SEMPA – Ein Ansatz des Semantischen Prozessmanagements zur Planung von Prozessmodellen

Bernd Heinrich, Marc Bewernik, Matthias Henneberger, Alexander Krammer, Florian Lautenbacher

SEMPA – Ein Ansatz des Semantischen Prozessmanagements zur Planung von Prozessmodellen

Inhaltsbeschreibung:

Derzeit erfolgt die Entwicklung und Modellierung von Prozessen weitgehend manuell. Prozessänderungen, die aufgrund bspw. von Marktentwicklungen notwendig sind, sind daher oftmals aufwändig. Der Beitrag stellt SEMPA vor, einen Ansatz zur teilautomatisierten Planung von Prozessmodellen. Ausgehend von der semantischen Beschreibung von Aktionen mit Hilfe einer Ontologie, kann für eine gegebene Problemstellung ein Prozessmodell mit Kontrollflussstrukturen konstruiert werden. Die prototypische Umsetzung und ein Prozessbeispiel aus dem Finanzdienstleistungsbereich veranschaulichen die Anwendung des Ansatzes.

Stichworte: Semantisches Prozessmanagement, Planung von Prozessmodellen, Planungsalgorithmen

Zusammenfassung:

Unternehmen müssen ihre Prozesse laufend an veränderte Marktentwicklungen anpassen. Die dafür notwendige Modellierung und Verbesserung der Prozesse ist jedoch derzeit oftmals noch mit hohem manuellem Aufwand verbunden. Im Beitrag wird ein Ansatz des Semantischen Prozessmanagements vorgestellt, der eine teilautomatisierte Erstellung (im Sinne einer Planung) von Prozessmodellen aus einzelnen Aktionen ermöglicht:

- Den Ausgangspunkt bilden mit Hilfe einer Ontologie semantisch beschriebene und in einer Prozessbibliothek gespeicherte Aktionen.
- Semantische Analysen und Inferenzen sind notwendig, um die Abhängigkeiten zwischen Aktionen abzuleiten und dadurch die Planung von Prozessmodellen zu ermöglichen. Dabei werden auch Kontrollflussstrukturen in den Prozessmodellen eingeplant.
- Im Unterschied zu Ansätzen der Webservice-Komposition können auf diese Weise technologieunabhängige Prozessmodelle erstellt werden, die danach - bspw. mit den Fachbereichen - abgestimmt werden können.

SEMPA – A Semantic Business Process Management Approach for the Planning of Process Models

Abstract:

Currently process modeling is mostly done manually. Therefore, the initial design of process models as well as changes to process models which are frequently necessary to react to new market developments or new regulations are time-consuming tasks. In this paper we introduce SEMPA, an approach for the partly automatic planning of process models. Using ontologies to semantically describe actions – as envisioned in Semantic Business Process Management –, a process model for a specified problem setting can be created automatically. In comparison to existing planning algorithms our approach creates process models including control structures and is able to cope with complex and numerical input and output parameters of actions. The prototypical implementation as well as an example taken from the financial services domain illustrate the practical benefit of our approach.

Keywords: Semantic Business Process Management, Planning of Process Models, Planning Algorithms

1 Einleitung

Infolge von Marktveränderungen müssen Unternehmen ihre Prozesse schnell restrukturieren bzw. neu entwickeln können. Dies betrifft sowohl die inner- als auch die zwischenbetrieblichen Prozesse, um bspw. auf geänderte Kundenanforderungen und Konkurrenzangebote mit eigenen Leistungen reagieren zu können. In vielen Fällen ist hier jedoch die flexible Erstellung und Anpassung der Prozessmodelle der Engpass (Becker, Kahn 2003; van der Aalst et al. 2006). Heutige Vorgehensweisen der Prozessmodellierung erscheinen im Hinblick auf diese Anforderung oftmals nur z. T. geeignet, da die eigentliche Erstellung i. d. R. manuell durch einen Modellierer – wenn auch durch ein Modellierungstool unterstützt – erfolgt. Die manuelle Modellierung ist selbst beim Vorliegen entsprechender Kenntnisse aufwändig, insbesondere wenn die Prozesse komplex und viele Unternehmensbereiche beteiligt sind. Zudem ist die Modellierung und fortlaufende Pflege oft mit hohem Kommunikations- und Abklärungsbedarf aufgrund einer unterschiedlich verwendeten Terminologie der Beteiligten verbunden (Kugeler und Rosemann 1998). Selbst wenn hier auf Referenzprozessmodelle zurückgegriffen wird, müssten diese ebenfalls aufgrund der Änderungen zuerst angepasst werden, was wiederum Zeit benötigt und Aufwand verursacht (darüber hinaus sind geeignete Referenzprozessmodelle für viele Domänen oftmals nur teilweise, in abstrakter Form oder gar nicht vorhanden). Eine schnelle und unter ökonomischen Aspekten sinnvolle Erstellung und Anpassung der Prozessmodelle ist somit nur bedingt möglich, weswegen sich letztlich auch viele Prozessmanagementabteilungen in Unternehmen mit dem Vorwurf auseinander setzen müssen, zu hohe Kosten – im Vergleich zum erzielbaren Nutzen der Prozessmodellierung – zu generieren.

Im Rahmen des Semantischen Prozessmanagements wird hier u. a. die Erhöhung des Automatisierungsgrads diskutiert, d. h. Prozessmodelle sollen autokomplettiert oder teilautomatisiert erstellt bzw. angepasst werden (Betz et al. 2006; Hepp et al. 2005). Ziel ist es, den bisherigen hohen (manuellen) Aufwand der Entwicklung und Pflege von Prozessmodellen zu verringern. Die Aufgabe einer automatisierten Modellerstellung ist dabei als Planungsproblem (Ghallab et al. 2004, Russel und Norvig 2004) zu verstehen. Deshalb wird von der Planung von Prozessmodellen gesprochen, die ihrerseits aus Aktionen (PA) erstellt werden. Die Grundlagen für eine solche Planung sind durch die Konzepte des Semantic Web zunehmend gegeben. So existieren Planungsansätze für die (semantische) Webservice-Komposition, die jedoch für die Planung von Prozessmodellen nur beschränkt übertragbar sind (Yan et al. 2006).

Im Rahmen des DFG-Projekts SEMPRO wurde für die Planung von Prozessmodellen ein eigener Ansatz – genannt SEMPA (SEMantic-based Planning Approach) – erarbeitet. Bild 1 verdeutlicht die Idee sowie die im Weiteren verwendeten Konzepte und Begriffe: Basierend auf einer semantischen Beschreibung der Anwendungsdomäne (Ontologie) und der Spezifikation von PA in einer Bibliothek (vgl. Betz et al. 2006; Hepp und Roman 2007) erstellt der Planer zulässige Prozessmodelle (d. h. Abfolgen von PA inklusive notwendiger Kontrollflussstrukturen wie ein Exclusive choice), die eine gegebene Problemstellung (bestehend aus einem Initial- und ein oder mehreren

Zielzuständen) erfüllen. Von teilautomatisiert wird deshalb gesprochen, da die erstellten Modelle als Vorschläge zu verstehen sind, die anschließend abzustimmen und unter ökonomischen Aspekten zu bewerten sind. Als Modellierungssprache werden derzeit UML-Aktivitätsdiagramme genutzt, wobei geplant ist, zukünftig auch andere Sprachen (bspw. Petri-Netze) zu unterstützen.

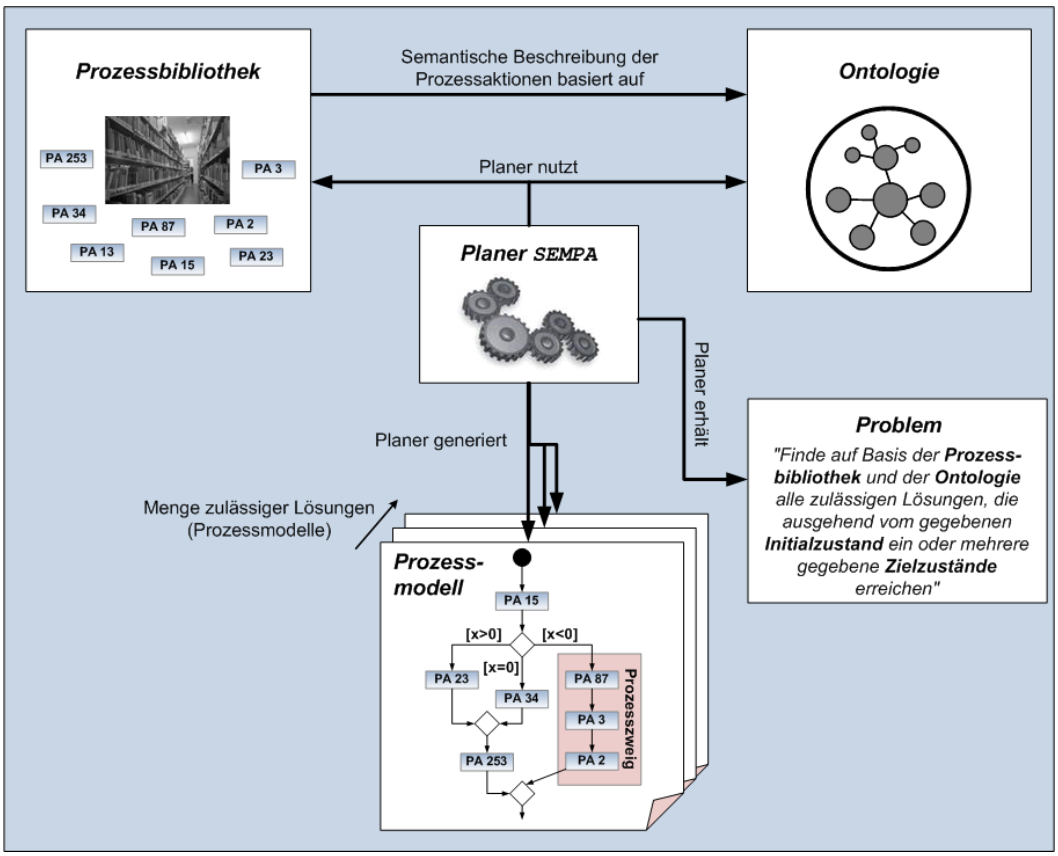


Bild 1 Grundlegende Idee des Projekts SEMPR0

Die Idee soll an einem vereinfachten Fallbeispiel verdeutlicht werden, das im Beitrag fortgeführt wird (für die formale Definition vgl. Kapitel 3). Ziel ist die Planung eines Prozessmodells für die *Verarbeitung von Aktienorders* eines Finanzdienstleisters (FDL). Im Initialzustand soll die zu verarbeitende *Aktienorder* (bspw. nach der Kundeneingabe) vorliegen. In den Zielzuständen ist definiert, dass die *Aktienorder* vom *Typ Kauf* oder *Verkauf* mit einer positiven *Ordersumme ausgeführt* und zugleich *risikoüberprüft* sein muss. Auf Basis dieser Problemstellung sowie vorhandener Prozessmodelle würde das Prozessmanagement nunmehr zusammen mit beteiligten Abteilungen ein geeignetes Modell entwickeln. Dies entspricht dem bei Finanzprodukten nicht unüblichen Fall, dass neue Prozessmodelle in Teilen auf existierende PA zurückgreifen. Diese Erstellung soll durch den Planungsansatz unterstützt werden. Aus Darstellungsgründen wurde hier die größere Anzahl an PA, die dem Bsp. ursprünglich zugrunde lag, auf die folgenden PA beschränkt: *Aktienorder validieren*, *Kompetenzprüfung durchführen, Erweiterte Kompetenzprüfung durchführen, Risikoprüfung durchführen, Aktienorder ausführen, Aktienorder verbuchen* (ausführlich beschrieben in Kapitel 3). Sind die PA semantisch annotiert, so kann bspw. geprüft werden, ob *Aktienorder* im Anwendungskontext einer Subklasse von *Order* entspricht. Dadurch können für

die Erstellung des Prozessmodells auch PA berücksichtigt werden, die sich nicht direkt auf eine *Aktienorder* beziehen. So ist *Kompetenzprüfung durchführen* eine PA, die bei allen Arten von *Orders* (bspw. *Aktienorder*, *Fondsorder*) anwendbar ist. Ohne semantische Annotation könnte ein Planungsansatz keine Verbindung zwischen *Orders* und *Aktienorders* herstellen und auch bei einer manuellen Modellierung verlangen derartige Relationen (in komplexeren Fällen) ohne eine explizierte Definition oft Abstimmungen. Somit soll durch die Verwendung der Ontologie auch der Abklärungs- und Abstimmungsbedarf bzgl. uneinheitlicher Terminologie reduziert werden. Zwar muss auch hier vorab eine Festlegung auf eine einheitliche Ontologie erfolgen. Der dafür erforderliche (einmalige) Aufwand relativiert sich jedoch durch den geringeren Aufwand für die (häufig erfolgenden) Entwicklungen bzw. Änderungen von Modellen.

Der Beitrag ist wie folgt strukturiert: In Kapitel 2 erfolgt eine Einordnung in das Forschungsgebiet des Semantischen Prozessmanagements, die Definition der Anforderungen an die Planung und die Diskussion bestehender Ansätze. Danach wird in Kapitel 3 der SEMPA-Planer vorgestellt. Hier steht die Fragestellung im Mittelpunkt, welche semantischen Analysen und Inferenzen notwendig sind, um die Planung von Prozessmodellen – insbesondere mit komplexen Kontrollflussstrukturen – zu ermöglichen. Das Kapitel 4 stellt die prototypische Implementierung des Ansatzes dar. Kapitel 5 fasst die wesentlichen Ergebnisse kritisch zusammen.

2 Literaturüberblick und Anforderungen

In Abschnitt 2.1 wird der Beitrag in den Bereich des Semantischen Prozessmanagements eingeordnet. Danach werden die Anforderungen an die Planung definiert. In den Abschnitten 2.3 und 2.4 werden bisherige Ansätze diskutiert.

2.1 Semantisches Prozessmanagement

Unter dem Stichwort Semantisches Prozessmanagement werden Ansätze diskutiert, die Konzepte und Technologien aus dem Bereich des Semantic Web auf das Prozessmanagement übertragen (Betz et al. 2006; Brockmans et al. 2006; Hepp et al. 2005; Drumm et al. 2006; Thomas und Fellmann 2006). Ziel des Semantischen Prozessmanagements ist es, einen höheren Automatisierungsgrad in der Erstellung und Anpassung von Prozessmodellen sowie in der Prozessdurchführung zu erreichen, indem bspw. Modellelemente semantisch angereichert werden. D. h. die darin verwendeten Begriffe sind als Konzepte in einer Ontologie dargestellt.

Zwar legen, wie Thomas und Fellmann (2006) feststellen, auch die seit Jahren bekannten und etablierten Ansätze für die Prozess- und Unternehmensmodellierung (bspw. ARIS, SOM) die Semantik der verwendbaren Metamodellelemente und deren Beziehungen fest (vgl. z. B. Scheer 1991, Ferstl und Sinz 2001). Jedoch bleibt die Begriffswahl für die Bezeichnung der Modellelemente und deren Semantik i. d. R. dem Modellierer überlassen. Wenn jedoch Begriffe nicht einheitlich verwendet werden, d. h. mit gleichen Begriffen oft unterschiedliche Teile der Diskurswelt bezeichnet werden oder gleiche Teile der Diskurswelt unterschiedlich benannt werden, sind Mehrdeutigkeiten die Folge (Thomas und Fellmann 2006). Diese führen u. a. zu Abklärungsbedarf bzw. zu einer aufwändigen Weiterentwicklung der Prozessmodelle.

Durch die Verwendung von Ontologien werden Begriffe konzeptualisiert und deren Relationen festgelegt. Dies soll insbesondere eine bessere maschinelle Be- und Verarbeitung der Modelle ermöglichen. Somit besteht ein wesentlicher Vorteil darin, semantisch angereicherte Modelle für eine automatisierte Verarbeitung mit Hilfe von Werkzeugen, wie bspw. Inferenzmaschinen, zugänglich zu machen (im Vergleich zu früheren Ansätzen wie bspw. Fachbegriffsmodelle).
Hepp et al. (2005) definieren hier u. a. das Ziel, eine erweiterte Abfragemöglichkeit vorhandener Modelle zu ermöglichen. Dabei verwenden sie unterschiedliche Ontologien und nutzen semantisch annotierte Modelle (gespeichert in einer Prozessbibliothek). Daneben beschreiben Thomas und Fellmann (2006) die semantische Anreicherung von EPKs, um deren Verständlichkeit zu erhöhen und eine Übersetzung in verschiedene Sprachen zu unterstützen. Dies sieht eine Ontologie vor, die alle relevanten Konzepte des Anwendungskontexts beschreibt und für die Definition der EPK-Metamodellelemente sowie der Modellelemente selbst dient. Bei Brockmans et al. (2006) soll die Zusammenführung von Prozessen, die auf unterschiedlichen Begrifflichkeiten basieren, durch eine semantische Integration unterstützt werden, indem die mittels Petri-Netzen dargestellten Prozessmodelle jeweils in eine OWL-Ontologie überführt und anschließend mit Hilfe einer weiteren Ontologie auf Ähnlichkeit untersucht werden. Ein derartiger Vergleich wird nach Betz et al. (2006) auch für die Autokomplettierung von Petri-Netz Modellen genutzt und in Hornung et al. (2007) durch die Nutzung von Geschäftsregeln erweitert. Erstellte Modellteile werden hierfür anhand von Ähnlichkeitsmaßen mit vorhandenen Modellen verglichen. Ein anderes Ziel verfolgt der Ansatz von Rao et al. (2006), der den Modellierer bei der Erstellung von Prozessmodellen unterstützt. Dabei wird den Modellierern beim semantischen Annotieren der PA (hier *Composable Elements* genannt) assistiert, die im weiteren Verlauf zur teilautomatischen Modellerstellung verwendet werden. In diesem Ansatz werden jedoch nur Vorschläge gemacht, welche PA jeweils zur Lösung einer Problemstellung generell zur Verfügung stehen, ein Prozessmodell wird hingegen nicht automatisch erstellt.
Zusammenfassend stehen in den genannten Beiträgen v. a. die semantische Anreicherung von Prozessmodellen im Mittelpunkt und weniger deren Planung, die jedoch z. T. als nächster Schritt angekündigt wird. Deswegen werden in Abschnitt 2.3 primär existierende Planer für die Komposition von Webservices diskutiert, da diese zum einen Analogien aufweisen und zum anderen schon einen höheren Entwicklungsstand haben. Zudem ergeben sich Anknüpfungspunkte zu Betz et al. (2006) oder Hepp und Roman (2007) bzgl. der Definition einer Prozessbibliothek, in der semantisch annotierte PA gespeichert sind. Hier soll bspw. analog zu Hepp et al. (2005) die Auszeichnung einer PA durch die semantische Beschreibung ihrer Input- und Outputparameter erfolgen.

2.2 Anforderungen an die Planung von Prozessmodellen

Auf Basis bisheriger Arbeiten wurde für die Planung von Prozessmodellen ein Anforderungskatalog erstellt (zu einzelnen Anforderungen vgl. Constantinescu et al. 2004; Lang und Su 2005;

Meyer und Kuropka 2006; Meyer und Weske 2006; Pathak et al. 2006a und 2006b; ter Beek et al. 2006):

(R1) *Input- und Outputparameter von PA*: Der Planer muss auf Basis der Input- und Outputparameter der PA deren Vorgänger-Nachfolger-Beziehung ermitteln. Benötigt bspw. eine PA *Aktienorder validieren* einen Inputparameter *Aktienorder*, so muss dieser entweder bereits im Initialzustand vorliegen oder zuvor als Outputparameter erzeugt worden sein. Die Beschreibung der Input- und Outputparameter der PA soll dabei einmalig und unabhängig von einer einzelnen Problemstellung zur Planung eines Prozessmodells erfolgen.

(R2) *0 bis n-fache Verwendung einer PA*: Der Planer muss entscheiden, ob eine PA für eine gegebene Problemstellung erforderlich ist. Zudem muss der Planer in der Lage sein - falls notwendig - eine PA im Prozessmodell mehrfach zu berücksichtigen.

(R3) *Datentypen*: Es muss möglich sein, Input- und Outputparametern von PA Datentypen - wie in Biron und Malhotra (2004) definiert - in der Ontologie zu zuweisen. Die *Ordersumme* kann bspw. vom Datentyp *positiveInteger* sein und besitzt damit einen definierten Wertebereich. Für jeden Parameter einer PA muss es zudem möglich sein festzulegen, ob die PA nur bestimmte Intervalle des Wertebereichs eines Inputparameters akzeptiert bzw. eines Outputparameters liefert. Dies wird im Folgenden als Restriktion bezeichnet. Bspw. soll eine PA *Erweiterte Kompetenzprüfung durchführen* nur für *Orders* mit einer *Ordersumme größer oder gleich 5.000* durchführbar sein.

(R4) *Atomare und zusammengesetzte Parameter*: Für die Planung von Prozessmodellen sind nicht nur – wie in anderen Ansätzen vereinfachend angenommen – atomare Input- und Outputparameter (bspw. Bezeichner mit Wahrheitswert) zu verarbeiten. Vielmehr ist es notwendig auch zusammengesetzte Parameter zu berücksichtigen, wie sie z. B. bei der Spezifikation einer *Order* – bestehend aus dem *Orderstatus*, *Handelsplatz*, der *Ordersumme*, usw. – erforderlich sind.

(R5) *Nicht-deterministisches Planen*: Es soll ein Prozessmodell und nicht nur eine einzelne Prozessausführung geplant werden. Aus diesem Grund sind alle *möglichen* Werte der Parameter zu berücksichtigen. Dies wird in der Literatur auch als nichtdeterministisches Planen bezeichnet (Russel und Norvig 2004; Meyer und Kuropka 2006). Folgt bspw. die PA *Erweiterte Kompetenzprüfung durchführen* mit ihrer Restriktion (*Ordersumme größer oder gleich 5.000*) nach einer PA, die eine *Ordersumme* im gesamten Wertebereich von *positiveInteger* liefern kann, so entsteht eine Differenzmenge. Der Planer muss dies berücksichtigen, wodurch u. U. alternative Abläufe im Prozessmodell notwendig werden.

(R6) *Semantikbasierte Inferenzmechanismen:* Damit die in der Ontologie hinterlegten Klassen und Relationen (bspw. subClassOf-Relation) berücksichtigt werden können, muss der Planer Inferenzmechanismen bei der Analyse der semantisch annotierten Input- und Outputparameter nutzen.

(R7) *Kontrollflussstrukturen*: Der Planer muss Prozessmodelle mit Kontrollflussstrukturen konstruieren können. Beispiele dafür sind Exclusive choice, Parallel split, Synchronization, Simple merge oder Arbitrary cycles (für eine Klassifikation auch in Bezug zu Workflow patterns vgl. van der Aalst et al. 2003).

(R8) *Menge zulässiger Lösungen*: Der Planer soll die Menge zulässiger Prozessmodelle (Lösungen) für eine gegebene Problemstellung liefern. Als zulässig wird jedes Prozessmodell bezeichnet, das ausgehend vom Initialzustand alle definierten Zielzustände erreicht. Zulässige Lösungen können sich demnach bspw. in der Reihenfolge der PA unterscheiden.

2.3 Analyse bisheriger Ansätze im Bereich Webservice-Komposition

Da im Bereich des Semantischen Prozessmanagements noch keine Planungsalgorithmen existieren, werden im Weiteren Ansätze zur automatisierten Komposition von Webservices betrachtet. Zwar gibt es hier in der Zielsetzung z. T. Unterschiede, jedoch bieten die Kompositionsansätze mit ihrem höheren Entwicklungsstand Orientierung. Zudem gleichen sich auch teilweise die definierten Anforderungen. So werden bspw. Services ebenso wie PA semantisch annotiert (bspw. mit OWL-S oder SWSF). Gemeinsam ist den meisten Ansätzen zudem, dass Sie Input- und Outputparameter sowie Vor- (preconditions) und Nachbedingungen (effects) zur Komposition nutzen. Wie bereits dargestellt, können in ähnlicher Form auch PA beschrieben werden. Da sich Vor- und Nachbedingungen bei der Planung von Prozessmodellen primär auf den Input und Output beziehen, können sie - wie später deutlich wird - über Restriktionen abgebildet werden. Insgesamt lassen sich zwischen der Planung von Prozessmodellen und der Webservice-Komposition Analogien finden. Tabelle 1 gibt einen Überblick über ausgewählte Ansätze, die für die Planung von Prozessmodellen untersucht wurden und stellt diese den obigen Anforderungen gegenüber. Die Tabelle verdeutlicht, dass einige Anforderungen von den bisherigen Ansätzen voll erfüllt werden (Haken in der entsprechenden Spalte). Eine Klammer um einen Haken drückt aus, dass die Anforderung nur teilweise unterstützt wird (aus Platzgründen beschränken wir uns im Weiteren auf die Diskussion wesentlicher Punkte).

Die erstgenannten Ansätze, wie SHOP2 oder Golog erfüllen zwar grundsätzlich die Anforderung an eine semantik-basierte Planung, sie benötigen jedoch umfangreiche Vorgaben, wie die bereits zu verwendenden Services, deren Vorgänger-Nachfolger-Beziehungen und Kontrollflussstrukturen. Die Anforderungen (R2), (R5) und (R7) sind deshalb nicht bzw. nicht vollständig erfüllt. Gerade die Erzeugung nichtdeterministischer Pläne bzw. die Erstellung von Prozessmodellen mit Kontrollflussstrukturen werden derzeit nur von wenigen Algorithmen angegangen (Pistore et al. 2005; Meyer und Weske 2006). Die meisten Ansätze erzeugen stattdessen rein sequenzielle Pläne. Mit Ausnahme von Meyer und Weske (2006), die auch auf Parallel splits eingehen (siehe auch Kuropka und Weske 2008), werden nur Exclusive choices betrachtet. Daneben zeigt sich, dass zusammengesetzte Input- und Outputparameter sowie numerische und alphanumerische Datentypen jeweils nur von einem Ansatz verarbeitet werden können. Der Ansatz von Constantinescu et al. (2004) sieht zwar bspw. numerische Datentypen vor, erlaubt jedoch zum einen keine

zusammengesetzten Parameter und geht darüber hinaus kaum auf die Nutzung von Inferenzmechanismen ein.

Tabelle 1 Ausgewählte Ansätze zur Webservice-Komposition im Überblick

Autoren	Planeralgorithmus	(R1) Input-/Outputparameter als Planungsbasis	(R2) 0 bis n-fache Verwendung einer Aktion	(R3) Komplexe Input-/Outputparameter	(R4) Datentypen	(R5) Nicht-deterministisches Planen	(R6) Semantik-basierte Inferenzmechanismen	(R7) Kontrollfluss-strukturen	(R8) Menge zulässiger Lösungen
Lang und Su 2005	And/Or-graphs	✓	✓	✓	-	-	(✓)	-	(✓)
Sirin et al. 2004	SHOP2 (HTN Planung)	✓	(✓)	-	-	-	(✓)	-	-
McIlraith und Son 2002	Golog language (situation calculus)	✓	(✓)	-	-	-	(✓)	-	-
Pathak et al. 2006a, Pathak et al. 2006b	Symbolic Transition Systems	✓	✓	-	-	(✓)	-	(✓)	✓
Pistore et al. 2005	State Transition Systems	✓	(✓)	-	-	✓	-	(✓)	-
Meyer und Weske 2006; Kuropka und Weske 2008	Heuristic Search via Hill Climbing	✓	✓	-	-	✓	-	(✓)	-
Constantinescu et al. 2004	Type-based Composition	✓	(✓)	-	✓	✓	(✓)	(✓)	-

2.4 Abgrenzung von bisherigen Planungsansätzen

Viele der Arbeiten des Abschnitts 2.3 beruhen auf Planungsansätzen aus der Künstlichen Intelligenz. Auch bei der hier diskutierten Problemstellung handelt es sich im Kern um ein Planungsproblem. Jedoch zeigt sich, dass eine Gegenüberstellung der bisherigen Planungsansätze mit den obigen Anforderungen kaum sinnvoll ist. Es existieren zwar Lösungsverfahren für die nichtdeterministische Planung (bspw. Bertoli et al. 2001; Cimatti et al. 1998; Kabanza et al. 1997; Weld et al. 1998). Diese bieten z. T. auch grundsätzlich geeignete Suchstrategien - z. B. Vorwärtstraversierung im Zustandsraum (vgl. Bertoli et al. 2001) - und können auch Hinweise für die Berücksichtigung von Exclusive choices in Prozessmodellen geben. Die Abbildung bspw. komplexer Parameter mit unterschiedlichen Datentypen wird von diesen Verfahren jedoch nicht behandelt. So greifen die meisten Planungsansätze auf Aussagenlogik bzw. Beschreibungslogik - meist in Anlehnung an die so genannte STRIPS-Sprache (Ghallab et al. 2004, S. 49) - für die Beschreibung von Aktionen bzw. deren Vor- und Nachbedingungen zurück. Auch weitere Anforderungen wie die semantische Analyse werden nicht berücksichtigt. Bisherige Planungsansätze können daher lediglich Ausgangspunkt für die Entwicklung eines eigenen Ansatzes sein.

Analogien finden sich auch zu Ansätzen des automatischen Programmierens und der Algorithmensynthese. Da bei der Algorithmensynthese (z. B. Minkwitz, 1993) i. A. als Eingabe

eine generische Beschreibung eines zu synthetisierenden Algorithmus (sequenzielle Abfolge von Operationen in einer Hochsprache) erwartet wird, jedoch bei der Prozessplanung im Initialzustand keine Abfolgen vorliegen, ist eine Übertragung schwierig. Beim automatischen Programmieren (vgl. Balzer, 1985) werden syntaktisch korrekte Programme durch automatisches Schließen erstellt, um ein definiertes Ziel zu erreichen. Dazu wird eine formale Spezifikation in eine Implementierung umgewandelt, wobei jedoch bspw. weder Kontrollflussstrukturen generiert werden, noch ein Reasoning erfolgt.

Insgesamt ergeben sich deshalb für die Prozessplanung spezielle neue Fragestellungen: Wie lassen sich mittels semantischer Analysen die Vorgänger-Nachfolger-Beziehungen zwischen PA ableiten? Wie lassen sich diese für die Anforderung des nichtdeterministischen Planens und für die Konstruktion von Kontrollflussstrukturen nutzen? Diese Fragen stellen einen wesentlichen Unterschied gegenüber bisherigen Planungsproblemen dar und werden auch von existierenden Webservice-Kompositionsansätzen nicht umfassend gelöst. Deshalb sollen diese Fragen den Schwerpunkt des Beitrags bilden.

3 Entwicklung des SEMPA-Planers

3.1 Überblick und Definitionen

Die Planung von Prozessmodellen erfolgt mit SEMPA in drei Schritten (vgl. Bild 2):

1) Im ersten Schritt sind mittels eines Inferenzmechanismus die für die Problemstellung benötigten PA sowie deren Vorgänger-Nachfolger-Beziehungen zu ermitteln. Basis sind hier die semantisch beschriebenen Parameter der PA. Algorithmisch erfolgt dies mit Hilfe einer im Zielzustand beginnenden Rückwärtstraversierung, wodurch ein Aktionsabhängigkeitsgraph (AAG) erstellt wird. Der AAG umfasst (vereinfacht) die notwendigen PA, deren Parameter als Knoten sowie die Vorgänger-Nachfolger-Beziehungen als Kanten.
2) Da der AAG noch keine *direkten* Vorgänger-Nachfolger-Beziehungen zwischen PA beschreibt, wird im zweiten Schritt mittels einer Vorwärtstraversierung der Aktionszustandsgraph (AZG) erstellt. Dieser besteht aus PA und Zuständen (als Menge an Parametern, die nach der bisherigen Einplanung von PA vorhanden sind).
3) Mit dem AZG liegt die Menge zulässiger Lösungen vor, allerdings noch nicht in der Form eines syntaktisch korrekten Prozessmodells. Der AZG wird daher im dritten Schritt verwendet, um Prozessmodelle (hier: UML-Aktivitätsdiagramm) mit Kontrollflussstrukturen zu erstellen.

Die Unterteilung des Planers folgt den nachstehenden Designentscheidungen:

- Um zu vermeiden, dass eine mehrmalige (redundante) semantische Analyse gleicher Informationen (v. a. Input-/Outputparameter) erfolgt, wird der Inferenzmechanismus im ersten Schritt eingesetzt. Die semantischen Informationen im AAG werden dann für die weiteren Schritte genutzt.
- Bereits mit der Erstellung des AAG können alle PA eliminiert werden, die aufgrund nicht benötigter Outputs nicht Teil einer zulässigen Lösung sein können und deshalb zur Reduktion der Laufzeit nicht weiter zu betrachten sind.

- Die Vorgänger-Nachfolger-Beziehungen im AAG müssen nur einmalig analysiert werden und können danach für mehrere Problemstellungen (Planung verschiedener Prozessmodelle) Verwendung finden.
- Der AZG enthält nach dem zweiten Schritt die *direkten* Vorgänger-Nachfolger-Beziehungen und die Zustände. Tritt in einem Ast *B* des AZG ein bereits in einem anderen Ast *A* gespeicherter Zustand erneut auf, kann geprüft werden, ob Ast *A* zu einer zulässigen Lösung oder zu einem Fehlerabbruch (Zielzustand ist nicht erreichbar – Ast ist wie im Bild 2 nicht im AZG zu berücksichtigen) geführt hat. Dadurch lassen sich identische Generierungs- und Prüfungsschritte vermeiden. Desweiteren ist es bei Berücksichtigung der Planungslaufzeit vorteilhaft, Kontrollflussstrukturen wie das Exclusive Choice erst im dritten Schritt zu planen, da ansonsten die Zustände im AZG anhand von Wertebereichen der Zustandsvariablen (für die eine jeweilige Aktion *ausschließlich ausführbar* ist) unterteilt werden müsste. Letzteres würde dazu führen, dass wesentlich nicht nur mehr Zustände im AZG zu betrachten sind. Vielmehr werden i. d. R. auch einige Pfade danach gar nicht den angestrebten Zielzustand erreichen, d. nutzlos.
- Der AZG stellt eine generische Basis dar, auf der Prozessmodelle mit verschiedenen Modellierungssprachen konstruierbar sind. Sprachenspezifische Eigenheiten sind demnach separiert von den anderen Schritten nur im dritten Schritt zu berücksichtigen.

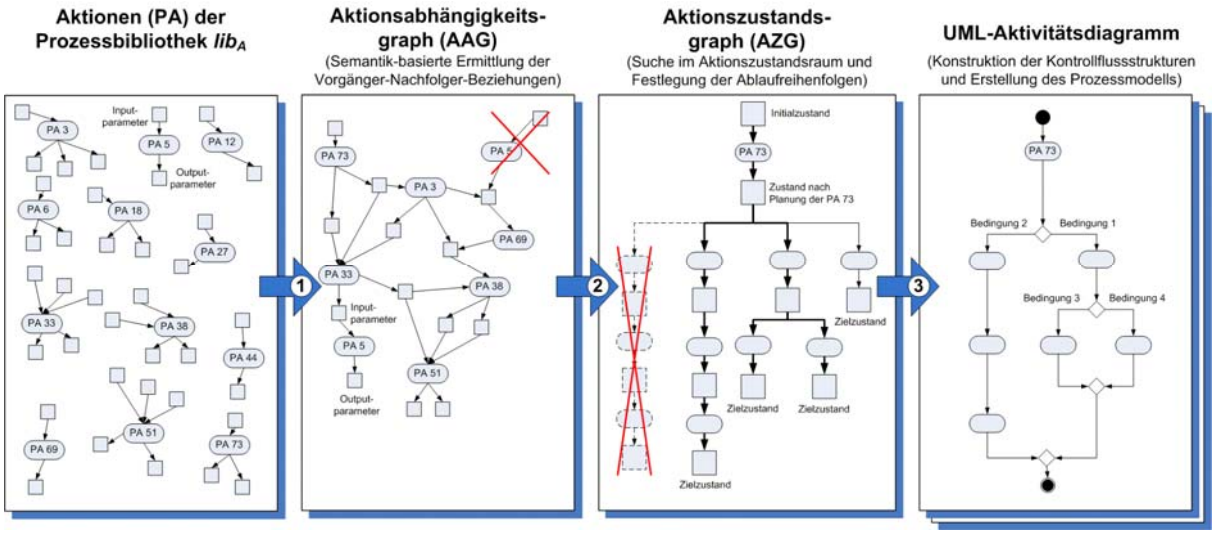


Bild 2 Teilschritte des SEMPA-Planers

Bevor - wie in der Einleitung begründet - v. a. auf den ersten Schritt des Planers genauer eingegangen wird, werden zunächst die grundlegenden Annahmen und Definitionen erläutert:

(A1) Sei P die Menge aller Input- und Outputparameter und *Dom* die Menge der Domänen dieser Parameter. Eine Domäne beschreibt den Wertebereich, für den ein Parameter definiert ist. Ein Parameter $p \in P$ ist entweder atomar und besteht aus einem Tripel der Form (l_p, dom_p, r_p) oder ist zusammengesetzt und enthält als Tupel $(l_p, \{(l_{p_1}, dom_{p_1}, r_{p_1}), \ldots, (l_{p_n}, dom_{p_n}, r_{p_n})\})$ eine Menge von $n \in \mathbb{N}$ atomaren Parametern. Da-

bei steht l_{p_k} für den Namen des *k*-ten Parameters, $dom_{p_k} \in Dom$ für die Domäne des Parameters p_k und r_{p_k} für die Restriktionen (dabei gilt $r_{p_k} \subseteq dom_{p_k}$ mit $r_{p_k} \neq \emptyset$ und $k \in 1..n$).

Parameter nehmen für die Planung die Rolle von Zustandsvariablen ein. Die Domäne definiert die Menge möglicher Ausprägungen eines Parameters und wird entweder als primitiver Datentyp (vgl. hierzu Datentypen des XML-Schemas in Biron und Malhotra (2004)) oder als ontologische Klasse dargestellt. Im Orderprozess hat bspw. der Parameter *Risikoprüfung* den Datentyp Boolean, während der Parameter *Orderzustand* als Domäne die ontologische Klasse *Zustand* besitzt, die in der Ontologie als Aufzählung ihrer möglichen Ausprägungen modelliert ist. Die Domäne kann durch Restriktionen auf Teilbereiche eingeschränkt werden.

(A2) Eine PA $a \in A$ mit *A* als Menge aller gegebenen PA ist gekennzeichnet durch ihre Inputparameter $In_a \subseteq P$, die für die Ausführung der PA benötigt werden, sowie ihre Outputparameter $Out_a \subseteq P$, welche nach Ausführung vollständig zur Verfügung stehen. Alle PA werden zusammen mit ihren Input- und Outputparametern gespeichert und stehen als Tripel (a, In_a, Out_a) in einer Prozessbibliothek lib_A zur Verfügung.

Wie in anderen Ansätzen soll eine PA über ihre Input- und Outputparameter spezifiziert werden. Alternativ könnten hier - bei Bedarf - auch Teilprozesse anstatt PA Verwendung finden, die dann analog zu (A2) zu beschreiben sind.

(A3) Es existiert eine Ontologie *Ont*, in der die Parameter mit ihren Domänen in Form von Klassen zusammen mit den binären Relationen dieser Klassen semantisch beschrieben sind.

Ein Parameter der Form (l_p, dom_p, r_p) wird dabei wie folgt in einer OWL-Ontologie umgesetzt: Der Name des Parameters l_p bezeichnet eine Klasse in der Ontologie. Die Domäne dom_p wird bei einem primitiven Datentyp (z. B. *positiveInteger*) in eine owl:DataProperty bzw. bei einer Klasse in eine owl:Class übersetzt. Es wird dabei von einer gegebenen Ontologie ausgegangen. Sollten mehrere Ontologien existieren, müssten diese mittels Techniken des Ontology-Matching zu einer Ontologie kombiniert werden (vgl. Kalfoglou und Schorlemmer 2003).

(A4) Ziel ist die Planung von Prozessmodellen als Lösung eines Problems, das definiert ist durch das Tupel *Prob* = ($lib_A, Ont, Init, Goals$). $Init \subseteq P$ ist die initiale Menge an Inputparametern (Initialzustand) und *Goals* die Zielzustände, d. h. die Menge der unterschiedlichen (Mengen von) Outputparametern ($Goals = \{Goal_1, \ldots, Goal_k, \ldots, Goal_m\}$ mit $k = 1, 2, \ldots, m \in \mathbb{N}$ und $Goal_k \subseteq P$).

Die Definitionen und Annahmen können anhand des einleitend eingeführten Beispiels verdeutlicht werden. Bild 3 stellt die Konzepte der Ontologie für das Beispiel dar. Dabei wird eine Klasse *Order* bestehend aus *Ordersumme*, *Ordertyp, Risikoprüfung* und *Orderzustand* definiert. Der *Ordertyp* ist entweder *Kauf* oder *Verkauf*, während der *Orderzustand* entweder *erfasst*, *gültig*, *ungültig*, *überprüft*, *ausgeführt* oder *verbucht* ist. Eine *Order* ist dabei entweder eine *Fonds*- oder eine *Aktienorder* (disjunkte Spezialisierungen). Die Klasse *Zustand* ist im Beispiel äquivalent zur Klasse *Orderzustand.* Eine *Order* kann einer *Risikoprüfung* unterzogen werden, die entweder *wahr* oder *falsch* als Ergebnis liefert.

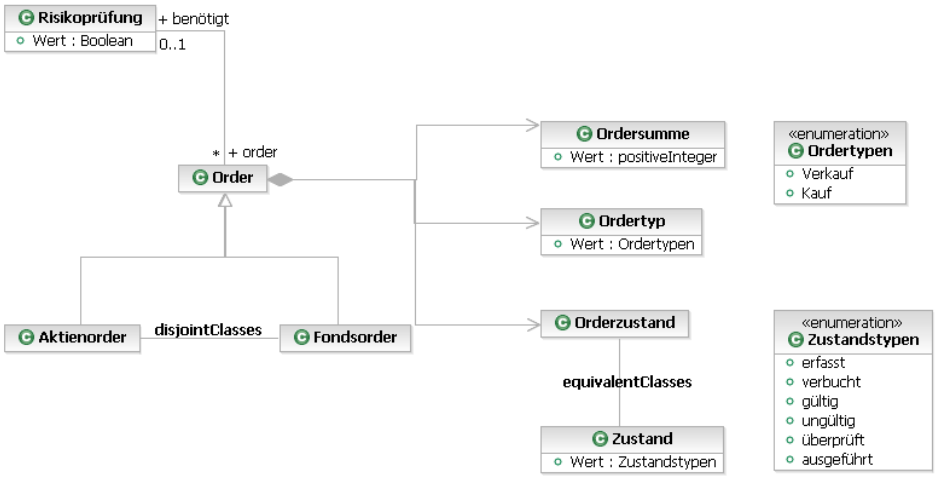


Bild 3 Ontologie im Beispiel

Basierend auf dieser Ontologie sind die eingeführten PA (Kapitel 1) semantisch annotiert. Dies ist in Bild 4 in der eingeführten Notation dargestellt.

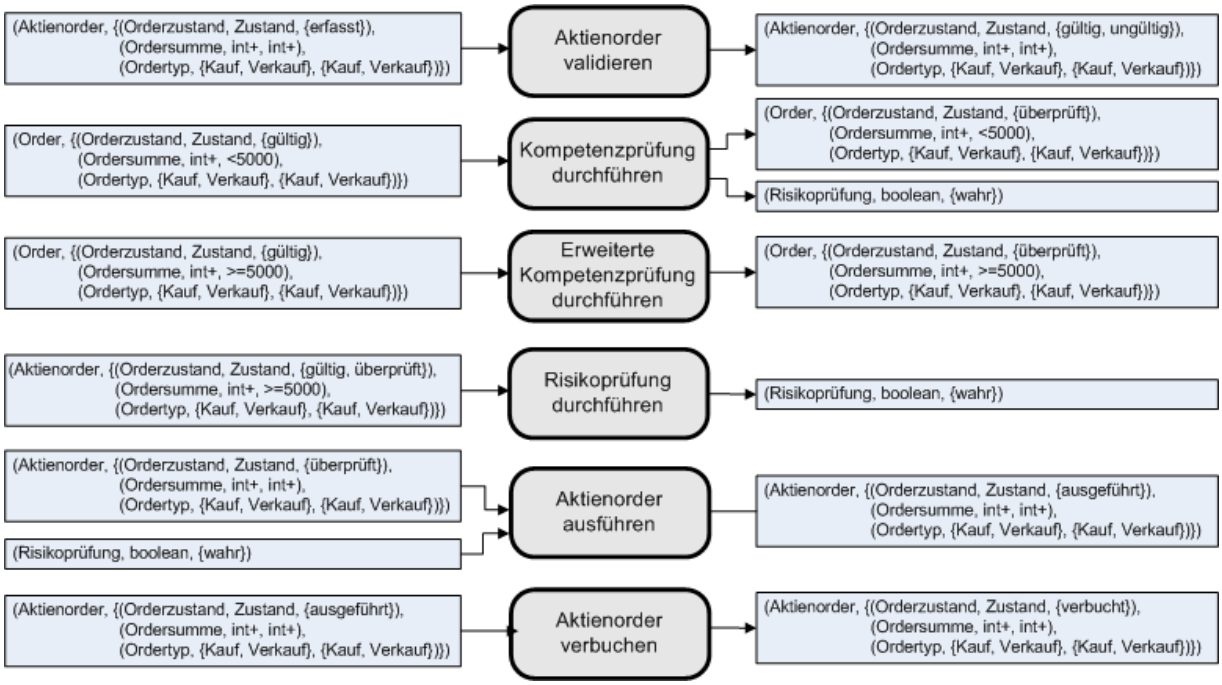
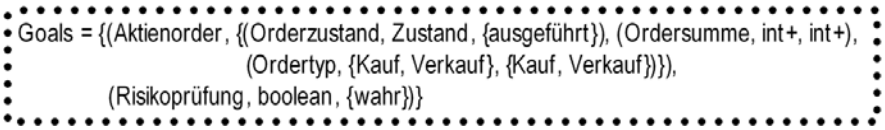


Bild 4 Semantische Annotation der PA

Als Problemstellung lassen sich der Initialzustand und die Zielzustände für die Planung wie folgt definieren: Im Initialzustand *Init* liegt eine Aktienorder vor, die bzgl. ihres *Orderzustands* erfasst ist, über eine positive ganzzahlige *Ordersumme* verfügt und vom Typ *Kauf* oder *Verkauf* ist:

Init = {(Aktienorder, {(Orderzustand, Zustand, {erfasst}), (Ordersumme, int+, int+), (Ordertyp, {Kauf, Verkauf}, {Kauf, Verkauf})})}

Daneben wird durch die Zielzustände *Goals* gefordert, dass eine *Aktienorder* vom *Ordertyp Kauf* oder *Verkauf* mit positiv ganzzahliger *Ordersumme* vorliegt, deren *Orderzustand* dem Zustand *ausgeführt* entspricht und für die eine *Risikoprüfung* durchgeführt wurde:



Basierend auf (A1) bis (A4) wird im Weiteren der SEMPA-Planer mit dem Schwerpunkt auf den ersten Teilschritt vorgestellt und am Fallbeispiel illustriert.

3.2 Semantische Analyse zur Ermittlung der Vorgänger-Nachfolger-Beziehungen

Im **ersten Schritt** müssen mittels Inferenzmechanismen die Vorgänger-Nachfolger-Beziehungen zwischen den PA für das gegebene Problem *Prob* ermittelt und damit der AAG erstellt werden. Vorgänger-Nachfolger-Beziehungen ergeben sich durch Abhängigkeiten der PA aufgrund ihrer Input- und Outputparameter.

Inferenzbasierte Ermittlung der Abhängigkeiten zwischen Aktionen

Es gilt also zunächst die Übereinstimmung von Input- und Outputparametern zu prüfen. Eine Übereinstimmung ist an zwei notwendige Bedingungen geknüpft: Erstens müssen semantische Relationen zwischen den als Klassen in der Ontologie repräsentierten Input- und Outputparametern existieren. Zweitens dürfen die durch die Restriktionen eingeschränkten Wertebereiche der Parameter nicht disjunkt sein (die Schnittmenge darf nicht der leeren Menge entsprechen). Treffen beide Bedingungen zu, so kann der Outputparameter als Inputparameter verwendet werden, d. h. es existiert eine Abhängigkeit zwischen den PA. Andernfalls liegt bzgl. des Input-Outputparameterpaares keine Abhängigkeit zwischen den PA vor.

Um die Bedingungen exakt zu definieren, betrachten wir jeweils einen Outputparameter $out_a=(l_{out_a}, dom_{out_a}, r_{out_a})$ einer PA *a* und einen Inputparameter $in_b=(l_{in_b}, dom_{in_b}, r_{in_b})$ einer PA *b*. Gleichzeitig unterscheiden wir zwischen einer vollständigen und einer partiellen Übereinstimmung des Outputparameters out_a mit in_b, wobei im ersten Fall der Outputparameter out_a ohne Einschränkung als Input der PA *b* verwendet werden kann, während im zweiten Fall Einschränkungen bspw. hinsichtlich des Wertebereichs existieren. Analog sprechen wir im ersten Fall von einer vollständigen und im zweiten Fall von einer partiellen Abhängigkeit der PA *b* von der PA *a.* Aus partiellen Abhängigkeiten lässt sich später u. a. auf ein Exclusive choice im Prozessmodell schließen. So ist vor Ausführung der PA *b* eine Fallunterscheidung bzgl. out_a notwendig, um denjenigen Teil des von PA *a* gelieferten Wertebereichs von out_a ebenfalls „abzudecken“, der von PA *b* mit dem Inputparameter in_b nicht verarbeitet werden kann.

Fehler! Verweisquelle konnte nicht gefunden werden. fasst die möglichen Konstellationen zusammen, welche im Weiteren detailliert beschrieben werden.

Tabelle 2 Bedingungen für die Übereinstimmung des Input-/Outputparameterpaars in_b, out_a

I. Analysiere semantische Relationen	II. Analysiere Parameterrestriktionen	Übereinstimmung
[1] $out_a = in_b \vee out_a \equiv in_b \vee out_a \sqsubset in_b \vee out_a \succ in_b$	[1.1] $r_{out_a} \subseteq r_{in_b}$	***vollständige Übereinstimmung***
	[1.2] $r_{out_a} \cap r_{in_b} \neq \varnothing \wedge r_{out_a} \setminus r_{in_b} \neq \varnothing$	***partielle Übereinstimmung***
	[1.3] $r_{out_a} \cap r_{in_b} = \varnothing$	***keine Übereinstimmung***
[2] $out_a \sqsupset in_b$	[2.1] $r_{out_a} \cap r_{in_b} \neq \varnothing$	***partielle Übereinstimmung***
	[2.2] $r_{out_a} \cap r_{in_b} = \varnothing$	***keine Übereinstimmung***
[3] $out_a \neq in_b \wedge out_a \not\equiv in_b \wedge out_a \prec in_b \wedge out_a \not\sqsubset in_b \wedge out_a \not\sqsupset in_b$		***keine Übereinstimmung***

Im ersten Schritt I werden die semantischen Relationen (vgl. Paolucci et al. 2002) analysiert (erste Bedingung für eine Übereinstimmung). Neben dem trivialen Fall der *Gleichheit* (=) sind hier insbesondere die Relationen *Äquivalenz* ($\equiv$), *Spezialisierung* ($\sqsubset$) sowie *Teil-Ganzes-* ($\prec$) zwischen Parametern von Bedeutung. Zur Beschreibung dieser Relationen sei die Menge $P_a \subseteq P$ die Menge aller atomaren Parameter von P sowie die Menge $P_z \subseteq P$ die Menge aller zusammengesetzten Parameter von P. Die *Äquivalenz* ($\equiv \subseteq P_a \times P_a$) liegt analog zur *Spezialisierung* ($\sqsubset \subseteq P_a \times P_a$) genau dann vor, wenn eine *equivalentClasses*-Relation bzw. eine *subClassOf*-Relation zwischen den in der Ontologie als Klassen repräsentierten Parametern durch den Inferenzmechanismus (Sirin et al. 2005) jeweils ableitbar ist. Dagegen ist ein Parameter out_a, durch eine *Teil-Ganzes-Relation* ($\prec \subseteq P \times P_z$) mit einem Parameter in_b assoziiert, wenn entweder out_a ein atomarer Parameter ist, der gleichzeitig ein Element des zusammengesetzten Parameters in_b ist. Oder out_a ist wie in_b ein zusammengesetzter Parameter, wobei alle atomaren Elemente von out_a auch Element von in_b sind.

Auf Basis dieser Relationen ist die erste Bedingung für eine *vollständige* Übereinstimmung eines Outputparameters out_a mit einem Inputparameter in_b erfüllt (vgl. Fall [1] in **Fehler! Verweisquelle konnte nicht gefunden werden.**),

a) wenn out_a *gleich* oder *äquivalent* zu in_b ist ($out_a = in_b \vee out_a \equiv in_b$).
b) wenn out_a eine *Spezialisierung* von in_b ist ($out_a \sqsubset in_b$). Dies gilt, da jede Ausprägung von out_a auch eine Ausprägung von in_b ist und somit immer von PA *b* verarbeitet werden kann.
c) wenn in_b ein *Teil* des verfügbaren Outputparameters out_a ist ($out_a \succ in_b$).

Hingegen ist die erste Bedingung für eine *partielle* Übereinstimmung erfüllt (vgl. Fall [2] in Tabelle *2***Fehler! Verweisquelle konnte nicht gefunden werden.**), wenn in_b eine *Spezialisierung* von out_a ist ($out_a \sqsupset in_b$), da eine Ausprägung von out_a nicht in jedem Fall gleichzeitig eine Ausprägung von in_b sein muss. In allen anderen Fällen liegt keine Übereinstimmung vor.

Zu beachten ist auch, dass zwei Parameter p_1, $p_m \subseteq P$ falls sie nicht unmittelbar durch eine Relation miteinander assoziiert sind, auch über mehrere Relationen $p_1 \otimes \ldots \otimes p_i \otimes \ldots \otimes p_m$ ($m \in \mathbb{N}$) verknüpft sein können (dabei ist $\otimes$ ein Platzhalter für eine der beschriebenen Relationen $\{=, \equiv, \sqsubseteq, \prec\}$ bzw. Umkehr-Relationen $\{\sqsupseteq, \succ\}$). Bspw. besteht auch eine (partielle) Übereinstimmung zwischen zwei Parametern *Aktienorder* und *Wertpapierauftrag*, wenn diese nicht direkt durch eine Relation verknüpft sind, aber bspw. *Aktienorder* eine Spezialisierung einer *Order* und eine *Order* wiederum äquivalent zu einem *Wertpapierauftrag* ist (Assoziation mittels zwei Relationen).

Im nächsten Schritt II des Vergleichs sind zusätzlich die Parameterrestriktionen zu analysieren (zweite Bedingung für eine Übereinstimmung). Eine vollständige Übereinstimmung liegt vor, wenn die erste Bedingung für eine vollständige Übereinstimmung erfüllt ist (Fall [1]) und gleichzeitig die Restriktion des Outputparameters (welche den durch eine Aktion verarbeitbaren Wertebereich des Parameters definiert) Teilmenge der Restriktion des Inputparameters ist (Fall [1.1]). Existiert dagegen eine nicht-leere Schnittmenge und eine Differenzmenge zwischen den Restriktionen, so liegt lediglich eine partielle Übereinstimmung vor (Fall [1.2]). Eine partielle Übereinstimmung liegt zudem vor, wenn im ersten Schritt festgestellt wurde, dass in_b eine *Spezialisierung* von out_a ist (Fall [2]) und die Restriktionen beider Parameter eine nicht-leere Schnittmenge besitzen (Fall [2.1]). Ist die Schnittmenge der Restriktionen beider Parameter die leere Menge, so ist der Outputparameter nicht als Input verarbeitbar und es besteht keine Übereinstimmung (Fälle [1.3] und [2.2]).

Abbildung der Abhängigkeiten im Aktionsabhängigkeitsgraphen

Aus dem Vergleich der Input- und Outputparameter ergeben sich die Abhängigkeiten und damit die Vorgänger-Nachfolger-Beziehungen der PA. Eine Vorgänger-Nachfolger-Beziehung zwischen zwei PA *a* und *b* wird im AAG so abgebildet, dass von der Vorgänger-PA *a* (Knoten im Graph) eine gerichtete Kante zu einem Outputparameter out_a (Knoten im Graph) führt und von diesem wiederum eine gerichtete Kante zur Nachfolger-PA *b* (Knoten im Graph) existiert (mit dem Inputparameter in_b). Dafür ist es notwendig, den AAG als bipartiten Graph $G = (V, E)$ mit Knotenmenge V und Kantenmenge E zu definieren:

(D1) Die Knotenmenge V besteht aus der Vereinigung der Partition $Part_a \subseteq lib_A$, welche die PA enthält, und der Partition $Part_p \subseteq P$, welche die Input- und Outputparameter dieser PA umfasst. $V := Part_a \cup Part_p$

(D2) Die Kantenmenge E des Graphen ist die Vereinigung der Inputkanten E_{in} mit den Outputkanten E_{out}. $E := E_{in} \cup E_{out}$

(D3) E_{in} ist definiert als Menge der Inputkanten $((p, a), p_{in})$, bestehend aus allen gerichteten Kanten (p, a) mit Kantenbeschriftung p_{in}. $p \in Part_p$ ist ein Parameter, der von der PA $a \in Part_a$ als Inputparameter $p_{in} \in In_a$ benötigt wird. Dabei muss p_{in} nicht notwendigerweise identisch zu p sein, sondern lediglich in einer semantischen Relation zu p stehen. $E_{in} := \{((p, a), p_{in}) \mid p \in Part_p, p_{in} \in In_a, a \in Part_a, p_{in}$ *steht mit p in semantischer Relation*$\}$

(D4) E_{out} ist definiert als die Menge aller gerichteten Outputkanten (a, p_{out}), wobei $p_{out} \in Part_p$ einen Outputparameter der PA $a \in Part_a$ darstellt.

$E_{out} := \{(a, p_{out}) \mid p_{out} \in Out_a, p_{out} \in Part_p, a \in Part_a\}$

Bild 5 zeigt einen kleinen Ausschnitt eines AAG, der die bisherigen Ausführungen verdeutlicht. Die PA *Kompetenzprüfung durchführen* besitzt den zusammengesetzten Inputparameter *Order*, bestehend aus *gültigem Orderzustand*, einer *Ordersumme* kleiner als 5.000 und dem *Ordertyp Kauf* oder *Verkauf* (notiert an der Kante des AAG). Die PA *Aktienorder validieren* liefert zwar den Parameter *Aktienorder*, jedoch mit den Parametern *Orderzustand gültig* oder *ungültig*, *Ordersumme int+* und *Ordertyp Kauf* oder *Verkauf*. In der Ontologie sind folgende Relationen gegeben: *Aktienorder* $\sqsubseteq$ *Order*, *Fondsorder* $\sqsubseteq$ *Order* und *Aktienorder* ist disjunkt zu *Fondsorder*. Die semantische Analyse legt somit eine vollständige Übereinstimmung (jede *Aktienorder* ist aufgrund der *subClassOf*-Relation immer auch eine *Order*) zwischen dem Input- und dem Outputparameter nahe (Fall [1] in obiger Tabelle). Jedoch ergibt der Vergleich der Restriktionen, dass von *Aktienorder validieren* eine *Aktienorder* geliefert werden kann, deren *Orderzustand ungültig* und deren *Ordersumme* größer oder gleich 5.000 ist. Da eine solche *Aktienorder* als Outputparameter nur partiell mit der verarbeitbaren *Order* übereinstimmt und von der PA *Kompetenzprüfung durchführen* somit nicht immer verarbeitet werden kann, liegt lediglich eine partielle Abhängigkeit vor (Fall [1.2] in obiger Tabelle). Der verarbeitbare Input wird im AAG durch die Beschriftung der Inputkante einer PA notiert und damit die Kante eindeutig als partielle Abhängigkeit zwischen den PA gekennzeichnet. Gleiches gilt auch für die ebenfalls dargestellte partielle Abhängigkeit der PA *Erweiterte Kompetenzprüfung durchführen* von der PA *Aktienorder validieren*.

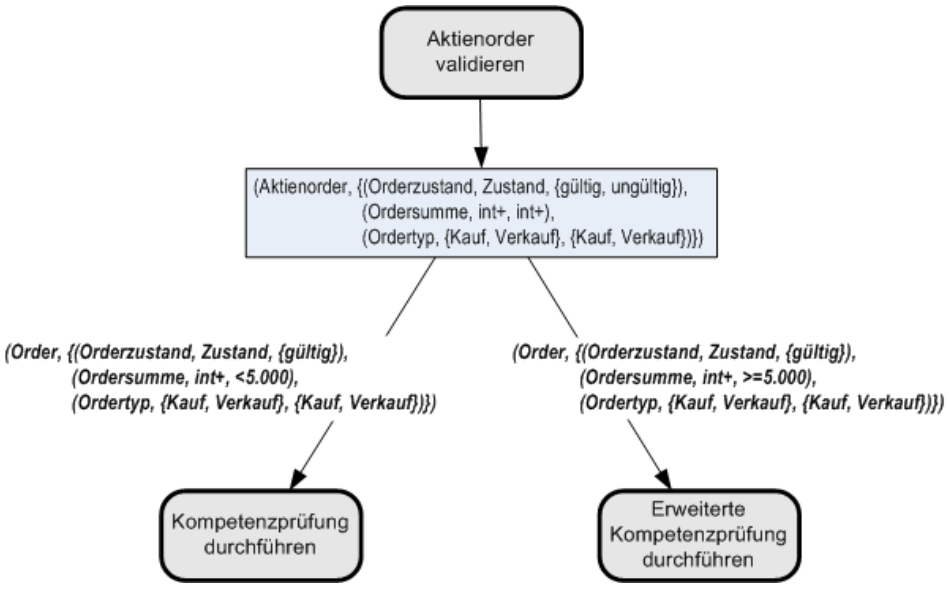


Bild 5 Ausschnitt zur Veranschaulichung des AAG

Zur algorithmischen Erstellung des AAG werden zunächst diejenigen PA ermittelt und dem Graph hinzugefügt, die mindestens einen Outputparameter liefern, der mit einem Parameter der Zielzustände *Goals* (vollständig oder partiell) übereinstimmt. Danach werden die von diesen PA benötigten Inputparameter als Knoten in den Graph aufgenommen. Iterativ werden nunmehr alle wei-

teren PA bestimmt und dem AAG inklusive ihres Inputs hinzugefügt, die mindestens einen Outputparameter liefern, der (auf Basis des Inferenzmechanismus) mit einem dieser benötigten Inputparameter (vollständig oder partiell) übereinstimmt. Dabei muss diese PA nicht notwendigerweise alle benötigten Inputparameter einer bereits im AAG vorhandenen PA als Output liefern. Sobald mindestens ein Outputparameter als einer der benötigten Inputparameter verwendet werden kann, besteht eine (partielle oder vollständige) Abhängigkeit und die liefernde PA wird dem AAG hinzugefügt. Weitere benötigte Inputparameter müssen dann von weiteren PA als Output zur Verfügung gestellt werden. Somit sind letztlich alle PA Teil des AAG, die potenziell Teil einer zulässigen Lösung sein können. Solche PA, welche Outputparameter liefern, die mit keinem der im AAG enthaltenen Inputparameter übereinstimmen und somit nicht Teil einer zulässigen Lösung sein können, werden nicht berücksichtigt. Die Erstellung des AAG terminiert entweder, wenn alle notwendigen Inputs der PA im Graph bereitgestellt werden bzw. im Initialzustand *Init* bereits vorliegen (da eine endliche Anzahl an PA vorhanden ist und jede PA maximal einmal im AAG aufgenommen werden kann – selbst wenn die PA später im Prozessmodell mehrmals verwendet wird –, ist eine Terminierung hier gewährleistet). Oder die Erstellung des AAG terminiert, falls ein benötigter Inputparameter weder von *Init* noch von einer anderen PA geliefert werden kann. In diesem Fall ist eine Interaktion mit dem Prozessplaner (im Sinne einer teilautomatisierten Planung) erforderlich. Dieser kann entweder *Init* um fehlende Inputparameter ergänzen oder die Prozessbibliothek erweitern und anschließend mit der Erstellung des AAG fortfahren.
Mit der Analyse der vollständigen und partiellen Abhängigkeiten lassen sich gerade im Vergleich zu den in Abschnitt 2.3 genannten Ansätzen insbesondere das nichtdeterministische Planen und die Erstellung von Prozessmodellen mit Kontrollflussstrukturen erst durchführen (siehe Abschnitt 3.4). Zudem ist es nunmehr möglich, auch Vorgänger-Nachfolger-Beziehungen zwischen PA mit zusammengesetzten Parametern zu ermitteln. Interessant ist außerdem, dass einmal identifizierte partielle und vollständige Abhängigkeiten zwischen PA auch bei weiteren Problemstellungen wiederverwendet werden können, solange sich die Parameter dieser PA nicht verändern. Konkret bedeutet dies: Sind zwei oder mehrere PA in unterschiedlichen Problemstellungen zu berücksichtigen, so brauchen die Abhängigkeiten (inkl. des Reasonings) nur einmal ermittelt werden. Sie können dann in mehrere AAG übernommen werden bzw. es wird auf gespeicherte Teile eines früheren AAG zurückgegriffen. Dadurch wird in der Anwendung, in welcher der Ansatz mit einer Vielzahl von PA und Abhängigkeiten konfrontiert ist, die Laufzeit verkürzt (so konnten bspw. bei einem anderen Prozessmodell „Abwicklung außerbörslicher Orders“ knapp die Hälfte der bereits im hier skizzierten Beispiel identifizierten Vorgänger-Nachfolger-Beziehungen ebenfalls Verwendung finden).

3.3 Nutzung der Abhängigkeiten für die Bestimmung der Ablaufreihenfolgen

Im **zweiten Schritt** wird der AZG in einer Vorwärtstraversierung erstellt, d. h. beginnend mit dem Initialzustand wird anhand der im AAG abgebildeten Abhängigkeiten ermittelt, welche PA in einem Zustand jeweils ausführbar sind. Jede Einplanung einer PA führt zu einem neuen Zustand.

Durch die Nutzung des AAG kann der Suchraum erheblich reduziert werden, da nun nicht mehr alle in der Prozessbibliothek verfügbaren PA betrachtet werden müssen.

Planungsverfahren berücksichtigen eine PA in einem Zustand dann, wenn alle für diese PA spezifizierten Bedingungen in dem Zustand zutreffen. Wir planen eine PA dagegen ein, sobald die Restriktionen der im Zustand existierenden Parameter eine Durchführung grundsätzlich zulassen (Zulässigkeitsbedingung). Dies trägt wesentlich zur Reduzierung der Suchzeit bei, da Zustände nicht in alle möglichen Teilzustände zerlegt werden müssen. Zudem werden Endlosschleifen erkannt und verhindert, die durch die wiederholte Einplanung von PA entstehen können (vgl. auch Bertoli et al. 2001).

Die genaue Bestimmung der Ablaufreihenfolge soll in diesem Beitrag aus Platzgründen nicht weiter formalisiert werden. Die zugrunde liegende Idee wird vielmehr direkt am Beispiel beschrieben. Bild 6 stellt dar, wie der AZG sukzessive erstellt wird (aus Gründen der Übersichtlichkeit sind nicht alle Zustände ausführlich beschrieben).

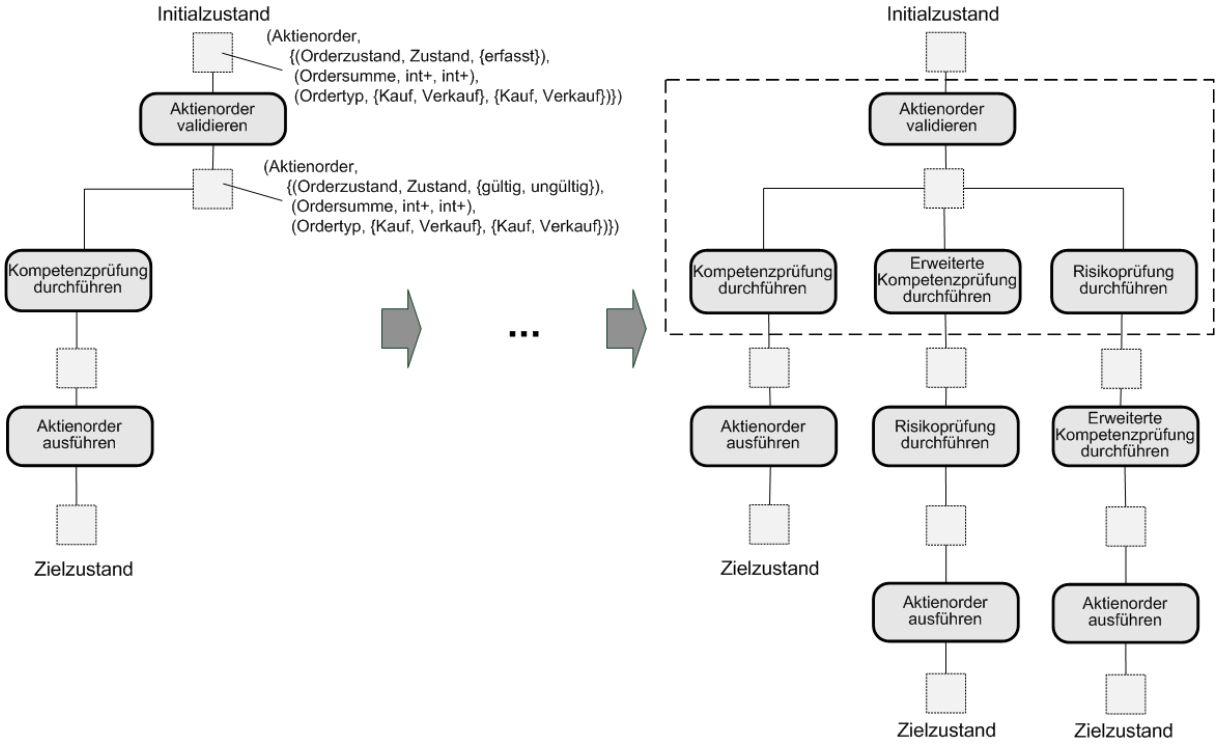


Bild 6 Erstellung des AZG

Im Initialzustand kann laut AAG nur die PA *Aktienorder validieren* ausgeführt werden. Dies führt zu dem in Bild 6 dargestellten Zustand, bei dem der *Orderzustand* entweder *gültig* oder *ungültig*, die *Ordersumme* ein *positiveInteger*-Wert und der *Ordertyp* ein Element aus der Menge {*Kauf, Verkauf*} ist. Danach wird mittels des AAG zunächst genau eine PA ermittelt, die im Zustand ausführbar ist. Im Beispiel ist dies die PA *Kompetenzprüfung durchführen*, die als Knoten dem AZG hinzugefügt wird und wiederum zu einem neuen Zustand führt. Die einzige PA, die danach ausführbar ist, entspricht der PA *Aktienorder ausführen*, die schließlich zu einem der gegebenen Zielzustände führt. Danach erfolgt im AZG ein Backtracking, bis ein Zustand erreicht wird, bei

dem die Einplanung bisher noch nicht analysierter PA zulässig ist (existiert kein derartiger Zustand, so sind alle Ablaufreihenfolgen im AZG bereits ermittelt und der Algorithmus beendet die Suche). Im Beispiel trifft dies auf den Zustand nach der PA *Aktienorder validieren* zu, d. h. hier sind neben der bereits identifizierten PA *Kompetenzprüfung durchführen* auch die PA *Erweiterte Kompetenzprüfung durchführen* und *Risikoprüfung durchführen* zulässig. Auch diese Zweige im Graph werden wie in Bild 6 dargestellt bis zum Erreichen eines Zielzustands geplant. Schon in diesem kleinen Beispiel zeigt sich der Vorteil des AAG, da notwendige Analysen von Vorgänger-Nachfolger-Beziehungen gleicher PA (für die verschiedenen Äste des AZG) nicht mehrmalig erfolgen müssen.

3.4 Erstellung der Prozessmodelle

Im **dritten Schritt** prüft SEMPA die im AZG dargestellten Zustände, leitet daraus ab, welche Kontrollflussstrukturen vorliegen und erzeugt schließlich die Prozessmodelle: Die derzeitige Lösung erstellt UML-Aktivitätsdiagramme, die eine anschauliche Darstellungsweise für Prozesse mit einer für die Planung ausreichenden formalen Definition (zumindest ab der Version 2.0) verbinden. Zudem unterstützen sie alle in Anforderung (R7) genannten Kontrollflussstrukturen (Russell et al. 2006).

Folgen zwei PA im AZG aufeinander und hat die zweite PA nur vollständige Abhängigkeiten bzgl. der im vorhergehenden Zustand verfügbaren Parameter, so liegt eindeutig eine Sequenz vor. Nach der ersten PA kann immer auch die zweite PA ausgeführt werden. Dagegen ist ein Exclusive choice im Prozessmodell erforderlich, wenn partielle Abhängigkeiten (obige Fälle [1.2] und [2.1]) auftreten. Hier ist zu prüfen, für welche Wertebereiche der Parameter eines Zustands eine PA ausgeführt werden kann und vice versa. Dies kann erfolgen, indem sukzessive die Restriktionen der Parameter in Partitionen im mathematischen Sinn (d. h. disjunkte mögliche Zustände) zerlegt werden. Die damit ermittelten Wertebereiche können für die Konstruktion von Exclusive choices verwendet werden.

Um die Idee beispielhaft zu verdeutlichen, betrachten wir den markierten, gestrichelten Bereich in Bild 6. In diesem sind die PA *Aktienorder validieren,* der nachfolgende Zustand, sowie die in diesem Zustand u. a. ausführbaren PA *Kompetenzprüfung durchführen, Erweiterte Kompetenzprüfung durchführen* und *Risikoprüfung durchführen* dargestellt. Wie in Abschnitt 3.2 für die PA *Kompetenzprüfung durchführen* verdeutlicht, können die PA jedoch nur bei bestimmten Parameterwerten der im Zustand vorliegenden *Order* auch tatsächlich ausgeführt werden (partielle Abhängigkeit). Dementsprechend konstruiert SEMPA an dieser Stelle zwei Exclusive choices. Ein Exclusive choice unterscheidet den *Orderzustand* hinsichtlich *gültig* bzw. *ungültig*. Ist die Order *ungültig* führt dies dazu, dass - mangels in diesem *Orderzustand* ausführbarer PA - direkt zum Endknoten des Aktivitätsdiagramms verzweigt wird. Der zweite Exclusive choice unterscheidet die Fälle *Ordersumme* größer oder gleich 5.000 und *Ordersumme* kleiner 5.000.

Für die Konstruktion von Parallel splits (und zugehöriger Synchronizations) werden die Pfade im AZG analysiert. Bspw. ist eine Parallelisierung zweier Prozessaktionen immer dann möglich,

wenn zwei Prozessaktionen voneinander unabhängig sind und daher im AZG zwei Pfade existieren, die sich lediglich in der Reihenfolge der beiden Prozessaktionen unterscheiden (wie auch im vorliegenden Beispiel). Die Konstruktion von Parallel splits hat Ähnlichkeiten mit der Reorganisation von Plänen, die im Bereich der AI Planung untersucht wird. Die Frage ist hier, wie ein sequenzieller Plan relaxiert werden kann, so dass in Teilen eine parallele Ausführung von Aktionen möglich ist (vgl. bspw. Bäckström 1998). Auch hier besteht eine wichtige Erweiterung unseres Ansatzes darin, dass semantische Relationen sowie Parameter mit unterschiedlichen Datentypen betrachtet werden, während existierende Ansätze i. A. nur einfache, nicht-typisierte Symbole (im Sinne der Aussagenlogik) verarbeiten können.

Simple merges schließlich werden ausgehend vom Prozessende konstruiert, indem Exclusive choices systematisch wieder zusammengeführt werden. Zum Einsatz kommt hier ein Algorithmus, der den Fluss eines (hypothetischen) Tokens durch das Aktivitätsdiagramm nachvollzieht: Zunächst wird der Graph vom Initialzustand bis zu den Zielzuständen durchlaufen. Auf dem Weg liegende Exclusive choices werden an den Knoten „protokolliert“, d. h. in einer geordneten Menge gespeichert. Der Weg den ein (hypothetisches) Token im Graphen genommen hat, ist anhand dieser Menge definiert. Anschließend prüft der Algorithmus beginnend mit den Zielzuständen, ob verschiedene Pfade im AZG durch Simple merges zusammengeführt werden können. Dabei kann eine korrekte „Schachtelung“ der Choices und Merges durch Überprüfung des gespeicherten „Token-Flusses“ sichergestellt werden.

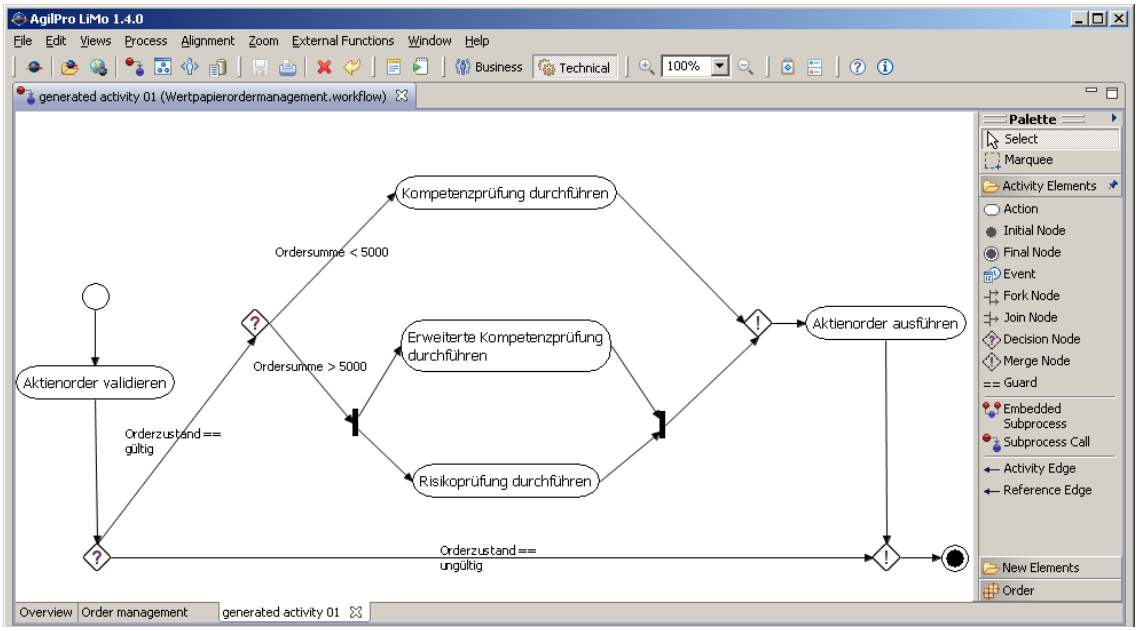


Bild 7 Darstellung einer zulässigen Lösung

Zusammenfassend werden im dritten Schritt die Kontrollflussstrukturen konstruiert und durch die entsprechenden Elemente der UML-Notation (z. B. Entscheidungsknoten für Exclusive choices) abgebildet. Aktuell werden dabei Exclusive choice, Parallel split, Synchronization und Simple merge erkannt. Bild 7 zeigt *eine zulässige Lösung*. Im Rahmen des betrachteten Prozessausschnittes geht dabei ein Wertpapierauftrag ein und wird einer Validierung unterzogen. Wird die

Ungültigkeit der Order festgestellt, ist der Prozess abzubrechen. Bei einer gültigen Order wird unterschieden, ob die Ordersumme kleiner als 5.000 ist oder vice versa. Bei einer Ordersumme kleiner als 5.000 erfolgt eine einfache Kompetenzprüfung, anderenfalls eine erweiterte Prüfung und eine Risikoprüfung. Sind alle Überprüfungen abgeschlossen, ist die Order auszuführen.

4 Prototypische Umsetzung und Validierung

Der SEMPA-Planer wurde prototypisch im Rahmen des Open-Source-Prozessmodellierungstools AgilPro (Bauer et al. 2007) umgesetzt. Mittels AgilPro können Prozesse modelliert, in verschiedenen Sichten dargestellt und simuliert werden. Die Spezifikation der Ontologie erfolgte dabei in OWL 2.0 (Motik et al. 2008). Obwohl der OWL 2.0-Entwurf erst Anfang 2008 beim W3C eingereicht wurde und eine Toolunterstützung in der Breite nicht gegeben ist, wird es infolge der größeren Mächtigkeit immer stärker eingesetzt. Für die Integration semantischer Informationen und zur Verarbeitung der OWL-Ontologie wird in SEMPA die OWL API (Bechhofer et al. 2003) verwendet. Als Inferenzmaschine wird Pellet (Sirin et al. 2007) genutzt. Pellet führt ein Reasoning auf einer Ontologie durch, welche die Konzepte der Domäne in Beschreibungslogik abbildet. Die OWL API ist ein Java-Framework, das sowohl OWL Lite als auch OWL DL unterstützt und eine Schnittstelle zu Inferenzmaschinen wie Pellet bietet. Durch diese Schnittstelle ist es ohne Probleme möglich, jederzeit zu einer anderen Inferenzmaschine zu wechseln. Neben den in Abschnitt 3.1 bereits aufgeführten Designentscheidungen wurde auch bei der prototypischen Implementierung deutlich, dass eine Unterteilung von SEMPA in drei Schritte auch hinsichtlich Wartung und Pflege sinnvoll und erforderlich ist. Die prototypische Implementierung sowie die geplanten Prozessmodelle wurden zudem hinsichtlich folgender Punkte analysiert:

a) *Analytische Prüfung:* Hier wurden die zugrunde liegenden Algorithmen bezüglich der Kriterien Vollständigkeit (alle zulässigen Lösungen werden auch ermittelt), Minimalität (es werden bspw. nur diejenigen PA in den erstellten Graphen berücksichtigt, die auch Teil einer zulässigen Lösung sein können), Terminierung (bei der Erstellung des AAG, des AZG und des Prozessmodells terminieren alle drei Teile des Ansatzes) und Zeitkomplexität (wie verändert sich die Laufzeit bei Veränderung jedes einzelnen Inputparameters des Modells/Algorithmus) untersucht. Es wurde dabei gezeigt, dass der Ansatz zu vollständigen Ergebnissen führt, im Schritt eins bedingt minimal (PA, die im ersten Schritt im AAG noch berücksichtigt werden, können jedoch bspw. aufgrund rekursiver Abhängigkeiten notwendiger Input-/Outputparameter nicht Teil einer zulässigen Lösung sein und werden erst in Schritt zwei – wenn die Reihenfolge der PA geplant wird und die Rekursivitäten transparent werden – eliminiert) aber in den Schritten zwei und drei minimal ist und in allen drei Teilen terminiert. Bzgl. der Zeitkomplexität gilt bspw., dass sich bei einer Erweiterung des Initialzustands bzw. der Zielzustände die Laufzeit – u. a. wegen der Designentscheidungen (vgl. Abschnitt 3.1 und 3.2) – unterproportional entwickelt. Bei einer Erhö-

hung der Anzahl der semantischen Relationen steigt die Laufzeit ebenso nur unterproportional an.

b) *Prüfung der Implementierung des Ansatzes:* Hier wurde die korrekte Umsetzung des Ansatzes untersucht. Neben der manuellen Prüfung des Sourcecode (structured walk through) durch Personen, die von den Programmierern verschieden waren, wurden eine Reihe von Tests mit dem JUnit-Framework durchgeführt. Hierzu gehören Testläufe mit Extremwerten, JUnit-Regressionstests, Unit-Tests (Modultests der einzelnen Schritte, d. h. zur Erstellung des AAG, des AZG und des Aktivitätsdiagramms) sowie Integrationstests der verschiedenen Module. Die implementierten Algorithmen wiesen nach Abschluss der Testarbeiten keine Fehler mehr auf.

c) *Formale Validierung der Ergebnisse:* Es lässt sich zeigen, dass die durch SEMPA erstellten Prozessmodelle (syntaktisch) korrekt (vgl. hierzu Weske 2007, van der Aalst 200, Sun et al. 2006) sind. Bspw. sind die Prozessmodelle frei von strukturellen Fehlern. So können keine unverbundenen Aktionen auftreten, da sich in Folge der Planung alle Aktionen auf dem Pfad vom (einzigen) Startknoten zum Endknoten befinden (siehe hierzu Weske 2007, S. 267). Gleichzeitig sind so genannte Datenanomalien (Sun et al. 2006) ausgeschlossen (d. h. eine Prozessaktion kann bspw. aufgrund fehlendem Inputs nicht ausgeführt werden, obwohl diese im Prozessmodell vorgesehen ist), da die Analyse der Abhängigkeiten aufgrund von Input und Outputparametern gerade Ausgangspunkt für die Konstruktion der Prozessmodelle ist.

d) *Operationelle Prüfung der Ergebnisse:* Hier wurden zwei Sachverhalte analysiert. Zum einen, inwieweit die definierten Anforderungen (bspw. Erstellung von Prozessmodellen mit Kontrollflussstrukturen wie einem Exclusive choice) erfüllt werden und sich auch in den Ergebnissen (resultierende Graphen bzw. Aktivitätsdiagramme) widerspiegeln. Zum anderen, inwieweit die Ergebnisse der teilautomatisierten Planung von Prozessmodellen mit (bisher vom FDL) manuell erstellten Prozessmodellen übereinstimmen bzw. diese verbessern. Mit einigen der bisherigen Problemstellungen wie bspw. dem Ordermanagement – dieses beinhaltet den im Beitrag betrachteten Prozess für die Verarbeitung von Orders – konnte dies durchgeführt werden. Hier entsprach das manuell erstellte Prozessmodell des FDL einer zulässigen Lösung (es kam nur zu Abweichungen im Layout der Prozesse). Darüber hinaus wurden einzelne zulässige Lösungen (in Form eines Aktivitätsdiagramms) mit Prozessmodellierern des FDL diskutiert, um bspw. eine mögliche Parallelisierung von Ablaufzweigen – die in zulässigen Lösungen vorgeschlagen wurde – umzusetzen.

5 Zusammenfassung und Ausblick

Im Beitrag wurden Teile des SEMPA-Ansatzes zur Planung von Prozessmodellen vorgestellt und anhand eines Beispielprozesses *Verarbeitung von Aktienorders* veranschaulicht. Ziel ist es dabei, den manuellen Modellierungsaufwand zu reduzieren. Die Ergebnisse lassen sich wie folgt zusammenfassen:

1. Der Beitrag beschreibt, wie PA semantisch ausgezeichnet werden können und berücksichtigt dabei Anforderungen, wie bspw. die Abbildung komplexer Parameter oder Restriktionen (Anforderungen R3, R4). Aus einer semantischen Analyse der Parameter können vollständige und partielle Abhängigkeiten zwischen PA abgeleitet werden (Anforderung R6), anhand derer die Planung von Prozessmodellen mit Kontrollflussstrukturen möglich ist. Besonders geeignet erscheint der Ansatz dabei für Domänen mit Problemstellungen, die keinen hohen Kreativitätsanteil im Prozessergebnis besitzen, die öfters Änderungen unterworfen sind und deren begriffliche Abstimmung zeitaufwändig ist (bereichsübergreifende Prozesse).
2. Der SEMPA-Planer bietet zahlreiche Erweiterungen gegenüber bisherigen Ansätzen. Der Planer konstruiert syntaktisch korrekte (Eshuis 2006) Aktivitätsdiagramme mit den Kontrollflussstrukturen Exclusive choice, Simple merge, Parallel split und Synchronization (Anforderung R7). Dabei ist es besonders wichtig, mehrere Zielzustände berücksichtigen zu können und damit auch alternative Prozesspfade zu erhalten. Diese Modelle sowie die semantischen Auszeichnungen der enthaltenen PA können die Basis sein, um geeignete Webservices zu identifizieren (Ang et al. 2006, Drumm et al. 2006). Dies unterstreicht den Vorteil einer im ersten Schritt technologieunabhängigen Planung von Prozessmodellen im Vergleich bspw. zur Webservice-Komposition: So ist insbesondere bei geschäftskritischen Prozessen nicht davon auszugehen, dass eine automatisierte Planung (direkt auf Workflow-Ebene) und direkte Ausführung durch Webservices ohne vorherige Abstimmung bspw. mit den beteiligten Fachbereichen erfolgen kann und soll. Demgegenüber kann durch die vorgestellte Planung das erstellte Prozessmodell als Zwischenschritt (im Sinne einer Entscheidungsvorlage) in einer verständlichen Darstellung abgestimmt werden. Auch kann auf Basis der zu den einzelnen PA verfügbaren Webservices eine Auswahl der unter ökonomischen Gesichtspunkten geeigneten Webservices nicht nur für eine PA, sondern auch für den gesamten Prozess *zur Laufzeit* (um bspw. kurzfristige Änderungen in der Verfügbarkeit benötigter Webservices zu berücksichtigen) erfolgen.
3. Ein weiterer Punkt ist die Generierung der Menge zulässiger Lösungen für eine definierte Problemstellung (Anforderung R8). Sie ist vom vorgenannten Punkt abzugrenzen, da nunmehr für einen Zielzustand alternative Prozessabläufe erstellt werden. Hierdurch wird es möglich zu untersuchen, welche Lösungen davon unter ökonomischen Aspekten gewählt werden sollen. Dies soll in einem nachgelagerten Schritt erfolgen, um zur Laufzeit die jeweils beste Realisierungsoption (bspw. welche Webservices sind bei welcher zulässigen Lösung verfügbar und wie sind diese bewertet) zu wählen.

Daneben sind auch kritische Punkte zu diskutieren, die den weiteren Forschungsbedarf definieren: Die beschriebenen Möglichkeiten der semantischen Analyse erlauben bereits die Abbildung von Relationen zwischen unterschiedlichen Begriffen in einer Ontologie. Bisher wird jedoch für die Planung von genau einer konsistenten Ontologie ausgegangen. Großes Potenzial liegt in der Erweiterung von SEMPA um Ansätze zur Zusammenführung verschiedener Ontologien, um die

Problematik unterschiedlicher Terminologien (welche die Anpassung und Weiterentwicklung von Prozessmodellen erschwert) noch stärker zu berücksichtigen. Ein weiterer Punkt ist der auf den ersten Blick sehr hoch erscheinende Aufwand für die Erstellung von Ontologien und der semantischen Auszeichnung von PA. Eine Mehrfachverwendung der PA nicht nur innerhalb eines Modells sondern bei verschiedenen Problemstellungen relativiert allerdings diesen Aufwand. Außerdem erscheint im Gegensatz zum bisherigen, hohen manuellen Wartungs- und Abstimmungsaufwand für Prozessmodelle durch die semantische Annotation eine einfachere Anpassbarkeit verwendeter PA möglich (entspricht gerade der Kernthese des Semantischen Prozessmanagements). So wird durch die semantische Auszeichnung Wissen über Prozessbestandteile expliziert, das bislang teilweise nur bei menschlichen Wissensträgern zur Verfügung stand und so bspw. bei jeder Anpassung dessen Interaktion notwendig machte. Um dies zu verbessern, gilt es neben dem Zurückgreifen auf bestehende Ontologien auch diesbezüglich vorhandene Werkzeuge und Methoden für SEMPA zu adaptieren. Ferner berücksichtigt der Ansatz derzeit noch nicht alle Kontrollflussstrukturen. Deshalb wird derzeit an der Berücksichtigung komplexer Prozessschleifen (Arbitrary cycles) gearbeitet. Hier können bereits vorhandene Informationen im AAG verwendet werden. Zudem soll die Möglichkeit ausgebaut werden, aus der Gesamtheit aller Lösungen einzelne zulässige Lösungen als Aktivitätsdiagramm zu extrahieren. Hier wird derzeit ein heuristisches Verfahren entwickelt, das im AZG diejenigen Zustände zuerst analysiert, die bereits eine Schnittmenge zu den geforderten Zielzuständen besitzen. Der entwickelte Ansatz bietet hierfür jeweils eine geeignete Basis.

Literatur

Ang, Cheng-Leong; Gu, Yuan; Sourina, Olga; Leng Gay, Robert Kheng (2005): An ARIS-based Transformation Approach to Semantic Web Service Development. In: International Conference on Cyberworlds. CW 2005, Singapur, November 2005, S. 297-306.

Bäckström, Christer (1998): Computational Aspects of Reordering Plans. In: Journal of Artificial Intelligence Research 9, S. 99-137.

Balzer, Robert (1985): A 15 Year Perspective On Automatic Programming. In: IEEE Transactions on Software Engineering. Vol. SE-11, No. 11, S. 1257-1268.

Bauer, Bernhard; Lautenbacher, Florian; Palfinger, Günther; Roser, Stephan (2007): „AgilPro": Modellierung, Simulation und Ausführung agiler Prozesse. In: ObjektSPEKTRUM 1/2007, S. 52-59.

Bechhofer, Sean; Lord, Phillip; Volz, Raphael (2003): Cooking the Semantic Web with the OWL API. In: Fensel, Dieter; Sycara, Katja; Mylopoulos, John (Hrsg.): ISWC 2003, Florida, USA, October 2003, S. 659-675.

Becker, Jörg; Kahn, Dieter (2003): The Process in Focus. In: Becker, Jörg; Kugeler, Martin; Rosemann, Michael (Hrsg.): Process Management. A Guide for the Design of Business Processes. Springer, Berlin u. a. 2003, S. 1–12.

Bertoli, Piergorgio; Cimatti, Alessandro; Roveri, Marco; Traverso, Paolo (2001): Planning in Nondeterministic Domains under Partial Observability via Symbolic Model checking. In: IJCAI 2001, AAAI Press, Seattle, Washington, USA, August 2001, S. 473-478.

Betz, Stephanie; Klink, Stefan; Koschmider, Agnes; Oberweis, Andreas (2006): Automatic User Support for Business Process Modeling. In: Hinkelmann, Knut; Karagiannis, Dimitris; Stojanovic, Nenad; Wagner, Gerd (Hrsg.): Workshop on Semantics for Business Process Management. ESWC 2006, Budva, Montenegro, S. 1-12.

Biron, Paul V.; Malhotra, Ashok (2004): XML Schema Part 2: Datatypes Second Edition. http://www.w3.org/TR/xmlschema-2, Abruf am 2007-07-11.

Brockmans, Saartje; Ehrig, Marc; Koschmider, Agnes; Oberweis, Andreas; Studer, Rudi (2006): Semantic Alignment of Business Processes. In: Manolopoulos, Yannis; Filipe, Joaquim; Constantopoulos, Panos; Cordeiro, José (Hrsg.): ICEIS 2006, Paphos, Zypern, Mai 2006, S. 191-196.

Cimatti, Alessandro; Roveri, Marco; Traverso, Paolo (1998): Automatic OBDD-based Generation of Universal Plans in Non-Deterministic Domains. In: AAAI/IAAI. S. 875-881.

Constantinescu, Ion; Faltings, Boi; Binder, Walter (2004): Large scale, type-compatible service composition. In: ICWS 2004, S. 506-513.

Drumm, C.; Lemcke, J.; Namiri, K. (2006): Integrating Semantic Web Services and Business Process Management: A Real Use Case. In: Workshop on Semantics for Business Process Management at the 3rd European Semantic Web Conference.

Eshuis, Rik (2006): Symbolic model checking of UML activity diagrams. In: ACM Transactions on Software Egineering and Methodology, 15 (1), S. 1-38.

Ferstl, Otto K.; Sinz, Elmar J. (2001): Grundlagen der Wirtschaftsinformatik, 4. Aufl., Oldenbourg, München, Wien.

Ghallab, Malik; Nau, Dana; Traverso, Paolo (2004): Automated Planning. Elsevier, San Francisco.

Hepp, Martin; Leymann, Frank; Domingue, John; Wahler, Alexander; Fensel, Dieter (2005): Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management. In: Tsai, Wei-Tek; Chung, Jen-Yao; Younas, Muhammad (Hrsg): IEEE ICEBE 2005, October 18-20, Beijing, China, S. 535-540.

Hepp, Martin; Roman, Dumitri (2007): An Ontology Framework for Semantic Business Process Management. In: Oberweis, Andreas; Weinhardt, Christof; Gimpel, Henner; Koschmider, Agnes; Pankratius, Victor; Schnizler, Björn (Hrsg.): 8. Internationale Tagung Wirtschaftsinformatik, Universitätsverlag Karlsruhe, S. 423-440.

Hornung, Thomas; Koschmider, Agnes; Oberweis, Andreas (2007): Rule-based Autocompletion of Business Process Models. In: CAiSE 2007, June 11-15, 2007, Trondheim, Norway.

Kabanza, Froduald; Barbeau, Michel; St-Denis, Richard (1997): Planning Control Rules for Reactive Agents. In: Artifi-cial Intelligence 95 (2), S. 409-438.

Kalfoglou, Yannis; Schorlemmer, Marco (2003): Ontology Mapping: The State of the Art. In: The Knowledge Engineering Review Journal, 18 (1), S. 1-31.

Kugeler, Martin; Rosemann, Michael (1998): Fachbegriffsmodellierung für betriebliche Informationssysteme und zur Unterstützung der Unternehmenskommunikation. In: Informationssystem-Architekturen. Hrsg.: GI-Fachgruppe 5.2, S. 8-15.

Kuropka, Dominik; Weske, Mathias (2008): Implementing a Semantic Service Provision Platform – Concepts and Experiences. In: Wirtschaftsinformatik 50 (1), S. 16-24.

Lang, Qianhui Althea; Su, Stanley Y. W. (2005): AND/OR Graph and Search Algorithm for Discovering Composite Web Services. In: Intl. Journal of Web Services Research, 2 (4), S. 46-64.

McIlraith, Sheila; Son, Tran Cao (2002): Adapting Golog for Composition of Semantic Web Services. In: Fensel, Dieter; Giunchiglia, Fausto; McGuiness, Deborah L.; Williams, Mary-Anne (Hrsg.): KR 2002, April 2002, S. 482-493.

Meyer, Harald; Weske, Mathias (2006): Automated Service Composition using Heuristic Search. Eder, Johann; Dustdar, Schahram (Hrsg.): BPM 2006, Wien, Österreich, LNCS 4102, S. 81-96.

Meyer, Harald; Kuropka, Dominik (2006): Requirements for Service Composition. In: Business Process Management Workshops, LNCS, Springer, Vol. 4103, 2006, S. 439-450.

Minkwitz, Torsten (1993): Algorithmensynthese für lineare Systeme mit Symmetrie. Doktorarbeit, Universität Karlsruhe, Informatik.

Motik, Boris; Patel-Schneider, Peter F.; Horrocks, Ian (2008): OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Working Draft, April 2008. http://www.w3.org/TR/owl2-syntax/, Abruf am 2008-09-22.

Pathak, Jyotishman; Basu, Samik; Honavar, Vasant (2006a): Modeling Web Services by Iterative Reformulation of Functional and Non-Functional Requirements. In: Dan, Asit; Lamersdorf, Winfried (Hrsg.): ICSOC 2006, Chicago, USA, 2006, S. 314-326.

Pathak, Jyotishman; Basu, Samik; Honavar, Vasant (2006b): Modeling Web Service Composition using Symbolic Transition Systems. AAAI Workshop on AI-Driven Technologies for Service-Oriented Computing 2006, AAAI Press, California, USA, 2006, S. 44-51.

Paolucci, Massimo; Kawamura, Takahiro; Payne, Terry R.; Sycara, Katia (2002): Semantic Matching of Web Services Capabilities. In: ISWC 2002, Sardinia, Italy.

Pistore, Marco; Traverso, Paolo; Bertoli, Piergorgio; Marconi, Annapaola (2005): Automated Synthesis of Composite BPEL4WS Web Services. In: ICWS 2005, IEEE Press, S. 293–301.

Rao, Jinghai; Dimitrov, Dimitar; Hofmann, Paul; Saded, Norman (2006): A Mixed Initiative Semantic Web Framework for Process Composition. In: ISWC 2006, Athens, GA, USA, November 5-9, 2006.

Russell, Nick; van der Aalst, Will M.P.; ter Hofstede, Arthur H.M.; Wohed, Petia (2006): On the Suitability of UML 2.0 Activity Diagrams for Business Process Modeling. In: Roddick, John F.; Hinze, Annika (Hrsg.): APCCM 2006, Hobart, Australia.

Russell, Stuart J.; Norvig, Peter (2004): Künstliche Intelligenz. Ein moderner Ansatz. 2. Aufl., Pearson.

Scheer, August-Wilhelm (1991): Architektur integrierter Informationssysteme – Grundlagen der Unternehmensmodellierung. Springer-Verlag, Berlin.

Sirin, Evren; Parsia, Bijan; Grau, Bernardo Cuenca; Kalyanpur, Aditya; Katz, Yarden (2005): Pellet: A practical OWL-DL reasoner. UMIACS Technical Report, 2005-68. http://pellet.owldl.com/papers/sirin05pellet.pdf, Abruf am 2007-09-12.

Sirin, Evren; Parsia, Bijan; Grau, Bernardo Cuenca; Kalyanput, Aditya; Katz, Yarden (2007): Pellet: A practical OWL-DL reasoner. In: Journal of Web Semantics, 2007, S. 51-53.

Sirin, Evren; Parsia, Bijan; Wu, Dan; Hendler, James; Nau, Dana (2004): HTN Planning for Web Service Composition Using SHOP2. http://www.mindswap.org/papers/SHOP-JWS.pdf, Abruf am 2007-04-20.

Sun, Sherry X.; Zhao, J. Leon; Nunamaker, Jay F.; Sheng, Olivia R. L. (2006): Formulating the Data-Flow Perspective for Business Process Management. In: Information Systems Research 17 (4), S. 374-391.

ter Beek, Maurice; Bucchiarone, Antonio; Gnesi, Stefania (2006): A Survey on Service Composition Approaches: From Industrial Standards to Formal Methods. Technical report, 2006-15. http://dienst.isti.cnr.it/Dienst/UI/2.0/Describe/ercim.cnr.isti/2006-TR-15, Abruf am 2007-01-16.

Thomas, Oliver; Fellmann, Michael (2006): Semantische Integration von Ontologien und Ereignisgesteuerten Prozessketten. In: Nüttgens, Markus; Rump, Frank; Mendling, Jan (Hrsg.): Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, EPK 2006, Wien, S. 7-23.

van der Aalst, Wil M. P. (2000): Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques. In: Business Process Management, S. 161-183

van der Aalst, Wil M. P.; ter Hofstede, Arthur H.M.; Kiepuszewski, Bartek; Barros, Alistair P. (2003): Workflow patterns. In: Distributed and Parallel Databases 14 (3), S. 5-51.

van der Aalst, Wil M. P.; Günther ,Christian; Recker, Jan C.; Reichert, Manfred (2006): Using Process Mining to Analyze and Improve Process Flexibility. In: BPMDS 2006, 5 – 6 June 2006, Luxembourg.

Weld, Daniel S.; Anderson, Corin R.; Smith, David E. (1998): Extending graphplan to handle uncertainty and sensing actions. In: AAAI98 and IAAI98. S. 26-30.

Weske, Mathias (2007): Business Process Management – Concepts, Languages, Architectures, Springer, Berlin Heidelberg.

Yan, Yuhong; Liang, Yong; Liang, Han (2006): Composing Business Processes with Partial Observable Problem Space in Web Services Environments. In: IEEE ICWS 2006, Chicago, S. 541-548.