objekten zu einzelnen Clustern kann durch eine Aufteilung des Raums (in Abbildung 59a durch Linien) veranschaulicht werden. Alternativ können Objekte auch zu mehreren Clustern gehören (Abbildung 59b). Der Output wird dann als Menge dargestellt, wobei die überlappenden und nicht überlappenden Mengenelemente als Cluster verstanden werden. Die Zugehörigkeit zu einem Cluster kann auch über Cluster-Wahrscheinlichkeiten ausgedrückt werden (Abbildung 59c). Dabei werden Objekte mit gleicher Wahrscheinlichkeit demselben Cluster zugeordnet. Cluster können aber auch eine hierarchische Struktur besitzen (Abbildung 59d) und werden in einem Dendogramm dargestellt. Auf der obersten Ebene gibt es zunächst nur wenige Cluster (in dieser Abbildung sind es drei), die anschließend zu Unterclustern verfeinert werden. Die Elemente auf der untersten Ebene sind enger geclustert als Elemente auf der obersten Ebene.

[70] http://www.spss.com/de/
[71] http://www.winstat.de/
[72] http://www.wias-berlin.de/software/ClusCorr98/
[73] http://www.gesis.org/Software/CLUSTAN/index.htm
[74] http://www.softguide.de/prog_p/pp_0093.htm

Abbildung 59: unterschiedliche Darstellung von Clustern



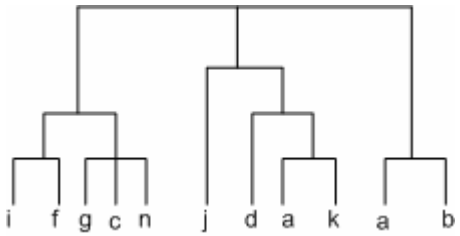
a)



b)

	1	2	3
a	0.4	0.1	0.5
b	0.1	0.8	0.1
c	0.3	0.34	0.4
d	0.1	0.1	0.8
e	0.4	0.2	0.4

c)



d)

Der Output von Clusteranalysen kann anhand von zwei Ansätzen evaluiert werden. Zum einen kann die betrachtete Gruppe von Objekten mit der Clusterlösung verglichen werden. Dabei wird angenommen, dass die vorgenommene Gruppierung eine „gute" Einteilung darstellt. Zum anderen können Methoden verwendet werden, die die Clustergüte mittels statischer Eigenschaften vergleichen; möglich wäre, einzelne Eigenschaften durch ein geeignetes Aggregationsverfahren zu einem Indexwert zu verschmelzen [WiFr05].

Die Berechnung der Ähnlichkeit von Objekten eines Clusters kann mit Ähnlichkeitsmetriken wie der quadrierten Euklidischen Distanz bestimmt werden:

$$d_{euklidisch}(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad ,$$

wobei d die Distanz und x und y zwei Koordinaten mit $x=(x_1,\ldots,x_n)$ und $y=(y_1,\ldots,y_n)$ sind.

Angenommen die beiden Fälle A und B haben die folgenden fünf Merkmale:

A: (6,8,2,1,3),

B: (7,5,3,6,2).

Dann errechnet sich eine Ähnlichkeit zwischen a und b nach der euklidischen Distanz:

$$d_{a,b} = \sqrt{(6-7)^2 + (8-5)^2 + ... + (3-2)^2} = 6,083 .$$

Zur Clusteranalyse wurden verschiedene Algorithmen vorgeschlagen. Ein entscheidendes Auswahlkriterium für einen Algorithmus ist das Skalenniveau der untersuchten Daten. Bacher [Bach94] unterscheidet nominale, ordinale, Intervall und metrisch skalierte Daten. In unserem Anwendungsszenario werden ordinalskalierte Variablen verwendet, da die Variablen (Ähnlichkeitswerte) Gleichheitsbeziehungen zulassen und der Größe nach sortierbar sind (0.4 < 0.5). Zwei Objekte sind einander bezüglich eines ordinalen Merkmals umso ähnlicher, je näher ihre Ausprägungen in der Rangordnung beieinander liegen.

Clusteranalysealgorithmen

Steinhausen und Langer [StLa77] teilen Clusteralgorithmen hinsichtlich ihres Gruppierungsresultats, des Gruppierungsprozesses und des Gruppierungskriteriums. Das Gruppierungsresultat kann entweder hierarchisch oder nicht-hierarchisch sein; beim Gruppierungsprozess wird zwischen iterativen und nicht-iterativen Verfahren unterschieden; beim Gruppierungskriterium lassen sich sequentielle und globale Verfahren differenzieren. Bei sequentiellen Verfahren wird durch die berechnete Distanz die gesuchte Gruppierung erreicht, während bei globalen Verfahren die Distanz aller Elemente benutzt wird.

Das Ergebnis der Clusterung kann entweder durch stufenweise Verfeierung (Division, top-down) oder Vergröberung (Agglomeration, bottom-up) erfolgen und kann mit Hilfe eines Dendogramms (siehe Abbildung 59d) dargestellt werden. Agglomerative und divisive Verfahren führen zu einer hierarchischen Darstellung der Datenstruktur. Nachfolgend werden die am häufigsten verwendeten agglomerativen Verfahren vorgestellt, mit denen ein Clustern von Benutzergruppen bestimmt werden soll [BEPW93]:

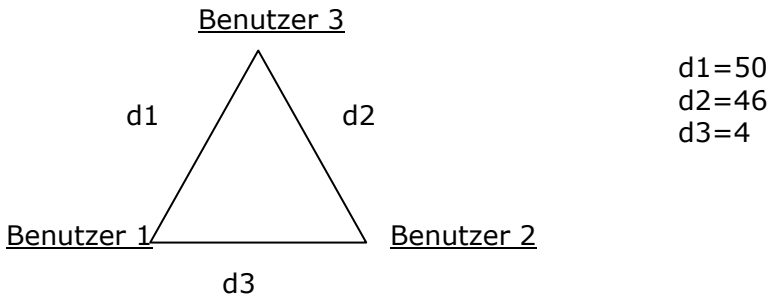
- Single Linkage (Nearest neighbor):

$$d_{single}(A,B) = \min_{a \in A, b \in B} \{d(a,b)\} .$$

  Dabei sei *d* die Distanz und a ein Element eines Clusters A und b ein Element eines Clusters B.

Vereint werden diejenigen Elemente zu einem Cluster, die die kleinste Distanz zueinander aufweisen. Abbildung 60 veranschaulicht die Berechnung der Distanz nach dem Single Linkage Verfahren. Die Distanz zwischen dem Cluster (Benutzer 1, Benutzer 2) und Benutzer 3 beträgt einmal 50 und einmal 46. Entsprechend dem Single Linkage Algorithmus wird für den zweiten Durchlauf der Clusterbildung eine Distanz von 46 herangezogen.

Abbildung 60: Berechnung der neuen Distanz beim Single Linkage Verfahren



- Complete Linkage (Furthest neighbor):

$$d_{complete}(A,B) = \max_{a \in A, b \in B} \{d(a,b)\} \ .$$

Herangezogen wird bei diesem Verfahren der größte Abstand zwischen den Elementen. In Abbildung 60 wäre beim zweiten Durchlauf die Distanz zwischen dem Cluster (Benutzer 1, Benutzer 2) und dem Element Benutzer 3 die Distanz 50 heranzuziehen.

- Average Linkage:

$$d_{average}(A,B) = \frac{1}{|A||B|} \sum_{a \in A, b \in B} d(a,b) \ .$$

Bei diesem Verfahren wird der Mittelwert aller Distanzen zwischen Elementen der beiden betrachteten Cluster herangezogen.

- Ward´s Methode:

$$d_{ward's}(A,B) = \frac{d(\bar{a},\bar{b})}{\frac{1}{|A|} + \frac{1}{|B|}} \ .$$

Bei diesem Verfahren wird nicht die Distanz zwischen Elementen bzw. Clustern in die Berechnung herangezogen, sondern es werden Cluster zusammengefasst, die ein vorgegebenes *Heterogenitätsmaß* am wenigsten erhöhen. Ziel dieses Verfahrens ist es, möglichst homogene Cluster dadurch zu bilden, dass durch die Vereinigung zu Gruppen die Varianz einer Gruppe möglichst nicht erhöht wird.

7.2 Vorgehensmodell für die Evaluation

Die Evaluation von Ähnlichkeitsmaßen für Geschäftsprozesse und die Einteilung von Modellierern zu Gruppen wird mit Hilfe einer Benutzerbefragung wie folgt durchgeführt (in Anlehnung an [StLa77]):

1. Präzisierung der Untersuchungsfragestellung
2. Auswahl der Variablen zur Erstellung eines Fragebogens
3. Durchführung der Befragung
4. Aufbereitung der Daten
5. Bestimmung von geeigneten Verfahren des maschinellen Lernens
6. Technische Durchführung
7. Analyse und Interpretation der Ergebnisse (Postanalyse)

Zu 1.
Bevor ein Fragebogen erstellt wird, muss die Untersuchung präzisiert werden. Oft werden zur Auswertung des Fragebogens Verfahren des maschinellen Lernens in Kombination mit anderen Verfahren eingesetzt. Eine eindeutige Problemdefinition soll den Stellenwert von Verfahren des maschinellen Lernens unterstreichen.

Zu 2.
Im Hinblick auf eine sinnvolle Auswertung des Fragebogens und einen effektiven Einsatz von Verfahren des maschinellen Lernens sollten die ausgewählten Variablen sich auf die Fragestellung der Untersuchung beziehen.

Zu 3.
Abhängig von der Untersuchungsfragestellung wird eine „kritische Masse" an Befragten bestimmt, die entweder willkürlich oder bewusst ausgewählt werden. Anschließend wird die Verteilung des Fragebogens festgelegt: entweder könnte der Fragebogen per E-Mail oder persönlich verteilt werden. Bei einer persönlichen Ver-

teilung kann die Beantwortung des Fragebogens im Beisein des Fragebogenerstellers erfolgen. Dies hat den Vorteil, dass die Beantwortung des Fragebogens weniger verfälscht wird als bei einer Befragung ohne den Fragebogenersteller (der Fragebogenersteller kann kontrollieren, dass der Befragte sich an die Rahmenbedingungen des Fragebogens hält). Zum anderen erhöht eine persönliche Befragung den Rücklauf, da der Befragte „gezwungen" ist, den Fragebogen an Ort zu Stelle auszufüllen.

Zu 4.
Nach Durchführung der Befragungen müssen die Daten aufbereitet werden. Die Auswertung der Fragebögen hängt wiederum vom Gegenstand der Untersuchung ab. Einfache statistische Diagramme können helfen, sich einen Überblick über die erhobenen Daten zu verschaffen. Eventuell müssen die erhobenen Daten um fehlende Daten korrigiert werden (je nachdem welche Kenngrößen berechnet werden sollen).

Zu 5.
Im Hinblick auf die Fragestellung der Untersuchung werden geeignete Verfahren des maschinellen Lernens ausgewählt.

zu 6.
Die Auswertung der Fragebögen mit Verfahren des maschinellen Lernens empfiehlt sich rechnergestützt durchzuführen (z.B. aus Gründen der Performance). Hierzu muss eine passende Software ausgewählt werden.

Zu 7.
Mit der Analyse der Ergebnisse soll die Angemessenheit der gefundenen Lösung von Seiten der Verfahren des maschinellen Lernens bewertet werden. Beurteilt wird beispielsweise der Einfluss bestimmter Variablen im Hinblick auf die Fragestellung der Untersuchung.

7.3 Ergebnisse der Evaluation

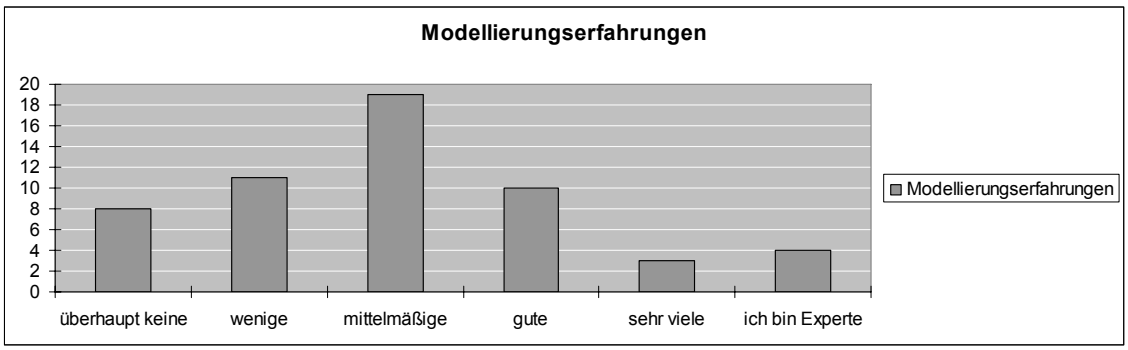
Als Fragestellung der Untersuchung wurde die Möglichkeit einer gruppenspezifischen Einteilung der Befragten definiert, um eine automatische Unterstützung bei der Geschäftsprozessmodellierung anbieten zu können. Weiterhin sollte die Befragung helfen, eine initiale Angabe von Gewichten (wie sie in Kapitel 4.6.5. vorgestellt wurden) zur Berechnung der kombinierten Ähnlichkeit festsetzen zu können.

Die Variablen der Fragestellung der Untersuchung sind Ähnlichkeitswerte und die von Benutzern als passend ausgewählten Folgeprozesse zur automatischen Prozessergänzung.
Zu Beginn der Befragung wurden bewusst 50 Personen entsprechend ihrer Modellierungserfahrungen ausgewählt. Um Benutzer in Gruppen einteilen zu können, wurden die Befragten entsprechend zweier Merkmale charakterisiert: nach ihren Kenntnissen in der Geschäftsprozessmodellierung und ihrer Vertrautheit mit der Modellierungsdomäne bei der Geschäftsprozessmodellierung.

7.3.1 Generelles

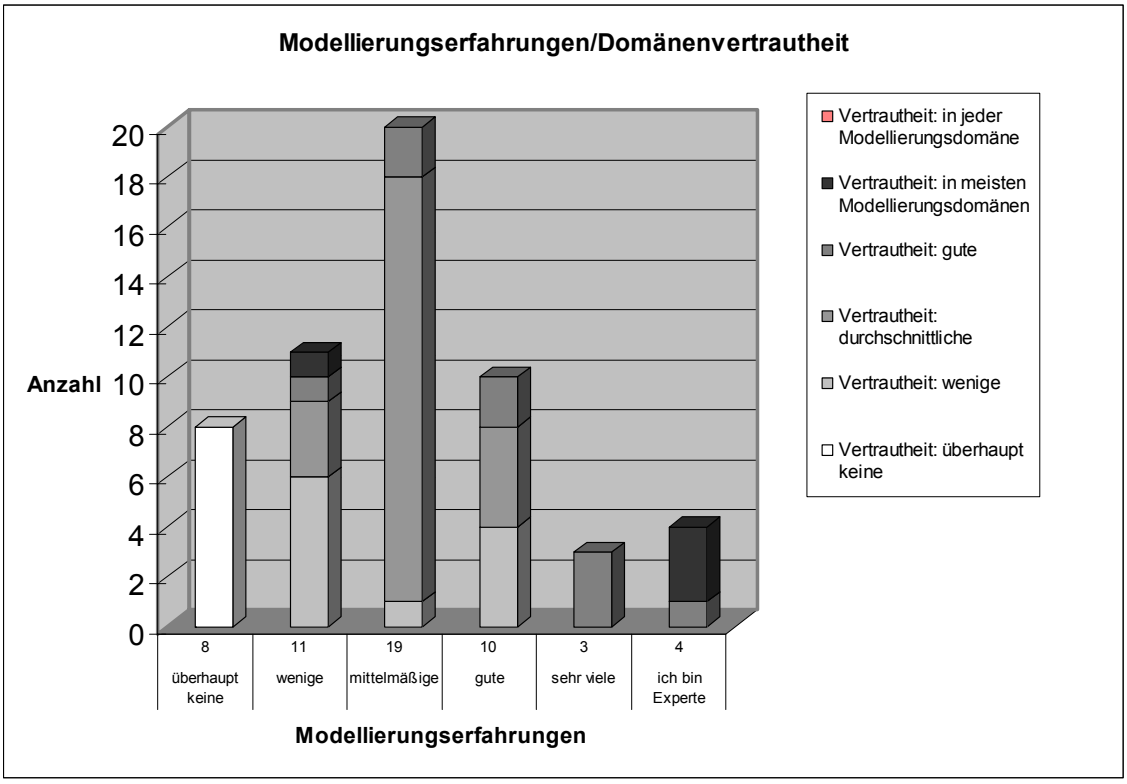
Unter Berücksichtigung der Zielsetzung der Befragung wurde ein standardisierter und strukturierter Fragebogen (siehe Anhang) erstellt. Der Fragebogen umfasste 11 Fragen. Bei den ersten beiden Fragen sollten die Befragten Aussagen über ihre Modellierungserfahrungen machen. Fragen 3 bis 9 beschäftigten sich mit der Beurteilung und Einschätzung von Ähnlichkeitswerten und die letzten beiden Fragen haben sich auf die Modellierungsunterstützung während der Geschäftsprozessmodellierung (siehe Kapitel 6) bezogen.
Alle Befragten bekamen dieselben elf Fragen vorgelegt, die in einer zielorientierten inhaltlichen Abfolge gestellt wurden. Insgesamt wurden 55 Personen befragt, da im Laufe der Befragung fünf weitere Personen als relevante Zielpersonen eingestuft wurden. Abbildung 61 zeigt die Aufteilung aller befragten Personen entsprechend ihrer Erfahrungen in der Geschäftsprozessmodellierung. Die Befragten konnten aus sechs Kategorien die auf sie passende Kategorie auswählen. Die meisten Befragten (19 Personen) gaben mittelmäßige Erfahrungen in der Geschäftsprozessmodellierung an.

Abbildung 61: Modellierungserfahrungen der Befragten



In der zweiten Frage mussten die Befragten ihre Vertrautheit mit der Modellierungsdomäne (z.B. Versicherungswesen, Medizin, Fertigung, Supply Chain Management, etc.) einschätzen, d.h. in welchem Umfang Informationen über die zu modellierende Domäne bestehen. Abbildung 62 zeigt, dass von den Personen mit mittelmäßigen Erfahrungen die meisten auch über eine durchschnittliche Vertrautheit mit der Modellierungsdomäne verfügen. Eindeutig ist die Vertrautheit mit der Modellierungsdomäne bei Personen, die sehr viele Erfahrungen in der Geschäftsprozessmodellierung haben: alle Befragten dieser Kategorie geben eine gute Vertrautheit mit der Modellierungsdomäne an. Bei den Personen, die sich als Experten eingestuft haben (4 Personen), haben 25% (1 Person) gute Vertrautheit mit der Modellierungsdomäne und 75% (3 Personen) kennen sich in den meisten Modellierungsdomänen aus. Diese Einschätzung der Experten könnte auf ihre routinierte Prozessmodellierung zurückgehen.

Abbildung 62: Vertrautheit mit der Modellierungsdomäne im Verhältnis zu Modellierungserfahrungen



7.3.2 Aufbereitung der erhobenen Daten

Bevor eine Datenreihe (Stichprobe) festgelegt wird, müssen die erhobenen Daten aufbereitet werden, in dem die Eigenschaften *Repräsentativität* und *Konsistenz* geprüft werden [BEPW93].

Bei der Überprüfung von Repräsentativität wird versucht, Aussagen über die Grundgesamtheit treffen zu können. Die Ergebnisse aus den erhobenen Daten sollen sich auf jede beliebige Stichprobe und damit auf die Gesamtmasse (Bevölkerung) übertragen lassen.

Zur Beantwortung des Fragebogens wurde eine bewusste Auswahl der Befragten vorgenommen. Von den acht Befragten, die angaben überhaupt keine Erfahrungen in der Geschäftsprozessmodellierung zu haben, verfügen alle über Kenntnisse in der Graphentheorie. Da ein Petri-Netz ein gerichteter, bipartiter Graph ist, wurde angenommen, dass diese Kategorie der Befragten die Geschäftsprozessmodelle (die mit Petri-Netzen modelliert wurden) lesen und verstehen können. Die übrigen Befragten wurden unter Berücksichtigung ihrer Erfahrungen in der Geschäftsprozessmodellierung ausgewählt.

Die Repräsentativität der Stichprobe könnte beispielsweise mit dem Chi-Quadrat-Test [FrPP98] überprüft werden. Allerdings lag der Schwerpunkt der Befragung auf einer initialen Angabe der Gewichtung *w* zur passenden Berechnung der kombinierten Ähnlichkeit und in der Einteilung von Modellierern in sehr ähnliche Gruppen; d.h. die Erfüllung der Repräsentativität und damit die Übertragung der Ergebnisse auf beliebige Stichproben war nicht im Fokus der Befragung. Wobei [LiKl02] argumentieren, dass Repräsentativität kein Qualitätsmerkmal für eine Untersuchung ist, da Repräsentativität zum einen keine mathematisch fundierten Forderungen ermöglicht und zum anderen *in einem ungeklärten, wenn nicht widersprüchlichen Zusammenhang zu anderen Variablen steht, die ohne Zweifel entscheidend sind für die Güte einer Auswahl* (z.B. der Umfang *n* der Teilgesamtheit oder Stichprobe und die Varianz des Untersuchungsmerkmals in der Grundgesamtheit als Ausdruck der Homogenität der Grundgesamtheit).

Bei der Überprüfung der Konsistenz der erhobenen Daten wurde untersucht, ob die Befragten konsistent bei der Schätzung von Ähnlichkeitswerten waren. Die Konsistenz wurde anhand zweier Prozesselementpaare untersucht. In Frage 6 sollten die Befragten die Ähnlichkeit des Paares *inspect request vs. check request* und *request checked vs. request inspected* einschätzen (Elemente, die aus dem Prozessmodell aus Frage 3 entnommen wurden; die Verben der Paare haben zwar eine unterschiedliche grammatikalische Form, aber eine gleiche Stammform). Von den 55 Befragten haben 42 Personen für die beiden Paare einen identischen Ähnlichkeits-

wert geschätzt. 13 Personen haben die Ähnlichkeit dieser Paare unterschiedlich bewertet, wobei 10 der „Abweichler" einen Unterschied von 0.1 eingeschätzt haben.
Insgesamt kann festgestellt werden, dass die erhobenen Daten in der Tendenz konsistent sind.
Dieser Konsistenztest wurde nur für Ähnlichkeitswerte, nicht aber für die Auswahl von passenden Folgeprozessen vorgenommen. D.h. für die Auswertung der Daten im Hinblick auf eine initiale Gewichtung von *w* werden die Antworten von 41 Befragten berücksichtigt. Bei der Auswertung der Fragen 10 und 11 (Auswahl von passenden Folgeprozessen) besteht die Datenbasis aus den Antworten der 55 Befragten.
Als eine weitere relevante Eigenschaft der Stichprobe wäre zu untersuchen, ob die Befragten eine gleiche Vorstellung von Ähnlichkeitswerten haben. Z.B. gaben Befragte mit hohem technischem Hintergrund an, dass ein Ähnlichkeitswert von 1.0 aufgrund vieler Einflüsse schwierig automatisch zu bestimmen wäre. Personen, die hingegen diese Kenntnisse nicht hatten, hatten höhere Erwartungen an automatisch berechnete Ähnlichkeitswerte.
Signifikante Unterschiede in der Bewertung der Ähnlichkeitswerte können mit dem Friedman-Test untersucht werden.

Friedman-Test

Der Friedman-Test (Rangvarianzanalyse) ermöglicht es, mehrere voneinander abhängige Stichproben zu untersuchen. Die Nullhypothese (H0) des Friedman-Tests behauptet, dass sich die unterschiedlichen Messungen der Individuen unter *k* Bedingungen (Messwiederholungen) unterscheiden [Posp06]. Wenn H0 zutrifft, dann verteilen sich die Rangplätze für die Messwerte unter den Bedingungen pro Untersuchungsobjekt nach Zufall. Triff H0 nicht zu, so werden sich unter verschiedenen Bedingungen unterschiedliche Rangsummen ergeben.
Die Rangvarianz zwischen den *k* Gruppen wird mit

$$\mathrm{RV} = \sum_j (\bar{r}_{.j} - \bar{r}_{..})^2$$

$\bar{r}_{.j}$ der mittlere Rangsumme der Gruppe j

$\bar{r}_{..}$ der mittlere Rangsumme gesamt

getestet.
Die Prüfgröße v (ob Rangsummenunterschiede zufällig oder nicht zufällig sind) ergibt sich aus

$$v = \frac{12}{nk(k+1)} \sum_j r_{.j}^2 - 3n(k+1)$$

dabei sei *k* die Anzahl der Gruppen.

Mit dem Friedman-Test soll überprüft werden, ob die Benutzer gleiche Bewertungen von automatisch berechneten Ähnlichkeitswerten hatten. Hierfür wurden die in Frage 7 vergebenen Noten (auf einer Skala von 1 bis 6) für die automatisch berechneten Ähnlichkeitswerte in einer Tabelle aufgelistet und anschließend wurde die mittlere Rangvarianz mit Hilfe des Friedman-Tests berechnet:

Tabelle 13: Mittlere Ränge ermittelt mit Hilfe des Friedman-Tests

	mittlerer Rang
Wert 1 (book travel vs. book travel)	3,44340
Wert 2 (inspect request vs. check request)	4,62264
Wert 3 (send confirmation vs. send information)	7,53774
Wert 4 (rejected vs. rejection sent)	8,25472
Wert 5 (check availability vs. check capacity)	7,02830
Wert 6 (request checked vs. request inspected)	8,36792
Wert 7 (confirmation agency vs. confirmation)	7,72642
Wert 8 (Availability vs. travel plan)	5,60377
Wert 9 (Quantity vs. Seat)	6,79245
Wert 10 (Paris vs. FRA)	5,40566
Wert 11 (client data vs. confirmation customer)	6,50000
Wert 12 (status data vs. capacity checked)	6,71698

Mit der Software WinStat wurden für den Friedman-Test die folgenden statistischen Daten ermittelt:

N	53
Chi-Quadrat	99,5951
df	11
p	$2{,}14789^{-16}$

Ergebnisse Friedman-Test

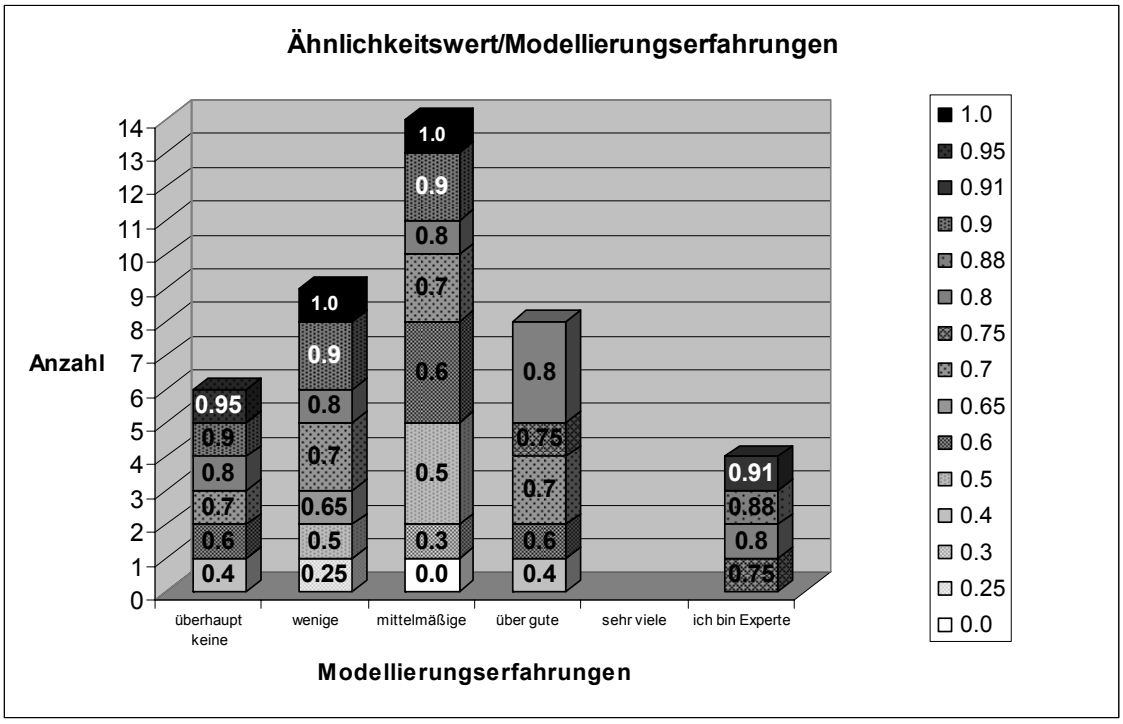
- Die mittleren Rangsummen der Werte 4 und 6 weisen ähnliche Werte auf. Das gleiche gilt für die Gruppen (Wert 8, Wert 10), (Wert 3, Wert 5, Wert 7), (Wert 9, Wert 11, Wert 12). Damit wurde der Wert 1 am besten bewertet (der auto-

matisch berechnete Ähnlichkeitswert stimmte am besten mit den Erwartungen der Benutzer überein). Wert 4 und 6 wurden am schlechtesten bewertet.

- Es bestehen signifikante Unterschiede zwischen den Bewertungen der automatisch berechneten Ähnlichkeitswerte. Der p-Wert von $2{,}14789^{-16}$ ist kleiner als der kritische Wert von 0,05 (beim zweiseitigen Test liegt der kritische Wert bei 0,05). Die Nullhypothese (H0) wird verworfen.

Zuletzt wird die Streuung der Datenreihe untersucht, um mögliche Ausreißer zu identifizieren.
Bei Frage 3 und 4 sollten die Benutzer die Ähnlichkeit zwischen zwei Prozessmodellen einschätzen. Die Abbildungen 63 und 64 zeigen die von den Befragten eingeschätzten Ähnlichkeitswerte (zunächst ohne Berücksichtigung der Vertrautheit mit der Modellierungsdomäne). Die Experten gaben für das erste Prozessmodell Ähnlichkeitswerte in einem Intervall von [0.75, 0.91] an. Bei einem Mittelwert von 0.835 stuften die Experten die Prozessmodelle als sehr ähnlich ein. Bei den übrigen Benutzergruppen ist der eingeschätzte Ähnlichkeitswert breit gestreut.

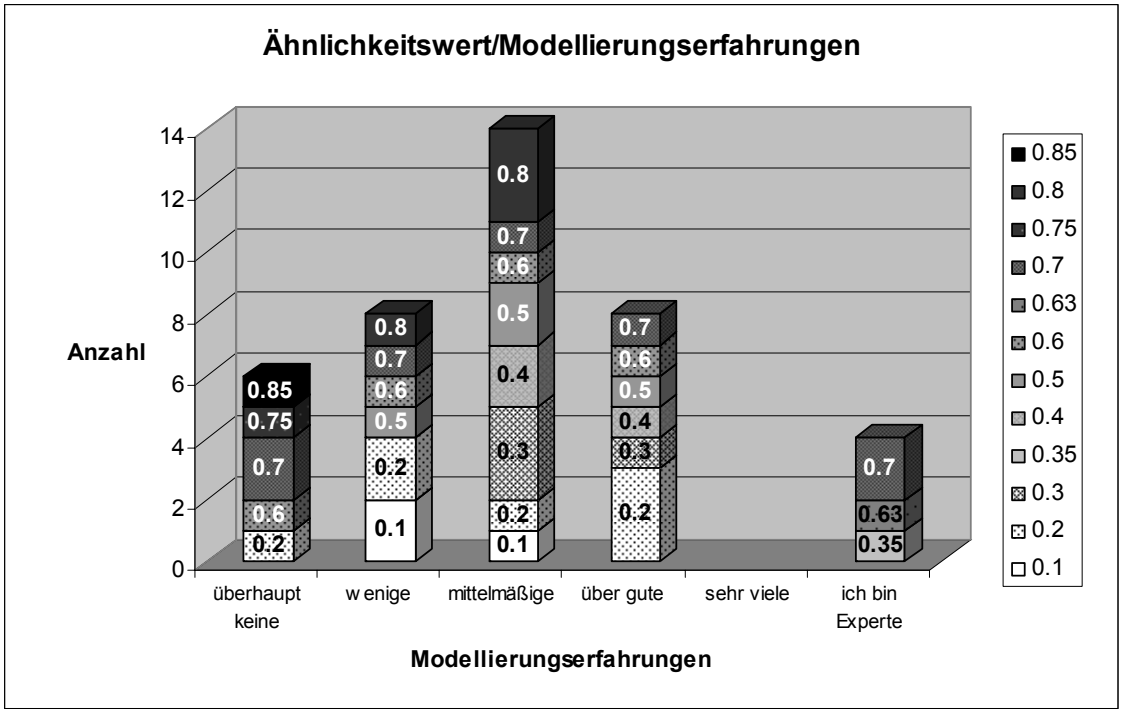
Abbildung 63: Eingeschätzte Ähnlichkeitswerte beurteilt nach Erfahrungen in der Geschäftsprozessmodellierung (Frage 3)



Beim zweiten Prozessmodell (Frage 4) gab die Mehrzahl der Experten ebenfalls einen Wert an, der nah an den durchschnittlichen Einschätzungen aller Experten lag.

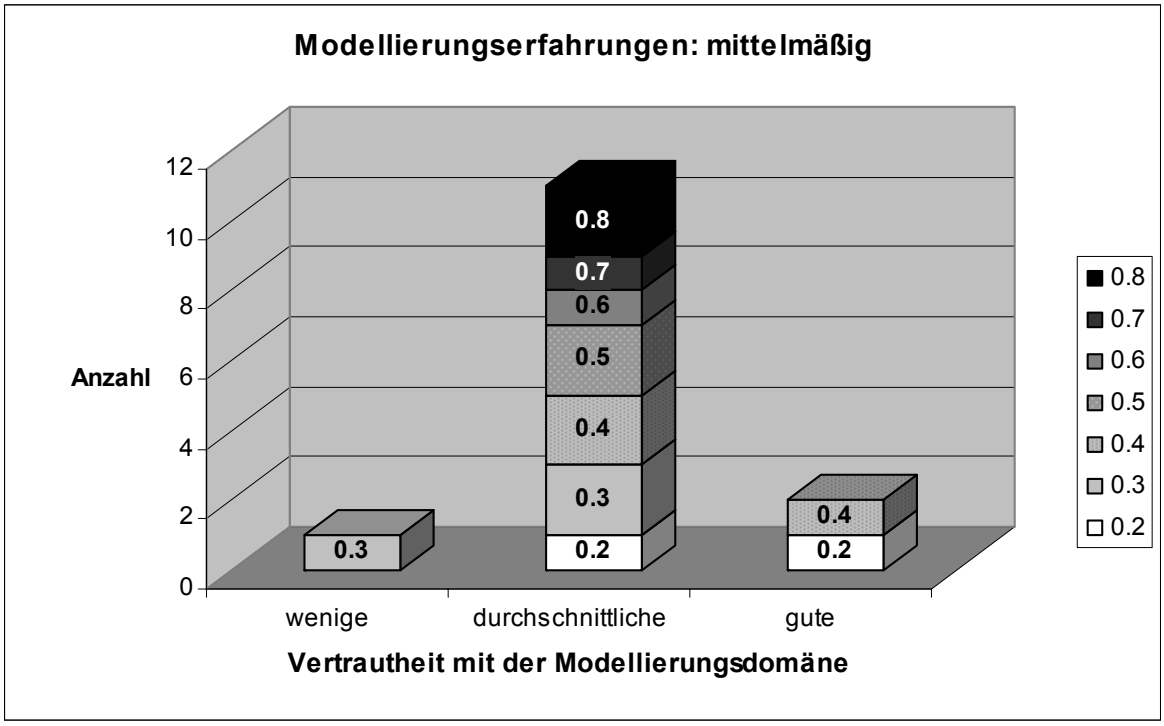
Sowohl beim ersten Prozessmodell (Abb. 63) als auch beim zweiten (Abb. 64) lässt sich bei den Befragten mit guten Erfahrungen in der Geschäftsprozessmodellierung eine geringere Abweichung zwischen den durchschnittlich eingeschätzten Ähnlichkeitswerten feststellen. Im Gegensatz zu den übrigen drei Gruppen, wo die maximale Abweichung bei 1.0 (Abb. 63) bzw. 0.7 (Abb. 64) (mittelmäßige Erfahrungen) gefolgt von 0.75 bzw. 0.7 (wenige Erfahrungen) und 0.55 bzw. 0.65 (überhaupt keine Erfahrungen) liegt. Die unterschiedliche Streuung der eingeschätzten Ähnlichkeitswerte könnte darauf hinweisen, dass Befragte ihre Erfahrungen überschätzt haben. Die Überschätzung ist daran erkennbar, dass die geschätzten Ähnlichkeitswerte von Befragten mit überhaupt keinen Modellierungserfahrungen weitaus weniger streuen; hier hatten die Befragten auch keinen Anreiz, ihre Erfahrungen zu überschätzen und in eine unzutreffende Gruppe eingeteilt zu werden.

Abbildung 64: Eingeschätzte Ähnlichkeitswerte beurteilt nach Erfahrungen in der Geschäftsprozessmodellierung (Frage 4)



Diese hohe Abweichung bleibt auch bei Befragten mit durchschnittlichen Modellierungserfahrungen unter Berücksichtigung ihrer Vertrautheit mit der Modellierungsdomäne bestehen, wie Abbildung 65 zeigt.

Abbildung 65: durchschnittlichen Modellierungserfahrungen nach Vertrautheit mit der Modellierungsdomäne (Frage 3)



Im nächsten Schritt werden die Spannweite, die Varianz und die Standardabweichung für Benutzer mit wenigen und durchschnittlichen Modellierungserfahrungen berechnet, um die Breite des Streubereichs zu untersuchen [FaKP04].
Die Spannweite (range) ergibt sich aus:

$$R = x_{max} - x_{min}$$

x_{max} = größter Beobachtungswert

x_{min} = kleinster Beobachtungswert

Für Frage 3 (abgekürzt mit F3) ergibt sich für Personen mit wenigen (w) und durchschnittlichen (d) Modellierungserfahrungen ein R_{F3_w} = 1.0 – 0.25 = **0.75** und R_{F3_d} = 1.0 - 0.0 = **1.0**

Für Frage 4 (F4) resultiert ein R_{F4_w} = 0.8 – 0.1 = **0.7** und R_{F4_d} = 0.8 – 0.1 = **0.7**

Die größte Spannweite besteht bei Frage 3 bei Modellierern mit durchschnittlichen Erfahrungen.

Der Mittelwert für beide Kategorien und Fragen lautet:

$\bar{x}_{F3_w}$ = **0.71111** bzw. $\bar{x}_{F3_d}$ = **0.61429** und

$\bar{x}_{F4_w}$ = **0.4** bzw. $\bar{x}_{F4_d}$ = **0.47857**

Die Varianz einer Datenreihe ergibt sich aus:

$$V^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + ... + (x_n - \bar{x})^2}{n}$$

n = Anzahl der Beobachtungswerte

x_i = i-ter Beobachtungswert

$\bar{x}$ = Mittelwert

Die Standardabweichung ergibt sich durch Ziehen der Wurzel aus der Varianz:

$$SD = \sqrt{V^2} = \sqrt{\text{Varianz}}$$

Eine geringe Standardabweichung bedeutet, dass alle Beobachtungswerte nahe am Mittelwert liegen. Eine hohe Standardabweichung deutet auf eine weite Streuung der Beobachtungswerte um den Mittelpunkt hin.

Für die Frage 3 ergeben sich die folgenden Varianzen und Standardabweichungen:

$V_{F3_ü}$ = **0,0417500** SD = **0,20432**

V_{F3_w} = **0,0529862** SD = **0,23018**

V_{F3_d} = **0,0674725** SD = **0,25975**

V_{F3_g} = **0,0188839** SD = **0,13741**

V_{F3_e} = **0,0053667** SD = **0,07325**

Die Varianz der geschätzten Ähnlichkeitswerte (und auch die Standardabweichung) von Modellierern mit durchschnittlichen Modellierungserfahrungen ist am größten;

d.h. die Werte liegen weiter auseinander als die geschätzten Ähnlichkeiten von den übrigen Benutzergruppen.
Die Statistiksoftware WinStat hat als Ausreißer die beiden Befragten mit der Nr. 30 (durchschnittliche Modellierungskenntnisse) und 43 (wenige Modellierungskenntnisse) durch statistische Verfahren bestimmt. Durch Eliminierung der Daten dieser beiden Befragten ergeben sich die folgenden Varianzen und Standardabweichungen:

V_{F3_w} = **0,0471429** SD = **0,21712**

V_{F3_d} = **0,0392308** SD = **0,19806**

Die Daten dieser beiden Befragten werden im weiteren Verlauf der Auswertungen (speziell bei neuronalen Netzen) nicht mehr betrachtet.

7.3.3 Anwendung von Verfahren des maschinellen Lernens

Zur Anwendung von neuronalen Netzen werden die von den Befragten vergebenen Noten ausgewertet. Eine durchschnittlich schlecht vergebene Note deutet darauf hin, dass der Benutzer die Ähnlichkeit eines Wortpaares höher bzw. niedriger als die automatisch berechnete geschätzt hätte. Durch neuronale Netze soll die initiale Gewichtung beim kombinierten Ähnlichkeitsmaß festgelegt werden.

Bestimmung einer initialen Gewichtung *w*

Die Ergebnisse der Befragung werden als Ausgabevektor von neuronalen Netzen verwendet. Tabelle 14 zeigt einen Beispieldatensatz aus den Fragen 6 und 8 der Umfrage (geordnet nach Elementtyp), auf den im weiteren Verlauf zur Bestimmung einer initialen Gewichtung zurückgegriffen wird. Die erste Spalte nennt den Namen des Elementpaars; die zweite Spalte veranschaulicht den automatisch berechneten Ähnlichkeitswert. Der dritte Wert gibt den vom Benutzer eingeschätzten Ähnlichkeitswert für dieses Paar wieder. In der vierten Spalte ist die Differenz zwischen dem eingeschätzten und dem automatisch berechneten Ähnlichkeitswert angegeben. Eine Differenz aus Ähnlichkeitswert-Software und Ähnlichkeitswert-Mensch nahe bei Null bedeutet, dass die Software den vom Menschen eingeschätzten Ähnlichkeitswerte gut „approximiert". Je größer die Differenz desto größer sind die unterschiedlichen Einschätzungen/Berechnungen; z.B. hätten die Befragten die Ähnlichkeit des Transitionspaares (*send confirmation* vs. *send information*) mit 0.5 geschätzt. Vom System wurde aber ein Ähnlichkeitswert von 0.8 berechnet.

Tabelle 14: Ähnlichkeitswerte für Transitionen aus Frage 6 bis 9

Name	**Ähnlichkeitswert - Software** (sim_{com})	**Ähnlichkeitswert – Mensch** (sim_{men})	**Differenz** = $(sim_{com}-sim_{men})^2$
book travel *vs.* book travel	0.9655	0.9904	0.00062001
inspect request *vs.* check request	0.8451	0.8265	0.00034596
send confirmation vs. send information	0.8429	0.5214	0.10336225
check availability vs. check capacity	0.5621	0.6349	0.00529984
categorize damage *vs.* categorize casualty	0.7599	0.7439	0.000256
assign evaluator *vs.* estimate casualty	0.4392	0.2822	0.024649
forward document *vs.* inform customer	0.2548	0.3282	0.00538756
finish casualty process *vs.* furnish option	0.2174	0.2151	0.00000529
make notice file *vs.* appraise damage	0.2101	0.2439	0.00114244
consult supervisor *vs.* appoint handling	0.1755	0.2420	0.00442225

Die größte Differenz zwischen Ähnlichkeitswert-Software und Ähnlichkeitswert-Mensch ergibt sich beim Stellenpaar *status data* vs. *capacity checked* (Tab. 15). Durch die Software wurde ein Ähnlichkeitswert von 0.0335 berechnet und die Befragten hatten durchschnittlich einen Ähnlichkeitswert von 0.3 angegeben.

Tabelle 15: Ähnlichkeitswerte für Stellen aus Frage 6 bis 9

Name	**Ähnlichkeitswert - Software** (sim_{com})	**Ähnlichkeitswert – Mensch** (sim_{men})	**Differenz** = $(sim_{com}-sim_{men})^2$
rejected vs. rejection	0.6409	0.7847	0.02067844
request checked vs. request inspected	0.5384	0.7571	0.04782969
confirmation agency vs. confirmation	0.2955	0.4704	0.03059001
availability vs. travel plan	0.1641	0.1747	0.00011236
client data vs. confirmation customer	0.0512	0.3245	0.07469289
status data vs. capacity checked	0.0335	0.3194	0.08173881
notification received *vs.* notification	0.6837	0.6873	0.00001296
low damage *vs.* low casualty	0.6598	0.7331	0.00537289
high casualty *vs.* high damage	0.6598	0.7332	0.00538756
collected notification loss *vs.* notification received	0.4705	0.4118	0.00344569
estimated *vs.* assigned	0.3935	0.2582	0.01830609
damage appraised *vs.* notice inserted	0.2133	0.2369	0.00055696
supervisor informed *vs.* documents evaluator	0.1974	0.2204	0.000529
customer informed *vs.* document forwarded	0.1297	0.3265	0.03873024

Die größte Differenz beim Attributenpaar besteht bei *Loss Name vs. Loss Name* (Tabe. 16) Der von der Software berechnetet Ähnlichkeitswert ist viel geringer als der von den Befragten geschätzte.

Tabelle 16: Ähnlichkeitswerte für Attribute aus Frage 6 bis 9

Name	Ähnlichkeitswert - Software	Ähnlichkeitswert – Mensch	Differenz $= (sim_{com}\text{-}sim_{men})^2$
Quantity 1 vs. Seat	0.1252	0.2776	0.02322576
Paris vs. FRA	0.0964	0.2663	0.02886601
Loss Name *vs.* Loss Name	0.75	0.9827	0.05414929

Als Eingabewerte für das neuronale Netz werden Ähnlichkeitswerte, berechnet mit den Ähnlichkeitsmaßen sim_{syn}, sim_{ling} und sim_{str}, verwendet. Tabellen 17 und 18 zeigen nur einen Ausschnitt der Eingabewerte für Transitionen und Stellen. Die vollständigen Eingabewerte sind im Anhang C Evaluation aufgelistet.

Tabelle 17: Eingabewerte für Transitionen (Ausschnitt)

Name	sim_{syn}	sim_{ling}	sim_{str}
book travel *vs.* book travel	1.0	1.0	1.0
inspect request *vs.* check request	0.6154	0.7436	1.0

Tabelle 18: Eingabewerte für Stellen (Ausschnitt)

Name	sim_{syn}	sim_{ling}	sim_{str}
rejected *vs.* rejection	0.625	0.625	1.0
request checked *vs.* request inspected	0.6667	0.7778	1.0

Tabelle 19: Eingabewerte für Attribute

Name	sim_{syn}	sim_{ling}	sim_{str}
Quantity 1 *vs.* Seat	0	0	0
Paris *vs.* FRAU	0	0	0
Loss Name *vs.* Loss Name	1	1	0.0907

Die Eingabewerte dienen als Eingabe für ein mehrschichtiges, vorwärtsgerichtetes, neuronales Netz. Ein initiale Gewichtung für die kombinierte Ähnlichkeit wird mit Hilfe der Software Matlab und dem integrierten Werkzeugkasten für neuronale Netze bestimmt. Gelernt werden soll die Differenz zwischen dem automatisch berechneten Ähnlichkeitswert und dem von Menschen eingeschätzten Ähnlichkeitswert. Die Elemente des neuronalen Netzes ergeben sich aus:

Eingabevektor des Netzes $\mathrm{P}_k = (a_1, a_2, ..., a_n)$,

Zielvektor des Netzes $\mathrm{T}_k = (y_1, y_2, ..., y_n)$,

Ein- und Ausgabevektor für versteckte Schichten: $\mathrm{S}_k = (s_1, s_2, ..., s_n)$,
$\mathrm{B}_k = (b_1, b_2, ..., b_n)$,

Ein- und Ausgabevektor für Ausgabeschicht: $\mathrm{L}_k = (l_1, l_2, ..., l_n)$, $\mathrm{C}_k = (c_1, c_2, ..., c_n)$,

Gewichte zwischen Eingabe- und mittlerer Schicht: w_{ij}, $i = 1, 2, ..., n,\ j = 1, 2, ..., p$,

Gewichte zwischen mittlerer und Ausgabeschicht: w_{jt}, $j = 1, 2, ..., n,\ t = 1$,

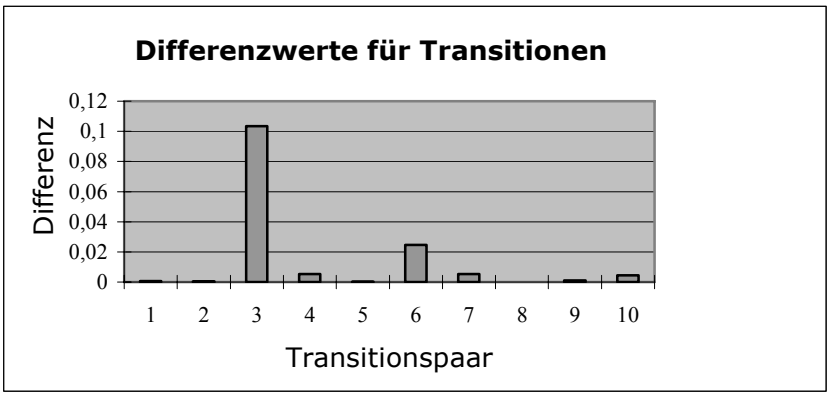
Schwellwerte für die mittlere Schicht: θ_j, $j = 1, 2, ..., p$,

Schwellwerte für die Ausgabeschicht: γ_j, $j = 1, 2, ..., p$,

Parameter $k = 1, 2, ..., m$.

Zunächst werden stochastische Zahlen zwischen [-1, 1] an jedes Gewicht w_{ij}; v_{jt} und die Schwellwerte θ_j; γ_j verteilt (siehe Schritt 1 des Backpropagation-Algorithmus). Dieser Schritt wird Initialisierung genannt. Danach müssen der Eingabevektor P_k und der Zielvektor T_k festgelegt werden. In unserem Fall sind die Eingabevektoren [sim_{syn} sim_{ling} sim_{str}]. Die Differenzwerte von Transitionen befinden sich in einem Intervall von [0, 0.1034]. Als Zielausgabe für das neuronale Netz wird eine Differenz von null angenommen.

Abbildung 66: Statistik der Differenz von Transitionsähnlichkeiten



Für Transitionen gibt es insgesamt 10 Eingabewerte (siehe Tabelle Anhang C) von denen 8 als Lernexemplare und zwei als Testexemplar verwendet werden (siehe Tabelle 20). Als Lernexemplar sollten die Transitionspaare mit der geringsten und der höchsten Differenz verwendet werden, da sie „schlechte" Beispiele darstellen. Als Testexemplare empfiehlt es sich, Paare mit einer mittelmäßigen Differenz zu verwenden. Der Zielvektor ist die Differenz von sim_{men} und sim_{com}, wobei die Gewichte bei Stellen, Transitionen und Attributen unterschiedlich sind.

Tabelle 20: Ein- und Ausgabe des BP-Netzes für Transitionen

Netz für Transitionen	Eingabe			Ausgabe	Zielausgabe
	sim_{syn}	sim_{ling}	sim_{str}	Differenz (sim_{com}-sim_{men})	
Lernexemplar	1	1	0	0.00062001	0
	0.6154	0.7436	0	0.00034596	0
	0.8125	0.875	0	0.10336225	0
	0.5	0.6667	0	0.00529984	0
	0.6471	0.7647	0	0.000256	0
	0.3125	0.3125	0	0.024649	0
	0.2	0.2	0	0.00538756	0
	0	0.3537	0	0.00000529	0
Testexemplar	0.0667	0.1548	0	0.00114244	0
	0.0625	0.0893	0	0.00442225	0

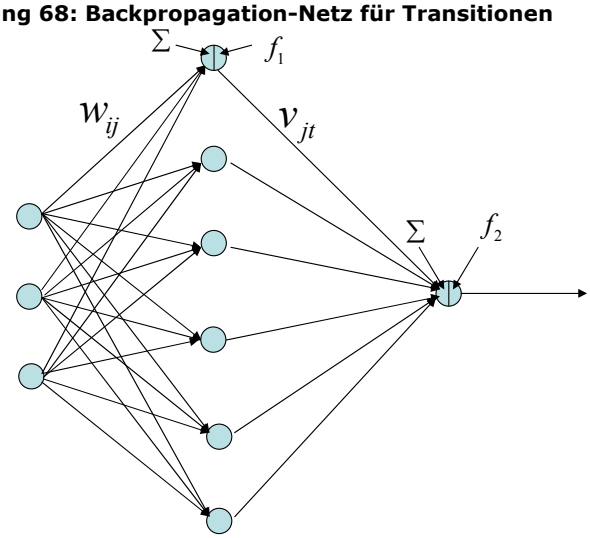
Im nächsten Schritt gilt es die Anzahl an Neuronen in der verstecken Sicht zu bestimmen, die zwischen drei bis acht liegen sollte. Nach einigen Testdurchgängen zeigt sich, dass die mittlere Schicht aus sechs Neuronen am nahesten dem Zielwert von Null liegt und die Anzahl an Berechnung am geringsten ist (siehe Tabelle 21).

Tabelle 21: Ergebnisvergleich mit unterschiedlichen Perzeptronen für Transitionen

Anzahl der Perzeptronen	3	4	5	6	7	8
Anzahl der Berechnungen	212	126	150	62	83	14
Ergebnis	10^{-24}	10^{-26}	10^{-30}	10^{-30}	10^{-27}	10^{-23}

Das resultierende Backpropagation-Netz mit sechs Neuronen zeigt Abbildung 67. Die Gewichte w_{ij} und v_{jt} entsprechen den Werten aus Tabelle 22. Die Aktivierungsfunktion ist eine Tangens Hyperbolicus Funktion. Für die Ausgabeschicht wird eine lineare Aktivierungsfunktion f_2 verwendet.

Abbildung 67: Abbildung 68: Backpropagation-Netz für Transitionen



Die entsprechenden Werte für die Gewichte w_{ij} und v_{jt} lauten:

Tabelle 22: Netzgewichte für Transitionen

w_{ij}	Perzeptron1	Perzeptron2	Perzeptron3	Perzeptron4	Perzeptron5	Perzeptron6
Eingabe1	-4.4899	-1.4703	2.1709	-5.6132	2.5953	4.3377
Eingabe2	1.1068	5.7061	-5.6895	-3.8117	-3.4044	2.6071
Eingabe3	3.0719	2.339	0.1296	5.1723	3.5212	1.4899

v_{jt}	Perzeptron1	Perzeptron2	Perzeptron3	Perzeptron4	Perzeptron5	Perzeptron6
Ausgabe	0.5302	0.2497	-0.0324	-0.0239	-0.0798	0.5326

	Perzeptron1	Perzeptron2	Perzeptron3	Perzeptron4	Perzeptron5	Perzeptron6
θ	-5.3516	5.7312	-2.2062	4.1074	-3.1386	-2.1479
γ	-0.591					

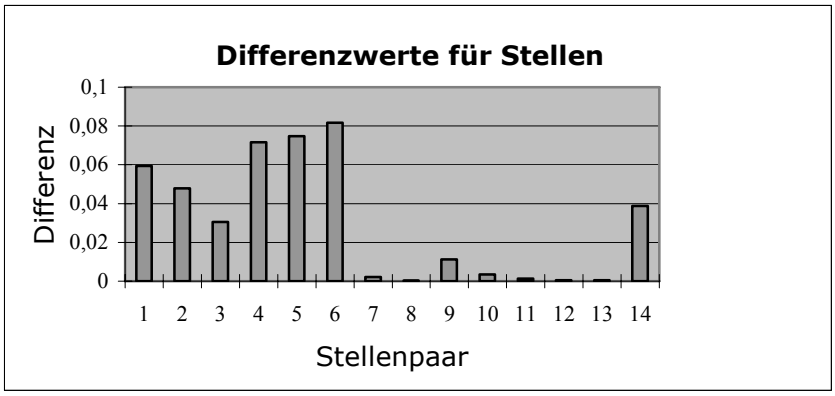
Anschließend wird das Netz noch mit den zwei Testexemplaren überprüft (Testergebnis siehe Tabelle 23). Laut Testergebnis konnte die Differenz zwischen Software und Mensch stark reduziert werden. Das heißt, dass das BP-Netz das Verhalten des Menschen gut simuliert.

Tabelle 23: Testergebnis für Transitionen

	$\text{Differenz}_{\text{alt}}$	$\text{Differenz}_{\text{neu}}$
Test1	0.00114244	0.00037964
Test2	0.00442225	0.00015876

Analog dazu kann ein BP-Netz erstellt werden, das die Gewichte $w_{c_{syn}}$, $w_{c_{ling}}$ und $w_{c_{str}}$ für Stellen bestimmt. Die Differenzwerte von Stellen liegen in einem Intervall von [0.0003, 0.0817].

Abbildung 69: Statistik der Differenz von Transitionsähnlichkeiten



Als Eingabe gibt es 14 Stellenpaare (Exemplare). Die letzten drei werden als Testexemplare verwendet und die übrigen sind Lernexemplare. Die Ein- und Ausgabe-

werte für das BP-Netz zeigt Tabelle 24. Die Zielausgabe ist - wie bei Transitionen - Null.

Tabelle 24: Ein- und Ausgabe des BP-Netzes für Stellen

Netz für Stellen	Eingabe			Ausgabe	Zielausgabe
	sim_{syn}	sim_{ling}	sim_{str}	Differenz (sim_{com}-sim_{men})	
Lernexemplar	0.2308	0.4545	1	0.07155625	0
	0	0.1879	1	0.07469289	0
	0	0	1	0.08173881	0
	0.3571	0.8	0	0.00214369	0
	0.5	0.8	1	0.00031329	0
	0.4615	0.7879	1	0.01115136	0
	0.1429	0.4833	1	0.00344569	0
	0.375	0.375	1	0.00124609	0
	0.2	0.2	1	0.00055696	0
	0.0526	0.0721	1	0.000529	0
	0.2353	0.2402	1	0.03873024	0
Testexemplar	0.625	0.625	1	0.05943844	0
	0.6667	0.7778	1	0.04782969	0
	0.4167	0.6667	1	0.03059001	0

Mit Hilfe der Eingabe- und Ausgabewerte wird das Netz mit den Lernexemplaren 5000-mal durchlaufen. Nach dem 5000ten Durchlauf muss das Netz den Rechenverlauf abbrechen, selbst wenn die Zielausgabe noch nicht erreicht ist. Als Ergebnis ergibt sich:

Tabelle 25: Ergebnisvergleich mit unterschiedlichen Perzeptronen für Stellen

Anzahl der Perzeptronen	3	4	5	6	7	8
Anzahl der Berechnungen	5000	1486	41	135	50	43
Ergebnis	0.000290495	10^{-24}	10^{-29}	10^{-28}	10^{-26}	10^{-27}

Das Ergebnis mit 5 Neuronen ist aufgrund der geringsten Berechnung am besten (am nächsten am Zielwert und geringste Anzahl an Berechnungen). Daraus resultieren die folgenden Gewichte für Stellen:

Tabelle 26: Netzgewichte für Stellen

w_{ij}	**Perzeptron1**	**Perzeptron2**	**Perzeptron3**	**Perzeptron4**	**Perzeptron5**
Eingabe1	-2.0408	4.4260	-3.5262	-6.5032	8.5613
Eingabe2	-6.0469	3.5450	-1.3713	6.9435	0.3552
Eingabe3	-0.8331	0.8592	4.0462	-3.9448	-4.6398

v_{jt}	Perzeptron1	Perzeptron2	Perzeptron3	Perzeptron4	Perzeptron5
Ausgabe	5.4800	-5.0213	1.4206	0.0586	0.6987

	Perzeptron1	Perzeptron2	Perzeptron3	Perzeptron4	Perzeptron5
θ	5.9443	0.4885	-1.3605	0.0786	-1.2325
γ	0.0638				

Laut Testergebnis (Tabelle 27) konnte die Differenz zwischen Software und Mensch stark reduziert werden. Das heißt, dass das BP-Netz das Verhalten des Menschen gut simuliert.

Tabelle 27: Testergebnis für Stellen

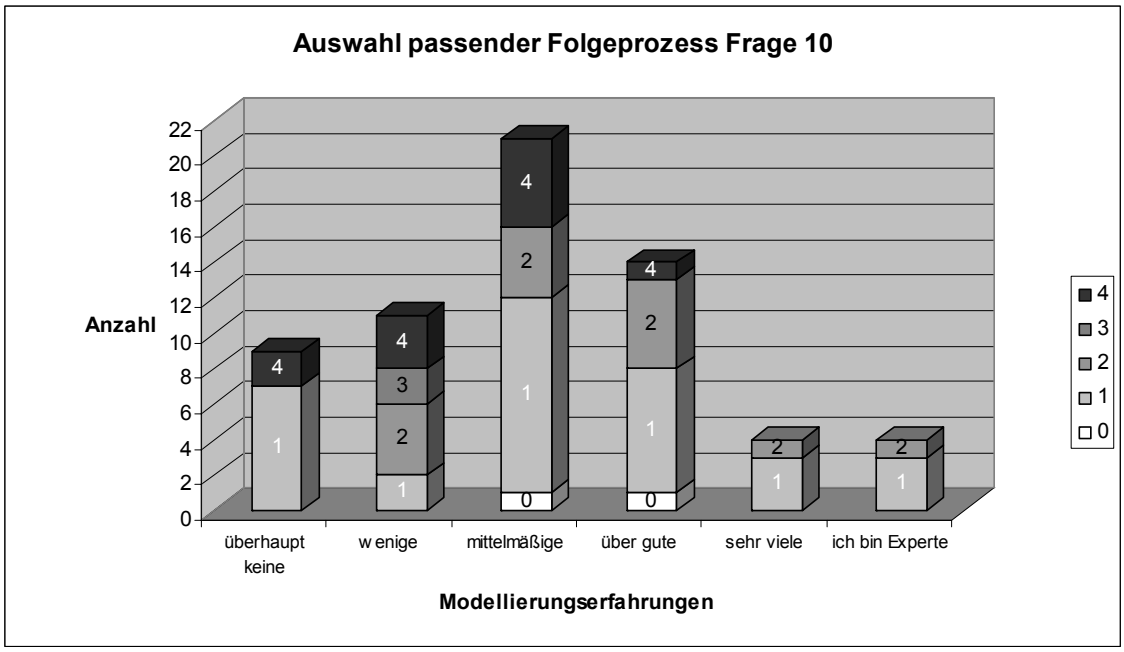
	$Differenz_{alt}$	$Differenz_{neu}$
Test1	0.05943844	0.0183
Test2	0.04782969	0.0013
Test3	0.03059001	0.0069

Die Anzahl an Exemplaren von Attributen ist zu gering (3 Exemplare), um ein sinnvolles Backpropagation-Netz aufzubauen. Bei einer größeren Anzahl an Exemplaren wäre die Berechnung aber analog zum Vorgehen bei Stellen und Transitionen.
Die berechneten Netzgewichte für Stellen und Transitionen lassen sich nicht normalisieren. Die Gewichte werden zufällig berechnet und variieren mit der Anzahl an Testexemplaren.

Clustern von Befragten

Im nächsten Schritt wurde der Fragebogen im Hinblick auf Clusterung von Benutzern ausgewertet. In Frage 10 war ein Ausschnitt eines Geschäftsprozesses angegeben. Die Befragten sollten aus vier vorgegebenen Alternativen einen passenden Folgeprozess auswählen, wobei Mehrfachnennungen möglich waren. Die Ergebnisse für Frage 10 sind in Abbildung 69 veranschaulicht. Die meisten Befragten haben Folgeprozess 1 ausgewählt (laut „Musterlösung" passt dieser Prozessteil 1 auch zum ursprünglichen Prozess). Die meisten Experten und Personen mit sehr viel Modellierungserfahrungen haben sich auf die gleiche Auswahl festgelegt (lediglich 25% der Personen mit sehr viel Modellierungserfahrungen haben noch einen weiteren bzw. einen anderen Folgeprozess ausgewählt). In der Abbildung sind die Mehrfachnennungen aus Übersichtsgründen nicht ersichtlich, werden allerdings beim Clustern berücksichtigt.

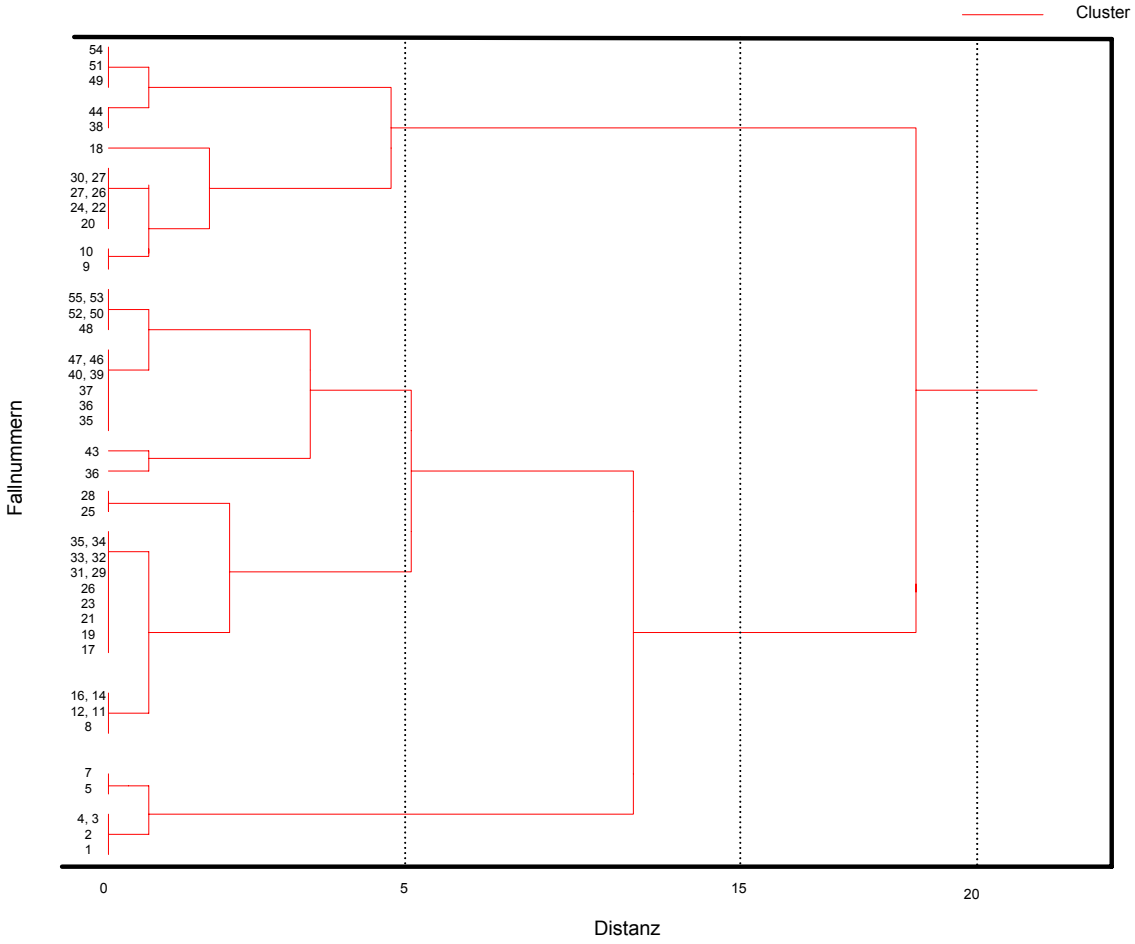
Abbildung 70: Auswahl von passenden Folgeprozessen in Frage 10



Mit der Statistiksoftware WinStat wurden Clusterungen berechnet[75]. Das Ergebnis nach Complete Linkage und Average Linkage war identisch. Die einfachste Clusterung nach Single Linkage schlug grobe Cluster nur auf drei Distanzebenen vor, weshalb dieses Verfahren nicht weiter betrachtet wird. Zur Auswertung der Clusteranalyse wurden die Fragebögen aufsteigend nach Modellierungserfahrungen nummeriert. Die vier Experten haben beispielsweise die Nummern 1, 2, 3 und 4.

[75] Leider sind die Graphiken, die diese Software erzeugt, bei vielen Clustern nicht leserlich. Deswegen wurden die Cluster nachgezeichnet. Die Distanz zwischen Cluster 5 und 15 ist deswegen nicht maßstabsgetreu, damit die Gruppierung bis zu einer Distanz von 5 detaillierter dargestellt werden kann.

Abbildung 71: Clustern von Benutzern nach Complete Linkage für Frage 10

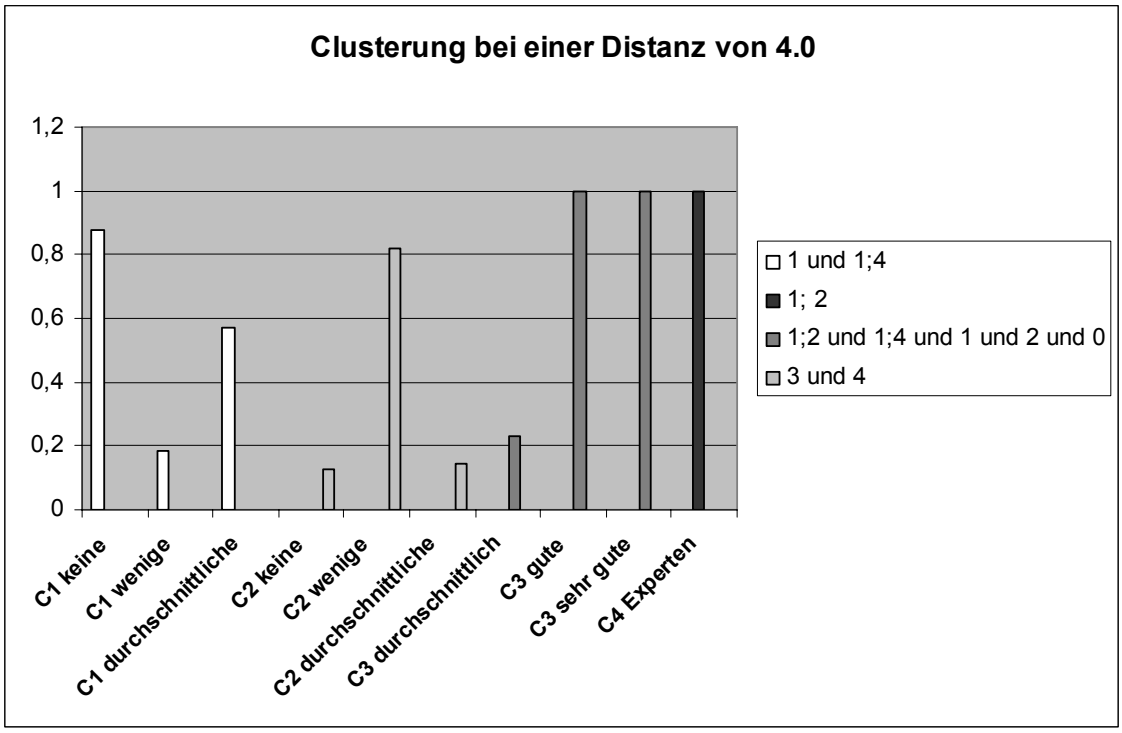


Entsprechend den Mehrfachnennungen der Benutzer werden bei einer Distanz von 0.5 sieben Cluster vorgeschlagen, wobei einige Benutzer noch überhaupt keinem Cluster zugeordnet werden können (z.B. Nr. 46, 42, 28, 37, usw.). Bei einer Ähnlichkeitsdistanz von 4.0 ergeben sich vier Cluster: Cluster 1 und Cluster 2 besteht aus Modellierern mit keinen, wenigen und durchschnittlichen Modellierungserfahrungen. Cluster 3 setzt sich zusammen aus Personen mit durchschnittlichen, guten und sehr guten Modellierungserfahrungen. Den letzten Cluster bilden die Experten.

Wie in Abbildung 71 ersichtlich, wurden Personen mit keinen, wenigen und durchschnittlichen Modellierungserfahrungen in zwei bzw. drei Cluster eingeordnet. Im dritten Cluster ist aber die Anzahl an Personen, die sich für die gleichen Antworten wie gute und sehr gute Modellierer entschieden haben, gering (das gleiche gilt für Cluster 2). Eine viel größere Anzahl an Modellierern mit durchschnittlichen Modellierungserfahrungen hat sich für die gleiche Antwort wie Personen ohne Modellie-

rungserfahrungen entschieden (Cluster 1). Die gleiche Analogie besteht im Cluster 1 für Personen mit wenigen Modellierungserfahrungen.

Abbildung 72: Clusterung bei einer Ähnlichkeitsdistanz von 4.0 für Frage 10

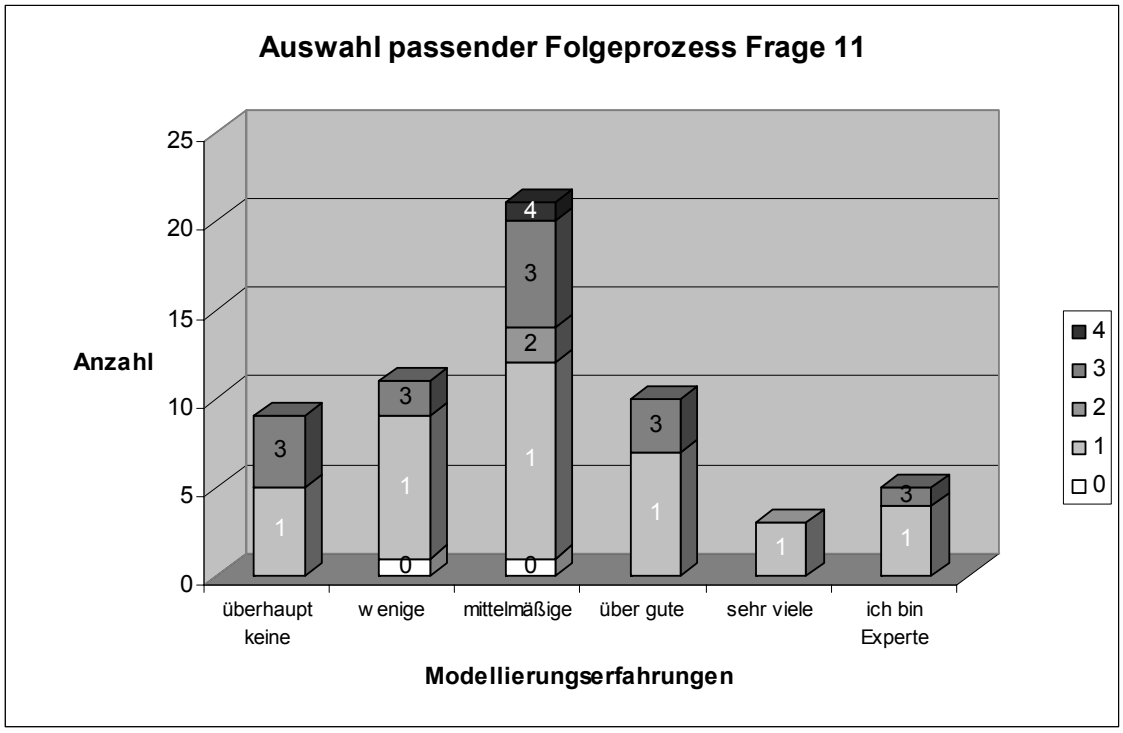


Deswegen ergeben sich anhand der Antworten aus Frage 10 die folgenden Cluster:

- **Cluster 1:** keine und durchschnittliche Modellierungserfahrungen
- **Cluster 2:** wenige Modellierungserfahrungen
- **Cluster 3:** gute und sehr gute Modellierungserfahrungen
- **Cluster 4:** Experten

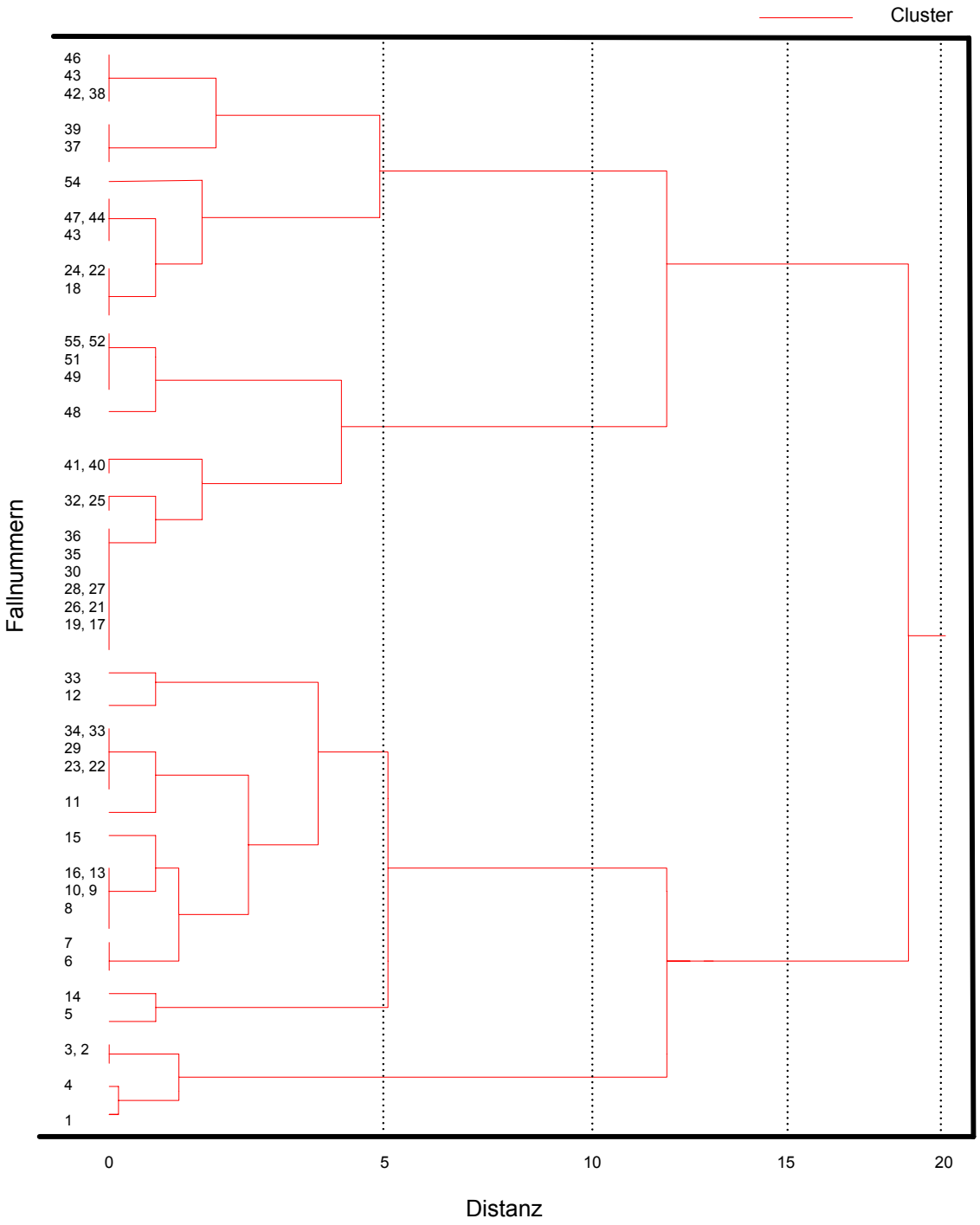
Bei der Frage 11 mussten die Befragten ebenfalls aus vier Alternativen den passenden Folgeprozess auswählen (Mehrfachnennungen waren wieder möglich). Im Gegensatz zur Frage 10 musste der Folgeprozess einer vorgegebenen Geschäftsregel entsprechen. Die Antworten der Befragten zeigt Abbildung 72. Die meisten Befragten haben sich wieder für den ersten Folgeprozess entschieden.

Abbildung 73: Auswahl von passenden Folgeprozessen in Frage 11



Ausgehend von diesen Antworten wurden Clusterungen nach Complete Linkage und Average Linkage durchgeführt. Sie führte wieder zu ähnlichen Vorschlägen wie für Frage 10.

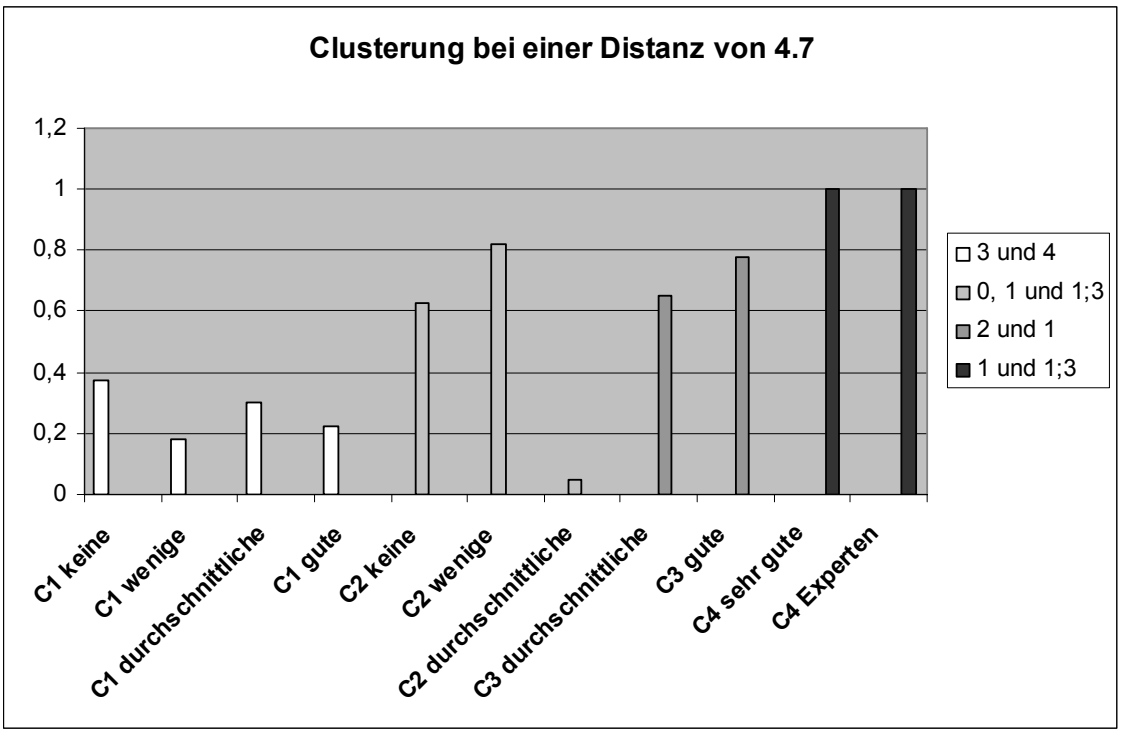
Abbildung 74: Clustern von Benutzern nach Complete Linkage für Frage 11



Bei einer Ähnlichkeitsdistanz von 4.7 werden 4 Cluster gefunden. Wieder gibt es Gruppen von Modellierern, die mehreren Clustern zugeordnet werden. Personen ohne Modellierungserfahrungen werden zu Cluster 1 und 2 zusammengefasst. Diejenigen, die angaben wenige Modellierungserfahrungen zu besitzen, werden Cluster 1 und 2 zugeordnet. Zweifache Clusterzuordnung gilt auch für Personen mit durchschnittlichen und guten Modellierungserfahrungen. Die gleichen Antworten von den

meisten Personen mit gar keinen Modellierungserfahrungen finden sich im Cluster 2. Gleiches gilt für Personen mit guten und durchschnittlichen Erfahrungen; die meisten gleichen Antworten finden sich im Cluster 3.

Abbildung 75: Clusterung bei einer Ähnlichkeitsdistanz von 4.7 für Frage 11

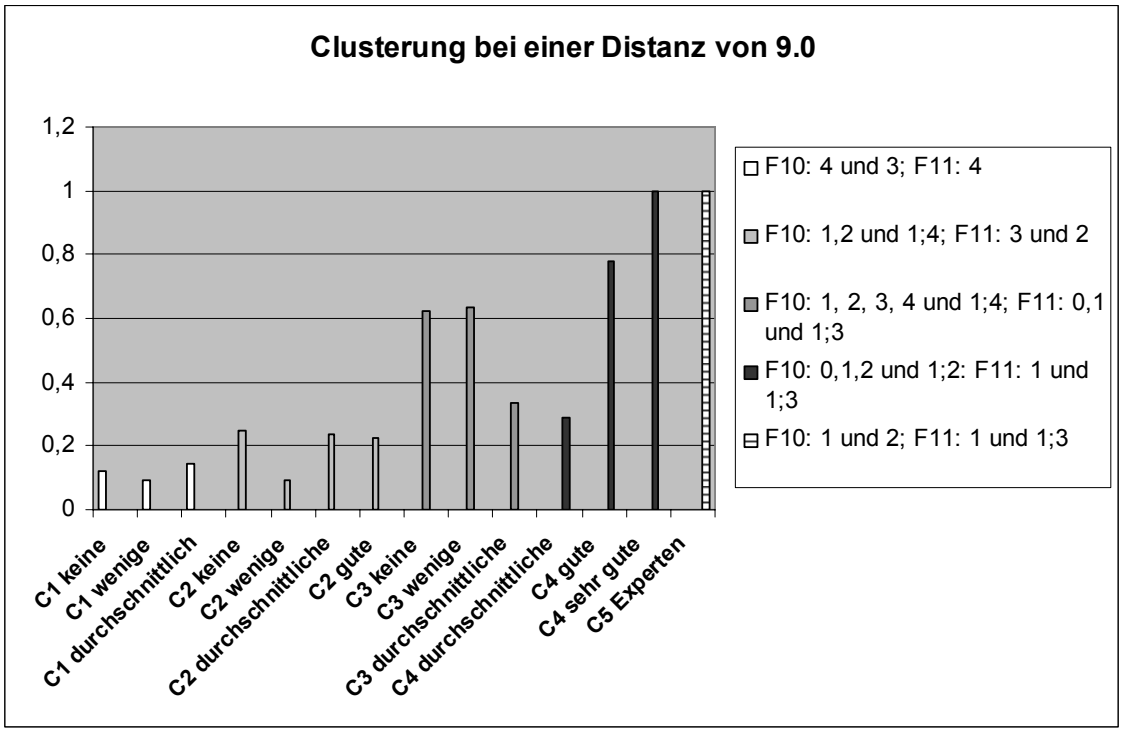


Ausgehend von den Antworten in Frage 11 und unter Berücksichtigung der Modellierungserfahrungen ergeben sich die folgenden Cluster:

- **Cluster 1:** keine und wenige Modellierungserfahrungen
- **Cluster 2:** durchschnittliche und gute Modellierungserfahrungen
- **Cluster 3:** sehr gute Modellierungserfahrungen und Experten

Damit ergeben sich ausgehend von den Ergebnissen der Fragen 10 und 11 unterschiedliche Cluster. Abbildung 75 zeigt die Clusterung bei einer Ähnlichkeitsdistanz von 9.0 für die beiden Fragen 10 und 11. Einige Befragte wurden wieder mehreren Cluster zugeordnet.

Abbildung 76: Clusterung bei einer Ähnlichkeitsdistanz von 9.0 für Frage 10 und 11



Insgesamt ergeben sich ausgehend von Frage 10 und 11 die folgenden Cluster:

- **Cluster 1:** keine, wenige und durchschnittliche Modellierungserfahrungen
- **Cluster 2:** gute und sehr gute Modellierungserfahrungen
- **Cluster 3:** Experten

Personen mit durchschnittlichen Modellierungserfahrungen könnten auch dem Cluster 2 zugeordnet werden.

7.3.4 Analyse und Interpretation der erhobenen Daten

Für die Fragen 10 und 11 konnten eindeutige Cluster gefunden werden. Unter Betrachtung beider Fragen konnten Personen mit durchschnittlichen Modellierungserfahrungen zwei Clustern zugeordnet werden. Allerdings gibt es keine Gruppe, die sich eindeutig für einen Folgeprozess entschieden hat. Experten haben sich bei der Frage 10 (siehe Abbildung 69) entweder für den Folgeprozess 1 oder für zwei Folgeprozesse, nämlich 1 und 2, entschieden.

Der Folgeprozess 1 hatte die gleiche Anzahl an Aktivitäten wie der Folgeprozess 3, und in ihm waren zudem auch alternative Prozessaktivitäten modelliert. Damit war dieser im Gegensatz zum Prozess 3 etwas „komplexer", da Folgeprozess 3 nur einen sequentiellen Ablauf hatte. Der gesamte Prozess 3 passt allerdings nicht zum

angegebenen Prozessausschnitt, da der Folgeprozess 3 eine Aktivität des Prozessausschnitts wiederholt (für den Folgeprozess 3 hat sich nur eine geringe Anzahl von Personen entschieden, insbesondere der Cluster mit Personen mit wenigen Modellierungserfahrungen). Im Folgeprozess 2 gibt es auch eine Aktivität, die bereits im vorgegebenen Prozessausschnitt modelliert wurde. Für den Folgeprozess 2 haben sich weitaus mehr Personen entschieden als für den Folgeprozess 3.
Für die Modellierungsunterstützung bedeutet das, dass einige Benutzer eine Wiederholung von Aktivitäten akzeptieren würden, im Prinzip aber sollte das vorzuschlagende Prozessteil keine Aktivität wiederholen. Hier sollte eine Abprüfung der Prozesselementnamen stattfinden, und der Modellierer sollte auf eine Wiederholung von Aktivitäten aufmerksam gemacht werden.
Zwar haben sich die meisten Befragten für den Folgeprozess 1 entschieden, aber keine Gruppe hat nur einen einzigen Folgeprozess ausgewählt. Damit kann die automatische Prozessergänzungsfunktion mehrere Folgeprozesse enthalten; die Anzahl sollte aber begrenzt sein.
Im Durchschnitt hat die Beantwortung eines Fragebogens 30 Minuten in Anspruch genommen. Die Beantwortung der Fragen 10 und 11 hat insgesamt etwa acht Minuten gedauert. Um die Modellierungszeit durch Auswahl eines passenden Folgeprozesses nicht unnötig zu verlängern, wäre es empfehlenswert, dem Benutzer nicht mehr als vier, eher nur drei Folgeprozesse vorzuschlagen, wie sie im Fragebogen vorgeschlagen wurden. Wobei Experten für die Auswahl passender Folgeprozesses weniger Zeit beansprucht hatten als Anfänger. Damit könnten Experten mehr Folgeprozesse vorgeschlagen werden als Anfängern.
Bei der Frage 11 mussten die Befragten bei der Auswahl eines passenden Folgeprozesses eine Geschäftsregel beachten. Zwar hat sich wieder kein Cluster eindeutig für einen Prozess ergeben, doch ist die Anzahl an ausgewählten Prozesskombinationen geringer als in Frage 10. D. h., je konkreter der zu modellierende Prozess, desto präziser die Auswahl an passenden Folgeprozessen.
Für die Modellierungsunterstützung könnte dies eine kleinere Anzahl an Vorschlägen implizieren. Von den drei Clustern für Frage 11 wurden nur drei Folgeprozesse ausgewählt. Überhaupt nicht ausgewählt wurde Folgeprozess 4. In diesem Prozess gibt es auch keine Aktivität, die einen Teil der Geschäftsregel abbildet. Damit müssen die vorzuschlagenden Folgeprozesse zur Einhaltung der Geschäftsregel beitragen.
In Frage 11 ist der Folgeprozess 2 mit vier Aktivitäten der kürzeste Prozess. Dieser Prozess könnte sich als Folgeprozess eignen, verändert aber nicht signifikant den Geschäftsprozess. Der am häufigsten ausgewählte Folgeprozess 1 besteht aus sechs Aktivitäten und beschreibt die Durchführung von drei Aktivitäten (Fertigung,

Sendung und Bezahlung der Ware). Diese drei Aktivitäten verändern den Gesamtprozess stärker als Folgeprozess 2. Dies könnte für die Modellierungsunterstützung bedeuten, dass der vorzuschlagende Folgeprozess mindestens sechs Aktivitäten enthalten sollte. Über die maximale Länge lässt sich nur schwer eine Aussage anhand der Stichprobe machen. Einige Befragte gaben an, dass sie lieber häufiger unterstützt werden möchten, als erst einmal Prozesse mit 15 Aktivitäten nachvollziehen zu müssen. Ein erster Richtwert für die Länge von Folgeprozessen könnte bei sechs bis zu zehn Aktivitäten liegen (da der passende Folgeprozess 2 mit fünf Aktivitäten fast gar nicht ausgewählt wurde und mit weniger als fünf Aktivitäten kein sinnvolles Prozessfragment modelliert werden kann).

Abbildung 75 zeigt die Clusterung von Befragten für die Antworten aus den Fragen 10 und 11. Ausgehend von diesen Daten wurden für die Fragen 10 und 11 drei Cluster gebildet. Die Anzahl an ausgewählten Folgeprozessen ist im Cluster 1 am größten. Die Experten haben sich für die wenigsten Folgeprozesse entschieden. Die Unentschlossenheit beim Cluster 1 könnte auf eine mögliche Hilfestellung bei der Prozessmodellierung hinweisen. Die automatische Unterstützung von Experten bei der Prozessmodellierung sollte zum einen eine geringere Anzahl an Folgeprozessen enthalten, und vor allem muss es sich um passende Folgeprozesse handeln.

Ein letzter wichtiger Punkt, der bei der automatischen Prozessergänzung beachtet werden müsste, ist die Umbenennung von Synonymen. Falls ein Folgeprozess einer Geschäftsregel entspricht, allerdings im Folgeprozess nur Synonyme vom bearbeitenden Geschäftsprozess verwendet werden, dann sollten die Synonyme im Folgeprozess durch die verwendeten Begriffe im bearbeitenden Geschäftsprozess umbenannt werden. Ansonsten könnte es sein, dass der Benutzer einen Vorschlag ablehnt, weil dort nicht die entsprechenden Begrifflichkeiten verwendet werden.

8 Entwicklung eines Prototyps für die Modellierungsunterstützung von Geschäftsprozessen

In den Kapitel 3, 4, 5, 6 und 7 wurden eine Erweiterung eines bestehenden XML-basierten Austauschformats, ein OWL-basiertes Beschreibungsformat, sechs Ähnlichkeitsmaße für Geschäftsprozessmodelle, die Anfragesprache SiMQL und die Modellierungsunterstützung für Geschäftsprozesse in Form einer Empfehlungskomponente vorgestellt. Die Methoden wurden in einem Prototyp namens Semantic Petri net Tool (SemPeT, http://aifbserver.aifb.uni-karlsruhe.de/sempet/index.htm) umgesetzt.

8.1 Inhaltliche und technische Anforderungen

Um einen flexiblen Einsatz von SemPeT in betrieblichen Anwendungen zu ermöglichen, wurden verschiedene inhaltliche und technische Anforderungen definiert, die von dem Prototyp zu erfüllen sind. Auf Basis der in den vorangehenden Kapiteln vorgestellten Methoden wurden folgende inhaltliche Anforderungen festgelegt:

- **Graphische Repräsentation von Prädikate/Transitionen-Netzen**
  Anwendern des Prototyps muss es möglich sein, Geschäftsprozesse mit Petri-Netzen (insbesondere mit Prädikate/Transitionen-Netzen) modellieren zu können.

- **Unterstützung eines XML-basierten Austauschformats**
  Als allgemeingültiges Speicherformat oder maschinenlesbares Beschreibungsformat soll der Prototyp eine XML-Schnittstelle anbieten.

- **Unterstützung eines OWL-basierten Beschreibungsformats**
  Eine automatische Verarbeitung und Manipulation der Geschäftsprozesse soll mit einer OWL-Schnittstelle unterstützt werden.

- **Benutzerdialog zur Ähnlichkeitsmessung**
  Der Prototyp soll helfen, Ähnlichkeitsberechnungen zwischen Geschäftsprozessmodellen durchführen zu können. Hierfür sollte ein Dialogfenster anstelle einer Konsoleneingabe angeboten werden, damit auch Informatiklaien das System bedienen können.

- **Benutzerdialog für OWL-basierte Anfragesprache**
  Aus einer Vielzahl von Geschäftsprozessen sollen Anwender die für sie relevanten Folgeprozesse mit Hilfe einer Anfragesprache extrahieren können. In einem entsprechenden Dialogfenster können Benutzer ihre Anfragen formulieren.

- **Benutzerdialog für die Modellierungsunterstützung**
  Anwender des Prototyps (insbesondere Modellierungsanfänger) sollen bei der Modellierung von Geschäftsprozessen unterstützt werden. Voraussetzung für die Modellierungsunterstützung ist ein Prozessrepository (Sammlung von Geschäftsprozessfragmenten) aus dem passende Folgeprozesse ausgewählt werden können. Das System soll den Anwender dahin gehend bei der Modellierung unterstützen, dass nur die für den Anwender relevanten Folgeprozesse aus dem Repository bereitgestellt werden.

Als technische Anforderungen wurden festgelegt:

- **Nutzung von Standard-Entwicklungskomponenten**
  Durch Berücksichtigung von Standards (Java, XML) bei der Entwicklung des Prototyps soll eine Kompatibilität der Produkte und eine spätere einfachere Erweiterbarkeit des Systems durch unveränderten Zugriff auf Klassen sichergestellt werden.

- **Möglichkeit zur Nutzung von unterschiedlichen Graphikkomponenten**
  Eine Trennung zwischen der Logik- und der Präsentationssicht des Systems soll die Wiederverwendung von bestehenden Graphikbibliotheken erleichtern und unterstützen.

8.2 Komponenten des Prototyps

Als Entwicklungsumgebung für den Prototyp wurde Eclipse[76] zunächst in Version 3.0 und später 3.1 verwendet. Die Java-Programmierumgebung von IBM ermöglicht eine automatische Codevervollständigung und bietet zahlreiche Integrationsmöglichkeiten für Programmierschnittstellen (APIs). Für die Programmiersprache Java sprechen nicht nur die Objektorientierung, die Plattformunabhängigkeit und die weite Verbreitung. Insbesondere gibt es für Java auch eine Vielzahl von weiterentwickelten Open-Source-Komponenten, die eine effiziente Erstellung und Verarbeitung von XML- und OWL-Dokumenten unterstützen. SemPeT ist unterteilt in Entwicklungs-, Logik-, und Präsentationskomponenten, die eine Trennung zwischen der Präsentation und der Logik der Applikation sicherstellen.

8.2.1 Entwicklungskomponente

Für die Entwicklungskomponente wurden Bibliotheken und Programmierschnittstellen verwendet, die in Java implementiert wurden. Als Hauptentwicklungskomponenten wurden in SemPeT JDOM, JENA2, KAON2, SWT, Draw2D und JFace benutzt.

JDOM: Java API für XML

JDOM[77] ist eine Java-basierte Bibliothek zum Lesen und Manipulieren von XML-Dokumenten; sie ist als Ergänzung bzw. Erweiterung von DOM[78] zu verstehen [Seeb02]. JDOM bietet im Gegensatz zu DOM die Möglichkeit, den vollständigen Baum eines XML-Dokuments auszugeben. Bei DOM muss vor dem Aufruf von Textinhalten immer erst zum jeweiligen Textknoten navigiert werden; der Aufruf von Textinhalten von einem übergeordneten Elementknoten aus ist nicht möglich. Neben der JDOM-API muss ein DOM-Parser installiert werden (z.B. Xerces), da JDOM keinen eigenen Parser enthält. JDOM besitzt mehrere Pakete, von denen org.jdom die Klasse `Document` enthält, die ein XML- bzw. DOM-Dokument repräsentiert.

Quelltext 27 zeigt die Erzeugung einer Instanz mit dem Namen `pnmlElement`, dessen Ausgabe in `myDocument` geschrieben wird.

[76] www.eclipse.org
[77] http://www.jdom.org/
[78] Document Object Model (DOM) ist eine Programmierschnittstelle für den Zugriff auf HTML- oder XML-Dokumente und wird seit 1998 vom W3C-Konsortium gepflegt und weiterentwickelt. Ziel von DOM ist die Entwicklung einer plattformunabhängigen Schnittstelle für die Verarbeitung von XML-Dokumenten.

Quelltext 27: Programmcode für die Erzeugung der Instanz `Document` mit JDOM

```
// Erzeuge ein Dokument
     Element pnmlElement = new Element("pnml");
     Document myDocument = new Document(pnmlElement);
```

JENA2: Java API für RDF/OWL

Die auf Java basierende JENA2-API[79] beinhaltet eine Reihe von Klassen und Methoden zur Manipulation von RDF und OWL. Im Paket com.hp.hpl.jena.vocabulary sind sämtliche Syntaxelemente von RDF, RDF Schema und OWL als Attribute der Vokabularklassen RDF, RDFS bzw. OWL implementiert. Im Paket com.hp.hpl.jena.rdf.model ist unter anderem die Klasse `ModelFactory` enthalten, welche Methoden definiert, mit denen unterschiedliche speicher- oder datenbankbasierte RDF- bzw. OWL-Modelle erzeugt werden können. In diesem Paket und in dem Paket com.hp.hpl.jena.ontology unterstützen die Methoden `Model`, `Resource`, `Property` und `Literal` das Erzeugen, Ändern und Anfragen von RDF-Elementen bzw. die Methoden `OntModel`, `OntClass`, `ObjectProperty`, `DatatypeProperty`, `Restriction` und `Ontology` das Erzeugen und Ändern von OWL-Elementen.
Quelltext 28 zeigt einen Ausschnitt aus dem Programmcode zur Implementierung eines Ontologiekonzepts und dessen Eigenschaften. Dabei wurde die Methode `createClass` in der Klasse `petri` definiert und durch einen *Resource type* (`PETRI`) und einen String (`"Place"`) beschrieben.

Quelltext 28: Programmcode für die Erzeugung eines Konzepts und einer Eigenschaft mit JENA2

```
// Erzeuge das Konzept Place und die Eigenschaft transRef
public static final OntClass Place=petri.createClass(PETRI+"Place");
transRef.addDomain(Place);
```

KAON2: Java API für OWL

KAON2 (Karlsruher Ontology Management System) stellt eine Alternative zu JENA2 für die Manipulation von RDF und OWL dar. Im Gegensatz zu JENA2 bietet KAON2 einen entscheidbaren Schlussfolgerungsmechanismus für OWL [HuMS04] an. Zudem unterstützt KAON2 die Modellierung von Regeln, die mit der Semantic Web Rule Language beschrieben werden (siehe Kapitel 6.2.1.). Die Integration von KAON2 in SemPeT erfolgte nach erfolgreicher Einbindung von JENA2; generell könnte

[79] http://jena.sourceforge.net/

KAON2 die OWL-Serialisierung von Petri-Netzen zu semantischen Geschäftsprozessmodellen ebenfalls übernehmen.

SWT: Widget-Toolkit API

Das Standard Widget Toolkit[80] (SWT) ist in JAVA implementiert und unterstützt die Programmierung von Benutzungsoberflächen. Zur Oberflächenprogrammierung werden nur Klassen des jeweiligen Client-Betriebssystems verwendet, um die Oberfläche im gewohnten Look-and-Feel zu erstellen. Die Erzeugung von Oberflächen ist unter anderem mit den Paketen `org.eclipse.swt`, `org.eclipse.swt.widgets`, `org.eclipse.swt.events` und `org.eclipse.swt.layout` möglich. Das Paket `org.eclipse.swt.widgets` gibt die beiden Objekte `Display` und `Shell` vor. Das Display-Objekt ist für die grundlegende Kommunikation mit dem verwendeten Toolkit und dem darunter liegenden Betriebssystem zuständig, während das Shell-Objekt für das Erzeugen eines Fensters verantwortlich ist. In einem Programm kann es durchaus mehrere Shell-Objekte und damit auch mehrere Fenster gleichzeitig geben. Graphische Vorgaben wie Font oder Layout sind im Paket `org.eclipse.swt.layout` enthalten. Klassen und Methoden, beispielsweise zur Bewegung der Maus, sind im Paket `org.eclipse.swt.events` implementiert.

Draw2D:

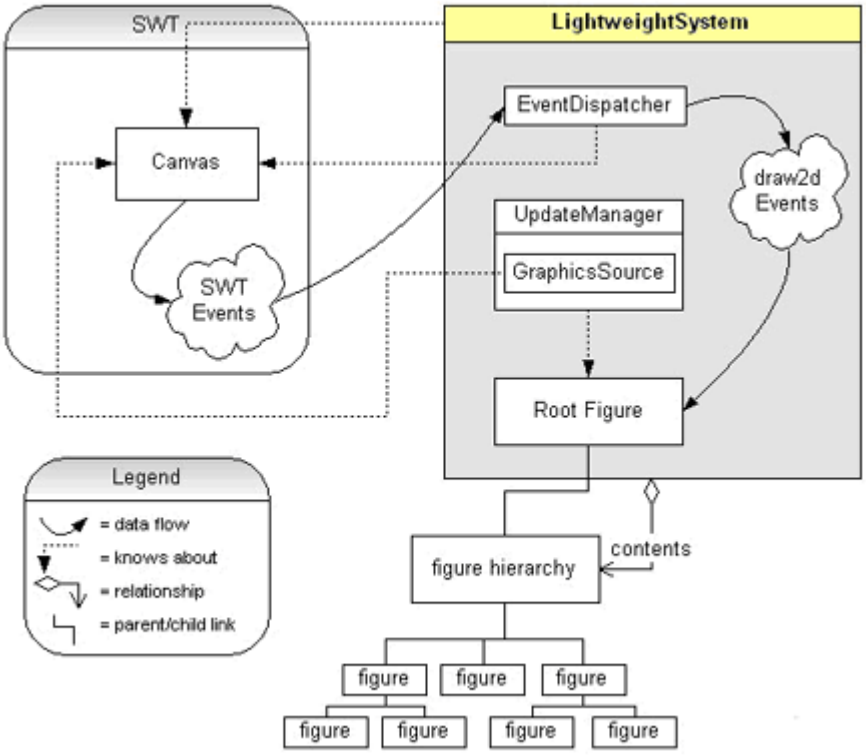
Die Graphikbibliothek Draw2D[81] unterstützt die Erstellung von graphischen Objekten und kann zusammen mit SWT genutzt werden. Das Zusammenspiel von SWT und Draw2D veranschaulicht Abbildung 76[82].

Draw2D-Figuren (IFigure) werden in einem LightweightSystem („Universum, in dem Draw2D-Objekte leben") erstellt und benötigen zunächst eine Zeichenfläche (SWT-Canvas), auf der sie modelliert werden können. Dabei ist jedes Canvas-Objekt eine Figur (IFigure), die wiederum Figuren enthalten kann und auf diese Weise eine Hierarchie bildet. Das Wurzelelement dieser Hierarchie ist die Root Figure.

Das LightweightSystem setzt SWT-Events in passende Draw2D-Events um und passt diese an die Figuren an. Ein Update-Manager stellt sicher, dass eintreffende SWT-PaintEvents sowie Änderungen in den Figures von den betroffenen Figures selbst durch Neuzeichnen sichtbar gemacht werden; zum Zeichnen wird eine eigene Graphics-Klasse vom Update-Manager übergeben; von SWT werden die Layoutkonstrukte FONT, COLORS und IMAGE wieder verwendet.

[80] http://www.eclipse.org/swt/
[81] http://www.eclipse.org/gef/
[82] Das Draw2D Developer's Guide (inkl. Abbildung) ist in der Hilfefunktion von Eclipse beschrieben

Abbildung 77: Zusammenspiel von SWT und Draw2D



Quelltext 29 zeigt die Implementierung der Benutzungsoberfläche von SemPeT inklusive einer Zeichenfläche (ohne Menübar). Die Benutzungsoberfläche (shell) hat eine Größe von 640 mal 480 Pixel. In dieser Benutzungsoberfläche gibt es eine Zeichenfläche (Canvas) mit einer entsprechenden Größe (50, 56, 2000, 2000). In der Zeichenfläche können Draw2D-Figuren gezeichnet werden (LightweightSystem(canvas)).

Quelltext 29: Programmcode zur Implementierung einer Benutzungsoberfläche und einer Zeichenfläche mit SWT und Draw2D

```
public Editor() {
        Display display = new Display();
        Shell shell = new Shell();
        Composite parent = new Composite(shell, SWT.FILL);

        shell.setSize(640, 480);
        Canvas canvas = new Canvas(parent, SWT.NONE);
        canvas.setBounds(50, 56, 2000, 2000);

        Figure contents = new Figure();
        XYLayout contentsLayout = new XYLayout();
        contents.setLayoutManager(contentsLayout);

        petrinet = new PetriNetImpl(contents, shell);

        LightweightSystem lws = new LightweightSystem(canvas);
```

```
        shell.setLayout(new FillLayout(SWT.VERTICAL));

        Figure f = new Figure();
        contents.add(f, new Rectangle(0, 0, 2000, 2000));
        lws.setContents(contents);

        shell.open();
    }
```

JFace:

Die Java-Bibliothek JFace[83] ermöglicht die Programmierung von GUIs und stellt unter anderem Methoden bereit, mit denen Dialoge (jface.dialogs.*), Fortschritte von lang andauernden Operationen (`jface.operation.*`) sowie Ressourcen - wie SWT FONTS und IMAGES - (`jface.resource.*`) verwaltet werden können.

In SemPeT werden alle drei Pakete benutzt; beispielsweise wird die Klasse `ProgressMonitorDialog` im Paket `jface.operation.IRunnableWithProgress` benutzt, um dem Benutzer den Fortschritt der Ähnlichkeitsberechnung anzuzeigen.

8.2.2 Logikkomponente

Die Logikkomponente ist in mehrere Pakete unterteilt, die unterschiedliche Aufgaben erfüllen. Die Umsetzung der in den vorhergehenden Kapiteln beschriebenen Methoden erfolgt mit den Klassen ActionXMLExport (XML-Serialisierung), ActionOWLExport (OWL-Serialisierung) und DialogRecommender (Empfehlungskomponente). Die Ähnlichkeitsmessung und die OWL-Anfragesprache für Ähnlichkeitswerte wurden in den Programmierschnittstellen `edu.unika.aifb.rules.nonOntologies` und `edu.unika.aifb.rules.query` realisiert.

XML-Serialisierung

Die Serialisierung von Petri-Netzen nach XML erfolgt mit der JDOM-API. Die Klasse `ActionXMLExport` enthält die Methode `exportXML()`, die die entsprechenden XML-Elemente aus Petri-Netz-Modellen erzeugt. Zunächst müssen die Pakete importiert werden, die alle notwendigen Werkzeuge für den Umgang mit JDOM bereitstellen. Darüber hinaus müssen auch die Eingabe-/Ausgabeklassen von Java eingebunden werden:

```
import java.io.*;
import org.jdom.*;
import org.jdom.input.*;
import org.jdom.output.*;
```

[83] http://dev.eclipse.org/viewcvs/index.cgi/org.eclipse.jface/

In dieser Klasse werden die XML-Elemente und -Attribute für Pr/T-Netze – wie im nachfolgenden Quelltext 30 auszugsweise veranschaulicht – erzeugt.

Quelltext 30: Programmcode für die XML-Serialisierung von Pr/T-Netzen

//**PNML**-Element wird erstellt

```
Element pnmlElement = new Element("pnml");
```

//**PNML**-Element wird als Wurzelelement eines neuen Dokuments bestimmt

```
Document myDocument = new Document(pnmlElement);
```

//**Net**-Element wird erstellt und dem PNML-Element hinzugefügt

```
Element netElement = new Element("net");
netElement.setAttribute("id", "n1");
netElement.setAttribute("type", "PrTML");
pnmlElement.addContent(netElement);
```

//**Page**-Element wird erstellt und bekommt id- und Pr/T-Netz-Attribute zugewiesen

```
Element pageElement = new Element("page");
pageElement.setAttribute("id", "Pr/T-Netz");
netElement.addContent(pageElement);
```

OWL-Serialisierung

JENA2 erzeugt eine Darstellung von semantischen Geschäftsprozessmodellen in OWL-Syntax mit der Methode `owlExport()`, die in der Klasse `ActionOWLExport` verwendet wird. Zunächst müssen die entsprechenden JENA2-Pakete, die eine OWL-Ausgabe unterstützen, und die Eingabe-/Ausgabeklassen von Java importiert werden:

```
import java.io.*;
import java.util.*;
import com.hp.hpl.jena.rdf.model.RDFWriter ;
```

Die OWL-Elemente der Pr/T-Netz-Ontologie werden in den beiden Klassen `Petri` und `PetriModel` erzeugt und müssen in die Klasse `ActionOWLExport` importiert werden:

```
import org.sempet.owl.Petri;
import org.sempet.owl.PetriModel;
```

In der Klasse `PetriModel` sind die Methoden `createClass()` und `createStatement()` definiert, mit denen die Konzepte und Eigenschaften der Pr/T-Netz-Ontologie instanziiert werden. Quelltext 31 zeigt die Erzeugung einer RDF/XML-Syntax, die zunächst nur aus dem Dokumentnamen (`modelName`), der Eigenschaft `hasNode` und dem Konzept `Place` (Instanzen für `Place` werden über die Methode `getLabel()` zugewiesen) besteht. Für alle Konzepte Place wird zunächst eine OWL-Klasse erzeugt und ein Leerzeichen im Konzeptnamen durch `'_'` ersetzt (send confirmation → send_confirmation), da OWL-Instanzen einer URI entsprechen müssen und kein Leerzeichen erlaubt ist. Anschließend wird eine RDF/XML-Syntax (createStatement) erzeugt.

Quelltext 31: Programmcode für die OWL-Serialisierung der Pr/T-Netz-Ontologie

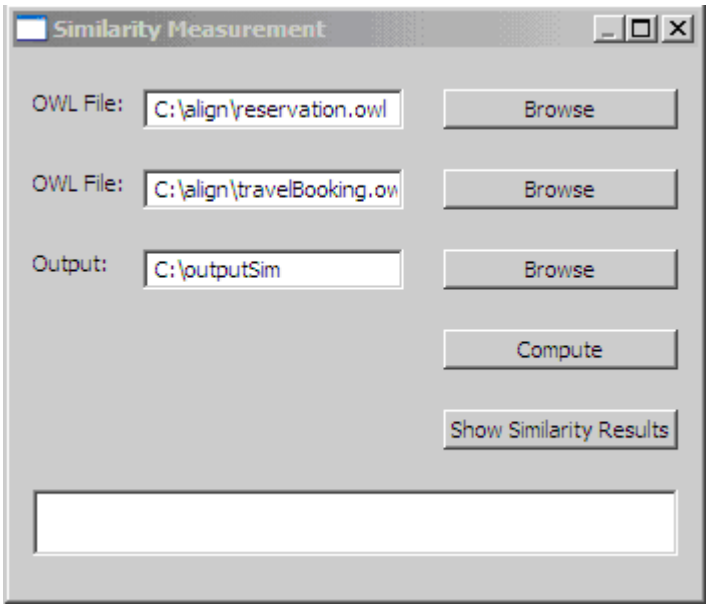
```
// Erzeuge ein OWL-Modell
   modelName = outputFile.getName().replace(".owl", "");
   pm = new PetriModel(modelName);

//Erzeuge die Eigenschaft hasNode und das Konzept Place
   Iterator<Place> itPlaces = petrinet.getPlaces().iterator();
    while (itPlaces.hasNext()) {
       Place p = itPlaces.next();
       pm.createClass(Petri.Place, p.getLabel().replace(' ', '_'));
       pm.createStatement(modelName,Petri.hasNode,p.getLabel().
       replace('','_'));
```

Berechnung von Ähnlichkeiten

Die automatische Berechnung von Ähnlichkeiten zwischen semantischen Geschäftsprozessmodellen wird von dem Alignment-Werkzeug FOAM[84] unterstützt. Da FOAM nur Ähnlichkeitsberechnungen zwischen Konzepten und Eigenschaften in Ontologien unterstützt, war es notwendig, FOAM an die Eigenschaften von semantischen Geschäftsprozessmodellen anzupassen. Das erweiterte und angepasste FOAM-Werkzeug kann in SemPeT unter `Tools` → `Similarity` aufgerufen werden. Für die Ähnlichkeitsmessung muss der Benutzer zwei semantische Geschäftsprozessmodelle in OWL-Syntax auswählen (siehe Abbildung 77) und anschließend einen Ausgabeordner für die Ähnlichkeitswerte bestimmen. Über den `Compute`-Button wird der Algorithmus zur Ähnlichkeitsberechnung aufgerufen. Der Benutzer kann sich die Ergebnisse aus der Ähnlichkeitsmessung über `Show Similarity Results` anzeigen lassen.

[84] Framework for Ontology Alignment and Mapping, http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/

Abbildung 78: Dialogfenster für Ähnlichkeitsberechnungen



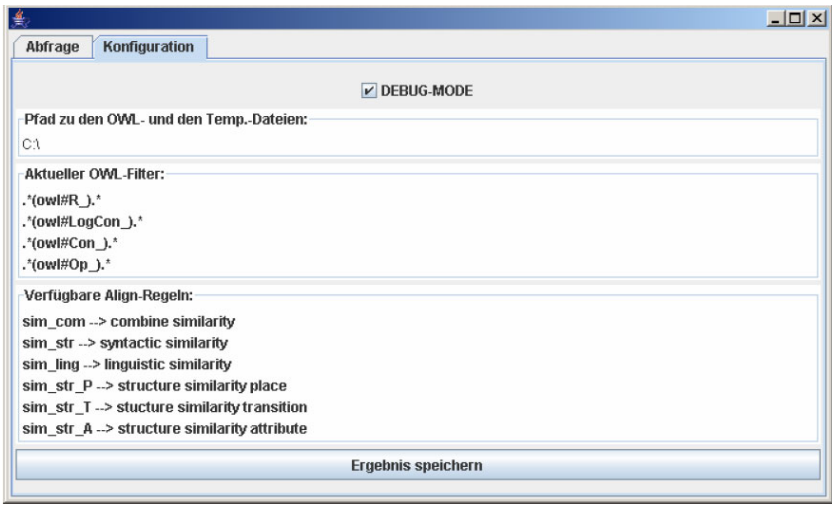
Die Berechnung der Ähnlichkeiten wird in FOAM mit KAON2 realisiert. Zudem werden noch die WordNet-Bibliotheken (Zugang über die API jwnl.jar) zur Berechnung der linguistischen Ähnlichkeit benötigt.

Die Berechnung der syntaktischen Ähnlichkeit ist in der Klasse `SyntacticNumber.java` implementiert. Mit Hilfe von WordNet wird die linguistische Ähnlichkeit über die Klassen `PetriWordnetlookup` und `Wordnetlookup` bestimmt. Die strukturelle Ähnlichkeit wird durch die Kontextelemente, die in der Klasse `PetriRule.java` definiert sind, berechnet. Die Bestimmung des Abstraktionsniveaus zwischen Instanznamen ist in der Klasse `WuPalmerSimilarity` implementiert.

Anfragesprache

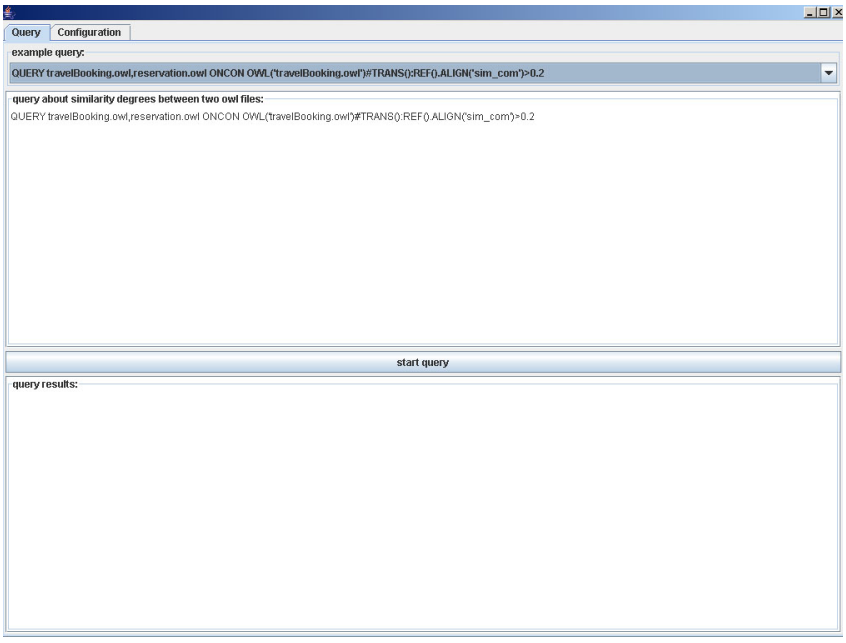
Die Benutzung der Anfrageoberfläche erfolgt über die Klasse QueryDialog. Dabei müssen zunächst die beiden semantischen Geschäftsprozessmodelle sowie ein temporäres Verzeichnis für die Speicherung der Daten aus der Ähnlichkeitsberechnung angegeben werden (siehe Abbildung 78). Eine weitere Konfigurationsmöglichkeit ist die Angabe eines Filters. Ein Filter ist eine Menge von regulären Ausdrücken, mit denen die in den semantischen Geschäftsprozessmodellen enthaltenen Elemente reduziert werden können. Der Filter ergibt sich direkt aus dem Kriterium Adäquatheit, das in Kapitel 5.3 beschrieben wurde. Standardmäßig besteht der Filter aus #R_, #LogCon, #Con_ und #Op_ und entfernt diese Instanzen aus dem Anfrageergebnis.

Abbildung 79: Konfiguration der Anfrage



Die folgende Abbildung zeigt die Eingabemaske für eine konkrete Anfrage. Neben der manuellen Eingabe einer Anfrage kann auch eine der vorhandenen exemplarischen Anfragen aufgerufen und bei Bedarf angepasst werden.

Abbildung 80: Dialogfenster der OWL-Anfragesprache



Nachdem eine Anfrage erstellt und gestartet wurde, wird die Anfrage im QueryMediator verarbeitet. Der QueryMediator übergibt die Anfrage dabei im ersten Schritt an den QueryParser. Konnte der QueryParser die Anfrage verarbeiten, wird anhand des zurückgegebenen Ergebnisses eine Ergebnisstruktur erzeugt. Diese Ergebnis-

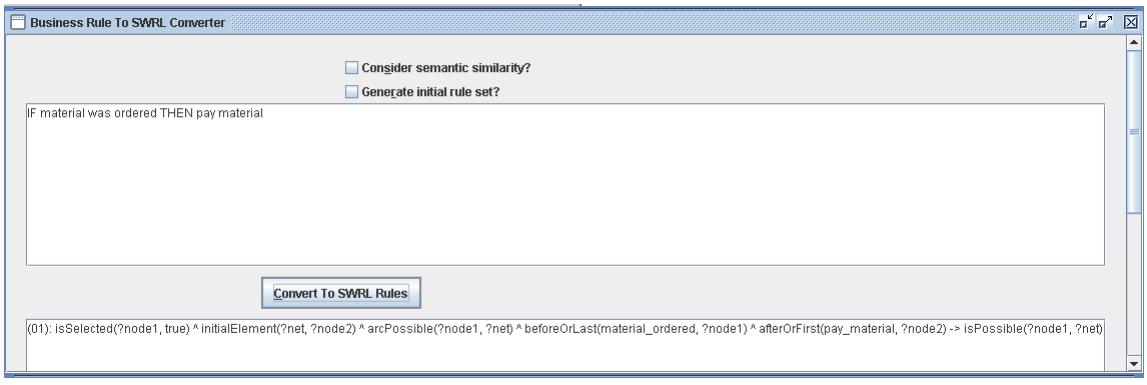
struktur wird in einer Instanz der Klasse QueryResult hinterlegt. Eine solche Query-Result-Instanz wird dann an den QueryDialog zurückgegeben, der das Ergebnis in dem entsprechenden Feld textuell darstellt.

Der QueryParser ist der Kern der Anfragekomponente. Eine Anfrage wird im ersten Schritt auf syntaktische Korrektheit geprüft. Ist diese Prüfung erfolgreich, wird die Anfrage in die einzelnen Teilanweisungen aufgeteilt („Parsing"). Jede geparste Einzelanweisung löst dabei eine Aktion aus, die in einem Puffer zwischengespeichert wird. Nachdem dieser Puffer aufgebaut ist, werden die abzuarbeitenden Aktionen anhand ihrer Funktion in Gruppen und – sofern notwendig – in eine Abarbeitungsreihenfolge gebracht. Die Abarbeitung ist dabei eine Folge von Funktionsaufrufen, die durch die definierte Grammatik gegeben ist. Zur programmatischen Umsetzung der Grammatik wurde das Programm JavaCC[85] verwendet. In der Klasse QueryResult wird ein ermitteltes Ergebnis gespeichert. Ein QueryResult kann entweder textuell durch den QueryDialog dargestellt oder als Datenstruktur weiterverarbeitet werden.

Modellierungsunterstützung

Die im Kapitel 6.3 vorgestellte Übersetzung von natürlichsprachlich beschriebenen Geschäftsregeln in SWRL-Regeln wurde mit einem Regel-Konverter realisiert, da eine manuelle Transformation von Regeln nach SWRL fehleranfällig ist. Zudem werden bei der automatischen Übersetzung Stoppwörter[86] entfernt. Über die SWRL-Regeln kann dann mit der Jess Rule-Engine[87] inferiert werden. Abbildung 80 zeigt die graphische Oberfläche des SWRL-Konverters. Im oberen Fenster wird die Regel mit der Syntax IF…THEN eingegeben und über den Button `Convert To SWRL Rules` nach SWRL konvertiert.

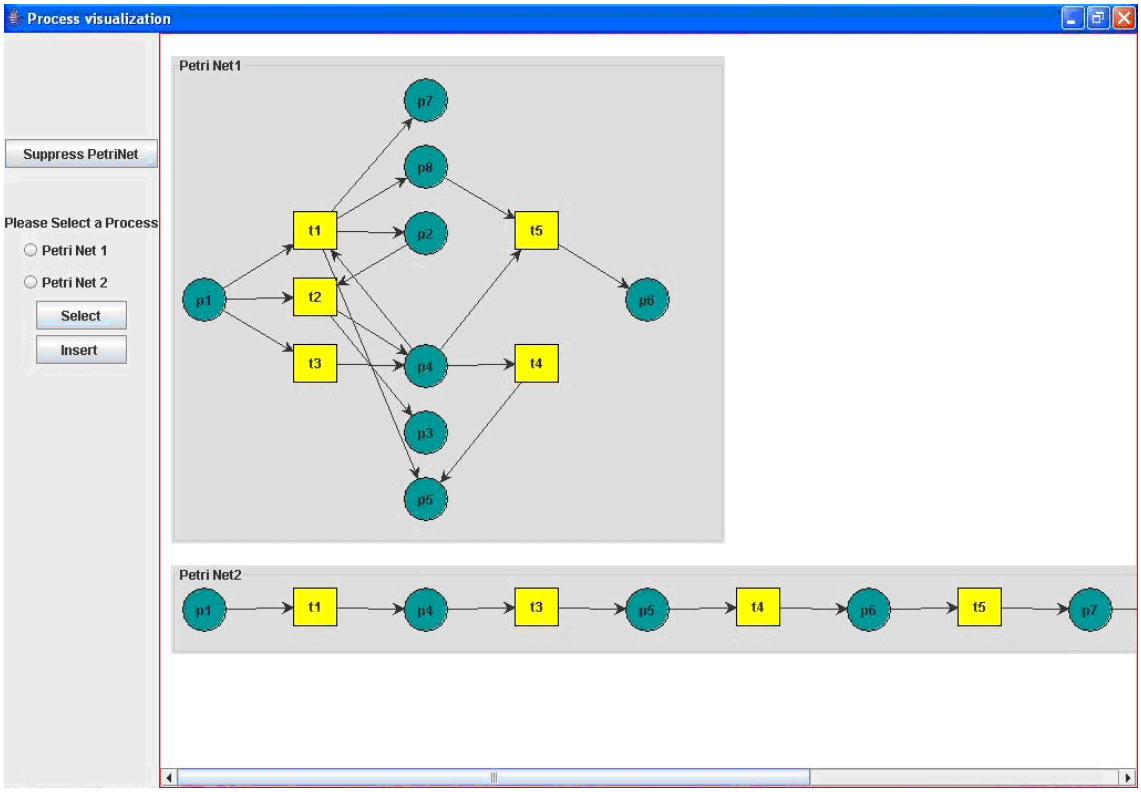
Abbildung 81: Graphische Oberfläche des SWRL-Konverters



[85] Java Compiler Compiler https://javacc.dev.java.net/
[86] http://www.unine.ch/Info/clef/englishST.text; Beispiele für Stoppwörter sind and, or, its, our, etc.
[87] http://herzberg.ca.sandia.gov/jess/

Die Realisierung der Schnittstelle zwischen dem SWRL-Konverter und der Modellierungsoberfläche von SemPeT erfolgte mit JENA und Jess. Die Klasse `DialogRecommendation` enthält die Methode `DialogRecommendation()`, die ein Dialogfenster für die Empfehlungskomponente bereitstellt (siehe Abbildung 81).
Der Benutzer kann den für ihn passenden Folgeprozess über den Button `Select` auswählen. Über den Button `Insert Elements` werden die ausgewählten Prozesselemente in der Zeichenfläche von SemPeT eingefügt.

Abbildung 82: Dialogfenster der Modellierungsunterstützung



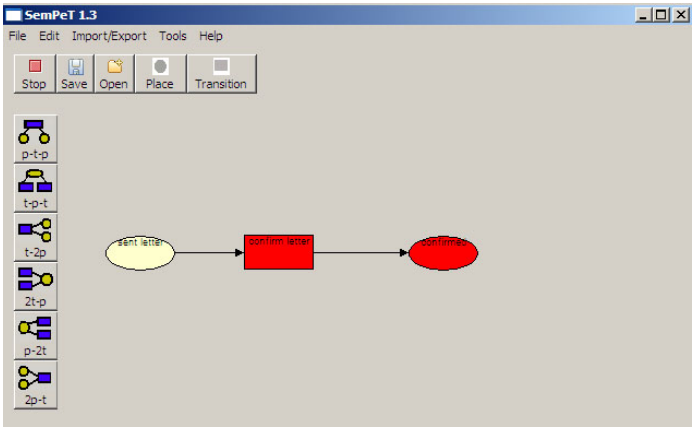
8.2.3 Präsentationskomponente

Die Oberfläche von SemPeT basiert auf SWT, das bereits beschrieben wurde. Der Benutzer kann aus den fünf Menüpunkten File, Edit, Import/Export, Tools und Help die für ihn relevanten Funktionsaufrufe auswählen. Über die Buttons Place und Transition können dann Stellen und Transitionen eingefügt werden. Zur Erzeugung einer Verknüpfung zwischen Stellen und Transitionen (und umgekehrt) muss der Benutzer zunächst die beiden zu verknüpfenden Elemente auswählen. Diese werden dann rot markiert und eine Verknüpfung wird erzeugt. Über die rechte Maustaste (über einem ausgewählten Element) kann der Benutzer die Elemente benennen.

Außerdem können so den Stellen die jeweiligen Attribute und den Transitionen die zugehörigen Inschriften zugewiesen werden. Die linke Menüleiste soll Benutzern die Modellierung erleichtern, indem vorgegebene Modellierungsmuster angeboten werden. So entspricht t-2p der Erzeugung einer Transition, die im Nachbereich zwei Stellen hat.

Der Export nach XML und OWL ist über den Menüpunkt Import/Export möglich. In der SemPeT Version 1.3. wurde bisher noch keine Funktion implementiert, die den Import von OWL- und XML-Dateien unterstützt. Zugang zu Ähnlichkeitsberechnung, Abfragesprache und Empfehlungssystem haben die Benutzer über den Menüpunkt Tools.

Abbildung 83: Oberfläche von SemPeT



SemPeT ist frei verfügbar und kann unter der URL http://aifbserver.aifb.uni-karlsruhe.de/sempet/index.htm heruntergeladen werden.

9 Anwendungsszenarien für semantische Geschäftsprozessmodelle und Ähnlichkeitsmaße

In diesem Kapitel werden mögliche Anwendungsszenarien für Ähnlichkeitsmaße und die OWL-basierte Anfragesprache für Ähnlichkeitsmaße vorgestellt. Im nächsten Abschnitt (9.1) wird eine Methode zur Integration von semantischen Geschäftsprozessmodellen resultierend aus verschiedenen Prozessmodellierungssprachen vorgestellt. Diese unterstützt die Berechnung von Ähnlichkeitswerten zwischen Prozesselementnamen verschiedener Modellierungssprachen, wie sie im Abschnitt 9.2 präsentiert werden.

9.1 Integration verschiedener semantischer Geschäftsprozessmodelle

Unternehmen, die in der gleichen Branche tätig sind, können nicht nur ein unterschiedliches Vokabular zur Beschreibung von Prozesselementnamen, sondern auch eine unterschiedliche Modellierungssprache für Geschäftsprozessmodelle verwenden. Zur Bestimmung der Ähnlichkeit von methodisch unterschiedlich modellierten Geschäftsprozessen müssten Abbildungsregeln definiert werden, damit semantisch korrespondierende Konstrukte der unterschiedlichen Modellierungssprachen identifiziert und verglichen werden können. Zur Berechnung der Ähnlichkeit zwischen Ereignisgesteuerten Prozessketten (EPK) und Petri-Netzen werden EPKs zunächst in OWL-Syntax beschrieben. Anschließend können Ähnlichkeitsmaßen (wie sie im Kapitel 4.6 definiert wurden) für methodisch unterschiedliche Modellierungssprachen angewendet werden.

9.1.1 Vorgehensweise

Zur Integration von semantischen Geschäftsprozessmodellen bieten sich – angelehnt an Ontologieintegrationsmethoden aus der Literatur – zwei Hauptparadigmen

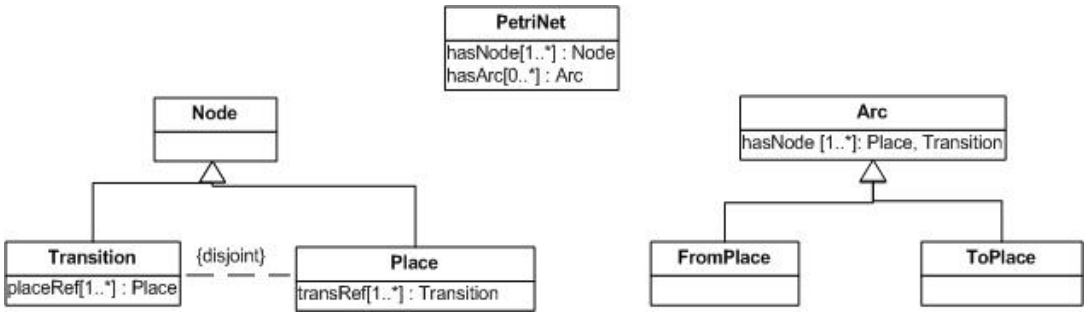
an [NoMu99, PiMa01]: es können Prozessmodelle zu einem zentralen Modell zusammengefasst werden (Ontologie-Zusammenführung) oder Prozessmodelle werden erweitert und aneinander angepasst (Ontologie-Anpassung). Bei der Ontologie-Zusammenführung wird ein einzelnes kohärentes, semantisches Geschäftsprozessmodell erzeugt, das die zwei ursprünglichen semantischen Geschäftsprozessmodelle vereinheitlicht. Wenn zwei semantische Geschäftsprozessmodelle angepasst werden, bleiben die zwei ursprünglichen Prozessmodelle erhalten – allerdings mit einer Vielzahl von Verbindungen, die zwischen ihnen eingerichtet wurden und die es den angepassten Prozessmodellen erlauben, die Informationen der anderen zu nutzen. Die technische Umsetzung des ersten Paradigmas findet sich bei *Mitra*, *Wiederhold* und *Kersten* [MiWK00].

In dieser Arbeit wird zur Bestimmung von Ähnlichkeitsmaßen zwischen EPK-basierten und Petri-Netz-basierten semantischen Geschäftsprozessmodellen eine Vorgehensweise analog dem Integrationsverfahren nach *Rizopoulos* und *McBrien* [RiMc05] und *Schmitt* und *Saake* [ScSa05] abgeleitet. Zur Berechnung von Ähnlichkeiten sind zwei Schritte notwendig: 1. *semantische Geschäftsprozessmodellvorbereitung* und *2*. *semantische Geschäftsprozessmodellabstimmung*.

Semantische Geschäftsprozessmodellvorbereitung

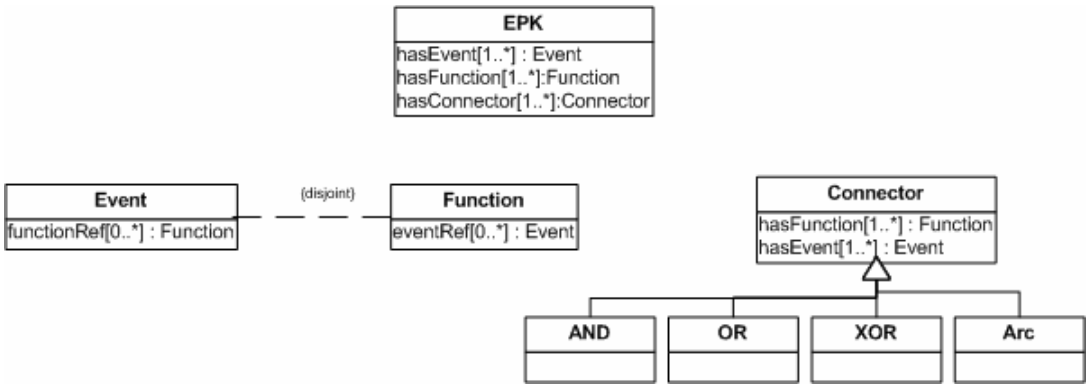
Methodisch unterschiedliche Modellierungssprachen benutzen verschiedene Konstrukte und Beziehungen zwischen den Konstrukten, um den Kontrollfluss von Geschäftsprozessmodellen darzustellen. Für die Identifikation der Struktur und der Beziehungen zwischen den Konstrukten wird ein Metamodell benötigt, das alle Konstrukte eindeutig beschreibt. Zur Darstellung des Metamodells wird die Unified Modeling Language (UML) verwendet. Die Abbildungen 83 und 84 zeigen die Elemente eines einfachen Petri-Netzes bzw. einer Ereignisgesteuerten Prozesskette in UML-Notation. Laut Definition 6 besteht ein Petri-Netz aus einer Menge von Stellen, einer Menge von Transitionen und einer Flussrelation. Im Kapitel 3.2 wurde aufbauend auf der Petri-Netz-Definition zunächst eine Ontologie für elementare Petri-Netze definiert, wie sie in Abbildung 83 veranschaulicht ist.

Abbildung 84: UML-basierte Darstellung der Petri-Netz-Ontologie



EPKs sind eine semiformale Modellierungssprache und bestehen aus Ereignisse, Funktionen und vier Typen von Konnektoren (AND, OR, XOR und Arc). Analog zur Petri-Netz-Ontologie besteht eine EPK-Ontologie aus den Konzepten Event, Function und Connector, der sich in einen AND-, XOR- und OR-Operator und eine direkte Kante aufteilen lässt.

Abbildung 85: UML-basierte Darstellung der EPK-Ontologie



Die UML-Notation der Petri-Netz- und der EPK-Ontologie veranschaulichen die Hauptelemente der beiden Geschäftsprozessmodellierungssprachen. Mit Hilfe dieser Notation können semantisch korrespondierende Konzepte und Beziehungen identifiziert werden.

Semantische Geschäftsprozessmodellabstimmung

Die semantische Geschäftsprozessmodellabstimmung ist der wichtigste Schritt des Integrationsprozesses. Die Konstrukte der beiden zu Grunde liegenden semantischen Geschäftsprozessmodelle (sGPM) sollen in ihrer Bedeutung verglichen werden. Anschließend wird die semantische Beziehung zwischen den Prozesselementen bestimmt. Nach *Rizopoulos* und *McBrien* gibt es vier Typen von semantischen Be-

ziehungen, die anhand ihrer intentionalen Domänen $D_i(A)$ und $D_i(B)$, d.h. alle möglichen validen Instanzen von Elementen A und B, miteinander verglichen werden können:

- Äquivalenz: zwei sGPM-Konstrukte A und B sind äquivalent genau dann, wenn $D_i(A) = D_i(B)$. Man schreibt $A \overset{s}{=} B$.
- Subsumption: sGPM-Konstrukt A ist Teilmenge von sGPM-Konstrukt B genau dann, wenn $D_i(A) \subset D_i(B)$. Man schreibt $A \overset{s}{\subset} B$.
- Durchschnitt: zwei sGPM-Konstrukte A und B besitzen einen Durchschnitt genau dann, wenn $D_i(A) \cap D_i(B) \neq \varnothing$. Wir schreiben $A \overset{s}{\cap} B \neq \varnothing$.
- Disjunktion: zwei sGPM-Konstrukte A und B besitzen keinen Durchschnitt genau dann, wenn $D_i(A) \not\cap D_i(B) = \varnothing$. Wir schreiben $A \not\cap B = \varnothing$.

Für die semantischen Geschäftsprozessmodelle für EPKs und Petri-Netze ergeben sich folgende Äquivalenzen wie in Tabelle 28 dargestellt. Im Gegensatz zu EPKs gibt es bei Petri-Netzen keine Verknüpfungsoperatoren; Operatoren werden in Petri-Netzen implizit und nicht wie bei EPK explizit modelliert; die OWL-Konstrukte *FromPlace* und *ToPlace* der Petri-Netz-Ontologie haben als korrespondierendes Konstrukt in der EPK-Ontologie *Arc* (allerdings mit jeweils unterschiedlichem Wertebereich).

Tabelle 28: Semantische Beziehungen zwischen Konstrukten der Petri-Netz- und EPK-Ontologie

Petri-Netz-Ontologie	**E**PK-Ontologie	Semantische Beziehung
PetriNet	EPK	$P \overset{s}{=} E$
Place	Event	$P \overset{s}{=} E$
Transition	Function	$P \overset{s}{=} E$
ToPlace	Arc (range *hasFunction*)	$P \overset{s}{=} E$
FromPlace	Arc (range *hasEvent*)	$P \overset{s}{=} E$
transRef	functionRef	$P \overset{s}{=} E$
placeRef	eventRef	$P \overset{s}{=} E$
hasNode	hasEvent, hasFunction	$P \overset{s}{=} E$
hasArc	hasConnector	$P \overset{s}{=} E$

In OWL-Syntax wird die Umsetzung der Äquivalenzbeziehung zwischen zwei sGPM-Konstrukten durch die Klasse `<equivalentClass>` bzw. die Eigenschaft `<equivalentProperty>` ausgedrückt. Im Quelltext 32 ist beispielhaft die Äquivalenzbeziehung zwischen `Place` und `Event` und zwischen `placeRef` und `eventRef` in OWL-Syntax modelliert.

Quelltext 32: Äquivalenzen von Klassen und Eigenschaften

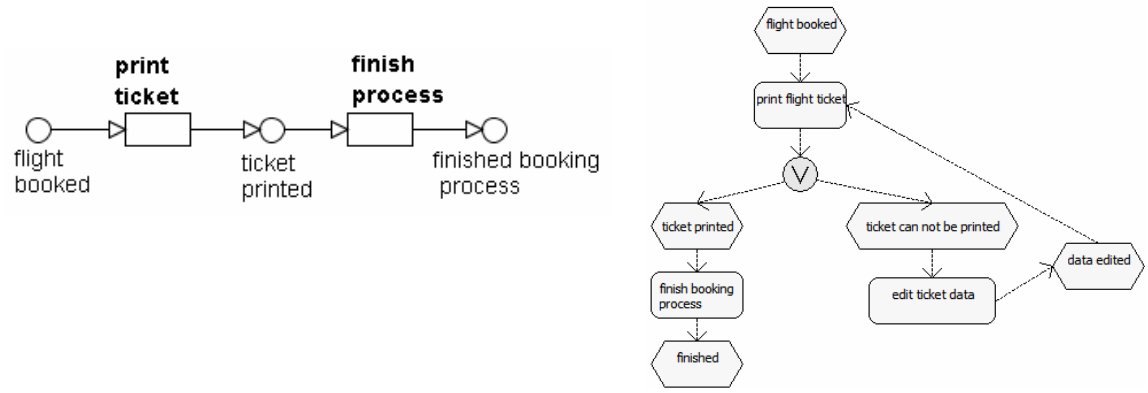
```
<owl:Class rdf:ID="Place">
  <owl:equivalentClass rdf:resource="Event"/>
</owl:Class>


<owl:Property rdf:ID="placRef">
  <owl:equivalentProperty rdf:resource="eventRef"/>
</owl:Class>
```

9.1.2 Realisierung

Am Beispiel des Petri-Netzes und der EPK in Abbildung 85 wird deren Serialisierung in OWL-Syntax erklärt. Beide Prozesse modellieren einen Ausschnitt einer Flugbu-

chung: *„Nachdem ein Flug gebucht wurde, wird das Flugticket versendet und der Buchungsprozess beendet*".

Abbildung 86: Ein Beispiel für ein Petri-Netz und ein EPK



Die Serialisierung von Petri-Netz-Elementen in OWL-Syntax wurde ausführlich im 3. Kapitel erklärt. Analog zu dieser Serialisierung ergibt sich für das EPK aus Abbildung 85 die OWL-Syntax wie im Quelltext 33 wiedergegeben ist. Auf die nachfolgende Funktion eines Ereignisses wird über den Wertebereich der Eigenschaft `eventRef` verwiesen. Über den Wertebereich der Eigenschaft `functionRef` wird auf nachfolgende Funktionen von Ereignissen verwiesen.

Quelltext 33: OWL-Syntax für Petri-Netz und EPK aus Abbildung 85

```
<rdf:RDF>
 <owl:Ontology rdf:about="">
  <owl:import rdf:resource="http://..../PNOntology.owl"/>
 </owl:Ontology>
 <PetriNet rdf:ID=”BookingFlights”>
   <hasNode rdf:resource="#print_ticket"/>
   <hasNode rdf:resource="#flight_booked"/>
   <hasNode rdf:resource="# ticket_printed"/>
   <hasNode rdf:resource="#finished_process"/>
   <hasNode rdf:resource="#booking_process_finished"/>
  <hasArc>
    <FromPlace rdf:ID="flight_booked__print_ticket"/>
      <hasNode rdf:resource="# flight_booked"/>
    </FromPlace>
    <ToPlace
rdf:ID=”finish_process__flight_booking_finshed”/>
      <hasNode rdf:resource="# flight_booking_finished"/>
    </ToPlace>
  ……
  </hasArc>
```

```
<rdf:RDF>
<owl:Ontology rdf:about="">
  <owl:import
rdf:resource="http://..../EPKOntology.owl"/>
 </owl:Ontology>
<EPK rdf:ID=“flightBooking“>
  <hasFunction rdf:resource="#print_flight-ticket"/>
  <hasEvent rdf:resource="#ticket_printed"/>
  <hasEvent rdf:resource="# flight_booked"/>
  <hasEvent rdf:resource="#finished"/>
  <hasEvent rdf:resource="#data_edited"/>
  <hasFunction rdf:resource="#finish_booking-
process"/>
  <hasFunction rdf:resource="#edit_ticket-data"/>
  <hasConnector>
   <Arc rdf:ID=”flight_booked__print_flight-ticket”/>
     <hasEvent rdf:resource”#flight_booked”/>
   </Arc>
   <Arc=”print_ticket__finish_booking_process”/>
   <hasFunction rdf:resource="# fin-
```

```
  <Place rdf:ID="flight_booked">
    <transRef>
      <Transition rdf:ID="print_ticket">
        ……
 </petri:PetriNet>
</rdf:RDF>
```

```
ish_booking_process />
    </Arc>
  ….
  </hasConnector>
  <Event rdf:ID=" flight_booked ">
    <functionRef>
      <Function rdf:ID="print_flight-ticket">
  …..
 </EPK>
</rdf:RDF>
```

9.2 Berechnung der Ähnlichkeit zwischen EPKs und Petri-Netzen

Aufbauend auf den Äquivalenzen in Tabelle 28 werden nur Namen von Ereignissen mit Namen von Stellen sowie Namen von Funktionen mit Namen von Transitionen usw. verglichen. Die einzelnen Ähnlichkeitsmaße sind analog zu Kapitel 4.6abgekürzt. Tabelle 29 zeigt Ähnlichkeitswerte zwischen Prozesselementnamen, die anhand von sGPM für EPKs und Petri-Netze bestimmt wurden. Die einzelnen Ähnlichkeitswerte ergeben sich aus Ähnlichkeitsmaßen, wie sie im Kapitel 4.6 definiert wurden; sim_{com} steht für die kombinierte Ähnlichkeit, sim_{syn} für die syntaktische Ähnlichkeit, usw.

Tabelle 29: Ähnlichkeitswerte für EPKs und Petri-Netzen

Instanzname	sim_{com}	sim_{syn}	sim_{ling}	$sim_{str\ P}$	$sim_{str\ A}$	$sim_{str\ V}$	$sim_{str\ T}$	sim_{abs}
#flight_booked, #flight_booked	0.9	0.1	0.67	1.0	0.0	0.0	0.90	0.0
#print_ticket, #print_flight-ticket	0.8	0.1	0.67	0.0	0.90	0.0	1.0	0.0
#finish_ process, #finish_booking_ process	0.74	0.4	0.0	0.0	0.0	0.0	0.9	1.0
#ticket_printed, #ticket_printed	0.75	0.39	0.0	0.95	0.0	0.0	0.0	1.0

Das Paar *flight_booked vs. flight_booked* hat eine kombinierte Ähnlichkeit von 0.9 und die Gesamtähnlichkeit von den beiden Prozessen in Abbildung 85 wird bei 0.8 liegen. Dieser Ähnlichkeitswert erlaubt Aussagen über die Ähnlichkeit von methodisch unterschiedlich modellierten Geschäftsprozessen zu treffen (in diesem Fall einer Ähnlichkeit von 0.8).

Durch Erweiterung der OWL-basierten Anfragesprache (siehe Kapitel 5) um die Konstrukte Event und Function lassen sich sowohl Petri-Netz-basierte als auch EPK-

basierte Prozessfragmente aus einem Prozessrepository abfragen. Beispielsweise könnten mit der folgenden Anfrage alle Stellen und Ereignisse mit einer kombinierten Ähnlichkeit größer 0.4 abgefragt werden:

Quelltext 34: Anfrage in SiMQL mit erweiterter Syntax

```
QUERY
   *
FROM
   travelBooking.owl,reservation.owl
ONCON
   OWL('travelBooking.owl')#PLACE().ALIGN('sim_com')>0.4,
#Event().ALIGN('sim_com')>0.4
```

Damit können die in dieser Arbeit vorgestellten Methoden auf andere Modellierungssprachen übertragen werden und sind nicht nur auf die Petri-Netz-Syntax beschränkt.

10 Schlussüberlegungen und Ausblick

In diesem Kapitel werden zunächst die wesentlichen Ergebnisse und der Hauptbeitrag dieser Arbeit zusammengefasst. Anschließend werden die technischen und methodischen Grenzen des vorgestellten Forschungsansatzes diskutiert. Die in dieser Arbeit gelegten Grundlagen für ein Unterstützungssystem während der Geschäftsprozessmodellierung können als Ausgangspunkt für weiterführende Konzepte genutzt werden. Abschließend wird daher ein Ausblick auf die sich daraus ergebenden Aufgabenstellungen für zukünftige Forschungsarbeiten gegeben.

10.1 Zusammenfassung

Mit Hilfe eines Modellierungsunterstützungssystems ist es möglich, Modellierern bei der Erstellung von Geschäftsprozessen zielgerichtet zu helfen. Ohne eine solche methodische Unterstützung kann es zum einen leicht zu Modellierungsfehlern in Geschäftsprozessen kommen, weil nicht bereits vorhandene (analysierte) Prozessteile aus einer Prozessbibliothek genutzt werden. Zum anderen hilft eine solche Unterstützung, Zeit bei der Geschäftsprozessmodellierung zu sparen.
Aufbauend auf den Grundlagen zu Ontologien und Geschäftsprozessen wurde die Idee einer Unterstützung während der Prozessmodellierung vorgestellt. Zur Realisierung dieser Idee wurden die einzelnen Komponenten des Unterstützungssystems skizziert, die in den nachfolgenden Kapiteln ausführlich beschrieben wurden.
Das im Kapitel 3 vorgestellte XML-basierte Beschreibungsformat für Petri-Netze kann als Speicherformat für Geschäftsprozessmodelle genutzt werden, die damit werkzeugunabhängig ausgetauscht und visualisiert werden können. Damit lassen sich Petri-Netz-Modelle in anderen Werkzeugen darstellen, benutzerindividuell verändern und zur Modellierungsunterstützung heranziehen. Zusätzlich wurde ein OWL-basiertes Beschreibungsformat für Petri-Netze konzipiert. Dieses Beschreibungsformat ist maschinenverständlich und erlaubt zudem den Einsatz von Infe-

renzmechanismen, wie sie zur Realisierung von Folgeprozessempfehlungen benötigt werden.

Nachdem bei der Modellierung von Geschäftsprozessen in einem unternehmensübergreifenden Anwendungskontext kein einheitliches Vokabular gefordert werden kann, wurden im Kapitel 4 Ähnlichkeitsmaße definiert. Die in der Literatur vorgestellten Ähnlichkeitsmaße für Ontologien und Textdokumente reichen nicht aus, um Ähnlichkeiten zwischen Prozessmodellen bzw. Prozesselementen berechnen zu können. Prozesselementnamen bestehen im Gegensatz zu Ontologiekonzepten aus mehr als einem Begriff (vgl. Elementname *bei hohem Schaden Gutachter beauftragen* vs. Ontologiekonzept *Gutachter*); in der Regel wird ein eindeutiges Substantiv und Verb zur Beschreibung von Prozesselementen gefordert. Textdokumente sind aus Sätzen aufgebaut und damit auch aus mehr Konstrukten als Elementnamen. Des Weiteren spielen bei Geschäftsprozessmodellen unterschiedliche Abstraktionsniveaus eine wichtige Rolle (Prozess-/Subprozessmodelle), die die Konsistenz und leichtere Wiederverwendung von Geschäftsprozessmodellen garantieren sollen. Ein solches unterschiedliches Abstraktionsniveau lässt sich in Textdokumenten nicht direkt ermitteln. Ein weiteres Unterscheidungskriterium zwischen Ontologien und Geschäftsprozessen sind Hierarchiebeziehungen; bei Ontologien bestehen Beziehungen zwischen einzelnen Konzepten (Klasse-/Subklasse) und bei Geschäftsprozessen wird eine Aktivität durch ein ganzes Prozessmodell verfeinert. Zur Berechnung von semantischen Ähnlichkeiten zwischen Geschäftsprozessmodellen konnte deswegen nur das syntaktische Ähnlichkeitsmaß aus der Literatur verwendet werden, da es unabhängig von der Anzahl an Zeichen zu sehr guten Ähnlichkeitswerten führt. Fünf weitere Ähnlichkeitsmaße wurden in dieser Arbeit definiert, welche die Berechnung der semantischen Ähnlichkeit von Geschäftsprozessmodellen ermöglichen.

Die Modellierungsunterstützung kann auf zwei unterschiedliche Weisen realisiert werden: zum einen über eine Anfragesprache (Kapitel 5) oder über eine (semi-) automatische Empfehlungskomponente (Kapitel 6).

Mit Hilfe der Anfragesprache können sich Modellierer zielgerichtet Knoten anzeigen lassen, die sie dann als passende Prozessfragmente in ihr zu bearbeitendes Prozessmodell einfügen können. Die bisher in der Literatur vorgeschlagene OWL-basierte Anfragesprache lässt sich nicht ohne signifikante Erweiterungen an die spezialisierten Anforderungen einer OWL-basierten Anfragesprache für Ähnlichkeitswerte anpassen und erweitern. Beispielsweise werden die zu berechnenden Ähnlichkeitswerte nicht in einer Relation, sondern in einem eigenen Format abgelegt. Damit muss die Anfragesprache neben OWL auch dieses spezielle Format (sie-

he Abbildung 45) verarbeiten können. Die Grammatik der neu konzipierten Anfragesprache wurde in EBNF formuliert; sie ist damit eine kontextfreie Grammatik, die eine kompakte Darstellung erlaubt und sich für automatische Verarbeitungen eignet.

Die Empfehlungskomponente schlägt basierend auf Geschäftsregeln passende Folgeprozesse vor. Geschäftsregeln beschreiben Unternehmensrestriktionen und bilden Vorgaben für Geschäftsprozessmodelle. Ausgehend von Geschäftsregeln ergeben sich unterschiedliche Prozessmodelle, die bei der Realisierung einer automatischen Modellierungsunterstützung beachtet werden müssen. Zur Modellierung von Geschäftsregeln wurde das im Kapitel 3 vorgestellte OWL-basierte Beschreibungsformat um SWRL-Regeln erweitert. Zwei Algorithmen wurden skizziert, mit denen natürlichsprachliche Geschäftsregeln in SWRL-Regeln transformiert werden können. Der besondere Vorteil von SWRL ist in der Möglichkeit zu sehen, bestimmte logische Beziehungen maschinenverständlich darstellen zu können (Unterstützung von Inferenz). Für die Berücksichtigung unterschiedlichen Vokabulars für Geschäftsprozessmodelle wurde ein dritter Algorithmus eingeführt, der neben Geschäftsregeln auch Prozesselemente vorschlägt, die linguistisch gleiche Prozessobjekte beschreiben, aber syntaktisch anders benannt wurden. Des Weiteren wurden in diesem Kapitel Methoden aufgezeigt, bei denen das Abstraktionsniveau beim Vorschlag von passenden Prozessfragmenten berücksichtigt wird. Hierfür wurden drei Verfeinerungsmuster definiert.

Die in dieser Arbeit vorgestellten neuen Methoden wurden in einer Benutzerumfrage evaluiert (Kapitel 7). Insgesamt wurden 55 Personen befragt, die neben der Ähnlichkeit von Geschäftsprozessen auch die vom System automatisch berechneten Ähnlichkeitswerte einschätzen und passende Prozessfragmente auswählen sollten. Die sich aus der Evaluation ergebenden Resultate wurden für eine initiale Angabe der Gewichtung w für das kombinierte Ähnlichkeitsmaß (siehe Kapitel 4.6) verwendet. Als Ergebnis der Auswahl passender Prozessfragmente durch die Benutzer wurde festgestellt, dass sich Modellierer zu drei ähnlichen Gruppen clustern lassen, die bei der Modellierung von Geschäftsprozessen unterschiedlich unterstützt werden sollten. Experten können mehr Prozess als Modellierungsanfängern vorgeschlagen werden, da diese Personengruppe aufgrund ihrer umfangreichen Modellierungserfahrungen sich schneller für passendere Folgeprozesse entscheiden kann. Den beiden anderen Gruppen (Personen mit gar keinen, wenigen und durchschnittlichen Modellierungserfahrungen sowie Personen, mit guten und sehr guten Modellierungserfahrungen) können zwei bis vier Prozessmodelle vorgeschlagen werden, aus denen sie den passenden Folgeprozess auswählen.

Das XML-basierte Austauschformat, das OWL-basierte Beschreibungsformat, die Ähnlichkeitsmaße und die Empfehlungskomponente wurden in einem Software-Prototyp namens SemPeT implementiert, der im Kapitel 8 beschrieben wurde. SemPeT ist frei verfügbar und kann unter der URL `http://aifbserver.aifb.uni-karlsruhe.de/sempet/index.htm` heruntergeladen werden.
An einem Anwendungsszenario wurde gezeigt, dass die in dieser Arbeit vorgestellten Methoden nicht nur für Petri-Netz-basierte Geschäftsprozessmodelle angewendet werden können, sondern auch für Modellierungssprachen wie Ereignisgesteuerte Prozessketten.

10.2 Hauptbeitrag

Die in dieser Arbeit vorgestellten Komponenten eines Modellierungsunterstützungssystems für Geschäftsprozesse stellen den ersten Ansatz zur Konzeption und Realisierung einer solchen Idee vor. Die bisher auf dem Markt erhältlichen Modellierungswerkzeuge für Geschäftsprozesse bieten keine solche Unterstützung an. Der Modellierer muss seine Prozessmodelle selbst erstellen, was in der Regel zu fehlerhaften Prozessmodellen führt. Damit ergab sich die in der vorgelegten Arbeit formulierte und gelöste Aufgabenstellung aus der aktuellen Marktsituation.
Im Rahmen einer Benutzerbefragung wurde bestätigt, dass sich Modellierer mit gar keinen bis durchschnittlichen Modellierungserfahrungen nicht eindeutig für einen passenden Folgeprozess entscheiden können und beim praktischen Einsatz mitunter fehlerhafte bzw. inkonsistente Folgeprozesse ausgewählt haben. Der Einsatz einer Modellierungsunterstützung in den am Markt verfügbaren Prozessmodellierungswerkzeugen könnte helfen, solche Modellierungsfehler weitgehend zu vermeiden.
Bei der Realisierung der Modellierungsunterstützung wurde darauf geachtet, dass unterschiedliche Modellierergruppen zielgerichtet unterstützt werden. Systemimplementierer, die Prozessmodelle zur technischen Realisierung von Informationssystemen weiterverwenden, können die Anfragesprache nutzen, die eine an SQL angelehnte und damit bekannte Syntax verwendet. Anwender in Fachabteilungen mit unterschiedlichen Modellierungserfahrungen können sich über die Empfehlungskomponente unterstützen lassen.
Des Weiteren findet sich in der Literatur auch kein Beitrag, der semantische Ähnlichkeitsmaße für Geschäftsprozessmodelle definiert. In der Literatur wird immer von der Einschränkung eines „kontrollierten" Vokabulars zur Modellierung von Geschäftsprozessen bzw. auch Geschäftsregeln ausgegangen. Die in dieser Arbeit vor-

gestellten Ähnlichkeitsmaße heben damit diese Einschränkung auf, da ein „kontrolliertes" Vokabular in Unternehmen im Allgemeinen nicht umsetzbar ist.
Das Zusammenfügen von Prozessmodellen und die Auswahl an passenden Schnittstellen erfolgten bislang durch manuellen Vergleich der zu integrierenden Prozessmodelle. Eine werkzeugbasierte Unterstützung war bisher nicht verfügbar. Die in dieser Arbeit vorgestellte Anfragesprache bietet erstmals eine solche zielgerichtete Unterstützung an.
Als Sprache zur Modellierung von Geschäftsprozessen wurden Petri-Netze vor allem aufgrund ihrer formalen Semantik ausgewählt. Im letzten Kapitel wurde allerdings gezeigt, dass die in dieser Arbeit vorgestellten Methoden auch auf andere Modellierungssprachen anwendbar und damit nicht nur auf eine Modellierungssprache begrenzt sind. Bei der Auswahl von passenden Folgeprozessen wäre es sogar technisch möglich, methodisch unterschiedlich modellierte Prozessfragmente auswählen zu können. Diese Idee wird im Ausblick skizziert.
Die in dieser Arbeit vorgestellten Methoden können auch zur Unterstützung bei der Datenmodellierung verwendet werden. Anstelle der Einhaltung von Geschäftsregeln bei der Realisierung der Modellierungsunterstützung würde Personen bei der Modellierung von Entity-Relationship Diagrammen bei der Einhaltung von konsistenten konzeptuellen Modellen geholfen werden. Möglich ist auch eine Unterstützung bei der Modellierung von UML-Diagrammen.

10.3 Technische Grenzen des Forschungsansatzes

Die technischen Grenzen der vorliegenden Arbeit ergeben sich aus vorgenommenen Einschränkungen des Softwareprototyps.
So ist die Berechnung von Ähnlichkeitswerten bislang auf zwei Geschäftsprozesse beschränkt. Diese Beschränkung gilt auch für die Formulierung von Anfragen mit SiMQL. Eine wünschenswerte Erweiterung wäre, Ähnlichkeitsberechnungen und Anfragen über mehrere Geschäftsprozessmodelle durchführen und formulieren zu können.
Bislang werden bei der Berechnung der Ähnlichkeit alle Elemente des einen Prozessmodells mit jedem einzelnen Element des zweiten Prozessmodells verglichen, was abhängig von der Prozessgröße einige Minuten dauern kann. Dieses Vorgehen führt bei der Ähnlichkeitsberechnung von großen Geschäftsprozessmodellen (größer als 100 Elemente) zu Performanceproblemen, die durch Verbesserung der verwendeten Algorithmen reduziert werden könnten.

10.4 Methodische Grenzen des Forschungsansatzes

Um Aussagen über die methodischen Grenzen des Forschungsansatzes machen zu können, werden Einschränkungen der verwendeten Sprachen und Methoden diskutiert.

- XML-basiertes Austauschformat:
  XML hat sich als Austausch- und Darstellungsformat von Dokumenten etabliert. Allerdings wird im XML-Standard nicht festgelegt, wie ein XML-Dokument aufgebaut werden soll. Der Aufbau des XML-Dokuments wird vom Benutzer individuell festgelegt. Daraus folgt, dass zur Umsetzung eines XML-basierten Austauschformats auch zusätzlich ein Vokabular verwendet werden muss, das die semantische Bedeutung von Elementen und Attributen festlegt (z.B. mit XML Schema oder RDF).
  In XML Schema ist es erlaubt, ein Element genauso zu benennen wie einen einfachen Typ, nicht jedoch, einen komplexen Typ so zu nennen wie einen einfachen Typ. Somit ist es nicht möglich, zu differenzieren, wann dasselbe Element mehr Kindelelemente hat und wann gar keine.
  Vergleichbar mit konzeptuellen Datenbankentwürfen sind Änderungen in XML (Schemaevolution) mit einem relativ hohen Aufwand verbunden. Dieser hohe Aufwand resultiert auch daraus, dass bislang noch kein graphisches Werkzeug für die konzeptuelle Modellierung von XML-Dokumenten existiert. Damit sind Änderungen bei XML-basierten Austauschformaten nur mit hohem manuellem Aufwand realisierbar.
  Im Kapitel 2 wurde bereits auf die Unzulänglichkeit von XML in Hinsicht auf die Modellierung von Ontologien und die Nutzung von Inferenzmechanismen hingewiesen. Die Unzulänglichkeit ergibt sich aus den mangelnden Vorgaben zur Beschreibung der Bedeutung von Elementen. Das XML-basierte Austauschformat reicht nicht aus, um eine ähnlichkeitsbasierte Modellierungsunterstützung für Geschäftsprozesse methodisch und technisch umzusetzen.
- OWL-basiertes Beschreibungsformat:
  OWL wurde in drei Versionen entworfen, die sich in ihrer Ausdrucksmächtigkeit unterscheiden: OWL Lite, OWL DL (Description Logic) und OWL Full. Für das vorgestellte OWL-basierte Beschreibungsformat von Petri-Netzen wurde OWL DL verwendet.
  Eine Einschränkung von OWL ist deren mangelnde Unterstützung von benutzerdefinierten Datentypen [PaHo05]. Bislang ist es in OWL nicht möglich, benutzerdefinierte Datentypen wie beispielsweise „atLeast18"

zu verwenden. Um eine Aussage wie *„ein Erwachsener ist eine Person, die mindestens 18 Jahre alt ist*" in OWL zu modellieren, könnte die OWL-Syntax wie folgt aussehen:

```
Class(Adult complete Person
restriction(age allValuesFrom(atLeast18))
```

Eine weitere Einschränkung von OWL ist, dass Eigenschaften binäre Beziehungen haben. D.h., jede Eigenschaft wird durch eine Domäne und einen Wertebereich beschrieben (z.B. die Eigenschaft „spricht" kann nicht bei der Klasse „Vortragender" den Wert „rhetorisch" haben aber bei „Gast" „flüssig"). Allerdings führt eine nicht binäre (dreistellige) Beziehung zur Unentscheidbarkeit in OWL-Aussagen und kann bei komplexen Beziehungen zu „verdrehten" Schlussfolgerungen führen.

Als weitere Einschränkung von OWL ist die unzureichende Unterstützung von prozeduralen Funktionen (z.B. arithmetische Funktionen, String-Manipulationen) zu nennen, die für eine Zusammenführung von mehreren „realen" Ontologien notwendig sind.

- Ähnlichkeitsmaße:

  Im Kapitel 4 wurden verschiedene Ähnlichkeitsmaße zur Berechnung der semantischen Ähnlichkeit zwischen Geschäftsprozessmodellen eingeführt. Das linguistische Ähnlichkeitsmaß funktioniert nur, wenn eine Hintergrundontologie vorhanden ist, aus der Kontextinformationen über bestimmte Begriffe extrahiert werden können. Falls für einen Begriff keine Informationen existieren, dann wird es einen leeren Rückgabewert geben. Das beeinträchtigt bzw. minimiert den Wert aus dem linguistischen und strukturellen Ähnlichkeitsmaß und führt zu unzureichenden Ergebnissen bei der Ähnlichkeitsberechnung. Anstatt eines Wörterbuchs als Hintergrundontologie schlagen [FaBR07] die Kombination von n-grams[88] und der Levenshtein-Distanz zur Berechnung der Ähnlichkeit zwischen XML-Schematas vor. Der Ähnlichkeitswert ergibt sich dann als Durchschnitt der beiden Ähnlichkeitsmetriken. Die Verwendung der Methode nach [FaBR07] anstelle des linguistischen Ähnlichkeitsmaßes müsste noch evaluiert werden.

[88] Mit n-grams sollen häufige Wortgruppen bestimmt werden. n steht für eine beliebige Zahl. Im Falle von 2-grams wird bei den Begriffen *Customer* (Cu st om er) und *Client* (Cl ie nt) nach gemeinsamen 2-grams gesucht. Eine Beispielimplementierung findet sich unter http://scorpius.informatik.uni-mannheim.de:8080/ir/ngrams.html

- Deskriptive Anfragesprache:
  Im Kapitel 5 wurde eine deskriptive Anfragesprache für Ähnlichkeitswerte (SiMQL) eingeführt. Im SQL-Standard sind nur wenige selbstdefinierte Datentypen möglich und so erlaubt die Anfrageformulierung in SiMQL bislang keine selbstdefinierten Datentypen, Operationen und Funktionen, die die Lesbarkeit von Anfragen und die Wiederverwendung verbessern könnten.
  Eine deskriptive Anfragesprache lässt die Realisierung der Anfrageverarbeitung durch das System vollkommen offen. So hat der Benutzer bei der Formulierung seiner Anfrage mit SiMQL keinen Einfluss auf die Auswertungsreihenfolge von Funktionen und Operationen und damit auf eine effiziente Auswertung. Eine nahe liegende Erweiterung von SiMQL wären ähnlich wie bei SQL (PL/SQL) Ablaufkonstrukte (Sequenzen, bedingte Ausführung und Schleifen), die man aus imperativen Sprachen kennt.
- SWRL-basierte Erweiterung von OWL:
  Im Kapitel 6 wurde das OWL-basierte Beschreibungsformat um SWRL-Prädikate erweitert, die eine Abbildung von Regeln in einem maschinenverständlichen und –interpretierbaren Format erlauben. Die SWRL-Spezifikation hat allerdings noch den Status einer *Member Submission* beim W3C und muss noch einigen Veränderungen unterzogen werden. So ist bislang in der Spezifikation keine Verneinung (NOT-Operator) realisiert.
  Generell werden selbstdefinierte Funktionen/Operationen (realisiert als built-ins) in der SWRL-Spezifikation durch Axiome und nicht durch Funktionen definiert. Das gilt auch für den selbstdefinierten built-in `swrl:semSimilarity`.

10.5 Ausblick

Die wachsende Menge an Daten und Informationen im Unternehmen führt auch zu großen Prozessmodellen, die kompakt dargestellt werden sollten. Eine Modellierungsunterstützung bietet den Vorteil, die Modellierung von Geschäftsprozessen zu erleichtern und die Qualität der Prozessmodelle durch Wiederverwendung von bereits modellierten Prozessaktivitäten zu verbessern. Eine solche Modellierungsunterstützung für Geschäftsprozesse kann als Ausgangspunkt für weiterführende Konzepte genutzt werden.

Technische Weiterentwicklung

Die Visualisierungskomponente der Folgeprozesse sollte benutzerfreundlich realisiert werden. Ein vorgeschlagener Folgeprozess kann aus sechs bis zehn Elementen bestehen und die maximale Anzahl an möglichen Folgeprozessen sollte vier sein. Hierbei ist zu überlegen, wie die Visualisierung am besten benutzergerecht implementiert werden kann.

Inhaltliche Weiterentwicklung

Die Modellierungsunterstützung wurde bisher nur für Geschäftsprozesse realisiert, die in Englisch beschrieben wurden. Wünschenswert für Anwender anderer Sprachen wäre die Nutzung einer solchen Funktion auch für weitere Sprachen (z.B. Deutsch). Denkbar wäre auch die Kombination von Prozessfragmenten in einem Prozessrepository in Deutsch und Englisch, wobei die Prozessfragmente vor deren Visualisierung für den Benutzer in die entsprechende Sprache umbenannt werden würden.

Des Weiteren können Prozessmodelle auch in unterschiedlichen Modellierungssprachen erstellt worden sein. Eine Kombination wäre auch hier möglich. Der Benutzer kann aus Prozessmodellen auswählen, die mit BPEL oder als EPK modelliert wurden, und kann dann die entsprechenden Folgeprozesse in das aktuell editierte Prozessmodell einfügen. Oder die Folgeprozesse werden vor der Visualisierung für den Benutzer in die aktuell verwendete Modellierungssprache transformiert.

Literaturverzeichnis

[AABC05] Arkin, A.; Askary, S.; Bloch, B.; Curbera, F.; Goland, Y.; Kartha, N.; Liu, C.K.; Thatte, S.; Yendluri, P.; Yiu, A.: Web services business process execution language version 2.0. wsbpel-specification-draft-01, OASIS, 2005.

[AaDK02] van der Aalst, W.; Desel, J.; Kindler, E.: On the semantics of EPCs: A vicious circle. In: M. Nüttgens, F. Rump (eds.): EPK 2002 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Trier, Germany, S. 71-79, November 2002.

[AaDO00] van der Aalst, W.; Desel, J.; Oberweis, A.: Business Process Management. Models, Techniques, and Empirical Studies, Lecture Notes in Computer Science, Vol. 1806, Springer-Verlag, Berlin Heidelberg New York, 2000.

[AaHV97] van der Aalst, W.; Hauschildt, D.; Verbeek, H.M.W.: A Petri-net-based Tool to Analyze Workflows. In B. Farwer, D. Moldt, and M. O. Stehr, editors, Proceedings of Petri Nets in System Engineering (PNSE'97), pages 78–90, Hamburg, Germany, September 1997. University of Hamburg (FBI-HH-B-205/97).

[Aals98] van der Aalst, W.: The Application of Petri Net to Workflow Management. The Journal of Circuist, Systems and Computers, Department of Mathematics and Computing Science, Eindhoven University of Technology, http://is.tm.tue.nl/staff/wvdaalst/publications/p53.pdf

[Aals00] van der Aalst, W.: Loosely coupled interorganizational workflows: modeling and analyzing workflows crossing organizational boundaries, Information Management, 37(2), Elsevier, 2000.

[AaWe01] van der Aalst, W.; Weske, M.: The P2P approach to interorganizational workflows, Proceedings of the 13th International Conference on Advanced Information Systems Engineering, S. 140–156. Lecture Notes in Computer Science, Klaus R. Dittrich, Springer-Verlag, 2001.

[Bach94] Bacher, J.: Clusteranalyse: Anwendungsorientierte Einführung. R. Oldenbourg Verlag Wien, München GmbH, 1994.

[BaKK95] Bause, F.; Kemper, P.; Kritzinger, P.: Abstract Petri Net Notation, Petri Net Newsletter, Nr. 49, 1995.

[Baum96] Baumgarten, B.: Petri-Netze, Grundlagen und Anwendungen, Spektrum Akademischer Verlag GmbH, 2. Auflage, 1996.

[BaWN82] Baird, J.; Wagner, M.; Noma, E.: Impossible cognitives spaces, Geographical Analysis 14, 1982.

[BCHW03] Billington, J.; Christensen, S.; van Hee, K.; Weber, M: The Petri Net Markup Language: Concepts, Technology, and Tools, Proceedings of the 24th International Conference on Applications and Theory of Petri Nets (ICATPN 2003), Eindhoven, The Netherlands, S. 483-505 Springer-Verlag, June, 2003.

[BDFL05] de Bruijn, J.; Domingue, J.; Fensel, D.; Lausen, H.; Polleres, A.: WSMO Primer. WSMO Final Draft, April 2005. http://www.wsmo.org/TR/d3/d3.1/v0.1

[Beck04] Beckett, D.: RDF/XML Syntax Specification (Revised). W3C Recommendation, World Wide Web Consortium, 2004, http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/

[BEKO06] Brockmans, S.; Ehrig, M.; Koschmider, A.; Oberweis, A.; Studer, R.: Semantic Alignment of Business Processes, Proceedings of the Eighth International Conference on Enterprise Information Systems (ICEIS 2006), S. 191-196, INSTICC Press, Paphos, Cyprus, May, 2006.

[BEPW93] Backhaus, K., Erichson, B.; Plinke, W.: Multivariante Analysemethoden. Eine anwendungsorientierte Einführung, Springer-Verlag, 11. Auflage, 2005.

[Berg02] Bergmann, R.: Experience Management: Foundations, Development Methodology, and Internet-Based Applications, Springer-Verlag, 2002.

[BKKO05] Betz, S.; Klink, S.; Koschmider, A.; Oberweis, A.: Automatic User Support for Business Process Modeling, Proceeding of the Workshop on Semantics for Business Process Management, S. 1-12, Budva, Montenegro, June, 2006.

[Blou01] Blough, D. S.: Avian Visual Cognition. The Perception of Similarity. Department of Psychology, R. G. Cook (Hrsg.), Brown University, http://www.pigeon.psy.tufts.edu/avc/dblough/ (2001).

[BoNo05] Bontas, E. P.; Mochol, M.: Towards a Cost Estimation Model for Ontology Engineering, Technical Report TR-B-05-03, Freie Universität Berlin, Germany, April 2005.

[BoTW01] Boley, H.; Tabet, S., Wagner, G.: Design Rationale for RuleML: A Markup Language for Semantic Web Rules. Proceedings of Semantic Web Workshop SWWS, Stanford, S. 381–401, 2001.

[Brau97] Braun, H.: Neuronale Netze. Optimierung durch Lernen und Evolution. Springer-Verlag, 1997.

[BRG] Business Rule Group, http://www.businessrulesgroup.org

[BrGu04] Brickley, D.; Guha, R.: RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, World Wide Web Consortium (W3C), 2004, Februar, http://www.w3.org/TR/2004/REC-rdf-schema-20040210/

[Broc02] Die Brockhaus Enzyklopädie. 2002.

[BrRR87] Brauer, W.; Reisig, W.; Rozenberg, G.: Petri Nets: Applications and Relationship to other Models of Concurrency, Advances in Petri Nets 1986, Springer-Verlag, 1987.

[Call03] Callan, R.: Neuronale Netze im Klartext, Pearson Studium, 2003.

[CaMM83] Carbonell, J. G., Michalski, R. S., Mitchell, T. M.: An overview of machine learning. In R. S. Michalski, J. G.Carbonell, T. M. Mitchell (Hrsg.), Machine Learning: An Artifical Intelligence Approach. Tioga Publishing Company, Palo Alto, 1983.

[Carm92] Carmo, M. P.: Riemannian Geometry, Birkhauser, 1. Auflage, 1992.

[Chen76] Chen, P. P.: The Entity-Relationship Model: Toward a Unified View of Data, ACM Transaction on Database Systems, 1(1), 1976.

[ChJB98] Chandrasekaran, B.; Josephson, J.; Benjamins, R.: The Ontology of Tasks and Methods. http://www.cse.ohio-state.edu/~chandra/Ontology-of-Tasks-Methods.PDF, 1998.

[Chom59] Chomsky, N.: On certain formal properties of grammars. In: Information and Control 1, S. 91–112, 1959.

[CIMP03] Carlisle, D.; Ion, P.; Miner, R.; Poppelier, N.: Mathematical Markup Language (MathML) Version 2.0 (Second Edition); W3C Recommendation 21 October 2003; http://www.w3.org/TR/2003/REC-MathML2-20031021/

[ClMe87] Clocksin, W.; Mellish, C.: Programming in Prolog. 3. Auflage, Springer-Verlag, 1987.

[ChGR94] Cherkassky, B.; Goldberg, A.; Radzik , T.: Shortest Paths Algorithms: Theory and Experimental Evaluation. In: Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms, S. 516-525, 1994.

[CoTh01] Cover, T. M.; Thomas, J. A.: Elements of Information Theory. John Wiley & sons, New York, 1991.

[Dave93] Davenport, T. H.: Process Innovation. Reengineering Work through Information Technology. Boston, Harvard Business School, 1993.

[DeOb96] Desel, J.; Oberweis, A.: Petri-Netze in der Angewandten Informatik: Einführung, Grundlagen und Perspektiven, Zeitschrift WIRTSCHAFTS-INFORMATIK, 38(4), 1996.

[DoJK05] Dostal, W.; Jeckle, M.; Kriechbaum, W.: Semantic und Web Services: Beschreibung von Semantik, Java Spektrum, April/Mai, 2/2004, S. 45-49.

[ebXML] ebXML Business Process Specification Schema v1.0, ebXML Context/Metamodel Group of the CC/BP Joint Delivery Team, 27. April 2001, http://www.ebxml.org/specdrafts/specificationschemav1.00.pdf

[EhKO07] Ehrig, M.; Koschmider, A.; Oberweis, A.: Measuring Similarity between Semantic Business Process Models. Proceedings of the Fourth Asia-Pacific Conference on Conceptual Modelling (APCCM 2007), vol-

ume 67, S. 71-80. Australian Computer Science Communications, Ballarat, Australia, 2007.

[FaBR07] Fabien, D.; Bellahsene, Z.; Roche, M.: A Context-based Measure for Discovering Approximate Semantic Matching between Schema Elements. Proceedings of the First International Conference on Research Challenges in Information Science, Ouarzazate, Morocco, April 23-26, 2007, S. 9-20.

[Fahr95] Fahrwinkel, U.: Methoden zur Analyse von Geschäftsprozessen zur Unterstützung des Business Process Reengineering, HNI – Verlagsschriftenreihe, Prof. Dr. –Ing. Jürgen Gausemeier (Hrsg.), Heinz Nixdorf Institut, Universität-GH Paderborn, 1995.

[FaKP04] Fahrmeir, L.; Künstler, R.; Pigeot, I.: Statistik, Springer-Verlag, 2004.

[Fehl92] Fehling, R.: Hierarchische Petrinetze, TU München, Verlag Dr. Kovac. 1992.

[Fell98] Fellbaum, C.: WordNet: An electronic lexical database, MIT Press, 1998.

[Frie01] Friedman-Hill, E.: Jess: the Java Expert System Shell, sandia national laboratories. http://herzberg.ca.sandia.gov/jess/ (2001)

[FrPP97] Freedman, D.; Pisani, R.; Purves, R.: Statistics. Third Edition. W. W. Norton & Company, 1997.

[GaDi93] Gatziu, S.; Dittrich, K. R.: Events in an active object-oriented database system. In: Rules in Database Systems. S. 23–39, 1993.

[GaJD05] Gasevic, D.; Jovanovic, J.; Devedzic, V.: Petrinet Infrastructure for the Semantic Web, Proceedings of the Symposium on Professional Practice in AI, S. 225-235, Toulouse, France, August, 2004.

[GeLa81] Genrich, H.J.; Lautenbach, K.: System Modelling with High-Level Petri Nets, Theoretical Computer Science, ScienceDirect 13, 1981.

[Genr86] Genrich, H. J.: Predicate/Transition Nets, in: [BeRR87].

[GKMS94] Herbst, H.; Knolmayer, G.; Myrach, T.; Schesinger, M.: The Specification of Business Rules: A Comparison of selected methodologies, Methods and Associated Tools for the Information System Life Cycle, S. 29 -46, Elsevier, University of Limburg, Maastricht, 1994.

[Golb04] Golbreich, C.: Combining Rule and Ontology Reasoners for the Semantic Web, in Rules and Rule Markup Languages for the Semantic Web: Third International Workshop, Grigoris Antoniou, Harold Boley (Hrsg.), Springer-Verlag, S. 6-22, 2004.

[Grub93a] Gruber, T. R.: A Translation Approach to Portable Ontology Specifications, Knowledge Acquisition, 5(2), 1993.

[Grub93b] Gruber, T. R.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing, Formal Ontologies in Conceptual Analysis and

Knowledge Representation, N. Guarino und R. Poli (Hrsg.), Kluwer Academic Publishers, 1993.

[Guar98] Guarino, N.: Formal Ontology in Information Systems: Proceedings of the 1st International Conference, Ross, M.; Brebbia, C. A.; Staples. G., Stapleton, J. (Hrsg.), Trento, Italy, Amsterdam, The Netherlands, IOS Press, 1998.

[Jens92] Jensen, K.: *Coloured Petri Nets, Volume 1: Basic Concepts*, EATCS Monographs on Theoretical Computer Schience, Springer-Verlag, Berlin – Heidelberg, 1992.

[HaCh93] Hammer, M.; Champy, J.: Business Reengineering. Die Radikalkur für das Unternehmen, Campus-Verlag, Frankfurt, New York; 5. Auflage, 1993.

[Hall02] von Halle, B.: Business Rules Applied. Wiley Computer Publishing, 2000.

[HaKa01] Han, J.; Kamber, M.: Data Mining. Concepts and Techniques. Academic Press. San Diego. 2001.

[HaST99] Horrocks, I.; Sattler, U.; Tobies, S.: Practical reasoning for expressive description logics. In Proceeding of the 6th International Conference on Logic for Programming and Automated Reasoning, S. 161–180. 1999.

[Heit97] Heit, E.: Features of similarity and category-based induction, Proceedings of the Interdisciplinary Workshop on Categorization and Similarity, University of Edinburgh, 1997.

[HeSc91] Heuer, A.; Scholl, M. H.: Principles of Object-Oriented Query Languages. In: Appelrath, H.-J. (Hrsg.): Datenbanksysteme in Büro,Technik und Wissenschaft (BTW'91), Bd. 270, Springer-Verlag, S.178–197, 1991.

[HeWe90] Heinemann, B.; Weihrauch, K.: Logik für Informatiker, B.G. Teubner-Verlag, 1990.

[Hjel01] Hjelm, J.: Creating the Semantic Web with RDF, Wiley Computer Publishing, 2001.

[HKKR05] Hitz, M.; Kappel, G.; Kapsammer, E.; Retschitzegger, W.: UML@Work, dpunkt Verlag, 2005.

[HoKM06] Hornung, T.; Koschmider, A.; Mendling, J.: Integration of Heterogeneous BPM Schemas: The Case of XPDL and BPEL, CAiSE Forum 2006. Proceedings at the 18th Conference on Advanced Information Systems Engineering, S. 23-26, Luxembourg, June 2006.

[HoKO07] Hornung, T.; Koschmider, A.; Oberweis, A.: Rule-based Autocompletion Of Business Process Models. In CAiSE Forum 2007. Proceedings at the 19th Conference on Advanced Information Systems Engineering (CAiSE). Trondheim, Norway, June 2007.

[Horr05] Horrocks, I.: Applications of description logics: State of the art and research challenges. Proceeding of the 13th International Conference on Conceptual Structures, S. 87–90, Springer-Verlag, 2005.

[HPBT04] Horrocks, I.; Patel-Schneider, P.; Boley, H.; Tabet, S.; Grosof, B.; Dean, M.: SWRL: A semantic web rule language combining OWL and RuleML. http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/ (2004)

[HuMS04] Hustadt, U.; Motik, B.; Sattler, U.: Reducing SHIQ– Description Logic to Disjunctive Datalog Programs, Proceedings of the 9th International Conference on Knowledge Representation and Reasoning, S.152–162, Dubois, Didier; Welty, Christopher A.; Williams, Mary-Anne; Whistler, Canada 2004.

[HuRy04] Huth, M.; Ryan, M.: Logic in Computer Science. Modelling and reason about systems. 2. Auflage, Cambridge University Press, June, 2004.

[KeEi06] Kemper, A.; Eicker, A.: Datenbanksysteme. Eine Einführung. 6. Auflage, Oldenbourg, 2006.

[KeNS92] Keller, G.; Nüttgens, M.; Scheer, A.-W.: Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter Prozeßketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik. Heft 89, Scheer, A.-W. (Hrsg.) Saarbrücken, 1992.

[KiLW95] Kifer, M.; Lausen, G.; Wu, J.: Logical foundations of object-oriented and frame-based languages, Journal of the ACM, 42(4), 1995.

[Kind03] Kindler, E.: Using the Petri Net Markup Language for Exchanging Business Processes, Software Engineering Group, Computer Science Department, University of Paderborn, 2003.

[Klei01] Klein, M.: Combining and relating ontologies: an analysis of problems and solutions, Proceedings of Workshop on Ontologies and Information Sharing, Gómez-Pérez, A.; Gr M. (Hrsg.), Seattle, USA, 2001.

[KMLP05] Kloppmann, M., Koenig, D., Leymann, F., Pfau, G., Rickayzen, A., von Riegen, C., Schmidt, P., Trickovic, I.: WS-BPEL Extension for Subprocesses BPEL-SPE. Joint white paper, IBM and SAP, https://www.sdn.sap.com/irj/servlet/prt/portal/prtroot/docs/library/uuid/5cbf3ac6-0601-0010-25ae-ccb3dba1ef47 (2005).

[KnEn04] Knolmayer, G.; Endl, R.: Regelbasierte Entwicklung betrieblicher Informationssysteme. Gestaltung flexibler Informationssysteme durch explizite Modellierung der Geschäftslogik, EUL-Verlag, 2004.

[Knut64] Knuth, D. E.: Backus Normal Form versus Backus Naur Form, Communications of the ACM 7, December, 1964.

[KoBl07] Koschmider, A.; Blanchard, E.: User Assistance for Business Process Model Decomposition. In First IEEE International Conference on Research Challenges in Information Science, S. 445-454. April 23-26, 2007 Ouarzazate, Marokko, April 2007.

[KoMe05] Koschmider, A.; Mevius, M.: A Petri Net Based Approach For Process Model Driven Deduction Of BPEL Code, International Workshop on Modeling Inter-Organizational Systems, R. Meersman and Z. Tari and P. Herrero et al. (editor), Springer-Verlag, S. 495-505, Cyprus, Oktober 2005.

[KoOb05] Koschmider, A.; Oberweis, A.: Ontology based Business Process Description Proceedings of the CAiSE´05 WORKSHOPS, no. 2, S. 321-333. J. Castro; E. Teniente (Hrsg.), Springer-Verlag, Porto, Portugal, June, 2005.

[KoOb07a] Koschmider, A.; Oberweis, A.: Modeling Semantic Business Process Models. In: P. Rittgen (Hrsg.), Handbook of Ontologies for Business Interaction. IDEA Group Publishing, 2007.

[KoOb07b] Koschmider, A.; Oberweis, A.: How To Detect Semantic Business Process Model Variants?, Proceedings of the 2007 ACM Symposium on Applied Computing, volume 2, Association for Computing Machinery, Seoul, Korea, March, 2007.

[KoRi05] Koschmider, A.; Ried, D.: Semantische Annotation von Petri-Netzen, Workshop für Algorithmen und Werkzeuge für Petrinetze (AWPN'05), S. 66-71. Schmidt, K.; Stahl, Chr., Informatik-Berichte, Humboldt-Universität zu Berlin, September, 2005.

[KoSO04] Koschmider, A.; Sommer, D.; Oberweis, A.: RDF-basierte Integration von E-Learning-Metadaten in einem Informationsportal, Informationssysteme im E-Business und E-Government (EMISA 2004), F. Feltz and A. Oberweis and B. Otjacques (Hrsg.), LNI, Volume 56, S. 48-59, Köllen-Verlag, 2004.

[Lacy05] Lacy, L. W.: OWL: Representing Information Using The Ontology Web Language, Trafford Publishing, Victoria, Canada, 2005.

[Laus88] Lausen, G.: Modeling and Analysis of the Behavior of Information Systems, IEEE Transactions on Software Engineering, vol. 14 (11), S. 1610-1620, 1988.

[Lehm23] Lehmen, A.: Lehrbuch der Philosophie auf aristotelisch-scholastischer Grundlage. Band I: Logik, Kritik, Ontologie, Herder, 6. Auflage, 1923.

[Lenz03] Lenz, K.; Oberweis, A.: Inter-organizational Business Process Management with XML Nets, in: Ehrig, H.; Reisig, W.; Rozenberg, G.; Weber, H. (Hrsg.): Petri Net Technology for Communication-Based Systems, Advances in Petri Nets, Springer-Verlag, S. 243-263, 2003

[Leve66] Levenshtein, V. I.: Binary Codes capable of correcting deletions, insertions, and reversals, Soviet Physics – Doklady 10(3), 1966.

[LiBM03] Li, Y.; Bandar, Z.; McLean, D.: An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources. In: IEEE Transaction on Knowledge and Data Engineering, 15(4), S. 871-882, 2003.

[LiKl02] von der Lippe, P.; Klodroba A.: Repräsentativität von Stichproben, http://www.uni-pots dam.de/u/Geooekologie/institut/hydrologie/download/kneis/uebung04_einfuehrungstext_20jun05.pdf

[Lin98] Lin, D.: An Information-Theoretic Definition of Similarity, ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning, 296-304, Jude W. Shavlik, Morgan Kaufmann Publishers Inc., San Francisco, USA, 1998.

[Mall22] Mally, E.: Studien zur Theorie der Möglichkeit und Ähnlichkeit : allgemeine Theorie der Verwandschaft gegenständlicher Bestimmungen, Wien, Hölder, 1922.

[MaMo03] Maedche, A.; Motik, B.: Repräsentations- und Anfragesprachen für Ontologien - eine Übersicht. Datenbank-Spektrum, 6, 2003.

[MaSt01] Maedche, A.; Staab, S.: Measuring similarity between ontologies, in Proceedings of the European Conference on Knowledge Acquisition and Management', Lecture Notes in Computer Science, Springer-Verlag, 2001.

[MBCD95] Marsan, M. A.; Balbo, G.; Conte, G.; Donatelli, S.; Franceschinis, G.: Modelling with Generalized Stochastic Petri-Nets. Wiley series in parallel computing, John Wiley & Sons, New York, 1995.

[MBLP03] Martin, D.; Burstein, M.; Lassila, O.; Paolucci, M.; Payne, T.; McIlraith, S.: Describing Web Services using OWL-S and WSDL; http://www.daml.org/services/owl-s/1.1/owl-s-wsdl.html

[McHa03] McGuinness, D. L.; van Harmelen, F.: OWL Web Ontology Language Overview, August, 2003, W3C Candidate Recommendation, 18 August 2003, World Wide Web Consortium (W3C), http://www.w3.org/TR/owl-features/

[MiCh91] Miller, G.; Charles, W. G.: Contextual correlates of semantic similarity. Language and Cognitive Processes, 6(1), S. 1-28, 1991.

[MiKo90] Michalski, R. S.; Kodratoff, Y.: Research in machine learning; recent progress, classification of methods, and future directions. In Y. Kodratoff; R.S. Michalski (Hrsg.): Machine Learning: An Artificial Intelligence Approach, volume III. Morgan Kaufmann, San Mateo, 1990.

[Mins75] Minsky, M.: A Framework for Representing Knowledge, in The Psychology of Computer Vision, Winston, P. (Ed.), McGraw-Hill, 1975.

[Mint03] Minter, S.: XHTML, CSS & Co - Die W3C-Spezifikationen für das Web-Publishing. Addison-Wesley, 2003.

[MiWK00] Mitra, P.; Wiederhold, G.; Kersten, M.: A Graph-Oriented Model for Articulation of Ontology Interdependencies, 7th International Conference on Extending Database Technology, Carlo Zaniolo, Peter C. Lockemann, Marc H. Scholl, Torsten Grust, Springer-Verlag, Konstanz, Germany, 2000.

[NoMc01] Noy, N. F. und McGuiness, D. L.: Ontology Development 101: A Guide to Creating Your First Ontology. Technischer Report KSL-01-05, Stanfort Knowledge Systems Laboratory, März, 2001.

[NoMu99] Noy, N. T.; Musen, M. A.: An Algorithm for Merging and Aligning Ontologies: Automation and Tool Support, Proceedings of the Workshop on Ontology Management, AAAI Press, Orlando, USA, 1999.

[Ober96] Oberweis, A.: Modellierung und Ausführung von Workflows mit Petri-Netzen, B. G. Teubner Verlagsgesellschaft, Stuttgart, 1996.

[OWLS04] The OWL Services Coalition: OWL-S: semantic Markup for Web-Services. http://www.daml.org/services, 2004.

[PaHo05] Pan, J.; Horrocks, I.: OWL-Eu: Adding Customised Datatypes into OWL, Journal of Web Semantics, 4(1), 2005.

[Pete77] Peterson, J. L.: Petri Nets, ACM Computer Survey, 9(3), 1977, ACM Press.

[Petr62] Petri, C. A.: Kommunikation mit Automaten, Schriften des Instituts für Instrumentelle Mathematik, Nr. 2, Bonn, 1962.

[PiMa01] Pinto, H. S.; Martins, J. P.: A methodology for ontology integration. Proceedings of the 1st. International conference on Knowledge capture, S. 131- 138, ACM Press, 2001.

[Port97] Porter, M. F.: An algorithm for suffix stripping, Readings in Information Retrieval, Elsevier, S. 313-317, 1997.

[Posp06] Pospeschill, M.: Statistische Methoden. Strukturen, Grundlagen, Anwendungen in Psychologie und Sozialwissenschaften. Elsevier, 2006.

[Reis85[Reisig, W.: Systementwurf mit Netzen. Springer-Verlag, 1985.

[Reis86] Reisig, W.: Petrinetze. Eine Einführung. Springer-Verlag, 1986.

[ReRo98a] Reisig, W.; Rozenberg, G.: Lectures on Petri Nets I: Basic Models. Springer-Verlag, 1998.

[ReRo98b] Reisig, W.; Rozenberg, G.: Lectures on Petri Nets II: Applications. Springer-Verlag, 1998.

[Resn99] Resnik, P.: Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language, Journal of Artificial Intelligence Research, 11, 1999.

[Rich92] Richter, M. M.: Classification and learning of similarity measures, Proceedings der Jahrestagung der Gesellschaft für Klassifikation. Opitz, Lausen, Klar (Hrsg.), Studies in Classification, Data Analysis and Knowledge Organisation, Springer-Verlag, 1992.

[RiMc05] Rizopoulos, N.; McBrien, P.: A general approach to the generation of conceptual model transformations, in 17th International Conference

(CAiSE 2005) Porto, Portugal, Oscar Pastor, João Falcão e Cunha (Hrsg.), Springer-Verlag, S. 326-341, June, 2005.

[RiSt04] von Hagen, C.; Stucky, W.: Business-Process- und Workflow-Management: Prozessverbesserung durch Prozess-Management. B. G. Teubner Verlag, 2004.

[RLKB06] Roman, D.; Lausen, H.; Keller, U.; de Bruijn, J.; Bussler, C.; Domingue, J.; Fensel, D.; Hepp, M.; Kifer, M.; König-Ries, B.; Kopecky, J.; Lara, R.; Oren, E.; Polleres, A.; Scicluna J.; Stollberg, M.: D2v1.2 Web Service Modeling Ontology http://www.wsmo.org/TR/d2/v1.2/, 2006.

[RMBB98] Rada, R.; Mili, H.; Bicknell, E.; Blettner, M.: Development and Application of a metric on Semantic Nets, IEEE Transactions on Systems, Man and Cybernetics 19(1), 1989.

[RoSc05] Roman, D.; Scicluna, J.: D15v0.1. Orchestration in WSMO, WSMO Working Draft 24 January 2005, http://www.wsmo.org/2004/d15/v0.1/20040529/

[Rose58] Rosenblatt, F.: The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. Psychological Review 65, S. 386-408, 1958.

[RuGo65] Rubenstein, H.; Goodenough, J. B.: Contextual correlated of synonyms. Communication of ACM, 8(10), 1965.

[RuHW86] Rumelhart, D. E.; Hinton, G. E.; Williams, R. G.: Learning representations by back-propagating errors, Nature 323, London, S. 533-536.

[SaOr00] Sadiq, W.; Orlowska, M. E.: Analyzing Process Models Using Graph Reduction Techniques. Information Systems, 25(2), 2000.

[Sche97] Scherer, A.: Neuronale Netze. Grundlagen und Anwendungen. Vieweg, 1997.

[Sche01] Scheer, A.-W.: Vom Geschäftsprozess zum Anwendungssystem. Springer-Verlag, 1998.

[ScSa05] Schmitt, I.; Saake, G.: A comprehensive database schema integration method based on the theory of formal concepts. Acta Informatica 41, 2005.

[Seeb02] Seeboerger-Weichselbaum, M.: Java/XML-Das bhv Taschenbuch, moderne industrie Verlag, Bonn, 2002.

[SEHH03] Stumme, G.; Ehrig, M.; Handschuh, S.; Hotho, A.; Maedche, A.; Motik, B.; Oberle, D.; Schmitz, C.; Staab, S.; Stojanovic, L.; Stojanovic, N.; Studer, R.; Sure, Y.; Volz, R. Zacharias, V.: The Karlsruhe view on ontologies. Technical report, University of Karlsruhe, Institute AIFB, Karlsruhe, Germany, 2003.

[Sova91] Sowa, J. F.: Principles of semantic networks. Explorations in the representation of knowledge, Morgan Kaufmann, San Mateo, California, 1991.

[StHa02] Stahlknecht, P.; Hasenkamp, U.: Einführung in die Wirtschaftsinformatik, 10. Auflage, Springer-Verlag, 2000.

[Stie96] Stier, W.: Empirische Forschungsmethoden. Springer-Verlag, 1996.

[StLa77] Steinhausen, D., Langer, K.: Clusteranalyse. Einführung in Methoden und Verfahren der automatischen Klassifikation. Walter de Gruyter & Co., 1997.

[StSt04] Staab, S.; Studer, R.: Handbook on Ontologies, International Handbooks on Information Systems, Springer-Verlag, 2004.

[Tver77] Tversky, A.: Features of similarity, Psychological Review, 84(4), 1977.

[UDDI01] UDDI Consortium. UDDI Executive White Paper, Nov. 2001 http://uddi.org/pubs/UDDI_Executive_White_Paper.pdf

[Ullm88] Ullman, J. D.: Principles of Database and Knowledge-Base Systems. Bd. I, Kapitel 3. Computer Science Press, 1988.

[VeVo75] Veenker, G.; Vorwerk, E.: Zur Selektion von ähnlichen Worten aus umfangreichen Datenbeständen, Bericht des Instituts für Informatik, Universität Bonn, Nr. 4, April, 1975.

[ViLB06] Vidal, J. C.; Lama, M.; Bugarin, A.: A High-level Petri Net Ontology Compatible with PNML. Petri Net Markup Language Forum 2006 (PNML 06), Turku, Finland, June, 2006.

[Vorw77] Vorwerk, E.: Untersuchungen über die Ähnlichkeit von Zeichenketten in großen Datenbeständen, Gesellschaft für Mathematik und Datenverarbeitung MBH Bonn, Oldenbourg Verlag, 1. Auflage, 1977.

[Webe02] Weber, M.: Allgemeine Konzepte zur software-technischen Unterstützung verschiedener Petrinetz-Typen. Dissertation, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, December 2002.

[WeKi03] Weber, M.; Kindler, E.: The Petri Net Kernel, In H. Ehrig, W. Reisig, G. Rozenberg, H. Weber, Petri Net Technology for Communication-Based Systems, Advances in Petri Nets, S. 109-124. Springer-Verlag, 2003.

[Whit04] White, S. A.: Business Process Modeling Notation. Specification, BPMI.org, http://www.bpmn.org/Documents/BPMN%20V1-0%20May%203%202004.pdf (2004).

[WiFr05] Witten, I. H.; Frank, E.: Data Mining, Practical Machine Learning Tools and Techniques, Second Edition, Elsevier, 2005.

[W3C00] World Wide Web Consortium: Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation, 6. Oktober 2000, http://www.w3.org/TR/REC-xml

[W3C02] World Wide Web Consortium: Web Service Architecture Requirements, Oct. 2002. http://www.w3c.org/TR/wsa-reqs

[W3C04a] World Wide Web Consortium: Resource Description Framework (RDF): Concepts and Abstract Syntax, Februar 2004, Recommendation, http://www.w3.org/TR/rdf-concepts/

[W3C04b] World Wide Web Consortium: RDF Primer, Februar 2004, Recommendation, http://www.w3.org/TR/rdf-primer/

[W3C04c] W3C. RDF/XML Syntax Specification (Revised), Februar 2004, Recommendation, http://www.w3.org/TR/rdf-syntax-grammar/

[W3C04d] W3C. RDF Semantics, Februar 2004, Recommendation, http://www.w3.org/TR/rdf-mt/

[W3C06a] World Wide Web Consortium: SPARQL Query Language for RDF, Working Draft, October 2006, http://www.w3.org/TR/rdf-sparql-query/

[WMC02] Workflow Management Coalition: Workflow Process Definition Interface – XML Process Definition Language. Document Number WFMC-TC-1025, October 25, 2002, Version 1.0, Workflow Management Coalition.

[WMC05] Workflow Management Coalition: Workflow Process Definition Interface – XML Process Definition Language. Document Number WFMC-TC-1025, October 3, 2005, Version 2.00, Workflow Management Coalition.

[WuPa94] Wu, Z.; Palmer, M.: Verb semantics and lexical selection. 32nd Meeting of the Association for Computational Linguistics, 1994, S. 133-138.

Anhang

Anhang A OWL-Serialisierung erweitert um SWRL-Teil

```
?xml version="1.0"?>
<rdf:RDF
   xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:petri="http://www.aifb.uni-
           karlsruhe.de/petri/PNOntology_neu.owl#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns="http://www.aifb.uni-karlsruhe.de/petri/PNOntology.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns:p1="http://www.owl-ontologies.com/assert.owl#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
  xml:base="http://www.aifb.uni-karlsruhe.de/petri/PNOntology.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.w3.org/2003/11/swrlb"/>
    <owl:imports rdf:resource="http://www.w3.org/2003/11/swrl"/>
  </owl:Ontology>
  <rdfs:Class rdf:ID="Value"/>
  <owl:Class rdf:ID="Insert">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasAttribute"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Class rdf:ID="IndividualDataItem"/>
        </owl:someValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Transition">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasLogicalConcept"/>
        </owl:onProperty>
        <owl:maxCardinality
            rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Node"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="placeRef"/>
        </owl:onProperty>
```

```
      <owl:minCardinality
         rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >0</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Operation"/>
<owl:Class rdf:ID="ToPlace">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Arc"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasNode"/>
      </owl:onProperty>
      <owl:someValuesFrom rdf:resource="#Transition"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="LogicalConcept">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasCondition"/>
      </owl:onProperty>
      <owl:cardinality
         rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality
         rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasOperation"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasAttribute"/>
      </owl:onProperty>
      <owl:minCardinality
         rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Delete">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasAttribute"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#IndividualDataItem"/>
```

```
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Attribute">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:minCardinality
          rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:minCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasValue"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Axiom_1">
  <owl:equivalentClass>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Transition"/>
        <owl:Class rdf:ID="Place"/>
      </owl:unionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:about="#Node">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:minCardinality
          rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >0</owl:minCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasNode"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf
          rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>
<owl:Class rdf:ID="PetriNet">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality
          rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="initialElement"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf
          rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom>
        <owl:Class>
          <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#Transition"/>
            <owl:Class rdf:about="#Place"/>
```

```
          </owl:unionOf>
        </owl:Class>
      </owl:someValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasNode"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:minCardinality
          rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >0</owl:minCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasArc"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Condition"/>
<owl:Class rdf:about="#Arc">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:minCardinality
          rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >0</owl:minCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasArc"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf
          rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>
<owl:Class rdf:ID="FromPlace">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasNode"/>
      </owl:onProperty>
      <owl:someValuesFrom>
        <owl:Class rdf:about="#Place"/>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Arc"/>
</owl:Class>
<owl:Class rdf:about="#IndividualDataItem">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality
          rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >0</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasMarking"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
```

```
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasAttribute"/>
      </owl:onProperty>
      <owl:minCardinality
          rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom rdf:resource="#IndividualDataItem"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasAttribute"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Place">
  <rdfs:subClassOf rdf:resource="#Node"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="transRef"/>
      </owl:onProperty>
      <owl:minCardinality
          rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >0</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality
          rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasMarking"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="arcPossible">
  <rdfs:range rdf:resource="#PetriNet"/>
  <rdfs:domain rdf:resource="#Node"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasNode">
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Transition"/>
        <owl:Class rdf:about="#Place"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
  <rdfs:domain rdf:resource="#PetriNet"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="findSuccessor">
  <rdfs:range rdf:resource="#Node"/>
  <rdfs:domain rdf:resource="#Node"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#transRef">
```

```
    <rdfs:range rdf:resource="#Transition"/>
    <rdfs:domain rdf:resource="#Place"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#hasCondition">
    <rdfs:range rdf:resource="#Condition"/>
    <rdfs:domain rdf:resource="#LogicalConcept"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#hasMarking">
    <rdfs:range rdf:resource="#IndividualDataItem"/>
    <rdfs:domain rdf:resource="#Place"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#hasOperation">
    <rdfs:range rdf:resource="#Operation"/>
    <rdfs:domain rdf:resource="#LogicalConcept"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="before">
    <rdfs:domain rdf:resource="#Node"/>
    <rdfs:range rdf:resource="#Node"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="afterOrFirst">
    <rdfs:range rdf:resource="#Node"/>
    <rdfs:domain rdf:resource="#Node"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="after">
    <rdfs:domain rdf:resource="#Node"/>
    <rdfs:range rdf:resource="#Node"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="afterOrLast">
    <rdfs:range rdf:resource="#Node"/>
    <rdfs:domain rdf:resource="#Node"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="beforeOrLast">
    <rdfs:domain rdf:resource="#Node"/>
    <rdfs:range rdf:resource="#Node"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasInscription">
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Delete"/>
          <owl:Class rdf:about="#Insert"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#FromPlace"/>
          <owl:Class rdf:about="#ToPlace"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#initialElement">
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Place"/>
          <owl:Class rdf:about="#Transition"/>
        </owl:unionOf>
```

```
    </owl:Class>
  </rdfs:range>
  <rdfs:domain rdf:resource="#PetriNet"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasRef">
  <rdfs:range rdf:resource="#Value"/>
  <rdfs:domain rdf:resource="#Value"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="beforeOrFirst">
  <rdfs:domain rdf:resource="#Node"/>
  <rdfs:range rdf:resource="#Node"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasAttribute">
  <rdfs:range rdf:resource="#Attribute"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#LogicalConcept"/>
        <owl:Class rdf:about="#Delete"/>
        <owl:Class rdf:about="#Insert"/>
        <owl:Class rdf:about="#IndividualDataItem"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#placeRef">
  <rdfs:range rdf:resource="#Place"/>
  <rdfs:domain rdf:resource="#Transition"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasArc">
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#FromPlace"/>
        <owl:Class rdf:about="#ToPlace"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
  <rdfs:domain rdf:resource="#PetriNet"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasValue">
  <rdfs:range rdf:resource="#Value"/>
  <rdfs:domain rdf:resource="#Attribute"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isPossible">
  <rdfs:range rdf:resource="#PetriNet"/>
  <rdfs:domain rdf:resource="#Node"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasLogicalConcept">
  <rdfs:range rdf:resource="#LogicalConcept"/>
  <rdfs:domain rdf:resource="#Transition"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="findPredecessor">
  <rdfs:domain rdf:resource="#Node"/>
  <rdfs:range rdf:resource="#Node"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="and">
  <rdfs:range
         rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Condition"/>
```

```
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasName">
  <rdfs:range
         rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="function">
  <rdfs:domain rdf:resource="#Operation"/>
  <rdfs:range
         rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="businessRuleSatisfied">
  <rdfs:range
         rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
  <rdfs:domain rdf:resource="#PetriNet"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="exists">
  <rdfs:range
         rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Condition"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="forall">
  <rdfs:domain rdf:resource="#Condition"/>
  <rdfs:range
         rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="query">
  <rdfs:domain rdf:resource="#PetriNet"/>
  <rdfs:range
         rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="isPossibleEvent">
  <rdfs:domain rdf:resource="#PetriNet"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="isSelected">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Transition"/>
        <owl:Class rdf:about="#Place"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range>
    <owl:DataRange>
      <owl:oneOf rdf:parseType="Resource">
        <rdf:rest rdf:parseType="Resource">
          <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-
          syntax-ns#nil"/>
          <rdf:first
          rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
          >false</rdf:first>
        </rdf:rest>
        <rdf:first
          rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >true</rdf:first>
      </owl:oneOf>
    </owl:DataRange>
  </rdfs:range>
</owl:DatatypeProperty>
```

```
<owl:DatatypeProperty rdf:ID="hasAddress">
  <rdfs:range
         rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:FunctionalProperty rdf:ID="hasCurrency">
  <rdfs:range
         rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdf:type
         rdf:resource="http://www.w3.org/2002/07/owl#DatatypePropert
         y"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasAmount">
  <rdf:type
         rdf:resource="http://www.w3.org/2002/07/owl#DatatypePropert
         y"/>
  <rdfs:range
         rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:FunctionalProperty>
```

Anhang B Initiale SWRL Regeln

1. Transition(?origin) ∧ Place(?successor) ∧ placeRef(?origin, ?successor)
   → findSuccessor(?origin,?successor)
2. Place(?origin) ∧ Transition(?successor) ∧ transRef(?origin, ?successor)
   → findSuccessor(?origin,?successor)
3. Transition(?origin) ∧ Place(?predecessor) ∧ transRef(?predecessor, ?origin)
   → findPredecessor(?origin,?predecessor)
4. Place(?origin) ∧ Transition(?predecessor) ∧ placeRef(?predecessor, ?origin)
   → findPredecessor(?origin,?predecessor)
5. findPredecessor(?node1,?node2) → after(?node1,?node2)
6. after(?node1,?intermediateNode) ∧ findPredecessor(?intermediateNode,?node2)
   → after(?node1,?node2)
7. after(?node1,?node2) → afterOrFirst(?node1,?node2)
8. isSelected(?node, true) → afterOrFirst(?node,?node)
9. after(?node1,?node2) → afterOrLast(?node1,?node2)
10. isSelected(?node,true)
    → afterOrLast(?node,?node)
11.findSuccessor(?node1,?node2)
    → before(?node1,?node2)
12. before(?node1,?intermediateNode) ∧ findSuccessor(?intermediateNode,?node2)
    → before(?node1,?node2)
13. before(?node1,?node2)
    → beforeOrFirst(?node,?node)
13. initialElement(?net,?node)
    → beforeOrFirst(?node1,?node2)
14. initialElement(?net,?node)
    → beforeOrFirst(?node1,?node2)
15. before(?node1,?node2)
    → beforeOrLast(?node,?node)
16. isSelected(?node, true)
    → beforeOrLast(?node,?node)
17. isSelected(?node, true) ∧ initialElement(?net,?node2) ∧ Transition(?node1) ∧ Place(?node2)
    → arcPossible(?node1,?net)
18. isSelected(?node, true) ∧ initialElement(?net,?node2) ∧ Place(?node1) ∧ Transition(?node2)
    → arcPossible(?node1,?net)

Anhang C Evaluation

Fragebogen zur Evaluation von Ähnlichkeitsmaßen für Geschäftsprozess-modelle

Frage 1)

Über welche Erfahrungen verfügen Sie in der Geschäftsprozessmodellierung?

- ☐ überhaupt keine
- ☐ wenige (ein paar Beispielprozesse modelliert)
- ☐ mittelmäßige (ab und zu modelliere ich Praxisbeispiele)
- ☐ über gute; allerdings modelliere ich unregelmäßig
- ☐ sehr viele (ich modelliere auch regelmäßig)
- ☐ ich bin Experte

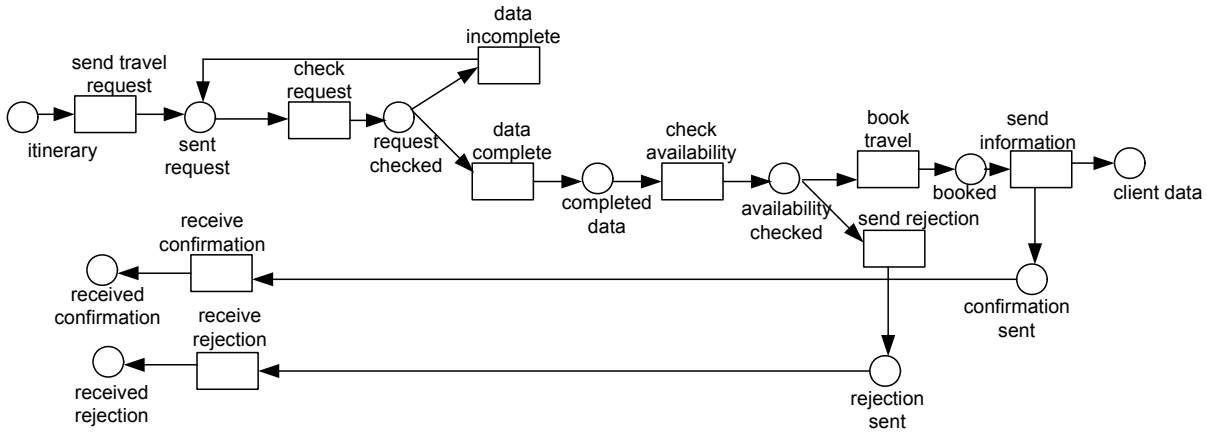
Frage 2)

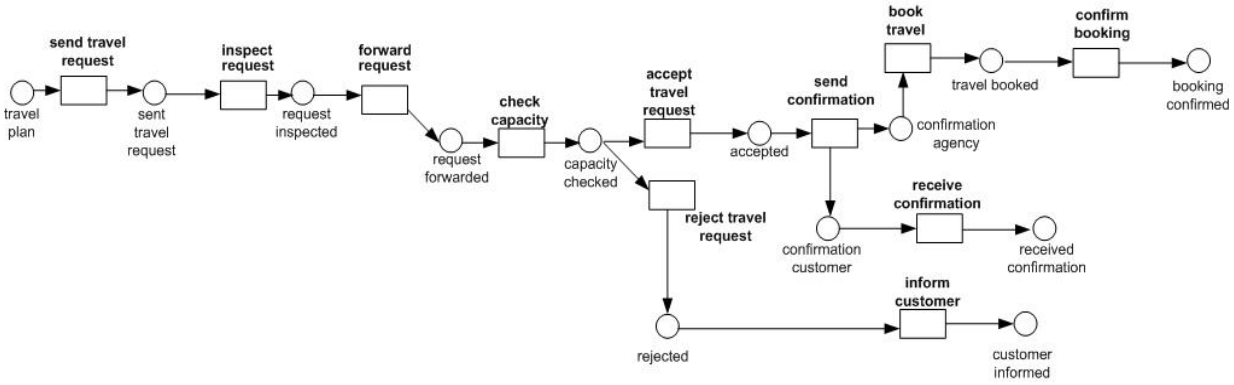
Wie würden Sie ihre Vertrautheit mit der Modellierungsdomäne einschätzen (Domänenkenntnisse bei der Modellierung von Geschäftsprozessen) (nur beantworten, falls Frage 1 mindestens mit *wenige* beantwortet wurde)

- ☐ ich verfüge über überhaupt keine Vertrautheit
- ☐ ich verfüge über wenige Kenntnisse in den Modellierungsdomänen
- ☐ ich verfüge über durchschnittliche Kenntnisse in den Modellierungsdomänen
- ☐ ich verfüge über gute Kenntnisse in den Modellierungsdomänen
- ☐ ich kenne mich in den meisten Modellierungsdomänen aus
- ☐ ich bin in jeder Modellierungsdomäne Experte

Frage 3)

Betrachten Sie die beiden Geschäftsprozessmodelle:

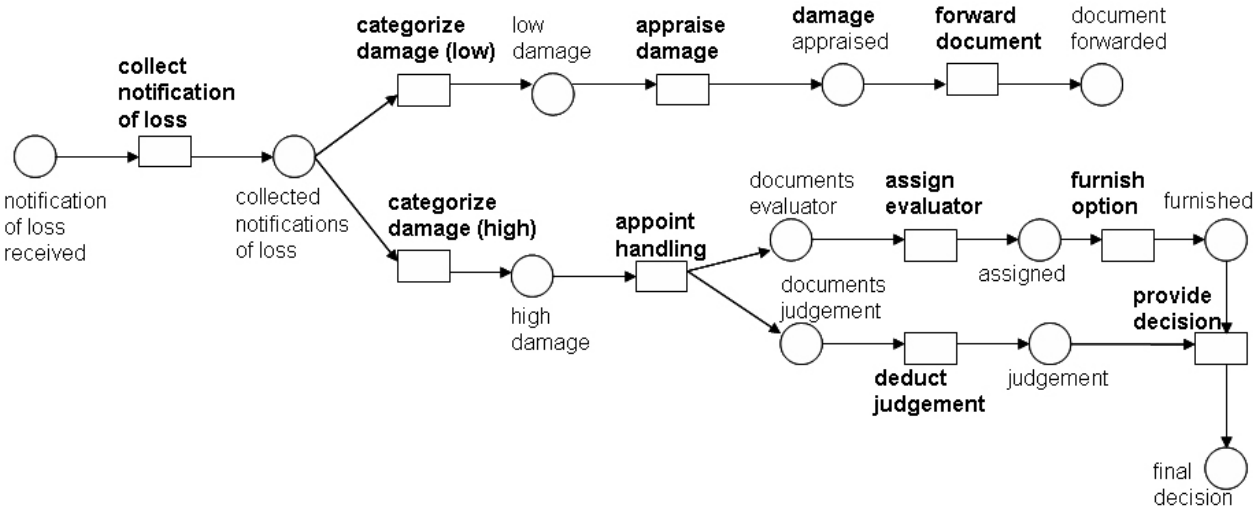


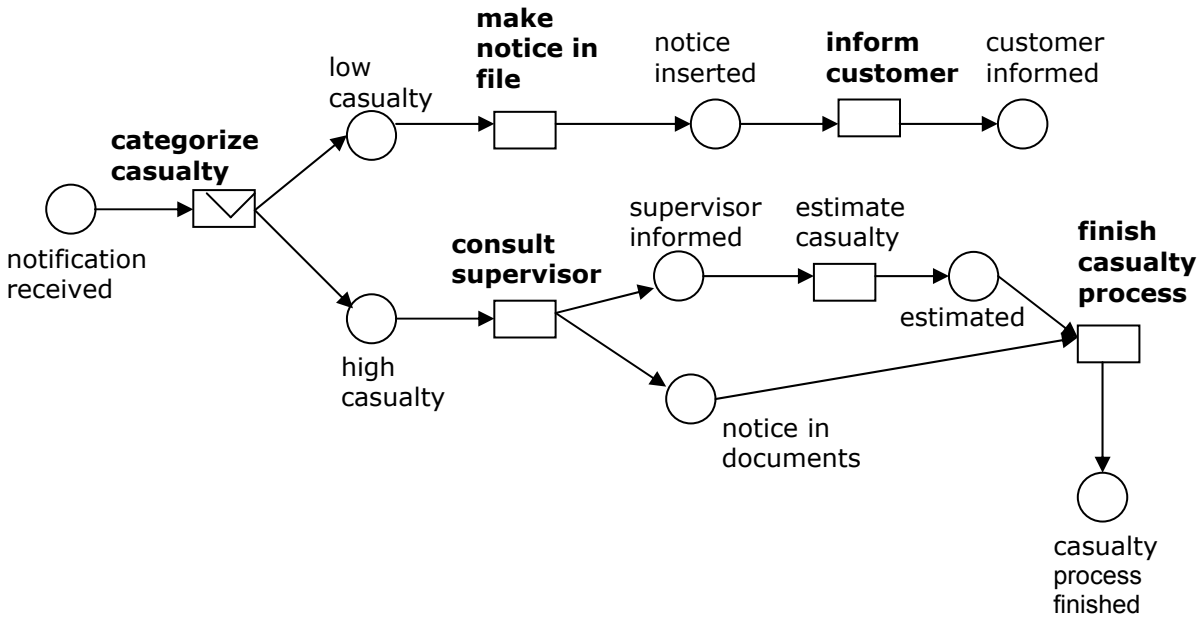


Vergleichen Sie die begriffliche Ähnlichkeit der Prozesselementnamen der beiden Geschäftsprozesse. Wenn Sie diese begriffliche Ähnlichkeit aufaddieren, wie hoch würden Sie die durchschnittliche Gesamtähnlichkeit der Geschäftsprozesse (ohne Berücksichtigung der Struktur) einschätzen? (0.0 = unähnlich, 1.0 = identisch)

Frage 4)

Betrachten Sie die beiden Geschäftsprozessmodelle:

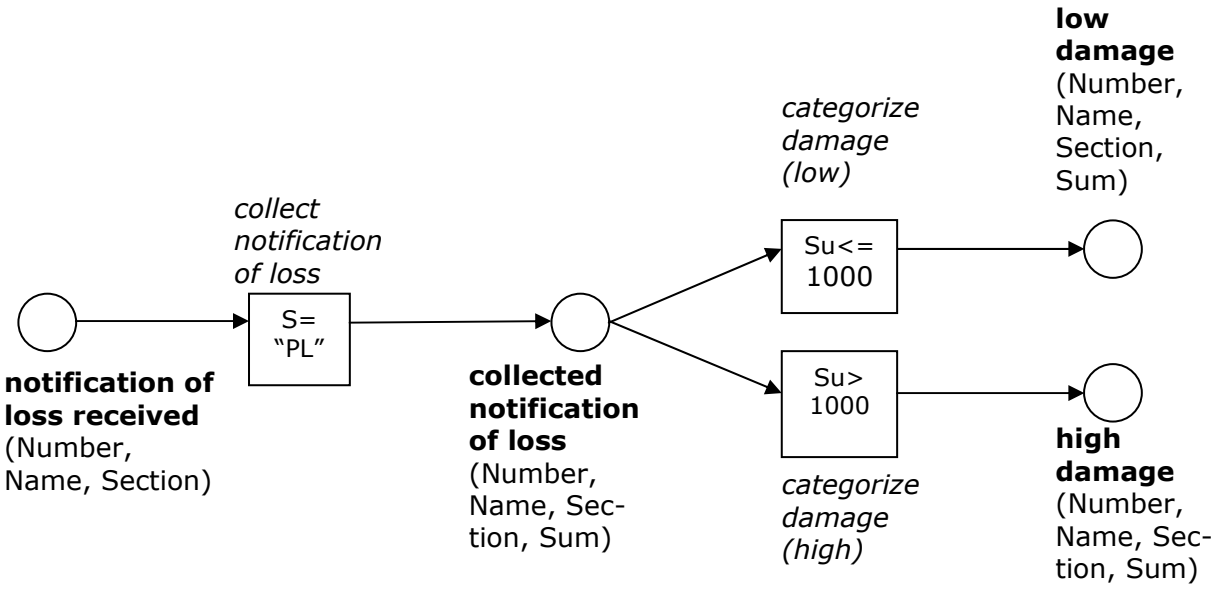


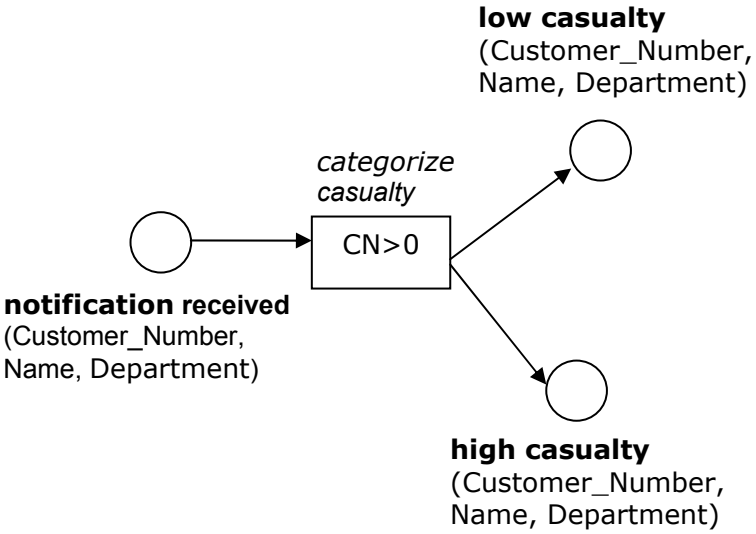


Vergleichen Sie die begriffliche Ähnlichkeit der Prozesselementnamen der beiden Geschäftsprozesse. Wenn Sie diese begriffliche Ähnlichkeit aufaddieren, wie hoch würden Sie die durchschnittliche Gesamtähnlichkeit der Geschäftsprozesse (ohne Berücksichtigung der Struktur) einschätzen? (0.0 = unähnlich, 1.0 = identisch)

Frage 5)

Betrachten Sie die beiden Geschäftsprozessmodelle:





Zu den einzelnen Prozesselementnamen wurden noch Attribute eingefügt. Wie hoch würden Sie nun die durchschnittliche Gesamtähnlichkeit der Geschäftsprozesse einschätzen?

Frage 6)

Betrachten Sie die folgenden Prozesselementpaare, die aus den Geschäftsprozessen in Frage 3 entnommen wurden:

Name	Ähnlichkeitswert
book travel *vs.* book travel	
inspect request *vs.* check request	
send confirmation vs. send information	
rejected vs. rejection (sent)	
check availability vs. check capacity	
request checked vs. request inspected	
confirmation agency vs. confirmation	
Availability vs. travel plan	
Quantity 1 vs. Seat (das sind Attribute)	
Paris vs. FRA (Attribute)	
client data vs. confirmation customer	
status data vs. capacity checked	

Wie hoch würden Sie die begriffliche Ähnlichkeit der Prozesselementpaare einschätzen?

Frage 7)

Eine automatische Berechnung der Ähnlichkeit zwischen den Prozessmodellen aus Frage 3 hat folgende Ergebnisse ausgegeben:

Name	Ähnlichkeitswert	Beurteilung
book travel *vs.* book travel	0.9655	
inspect request *vs.* check request	0.8451	
send confirmation *vs.* send information	0.8429	
rejected *vs.* rejection	0.6409	
check availability *vs.* check capacity	0.5621	
request checked *vs.* request inspected	0.5384	
confirmation agency *vs.* confirmation	0.2955	
Availability *vs.* travel plan	0.1641	
Quantity *vs.* Seat	0.1252	
Paris *vs.* FRA	0.0964	
client data *vs.* confirmation customer	0.0512	
Status data *vs.* capacity checked	0.0335	

Wie beurteilen Sie die berechneten Ähnlichkeitswerte? (1 sehr gut; 6 ungenügend)

Frage 8)

Betrachten Sie die folgenden Prozesselementpaare, die aus den Geschäftsprozessen in Frage 4 und 5 entnommen wurden:

Name	Ähnlichkeitswert
categorize damage *vs.* categorize casualty	
Loss Name *vs.* Loss Name (Attribute)	
Notification received *vs.* notification	
low damage *vs.* low casualty	
high casualty *vs.* high damage	
collected notification loss *vs.* notification received	
assign evaluator *vs.* estimate casualty	
estimated *vs.* assigned	
forward document *vs.* inform customer	
finish casualty process *vs.* furnish option	
damage appraised *vs.* notice inserted	
make notice file *vs.* appraise damage	
supervisor informed *vs.* documents evaluator	
consult supervisor *vs.* appoint handling	
customer informed *vs.* document forwarded	

Wie hoch würden Sie die begriffliche Ähnlichkeit der Prozesselementpaare einschätzen?

Frage 9)

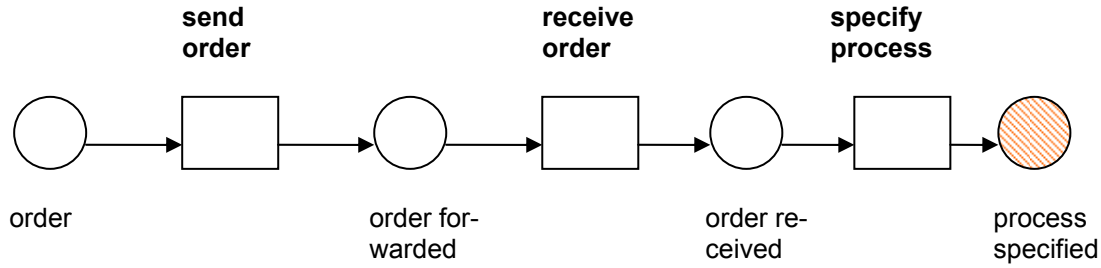
Eine automatische Berechnung der Ähnlichkeit zwischen den Prozessmodellen aus Frage 3 hat folgende Ergebnisse ausgegeben:

Name	Einschätzung	Beurteilung
categorize damage *vs.* categorize casualty	0.7599	
Loss Name *vs.* Loss Name	0.75	
notification received *vs.* notification	0.6837	
low damage *vs.* low casualty	0.6598	
high casualty *vs.* high damage	0.6598	
Collected notification loss *vs.* notification received	0.4705	
assign evaluator *vs.* estimate casualty	0.4392	
estimated *vs.*assigned	0.3935	
forward document *vs.* inform customer	0.2548	
finish casualty process *vs.* furnish option	0.2174	
damage appraised *vs.* notice inserted	0.2133	
make notice file *vs.* appraise damage	0.2101	
supervisor informed *vs.* documents evaluator	0.1974	
consult supervisor *vs.* appoint handling	0.1755	
customer informed *vs.* document forwarded	0.1297	

Wie beurteilen Sie die berechneten Ähnlichkeitswerte? (1 sehr gut; 6 ungenügend)

Frage 10)

Stellen Sie sich vor, Sie würden den folgenden Geschäftsprozess modellieren:



Die automatische Empfehlungskomponente schlägt ab der Stelle „process specified" die folgenden Prozessfragmente vor:

```
  - check order (Transition)
    - order checked (Stelle)
      - order 24000 (Transition)
        - inform manager
           - manager informed
      - order under
24000(Transition)
        - accept order
          - order accepted
```

```
- check inquiry
   - inquiry not ok
     - reject inquiry
       - inquiry rejected
   - inquiry ok
     - make offer
       - offer made
         - send offer
           - offer sent
             - receive order
               - order received
                 - check order
                   - order checked
```

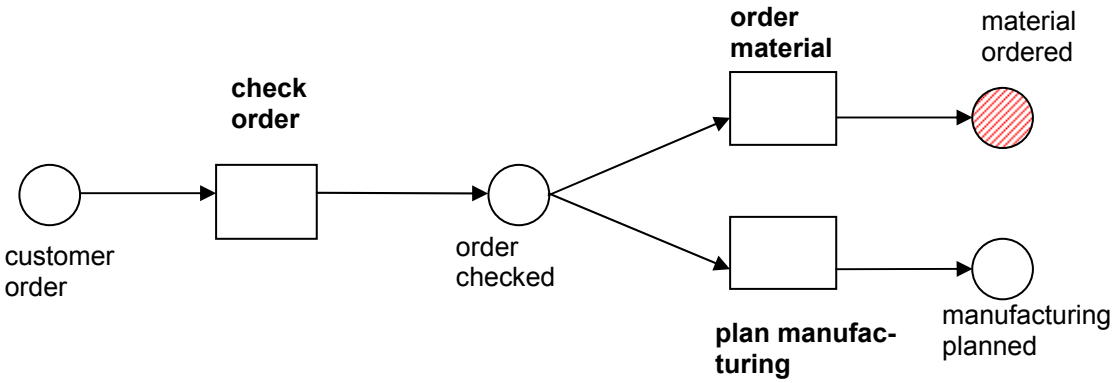
```
- send order
  - order sent
    - send cancellation
      - cancellation sent
        - receive response
          - response received
            - accept cancellation
              - accepted
```

```
- check inventory
 - inventory checked
  - consider requirement
    - requirement considered
      - generate order
        - order generated
          - release order
            - order terminated
      - open item
        - item opened
          - calculate requirement
            - requirement calculated
```

Welche Prozessfragmente bzw. Teile der Prozessfragmente würden Sie als passende Prozessteile nach der Stelle „process specified" auswählen?

Frage 11)

Stellen Sie sich vor, Sie würden den folgenden Geschäftsprozess modellieren:



Angenommen Sie müssen sich bei der Prozessmodellierung nur an eine der beiden Geschäftsregeln halten:

1) IF order was checked THEN manufacture item
2) A customer order requires to check the order THEN manufacture item AND send article

Die automatische Empfehlungskomponente schlägt ab der Stelle „material ordered" die folgenden Prozessfragmente vor:

```
- manufacture item
  - item manufactured
    - ship item
      - item shipped
        - customer pay
          - item paid
```

```
- document order
  - order documented
    - order shipment
      - shipment ordered
```

```
  - assay order
    - order assayed
      - request material
        - material requested
          - compensate material
            - material compensated
             - plan fabrication
              - fabrication planned
                - fabricate production
                  - production manu
factured
                    - send article
                      - article sent
```

```
- register packing data
  - packing data registered
    - register material cost
      - material cost registered
        - create procalculation
          - procalculation created
```

Welche Prozessfragmente bzw. Teile der Prozessfragmente würden Sie als passende Prozessteile nach der Stelle „material ordered" auswählen?

Alle Eingabewerte des neuronalen Netzes

- Eingabewerte für Transitionen

Name	sim_{syn}	sim_{ling}	sim_{str}
book travel *vs.* book travel	1.0	1.0	1.0
inspect request *vs.* check request	0.6154	0.7436	1.0
send confirmation vs. send information	0.8125	0.875	1.0
check availability vs. check capacity	0.5	0.6667	1.0
categorize damage *vs.* categorize casualty	0.6471	0.7647	1.0
assign evaluator *vs.* estimate casualty	0.3125	0.3125	1.0
forward document *vs.* inform customer	0.2	0.2	1.0
finish casualty process *vs.* furnish option	0	0.3537	1.0
make notice file *vs.* appraise damage	0.0667	0.1548	1.0
consult supervisor *vs.* appoint handling	0.0625	0.0893	1.0

- Eingabewerte für Stellen

Name	sim_{syn}	sim_{ling}	sim_{str}
rejected vs. rejection	0.625	0.625	1.0
request checked vs. request inspected	0.6667	0.7778	1.0
confirmation agency vs. confirmation	0.4167	0.6667	1.0
Availability vs. travel plan	0.0909	0.0909	1.0
client data vs. confirmation customer	0	0.1879	1.0
status data vs. capacity checked	0	0	1.0
Notification received *vs.* notification	0.3571	0.8	0
low damage *vs.* low casualty	0.4	0.6	1.0
high casualty *vs.* high damage	0.4	0.6	1.0

collected notification loss *vs.* notification received	0.1429	0.4833	1.0
estimated *vs.* assigned	0.375	0.375	1.0
damage appraised *vs.* notice inserted	0.2	0.2	1.0
supervisor informed *vs.* documents evaluator	0.0526	0.0721	1.0
customer informed *vs.* document forwarded	0.2353	0.2402	1.0

Index