Hasso-Plattner-Institut für Softwaresystemtechnik
Fachgebiet Business Process Technology

Integrierte Prozessplanung und -ausführung

Dissertation

**zur Erlangung des akademischen Grades
"doctor rerum naturalium"
(Dr. rer. nat.)**

**eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
der Universität Potsdam**

**von
Hilmar Schuschel**

Potsdam, den 10.11.2006

Danksagung

Die vorliegende Dissertation habe ich als wissenschaftlicher Mitarbeiter des Fachbereichs *Business Process Technology* von Herrn Prof. Dr. Mathias Weske am Hasso-Plattner-Institut für Softwaresystemtechnik an der Universität Potsdam angefertigt. Ich möchte mich an dieser Stelle herzlich bei allen Personen bedanken, die zum Gelingen dieser Arbeit beigetragen haben. Ich danke meinem Doktorvater Mathias Weske für die intensive Unterstützung in allen Phasen des Promotionsprozesses von der anfänglichen Themensuche und Einarbeitung bis hin zur abschließenden Disputation; und bei dieser Gelegenheit auch dafür, dass er bereits zur Zeiten meines Studiums in Münster mein Interesse für Forschung und Lehre geweckt hat und mich seitdem Schritt für Schritt an das wissenschaftliche Arbeiten herangeführt hat. Es war ein langer Weg von dem Projektseminar „Integration heterogener Datenbanksysteme durch CORBA-Technologie“ bis zur Verteidigung meiner Doktorarbeit. Vielen Dank auch an meinen langjährigen Freund und Kollegen Jens Hündling, mit dem ich studiert, gearbeitet und die letzten Jahre das Büro geteilt habe. Unsere unzähligen Diskussionen zu Fragestellungen aus dieser Arbeit haben einen großen Beitrag geliefert. In einem Prototyp namens Plaengine wurden viele der entwickelten Konzept softwaretechnisch realisiert. Ein großes Dankeschön für ihre Mitarbeit geht an die Projektbeteiligten Christian Günther, Harald Meyer und Hagen Overdick. Dank auch an alle meine Kollegen und Freunde, die mit Rat und Tat geholfen haben, insbesondere an Katrin Heinrich, Dominik Kuropka, Frank Puhlmann, Marc Nienhaus und Arnd Schnieders. Zu guter Letzt vielen Dank an meine Eltern Heinz und Hildegard für alles und an meine Frau Zlatina für ihre Unterstützung und große Geduld während der Anfertigung dieser Dissertation.

Potsdam, November 2006

Kurzfassung

Flexibilität und eine gute informationstechnische Unterstützung der Geschäftsprozesse sind für die Wettbewerbsfähigkeit von Unternehmen auf heutigen Märkten elementare Voraussetzungen. Workflow-Management-Systeme steuern die Ausführung von Geschäftsprozessen. Im Gegensatz dazu wird die Planung und Überwachung der Prozesse zwar durch Werkzeuge unterstützt, bleibt aber eine manuelle Aufgabe. Ein wichtiger Aspekt der Flexibilität im Kontext von Geschäftsprozessen ist die Fähigkeit, angemessen auf unerwartete Ereignisse während der Ausführung zu reagieren. Hierzu kann es notwendig sein, von der Ausführungsphase zurück in die Planungsphase zu wechseln. Dieser Wechsel von automatischer Ausführung zurück zu manueller Planung ist bei klassischen Workflow-Management-Systemen technisch und organisatorisch aufwendig und fehleranfällig.

In dieser Arbeit wird mit der Integration von Prozessplanung und -ausführung ein Konzept vorgestellt, das die oben beschriebenen Probleme klassischer Workflow-Management-Systeme löst. Die Idee hierbei ist die Automatisierung der Planung und darauf aufbauend die konzeptionelle und technische Integration von Prozessplanung und -ausführung. Anstelle einer manuellen Modellierung von Prozessdefinitionen, erfolgt eine formale, semantische Spezifikation der verfügbaren Aktivitäten sowie des zu bearbeitenden Geschäftsfalls. Hierdurch können Planungstechniken der Künstlichen Intelligenz eingesetzt werden, um automatisch für jeden Geschäftsfall eine passende Prozessdefinition zu planen. Es wird ein System modelliert, in dem Prozesse automatisch geplant, ausgeführt, überwacht und gegebenenfalls zur Laufzeit angepasst werden. Insbesondere werden hierbei auch Aktivitäten berücksichtigt, die sich bei der Anpassung eines Prozesses gerade in Ausführung befinden. Hierdurch wird eine Prozessanpassung möglich, ohne dass die Ausführung dafür unterbrochen werden muss. Ein weiterer Aspekt ist, dass Prozessdefinitionen nicht mehr im Voraus geplant werden, sondern erst wenn ein konkreter Geschäftsfall vorliegt. Folglich werden alle seine individuellen Besonderheiten automatisch berücksichtigt.

Aufbauend auf der informalen Konzeption eines Systems zur integrierten Prozessplanung und -ausführung wird ein formales Modell dieses Systems formuliert und untersucht. Neben der Präzisierung der vorgestellten Konzepte, erlaubt die Formalisierung den Nachweis einer Reihe von Systemeigenschaften. Es wird gezeigt, unter welchen Bedingungen der Zustand eines Geschäftsfalls durch Planung und Ausführung eines Prozesses in einen Zielzustand überführt wird und nachgewiesen, dass die spezifizierte Vorgehensweise eine konsistente Anpassung eines Prozesses zur Laufzeit ermöglicht. Schließlich wird die Validierung der Arbeit in Form einer prototypischen Realisierung beschrieben und damit die softwaretechnische Umsetzbarkeit der Integration von Techniken aus Workflow-Management und Planung gezeigt.

Abstract

Flexibility and good support for their business processes is crucial for the competitiveness of companies on today's markets. In this context workflow management systems support companies by controlling the execution of business processes. In contrast, modelling and monitoring of processes remain manual tasks, although they are supported by tools. An important aspect of flexibility in the context of business processes is the ability to react appropriately to unanticipated events that occur during process execution and endanger the achievement of the business goal. Therefore, it can be necessary to return from process execution to process planning. Due to the fact that workflow management systems only automate process execution, this procedure is costly and error-prone.

This thesis presents a concept for integrated process planning and execution that solves the shortcomings of workflow management systems mentioned above. The idea is to automate process planning and to technically integrate process planning with process execution. Instead of manually modelling processes, the available activities are specified semantically. This allows for the application of artificial intelligence planning techniques to plan an individual process for every single business case. Integrating these planning techniques with workflow management techniques, a system is modelled that automatically plans, controls, and monitors processes. Additionally, a process can be adapted to react to unanticipated events at runtime. During the adaptation all running activities and their anticipated effects are taken into account. Thus, processes can be adapted without interrupting the execution of activities. Another aspect is that in contrast to workflow management systems process definitions are not planned before an actual business case has arrived at the company. Therefore, all the case particularities are automatically taken into account, because they are known prior to planning.

Based on the concept of an integrated process planning and execution system, a formal model is constructed und analysed. This allows to refine the informal concepts and to derive a number of system properties. It is shown, under which conditions planning and executing a corresponding process transfers the state of a business case into a goal state. Furthermore, the correctness of the presented approach to adapt processes at runtime is deduced. Finally, the thesis is validated by a prototypical software implementation of the presented concepts.

Publikationen basierend auf dieser Dissertation

- **SCHUSCHEL, Hilmar** ; WESKE, Mathias: Fallbehandlung: Ein neuer Ansatz zur Unterstützung Prozessorientierter Informationssysteme. In: *Promise 2002: Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen*, Gesellschaft für Informatik, 2002 (LNI 21), S. 52–63
- **SCHUSCHEL, Hilmar** ; WESKE, Mathias: Integrated Workflow Planning and Coordination. In: *14th International Conference on Database and Expert Systems Applications*, Springer, 2003 (LNCS 2736), S. 771–781
- **SCHUSCHEL, Hilmar** ; WESKE, Mathias: Automated Planning in a Service-Oriented Architecture. In: *13th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, IEEE Computer Society Press, 2004, S. 75–80
- **SCHUSCHEL, Hilmar** ; WESKE, Mathias: Triggering Replanning in an Integrated Workflow Planning and Enactment System. In: *8th East-European Conference on Advances in Databases and Information Systems*, Springer, 2004 (LNCS 3255), S. 322–335
- **SCHUSCHEL, Hilmar** ; WESKE, Mathias: Plaengine: Ein System zur Planung und Ausführung von Workflows. In: *11. Fachtagung des GI-Fachbereichs Datenbanken und Informationssysteme*, Gesellschaft für Informatik, 2005 (LNI 65), S. 225–234
- GAJEWSKI, Michal ; MEYER, Harald ; MOMOTKO, Mariusz ; **SCHUSCHEL, Hilmar** ; WESKE, Mathias: Dynamic Failure Recovery of Generated Workflows. In: *16th International Conference and Workshop on Database and Expert Systems Applications*, IEEE Computer Society Press, 2005, S. 982–986

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Als Reaktion auf den starken Wettbewerb in den heutigen dynamischen Märkten, schöpfen Unternehmen immer weiter das Potenzial der Informationstechnologie aus, um die Effizienz und Flexibilität ihrer Organisation zu verbessern. Ein zentraler Punkt für die Wettbewerbsfähigkeit eines Unternehmens ist hierbei die Leistungsfähigkeit seiner Geschäftsprozesse. [51, 106] Ein *Prozess* ist eine partiell geordnete Menge von *Aktivitäten* die dazu dient, ein Ziel zu erreichen. [38] Entsprechend dient ein *Geschäftsprozess* [77] dazu, eine gegebene Situation so zu verändern, dass die Unternehmung ihr *Geschäftsziel* realisiert. Alle relevanten Informationen über diese Situation werden in Kombination mit dem zugeordneten Geschäftsziel als *Geschäftsfall* bezeichnet. Beispiele für Geschäftsfälle sind eine Buchbestellung oder eine Kreditwürdigkeitsanalyse.

Aus organisatorischer Sicht umfasst der Lebenszyklus eines Prozesses unter anderem die Phasen Planung und Ausführung. [28] In der *Planung* wird festgelegt, welche Aktivitäten in welcher Reihenfolge ausgeführt werden müssen, damit für einen Geschäftsfall das Geschäftsziel erreicht wird. Das Ergebnis der Planung ist eine *Prozessbeschreibung*. In der anschließenden Phase der *Ausführung* werden die einzelnen Aktivitäten entsprechend der Prozessbeschreibung ausgeführt und der Prozessfortschritt überwacht.

Im Kontext der Organisationstheorie hat Nordsieck [77] bereits Anfang der 70er Jahre die informationstechnische Unterstützung von Geschäftsprozessen vorhergesagt. Einen der ersten Schritte in diese Richtung machten Ellis und Nutt mit dem Büroautomatisierungssystem Officetalk-Zero [36]. Zisman entwickelte mit SCOOP [117] ein prototypisches System, das die informationstechnische Unterstützung auf die Ebene von Geschäftsprozessen hob. Es dauerte jedoch noch bis in die späten 80er Jahre, bevor die Voraussetzungen in den Bereichen Dokumentenmanagement, Kommunikationssysteme und Datenbanktechnologie gegeben waren, damit die Unterstützung von Geschäftsprozessen in Form von Workflow-Management-Systemen [42, 55, 68] große wirtschaftliche Bedeutung erlangte. [118]

Der Fokus dieser Systeme liegt dabei auf der Ausführungsphase eines Prozesses. [68] Die Idee hierbei ist, eine Prozessbeschreibung in Form einer formalisierten *Prozessdefinition* zur automatischen Steuerung der Prozessausführung zu nutzen. Das heißt, dass ein Workflow-Management-System auf Basis einer Prozessdefinition entscheidet, wann welche Aktivität auszuführen ist. Der wesentliche Vorteil dieser Vorgehensweise ist, dass die Prozesslogik nicht organisatorisch und softwaretechnisch fest implementiert wird, sondern durch Veränderung der Prozessdefinition leicht geändert werden kann. Die Planungsphase ist bei Workflow-Management-Systemen nicht automatisiert, sondern lediglich durch Werkzeuge unterstützt. Es wird ein Experte in Form eines Prozessmodellierers benötigt, der die Planung eines Prozesses übernimmt. Die Werkzeugunterstützung für den Prozessmodellierer besteht typischer Weise in der Möglichkeit zur graphischen Modellierung einer Prozessdefinition am Rechner und in der automatischen Prüfung formaler Korrektheitseigenschaften der Prozessdefinition.

Ein Schwachpunkt von klassischen Workflow-Management-Systemen ist die Berücksichtigung von unerwarteten Ereignissen zur Laufzeit eines Prozesses. [34, 49, 53, 83] Beispielsweise kann die Ausführung einer Aktivität fehlschlagen und die von der Ausführung der Aktivität erwarteten Resultate liegen nicht oder nur unvollständig vor. Nachfolgende Aktivitäten, die diese Resultate zur Ausführung benötigen, können daher nicht gestartet werden. Soll der Prozess nicht an dieser Stelle erfolglos abgebrochen werden, muss ein korrigierender Eingriff erfolgen. Eine Lösung einer solchen Situation ist der Rücksprung in die Planungsphase des Prozesses. Unter Berücksichtigung der bisher ausgeführten Aktivitäten und eingetretenen Ereignisse passt ein Prozessmodellierer die Prozessdefinition an und initiiert anschließend die Fortsetzung der Ausführung. Ein Kritikpunkt an klassischen Workflow-Management-Systemen ist, dass ein solcher Rücksprung in die Planungsphase informationstechnisch nur unzureichend unterstützt wird. Es existieren eine Reihe von Forschungsansätzen zur Prüfung der Korrektheit einer Anpassung von Prozessdefinitionen zur Laufzeit. [84] Diese Ansätze beschränken sich auf die Prüfung einer rein formalen, strukturellen Korrektheit der Anpassung. Auf diese Weise kann zwar eine technisch reibungslose Prozessausführung nach der Anpassung sichergestellt werden, jedoch ist damit nicht gewährleistet, dass mit der Ausführung auch das gewünschte Geschäftsziel erreicht wird. Eine derartige Aussage ist bei diese Ansätzen prinzipiell nicht möglich, da die eigentliche Funktionalität der Aktivitäten nicht modelliert wird. Der Prozessmodellierer wird zwar durch die Prüfung der formalen Korrektheit unterstützt, die Verantwortung dafür, dass die Prozessanpassungen zum gewünschten Ziel führen, trägt er jedoch weiterhin. Beispielsweise kann zwar erkannt werden, dass eine Aktivität als Eingabe einen bestimmten Typ von Dokumenten benötigt, es kann jedoch nicht erkannt werden, wenn bei der Anpassung einer Prozessdefinition statt einer Aktivität zur Sicherung eines Dokuments eine Aktivität zur Löschung des Dokuments eingefügt wurde.

Aus den obigen Darstellungen lässt sich unmittelbar ein weiteres grundsätzliches Problem klassischer Workflow-Management-Systeme beim Rücksprung in

die Planungsphase ableiten: Da die Planung nicht automatisiert ist, wird zur Anpassung der Prozessdefinition immer ein Prozessmodellierer benötigt. Dies stellt ein organisatorisches Problem dar, da die Verfügbarkeit eines Prozessmodellierers nicht nur bei der initialen Planung eines Prozesses, sondern auch zur Laufzeit des Prozesses für den Fall einer Prozessanpassung erforderlich ist. Des Weiteren ist die Anpassung der Prozessdefinition zur Laufzeit im Gegensatz zur initialen Planung typischerweise zeitkritisch, da sie die Bearbeitung eines konkreten Geschäftsfalls verzögert. Als Folge können leicht Fehler bei der Anpassung der Prozessdefinition unterlaufen, insbesondere wenn der Zeitdruck hoch oder ein qualifizierter Prozessmodellierer nicht verfügbar ist. Folglich sind Workflow-Management-Systeme schlecht für Anwendungen geeignet, in denen die Anpassung einer Prozessdefinition zur Laufzeit häufig notwendig oder besonders zeitkritisch ist.

Eine weitere Problematik beim Einsatz von Workflow-Management-Systemen ist die Verwendung einer im Voraus modellierten, gemeinsamen Prozessdefinition für die Bearbeitung einer Reihe von ähnlichen Geschäftsfällen. Beispielsweise werden alle Buchbestellungen auf Basis der gleichen Prozessdefinition bearbeitet. Die Ursache für dieses Vorgehen liegt darin, dass die manuelle Planung einer individuellen Prozessdefinition für jeden einzelnen Geschäftsfall, wie zum Beispiel die Buchbestellung von Herrn Müller am 28.12.2005, viel zu aufwendig wäre. Die Verwendung einer gemeinsamen Prozessdefinition für eine Reihe ähnlicher Geschäftsfälle ist adäquat, falls die Struktur der Geschäftsfälle sehr homogen beziehungsweise antizipierbar ist. In diesem Fall kann die Prozessdefinition so gestaltet werden, dass alle möglichen Ausprägungen kommender Geschäftsfälle berücksichtigt werden. Muss zum Beispiel beim Prozess zur Bearbeitung einer Buchbestellung bei einem Endkunden teilweise anders vorgegangen werden als bei einem Großkunden, so werden die entsprechenden Teile der Prozessdefinition in Form von alternativen Pfaden modelliert, die in Abhängigkeit vom Typ des Kunden bei der Ausführung ausgewählt werden. Die Prozessdefinition enthält somit sowohl die Information zur Handhabung einer Endkunden- als auch einer Großkundenbestellung. Ein Problem bei dieser Vorgehensweise ist, dass die Prozessdefinition unter Umständen durch die hohe Zahl alternativer Ausführungspfade sehr groß und unübersichtlich ist. Folglich ist sie schwer verständlich und schlecht zu überwachen. Ein weiteres Problem ergibt sich, wenn die Struktur der Geschäftsfälle sehr unterschiedlich und schlecht vorhersehbar ist. In diesem Fall ist es kaum möglich, im Voraus eine Prozessdefinition zu finden, mit der sich alle diese Geschäftsfälle bearbeiten lassen. Ein Beispiel hierfür sind Buchbestellungen, bei denen jeder Kunde Sonderwünsche hat. Wurde ein solcher Sonderwunsch bei der Erstellung der Prozessdefinition nicht antizipiert, so kann dieser Geschäftsfall nur bearbeitet werden, wenn manuell eingegriffen wird. Wurde beispielsweise bei der Modellierung einer Prozessdefinition für eine Bestellung über das Internet eine von der Rechnungsadresse abweichende Lieferadresse nicht vorgesehen, so kann dies nur durch einen manuellen Eingriff realisiert werden.

Derzeit gewinnen in der Informatik dienstbasierte Architekturen (engl. Service Oriented Architectures) [20] ständig an Bedeutung. In einer dienstbasierten Archi-

tektur werden Aktivitäten als Dienste von Dienstanbietern (engl. Service Provider) in einem Dienstverzeichnis (engl. Service Repository) publiziert. Das Dienstverzeichnis kann dann bei Bedarf von einem Dienstanforderer (engl. Service Consumer) nach einem gewünschten Dienst durchsucht werden. Wird der Dienst gefunden, erhält der Dienstanforderer die Adresse des entsprechenden Dienstanbieters. Charakteristisch für eine dienstbasierte Architektur ist, dass ständig neue Dienste angeboten werden können beziehungsweise das Angebot beendet werden kann. Für den Einsatz von Workflow-Management-Systemen in dienstbasierten Architekturen bedeutet dies, dass ständig neue Aktivitäten verfügbar werden, beziehungsweise plötzlich nicht mehr verfügbar sind. Dies führt zu dem Problem, dass Aktivitäten, die zum Zeitpunkt der Modellierung zur Verfügung standen und in der Prozessdefinition verwendet wurden, bei der Prozessausführung nicht mehr verfügbar sind. Auf der anderen Seite werden Aktivitäten nicht berücksichtigt, die zwar vor der Prozessausführung aber erst nach der Modellierung der Prozessdefinition verfügbar werden. Dies schränkt die Anwendungsmöglichkeiten von Workflow-Management-Systemen in dienstbasierten Architekturen stark ein.

1.2 Zielsetzung und wissenschaftlicher Beitrag

In dieser Arbeit wird ein Konzept zur integrierten Prozessplanung und -ausführung vorgestellt, das die oben beschriebenen Probleme klassischer Workflow-Management-Systeme löst. Der Lösungsansatz hierbei ist die Automatisierung der Planung durch den Einsatz von Planungstechniken der Künstlichen Intelligenz (KI) sowie deren konzeptionelle und softwaretechnische Integration mit Techniken zur Prozessausführung und -überwachung des Workflow-Managements.

Anstelle einer manuellen Modellierung von Prozessdefinitionen, erfolgt eine formale, semantische Spezifikation der zur Verfügung stehenden Aktivitäten sowie des Zustands und Ziels des zu bearbeitenden Geschäftsfalls. Es werden die Bedingungen festgelegt, die erfüllt sein müssen, damit eine Aktivität ausgeführt werden kann, und es werden die Effekte beschrieben, die die Ausführung der Aktivität auf den Zustand des Geschäftsfalls hat. Hierauf aufsetzend werden KI-Planungstechniken eingesetzt, um automatisch für diesen Geschäftsfall eine Prozessdefinition zu planen. Durch die Automatisierung werden Dauer und Kosten der Planung reduziert, wodurch es praktikabel wird, für jeden einzelnen Geschäftsfall eine individuelle Prozessdefinition zu planen. Dies hat zur Folge, dass der Planungszeitpunkt verschoben werden kann. Anstatt Geschäftsfälle antizipieren zu müssen, wird erst geplant, wenn ein konkreter Geschäftsfall vorliegt. Damit wird das Problem gelöst, dass bei der Antizipation von Geschäftsfällen mögliche Besonderheiten übersehen werden. Da alle Informationen über den konkreten Geschäftsfall zum Zeitpunkt der Planung vorliegen, besitzt eine individuelle Prozessdefinition keine alternativen Ausführungspfade, die für diesen konkreten Geschäftsfall nicht relevant sind. Daher ist eine individuelle Prozessdefinition in der Regel einfacher strukturiert und somit leichter verständlich als eine entsprechende Prozessdefini-

tion bei einem klassischen Workflow-Management-System, mit der eine Vielzahl von Geschäftsfällen bearbeitet werden soll. Eine individuelle Prozessplanung ist insbesondere in einer dienstbasierten Architektur von Vorteil, in der ständig neue Aktivitäten verfügbar werden und andere Aktivitäten nicht mehr angeboten werden. Da erst beim Eintreffen eines konkreten Geschäftsfalls die Prozessdefinition geplant wird, liegt typischer Weise weitaus weniger Zeit zwischen Planung und Ausführung eines Prozesses als bei klassischen Workflow-Management-Systemen. Hierdurch wird die Wahrscheinlichkeit verringert, dass sich zwischenzeitlich die Verfügbarkeit von Aktivitäten ändert.

Die Integration von Prozessplanung und -ausführung ermöglicht darüber hinaus, dass eine Prozessausführung nicht nur automatisch gesteuert, sondern auch automatisch überwacht und gegebenenfalls zur Berücksichtigung von unerwarteten Ereignissen angepasst werden kann. Dies wird durch die Planung einer neuen Prozessdefinition zur Laufzeit des Prozesses basierend auf dem aktuellen Zustand und Ziel des Geschäftsfalls erreicht. Da sich alle auftretenden Ereignisse im aktuellen Zustand des Geschäftsfalls widerspiegeln, werden bei dieser erneuten Planung auch die ursprünglich unerwarteten Ereignisse berücksichtigt. Aktivitäten, die sich während der Prozessanpassung gerade in Ausführung befinden, werden bei der Planung einer neuen Prozessdefinition gesondert betrachtet, da sie den Zustand des Prozesses während des Planungsvorgangs verändern können. Ein zentraler Vorteil der automatisierten Prozessanpassung ist, dass auch beim Auftritt von unerwarteten Ereignissen zur Laufzeit eines Prozesses nicht das organisatorische Problem besteht, die Verfügbarkeit eines Prozessmodellierers sicherzustellen. Ebenso findet die Prozessüberwachung automatisch statt. Dies hat zur Folge, dass Probleme bei der Prozessausführung schnell entdeckt und behoben werden, ohne dass ein manueller Eingriff notwendig ist. Eine korrekte Funktion des Planungsalgorithmus und eine konsistente Spezifikation der zur Verfügung stehenden Aktivitäten vorausgesetzt, können durch die Automatisierung Fehler bei einer Prozessanpassung vermieden werden. Dies ist insbesondere von Vorteil, falls die Prozessanpassung zeitkritisch ist und möglichst ohne Unterbrechung der Prozessausführung stattfinden soll.

Zur Erreichung dieser Ziele wird in dieser Arbeit ein konstruktivistischer Ansatz [40] gewählt, wie es für die wissenschaftliche Betrachtung und Analyse von informationstechnischen Systemen üblich ist. [118] Die Diskurswelt wird abgrenzt und relevante Begriffe werden eindeutig und zirkelfrei definiert. Basierend auf grundlegenden Annahmen wird ein informales Modell einer integrierten Prozessplanung und -ausführung aufgestellt. Es wird beschrieben, wie Prozesse individuell für einen Geschäftsfall geplant, ausgeführt und überwacht werden. Des Weiteren werden Verfahren zur automatischen Anpassung von laufenden Prozessen entwickelt. Aufbauend auf der Beschreibung des informalen Modells zur integrierten Prozessplanung und -ausführung, wird ein formales Modell dieses Systems formuliert und untersucht. Einerseits werden damit die vorgestellten Konzepte präzisiert, und andererseits können so eine Reihe von Systemeigenschaften hergeleitet werden. Es wird gezeigt, unter welchen Voraussetzungen durch die Bearbeitung in

dem System das Geschäftsziel eines Geschäftsfalls erreicht wird. Darüber hinaus wird nachgewiesen, dass die spezifizierte Vorgehensweise eine konsistente Anpassung einer Prozessdefinition zur Laufzeit ermöglicht. Als Validierung und weitere Konkretisierung der in dieser Arbeit vorgestellten Konzepte, wird ein System zur integrierten Prozessplanung und -ausführung prototypisch realisiert. Entwurf und Implementierung demonstrieren die softwaretechnische Umsetzbarkeit der Integration von Techniken aus Workflow-Management und KI-Planung.

Gegenstand dieser Arbeit ist die Integration von Prozessplanung und -ausführung. Die Integration basiert auf den Artefakten, die die Ein- beziehungsweise die Ausgabe einer Planung darstellen: die formale, semantische Spezifikation von Aktivitäten sowie von Zustand und Ziel des zu bearbeitenden Geschäftsfalls als Eingabe und die Prozessdefinition als Ausgabe. Der Zusammenhang von Ein- und Ausgabe wird deskriptiv spezifiziert. Demgegenüber wird nicht betrachtet, wie die Prozessdefinitionen geplant werden. Dies ist Gegenstand der KI-Planung und ist für die dargestellte Integration nicht relevant. Ebenso wird in dieser Arbeit die Verfügbarkeit und Zuordnung von Ressourcen für die Ausführung von Aktivitäten nicht betrachtet. Hier existieren Lösungen im Bereich des Workflow-Managements, die problemlos mit den in dieser Arbeit vorgestellten Konzepten kombiniert werden können. Grundsätzlich lässt sich auch die Zuordnung von Ressourcen mit Techniken der KI unterstützen. Dieser Aspekt wird jedoch bewusst ausgeklammert.

1.3 Aufbau der Arbeit

Die Arbeit gliedert sich wie folgt: In Kapitel 2 erfolgt eine kurze Einführung in die für diese Arbeit relevanten Konzepte und Techniken aus den Disziplinen Workflow-Management und KI-Planung.

In Kapitel 3 wird das grundlegende Konzept zur Integration von Prozessplanung und -ausführung vorgestellt. Hierzu werden einander entsprechende, fachspezifische Entitäten der beiden Disziplinen Workflow-Management und KI-Planung identifiziert. Darauf aufbauend wird die Möglichkeit zur Planung individueller Prozessdefinitionen für jeden einzelnen Geschäftsfall aufgezeigt. Schließlich werden die Besonderheiten des Einsatzes von KI-Planungstechniken in einer Geschäftsdomäne erläutert und diskutiert.

Kapitel 4 erweitert die in Kapitel 3 vorgestellte Lösung um ein Konzept für die Anpassung einer Prozessdefinition zur Laufzeit. Die Anpassung wird in einzelne Teilaufgaben zerlegt. Für jede Teilaufgabe werden Verfahren und Methoden entwickelt. Insbesondere werden drei alternative Lösungen zur Berücksichtigung von Aktivitäten vorgestellt, die sich bei der Anpassung der Prozessdefinition gerade in Ausführung befinden. Die Lösungen werden im Hinblick darauf verglichen, inwieweit die Ausführung für eine Prozessanpassung unterbrochen werden muss.

In Kapitel 5 wird ein formales Modell für ein System zur integrierten Prozessplanung und -ausführung formuliert und untersucht. Die in den Kapiteln 3 und 4 entwickelten Konzepte werden dabei präzisiert und formalisiert. Neben der Präzi-

sierung der vorgestellten Konzepte, ermöglicht die formale Darstellung ein detaillierteres Verständnis der Verknüpfung von Prozessplanung und -ausführung. Zunächst werden die relevanten Abläufe und involvierten Artefakte modelliert. Darauf aufbauend wird eine Reihe von Eigenschaften des Systems hergeleitet. Es wird gezeigt, unter welchen Bedingungen der Zustand eines Geschäftsfalls durch Planung und Ausführung in einen Zielzustand überführt wird und nachgewiesen, dass die spezifizierte Vorgehensweise eine konsistente Anpassung eines Prozesses zur Laufzeit ermöglicht.

Kapitel 6 beschreibt die softwaretechnische Realisierung eines Prototypen namens *Plaengine* zur integrierten Prozessplanung und -ausführung. Es werden Entwurfsaspekte wie Dienstorientierung, Persistenz, Wiederverwendung und Modularisierung vorgestellt und eine entsprechende Systemarchitektur der wesentlichen Komponenten hergeleitet. Anschließend wird die statische Struktur der relevanten konzeptionellen und technischen Artefakte aus Planung und Ausführung und ihre Verknüpfung in einem Klassendiagramm dargestellt.

In Kapitel 7 werden die in dieser Arbeit aufgezeigten Konzepte und Lösungen diskutiert. Es wird besprochen, inwieweit die bei der Modellbildung getroffenen Annahmen eine adäquate Abstraktion beziehungsweise Einschränkung darstellen und wie das Modell bei Aufhebung einzelner Annahmen geändert werden müsste. Des Weiteren wird die Verknüpfung von Workflow-Management und KI-Planung auf konzeptioneller Ebene und auf softwaretechnischer Ebene reflektiert. Anschließend werden Anwendungsgebiete identifiziert, in denen sich aufgrund ihrer Charakteristika der Einsatz einer integrierten Prozessplanung und -ausführung als besonders vorteilhaft darstellt und Grenzen des Ansatzes diskutiert. Schließlich werden sinnvolle Erweiterungen der vorgestellten Konzepte und Verfahren skizziert.

In Kapitel 8 werden die verwandten Arbeiten vorgestellt und diskutiert. Hierzu werden die verwandten Arbeiten in vier Gebiete unterteilt: Arbeiten zur dynamischen Adaption von Workflows, Ansätze für eine Daten- und kundenorientierte Prozesssteuerung, Arbeiten zur Automatisierung der Planung von Geschäftsprozessen und Arbeiten zur automatisierten Prozessüberwachung. Es werden die zentralen Ideen der einzelnen Ansätze beschrieben und von dieser Arbeit abgegrenzt. Als Abschluss der Arbeit enthält Kapitel 9 eine Zusammenfassung der Ergebnisse, einen Ausblick auf die zukünftigen Forschungsschwerpunkte und ein Fazit.

Kapitel 2

Grundlagen

In diesem Kapitel werden die Grundlagen der Wissenschaftsdisziplinen Workflow-Management und Planung in der Künstlichen Intelligenz erläutert, die für diese Arbeit relevant sind. Hierbei werden jeweils die für die Disziplin spezifischen Termini verwendet, so dass teilweise für einander entsprechende Entitäten der beiden Disziplinen unterschiedliche Bezeichnungen verwendet werden. Eine Zusammenführung der Terminologie der beiden Disziplinen erfolgt als Teil der Integration von Prozessplanung und -ausführung im folgenden Kapitel.

2.1 Workflow-Management

Die zentrale Aufgabe eines Workflow-Management-Systemes [42, 55, 68, 94, 108] ist die Steuerung von Prozessen auf Basis von expliziten Prozessdefinitionen. Explizit bedeutet hier, dass Arbeitsabläufe nicht zusammen mit der übrigen Funktionalität der Anwendungssysteme als Programmcode fest implementiert sind, sondern als formale Prozessdefinitionen mittels einer Prozessbeschreibungssprache spezifiziert werden. Die *Prozessbeschreibungssprache* beschreibt dabei die Konstrukte, die zur Spezifikation einer Prozessdefinition zur Verfügung stehen. Ein wesentlicher Vorteil dieser Vorgehensweise ist, dass zur Änderung von Arbeitsabläufen nur die Modifikation der entsprechenden Prozessdefinition notwendig ist und keine Änderung der Software einer oder mehrerer Anwendungen. Bei Workflow-Management-Systemen wird unterschieden zwischen Designzeit und Laufzeit. [68, 108, 118] In der ersten Phase, die als *Designzeit* bezeichnet wird, werden alle Vorbereitungen für den Betrieb des Systems getroffen. Dies umfasst insbesondere die Spezifikation der Prozessdefinitionen. Die anschließende Betriebsphase des Systems wird als *Laufzeit* bezeichnet. In dieser Phase werden einzelne Geschäftsfälle bearbeitet, indem ihnen eine Prozessdefinition zugeordnet, instanziiert und diese Instanz ausgeführt wird.

Abbildung 2.1 gibt eine Übersicht über die Funktionsweise eines Workflow-Management-Systemes und seine organisatorische Einbettung. Es sind die drei für diese Arbeit wesentlichen Komponenten eines Workflow-Management-Systemes

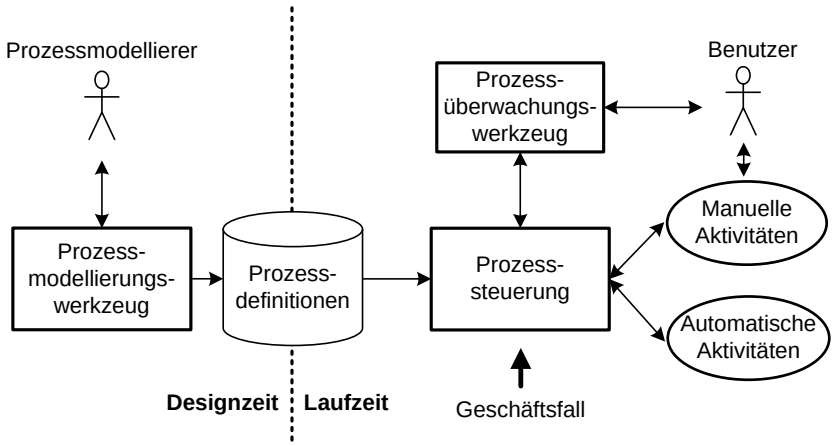


Abbildung 2.1: Klassisches Workflow-Management-System

dargestellt: das Prozessmodellierungswerkzeug, die Prozesssteuerung und das Prozessüberwachungswerkzeug. Unter einem Werkzeug wird in diesem Zusammenhang Software-Komponenten verstanden, die den Benutzer bei einer Tätigkeit zwar unterstützen, die eigentliche Funktionalität jedoch nicht selbständig ausführen. Das Prozessmodellierungswerkzeug unterstützt den Prozessmodellierer bei der Spezifikation von Prozessdefinitionen. Die erstellten Prozessdefinitionen werden für die spätere Verwendung in der Laufzeit gespeichert. Diese gespeicherten Prozessdefinitionen bilden den Übergang zwischen Designzeit und Laufzeit. Die zentrale Komponente in der Laufzeit ist die Prozesssteuerung. Ihre Aufgabe ist die Steuerung der Ausführung der Prozessdefinitionen, das heißt der Start der richtigen Aktivitäten zum richtigen Zeitpunkt. Es werden manuelle und automatische Aktivitäten unterschieden. *Manuelle Aktivitäten* erfordern im Gegensatz zu *automatischen Aktivitäten* die Interaktion mit einem Benutzer. Bei der Überwachung des Prozesses wird der Benutzer durch ein Prozessüberwachungswerkzeug unterstützt. Dieses liefert Informationen über den Bearbeitungsfortschritt der Prozesse und den Zustand der einzelnen Aktivitäten.

Der Rest dieses Abschnitts gliedert sich wie folgt. In Abschnitt 2.1.1 werden die für den Zweck dieser Arbeit wesentlichen Aspekte von Prozessbeschreibungssprachen erläutert. Abschnitt 2.1.1 beschreibt die typische Vorgehensweise bei Workflow-Management-Systemen, zur Planung von Prozessdefinitionen ähnliche Geschäftsfälle zu Geschäftsfalltypen zusammenzufassen. In Abschnitt 2.1.3 werden die oben angesprochenen Systemkomponenten genauer erläutert.

2.1.1 Prozessbeschreibungssprachen

Prozessbeschreibungssprachen [3, 110] dienen der formalen Beschreibung von Prozessen in Form von Prozessdefinitionen. Das zentrale Ziel dieser Beschreibung ist

die informationstechnische Unterstützung der Prozesse durch ein Workflow-Management-System. Im Kontext dieser Arbeit sind hierbei die Modellierung von Kontroll- und Datenfluss relevant. Andere Aspekte einer Prozessbeschreibungssprache wie die Zuordnung von Aktivitäten zu Organisationseinheiten, Rollen und Ressourcen sind für die hier vorgestellte Integration von Prozessplanung und -ausführung nicht wesentlich.

Der *Kontrollfluss* spezifiziert, wann welche der Aktivitäten eines Prozesses ausgeführt werden dürfen. Hierbei werden im Wesentlichen sequentieller, paralleler und alternativer Kontrollfluss unterschieden. *Sequentieller Kontrollfluss* ordnet einer Menge von Prozessteilen eine Ausführungsreihenfolge zu. Ein Prozessteil kann dabei entweder eine einzelne Aktivität sein oder sich wieder aus einer Menge kleinerer Prozessteile und ihrem Kontrollfluss zusammensetzen. Folgt ein Prozessteil T_1 auf einen Prozessteil T_2, so bedeutet dies, dass Teil T_1 beendet sein muss, bevor die Ausführung von Teil T_2 beginnen darf. *Paralleler Kontrollfluss* legt eine Menge von Prozessteilen fest, die nebenläufig ausgeführt werden können. Dies bedeutet, dass zwischen diesen Prozessteilen keine zeitlichen Abhängigkeiten bezüglich ihrer Ausführung bestehen. *Alternativer Kontrollfluss* legt eine Menge von Prozessteilen fest, die alternativ ausgeführt werden können. Das heißt, es wird spezifiziert unter welchen Bedingungen welche dieser Prozessteile ausgeführt werden.

In Bezug auf den Kontrollflusses werden block- und graphbasierte Prozessbeschreibungssprachen unterschieden. *Blockbasierte Prozessbeschreibungssprachen*, wie beispielsweise Microsoft's XLANG [102], fassen jeweils eine Menge von Aktivitäten zu einem Block zusammen und ordnen diesem Block einen bestimmten Kontrollfluss zu. Diese Blöcke können ebenso wie Aktivitäten zu größeren Blöcken zusammengefasst werden. Auf diese Weise bildet sich eine hierarchische Struktur von Kontrollblöcken. Demgegenüber bilden bei *graphbasierten Prozessbeschreibungssprachen* [68], wie zum Beispiel IBM's Web Service Flow Language (WSFL) [67], gerichtete Graphen das zentrale Modellierungskonstrukt. Die Knoten eines solchen Graphen repräsentieren entweder einzelne Aktivitäten oder Sub-Graphen. Die Kanten des Graphen werden als *Kontrollkonnektoren* bezeichnet und spezifizieren den Kontrollfluss. Ein Kontrollkonnektor von einer Aktivität A_1 zu einer Aktivität A_2 bringt zum Ausdruck, dass die Ausführung von A_1 beendet sein muss, bevor die Ausführung von A_2 gestartet werden kann.

Aktivitäten sind häufig informationsverarbeitende Aktivitäten. Diesen Aktivitäten müssen zur Ausführung bestimmte Daten zur Verfügung gestellt werden und sie erzeugen bei ihrer Ausführung neue Daten. Beispielsweise benötigt eine Aktivität *multiplizieren* zwei Faktoren als Eingabe und liefert ein Produkt als Ausgabe. Die Eingabe einer Aktivität wir in Form von *Eingabeparametern* spezifiziert und die Ausgabe in Form von *Ausgabeparametern*. Die Aktivität *multiplizieren* besitzt entsprechend zwei Eingabeparameter und einen Ausgabeparameter. Ist eine informationsverarbeitende Aktivität Teil einer Prozessdefinition, so spezifiziert der *Datenfluss*, woher die Eingabe für eine Aktivität kommt und wohin die Ausgabe der Aktivität weiter geleitet wird. Im Fall von graphbasierten Prozessbeschreibungssprachen kann der Datenfluss zum Beispiel in Form von *Datenkonnektoren*

spezifiziert werden. Ein Datenkonnektor verbindet einen Ausgabeparameter einer Aktivität mit dem Eingabeparameter einer anderen Aktivität. Das Workflow-Management-System hat dann die Aufgabe entsprechend die Daten zwischen den Aktivitäten weiterzuleiten.

2.1.2 Typisierung von Geschäftsfällen

In der Regel werden ähnliche Geschäftsfälle zu Geschäftsfalltypen zusammengefasst, so dass nicht für jeden Geschäftsfall, sondern nur für jeden Geschäftsfalltyp eine Prozessdefinition benötigt wird. Die Idee hierbei ist, dass ähnliche Geschäftsfälle auf Basis der gleichen Prozessdefinition bearbeitet werden können. Ein Beispiel für zwei derartige Geschäftsfälle ist die Buchbestellung von Herrn Müller am 28.12.2005 und die Buchbestellung von Herrn Meier vom 31.12.2005. Obwohl sich hier einige Ausprägungen der Geschäftsfälle unterscheiden, können beide auf Basis der gleichen Prozessdefinition bearbeitet werden. Es würde sich hier anbieten, alle Buchbestellungen zu einem Geschäftsfalltyp Buchbestellung zusammenzufassen.

In Bezug auf Designzeit und Laufzeit ergeben sich bei der Typisierung von Geschäftsfällen folgende Vorgehensweisen. Zur Designzeit müssen zunächst die Geschäftsfälle antizipiert werden, die zukünftig bearbeitet werden sollen. Darauf aufbauend werden Geschäftsfalltypen definiert, die jeweils eine Menge einzelner Geschäftsfälle zusammenfassen. Dies bezeichnen wir als *Typisierung von Geschäftsfällen*. Anschließend wird für jeden Geschäftsfalltyp eine Prozessdefinition modelliert und diesem zugeordnet. Zur Laufzeit wird für eingehende Geschäftsfälle soweit möglich der entsprechende Geschäftsfalltyp bestimmt. Dann wird die dem Geschäftsfalltyp zugeordnete Prozessdefinition mit den Werten des Geschäftsfalls ausgeführt. Diese konkrete Ausführung der Prozessdefinition für einen bestimmten Geschäftsfall wird als *Prozessinstanz* bezeichnet. Entsprechend werden auch die Arbeitsschritte einer Prozessdefinition als *Aktivitätsdefinitionen* und die Arbeitsschritte einer Prozessinstanz als *Aktivitätsinstanzen* bezeichnet. Falls es eindeutig ist, ob eine Aktivitätsdefinition oder eine Aktivitätsinstanz gemeint ist, wird im weiteren Verlauf dieser Arbeit vereinfachend von einer Aktivität gesprochen.

2.1.3 Systemkomponenten

In diesem Abschnitt werden die drei in Abbildung 2.1 dargestellten Komponenten eines Workflow-Management-Systems beschrieben: die Prozesssteuerung, das Prozessmodellierungs- und das Prozessüberwachungswerkzeug.

Prozesssteuerung

Die Prozesssteuerung ist die zentrale Komponente eines Workflow-Management-Systems und hat die Aufgabe Prozessausführungen zu steuern. Ausgangspunkt

hierbei ist ein Geschäftsfall und eine zugeordnete Prozessdefinition. Zur Bearbeitung des Geschäftsfalls wird eine Prozessinstanz von der Prozessdefinition erzeugt. Dieser Vorgang wird als *Instanziierung* bezeichnet. Dabei werden die konkreten Werte des Geschäftsfalls in die Prozessinstanz übernommen. Anschließend wird die Prozessinstanz gestartet. Dabei werden die Aktivitäten gestartet, die entsprechend dem Kontrollfluss als erste aufgeführt werden können. Manuelle Aktivitäten werden, sobald sie gestartet sind, dem Benutzer auf einer *Arbeitsliste* zur Bearbeitung angeboten. Automatische Aktivitäten werden zur Bearbeitung an die zugeordneten Programme weitergeleitet. Dabei sorgt die Prozesssteuerung auch dafür, dass Daten dem spezifizierten Datenfluss entsprechend an die auszuführenden Aktivitäten weitergeleitet werden. Sobald die Bearbeitung einer Aktivität abgeschlossen ist, überprüft die Prozesssteuerung, ob dem Kontrollflusses folgend weitere Aktivitäten gestartet werden können.

Prozessmodellierungswerkzeug

Bei der Modellierung einer Prozessdefinition für einen Geschäftsfalltyp wird der Prozessmodellierer in der Regel durch ein Prozessmodellierungswerkzeug vom Workflow-Management-System unterstützt. Eine typische Funktionalität ist hier die Möglichkeit einer graphischen Modellierung einer Prozessdefinition. Bei dieser rechnergestützten Modellierung können automatisch formale Prüfungen der Korrektheit der Prozessdefinition durchgeführt werden. Falls zum Beispiel ein Kontrollkonnektor von einer Aktivität A zu einer Aktivität B führt und gleichzeitig ein Datenkonnektor von der Aktivität B zur Aktivität A, dann kann dieser Widerspruch zwischen Kontroll- und Datenfluss vom Prozessmodellierungswerkzeug erkannt werden.

Prozessüberwachungswerkzeug

Das Prozessüberwachungswerkzeug unterstützt den Benutzer bei der Überwachung der laufenden Prozessinstanzen. Der Benutzer sieht typischer Weise eine graphische Darstellung der Prozessinstanz, aus der der Ausführungszustand der enthaltenen Aktivitätsinstanzen und die aktuelle Belegung aller Eingabe- und Ausgabeparameter ersichtlich wird. Auf diese Weise kann der Benutzer schnell einen Überblick über den Bearbeitungsstand einer Prozessinstanz erhalten und gegebenenfalls Probleme bei der Ausführung erkennen. Neben dieser Informations-Funktion bietet das Prozessüberwachungswerkzeug auch Möglichkeiten in den Ablauf der Prozessausführung einzugreifen. Beispielsweise kann die Ausführung an einzelnen Stellen unterbrochen werden, oder einzelnen Aktivitäten können übersprungen werden.

2.2 Planung in der Künstlichen Intelligenz

Die Planung [45, 89] als Teildisziplin der Künstlichen Intelligenz (KI) beschäftigt sich mit der Erzeugung eines zielgerichteten Handelns für Akteure. Ein Akteur kann in diesem Zusammenhang zum Beispiel ein Mensch, ein Roboter oder auch ein soziotechnisches System sein. Forschungsgegenstand der KI-Planung sind insbesondere die Entwicklung von formalen Sprachen zur Beschreibung von Zuständen, Zielen und Aktionen eines Akteurs, sowie die Entwicklung von Algorithmen zur Planung von geordneten Mengen von zielgerichteten Aktionen. KI-Planung ist wie folgt definiert: Gegeben ist ein *Planungsproblem*, das aus einem Ausgangszustand, einem Ziel, sowie einer Menge möglicher Aktionen besteht. KI-Planung bezeichnet die Suche nach einem *Plan* als eine geordnete Menge von Aktionen, die den Ausgangszustand in einen Zustand überführt, in dem das Ziel gilt. Während der Forschungsschwerpunkt der KI-Planung auf der effizienten Suche nach einem Plan zu einem gegebenen Planungsproblem liegt, spielt in dieser Arbeit die Suche selbst keine relevante Rolle, da nur die Anwendung der KI-Planung betrachtet wird. Dementsprechend wird im folgenden zwar der Zusammenhang von Planungsproblem und Plan betrachtet, aber nicht beschrieben, welche Techniken und Algorithmen entwickelt wurden, um von einem Planungsproblem zu einem passenden Plan zu kommen.

Zunächst werden in Abschnitt 2.2.1 kurz die relevanten Begriffe der KI-Planung definiert. Anschließend werden in Abschnitt 2.2.2 zwei Sprachen zur Spezifikation von Planungsproblemen vorgestellt. Abschnitt 2.2.3 beschreibt die unterschiedlichen Arten von Plänen, die ein Planer ausgeben kann. Schließlich wird in Abschnitt 2.2.4 der Zusammenhang von Planungsproblem und Plan in der KI-Planung anhand eines Beispiels illustriert.

2.2.1 Planungsterminologie

Der *Ausgangszustand* ist der momentane Zustand des relevanten Teils der Welt. Ein *Ziel* ist ein partiell definierter Zustand, der die Menge der gewünschten *Zielzustände* beschreibt. Das Mittel zur Überführung des Ausgangszustands in einen Zielzustand sind die Aktionen. Eine *Aktion* ist ein logisch zusammengehöriger Arbeitsschritt. Die Bedingungen, die festlegen, in welchem Zustand eine Aktion ausgeführt werden kann, werden als *Vorbedingungen* bezeichnet. Demgegenüber bezeichnen *Effekte* die Auswirkungen der Ausführung der Aktion auf den Zustand. Die Menge aller Aktionen wird als *Domäne* bezeichnet. Ein *Planungsproblem* besteht aus Ausgangszustand, Ziel und Domäne. Unter einem *Planer* versteht man eine Softwarekomponente, die ein Planungsproblem als Eingabe nimmt und als Ausgabe die Lösung dieses Planungsproblems in Form eines Plans liefert. Ein *Plan* spezifiziert eine Menge von Aktionen und legt fest, wann beziehungsweise in welcher Reihenfolge die Aktionen ausgeführt werden müssen. Ein Plan löst ein Planungsproblem, wenn die Ausführung der Aktionen des Plans den Ausgangszustand in einen Zielzustand überführt.

2.2.2 Sprachen zur Spezifikation von Planungsproblemen

Zur formalen, semantischen Spezifikation von Planungsproblemen, das heißt von Zuständen, Zielen und Aktionen, werden logische Ausdrücke verwendet. [45] Hierbei gilt es, eine Sprache zu finden, die ausdrucksstark genug ist, um eine große Vielfalt an Problemen zu beschreiben, aber einschränkend genug ist, um effiziente Algorithmen dafür zuzulassen. In diesem Abschnitt werden mit STRIPS und PDDL zwei auf Prädikatenlogik basierende Sprachen beschrieben. STRIPS ist eine sehr elementare Sprache und dient hier dem grundlegenden Verständnis der Spezifikation von Planungsproblemen. PDDL ist demgegenüber weitaus ausdrucksstärker und spielt eine wichtige Rolle bei der in dieser Arbeit vorgestellten Integration von Prozessplanung und -ausführung.

STRIPS

Der *Stanford Research Institute Problem Solver* (STRIPS) wurde 1971 in [39] vorgestellt und definiert eine Repräsentationssprache für Planungsprobleme. Es handelt sich dabei um eine eingeschränkte Form der Prädikatenlogik, die sich, teilweise leicht abgewandelt, zur Sprache vieler klassischer Planer entwickelt hat. In STRIPS wird die Welt in eine Menge von Aussagen zerlegt und ein Zustand als Konjunktion von positiven, funktionsfreien Grundliteralen repräsentiert. Es gilt die Annahme der *geschlossenen Welt* [89], das heißt, alle Bedingungen, die in einem Zustand nicht erwähnt werden, werden als falsch angenommen. Ein Ziel wird als Konjunktion positiver Grundliterale repräsentiert. Ein Zustand S erfüllt ein Ziel G, genau dann wenn S alle Atome in G enthält. Mögliche Aktionen werden in Form von *Aktionsschemata* spezifiziert. Aktionen werden aus einem Aktionsschema durch Instanziierung der Variablen abgeleitet. Ein Aktionsschema besteht aus folgenden Teilen:

- Aktionsname und Parameterliste
- Vorbedingungen als Konjunktion funktionsfreier positiver Literale
- Effekte als Konjunktion funktionsfreier Literale

Nach der Definition der Syntax von STRIPS folgt nun die Definition der Semantik. Hierzu wird spezifiziert, wie sich die Ausführung einer Aktion auf einen Zustand auswirkt: Eine Aktion ist in jedem Zustand *anwendbar*, der ihre Vorbedingungen erfüllt. Für die Überprüfung der Anwendbarkeit, werden Variablen in der Vorbedingung gegebenenfalls mit konkreten Werten substituiert. Ist eine Aktion anwendbar und wird ausgeführt in einem Zustand S, so ergibt sich ein Folgezustand S', der gleich dem Zustand S ist, außer dass die positiven Literale der Effekte der Aktion im Folgezustand S' als wahr angenommen und die negativen Literale als falsch. Ein Beispiel, dass unter anderem die Spezifikation eines Planungsproblems in STRIPS illustriert folgt in Abschnitt 2.2.4.

PDDL

Die *Planning Domain Definition Language* [44] (PDDL) wurde ursprünglich für einen Wettbewerb von KI-Planern auf der *Artificial Intelligence Planning Systems*-Konferenz im Jahr 1998 entwickelt. Die Idee hierbei war, dass ein Vergleich der Performanz verschiedener Planer nur möglich ist, wenn sie die gleiche Sprache zur Beschreibung von Planungsproblemen verwenden. PDDL wird seitdem regelmäßig für diesen Wettbewerb eingesetzt und hat sich darüber hinaus zu einem Standard zur Beschreibung von Planungsproblemen entwickelt.

Den Kern von PDDL bildet der STRIPS-Formalismus. Der Sprache wurden jedoch Erweiterungen hinzugefügt, um die Ausdrucksfähigkeit zu erhöhen. Hierzu gehören unter anderem die Typisierung von Variablen, All- und Existenzquantifikation und Negation. Die Version 2.1 [41] von PDDL beinhaltet darüber hinaus Erweiterungen für numerisches und temporales Planen mit Optimierungsfunktionen. Mit diesen Erweiterungen lassen sich zum Beispiel Kosten und Dauer einer Aktion modellieren und so beim Planen die Gesamtkosten und die Ausführungsdauer des Plans optimieren. Im Gegensatz zur ursprünglichen Version von PDDL beinhaltet PDDL 2.1 des Weiteren eine formale Spezifikation der Semantik der Sprache. Insbesondere wird definiert, wann ein Plan für ein Planungsproblem gültig ist.

2.2.3 Kategorisierung von Plänen

Im Gegensatz zu PDDL für die Spezifikation von Planungsproblemen, hat sich für die Darstellung von Plänen bislang kein Standard etabliert. In der klassischen KI-Planung ohne die Berücksichtigung von Unsicherheit und Zeit, wird zwischen einem vollständig geordneten Plan und und einem partiell geordneten Plan unterschieden. [45] Ein *vollständig geordneter Plan*, wie er zum Beispiel vom Planer INTERPLAN [100] erzeugt wird, besteht aus einer Menge von Aktionen, die vollständig geordnet sind. Ein vollständig geordneter Plan löst ein Planungsproblem, wenn die Aktionen des Plans, falls sie in der spezifizierten Reihenfolge ausgeführt werden, den Ausgangszustand in einen Zielzustand überführen. Ein *partiell geordneter Plan* besteht aus einer Menge von Aktionen, die partiell geordnet sind. Angenommen ein partiell geordneter Plan P besteht aus den Aktionen A, B und C und es gelten die *Ordnungsbedingungen*, dass A irgendwann vor B ausgeführt werden muss und A irgendwann vor C. Dieser Plan lässt offen, ob B vor oder nach C ausgeführt wird. Die Ausführung eines partiell geordneten Plans wird als Linearisierung interpretiert. Ein partiell geordneter Plan löst ein Planungsproblem, wenn jede seiner Linearisierungen den Ausgangszustand in einen Zielzustand überführt. [89] Dies bedeutet, dass der Plan P ein gegebenes Planungsproblem löst, falls sowohl die Linearisierung $A\ B\ C$ als auch die Linearisierung $A\ C\ B$ das Planungsproblem lösen. Ein partiell geordneter Plan besitzt gegenüber einem vollständig geordnetem Plan den Vorteil, dass der Akteur bei der Ausführung des Plans flexibler ist. In dem Beispiel kann er sich entscheiden, ob er nach der Ausführung der Aktion A zunächst B ausführt oder zunächst C. Beispiele für Planer die partiell

geordnete Pläne ausgeben sind NOAH [90], NONLIN [101], MOLGEN [98] und SIPE [57].

Die klassische KI-Planung abstrahiert von der Ausführungsdauer einer Aktion. Es wird lediglich betrachtet, ob eine Aktion vor oder nach einer anderen Aktion stattfindet. Dementsprechend wird bei einem klassischen partiell geordnetem Plan auch keine Nebenläufigkeit von Aktionen betrachtet. Auf das obige Beispiel bezogen bedeutet dies, dass zwar die Reihenfolge der Aktionen B und C beliebig ist, jedoch die Bedeutung einer gleichzeitigen beziehungsweise nebenläufigen Ausführung nicht erfasst wird. Einen Plan für ein Planungsproblem, das die Ausführungsdauer von Aktionen berücksichtigt bezeichnen wir als *temporalen Plan*. Für die Darstellung von temporalen Plänen gibt es zwei grundsätzliche Ansätze: Einerseits kann ein solcher Plan als partiell geordneter Plan dargestellt werden. Die Ordnungsbedingung, dass Aktion A vor B ausgeführt werden muss, bedeutet in diesem Fall, dass die Aktion A beendet sein muss, bevor Aktion B gestartet werden kann. Zwei nicht geordnete Aktionen können dagegen auch zeitlich überlappend ausgeführt werden. Andererseits kann ein temporaler Plan auch durch Festlegung der Startzeitpunkte der enthaltenen Aktionen spezifiziert werden. Einen derartigen Plan bezeichnen wir als *startzeit-gebundenen Plan*. Angenommen die Aktion A hat eine Dauer von 3 Zeiteinheiten, so können anstelle einer Ordnungsbedingung „A vor B“ in einem partiell geordneten Plan die Startzeitpunkte 0 für Aktion A und 3 für Aktion B in einem startzeit-gebundenen Plan festgelegt werden. Startzeit-gebundene Pläne werden beispielsweise von den Planern SAPA [30], TLPlan [14] und TGP [97] erzeugt. Ein partiell geordneter Plan lässt einem Akteur in der Regel mehr Freiheit als ein entsprechender startzeit-gebundener Plan. [104] Ein Ansatz zur Transformation von startzeit-gebundenen Plänen in partiell geordnete Pläne findet sich in [31].

In der klassischen KI-Planung wird davon ausgegangen, dass die Effekte von Aktionen deterministisch sind. Das heißt, dass die Ausführung einer Aktion stets genau die spezifizierten Effekte hat. Ein Ansatz, um auch undeterministische Effekte zu berücksichtigen, wird als *Kontingenzplanen* [82] bezeichnet. Hierbei wird davon ausgegangen, dass nach Ausführung einer Aktion, der neue Zustand beobachtbar ist. Es werden Prüfaktionen — so genannte *Sensing Actions* — und anschließende Verzweigungen in den Plan eingebaut, die es erlauben, die Effekte von Aktionen zu überprüfen und eine entsprechende Verzweigung auszuwählen. Ein derartiger Plan wird als *bedingter Plan* bezeichnet. Bedingte Pläne können neben Verzweigungen auch Zyklen enthalten. Falls beispielsweise eine Aktion D nur manchmal den gewünschten Effekt hat, kann anschließend eine Prüfaktion ausgeführt werden und gegebenenfalls ein Rücksprung zur Aktion D stattfinden, in der Hoffnung, dass die erneute Ausführung das gewünschte Ergebnis hat.

2.2.4 Beispiel

Zur Illustration der KI-Planung wird in diesem Abschnitt ein beispielhaftes Planungsproblem dargestellt und ein Plan beschrieben, der dieses Planungsproblem

löst. Für die Spezifikation des Planungsproblems wird STRIPS verwendet, da Syntax und Semantik leicht verständlich ist. Zur weiteren Vereinfachung beschränken wir uns dabei zusätzlich auf aussagenlogische Ausdrücke.

Informal beschrieben geht es in unserem Beispiel darum, Socken und Schuhe anzuziehen und den Müll wegzubringen. Dabei ist die Ausgangssituation die, dass der Akteur barfuß ist. Es folgt nun die formale Beschreibung dieses Planungsproblems. Hierzu verwenden wir folgende Aussagensymbole:

SockeRechts : Am rechten Fuß trägt der Akteur eine Socke.
SockeLinks : Am linken Fuß trägt der Akteur eine Socke.
SchuhRechts : Am rechten Fuß trägt der Akteur einen Schuh.
SchuhLinks : Am linken Fuß trägt der Akteur einen Schuh.
keinSchuhRechts : Am rechten Fuß trägt der Akteur keinen Schuh.
keinSchuhLinks : Am linken Fuß trägt der Akteur keinen Schuh.
MüllEntsorgt : Der Müll ist entsorgt.

Die formale Beschreibung des Ausgangszustands S für das obige Beispiel ist somit: *keinSchuhRechts* $\wedge$ *keinSchuhLinks*. Aufgrund der Annahme der geschlossenen Welt, ist hiermit auch zum Ausdruck gebracht, dass die Aussagen *SchuhRechts*, *SchuhLinks*, *SockeRechts*, *SockeLinks* und *MüllEntsorgt* falsch sind. Zur Veränderung eines Zustandes stehen folgende Aktionen zur Verfügung:

Aktion(*SchuhRechtsAnziehen*,
Vorbedingung: *keinSchuhRechts*,
Effekt: $\neg$*keinSchuhRechts* $\wedge$ *SchuhRechts*)

Aktion(*SchuhLinksAnziehen*,
Vorbedingung: *keinSchuhLinks*,
Effekt: $\neg$*keinSchuhLinks* $\wedge$ *SchuhLinks*)

Aktion(*SockeRechtsAnziehen*,
Vorbedingung: *keinSchuhRechts*,
Effekt: *SockeRechts*)

Aktion(*SockeLinksAnziehen*,
Vorbedingung: *keinSchuhLinks*,
Effekt: *SockeLinks*)

Aktion(*MüllEntsorgen*,
Vorbedingung: *SockeRechts* $\wedge$ *SockeLinks* $\wedge$ *SchuhRechts* $\wedge$ *SchuhLinks*,
Effekt: *MüllEntsorgt*)

Betrachten wir die Aktion *SockeRechtsAnziehen*. Eine Socke kann man nur anziehen, wenn man an dem entsprechenden Fuß noch keinen Schuh trägt. Dies wird

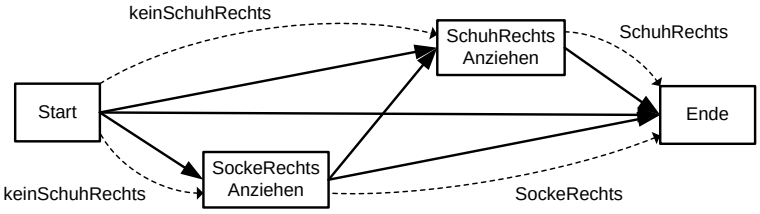


Abbildung 2.2: Plan P_1 zum Ziel *SockeRechts* $\wedge$ *SchuhRechts*

durch die Vorbedingung *keinSchuhRechts* zum Ausdruck gebracht. Hierzu ist anzumerken, dass nicht die Negation der Aussage *SchuhRechts* zur Formulierung dieser Vorbedingung verwendet werden kann, da, wie in Abschnitt 2.2.2 erläutert, in STRIPS bei Vorbedingungen nur positive Literale verwendet werden dürfen. Die Aktion *MüllEntsorgen* hat unter anderem die Vorbedingung, dass der Akteur Socken angezogen hat. Dies wurde für dieses Beispiel bewusst so festgelegt, um bestimmte Aspekte der KI-Planung zu verdeutlichen. Im Ausgangszustand S sind bis auf die Aktion *MüllEntsorgen* alle Aktionen anwendbar, da ihre Vorbedingungen erfüllt sind. Die Ausführung der Aktion *SchuhLinksAnziehen* beispielsweise würde den Ausgangszustand S in den Folgezustand S' = *keinSchuhRechts* $\wedge$ *SchuhLinks* überführen.

Das erste Ziel G_1, das wir betrachten, sei, dass der Akteur am rechten Fuß Socken und Schuhe trägt. Die formale Beschreibung dieses Ziels ist G_1 = *SockeRechts* $\wedge$ *SchuhRechts*. Das heißt, dass jeder Zustand, in dem die Aussagen *SockeRechts* und *SchuhRechts* wahr sind, ein Zielzustand ist. Durch die Beschreibung der zur Verfügung stehenden Aktionen, des Ausgangszustands und des Ziels ist ein Planungsproblem spezifiziert. Eine Lösung dieses Planungsproblems ist jeder Plan, der bei Ausführung den Ausgangszustand in einen Zielzustand überführt. In diesem Beispiel betrachten wir im speziellen Pläne als partiell geordnete Menge von Aktionen.

Abbildung 2.2 zeigt einen Plan P_1, der eine Lösung für das Planungsproblem darstellt. Die Rechtecke stehen für Aktionen und sind jeweils mit dem Namen der Aktion beschriftet. Neben den Aktionen *SockeRechtsAnziehen* und *SchuhRechtsAnziehen* enthält der Plan die beiden „Dummy“-Aktionen *Start* und *Ende*. Die Aktion *Start* besitzt keine Vorbedingungen und hat als Effekt den Ausgangszustand *keinSchuhRechts* $\wedge$ *keinSchuhLinks*. Die Aktion *Start* besitzt keine Effekte und hat als Vorbedingung das Ziel *SockeRechts* $\wedge$ *SchuhRechts*. Diese beiden Aktionen werden in der KI-Planung bei partiell geordneten Plänen üblicherweise eingeführt, um die Erstellung und Darstellung eines Planes zu vereinfachen. Die durchgezogenen Pfeile stellen *Ordnungsbedingungen* dar und bringen zum Ausdruck, in welcher Reihenfolge Aktionen ausgeführt werden müssen. Zum Beispiel bedeutet der durchgezogen Pfeile von der Aktion *SockeRechtsAnziehen* zur Aktion *SchuhRechtsAnziehen*, dass die Aktion *SockeRechtsAnziehen* irgendwann vor

der Aktion *SchuhRechtsAnziehen* ausgeführt werden muss, aber nicht unbedingt unmittelbar davor. Ein Sonderfall ist hierbei die Aktion *Start*, die in jedem Plan durch eine Ordnungsbedingung mit der Aktion *Ende* verbunden ist. Für den Plan P_1 existiert nur eine Linearisierung: die Sequenz der Aktionen *SockeRechtsAnziehen* und *SchuhRechtsAnziehen*. Die Ausführung der Aktion *SockeRechtsAnziehen* überführt den Ausgangszustand S in den Folgezustand S_1' = *keinSchuhRechts* $\wedge$ *keinSchuhLinks* $\wedge$ *SockeRechts*. Die Ausführung der Aktion *SchuhRechtsAnziehen* überführt anschließend den Zustand S_1' in den Folgezustand S_1'' = *SchuhRechts* $\wedge$ *keinSchuhLinks* $\wedge$ *SockeRechts*. Der Zustand S_1'' ist ein Zielzustand in Bezug auf das Ziel G_1, da S_1'' alle Atome in G_1 enthält.

Die gestrichelten Pfeile in Abbildung 2.2 stellen kausale Verknüpfungen dar. Eine *kausale Verknüpfung* von einer Aktion *A* zu einer Aktion *B* besteht genau dann, wenn die Aktion *A* einen Effekt hat, der eine Vorbedingung der Aktion *B* erzielt. Jede kausale Verknüpfung impliziert direkt eine Ordnungsbedingung: Eine kausale Verknüpfung von einer Aktion *A* zu einer Aktion *B* impliziert eine Ordnungsbedingung von *A* nach *B*. Auf diese Weise ist sichergestellt, dass der Effekt von *A* erzielt wird, bevor *B* ausgeführt wird. In der Abbildung sind die kausalen Verknüpfungen mit den Effekten annotiert, auf denen sie beruhen. Beispielsweise besteht eine kausale Verknüpfungen von der Aktion *Start* zu der Aktion *SockeRechtsAnziehen*, die auf dem Effekt *keinSchuhRechts* der Aktion *Start* beruht, der die entsprechende Vorbedingung der Aktion *SockeRechtsAnziehen* erzielt. Diese kausale Verknüpfung impliziert direkt die Ordnungsbedingung von der Aktion *Start* zu der Aktion *SockeRechtsAnziehen*. Des Weiteren kann eine kausale Verknüpfung zusätzliche Ordnungsbedingung erfordern, um die kausale Verknüpfung zu schützen. Beispielsweise darf zwischen den Aktionen *Start* und *SockeRechtsAnziehen* keine Aktion ausgeführt werden, die den Effekt *keinSchuhRechts* der Aktion *Start* negiert. Dies könnte beispielsweise durch die Aktion *SchuhRechtsAnziehen* geschehen. Aus diesem Grund muss eine Ordnungsbedingung eingefügt werden, die die Aktion *SchuhRechtsAnziehen* entweder vor die Aktionen *Start* oder hinter die Aktion *SockeRechtsAnziehen* ordnet. Das Vorziehen einer Aktion wird in diesem Zusammenhang als Promotion bezeichnet, das Verlagern nach hinten als Demotion. Da die Aktion *Start* den Anfang des Plans kennzeichnet, wird in dem Beispiel die zweite Alternative gewählt. Die Ordnungsbedingung von der Aktion *SockeRechtsAnziehen* zur Aktion *SchuhRechtsAnziehen* wird somit eingefügt, um die kausale Verknüpfung von der Aktion *Start* zu der Aktion *SockeRechtsAnziehen* zu schützen.

Zusammenfassend gliedern sich die insgesamt sechs Ordnungsbedingung des Plans P_1 in vier direkt implizierte Ordnungsbedingungen, eine Ordnungsbedingung zum Schutz einer kausalen Verknüpfung und der grundsätzlich vorhandenen Ordnungsbedingung zwischen den Aktionen *Start* und *Ende*. Des Weiteren ist anzumerken, dass der Plan P_1 nur eine von mehreren Lösungen des obigen Planungsproblems darstellt. Beispielsweise kann die Aktion *SockeRechtsAnziehen* auch mehrfach ausgeführt werden, bevor die Aktion *SchuhRechtsAnziehen* ausgeführt wird. Ein Planer richtet sich in diesem Zusammenhang in der Regel nach

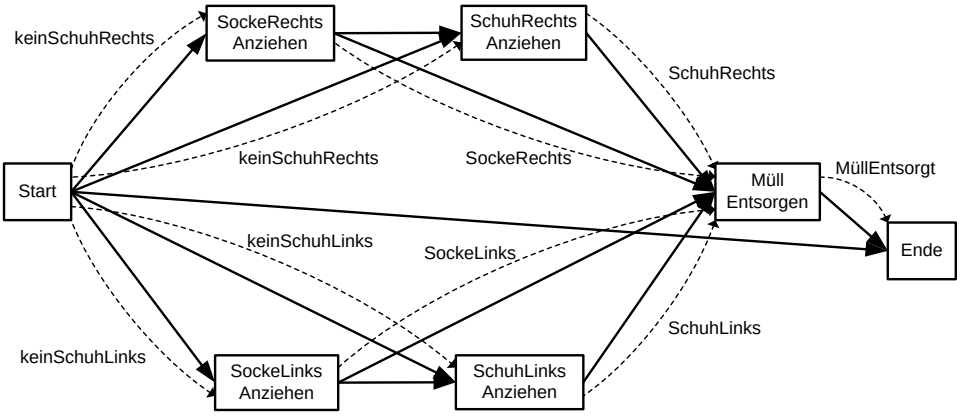


Abbildung 2.3: Plan P_2 zum Ziel *MüllEntsorgt*

Optimierungskriterien, um einen Plan auszuwählen. Klassische Planer arbeiten üblicherweise mit dem Optimierungskriterium, die Anzahl der Aktionen im Plan zu minimieren.

Betrachten wir als weiteres Beispiel ein Planungsproblem mit dem Ziel *MüllEntsorgt*. Der Ausgangszustand und die Domäne seinen die gleichen wie oben. Ein Plan P_2, der dieses Planungsproblem löst, ist in Abbildung 2.3 dargestellt. Der Plan P_2 besitzt sechs mögliche Linearisierungen. Eine Linearisierung ist zum Beispiel: *SockeRechtsAnziehen*, *SockeLinksAnziehen*, *SchuhLinksAnziehen*, *SchuhRechtsAnziehen*, *MüllEntsorgen*.

2.3 Zusammenfassung

In diesem Kapitel wurden grundlegende Konzepte, Komponenten, Sprachen und Artefakte der Disziplinen Workflow-Management und KI-Planung vorgestellt, die für eine Integration von Prozessplanung und -ausführung relevant sind. Es wurde dargestellt, welche Arten von Prozessbeschreibungssprachen im Workflow-Management Verwendung finden, um Prozesse in Form vom Prozessdefinitionen zu spezifizieren. Mit der Typisierung von Geschäftsfällen, wurde ein Konzept vorgestellt, wie eine Menge gleichstrukturierter Geschäftsfälle auf Basis der gleichen Prozessdefinitionen bearbeitet werden kann. Mit der Prozesssteuerung, dem Prozessmodellierungs- und dem Prozessüberwachungswerkzeug wurden drei für die Integration von Prozessplanung und -ausführung wesentlichen Komponenten eines Workflow-Management-Systems in ihrer Funktionalität und ihrem Zusammenspiel beschrieben.

In Bezug auf die KI-Planung wurden im Wesentlichen die Eingabe- und die Ausgabe-Artefakte eines Planers dargestellt und ihr Zusammenhang erklärt. Zunächst wurden mit STRIPS und PDDL zwei grundlegende Sprachen zur Spezifi-

kation von Planungsproblemen vorgestellt. Anschließend wurde erläutert, welche Arten von Plänen in der KI-Planung Verwendung finden.

Kapitel 3

Integration von Prozessplanung und -ausführung

In diesem Kapitel wird ein Konzept zur integrierten Prozessplanung und -ausführung vorgestellt, mit dem die in der Einleitung dargestellten Probleme klassischer Workflow-Management-Systeme gelöst werden. Die bereits im vorhergehenden Grundlagenkapitel erkennbar werdende konzeptionelle Entsprechung zwischen einem Plan in der KI-Planung und einer Prozessdefinition im Workflow-Management bildet dabei den zentralen Verknüpfungspunkt zwischen Planung und Ausführung.

Aufbauend auf den im letzten Kapitel beschriebenen Grundlagen aus den Bereichen Workflow-Management und KI-Planung, werden in Abschnitt 3.1 zunächst die Grundzüge des Konzepts zur integrierten Prozessplanung und -ausführung erläutert. Zur Illustration wird dabei das Beispiel einer Angebotsanfrage eingeführt, das auch in den folgenden Kapiteln Verwendung findet. Abschnitt 3.2 beschreibt die Planung individueller Prozessdefinitionen für jeden Geschäftsfall und grenzt diese Vorgehensweise von der in Abschnitt 2.1.2 erläuterten Typisierung von Geschäftsfällen ab, die bei klassischen Workflow-Management-Systemen üblich ist. In Abschnitt 3.3 werden die Besonderheiten der Planung in einer Geschäfts-Domäne diskutiert. Abschließend fasst Abschnitt 3.4 die wesentlichen Konzepte zur Integration von Prozessplanung und -ausführung zusammen.

3.1 Konzept

In dem vorhergehenden Kapitel wurde die grundlegende Arbeitsweise eines Workflow-Management-Systems und seine technische sowie organisatorische Einbettung in die Anwendungsumgebung vorgestellt. In der Einleitung wurden in diesem Zusammenhang zwei zentrale Probleme aufgezeigt:

- Falls eine Prozessdefinition zur Laufzeit aufgrund eines nicht vorhergesehenen Ereignisses geändert werden muss, ist ein manueller Eingriff durch einen Prozessmodellierer notwendig.

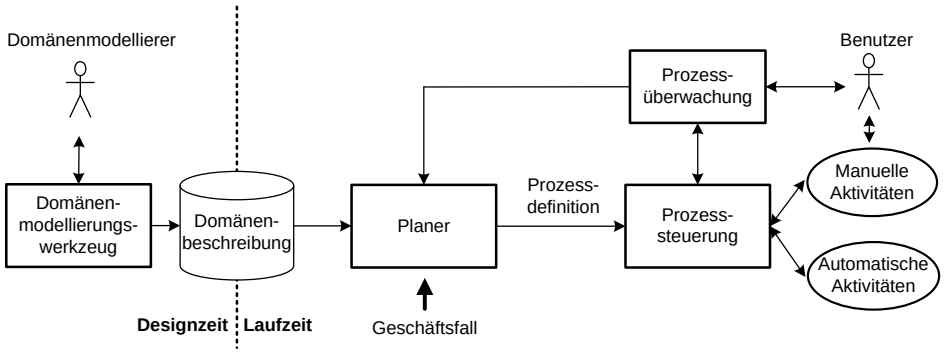


Abbildung 3.1: System zur integrierten Prozessplanung und -ausführung

- Da eine individuelle Planung für jeden Geschäftsfall zu aufwendig ist, werden Prozessdefinitionen im Voraus für Geschäftsfalltypen geplant und hierdurch Besonderheiten einzelner Geschäftsfälle gegebenenfalls nicht berücksichtigt.

Die gemeinsame Ursache dieser Probleme liegt darin, dass Planung und Ausführung sehr eng miteinander verwoben sind und klassische Workflow-Management-Systeme einseitig die Ausführung automatisieren. Die Idee zur Lösung ist hier die Automatisierung der Planung und darauf aufbauend die technische Integration von Planung und Ausführung. Im folgenden wird dieser Lösungsansatz zu einem Konzept für ein integriertes System zur Prozessplanung und -ausführung weiterentwickelt, das für jeden Geschäftsfall automatisch eine individuelle Prozessdefinition plant, diese ausführt, die Notwendigkeit einer Neuplanung erkennt, gegebenenfalls eine aktualisierte Prozessdefinition erzeugt und die Ausführung entsprechend angepasst.

Abbildung 3.1 gibt eine Übersicht über die Funktionsweise dieses Systems und seine organisatorische Einbettung. Das System besteht aus vier wesentlichen Komponenten: Planer, Prozesssteuerung, Prozessüberwachung und Domänenmodellierungswerkzeug. Die Prozesssteuerung ist für die Steuerung der Prozessausführung auf Basis einer Prozessdefinition verantwortlich. Sie besitzt somit die gleiche Funktionalität wie die entsprechende Komponente eines Workflow-Management-Systems, wie in Abschnitt 2.1.3 beschrieben. Der zentrale Unterschied besteht darin, dass Prozessdefinitionen nicht zur Designzeit manuell geplant werden. Es findet keine Geschäftsprozessmodellierung durch einen Prozessmodellierer statt. Stattdessen wird bei Ankunft eines Geschäftsfalls automatisch eine individuelle Prozessdefinition für diesen Geschäftsfall geplant. Dies ist die Aufgabe der *Planer*-Komponente. Zur Automatisierung der Planung kommen hierbei Planungstechniken der Künstlichen Intelligenz zum Einsatz. Grundlage für die Verknüpfung von Techniken der Disziplinen Workflow-Management und KI-Planung ist die Analogie zwischen einer Prozessdefinition im Workflow-Management und einem Plan

in der KI-Planung. Die Zuordnung einander entsprechender, fachspezifischer Entitäten zwischen Workflow-Management und KI-Planung wird detailliert im folgenden Abschnitt 3.1.1 besprochen. Wie in Abschnitt 2.2 erläutert, werden für die Planung ein Ausgangszustand, ein Ziel und eine Domänenbeschreibung benötigt. Ausgangszustand und Ziel charakterisieren den Geschäftsfall. Der Ausgangszustand ist die formale, logische Beschreibung des momentanen Zustands des Geschäftsfalls. Das Ziel beschreibt den gewünschten Zustand. Die Domänenbeschreibung spezifiziert die Ressourcen und Geschäftsregeln der Unternehmung. Zentraler Bestandteil dieser Domänenbeschreibung ist die Spezifikation der Aktivitäten, die für die Bearbeitung von Geschäftsfällen zur Verfügung stehen. Die Domänenbeschreibung wird von einem Domänenmodellierer mit Unterstützung eines Domänenmodellierungswerkzeug erzeugt. Im Gegensatz zu einem klassischen Workflow-Management-System wie in Abbildung 2.1 auf Seite 10 dargestellt, bildet bei der integrierten Prozessplanung und -ausführung nicht die Menge der Prozessdefinitionen sondern die Domänenbeschreibung den Übergang von Designzeit zur Laufzeit. Da Prozessdefinitionen erst nach dem Eintreffen konkreter Geschäftsfälle geplant werden, sind der Planer und die Prozessdefinitionen der Laufzeit zuzuordnen. Die individuelle Planung auf Basis eines konkreten Geschäftsfalls ermöglicht die Berücksichtigung von Besonderheiten einzelner Geschäftsfälle, ohne dass diese Besonderheiten bei der Typisierung von Geschäftsfällen wie in Abschnitt 2.1.2 beschrieben, antizipiert werden müssen. Somit ist eines der in der Einleitung dargestellten, zentralen Probleme klassischer Workflow-Management-Systeme gelöst. Abschnitt 3.2 beschreibt die individuelle Planung im Detail.

Des Weiteren erlaubt die integrierte Prozessplanung und -ausführung eine automatisierte Prozessanpassung zur Laufzeit. Folglich ist die Verfügbarkeit eines Prozessmodellierers zur Laufzeit nicht erforderlich, sogar wenn ein Prozess zur Berücksichtigung unerwarteter Ereignisse angepasst werden muss. Damit aufgrund von nicht vorhergesehenen Ereignissen automatisch eine Prozessanpassung durchgeführt werden kann, muss zunächst die Notwendigkeit der Prozessanpassung automatisch erkannt werden. Aus diesem Grund besitzt das in Abbildung 3.1 dargestellte System zur integrierten Prozessplanung und -ausführung anstelle eines Prozessüberwachungswerkzeugs eine aktive *Prozessüberwachung*. Diese erlaubt dem Benutzer eine manuelle Überwachung des Systems, führt diese Überwachung aber auch automatisch selbst durch. Hierzu werden nach der Beendigung einer Aktivität die tatsächlichen Effekte mit den spezifizierten Effekten verglichen. Voraussetzung hierfür ist, dass die tatsächlichen Effekte einer Aktivität und somit die Änderungen des Zustands des Geschäftsfalls überprüft werden können. Sobald eine Abweichung erkannt wird, die eine Prozessanpassung notwendig macht, wird der Planer mit der Planung einer neuen Prozessdefinition beauftragt. Hierzu erhält der Planer als neuen Ausgangszustand den aktuellen Zustand des Geschäftsfalls. Der Planer gibt die neue Prozessdefinition an die Prozesssteuerung weiter, die die Ausführung entsprechend anpasst. Wesentlich ist hierbei, dass kein Modellierungsexperte zur Laufzeit benötigt wird, da die Domänenbeschreibung unverändert bleibt. Die automatisierte Prozessanpassung wird im Detail in Kapitel 4 erläutert.

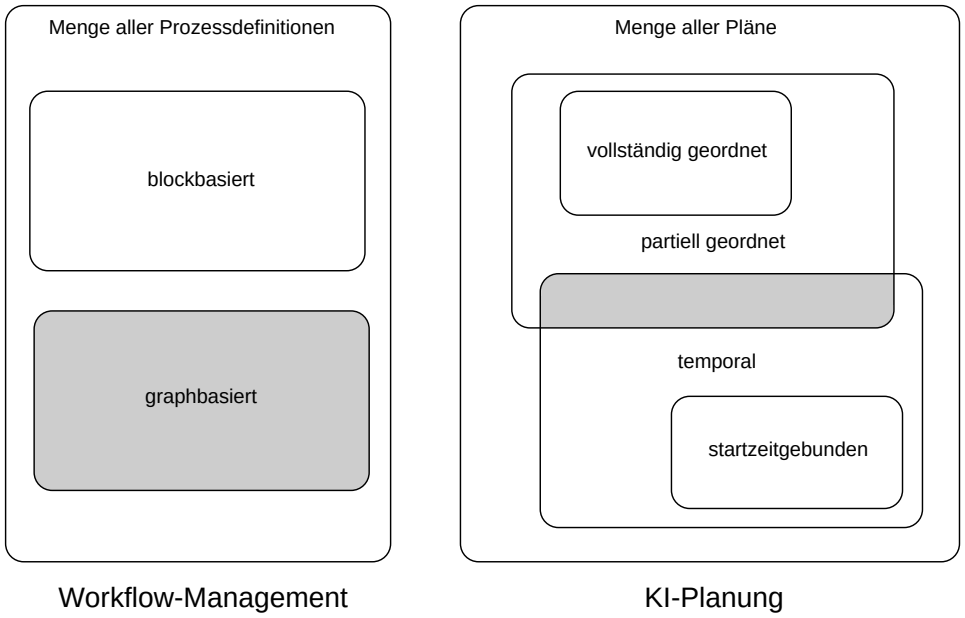


Abbildung 3.2: Analogie zwischen Prozessdefinition und Plan

3.1.1 Zuordnung fachspezifischer Entitäten

Wie im vorhergehenden Abschnitt erläutert, müssen für die Realisierung einer integrierten Prozessplanung und -ausführung Konzepte und Techniken der Disziplinen Workflow-Management und KI-Planung verknüpft werden. Das zentrale Element dieser Verknüpfung bildet die Analogie zwischen einer Prozessdefinition im Workflow-Management und einem Plan in der KI-Planung. In diesem Abschnitt werden diese und weitere Analogien zwischen den im Grundlagenkapitel vorgestellten Konzepten des Workflow-Managements und der KI-Planung identifiziert und beschrieben.

Im Workflow-Management wird ein logisch zusammengehöriger Arbeitsschritt als Aktivität bezeichnet. Das entsprechende Konzept in der KI-Planung ist die Aktion. Im Workflow-Management legt eine Prozessdefinition fest, in welcher Reihenfolge welche Aktivitäten auszuführen sind. Analog hierzu legt in der KI-Planung ein Plan fest, in welcher Reihenfolge welche Aktionen auszuführen sind, um den Ausgangszustand in einen Zielzustand zu überführen. Hierbei ist zu beachten, dass es sowohl im Workflow-Management unterschiedliche Arten von Prozessdefinitionen als auch in der KI-Planung unterschiedliche Arten von Plänen gibt. Abbildung 3.2 zeigt die Mengenbeziehungen der im Grundlagenkapitel beschriebenen Arten von Prozessdefinition und Plänen. Die grauen Flächen stellen dabei die Mengen der einander entsprechenden Prozessdefinitionen beziehungsweise Plänen dar. Eine direkte Entsprechung gibt es nur zwischen einer graphbasierten Prozessdefinition im Workflow-Management und einem temporalen, partiell geord-

Workflow-Management	**KI-Planung**
Aktivität	Aktion
graphbasierte Prozessdefinition	partiell geordneter, temporaler Plan
graphbasierte Prozessdefinition mit alternativem Kontrollfluss	bedingter, partiell geordneter, temporaler Plan
Kontrollkonnektor	Ordnungsbedingung
Datenkonnektor	kausale Verknüpfung basierend auf Datenfluss

Tabelle 3.1: Analoge Entitäten aus Workflow-Management und KI-Planung

neten Plan in der KI-Planung. In der Abbildung überschneiden sich die Mengen der blockbasierten und graphbasierten Prozessdefinitionen nicht. An dieser Stelle wird von Prozessdefinitionen basierend auf hybriden Prozessbeschreibungssprachen wie zum Beispiel der Business Process Execution Language for Web Services (BPEL4WS) [78] abstrahiert, die sowohl blockbasierte als auch graphbasierte Elemente enthalten. Für die blockbasierten Elemente dieser Prozessdefinitionen gibt es keine Entsprechung in der KI-Planung.

Bei der Analogie zwischen einer graphbasierten Prozessdefinition und einem temporalen, partiell geordneten Plan in der KI-Planung entsprechen die Kontrollkonnektoren einer Prozessdefinition den Ordnungsbedingungen eines Plans. Eine Prozessdefinition mit alternativem Kontrollfluss im Workflow-Management entspricht dabei einem bedingten Plan in der KI-Planung. Datenkonnektoren einer Prozessdefinition spiegeln sich in kausalen Verknüpfungen in einem Plan wider. Eine derartige kausale Verknüpfung bringt zum Ausdruck, dass eine Aktion als Effekt Daten bereitstellt, deren Verfügbarkeit eine Vorbedingung einer anderen Aktion ist. Das heißt bei Datenabhängigkeiten zwischen zwei Aktionen gibt es eine direkte Entsprechung zwischen Datenkonnektor und kausaler Verknüpfung. Allerdings kann es auch kausale Verknüpfungen geben, die keine Entsprechung in einem Datenkonnektor besitzen. Dies ist der Fall, wenn die Abhängigkeit zwischen zwei Aktionen nicht auf Datenfluss basiert. Ist beispielsweise eine Aktion A das Verpacken eines Artikels und eine Aktion B sein Versand, so wird in einem Plan eine kausale Verknüpfung von A nach B sein, da A einen Effekt besitzt der Vorbedingung von B ist. Eine entsprechende Prozessdefinition besitzt jedoch keinen Datenkonnektor von A nach B, da kein Datenfluss zwischen den Aktionen stattfindet. Die Tabelle 3.1 gibt einen zusammenfassenden Überblick über die analogen Konzepte aus den Disziplinen Workflow-Management und KI-Planung. Entsprechend ihrer analogen Bedeutung in den jeweiligen Disziplinen werden im Folgenden die Begriffe Aktivität und Aktion, sowie Kontrollkonnektor und Ordnungsbedingung synonym verwendet. Mit einer Prozessdefinition wird, soweit nicht anders spezifiziert, eine graphbasierte Prozessdefinition bezeichnet; mit einem Plan entsprechend ein partiell geordneter, temporaler Plan.

3.1.2 Grundlegende Annahmen

Für diese Arbeit treffen wir eine Reihe von grundlegenden Annahmen:

- A1: Die Effekte einer Aktivität treten bei der Beendigung ihrer Ausführung ein.
- A2: Am Ende der Ausführung einer Aktivität lässt sich der Folgezustand feststellen.
- A3: Es gilt die Annahme der geschlossenen Welt. Das heißt, dass alle Aussagen, die für einen Zustand nicht explizit als wahr spezifiziert sind, werden als falsch angenommen.
- A4: Jede Aktivität besitzt eine endliche Ausführungsdauer. Die Verfügbarkeit von Ressourcen für die Ausführung einer Aktivität ist stets gegeben.
- A5: Die Ausführung von Aktivitäten kann im Allgemeinen nicht unterbrochen werden.
- A6: Zu einem gegebenen Planungsproblem wird immer eine Prozessdefinition gefunden, die das Planungsproblem löst, falls eine solche Prozessdefinition existiert.
- A7: Die möglichen Ausführungszustände und Zustandsübergänge einer Aktivitätsinstanz sind in dem Zustandsautomaten in Abbildung 3.3 dargestellt.

Annahme A2 bringt zum Ausdruck, dass sich die Effekte einer Aktivität bei ihrer Beendigung überprüfen lassen. Dies bedeutet, dass entweder die Aktivität Informationen darüber zurückliefert welche Effekte sie tatsächlich hatte, oder eine andere Möglichkeit besteht, den aktuellen Zustand des Geschäftsfalls festzustellen. Diese Annahme ist wichtig für die automatische Überwachung einer Prozessausführung.

Entsprechend der Annahme A4 wird in dieser Arbeit die Verfügbarkeit von personellen oder maschinellen Ressourcen für die Ausführung einer Aktivität nicht betrachtet. Zur Zuordnung von Ressourcen für die Ausführung einer Aktivität können Techniken aus dem Workflow-Management, wie beispielsweise eine dynamische Rollenauflösung [68], zum Einsatz kommen. Diese Techniken lassen sich problemlos mit dem hier vorgestellten Konzept zur integrierten Prozessplanung und -ausführung kombinieren. Eine Berücksichtigung von Ressourcen bei der Planung von Prozessdefinition ist zwar prinzipiell möglich, wird hier aber nicht betrachtet.

Die Mittel zur Lösung von Planungsproblemen werden entsprechend Annahme A6 in dieser Arbeit als gegeben vorausgesetzt. An dieser Stelle wird auf bestehende Techniken und softwaretechnische Realisierungen im Wissenschaftsgebiet der KI-Planung verwiesen. Für diese Arbeit spielt zwar der Zusammenhang von Planungsproblem und Prozessdefinition eine wichtige Rolle, der Weg, wie diese Prozessdefinition gefunden wird, ist jedoch nicht relevant.

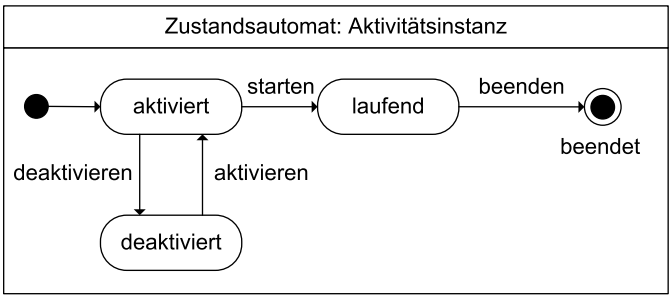


Abbildung 3.3: Zustände einer Aktivitätsinstanz

In dieser Arbeit wird entsprechend der Annahme A7 ein sehr einfaches Zustandsmodell für Aktivitätsinstanzen verwendet, das in Abbildung 3.3 als Zustandsautomat in der Unified Modeling Language (UML) [19] dargestellt ist. Dabei wird von Fehler- und Unterbrechungszuständen wie *abgebrochen* oder *unterbrochen* abstrahiert.

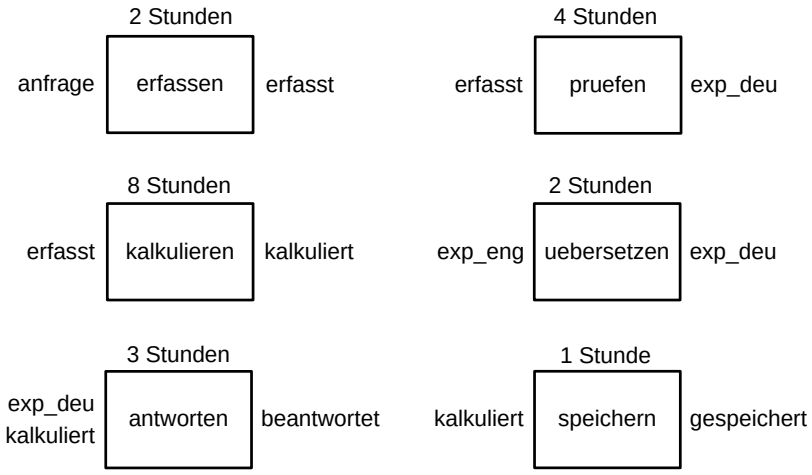
3.1.3 Beispiel

Betrachten wir zur Illustration der integrierten Prozessplanung und -ausführung das Beispiel der Bearbeitung einer Angebotsanfrage. Hierbei übermittelt ein Kunde einem Unternehmen eine Angebotsanfrage nach einem bestimmten Produkt und erwartet eine Antwort in Form eines Angebots oder einer Mitteilung, dass das Produkt nicht lieferbar ist. Die Bearbeitung der Angebotsanfrage wird hier zum Zweck der Übersichtlichkeit stark vereinfacht dargestellt. Abbildung 3.4 zeigt die Domänenbeschreibung für dieses Beispiel. Im Gegensatz zur Beispieldomäne in Abschnitt 2.2.4 verwenden wir hier eine graphische Notation zur Spezifikation der Aktivitäten. Ein weiterer Unterschied ist die Berücksichtigung von Zeit und der Ausführungsdauer von Aktivitäten. Bezüglich der Vorbedingungen und Effekte gilt, dass die Vorbedingungen einer Aktivität für ihre gesamte Ausführungsdauer gelten müssen und dass die Effekte bei der Beendigung der Aktivität eintreten. Für dieses Beispiel beschränkt sich die Spezifikation der Vorbedingungen und Effekte auf aussagenlogische Symbole, denen ihre Bedeutung in der Legende der Abbildung zugeordnet wird. Die Vorbedingungen beziehungsweise Effekte werden jeweils auf der linken beziehungsweise rechten Seite der betreffenden Aktivität aufgelistet und sind als Konjunktion zu interpretieren. Auf eine ausdrucksstarke Art der Modellierung einer Domäne wurde hier zugunsten einer leichteren Verständlichkeit des Beispiels verzichtet. Die in dieser Arbeit vorgestellten Konzepte lassen sich jedoch ebenso mit ausdrucksstarken Modellierungssprachen wie PDDL oder F-Logik [58] umsetzen.

Der aktuelle Zustand des Geschäftsfalls sei, dass eine Nachricht mit einer Angebotsanfrage eingegangen ist. Das heißt, dass die Aussage *anfrage* gilt. Das Ziel des Geschäftsfalls sei, dass die Angebotsanfrage beantwortet ist. Das heißt, dass die

Aussage *beantwortet* gilt. Die Domänenbeschreibung und der Geschäftsfall stellen zusammen ein Planungsproblem dar, und bilden somit, wie in Abbildung 3.1 dargestellt, die Eingabe für den Planer. Dieser generiert die in Abbildung 3.5 gezeigte Prozessdefinition P. Die Rechtecke stellen Aktivitäten dar. Durchgezogene Pfeile repräsentieren Kontrollkonnektoren. Damit die Abbildung 3.5 übersichtlich ist, wurde hier, wie auch bei den folgenden Abbildungen, auf die Darstellung der „Dummy"-Aktivitäten *Start* und *Ende* verzichtet. Durch die Berücksichtigung der Ausführungsdauer von Aktivitäten ist die Prozessdefinition P anders zu interpretieren, als die in Abbildung 2.3 auf Seite 21 gezeigte Prozessdefinition P_2. P stellt im Gegensatz zu P_2 einen temporalen Plan dar. Ein Kontrollkonnektor in Abbildung 3.5 bringt somit zum Ausdruck, dass die Aktivität am Ursprung des Konnektors beendet sein muss, bevor die Aktivität am Ziel des Konnektors gestartet werden kann. Während, wie in Abschnitt 2.2.4 beschrieben, durch die Prozessdefinition P_2 alternative, sequentielle Ausführungen beschreiben wurden, beschreibt P auch die Möglichkeit zu nebenläufigen Ausführung. Beispielsweise können in P die Aktivitäten *pruefen* und *kalkulieren* nebenläufig ausgeführt werden. Das heißt, dass sie sich in ihrer Ausführung zeitlich überlappen können. In Abschnitt 2.2.3 wird dieser Unterschied zwischen klassischen partiell geordneten Plänen und temporalen Plänen detailliert erläutert.

Die Vorbedingungen der Aktivität *erfassen* sind bereits im Ausgangszustand erfüllt. Somit ist die Aktivität *erfassen* direkt zu Beginn des Prozesses anwendbar. Die Ausführung der Aktivität *erfassen* hat den Effekt *erfasst*, der eine Vorbedingung der Aktivität *kalkulieren* ist. Es besteht demnach eine kausale Verknüpfung zwischen *erfassen* und *kalkulieren*. Kausale Verknüpfungen sind in der Abbildung durch gestrichelte Pfeile dargestellt und mit den Effekten annotiert, auf denen die kausale Verbindung basiert. Eine kausale Verknüpfung impliziert eine Ausführungsreihenfolge, die sich im entsprechenden Kontrollkonnektor widerspiegelt. Zur Ausführung leitet der Planer die Prozessdefinition an die Prozesssteuerung weiter. Die Prozessdefinition wird dann instanziiert und gestartet. Nach der Instanziierung befinden sich alle Aktivitätsinstanzen entsprechend dem Zustandsautomat aus Abbildung 3.3 im Ausführungszustand *aktiviert*. Als erste Aktivität wird die Aktivität *erfassen* gestartet, da sie keinen eingehenden Kontrollkonnektor besitzt. Bei der Beendigung der Aktivität ändert sich der Zustand des Geschäftsfalls. In dem Folgezustand gelten die Aussagen *anfrage* und *erfasst*, die die Effekte der Aktivität *erfassen* sind. Damit sind die Vorbedingungen sowohl der Aktivität *pruefen* als auch der Aktivität *kalkulieren* erfüllt. Entsprechend des Kontrollflusses werden diese beiden Aktivitäten als nächstes gestartet. Die entsprechende Prozessinstanz in Ausführung ist in Abbildung 3.6 dargestellt. Für jede Aktivität ist in Klammern der Ausführungszustand der Aktivität vermerkt. Die beiden nebenläufigen Aktivitäten *pruefen* und *kalkulieren* befinden sich in Ausführungszustand *laufend*. Laut Spezifikation hat die Aktivität *pruefen* den Effekt *exp_deu*. Dieser Effekt bildet die kausale Verknüpfung von der Aktivität *pruefen* und der Aktivität *antworten*. Sobald die beiden Aktivitäten *pruefen* und *kalkulieren* beendet sind, kann die Aktivität *antworten* starten. Sobald die Aktivität *antworten* endet, tritt ihr



Legende:

Ausführungsdauer

Vorbedingungen | Name der Aktivität | Effekte

Symbol	Bedeutung
anfrage	Eine Nachricht mit einer Angebotsanfrage ist eingegangen.
erfasst	Die Angebotsanfrage ist erfasst.
exp_deu	Die Exportbedingungen für das Produkt liegen in Deutsch vor.
exp_eng	Die Exportbedingungen für das Produkt liegen in Englisch vor.
kalkuliert	Der Preis für das Produkt ist kalkuliert.
beantwortet	Die Angebotsanfrage ist beantwortet.
gespeichert	Die Kalkulation ist gespeichert.

Abbildung 3.4: Beispieldomäne

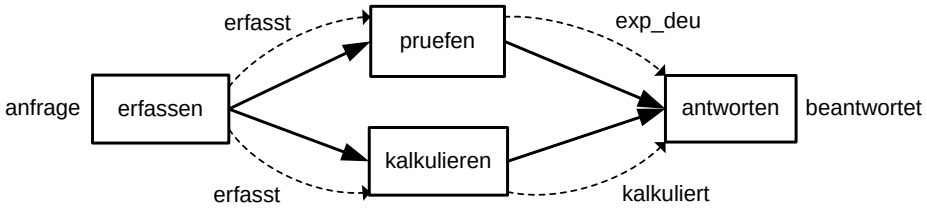


Abbildung 3.5: Prozessdefinition zur Angebotsanfrage

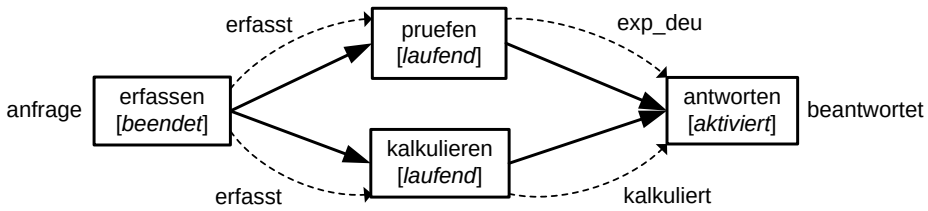


Abbildung 3.6: Prozessinstanz zur Angebotsanfrage in Ausführung

Effekt *beantwortet* ein und überführt damit den Zustand des Geschäftsfalls in einen Zielzustand.

3.2 Individuelle Planung

Bei klassischen Workflow-Management-Systemen wird, wie in Abschnitt 2.1.2 beschrieben, eine Prozessdefinition nicht für einen einzelnen Geschäftsfall modelliert, sondern für eine Menge ähnlicher Geschäftsfälle, die zu einem Geschäftsfalltyp zusammengefasst werden. Dieses Vorgehen ist im hohen Aufwand für eine manuelle Modellierung einer Prozessdefinition begründet. Beispielsweise wird nicht für jede einzelne Angebotsanfrage eine Prozessdefinition modelliert, sondern nur eine Prozessdefinition für alle Geschäftsfälle des Geschäftsfalltyps Angebotsanfrage. Individuelle Unterschiede zwischen einzelnen Geschäftsfällen eines Geschäftsfalltyps werden bei der Modellierung in Form von alternativen Ausführungspfaden berücksichtigt. Voraussetzung für die Modellierung einer Prozessdefinition für einen Geschäftsfalltyp ist die Möglichkeit, zukünftige Geschäftsfälle zu antizipieren. Dies ist notwendig, da zum Modellierungszeitpunkt der Prozessdefinition die Geschäftsfälle, die mit der Prozessdefinition bearbeitet werden sollen, noch nicht vorliegen. Aus obigen Erläuterungen lässt sich ableiten, dass die Verwendung einer Prozessdefinition für eine Menge ähnlicher Geschäftsfälle adäquat ist, falls die Struktur der Geschäftsfälle sehr homogen beziehungsweise antizipierbar ist. Ist dies nicht der Fall, ergeben sich folgende Probleme:

1. Eine Prozessdefinition für einen Geschäftsfalltyp besitzt unter Umständen mehr alternative Ausführungspfade als für die Bearbeitung eines konkreten Geschäftsfalls notwendig. Folglich ist sie unübersichtlicher als eine individuelle Prozessdefinition und ein konkreter Geschäftsfall schwerer zu überwachen.

2. Sind Besonderheiten der Geschäftsfälle nicht vorhersehbar, so ist es nicht möglich, im Voraus eine Prozessdefinition zu finden, mit der sich alle Geschäftsfälle bearbeiten lassen.

Hier stößt das Einsatzgebiet von klassischen Workflow-Management-Systemen an seine Grenzen, da eine manuelle, individuelle Planung zu aufwendig ist.

Bei einer integrierten Prozessplanung und -ausführung werden Prozessdefinitionen automatisch durch einen Planer generiert. Dies bedeutet, dass die Planung einer Prozessdefinition jederzeit, schnell und mit geringem Aufwand durchgeführt werden kann. Hierdurch wird es möglich, individuelle Prozessdefinitionen für jeden einzelnen Geschäftsfall zu planen. Die Planung erfolgt dabei im Gegensatz zu klassischen Workflow-Management-Systemen erst nach dem Eintreffen eines konkreten Geschäftsfalls. Folglich ist es nicht notwendig, Besonderheiten von Geschäftsfällen zu antizipieren, wie dies bei der Typisierung von Geschäftsfällen erforderlich ist. Stattdessen werden Besonderheiten eines Geschäftsfalls automatisch bei der Planung berücksichtigt und führen zu einer individuellen auf den konkreten Geschäftsfall zugeschnittenen Prozessdefinition. Da bei einer individuellen Planung nur die tatsächlichen Gegebenheiten eines vorliegenden Geschäftsfalls und keine potentiellen Besonderheiten berücksichtigt werden müssen, besitzt die resultierende Prozessdefinition keine alternativen Ausführungspfade, die für den vorliegenden Geschäftsfall nicht relevant sind. Daher ist eine individuelle Prozessdefinition in der Regel einfacher strukturiert und somit leichter verständlich als eine entsprechende Prozessdefinition für einen Geschäftsfalltyp, mit der alle Geschäftsfälle eines bestimmten Geschäftsfalltyps bearbeitet werden sollen.

Zur Illustration der individuellen Planung von Geschäftsfällen kommen wir auf das Beispiel der Angebotsanfrage aus Abschnitt 3.1.3 zurück. Dort wurde der folgende Geschäftsfall betrachtet: Der aktuelle Zustand ist, dass eine Nachricht mit einer Angebotsanfrage eingegangen ist, das heißt, die Aussage *anfrage* gilt. Das Ziel ist, dass die Angebotsanfrage beantwortet ist. Das heißt, dass die Aussage *beantwortet* gilt. Die entsprechende Prozessdefinition zur Bearbeitung dieses Geschäftsfalls ist in Abbildung 3.5 dargestellt.

Angenommen beim Eintreffen einer Angebotsanfrage liegen die Exportbedingungen für das gewünschte Produkt bereits vor, da für dieses Produkt eine Prüfung der Exportbedingungen für das betreffende Land schon in einem vorherigen Geschäftsfall stattgefunden hat oder der Kunde die Exportbedingungen bereits recherchiert hat. In diesem Fall wird der aktuelle Zustand des Geschäftsfalls durch die Aussagen *anfrage* und *exp_deu* beschrieben. Das Ziel ist wieder, dass die Aussage *beantwortet* gilt. Eine Prozessdefinition zur Bearbeitung dieses Geschäftsfalls ist in Abbildung 3.7 dargestellt. Im Gegensatz zur Prozessdefinition in Abbildung 3.5 ist die Aktivität *pruefen* nicht enthalten. Diese Aktivität wird nicht benötigt, da die Vorbedingung *exp_deu* der Aktivität *antworten* bereits im Ausgangszustand erfüllt ist.

Betrachten wir als nächstes den Fall, dass bei einer Angebotsanfrage eine zusätzliche Anforderung ist, dass das Ergebnis der Kalkulation gespeichert wird. In diesem Fall ist der aktuelle Zustand des Geschäftsfalls der gleiche wie in dem Ausgangsbeispiel in Abschnitt 3.1.3: Es gilt die Aussage *anfrage*. Allerdings ist das Ziel in diesem Fall, dass nicht nur die Aussage *beantwortet* gilt sondern auch die Aussage *gespeichert*. Hiermit ergibt sich bei der Planung die in Abbildung 3.8 dargestellte Prozessdefinition. Im Gegensatz zur Prozessdefinition in Abbildung 3.5 ist die zusätzliche Aktivität *speichern* enthalten. Diese hat die Vorbedingung *kal-*

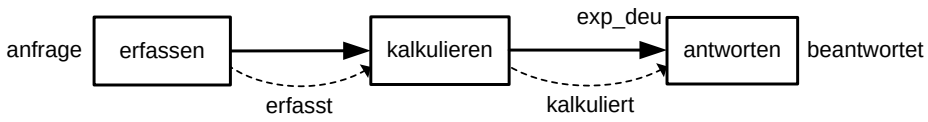


Abbildung 3.7: Angebotsanfrage ohne Exportprüfung

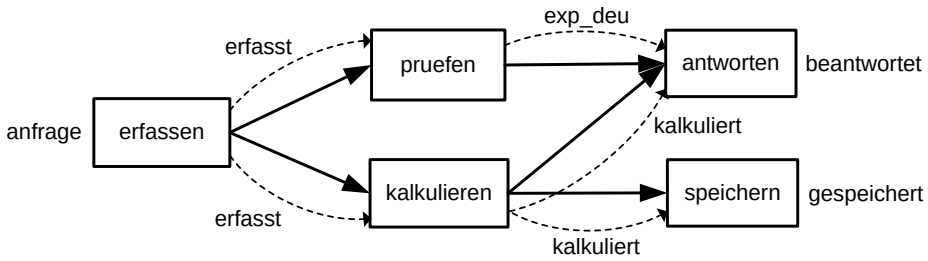


Abbildung 3.8: Angebotsanfrage mit Speicherung der Kalkulation

kuliert. Daher muss die Aktivität *speichern* in jedem Fall nach der Aktivität *kalkulieren* ausgeführt werden.

Die Beispiele zeigen, wie Besonderheiten von Geschäftsfällen bei einer integrierten Prozessplanung und -ausführung automatisch durch die individuelle Planung berücksichtigt werden. Die Besonderheiten eines Geschäftsfalls spiegeln sich dabei in seinem aktuellen Zustand oder in seinem Ziel wider. Bei einer Typisierung von Geschäftsfällen, wie sie bei klassischen Workflow-Management-Systemen üblich ist, müssen diese Besonderheiten antizipiert und in Form von alternativen Ausführungspfaden in einer Prozessdefinition berücksichtigt werden. Auf diese Weise enthält die Prozessdefinition sämtliche Informationen zur Handhabung aller Geschäftsfälle eines bestimmten Geschäftsfalltyps. Tritt jedoch bei einem Geschäftsfall eine Besonderheit auf, die nicht vorhergesehen wurde, so kann der entsprechende Geschäftsfall mit dieser Prozessdefinition nicht bearbeitet werden. In diesem Fall wird ein manueller Eingriff notwendig.

Alternative Ausführungspfade sind in einer individuellen Prozessdefinition nur dann notwendig, wenn sich Informationen erst zur Laufzeit des Prozesses ergeben, die für die Art und Weise der Bearbeitung des Geschäftsfalls entscheidend sind. Da diese Informationen zum Planungszeitpunkt noch nicht vorliegen, müssen alle relevanten Möglichkeiten in Form alternativen Ausführungspfaden berücksichtigt werden. Betrachten wir hierzu das Beispiel der Angebotsanfrage. Angenommen es soll im Anschluss an die Kalkulation eine Prüfung ihres Ergebnisses stattfinden, falls die kalkulierten Kosten 5000 € übersteigen. Bei einer eintreffenden An-

gebotsanfrage ist Höhe der zu kalkulierenden Kosten noch unbekannt und kann somit nicht zur Planung verwendet werden. Daher müssen in der Prozessdefinition der Aktivität *kalkulieren* alternative Ausführungspfade folgen: Ein Pfad der die Prüfungs-Aktivität enthält und durchlaufen wird, wenn die kalkulierten Kosten 5000 € übersteigen und ein zweiter Pfad, der die keine Prüfungs-Aktivität enthält und durchlaufen wird, wenn die kalkulierten Kosten 5000 € nicht übersteigen.

3.3 Planung von Geschäftsprozessen

In der KI wurden viele unterschiedliche Sprachen zur Spezifikation von Planungsproblemen und darauf basierende Planer entwickelt. Die Vielzahl der Sprachen ist insbesondere darin begründet, dass ein Widerspruch zwischen Ausdrucksstärke der Sprache und Effizienz der Planung existiert. Für jedes Anwendungsgebiet gilt es, eine Sprache zu finden, die ausdrucksstark genug ist, um eine große Vielfalt an Problemen zu beschreiben, aber einschränkend genug, um effiziente Algorithmen dafür zuzulassen. Dementsprechend ist nicht jede Sprache für jedes Anwendungsgebiet geeignet. In diesem Abschnitt wird diskutiert, welche Spracheigenschaften für die Planung von Geschäftsprozessen benötigt werden, beziehungsweise wünschenswert sind und in welchen Zusammenhang dies zu der Art des erzeugten Plans steht.

Für den Einsatz von KI-Planungstechniken bei der Planung von Geschäftsprozessen müssen alle in diesem Anwendungsgebiet relevanten Aspekte spezifiziert, bei der Planung berücksichtigt und im resultierenden Plan dargestellt werden können. In Geschäftsprozessen spielen monetäre Werte wie die Kosten zur Ausführung einer Aktivität und andere numerische Werte eine wichtige Rolle. Ohne die Berücksichtigung dieser Werte kann keine sinnvolle Planung stattfinden. Aus diesem Grund muss eine Sprache zur Definition von Planungsproblemen in einer Geschäftsdomäne arithmetische Funktionen und eine zielgerichtete Optimierung numerischer Werte unterstützen. Ein Beispiel hierfür wäre die Optimierung der Gesamtkosten eines Prozesses. Hierzu müssen sich die Gesamtkosten als Summe der Kosten für die Ausführung aller Aktivitäten des Prozesses spezifizieren lassen und die Gesamtkosten als zu minimierender Wert festlegen lassen.

Die Ausführungsdauer einer Aktivität lässt sich im diesem Zusammenhang zwar auch als numerischer Wert darstellen, sie nimmt aber in der KI-Planung eine besondere Rolle ein. Im Gegensatz zu beispielsweise monetären Werten kann die Gesamtausführungsdauer eines Prozesses nicht einfach als Summe der Ausführungsdauer aller Aktivitäten des Prozesses aufgefasst werden. Dies gilt nur für rein sequentielle Prozesse, die vollständig geordneten Plänen in der KI-Planung entsprechen. Komplizierter wird dies, wenn, wie es für eine Geschäftsdomäne typisch ist, auch eine nebenläufige Ausführung von Aktivitäten möglich sein soll. Die Unterstützung der Spezifikation und Optimierung von Nebenläufigkeit und Ausführungsdauer wird in der KI-Planung als *temporales Planen* bezeichnet. In Abschnitt 2.2.3 wurde bei temporalen Plänen zwischen partiell geordneten Plä-

nen und startzeitgebundenen Plänen unterschieden. Wie bereits in Abschnitt 3.1.1 dargestellt, existiert in der Geschäftsdomäne eine direkte Entsprechung zwischen einer graphbasierten Prozessdefinition im Workflow-Management und einem temporalen, partiell geordneten Plan in der KI-Planung.

Des Weiteren lassen sich in einer Geschäftsdomäne nicht immer alle Effekte von Aktivitäten genau vorhersagen. Betrachten wir beispielsweise die Aktivität *unzuverlässiger Versand*. Diese kann den Effekt haben, dass der Geschäftspartner die Sendung erhält oder auch nicht. Auf der anderen Seite stehen in einer Geschäftsdomäne häufig Aktivitäten zur Verfügung, die es erlauben während der Prozessausführung Informationen über den Zustand des Geschäftsfalls zu sammeln. Denkbar wäre zum Beispiel eine Aktivität *Nachfrage Geschäftspartner*, die die Information liefert, ob die Sendung angekommen ist. Derartige Aktivitäten werden in der KI-Planung auch als *Sensing Actions* bezeichnet. Das Ergebnis von Sensing Actions kann dann benutzt werden, um zwischen alternativen Ausführungspfaden im Plan auszuwählen. In der KI-Planung wird dies als *Kontingenzplanen* [82] bezeichnet. Ein entsprechender Plan mit alternativen Ausführungspfaden wird als *bedingter Plan* bezeichnet.

Neben einer ausreichenden Ausdrucksmächtigkeit der Sprache ist es auch wünschenswert, dass eine einfache Spezifikation von Planungsproblemen möglich ist. Eine Beschränkung auf Aussagenlogik erscheint in einer Geschäftsdomäne als unangemessen, da insbesondere Quantoren und Variablen häufig hilfreich bei der Spezifikation eines Sachverhaltes sind. Möchte man beispielsweise ausdrücken, dass eine Aktivität zur Verfügung steht, die jedes beliebige Dokument von Englisch nach Deutsch übersetzen kann, so lässt sich über eine Variable der Zusammenhang herstellen, dass es sich bei dem resultierenden Dokument um das identische Dokument in einer anderen Sprache handelt. Auf diese Weise lässt sich der Sachverhalt mit einer einzigen Aktivität darstellen. Beschränkt man sich auf Aussagenlogik, kann der gleiche Sachverhalt dargestellt werden, wenn man eine endliche Menge von Dokumenten voraussetzt. In diesem Fall muss für jedes einzelne Dokument eine dedizierte Übersetzungs-Aktivität spezifiziert werden.

Ebenfalls hilfreich für eine einfache Modellierung einer Geschäftsdomäne ist die Unterstützung von *bedingten Effekten*. Ein bedingter Effekt ist ein Effekt, der abhängig von dem Zustand ist, in dem die Aktivität ausgeführt wird. Betrachten wir als einfaches Beispiel eine Maschine, die durch Druck auf eine Taste an- und ausgeschaltet werden kann. Die Ausführung einer Aktivität *drückeTaste* hat nun den Effekt, dass die Maschine an ist, falls sie vorher aus war und umgekehrt. Ein bedingter Effekt erlaubt nun die Modellierung dieses Zusammenhangs als eine Aktivität. Bei einem Verzicht auf bedingte Effekte ist dieser Zusammenhang ebenfalls modellierbar. Allerdings müssten dann das Ein- und das Ausschalten als separate Aktivitäten betrachtet werden.

Aktuelle Planer wie SAPA [30, 29], LPG [43] oder Metric-FF [52] und der in Abschnitt 2.2.2 vorgestellte Sprachstandard PDDL in Version 2.1 unterstützen die meisten der in einer Geschäftsdomäne benötigten Eigenschaften. Allerdings haben KI-Planer, obwohl sie sich bereits seit mehr als 35 Jahren in der Entwicklung be-

finden, erst in den letzten Jahren ein Niveau erreicht, das ihren Einsatz zur Planung von Geschäftsprozessen sinnvoll erscheinen lässt. Ein weiterer wichtiger Aspekt in diesem Zusammen wird jedoch momentan noch nicht ausreichend von KI-Planern unterstützt: die Nachvollziehbarkeit des Planungsergebnisses. Während bei klassischen Einsatzgebieten der KI-Planung wie zum Beispiel eine Robotersteuerung eine vollständige Automatisierung angestrebt wird, ist es bei der Planung von Geschäftsprozessen gut vorstellbar, dass das Ergebnis der Planung in Form der Prozessdefinition vor der Ausführung noch manuell überprüft wird. Hierbei ist es von zentraler Bedeutung, dass der Benutzer die Zusammenhänge zwischen der Spezifikation der Domäne, dem Planungsproblem und der daraus resultierenden Prozessdefinition erkennen kann beziehungsweise gegebenenfalls vom Planer erklärt bekommt. Da diese Anforderung bei den bisherigen Einsatzgebieten der KI-Planung keine wesentliche Rolle spielte, bieten aktuelle Planer in dieser Hinsicht noch eine sehr eingeschränkte Funktionalität.

3.4 Zusammenfassung

In diesem Kapitel wurde das grundlegende Konzept zur Integration von Prozessplanung und -ausführung erläutert. Hierbei wurden zunächst sich einander entsprechende Entitäten der Disziplinen Workflow-Management und KI-Planung identifiziert und anschließend grundlegende Annahmen festgelegt. Darauf aufbauend wurde ein System konzipiert, in dem Techniken aus beiden Disziplinen derart miteinander verknüpft werden, dass Geschäftsfälle automatisch durch Planung und Ausführung eines passenden Prozesses bearbeitet werden. Als wesentlicher Aspekt wurde dabei herausgestellt, dass für jeden Geschäftsfall eine individuelle Planung stattfindet und damit alle Besonderheiten eines Geschäftsfalls automatisch berücksichtigt werden. Des Weiteren wurde erläutert, dass eine individuelle Prozessdefinition für einen konkreten Geschäftsfall in der Regel weitaus einfacher strukturiert ist als eine Prozessdefinition, mit der eine Menge ähnlicher Geschäftsfälle bearbeitet werden soll. Schließlich wurde untersucht, welche besonderen Anforderungen bezüglich der KI-Planung in einer Geschäftsdomäne gelten. In diesem Zusammenhang wurde insbesondere die benötigte Ausdrucksmächtigkeit einer Sprache zur Spezifikation von Planungsproblemen diskutiert.

Kapitel 4

Automatisierte Prozessanpassung

In Kapitel 3 wurde beschrieben, wie bei einer integrierten Prozessplanung und -ausführung der Zustand eines Geschäftsfalls in einen Zielzustand überführt wird. Dabei wurde bisher von unerwarteten Ereignissen zur Laufzeit der entsprechenden Prozessinstanz abstrahiert, die unter Umständen die Erreichung eines Zielzustands verhindern. In diesem Kapitel wird ein Konzept vorgestellt, wie unerwartete Ereignisse automatisch erkannt und falls notwendig durch eine adäquate Anpassung der Prozessinstanz berücksichtigt werden können. Dies wird im Folgenden als *automatisierte Prozessanpassung* bezeichnet. Diese Möglichkeit einer automatisierten Prozessanpassung zur Laufzeit ist, wie in der Einleitung dargestellt, eine zentrale Motivation für die Integration von Planung und Ausführung.

Im folgenden Abschnitt 4.1 werden Ereignisse identifiziert, die eine Prozessanpassung zur Laufzeit notwendig machen. Abschnitt 4.2 beschreibt den grundlegenden Lösungsansatz für eine automatisierte Prozessanpassung zur Laufzeit und untergliedert diese Aufgabe. Anschließend wird in Abschnitt 4.3 die besondere Problematik einer Prozessanpassung zur Laufzeit aufgezeigt. Für diese Problematik werden in Abschnitt 4.4 drei Lösungsalternativen vorgestellt und miteinander verglichen. Abschließend fasst Abschnitt 4.5 die wesentlichen Konzepte der automatisierten Prozessanpassung zusammen.

4.1 Motivation

Bisher haben wir die grundlegende Vorgehensweise zur Bearbeitung eines Geschäftsfalls bei der integrierten Prozessplanung und -ausführung betrachtet: Zunächst erfolgt die Planung einer individuellen Prozessdefinition für den Geschäftsfall. Anschließend wird diese Prozessdefinition instanziiert und die Instanz ausgeführt, wodurch der aktuelle Zustand des Geschäftsfalls in einen Zielzustand überführt wird. Dabei wurden bisher Ereignisse zur Laufzeit eines Prozesses nicht betrachtet, die die Erreichung eines Zielzustands gefährden. Kritisch sind hierbei folgende Arten von Ereignissen:

- Die Ausführung einer Aktivität hat nicht die erwarteten Effekte.

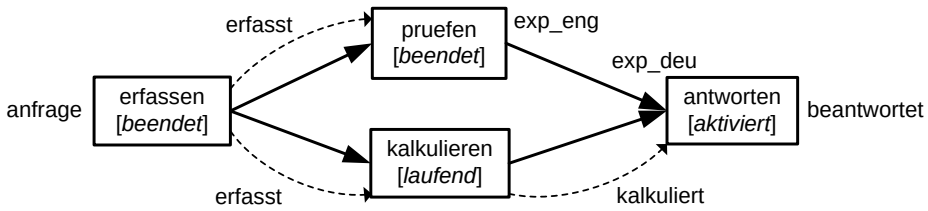


Abbildung 4.1: Unerwarteter Effekt

- Das Ziel ändert sich.
- Die Domäne ändert sich.
- Der Zustand des Geschäftsfalls ändert sich durch einen Eingriff von Außen.
- Ein Zeitlimit für die Ausführungsdauer einer Aktivität wird überschritten.

Hierzu sei angemerkt, dass eine Änderung der Domäne zum Beispiel die Verfügbarkeit einer neuen Aktivität sein kann oder die Änderung der Vorbedingungen oder Effekte einer bestehenden Aktivität. Die Problematik bei diesen Ereignissen ist, dass sie zum Planungszeitpunkt nicht vorhersehbar waren und somit bei der Planung nicht berücksichtigt wurden. Daher kann in diesen Fällen nicht mehr garantiert werden, dass eine Fortsetzung der Prozessausführung den Zustand des Geschäftsfalls in einen Zielzustand überführt.

Greifen wir zur Illustration das Beispiel aus Abschnitt 3.1.3 auf und betrachten die in Abbildung 3.6 dargestellte, laufende Prozessinstanz zur Bearbeitung einer Angebotsanfrage. Die beiden nebenläufigen Aktivitäten *pruefen* und *kalkulieren* befinden sich in Ausführungszustand *laufend*. Laut Spezifikation hat die Aktivität *pruefen* den Effekt *exp_deu*. Dieser Effekt bildet die kausale Verknüpfung der Aktivitäten *pruefen* und *antworten*. Betrachten wir nun den Fall, dass die Ausführung einer Aktivität nicht die erwarteten Effekte hat. Nehmen wir an, dass die Aktivität *pruefen* endet und anstelle des Effekts *exp_deu* den Effekt *exp_eng* hat. Diese Situation ist in Abbildung 4.1 dargestellt. Die kausale Verknüpfung zwischen der Aktivität *pruefen* und der Aktivität *antworten* ist nicht mehr gegeben.

Wird die Ausführung der Prozessinstanz ohne weitere Maßnahmen fortgeführt, so schlägt der Start der Aktivität *antworten* fehl, da die Vorbedingung *exp_deu* nicht erfüllt ist. Eine einfache Fortsetzung der Prozessausführung ist somit nicht sinnvoll. Stattdessen muss die Prozessinstanz angepasst werden, um trotz des erwarteten Effekts der Aktivität *pruefen* das Ziel zu erreichen. In diesem Beispiel könnte das Problem behoben werden, indem vor dem Start der Aktivität *antworten* die Aktivität *uebersetzen* ausgeführt wird, da ihre Vorbedingung im aktuellen Zustand erfüllt ist und ihr Effekt wiederum die Vorbedingung der Aktivität *antworten* erfüllt. Die Aufgabenstellung für dieses Kapitel ist nun eine allgemeine Vorgehensweise zu finden, wie eine Prozessanpassung zur Laufzeit bei der integrierten Prozessplanung und -ausführung durchgeführt werden kann.

4.2 Lösungsansatz

In Abschnitt 4.1 wurden Ereignisse zur Laufzeit einer Prozessinstanz beschrieben, die die Erreichung eines Zielzustands gefährden. Da sich diese Ereignisse zum Planungszeitpunkt nicht antizipieren lassen, können sie nicht in der Prozessdefinition berücksichtigt werden. So lässt sich zum Planungszeitpunkt nicht vorhersehen, dass zur Laufzeit der Prozessinstanz beispielsweise das Ziel geändert wird. Ein Grund hierfür könnte sein, dass sich der Bedarf eines Kunden zwischenzeitlich geändert hat und er im Beispiel der Angebotsanfrage plötzlich an einer größeren Menge eines Produktes, als ursprünglich angefragt wurde, interessiert ist. Der Lösungsansatz zur Handhabung dieses Problems, ist die Planung einer neuen Prozessdefinition basierend auf den aktuellen Informationen über Zustand, Ziel und Domäne. Die Idee dahinter ist, dass sich alle beim ursprünglichen Planen nicht vorhersehbaren Ereignisse in diesen aktuellen Werten widerspiegeln. In dem Beispiel aus Abschnitt 4.1 spiegelt sich der unerwartete Effekt *exp_eng* der Aktivität *pruefen* im aktuellen Zustand des Geschäftsfalls wieder. Im aktuellen Zustand gelten die Aussagen: *anfrage*, *erfasst* und *exp_eng*. Eine erneute Planung basierend auf diesem Zustand ergibt eine Prozessdefinition, die einen neuen Weg beschreibt das Ziel zu erreichen. Eine derartige erneute Planung einer Prozessdefinition bezeichnen wir im Folgenden als *Umplanung*. Damit eine automatisierte Prozessanpassung zur Laufzeit möglich wird, müssen neben dem zentralen Schritt der Umplanung auch die vorhergehende Überwachung des Prozesses und die nachfolgende Fortsetzung der Prozessausführung automatisiert werden. Die automatisierte Prozessanpassung umfasst somit folgende Schritte:

1. Überwachung: Ereignisse, die die Erreichung des Ziels gefährden, werden erkannt und eine Umplanung wird ausgelöst.

2. Umplanung: Eine neue Prozessdefinition basierend auf den aktuellen Werten von Zustand, Ziel und Domäne wird geplant und instanziiert.

3. Fortsetzung: Die Ausführung wird entsprechend der neuen Prozessinstanz fortgesetzt.

Betrachten wir bezüglich der automatisierten Prozessanpassung noch einmal die Komponenten des in Abbildung 3.1 dargestellten Systems zur integrierten Prozessplanung und -ausführung: Der Planer hat bereits eine initiale Prozessdefinition generiert, die nun von der Prozesssteuerung ausgeführt wird. Die Prozessüberwachung hat die Aufgabe, auf die in Abschnitt 4.1 beschriebenen Ereignisse zu reagieren und gegebenenfalls eine Umplanung auszulösen. Dabei ist wichtig, dass der aktuelle Zustand des Geschäftsfalls protokolliert wird, da dieser für die Umplanung benötigt wird. Entsprechend der Annahme A2 (vergleiche Abschnitt 3.1.2) kann am Ende der Ausführung einer Aktivität der Folgezustand festgestellt werden. Wird eine Umplanung ausgelöst, so übergibt die Prozessüberwachung die Informationen über den aktuellen Zustand des Geschäftsfalls an den Planer. Zusammen mit

der Domänenbeschreibung und dem Ziel des Geschäftsfalls liegen dem Planer damit alle Informationen vor, die er für die Planung benötigt. Der Planer versucht daraufhin eine neue Prozessdefinition zu finden, die eine Lösung für das Planungsproblem darstellt. Falls keine neue Prozessdefinition gefunden werden kann, ist ein manueller Eingriff notwendig oder die Bearbeitung des Geschäftsfalls muss erfolglos beendet werden. Ein manueller Eingriff könnte beispielsweise eine Modifikation des Ziels sein, so dass zumindest ein Teilziel des ursprünglichen Ziels erreicht wird oder eine Erweiterung der Domäne um eine zusätzliche Aktivität, die es dem Planer ermöglicht, doch noch eine neue Prozessdefinition zu finden. Wird eine neue Prozessdefinition gefunden, so übergibt der Planer diese an die Prozesssteuerung. Die Prozesssteuerung instanziiert die neue Prozessdefinition und setzt die Ausführung entsprechend der neuen Prozessinstanz fort.

Während der Bearbeitung eines Geschäftsfalls kann eine derartige automatisierte Prozessanpassung auch mehrfach notwendig werden. In diesem Fall durchläuft die Bearbeitung abwechselnd Planungs- und Ausführungsphasen, bis das Ziel erreicht ist oder die Bearbeitung erfolglos abgebrochen werden muss. Solange der Planer bei der Umplanung eine neue Prozessdefinition finden kann, ist für die gesamte Prozessanpassung kein manueller Eingriff notwendig. Weder muss der Prozess manuell überwacht werden, noch muss die Prozessinstanz manuell angepasst werden. Ein manueller Eingriff durch einen Domänenmodellierer ist dabei nicht notwendig, da nur die Prozessdefinition und nicht die Domänenbeschreibung verändert werden muss. Somit ist eine Prozessanpassung unabhängig von der Verfügbarkeit eines Domänenmodellierers durchführbar. Dies ist ein wesentlicher Vorteil gegenüber klassischen Workflow-Management-Systemen, bei denen im Falle einer Prozessanpassung ein Prozessmodellierer eine neue Prozessdefinition spezifizieren muss.

4.3 Problematik der Umplanung zur Laufzeit

Bei der initialen Planung einer Prozessdefinition wie in Abschnitt 3.1 beschrieben, wird davon ausgegangen, dass alle Veränderungen des Zustands des Geschäftsfalls ausschließlich auf die Effekte von Aktivitäten dieser Prozessdefinition zurückzuführen sind. Bei der Umplanung ist die Situation eine andere: Aktivitäten können bereits entsprechend der ursprünglichen Prozessdefinition von der Prozesssteuerung gestartet worden sein und sich zum Zeitpunkt der Umplanung in Ausführung befinden. Werden diese Aktivitäten bei der Anpassung des Prozesses nicht berücksichtigt, ergibt sich folgendes Problem: Der Planer generiert eine neue Prozessdefinition basierend auf den aktuellen Werten von Zustand, Ziel und Domäne. Im aktuellen Zustand spiegeln sich jedoch noch nicht die Effekte der noch laufenden Aktivitäten wider. Somit können diese laufenden Aktivitäten bei der Umplanung von Planer nicht berücksichtigt werden. Als Folge generiert der Planer unter Umständen eine nicht optimale oder fehlerhafte Prozessdefinition.

Betrachten wir hierzu das Beispiel einer Angebotsanfrage. Gegeben sei die in

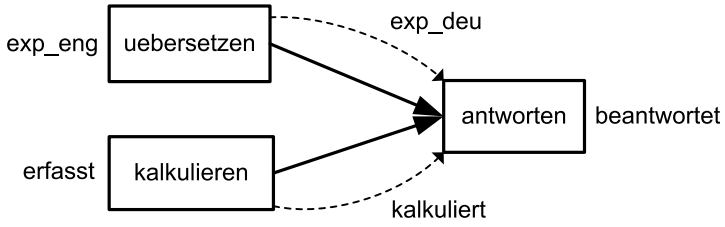


Abbildung 4.2: Prozessdefinition ohne Berücksichtigung laufender Aktivitäten

Abbildung 4.1 dargestellte Situation. Im aktuellen Zustand gelten die Aussagen: *anfrage*, *erfasst* und *exp_eng*. Eine erneute Planung basierend auf diesem Zustand ergibt die in Abbildung 4.2 dargestellte Prozessdefinition. Diese Prozessdefinition enthält die Aktivität *kalkulieren*, obwohl sich diese Aktivität bereits in Ausführung befindet. Werden nun durch die Prozesssteuerung die Aktivitäten entsprechend der neuen Prozessdefinition gestartet, so wird die Aktivität *kalkulieren* unnötigerweise ein zweites Mal ausgeführt. In diesem Beispiel zeigt sich bereits, dass diese Vorgehensweise nicht angemessen ist. Es lassen sich auch Beispiele konstruieren, bei denen Aktivitäten, die bei der Umplanung noch laufen, mit Aktivitäten der neuen Prozessdefinition derart in Konflikt stehen, dass das Ziel nicht mehr erreicht wird.

4.4 Lösungsalternativen

Für das in Abschnitt 4.3 beschriebene Problem der bei der Umplanung laufenden Aktivitäten werden im Folgenden drei alternative Verfahren zur Lösung vorgestellt und verglichen.

4.4.1 Umplanung nach Ausführungsunterbrechung

Eine einfache Lösung für das Problem der bei der Umplanung noch laufenden Aktivitäten besteht darin, alle aktivierten Aktivitäten zu deaktivieren und zu warten, bis alle laufenden Aktivitäten beendet sind. Dann kann der Zustand durch keinen Effekt einer laufenden Aktivität mehr verändert werden. Somit ist die Ausgangssituation für eine Umplanung die gleiche wie für eine initiale Planung. Die Umplanung gliedert sich dann in folgende Schritte:

- **Schritt 1**:Deaktiviere alle aktivierten Aktivitäten.
- **Schritt 2**:Warte, bis alle laufenden Aktivitäten beendet sind.
- **Schritt 3**: Plane eine neue Prozessdefinition basierend auf den aktuellen Werten von Zustand, Ziel und Domäne.
- **Schritt 4**:Instanziiere die neue Prozessdefinition.

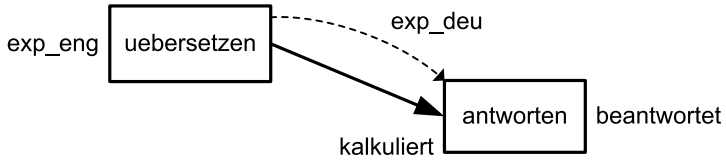


Abbildung 4.3: Neue Prozessdefinition nach Ausführungsunterbrechung

Übertragen auf das Beispiel der Angebotsanfrage aus Abbildung 4.1 ergibt sich folgendes Vorgehen: Die Aktivität *antworten* wird deaktiviert (**Schritt 1**) und anschließend wird gewartet, bis die Aktivität *kalkulieren* beendet ist (**Schritt 2**). Erst dann folgt **Schritt 3** und die Umplanung wird gestartet. Im aktuellen Zustand des Geschäftsfalls gelten dann die Aussagen: *anfrage*, *erfasst*, *exp_eng* und *kalkuliert*. Eine erneute Planung basierend auf diesem Zustand ergibt die in Abbildung 4.3 dargestellte Prozessdefinition. Diese Prozessdefinition enthält keine Aktivität *kalkulieren* mehr, da sich ihr Effekt *kalkuliert* bereits im aktuellen Zustand widerspiegelt und damit auch bei der Planung berücksichtigt wurde. Schließlich wird in **Schritt 4** die Prozessdefinition an die Prozesssteuerung übergeben und instanziiert. Die weitere Ausführung überführt den Zustand des Geschäftsfalls in einen Zielzustand. Würde keine neue Prozessdefinition gefunden werden, wäre wie in Abschnitt 4.2 beschrieben ein manueller Eingriff notwendig.

Kritische Diskussion des Verfahrens Die beschriebene Vorgehensweise der Unterbrechung der Prozessausführung hat den Nachteil, dass die Bearbeitung des Geschäftsfalls unter Umständen unnötig verzögert wird: Nach der Deaktivierung aller aktivierten Aktivitäten wird gewartet, bis alle laufenden Aktivitäten beendet sind. Hierbei wird gegebenenfalls die Möglichkeit verpasst, Aktivitäten bereits zu starten, die nicht auf die Beendigung der noch laufenden Aktivitäten angewiesen sind. Betrachten wir hierzu wieder das obige Beispiel. Es wird gewartet, bis die Aktivität *kalkulieren* beendet ist. Die neue Prozessdefinition enthält jedoch die Aktivität *uebersetzen*, die bereits direkt nach Beendigung der Aktivität *pruefen* gestartet und somit nebenläufig zur Aktivität *kalkulieren* ausgeführt werden könnte.

4.4.2 Antizipation eines Zustandsraums

In diesem Abschnitt wird eine zweite Lösungsalternative zur Berücksichtigung der bei der Umplanung laufenden Aktivitäten vorgestellt. Ausgangspunkt hierbei ist die Kritik an dem Verfahren *Umplanung nach Ausführungsunterbrechung*. Hierbei wurde nach Deaktivierung der aktivierten Aktivitäten auf die Beendigung der noch laufenden Aktivitäten gewartet. Wie im Beispiel dargestellt, werden dadurch unter Umständen Möglichkeiten zur einer nebenläufigen Ausführung vergeben. Die Idee bei Lösung *Antizipation eines Zustandsraums* ist es, nicht mit der Planung zu warten, bis alle laufenden Aktivitäten beendet sind. Stattdessen wird unmittelbar nach

Eintritt eines unerwarteten Ereignisses eine neue Prozessdefinition geplant und die Ausführung fortgesetzt. Dabei wird der Zustand nach Beendigung der noch laufenden Aktivitäten antizipiert. Das heißt, dass der für die Planung benötigte Ausgangszustand auf Basis des aktuellen Zustands und der noch laufenden Aktivitäten ermittelt wird. Die Umplanung gliedert sich somit in folgende Schritte:

- **Schritt 1**: Entferne den nicht ausgeführten Teil der Prozessinstanz.
- **Schritt 2**: Antizipiere den Zustandsraum nach Beendigung aller noch laufenden Aktivitäten.
- **Schritt 3**: Plane eine neue Prozessdefinition basierend auf dem antizipierten Zustandsraum und den aktuellen Werten von Ziel und Domäne.
- **Schritt 4**: Instanziiere die neue Prozessdefinition.
- **Schritt 5**: Verbinde die ursprüngliche und die neue Prozessinstanz durch das Einfügen von Kontrollkonnektoren, die aufgrund der kausalen Verknüpfungen notwendig sind.

Dieses Vorgehen wird nun an dem Beispiel aus Abbildung 4.1 illustriert: In **Schritt 1** wird zunächst die Aktivität *antworten* aus der ursprünglichen Prozessdefinition entfernt wird. Anschließend wird in **Schritt 2** der Zustand des Geschäftsfalls nach Beendigung der Aktivität *kalkulieren* antizipiert. Sobald die Aktivität *kalkulieren* beendet ist, werden folgende Aussagen gelten: *anfrage*, *erfasst*, *exp_eng* und *kalkuliert*. Im Gegensatz zu diesem Beispiel ist im Allgemeinen der antizipierte Zustand nach Beendigung aller laufenden Aktivitäten nicht immer eindeutig: Nehmen wir zum Beispiel an, es laufen bei der Umplanung noch zwei Aktivitäten *A* und *B*. *A* habe den Effekt *Lampe leuchtet* und *B* den Effekt *Lampe leuchtet nicht*. *B* hat demnach den negierten Effekt von *A*. In diesem Fall ist der voraussichtliche Zustand nach Beendigung der beiden laufenden Aktivitäten nicht eindeutig. Welcher Zustand folgt, hängt davon ab, welche der Aktivitäten *A* oder *B* zuerst endet. Allerdings lässt sich die Menge möglicher Folgezustände bestimmen, die als *Zustandsraum* bezeichnet wird. Hierzu muss vom aktuellen Zustand ausgehend jede mögliche Reihenfolge der Beendigung der noch laufenden Aktivitäten verfolgt werden. Die Menge der dabei erreichbaren Folgezustände bildet dann den Zustandsraum.

In **Schritt 3** wird basierend auf dem Zustandsraum eine neue Prozessdefinition geplant. Die Aufgabe für den Planer besteht dabei darin, eine Prozessdefinition zu finden, die jeden der Zustände des Zustandsraums in einen Zielzustand überführt. In dem Beispiel der Angebotsanfrage ergibt sich wie bei der Lösung *Umplanung nach Ausführungsunterbrechung* die in Abbildung 4.3 dargestellte Prozessdefinition.

Schritt 4 ist die Instanziierung der neuen Prozessdefinition. Schließlich wird in **Schritt 5** die ursprüngliche und die neue Prozessinstanz durch das Einfügen von Kontrollkonnektoren verbunden. Bevor das Vorgehen hierzu erläutert wird,

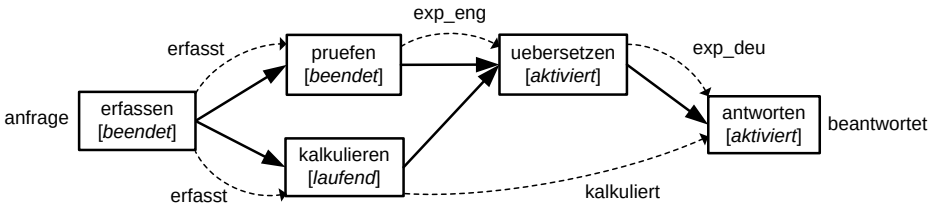


Abbildung 4.4: Naiv verbundene Prozessinstanz

erfolgt zunächst die Beschreibung eines naiven Ansatzes zur Verknüpfung der Prozessinstanzen, um das gewählte Vorgehen zu motivieren: Ein naiver Ansatz wäre, alle Aktivitäten der ursprünglichen Prozessinstanz mit allen Aktivitäten der neuen Prozessinstanz durch Kontrollkonnektoren zu verbinden. Auf diese Weise ist sichergestellt, dass alle kausalen Abhängigkeiten berücksichtigt sind: Durch die eingefügten Kontrollkonnektoren kann keine der Aktivitäten der neuen Prozessinstanz gestartet werden, bevor alle noch laufenden Aktivitäten beendet sind. Da nach Beendigung aller noch laufenden Aktivitäten ein Zustand aus dem Zustandsraum erreicht ist, sind alle kausalen Verknüpfungen berücksichtigt. In dem Beispiel ergäbe sich die in Abbildung 4.4 dargestellte Prozessinstanz. Hierbei sind Kontrollkonnektoren nicht abgebildet, die sich aus anderen Kontrollkonnektoren ableiten lassen. Beispielsweise ist der Kontrollkonnektor von der Aktivität *kalkulieren* zur Aktivität *antworten* nicht dargestellt, da er sich aus den Kontrollkonnektoren von *kalkulieren* nach *uebersetzen* und von *uebersetzen* nach *antworten* ableiten lässt. Das Problem bei dieser Vorgehensweise ist, dass die Nebenläufigkeit nicht höher ist als bei der Lösung *Umplanung nach Ausführungsunterbrechung*. Es wird zwar mit der Planung einer neuen Prozessdefinition nicht gewartet, bis die Aktivität *kalkulieren* beendet ist, sondern geplant, sobald die Aktivität *pruefen* beendet ist. Dennoch werden die Aktivitäten aus der neuen Prozessdefinition nicht gestartet bevor die Aktivität *kalkulieren* beendet ist. Insbesondere erfolgt keine nebenläufige Ausführung der Aktivitäten *kalkulieren* und *uebersetzen*.

Damit durch die Antizipation des Zustandsraums nach Beendigung aller noch laufenden Aktivitäten ein höheres Maß an Nebenläufigkeit möglich wird, erfolgt die Verknüpfung in Schritt 5 wie folgt: Anstatt alle Aktivitäten der ursprünglichen Prozessinstanz mit allen Aktivitäten der neuen Prozessinstanz durch Kontrollkonnektoren zu verbinden, werden Kontrollkonnektoren nur dann eingefügt, wenn kausale Verknüpfungen es erfordern. Das heißt, ein Kontrollkonnektor wird eingefügt, falls er direkt durch eine kausale Verknüpfung impliziert ist oder falls er zum Schutz einer kausalen Verknüpfung notwendig ist. In dem Beispiel werden bei der Verknüpfung somit zwei Kontrollkonnektoren eingefügt. Die entsprechende Prozessinstanz I_1 ist in Abbildung 4.5 dargestellt. Ein Kontrollkonnektor ist direkt durch die kausale Verknüpfung *exp_eng* von der Aktivität *pruefen* zur Aktivität *uebersetzen* impliziert und ein weiterer direkt durch die kausale Verknüp-

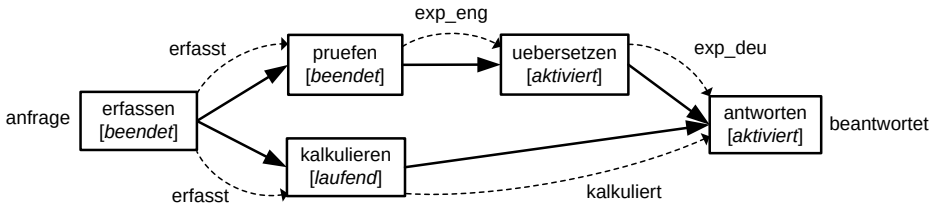


Abbildung 4.5: Verbundene Prozessinstanz I_1

fung *kalkuliert* von *kalkulieren* nach *antworten*. Durch diese Form der Verbindung der ursprünglichen und der neuen Prozessinstanz kann gegebenenfalls ein höheres Maß an Nebenläufigkeit als bei dem Verfahren *Umplanung nach Ausführungsunterbrechung* möglich werden. In dem Beispiel kann die Aktivität *uebersetzen* direkt gestartet werden, ohne dass auf die Beendigung der Aktivität *kalkulieren* gewartet werden muss.

Kritische Diskussion des Verfahrens Betrachten wir als nächstes eine Modifikation des obigen Beispiels: Die in Abbildung 3.4 dargestellte Domänenbeschreibung sei um eine Aktivität erweitert. Neben der Aktivität *antworten* enthält die Domänenbeschreibung eine Aktivität *antworten2*, die sich von der Aktivität *antworten* in folgenden Punkten unterscheidet: Die Vorbedingungen der Aktivität *antworten2* sind *exp_eng* und *kalkuliert* und ihre Ausführungsdauer beträgt 4 Stunden. Im Gegensatz zur Aktivität *antworten* benötigt die Aktivität *antworten2* demnach die Exportbedingungen auf Englisch und besitzt eine um eine Stunde längere Ausführungsdauer. Als nächstes untersuchen wir die Auswirkungen dieser Modifikation der Domäne auf das Ergebnis der Anwendung der Lösung *Antizipation eines Zustandsraums* auf das Beispiel aus Abbildung 4.1. Bis zum Schritt 3 der Planung einer neuen Prozessdefinition hat die Modifikation der Domäne keine Auswirkungen. Der antizipierte Zustandsraum enthält genau einen Zustand. In diesem Zustand gelten die Aussagen: *anfrage*, *erfasst*, *exp_eng* und *kalkuliert*. Eine Planung basierend auf diesem Zustand hat aufgrund der Modifikation der Domäne nun ein weiteres mögliches Ergebnis. Neben der in Abbildung 4.3 dargestellten Prozessdefinition P_1 gibt es jetzt eine weitere Prozessdefinition P_2, die den obigen Zustand in einen Zielzustand überführt. P_2 enthält als einzige Aktivität die Aktivität *antworten2*. Die Vorbedingungen dieser Aktivität sind im obigen Zustand erfüllt und ihr Effekt *beantwortet* überführt diesen Zustand in einen Zielzustand. Ein Planer, der die Ausführungsdauer des Prozesses optimiert, wählt P_2 aus, da diese eine kürzere Ausführungsdauer besitzt: Die Prozessdefinition P_1 besitzt eine Ausführungsdauer von 2 Stunden (Aktivität *uebersetzen*) + 3 Stunden (Aktivität *antworten*) = 5 Stunden. Demgegenüber besitzt P_2 eine Ausführungsdauer von nur 4 Stunden (Aktivität *antworten2*). In Schritt 4 wird P_2 instanziiert. Anschließend wird in Schritt 5 die resultierende neue Prozessinstanz I_2 unter Berücksichtigung der kausalen Ver-

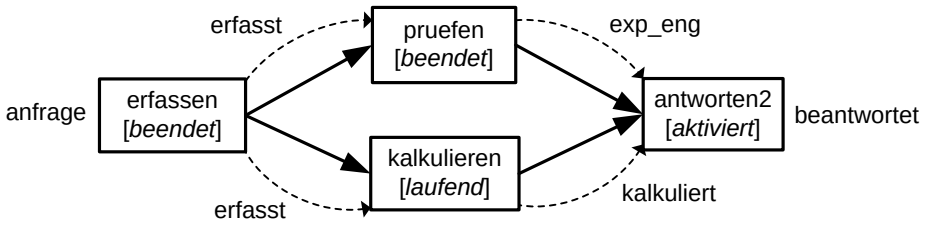


Abbildung 4.6: Verbundene Prozessinstanz I_2

knüpfungen mit der ursprünglichen Prozessinstanz durch das Einfügen von Kontrollkonnektoren verbunden. Abbildung 4.6 zeigt die resultierende Prozessinstanz. Die restliche Ausführungsdauer dieser Prozessinstanz unter der Annahme, dass alle Aktivitäten gestartet werden, sobald der Kontrollfluss dies zulässt, ergibt sich wie folgt: Zunächst ermitteln wir die restliche Ausführungsdauer der Aktivität *kalkulieren*. Die Aktivität *kalkulieren* wurde zeitgleich mit der Aktivität *pruefen* gestartet. Die Umplanung findet statt, sobald die Aktivität *pruefen* endet. Da die Ausführungsdauer der Aktivität *pruefen* 4 Stunden und die Ausführungsdauer der Aktivität *kalkulieren* 8 Stunden beträgt, besitzt die Aktivität *kalkulieren* zum Beginn der Umplanung eine restliche Ausführungsdauer von 4 Stunden. Wenn wir davon ausgehen, dass die Umplanung eine Stunde benötigt, besitzt die Aktivität *kalkulieren* nach der Umplanung noch eine restliche Ausführungsdauer von 3 Stunden. Nach der Beendigung der Aktivität *kalkulieren* wird die Aktivität *antworten2* gestartet. Diese besitzt eine Ausführungsdauer von 4 Stunden. Somit ergibt sich eine restliche Ausführungsdauer von insgesamt 7 Stunden für die Prozessinstanz.

Nehmen wir nun an der Planer hätte nicht die Prozessdefinition P_2 ausgewählt, sondern die Prozessdefinition P_1. In diesem Fall ergibt sich die in Abbildung 4.5 dargestellte Prozessinstanz I_1. Die restliche Ausführungsdauer dieser Prozessinstanz berechnet sich wie folgt: Wie oben beschrieben, besitzt die Aktivität *kalkulieren* nach der Umplanung noch eine restliche Ausführungsdauer von 3 Stunden. Nebenläufig zur Aktivität *kalkulieren*, wird die Aktivität *uebersetzen* mit einer Ausführungsdauer von 2 Stunden ausgeführt. Somit ist bei Beendigung der Aktivität *kalkulieren* auch die Aktivität *uebersetzen* bereits beendet und die Aktivität *antworten* mit einer Ausführungsdauer von 3 Stunden wird gestartet. Folglich ergibt sich eine restliche Ausführungsdauer von insgesamt 6 Stunden für die Prozessinstanz.

Bei dem Vergleich der restlichen Ausführungsdauer der Prozessinstanzen I_1 (6 Stunden) und I_2 (7 Stunden), fällt auf, dass der Planer bei der Auswahl der neuen Prozessdefinition P_1 oder P_2 bezüglich der restlichen Ausführungsdauer die falsche gewählt hat, obwohl er die Ausführungsdauer bei der Planung optimiert. Dies liegt daran, dass der Planer die Ausführungsdauer der neuen Prozessdefinition optimiert und nicht die restliche Ausführungsdauer der daraus resultierenden Prozessinstanz. In dem Beispiel hat die Prozessdefinition P_2 mit 4 Stunden eine

kürzere Ausführungsdauer als die Prozessdefinition P_1 mit 5 Stunden. Dennoch ergibt sich auf Basis der Prozessdefinition P_1 die Prozessinstanz I_1 mit einer kürzeren restlichen Ausführungsdauer im Vergleich zur Prozessinstanz I_2. Die Ursache für dieses Problem liegt darin, dass der Planer keine Informationen über die noch laufenden Aktivitäten erhält, sondern lediglich den antizipierten Zustandsraum. Insbesondere fehlt dem Planer die Information, wann genau die Effekte der einzelnen noch laufenden Aktivitäten zu erwarten sind. Folglich kann er auch nicht erkennen, dass im obigen Beispiel die Prozessdefinition P_1 die bessere Wahl ist, da wie aus Abbildung 4.5 ersichtlich, die Aktivität *uebersetzen* nebenläufig zur Aktivität *kalkulieren* ausgeführt werden kann. Diese Überlegungen führen unmittelbar zur dritten Lösungsalternative, die im folgenden Abschnitt vorgestellt wird.

4.4.3 Domänenerweiterung um Stellvertreteraktivitäten

In diesem Abschnitt wird eine dritte Lösung zur Berücksichtigung der bei der Umplanung laufenden Aktivitäten vorgestellt, die im folgenden Kapitel auch formal beschrieben wird. Diese Lösung setzt genau an der in Abschnitt 4.4.2 beschriebenen Kritik an dem Verfahren *Antizipation eines Zustandsraums* an. Bei der Planung einer neuen Prozessdefinition wurde bei diesem Verfahren vom Planer die Prozessdefinition P_2 gewählt, obwohl die Prozessdefinition P_1 bezüglich der restlichen Ausführungsdauer günstiger gewesen wäre. Die Ursache für dieses Problem wurde im Abschnitt 4.4.2 darauf zurückgeführt, dass der Planer keine Informationen über die noch laufenden Aktivitäten erhält.

Die Idee bei dem Verfahren *Domänenerweiterung um Stellvertreteraktivitäten* ist, die laufenden Aktivitäten bei der Planung zu berücksichtigen, indem sie in Form von Stellvertreteraktivitäten für den speziellen Umplanungsvorgang der Domänenbeschreibung hinzugefügt werden. Eine Stellvertreteraktivität besitzt dabei die gleichen Vorbedingungen und Effekte, wie die laufende Aktivität, die sie während der Umplanung repräsentiert. Sie wird nur für die Umplanung benötig und danach wieder durch die repräsentierte Aktivität ersetzt. Die Umplanung gliedert sich in folgende Schritte:

- **Schritt 1**: Deaktiviere alle aktivierten Aktivitäten.
- **Schritt 2**: Erzeuge Stellvertreteraktivitäten für alle laufenden Aktivitäten und füge sie der Domänenbeschreibung hinzu.
- **Schritt 3**: Plane eine neue Prozessdefinition basierend auf der erweiterten Domänenbeschreibung und den aktuellen Werten von Zustand und Ziel unter besonderer Berücksichtigung der Stellvertreteraktivitäten.
- **Schritt 4**: Instanziiere die neue Prozessdefinition unter besonderer Berücksichtigung der Stellvertreteraktivitäten.

Im Folgenden wird das Vorgehen an dem Beispiel der Angebotsanfrage aus Abbildung 4.1 illustriert. In **Schritt 1** wird zunächst die aktivierte Aktivität *antworten* deaktiviert. In **Schritt 2** wird für jede laufende Aktivität eine entsprechende

Stellvertreteraktivität erzeugt und der Aktivität zugeordnet. Jede Stellvertreteraktivität besitzt die gleichen Vorbedingungen und Effekte, wie die Aktivität der sie zugeordnet ist und ihr wird eine Ausführungsdauer zugeordnet, die der restlichen Ausführungsdauer der laufenden Aktivität entspricht. In dem Beispiel der Angebotsanfrage wird für die laufende Aktivität *kalkulieren* der Prozessinstanz aus Abbildung 4.1 eine Stellvertreteraktivität *kalkulieren** erzeugt, die die gleichen Vorbedingungen und Effekte hat wie die Aktivität *kalkulieren*. Als Ausführungsdauer wird der Stellvertreteraktivität *kalkulieren** die restliche Ausführungsdauer der Aktivität *kalkulieren* nach der Umplanung zugeordnet. Diese beträgt, wie in Abschnitt 4.4.2 beschrieben 3 Stunden. Abschließend wird die Stellvertreteraktivität *kalkulieren** der Domänenbeschreibung für diese spezielle Umplanung hinzugefügt.

Schritt 3 ist die Planung einer neuen Prozessdefinition. Für die Planung werden die erweiterte Domänenbeschreibung sowie die aktuellen Werte von Zustand und Ziel verwendet. In dem Beispiel gelten im aktuellen Zustand die Aussagen: *anfrage, erfasst* und *exp_eng*. Die Stellvertreteraktivitäten aus der erweiterten Domänenbeschreibung können bei der Planung wie jede andere Aktivität dazu benutzt werden, eine neue Prozessdefinition zu erstellen. Im Gegensatz zu den anderen Aktivitäten gelten für die Stellvertreteraktivitäten folgende, zusätzliche Bedingungen:

- Jede der Stellvertreteraktivitäten aus der Domäne muss genau einmal in der neuen Prozessdefinition vertreten sein.
- Keine der Stellvertreteraktivitäten darf in der neuen Prozessdefinition eingehende Kontrollkonnektoren besitzen.

Beide Bedingungen sind durch die Tatsache begründet, dass die Aktivität, für die die Stellvertreteraktivität erzeugt wurde, bereits läuft. Folglich muss die Stellvertreteraktivität auch Teil der zukünftigen Ausführung sein. Ebenso folgt daraus, dass keine Aktivität vor einer Stellvertreteraktivität ausgeführt werden kann. In dem Beispiel der Angebotsanfrage überführen sowohl die in Abbildung 4.7 dargestellte Prozessdefinition P_1^* als auch die in Abbildung 4.8 dargestellte Prozessdefinition P_2^* den aktuellen Zustand des Geschäftsfalls in einen Zielzustand und erfüllen darüber hinaus die beiden Bedingungen bezüglich der Stellvertreteraktivitäten. Ein Planer, der die Ausführungsdauer der Prozessdefinition optimiert, wählt die Prozessdefinition P_1^*, da sie eine Ausführungsdauer von 6 Stunden hat und die Prozessdefinition P_2^* eine Ausführungsdauer von 7 Stunden.

In **Schritt 4** folgt die Instanziierung der neuen Prozessdefinition. Hierbei müssen die Stellvertreteraktivitäten besonders berücksichtigt werden, da die entsprechenden laufenden Aktivitäten bereits existieren und nicht mehr instanziiert werden müssen. In dem Beispiel bedeutet dies, dass bei der Instanziierung der Prozessdefinition P_1^* nur die Aktivitäten *uebersetzen* und *beantworten* instanziiert werden. Die Stellvertreteraktivität *kalkulieren** hingegen wird nicht instanziiert. Stattdessen wird die entsprechende, laufende Aktivitätsinstanz *kalkulieren* in die neue Prozessinstanz I_1^* übernommen, die in Abbildung 4.9 dargestellt ist.

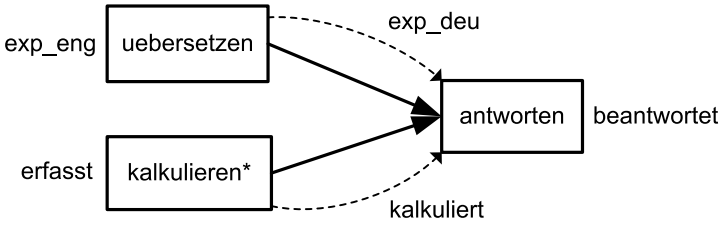


Abbildung 4.7: Prozessdefinition P_1^*

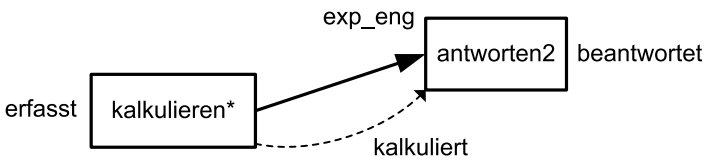


Abbildung 4.8: Prozessdefinition P_2^*

Die restliche Ausführungsdauer der Prozessinstanz I_1^* ergibt sich wie folgt: Die laufende Aktivität *kalkulieren* besitzt eine restliche Ausführungsdauer von 3 Stunden. Nebenläufig zur Aktivität *kalkulieren*, wird die Aktivität *uebersetzen* mit einer Ausführungsdauer von 2 Stunden ausgeführt. Somit ist bei Beendigung der Aktivität *kalkulieren* auch die Aktivität *uebersetzen* bereits beendet und die Aktivität *antworten* mit einer Ausführungsdauer von 3 Stunden wird gestartet. Folglich ergibt sich eine restliche Ausführungsdauer von insgesamt 6 Stunden für die Prozessinstanz. Hätte der Planer hingegen die Prozessdefinition P_2^* ausgewählt, dann würde die restliche Ausführungsdauer einer entsprechenden Prozessinstanz I_2^* 7 Stunden betragen. Somit hat der Planer durch die Optimierung der Ausführungsdauer der neuen Prozessdefinition in diesem Beispiel auch die richtige Prozessdefinition in Bezug auf die restliche Ausführungsdauer der sich hieraus ergebenden Prozessinstanz gewählt.

Kritische Diskussion des Verfahrens Es bleibt zu zeigen, ob ein Planer, der die Ausführungsdauer der neuen Prozessdefinition optimiert, auch in anderen Fällen immer die optimale Prozessdefinition in Bezug auf die resultierende Prozessinstanz wählt. Wie bereits in dem Beispiel auffällt, sind die Prozessdefinition P_1^* und die Prozessinstanz I_1^* von der Struktur her äquivalent. Entsprechend ist auch die restliche Ausführungsdauer der Prozessinstanz I_1^* mit der Ausführungsdauer der Prozessdefinition P_1^* identisch. Dies war in bei der Lösung *Antizipation eines Zustandsraums* nicht der Fall. Bei der Lösung *Domänenerweiterung um Stellvertreteraktivitäten* hingegen ist diese strukturelle Äquivalenz immer gegeben, da sich

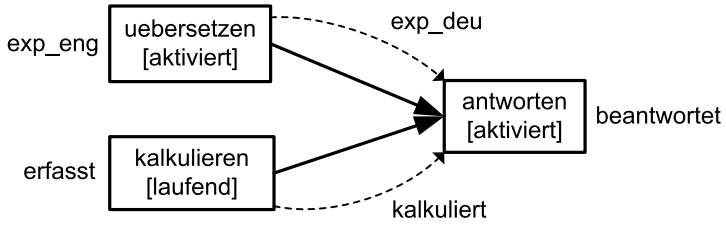


Abbildung 4.9: Prozessinstanz I_1^* basierend auf Prozessdefinition P_1^*

die neue Prozessinstanz im Prinzip durch eine Instanziierung der neuen Prozessdefinition ergibt. Die einzige Besonderheit hierbei sind die Stellvertreteraktivitäten. Diese werden jedoch bei der Instanziierung nur durch Aktivitäten ersetzt, die die gleiche Ausführungsdauer besitzen. Beispielsweise wurde die Stellvertreteraktivität *kalkulieren** durch die laufende Aktivitätsinstanz *kalkulieren* ersetzt, deren Restlaufzeit gleich der entsprechend spezifizierten Ausführungsdauer der Stellvertreteraktivität *kalkulieren** ist. Somit ergibt sich bei dem Verfahren *Domänenerweiterung um Stellvertreteraktivitäten* immer eine Prozessinstanz mit einer Ausführungsdauer, die der Ausführungsdauer der neuen Prozessdefinition entspricht. Folglich liefert ein Planer, der die Ausführungsdauer der neuen Prozessdefinition optimiert auch die optimale Lösung in Bezug auf die neue Prozessinstanz.

Bezüglich der Lösung *Domänenerweiterung um Stellvertreteraktivitäten* bleibt anzumerken, dass die spezifizierte Ausführungsdauer der Aktivitäten nur relevant für die Optimalität der neuen Prozessdefinition ist und nicht für die Erreichung eines Zielzustands. Das heißt, dass der Planer nur dann sinnvolle Entscheidungen bezüglich der Ausführungsdauer möglicher Prozessdefinitionen treffen kann, wenn die spezifizierte Ausführungsdauer der Aktivitäten mit ihrer tatsächlichen Ausführungsdauer übereinstimmt. Wesentlich ist, dass auch wenn diese Übereinstimmung nicht zutrifft, die Prozessdefinition ihre grundlegende Eigenschaft behält, den Zustand des Geschäftsfalls in einen Zielzustand zu überführen. Dies liegt darin begründet, dass bei der Planung entsprechend der Lösung *Domänenerweiterung um Stellvertreteraktivitäten* alle kausalen Abhängigkeiten zwischen den Aktivitäten berücksichtigt werden und sich in entsprechenden Kontrollkonnektoren in der resultierenden Prozessdefinition widerspiegeln. Das heißt, dass falls eine Aktivität länger dauert als spezifiziert, alle von ihr abhängigen Aktivitäten nicht gestartet werden bis diese Aktivität beendet ist, da entsprechende Kontrollkonnektoren in der Prozessdefinition den Kontrollfluss entsprechend einschränken. In Kapitel 5 wird diese Eigenschaft der Lösung *Domänenerweiterung um Stellvertreteraktivitäten* formal nachgewiesen.

4.5 Zusammenfassung

In diesem Kapitel wurde das grundlegende Konzept zur Integration von Prozessplanung und -ausführung aus Kapitel 3 um ein Konzept zur automatisierten Prozessanpassung zur Laufzeit erweitert. Als Lösungsansatz zur Behandlung von unerwarteten Ereignissen während der Prozessausführung wurde die Planung einer neuen Prozessdefinition basierend auf dem aktuellen Zustand des Geschäftsfalls vorgestellt. Anschließend wurde eine zentrale Problematik dieses Lösungsansatzes identifiziert: Der aktuelle Zustand des Geschäftsfalls kann sich während der Planung einer neuen Prozessdefinition durch die noch laufenden Aktivitäten ändern. Zur Berücksichtigung derartiger Zustandsänderungen wurden alternative Verfahren entwickelt und miteinander verglichen. Das erste vorgeschlagene Verfahren ist die *Umplanung nach Ausführungsunterbrechung*. Hiermit wird die Problematik noch laufender Aktivitäten gelöst, indem gewartet wird, bis alle laufenden Aktivitäten beendet sind. Da hierdurch unter Umständen unnötig lange mit dem Start neuer Aktivitäten gewartet wird, wurde das Verfahren *Antizipation eines Zustandsraums* entwickelt. Hierbei werden die kausalen Abhängigkeiten zwischen den laufenden Aktivitäten und den Aktivitäten des umgeplanten Prozesses berücksichtigt, um neue Aktivitäten frühestmöglich zu starten. Da bei diesem Verfahren ein Planer nur die Ausführungsdauer des neuen Prozesses optimieren, nicht aber die restliche Ausführungsdauer der daraus resultierenden Prozessinstanz, wurde hierzu das Verfahren *Domänenerweiterung um Stellvertreteraktivitäten* vorgestellt. Bei diesem Verfahren wird die erwartete, restliche Ausführungsdauer laufender Aktivitäten berücksichtigt, indem die Domäne für die Umplanung um Stellvertreteraktivitäten für diese laufenden Aktivitäten erweitert wird. Die Tabelle 4.1 gibt einen Überblick über die vorgestellten, alternativen Verfahren zur Berücksichtigung von Zustandsänderungen der bei der Umplanung laufenden Aktivitäten. Das Verfahren *Domänenerweiterung um Stellvertreteraktivitäten* wird in Kapitel 5 formal modelliert und untersucht.

Verfahren und ihre Eigenschaften	Problematik laufender Aktivitäten während der Umplanung gelöst	Berücksichtigung von kausalen Abhängigkeiten	Berücksichtigung der restlichen Ausführungsdauer laufender Aktivitäten
Umplanung nach Ausführungsunterbrechung	Ja	Nein	Nein
Antizipation eines Zustandsraums	Ja	Ja	Nein
Domänenerweiterung um Stellvertreteraktivitäten	Ja	Ja	Ja

Tabelle 4.1: Vergleich der Verfahren zur Prozessanpassung

Kapitel 5

Formales Modell

In diesem Kapitel wird ein formales Modell einer integrierten Prozessplanung und -ausführung formuliert und untersucht. Die in den Kapiteln 3 und 4 entwickelten Konzepte werden dabei präzisiert und formalisiert. Bezüglich der automatisierten Prozessanpassung wird dabei die in Abschnitt 4.4.3 vorgestellte Lösung *Domänenerweiterung um Stellvertreteraktivitäten* umgesetzt. Neben der Präzisierung der vorgestellten Konzepte, ermöglicht die formale Darstellung ein detaillierteres Verständnis der Verknüpfung von Planung und Ausführung. Des Weiteren erlaubt die Formalisierung den Nachweis einer Reihe von Eigenschaften des Modells. Beispielsweise kann gezeigt werden, unter welchen Bedingungen der Zustand eines Geschäftsfalls durch Planung und Ausführung in einen Zielzustand überführt wird. Ebenso wird nachgewiesen, dass die spezifizierte Vorgehensweise eine konsistente Anpassung eines Prozesses zur Laufzeit ermöglicht.

Zunächst werden die relevanten Abläufe bei der Prozessplanung und -ausführung und die dabei involvierten Artefakte wie zum Beispiel Geschäftsfälle, Aktivitätsdefinitionen, Prozessdefinition und Prozessinstanzen modelliert. Bei der Spezifikation des formalen Modells wird von allen für eine Integration von Planung und Ausführung nicht essenziellen Eigenschaften der zu modellierenden Artefakte abstrahiert. Hierzu gehören auch Aspekte wie Rollenauflösung und Ressourcenzuweisung. Zur Vereinfachung wird in Bezug auf die Prozessdefinition nur sequentieller und nebenläufiger Kontrollfluss betrachtet. Trotz dieser Vereinfachungen ist eine umfangreiche Menge von Definition und Prozedurbeschreibungen notwendig, um die relevanten Aspekte des Systems zu erfassen.

Die grundlegende Idee bei der Formalisierung des Systems ist die Folgende: Die Funktionalität des Planers wird deskriptiv spezifiziert. Das heißt, es wird definiert, unter welchen Bedingungen eine Prozessdefinition für ein Planungsproblem gültig beziehungsweise optimal ist. Damit wird von einem konkreten Planungsalgorithmus abstrahiert und Aussagen können davon unabhängig getroffen werden. Die Funktionalität der Prozesssteuerung und der Prozessüberwachung wird durch Prozeduren und auf Basis von Ereignis-Bedingungs-Aktions-Regeln (engl. Event-Condition-Action Rules) [25] modelliert.

Bezüglich des formalen Modells treffen wir zusätzlich zu den bereits in Abschnitt 3.1.2 aufgestellten Annahmen A1 bis A7 die folgenden Annahmen:

- A8: Der Zustand eines Geschäftsfalls lässt sich durch eine endliche Menge von Aussagen beschreiben.
- A9: Die Effekte einer Aktivität lassen sich in Aussagenlogik beschreiben.
- A10: Bezüglich der Prozeduren besitzt nur die Planung einer Prozessdefinition eine relevante Bearbeitungsdauer. Diese Bearbeitungsdauer ist bekannt.
- A11: Der Zustand eines Geschäftsfalls wird nur durch die Ausführung von Aktivitäten verändert.
- A12: Während der Bearbeitung eines Geschäftsfalls finden keine Änderungen des Ziels oder der Domäne statt.

Die Annahmen A11 und A12 bringen zum Ausdruck, dass während der Bearbeitung eines Geschäftsfalls keine Eingriffe von außen auf den Geschäftsfall oder die Domäne stattfinden. In Abschnitt 5.3.4 wird eine erweiterte Prozessüberwachung vorgestellt, die eine korrekte Bearbeitung von Geschäftsfällen auch für den Fall sicherstellt, dass die Annahmen A11 und A12 nicht zugrunde gelegt werden können. Eine ausführliche Diskussion dieser Annahmen erfolgt in Kapitel 7.

Das grundlegende Modell zur integrierten Planung und Ausführung wird in den Abschnitten 5.1 und 5.2 vorgestellt. Dabei werden aufeinander aufbauend alle wesentlichen Entitäten wie beispielsweise Zustand, Geschäftsfall und Prozessdefinition definiert und Abläufe in Form von Prozeduren spezifiziert. In Abschnitt 5.3 wird das Modell um die Möglichkeit einer automatisierten Prozessanpassung zur Laufzeit erweitert. Zur Illustration wird in dem gesamten Kapitel das Beispiel einer Angebotsanfrage verwendet, das bereits in den vorhergehenden Kapiteln vorgestellt wurde. Dafür wird das Beispiel noch einmal in formaler Form eingeführt.

5.1 Planung

Die Grundlage des formalen Modells bilden der Geschäftsfall und sein Zustand. Ein Zustand ist wie folgt modelliert:

Definition 1 Zustand Sei $L = \{A_1, ..., A_n\}$ eine endliche Menge von Aussagensymbolen. Ein Zustand S ist ein Paar $S = (prop_S, t_S)$, wobei

- $prop_S \subseteq L$ als eine Konjunktion von Aussagen verstanden wird, die den Zustand beschreibt. Eine Aussage A ist wahr in dem Zustand S genau dann, wenn $A \in prop_S$,
- $t_S \in \mathbb{R}$ der Zeitpunkt des Zustandes ist.

$\diamond$

Die Menge $prop_S$ legt damit fest, welche Aussagen aus L zum Zeitpunkt t_S zutreffend sind. Die Modellierung eines Zeitpunkts t_S für einen Zustand S ist dabei wesentlich, um im Folgenden darauf aufbauend die Semantik der Ausführungsdauer von Aktivitäten erfassen zu können.

Beispiel 1 An dieser Stelle wird das Beispiel einer Angebotsanfrage wieder aufgegriffen, das bereits in den vorhergehenden Kapiteln verwendet wurde. Allerdings wird es hier in formaler Form dargestellt. Zunächst definieren wir die Menge L aller Aussagen, mit denen ein Zustand beschrieben werden kann.
Sei $L = \{A_{anfrage}, A_{erfasst}, A_{exp\_deu}, A_{exp\_eng}, A_{kalkuliert}, A_{beantwortet}\}$, wobei die einzelnen Aussagen folgende Bedeutung haben:
$A_{anfrage}$: Eine Nachricht mit einer Angebotsanfrage ist eingegangen.
$A_{erfasst}$: Die Angebotsanfrage ist erfasst.
A_{exp_deu} : Die Exportbedingungen für das Produkt liegen in Deutsch vor.
A_{exp_eng} : Die Exportbedingungen für das Produkt liegen in Englisch vor.
$A_{kalkuliert}$: Der Preis für das Produkt ist kalkuliert.
$A_{beantwortet}$: Die Angebotsanfrage ist beantwortet.
Folglich wird der Zustand, der ausdrückt, dass zum Zeitpunkt $t_{Eingang} = 0$ eine Angebotsanfrage vorliegt, durch das Paar $S = (\{A_{anfrage}\}, t_{Eingang})$ beschrieben. ◇

Ein Zustand kann durch die Ausführung von Aktivitäten verändert werden. Dabei legt eine Aktivitätsdefinition die erwarteten Ausführungseigenschaften einer Aktivität fest.

Definition 2 Aktivitätsdefinition Eine Aktivitätsdefinition d ist ein Tupel $d = (pre_d, eff_d^-, eff_d^+, dur_d)$, wobei

- $pre_d \subseteq L$ eine Menge von Aussagen ist, die als *Vorbedingungen* von d bezeichnet werden. Die Vorbedingungen sind in einem Zustand S *erfüllt* genau dann, wenn $pre_d \subseteq prop_S$.
- $eff_d^-, eff_d^+ \subseteq L$ Mengen von Aussagen sind, die als *Effekte* von d bezeichnet werden. Die beiden Mengen von Effekten müssen disjunkt sein, das heißt, $eff_d^- \cap eff_d^+ = \emptyset$.
- $dur_d \in \mathbb{R}^{\geq 0}$ die erwartete Ausführungsdauer ist.

Die Menge aller Aktivitätsdefinitionen wird als Domäne D bezeichnet. ◇

Die beiden Mengen eff_d^- und eff_d^+ beschreiben die Auswirkungen einer Ausführung einer Aktivität d auf den Zustand in dem sie ausgeführt wird. Der Zustandsübergang erfolgt bei Beendigung der Aktivität. Die neue Menge $prop_{S'}$ des Folgezustands S' eines Zustands S ergibt sich unter Berücksichtigung der Effekte von d wie folgt: $prop_{S'} = (prop_S \setminus eff_d^-) \cup eff_d^+$. Dies gilt allerdings nur für den Fall, dass keine anderen Aktivitäten gleichzeitig enden. In allgemeiner Form wird die

Zustandsänderung durch die Ausführung von Aktivitäten in Definition 9 beschrieben. Vorbedingungen legen die Zustände fest, in denen eine Aktivität ausgeführt werden muss, um die spezifizierten Effekte zu haben.

Beispiel 2 Zur Fortsetzung des Beispiels folgt nun die Spezifikation der Domäne D mit den Aktivitätsdefinitionen, die zur Bearbeitung einer Angebotsanfrage zur Verfügung stehen:
$D = \{d_{erfassen}, d_{pruefen}, d_{kalkulieren}, d_{antworten}, d_{erfassen}, d_{antworten2}, d_{uebersetzen}\}$
wobei die einzelnen Aktivitätsdefinitionen wie folgt spezifiziert sind:
$d_{erfassen} = (\{A_{anfrage}\}, \emptyset, \{A_{erfasst}\}, 2)$
$d_{pruefen} = (\{A_{erfasst}\}, \emptyset, \{A_{exp\_deu}\}, 4)$
$d_{kalkulieren} = (\{A_{erfasst}\}, \emptyset, \{A_{kalkuliert}\}, 8)$
$d_{antworten} = (\{A_{exp\_deu}, A_{kalkuliert}\}, \emptyset, \{A_{beantwortet}\}, 3)$
$d_{antworten2} = (\{A_{exp\_eng}, A_{kalkuliert}\}, \emptyset, \{A_{beantwortet}\}, 4)$
$d_{uebersetzen} = (\{A_{exp\_eng}\}, \emptyset, \{A_{exp\_deu}\}, 2)$
Die Bedeutung der Aktivitätsdefinitionen ist damit durch ihre Vorbedingungen und Effekte bestimmt. Beispielsweise hat $d_{kalkulieren}$ die Vorbedingung $A_{erfasst}$ (das heißt die Angebotsanfrage ist erfasst) und den Effekt $A_{kalkuliert}$ (das heißt der Preis für das Produkt ist kalkuliert).

In dem Beispiel sind die Mengen eff_d^- leer, da es sich um einen rein informationsverarbeitenden Prozess handelt und keine Informationen gelöscht werden. Würde beispielsweise die eingegangene Nachricht mit der Angebotsanfrage nach Erfassung gelöscht, dann wäre die entsprechende Aktivitätsdefinition $d^*_{erfassen} = (\{A_{anfrage}\}, \{A_{anfrage}\}, \{A_{erfasst}\}, 2)$. Zur Angemessenheit der Modellierung sei gesagt, dass durch die Beschränkung auf Aussagenlogik anstelle von Prädikatenlogik beispielsweise die Aktivitätsdefinition $d_{uebersetzen}$ sehr spezifisch nur für Exportbedingungen definiert ist. Bei Verwendung von Prädikatenlogik ließe sich eine solche Aktivität allgemeiner formulieren, so dass jedes Dokument übersetzt werden kann. Demgegenüber muss bei Verwendung von Aussagenlogik für jedes Dokument eine dedizierte Übersetzungs-Aktivität spezifiziert werden.

In dem Zustand $S = (\{A_{anfrage}\}, t_{Eingang})$ sind als einzige die Vorbedingungen der Aktivitätsdefinition $d_{erfassen}$ erfüllt. Falls diese Aktivität ausgeführt wird, gilt für den Folgezustand S':
$\mathit{prop}_{S'} = (\mathit{prop}_S \setminus \mathit{eff}^-_{d_{erfassen}}) \cup \mathit{eff}^+_{d_{erfassen}} \Leftrightarrow$
$\mathit{prop}_{S'} = (\{A_{anfrage}\} \setminus \emptyset) \cup \{A_{erfasst}\} \Leftrightarrow$
$\mathit{prop}_{S'} = \{A_{anfrage}, A_{erfasst}\}$
Wie bereits oben erwähnt, gilt dies nur, falls keine anderen Aktivitäten gleichzeitig enden. Allgemein wird die Ausführung von Aktivitäten in Definition 9 beschrieben. ◇

Definition 3 Ziel Ein Ziel $\mathit{goal} \subseteq L$ bestimmt die Anforderungen an einen Zustand, die dieser erfüllen muss, damit er ein *Zielzustand* ist. Ein Zustand S ist ein Zielzustand in Bezug auf ein Ziel *goal* genau dann, wenn $\mathit{goal} \subseteq \mathit{prop}_S$. ◇

Ein Zustand ist also genau dann ein Zielzustand, wenn alle Aussagen des Ziels in dem Zustand wahr sind. Aufbauend auf den Definitionen von Zustand und Ziel, wird nun ein Geschäftsfall definiert:

Definition 4 Geschäftsfall Ein Geschäftsfall c ist ein Paar c = (*state*$_c$, *goal*$_c$), wobei *state*$_c$ der aktuelle Zustand des Geschäftsfalls und *goal*$_c$ das Ziel des Geschäftsfalls ist.

◇

Somit ist ein Geschäftsfall durch seinen aktuellen Zustand und sein Ziel charakterisiert. Weitere Informationen, die zu einem Geschäftsfall gehören, wie zum Beispiel Daten und Dokumente, werden nicht modelliert, da sie für die hier vorgestellte Integration von Prozessplanung und -ausführung nicht relevant sind.

Beispiel 3 Das Ziel im laufenden Beispiel sei *goal* = $\{A_{beantwortet}\}$. Somit ist zum Beispiel der Zustand S mit *prop*$_S$ = $\{A_{erfasst}, A_{beantwortet}\}$ ein Zielzustand. Der entsprechende Geschäftsfall sei $c = ((\{A_{anfrage}\}, t_{Eingang}), \{A_{beantwortet}\})$. Das heißt, es liegt eine Angebotsanfrage zum Zeitpunkt $t_{Eingang}$ vor, die beantwortet werden soll. ◇

Ist der aktuelle Zustand wie in dem Beispiel kein Zielzustand, so muss der Geschäftsfall bearbeitet werden, um seinen Zustand in einen Zielzustand zu überführen. Dies geschieht in folgenden Schritten: Der erste Schritt ist die Spezifikation eines entsprechenden Planungsproblems. Anschließend wird eine Prozessdefinition ermittelt, die eine Lösung für dieses Planungsproblem darstellt. Falls keine Prozessdefinition gefunden werden kann, ist ein manueller Eingriff notwendig. Ansonsten wird die Prozessdefinition instanziiert und die Instanz gestartet. Dieser Ablauf zur Bearbeitung eines Geschäftsfalls wird nun formal als Prozedur 1 *bearbeiteGeschäftsfall* spezifiziert. Entsprechend dem Ablauf dieser Prozedur werden im Folgenden die benötigten Entitäten und Unterprozeduren beschrieben. Im ersten Schritt wird ein Planungsproblem für den Geschäftsfall erzeugt.

Definition 5 Planungsproblem Ein Planungsproblem T ist ein Tupel T = (*act*$_T$, *init*$_T$, *goal*$_T$), wobei *act*$_T \subseteq D$ eine endliche Menge von Aktivitätsdefinitionen, *init*$_T$ der *Ausgangszustand* und *goal*$_T$ das Ziel ist. ◇

Ein Planungsproblem umfasst neben Ziel und Ausgangszustand die Menge der verfügbaren Aktivitätsdefinitionen. Die Prozedur 2 *erzeugePlanungsproblem* liefert das Planungsproblem für einen gegebenen Geschäftsfall. Die Domäne D bilden dabei die verfügbaren Aktivitäten der Organisation. Das Planungsproblem bildet die Eingabe für die Planung. Dies entspricht auch dem konzeptionellen Modell in Abbildung 3.1, in der die zur Planung benötigten Informationen aus dem Geschäftsfall und aus der Domäne kommen. Die Ausgabe der Planung ist eine Prozessdefinition, die ein Lösung für das Planungsproblem darstellt.

Prozedur 1 bearbeiteGeschäftsfall(c)

Eingabe: Geschäftsfall c
Variablen:
Planungsproblem T
Prozessdefinition P
Prozessinstanz I
Funktion ϕ
Boolean *gefunden*

erzeugePlanungsproblem(c, T)
planeProzess(T, P, *gefunden*)
if *gefunden* **then**
instanziiereProzess(P, I, ϕ)
starteInstanzen(P, I, ϕ)
else
manueller Eingriff
end if

Prozedur 2 erzeugePlanungsproblem(c, T)

Eingabe: Geschäftsfall c
Ausgabe: Planungsproblem T

erzeuge neues Planungsproblem T
$init_T := state_c$
$goal_T := goal_c$
$act_T := D$

Definition 6 Prozessdefinition Eine Prozessdefinition P ist ein Paar (G_P, δ_P), wobei

- G_P = (rep_P, con_P) ein Graph ist, so dass
  - rep_P eine endliche Menge von Knoten ist, die als *Aktivitätsrepräsentanten* bezeichnet werden. Jeder Aktivitätsrepräsentant repräsentiert eine Aktivitätsdefinition im Kontext der Prozessdefinition.
  - $con_P \subseteq rep_P \times rep_P$ eine Menge von gerichteten Kanten ist, die als *Kontrollkonnektoren* bezeichnet werden. Ein Aktivitätsrepräsentant r_x wird genau dann als *direkter Vorgänger* eines Aktivitätsrepräsentanten r_y in P bezeichnet, wenn $(r_x, r_y) \in con_P$.
  - der Graph azyklisch ist. das heißt es existiert keine Folge W von Aktivitätsrepräsentanten aus rep_P mit $W = (r_1, ..., r_m)$, so dass:
    1. $\forall\ i \in \{1, ..., m-1\} : (r_i, r_{i+1}) \in con_P$, das heißt W einen gerichteten Weg in G_P darstellt.
    2. $r_1 = r_m$, das heißt Start- und Endknoten von W identisch sind.
- δ_P eine Funktion von rep_P nach D ist, die jedem Aktivitätsrepräsentanten aus rep_P die Aktivitätsdefinition zuordnet, die er repräsentiert.

◇

Zusammenfassend ist eine Prozessdefinition als ein Paar aus einem gerichteten, azyklischen, endlichen Graphen ohne Mehrfachkanten und einer Abbildung dessen Knoten auf Aktivitätsdefinitionen definiert. Eine Prozessdefinition drückt aus, welche Aktivitäten ausgeführt werden müssen, um den Ausgangszustand in einen Zielzustand zu überführen. Kontrollkonnektoren schränken dabei die Nebenläufigkeit der Ausführung ein, indem sie festlegen, welche Aktivitäten beendet sein müssen, bevor eine bestimmte Aktivität gestartet werden kann. Dabei ist die Menge der Kontrollkonnektoren nicht direkt auf der Menge der Aktivitätsdefinitionen definiert, da eine Aktivitätsdefinition auch mehrfach in einer Prozessdefinition auftreten kann. Aus diesem Grund werden Aktivitätsdefinitionen in einer Prozessdefinition durch Aktivitätsrepräsentanten vertreten.

Beispiel 4 Betrachten wir die Prozessdefinition $P = ((rep_P, con_P), \delta_P)$, wobei
$rep_P = \{r_1, r_2, r_3, r_4\}$
$con_P = \{(r_1, r_2), (r_1, r_3), (r_3, r_4), (r_2, r_4)\}$
$\delta_P = \{(r_1, d_{erfassen}), (r_2, d_{pruefen}), (r_3, d_{kalkulieren}), (r_4, d_{antworten})\}$
Der entsprechende Graph $G_P = (rep_P, con_P)$ ist in Abbildung 5.1 dargestellt. Die Knoten stellen Aktivitätsrepräsentanten dar, wobei in Klammern für jeden Aktivitätsrepräsentanten r auch die Aktivitätsdefinition $d = \delta_P(r)$ angegeben ist. Die Vorbedingungen von $d_{erfassen}$ sind bereits im Ausgangszustand erfüllt. Somit kann $d_{erfassen}$ direkt zu Beginn des Prozesses ausgeführt werden. $d_{erfassen}$

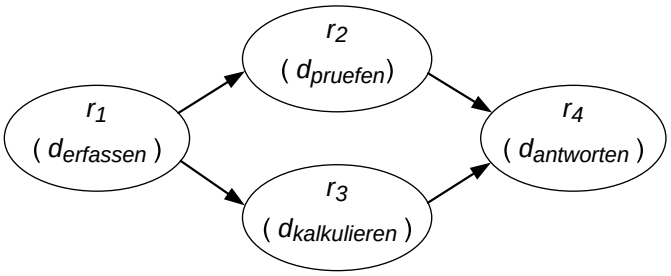


Abbildung 5.1: Graph der Prozessdefinition zur Angebotsanfrage

hat den Effekt $A_{erfasst}$, der eine Vorbedingung der Aktivität $d_{kalkulieren}$ ist. Es besteht demnach eine kausale Verknüpfung zwischen $d_{erfassen}$ und $d_{kalkulieren}$. Diese kausale Verknüpfung impliziert eine Ausführungsreihenfolge, die sich im entsprechenden Kontrollkonnektor widerspiegelt. ◇

In dem Beispiel 4 wurde bereits eine Prozessdefinition vorgestellt, bei deren Ausführung der Ausgangszustand in einen Zielzustand überführt wird. Die formale Definition von Gültigkeit und Optimalität einer Prozessdefinition für ein Planungsproblem folgt weiter unten. Hierzu müssen zunächst noch Instanzen von Aktivitäten und Prozessen, sowie die Auswirkung ihrer Ausführung auf den Zustand des Geschäftsfalls definiert werden. Eine konkrete Ausführung einer Prozessdefinition wird in Form einer Prozessinstanz modelliert, und entsprechend die Ausführung einer Aktivitätsdefinition in Form einer Aktivitätsinstanz.

Definition 7 Aktivitätsinstanz Eine Aktivitätsinstanz i ist ein Tupel i = ($exec_i$, $t_{start}(i)$, $t_{end}(i)$, pre_i, eff_i^-, eff_i^+), wobei

- $exec_i \in$ {aktiviert, deaktiviert, laufend, beendet} der Ausführungszustand von i ist.
- $t_{start}(i) = \begin{cases} \text{Startzeitpunkt } t_S \in \mathbb{R} & : \quad exec_i \in \{\text{laufend, beendet}\} \\ \text{undefiniert} & : \quad \text{sonst} \end{cases}$
- $t_{end}(i) = \begin{cases} \text{Endzeitpunkt } t_E \in \mathbb{R} \text{ mit } t_E \geq t_S & : \quad exec_i \in \{\text{beendet}\} \\ \text{undefiniert} & : \quad \text{sonst} \end{cases}$
- $pre_i \subseteq L$ eine Menge von Aussagen ist, die als *Vorbedingungen* bezeichnet werden.
- eff_i^-, $eff_i^+ \subseteq L$ Mengen von Aussagen sind, die als *Effekte* bezeichnet werden.

Eine Aktivitätsinstanz i ist genau dann beendet, wenn $exec_i$ = beendet. Eine Aktivitätsinstanz i ist eine Instanz einer Aktivitätsdefinition d genau dann, wenn $(pre_i = pre_d) \wedge (eff_i^- = eff_d^-) \wedge (eff_i^+ = eff_d^+)$. Die Differenz $t_{end}(i) - t_{start}(i)$ einer

beendeten Aktivitätsinstanz i wird als *Ausführungsdauer* von i bezeichnet. Eine beendete Aktivitätsinstanz i ist eine *ausführungsdauer-treue* Instanz einer Aktivitätsdefinition d genau dann, wenn i eine Instanz von d ist und die Ausführungsdauer von i gleich der erwarteten Ausführungsdauer von d ist. $\diamond$

Eine Aktivitätsinstanz besitzt im Gegensatz zu einer Aktivitätsdefinition konkrete Start- und Endzeitpunkte. Des Weiteren hat sie einen Ausführungszustand der ihren Bearbeitungsfortschritt widerspiegelt. Die möglichen Zustandsübergänge sind bereits in Abbildung 3.3 auf Seite 29 dargestellt. Der initiale Zustand für einer Aktivitätsinstanz i ist der Zustand *aktiviert*. Dies bedeutet sie ist startbereit. Wird sie gestartet befindet sie sich im Zustand *laufend*. Der Startzeitpunkt wird mit $t_{start}(i)$ bezeichnet und der Endzeitpunkt mit $t_{end}(i)$. Die Deaktivierung einer Aktivitätsinstanz wird im Rahmen der Umplanung im Kapitel 4 erläutert.

Definition 8 Prozessinstanz Eine Prozessinstanz ist eine endliche Menge von Aktivitätsinstanzen. Eine Prozessinstanz ist beendet genau dann, wenn alle ihre Aktivitätsinstanzen beendet sind. Eine Prozessinstanz I ist eine ϕ-Instanz einer Prozessdefinition P genau dann, wenn folgende Kriterien erfüllt sind:

- *Vollständigkeit:* ϕ ist eine totale, bijektive Funktion von I nach rep_P, so dass für alle $i \in I$ gilt: i ist eine Instanz der Aktivitätsdefinition $\delta_P(\phi(i))$.
- *Ausführungskonsistenz:* Für alle $i, j \in I$ mit $(\phi(i), \phi(j)) \in con_P$ gilt:
  1. $exec_i \notin$ {beendet} $\Rightarrow$ $exec_j \notin$ {laufend, beendet}
  2. $exec_i \in$ {beendet} $\wedge$ $exec_j \in$ {laufend, beendet} $\Rightarrow t_{end}(i) \leq t_{start}(j)$

Eine Prozessinstanz I ist eine Instanz einer Prozessdefinition P genau dann, wenn eine Funktion ϕ existiert, so dass I eine ϕ-Instanz von P ist. Die Ausführungsdauer einer beendeten Prozessinstanz I ist die Dauer vom Start der ersten Aktivitätsinstanz bis zum Ende der letzten Aktivitätsinstanz, das heißt die Differenz $\max_{i \in I}(t_{end}(i)) - \min_{j \in I}(t_{start}(j))$. I ist eine *ausführungsdauer-treue* ϕ-Instanz einer Prozessdefinition P, genau dann wenn I eine ϕ-Instanz von P ist und für jede beendete Aktivitätsinstanz $i \in I$ gilt, dass i eine ausführungsdauer-treue Aktivitätsinstanz der Aktivitätsdefinition $\delta_P(\phi(i))$ ist. $\diamond$

In Definition 8 wird der Begriff der ϕ-Instanz unter Berücksichtigung des Ausführungszustands der Prozessinstanz definiert. Dies ist wichtig, da wir im Folgenden für die Beweisführung die ϕ-Instanz-Beziehung sowohl für beendete, als auch für neu instanziierte und für in Ausführung befindliche Prozessinstanzen betrachten. Für den Spezialfall einer beendeten Prozessinstanz vereinfacht sich das Kriterium für die Ausführungskonsistenz wie folgt: Für alle $i, j \in I$ mit $(\phi(i), \phi(j)) \in con_P$ gilt: $t_{end}(i) \leq t_{start}(j)$. das heißt es darf keine Aktivitätsinstanz geben, die ihren Startzeitpunkt vor dem Endzeitpunkt eines ihrer direkten Vorgänger in Bezug auf die zugeordneten Aktivitätsrepräsentanten hat.

Beispiel 5 Betrachten wir die Prozessinstanz $I = \{i_1, i_2, i_3, i_4\}$, wobei für die einzelnen Aktivitätsinstanzen gilt:
$i_1 = (\text{beendet}, 0, 2, \{A_{anfrage}\}, \emptyset, \{A_{erfasst}\})$
$i_2 = (\text{beendet}, 2, 6, \{A_{erfasst}\}, \emptyset, \{A_{exp\_deu}\})$
$i_3 = (\text{laufend}, 2, \text{undefiniert}, \{A_{erfasst}\}, \emptyset, \{A_{kalkuliert}\})$
$i_4 = (\text{aktiviert}, \text{undefiniert}, \text{undefiniert}, \{A_{exp\_deu}, A_{kalkuliert}\}, \emptyset, \{A_{beantwortet}\})$
Betrachten wir diese Prozessinstanz I und die Prozessdefinition P aus Beispiel 4, so gilt für folgende Funktion ϕ, dass I eine ϕ-Instanz von P ist:

$$\phi = \{(i_1, r_1), (i_2, r_2), (i_3, r_3), (i_4, r_4)\}$$

Nach Definition 8 sind sowohl die Vollständigkeit als auch die Ausführungskonsistenz gegeben. Die Aktivitätsinstanz i_1 ist eine ausführungsdauer-treue Instanz der Aktivitätsdefinition $\delta_P(\phi(i_1)) = d_{erfassen}$. Ebenso ist die Instanz i_2 ist eine ausführungsdauer-treue Instanz der Aktivitätsdefinition $\delta_P(\phi(i_2)) = d_{pruefen}$. $\diamond$

Im Folgenden wird beschrieben, welche Zustandsänderung durch die Ausführung von Aktivitätsinstanzen einer Prozessinstanz erwartet wird. Dabei werden die Aktivitätsinstanzen die zum gleichen Zeitpunkt enden zu einem *Geschehnis* zusammengefasst.

Definition 9 Geschehnis Die *Endzeitsequenz* $\{t_x\}_{x=0\ldots k}$ für eine Prozessinstanz I ist die geordnete Menge der Zeitpunkte über die Menge der Endzeitpunkte $t_{end}(i)$ aller beendeten Aktivitätsinstanzen $i \in I$. Das *Geschehnis* $H \subseteq I$ zum Zeitpunkt $t \in \{t_x\}_{x=0\ldots k}$ ist die Menge der Aktivitätsinstanzen $i \in I$ mit der Endzeit $t_{end}(i) = t$. Ein Geschehnis H ist *konsistent* genau dann, wenn keine Aktivitätsinstanz $i \in H$ den negierten Effekt einer anderen hat, das heißt, $\bigcup_{i \in H} \mathit{eff}_i^- \cap \bigcup_{j \in H} \mathit{eff}_j^+ = \emptyset$. Eine Aktivitätsinstanz $i \in H$ ist genau dann *anwendbar*, wenn ihre Vorbedingungen für jeden Zustand während ihrer Ausführungsdauer erfüllt sind. Das erwartete Resultat des Eintritts eines Geschehnisses H zu einer Zeit $t_H \in \{t_x\}_{x=0\ldots k}$ in dem Zustand $S = (\mathit{prop}_S, t_S)$ ist undefiniert genau dann, wenn H nicht konsistent ist oder eine Aktivitätsinstanz aus H nicht anwendbar ist. Ansonsten ist das erwartete Resultat der Zustand $S' = (\mathit{prop}_{S'}, t_H)$, wobei

$$\mathit{prop}_{S'} = (\mathit{prop}_S \setminus \bigcup_{i \in H} \mathit{eff}_i^-) \cup \bigcup_{i \in H} \mathit{eff}_i^+$$

$\diamond$

Beispiel 6 Betrachten wir die Prozessinstanz I aus Beispiel 5. Die entsprechende Endzeitsequenz enthält die Endzeiten der beiden beendeten Aktivitätsinstanzen: t_0 = 2 und t_1 = 6. Für das Geschehnis $H_0 = \{i_1\}$ zum Zeitpunkt t_0 = 2 gilt: Da zum Zeitpunkt t_0 = 2 nur eine Aktivitätsinstanz endet, enthält das Geschehnis H_0 nur ein Element. Somit hat keine Aktivitätsinstanz den negierten Effekt einer anderen und H_0 ist konsistent. Der Ausgangszustand S_0 sei der in Beispiel 1 beschriebene:

$S_0 = (\{A_{anfrage}\}, 0)$, das heißt zum Zeitpunkt $t_{Eingang} = 0$ liegt eine Angebotsanfrage vor. Da bis zum Zeitpunkt $t_0 = 2$ keine Zustandsänderungen auftreten, gilt für jeden Zustand S mit $0 \leq t_S \leq 2$ gilt, dass $pre_{i_1} \subseteq prop_{S_0}$. Damit ist i_1 anwendbar. Folglich ist das erwartete Resultat des Eintritts des Geschehnisses H_0 zum Zeitpunkt $t_0 = 2$ in dem Zustand $S_0 = (\{A_{anfrage}\}, 0)$ der Zustand S_1 = (*prop*$_{S_1}$, 2), wobei *prop*$_{S_1} = (\{A_{anfrage}\} \setminus \emptyset) \cup \{A_{erfasst}\}$. Betrachten wir nun das Geschehnis $H_1 = \{i_2\}$ zum Zeitpunkt t_1 = 6. H_1 ist konsistent und i_2 ist anwendbar. Folglich ist das erwartete Resultat des Eintritts des Geschehnisses H_1 zum Zeitpunkt $t_1 = 6$ in dem Zustand $S_1 = (\{A_{anfrage}, A_{erfasst}\}, 2)$ der Zustand $S_2 = (\{A_{anfrage}, A_{erfasst}, A_{exp\_deu}\}, 6)$. ◇

Aufbauend auf der Definition eines Geschehnisses können nun Eigenschaften von Prozessinstanzen und Prozessdefinitionen bezüglich eines Planungsproblems definiert werden.

Definition 10 Ausführbarkeit und Gültigkeit einer Prozessinstanz für ein Planungsproblem Eine Prozessinstanz I für ein Planungsproblem T ist *ausführbar* genau dann, wenn I eine Endzeitsequenz $\{t_x\}_{x=0...k}$ definiert und es eine Folge von Zuständen $\{S_x\}_{x=0...k+1}$ gibt, so dass S_0 der Ausgangszustand *init*$_T$ von T ist und für alle $x = 0...k$, der Zustand S_{x+1} das erwartete Resultat des Eintritts des Geschehnisses zum Zeitpunkt t_x ist. Der Zustand S_{k+1} wird als der *finale Zustand* von I bezeichnet und die Folge von Zuständen $\{S_x\}_{x=0...k+1}$ als die *Spur* von I. Eine beendete Prozessinstanz I für ein Planungsproblem T ist *gültig* genau dann, wenn I ausführbar ist und einen finalen Zustand hat der ein Zielzustand von T ist, das heißt *goal*$_T \subseteq$ *prop*$_{S_{x+1}}$. ◇

Beispiel 7 Gegeben sei ein Planungsproblem T = (*act*$_T$, *init*$_T$, *goal*$_T$). Es gilt *act*$_T = D$, *init*$_T$ = $(\{A_{anfrage}\}, 0)$ und *goal*$_T$ = $\{A_{beantwortet}\}$. Betrachten wir diesbezüglich die Prozessinstanz I, die dazugehörige Endzeitsequenz und den Ausgangszustand S_0 aus Beispiel 6. Der Ausgangszustand S_0 entspricht dem Ausgangszustand *init*$_T$. Wie in Beispiel 6 gezeigt, definiert I eine Endzeitsequenz im Sinne von Definition 10. Somit ist I für T ausführbar. Der finale Zustand von I ist S_2 = $(\{A_{anfrage}, A_{erfasst}, A_{exp\_deu}\}$, 6). Da I nicht beendet ist, kann I nicht gültig sein.

Betrachten wir nun eine Prozessinstanz I', die sich von I dadurch unterscheidet, dass sie bereits beendet ist:
$I' = \{i'_1, i'_2, i'_3, i'_4\}$, wobei für die einzelnen Aktivitätsinstanzen gilt:
$i'_1 = (\text{beendet}, 0, 2, \{A_{anfrage}\}, \emptyset, \{A_{erfasst}\})$
$i'_2 = (\text{beendet}, 2, 6, \{A_{erfasst}\}, \emptyset, \{A_{exp\_deu}\})$
$i'_3 = (\text{beendet}, 2, 10, \{A_{erfasst}\}, \emptyset, \{A_{kalkuliert}\})$
$i'_4 = (\text{beendet}, 10, 13, \{A_{exp\_deu}, A_{kalkuliert}\}, \emptyset, \{A_{beantwortet}\})$
Die Spur von I' bezüglich T ist die Zustandsfolge $< S'_0, S'_1, S'_2, S'_3, S'_4 >$, wobei
S'_0 = $(\{A_{anfrage}\}$, 0)
S'_1 = $(\{A_{anfrage}, A_{erfasst}\}$, 2)

S'_2 = ($\{A_{anfrage}, A_{erfasst}, A_{exp\_deu}\}$, 6)
S'_3 = ($\{A_{anfrage}, A_{erfasst}, A_{exp\_deu}, A_{kalkuliert}\}$, 10)
S'_4 = ($\{A_{anfrage}, A_{erfasst}, A_{exp\_deu}, A_{kalkuliert}, A_{beantwortet}\}$, 13)
Somit ist I' für T ausführbar. Des Weiteren gilt für den finalen Zustand S'_4 von I', dass $\mathit{goal}_T \subseteq \mathit{prop}_{S_4}$. Folglich ist I' für T gültig. ◇

Definition 11 Gültigkeit und Optimalität einer Prozessdefinition für ein Planungsproblem Eine Prozessdefinition P für ein Planungsproblem T ist *gültig* genau dann, wenn alle beendeten Prozessinstanzen, die eine Instanz von P sind und bei denen der Eintritt jedes Geschehnisses das erwartete Resultat hat, gültig sind. Die *minimale Ausführungsdauer* von P ist die kürzeste Ausführungsdauer aller beendeten, ausführungsdauer-treuen Prozessinstanzen, die Instanz von P sind. P ist *optimal* genau dann, wenn P gültig ist und keine andere gültige Prozessdefinition mit einer kürzeren minimalen Ausführungsdauer existiert. ◇

Die Eigenschaften einer Prozessdefinition werden über die Eigenschaften aller existierender Instanzen der Prozessdefinition definiert. Wesentlich ist hierbei die Einschränkung, dass dabei nur Prozessinstanzen berücksichtigt werden, bei denen der Eintritt jedes Geschehnisses das erwartete Resultat hat. Ohne diese Einschränkung wäre eine sinnvolle Definition nicht möglich, da bezüglich Prozessinstanzen mit unerwarteten Resultaten von Geschehnissen keine Aussagen über das Erreichen eines Zielzustands gemacht werden können.

Basierend auf der Definition der Gültigkeit und Optimalität einer Prozessdefinition für ein Planungsproblem folgt nun die Beschreibung der *planeProzess*-Prozedur.

Prozedur 3 planeProzess(T, P, *gefunden*)

Eingabe: Planungsproblem T
Ausgabe: Prozessdefinition P, Boolean *gefunden*

Falls gültige Prozessdefinitionen für das Planungsproblem T existieren,
wird eine gültige Prozessdefinitionen P und *gefunden* = true
zurückgeliefert. Ansonsten ist P undefiniert und *gefunden* = false.

Wie Eingangs dieses Unterkapitels erläutert, wurde hier eine deskriptive Spezifikation der Planungs-Prozedur gewählt, um das formale Modell unabhängig von einem konkreten Planungsalgorithmus zu halten. Stattdessen wurden die wesentlichen, relevanten Konzepte wie Zustand, Endzeitsequenz und Geschehnis in Anlehnung an die formale Semantik von PDDL 2.1 [41] definiert. Die Idee hierbei ist, dass jeder Planer, der in der Lage ist, PDDL 2.1 zu handhaben, auch in der Lage ist, eine gültige Prozessdefinition für ein Planungsproblem zu liefern.

5.2 Ausführung

In diesem Abschnitt wird beschrieben, wie eine Prozessdefinition instanziiert und ihre Instanz ausgeführt wird. Bei der Ausführung werden die in der Prozessdefinition spezifizierten Aktivitäten ausgeführt. Dabei wird die Nebenläufigkeit durch Kontrollkonnektoren eingeschränkt. Die Ausführung beginnt mit der Instanziierung. Das bedeutet, dass zu einer gegebenen Prozessdefinition eine Prozessinstanz mit den entsprechenden Aktivitätsinstanzen erzeugt wird.

Prozedur 4 instanziiereAktivität(d, i)

Eingabe: Aktivitätsdefinition d
Ausgabe: Aktivitätsinstanz i

erzeuge neue Aktivitätsinstanz i
$exec_i :=$ aktiviert
$pre_i := pre_d$
$eff_i^+ := eff_d^+$
$eff_i^- := eff_d^-$

Lemma 1 Die Prozedur *instanziiereAktivität(d, i)* erzeugt zu einer Aktivitätsdefinition d eine Aktivitätsinstanz i die Instanz von d ist.

Beweis Es gilt offensichtlich $(pre_i = pre_d) \wedge (eff_i^- = eff_d^-) \wedge (eff_i^+ = eff_d^+)$. Damit ist i nach Definition 7 eine Instanz von d. □

Aufbauend auf der Instanziierung von Aktivitätsdefinitionen durch die Prozedur 4, beschreibt die Prozedur 5 die Instanziierung einer Prozessdefinition.

Prozedur 5 instanziiereProzess(P, I, ϕ)

Eingabe: Prozessdefinition P
Ausgabe: Prozessinstanz I, Funktion ϕ
Variablen: Aktivitätsinstanz i

$I := \emptyset$
$\phi := \emptyset$
for all $r \in rep_P$ **do**
  instanziiereAktivität($\delta_P(r)$, i)
  füge i zu I hinzu
  füge (i, r) zu ϕ hinzu
end for

Beispiel 8 Gegeben sei die in Beispiel 4 dargestellte Prozessdefinition P. Die Ausführung der Prozedur *instantiiereProzess(P, I, ϕ)* liefert dann eine Prozessinstanz

$I = \{i_1, i_2, i_3, i_4\}$, wobei
$i_1 =$ (aktiviert, undefiniert, undefiniert, $\{A_{anfrage}\}, \emptyset, \{A_{erfasst}\}$)
$i_2 =$ (aktiviert, undefiniert, undefiniert, $\{A_{erfasst}\}, \emptyset, \{A_{exp\_deu}\}$)
$i_3 =$ (aktiviert, undefiniert, undefiniert, $\{A_{erfasst}\}, \emptyset, \{A_{kalkuliert}\}$)
$i_4 =$ (aktiviert, undefiniert, undefiniert, $\{A_{exp\_deu}, A_{kalkuliert}\}, \emptyset, \{A_{beantwortet}\}$)
und die Funktion $\phi = \{(i_1, r_1), (i_2, r_2), (i_3, r_3), (i_4, r_4)\}$. Es gilt, dass I eine ϕ-Instanz von P ist, da nach Definition 8 sowohl die Vollständigkeit als auch die Ausführungskonsistenz gegeben sind (vergleiche Beispiel 5). $\diamond$

Lemma 2 Die Prozedur *instantiiereProzess(P, I, ϕ)* erzeugt zu einer Prozessdefinition P eine Prozessinstanz I und eine Funktion ϕ, so dass I eine ϕ-Instanz von P ist.

Beweis Es wird gezeigt, dass I eine ϕ-Instanz von P ist, indem wir die in Definition 8 geforderte Vollständigkeit und Ausführungskonsistenz nachweisen:

- Vollständigkeit: Zum Nachweis der Vollständigkeit ist zu zeigen, dass ϕ eine totale, bijektive Funktion von I nach rep_P ist, so dass für alle $i \in I$ gilt: i ist eine Instanz der Aktivitätsdefinition $\delta_P(\phi(i))$.
  - ϕ ist linkstotal und linkseindeutig, da nach jedem Einfügen einer Aktivitätsinstanz i in I genau ein Paar $(i, .)$ in ϕ eingefügt wird.
  - ϕ ist rechtstotal und rechtseindeutig, da für jeden Aktivitätsrepräsentanten $r \in rep_P$ die Schleife genau einmal durchlaufen wird und bei jedem Durchlauf genau ein Paar $(., r)$ in ϕ eingefügt wird.
  - Da ausschließlich Paare (i, r) in ϕ eingefügt werden, bei denen die Aktivitätsinstanz i durch die Prozedur *instanziiereAktivität($\delta_P(r)$, i)* erzeugt wurde und $r = \phi(i)$ ist, gilt nach Lemma 1 für alle Paare in ϕ, dass i eine Instanz der Aktivitätsdefinition $\delta_P(\phi(i))$ ist.
- Ausführungskonsistenz: Die Ausführungskonsistenz ist gegeben, da beide Bedingungen für die Ausführungskonsistenz nur verletzt werden können, wenn zumindest eine Aktivitätsinstanz $i \in I$ existiert, für die gilt $exec_i \in$ {laufend, beendet}. Dies ist nicht möglich, da bei der Instanziierung nur Aktivitätsinstanzen im Ausführungszustand *aktiviert* erzeugt werden.

□

Im letzten Schritt der Prozedur 1 *bearbeiteGeschäftsfall* wird die Ausführung der Prozessinstanz durch die Unterprozedur 6 *starteInstanzen* angestoßen. Diese Prozedur startet die Aktivitätsinstanzen der Prozessinstanz, die entsprechend der Prozessdefinition momentan ausgeführt werden können.

Beispiel 9 Betrachten wir die Prozessinstanz I, die Prozessdefinition P und die Funktion ϕ aus Beispiel 8. Bei der Ausführung der Prozedur 6 *starteInstanzen(P,*

Prozedur 6 starteInstanzen(P, I, ϕ)

Eingabe: Prozessdefinition P, Prozessinstanz I, Funktion ϕ

Variablen: Boolean *kannStarten*

for all $i \in I$ **do**
  if *exec*$_i = aktiviert$ **then**
    kannStarten := true
    for all $j \in I$ mit $(\phi(j), \phi(i)) \in con_P$ **do**
      if *exec*$_j \neq beendet$ **then**
        kannStarten := false
      end if
    end for
    if *kannStarten* = true **then**
      starte Aktivitätsinstanz i
    end if
  end if
end for

I, ϕ*)* wird die äußere Schleife für jede Aktivitätsinstanz in I durchlaufen. Die Bedingung *exec*$_i = aktiviert$ ist dabei für alle Aktivitätsinstanzen erfüllt. Betrachten wir die innere Schleife beim Durchlauf mit der Aktivitätsinstanz i_1. Da i_1 in Bezug auf die Prozessdefinition P keinen direkten Vorgänger hat, wird die innere Schleife nicht durchlaufen und die Aktivitätsinstanz i_1 gestartet. Betrachten wir nun die innere Schleife beim Durchlauf mit der Aktivitätsinstanz i_2. Die Aktivitätsinstanz i_2 hat in Bezug auf P die Aktivitätsinstanz i_1 als direkten Vorgänger und die innere Schleife wird somit einmal durchlaufen. Da i_1 nicht beendet ist, wird die Variable *kannStarten* auf *false* gesetzt und folglich i_2 nicht gestartet. Ebenso werden die Aktivitätsinstanzen i_3 und i_4 nicht gestartet, da auch sie direkte Vorgänger in Bezug auf P haben, die noch nicht beendet sind. ◇

Die Prozedur 6 startet alle Aktivitätsinstanzen im Ausführungszustand *aktiviert*, die keinen Vorgänger in einem anderen Ausführungszustand als *beendet* haben. Die weitere Ausführung der Prozessinstanz erfolgt reaktiv auf Basis von *Ereignis-Bedingungs-Aktions-Regeln* [25]:

Fortsetzungsregel
Ereignis: Eine Aktivitätsinstanz endet.
Bedingung: Der Eintritt des Geschehnisses hat das erwartete Resultat und die Prozessinstanz ist nicht beendet.
Aktion: Führe Prozedur *starteInstanzen* aus.

Beendigungsregel
Ereignis: Eine Aktivitätsinstanz endet.
Bedingung: Der Eintritt des Geschehnisses hat das erwartete Resultat und die Prozessinstanz ist beendet.
Aktion: Beende Ausführung des Geschäftsfalls.

Durch die Fortsetzungsregel werden jeweils bei dem Eintritt eines Geschehnisses die Nachfolger der beendeten Aktivitätsinstanzen gestartet, falls diese *aktiviert* sind und keinen Vorgänger mehr in einem anderen Zustand als *beendet* haben. Sobald alle Aktivitätsinstanzen ausgeführt sind, beendet die Beendigungsregel die Ausführung des Geschäftsfalls, da davon ausgegangen wird, dass der Zustand des Geschäftsfalls nun ein Zielzustand ist. Beide Regeln haben die Bedingung, dass der Eintritt des Geschehnisses das erwartete Resultat hat. Eine dritte Regel, die den Fall abfängt, dass der Eintritt des Geschehnisses nicht das erwartete Resultat hat, wird in Abschnitt 5.3.1 im Zusammenhang mit der Auslösung der Umplanung vorgestellt.

Die Prozedur 3 *planeProzess* liefert eine gültige Prozessdefinition P. Nach Definition 11 bedeutet dies, dass eine beendete Prozessinstanz, die Instanz von P ist, als finalen Zustand einen Zielzustand hat. In Lemma 2 haben wir gezeigt, dass die Prozedur 5 *instanziiereProzess* eine Prozessinstanz I liefert, die Instanz der Prozessdefinition P ist. Demnach ist noch zu zeigen, dass sich dies nicht ändert, bis die Prozessinstanz beendet ist:

Lemma 3 Ist die Prozessinstanz I eine ϕ-Instanz der Prozessdefinition P, bleibt sie dies auch bei ihrer Ausführung, falls alle Geschehnisse das erwartete Resultat haben.

Beweis Wir betrachten die Definition 8 und zeigen, dass die Vollständigkeit und Ausführungskonsistenz durch die Ausführung nicht verletzt werden können. Dabei müssen jeweils die beiden Ereignisse betrachtet werden, die die Prozessinstanz I verändern: a.) die Ausführung der Prozedur *starteInstanzen* und b.) die Beendigung einer Aktivitätsinstanz $i \in I$.

- *Vollständigkeit:*
  a.) Die Ausführung der Prozedur *starteInstanzen(P, I, ϕ)* kann die Vollständigkeit offensichtlich nicht verletzen, da sich bezüglich I lediglich der Ausführungszustand von Aktivitätsinstanzen aus I ändern kann.

  b.) Das gleiche gilt für die Beendigung einer Aktivitätsinstanz $i \in I$.

- *Ausführungskonsistenz:*
  a.) In Bezug auf die Prozedur *starteInstanzen* ist zu prüfen, ob es bei der Ausführung zu dem Fall kommen kann, dass eine Aktivitätsinstanz $i \in I$ vor einer Aktivitätsinstanz $j \in I$ gestartet wird, wobei $(\phi(j), \phi(i)) \in con_P$. Dies

ist nicht möglich, da in der Prozedur vor dem Start einer jeden Aktivitätsinstanz i geprüft wird, ob alle Aktivitätsinstanzen $j \in I$ mit $(\phi(j), \phi(i)) \in$ *con*$_P$ beendet sind.

b.) Für den Fall der Beendigung einer Aktivitätsinstanz untersuchen wir die beiden Bedingungen für die Ausführungskonsistenz: 1.) Für alle $i, j \in I$ mit ($\phi(i)$, $\phi(j)$) $\in$ *con*$_P$ gilt: *exec*$_i \notin$ {beendet} $\Rightarrow$ *exec*$_j \notin$ {laufend, beendet}. Diese Bedingung kann nicht verletzt werden, da Aktivitätsinstanzen ausschließlich durch die Prozedur *starteInstanzen* gestartet werden und eine Aktivitätsinstanz $j \in I$ durch *starteInstanzen* nicht gestartet wird, falls ein direkter Vorgänger noch nicht beendet ist. Somit kann diese auch nicht vor einem direkten Vorgänger enden. 2.) Für alle $i, j \in I$ mit ($\phi(i)$, $\phi(j)$) $\in$ *con*$_P$ gilt: *exec*$_i \in$ {beendet} $\wedge$ *exec*$_j \in$ {laufend, beendet} $\Rightarrow t_{end}(i) \leq t_{start}(j)$. Diese Bedingung kann nicht verletzt werden, weil j niemals vor i enden kann, da analog zur obigen Argumentation Aktivitätsinstanzen ausschließlich durch die Prozedur *starteInstanzen* gestartet werden und j von *starteInstanzen* nicht gestartet wird, bevor i beendet ist.

□

Im Folgenden wird gezeigt, dass jede Ausführung einer Prozessinstanz in endlicher Zeit beendet ist. Dies geschieht in zwei Schritten. Zunächst wird in Lemma 4 für eine konkrete Aktivitätsinstanz bewiesen, dass sie in endlicher Zeit endet, falls alle direkten Vorgänger in Bezug auf die Prozessdefinition in endlicher Zeit enden. Darauf aufbauend zeigt Theorem 5, dass alle Aktivitätsinstanzen in endlicher Zeit beendet sind.

Lemma 4 Gegeben sei eine Prozessinstanz I, die ϕ-Instanz einer Prozessdefinition P ist und eine Aktivitätsinstanz $i \in I$, die sobald gestartet in endlicher Zeit endet. Die Prozedur *starteInstanzen(*P, I, ϕ*)* sei ausgeführt. Für alle $j \in I$ mit $(\phi(j), \phi(i)) \in$ *con*$_P$ gelte, dass j in endlicher Zeit endet. Der Eintritt eines Geschehnisses habe immer das erwartete Resultat. Dann folgt, dass i in endlicher Zeit endet.

Beweis Bezeichne J die Menge $\{j \in I : (\phi(j), \phi(i)) \in$ *con*$_P\}$. Fallunterscheidung:

1. Falls $J = \emptyset$, dann wird i bei der Ausführung der Prozedur *starteInstanzen* gestartet und endet nach der Annahme in endlicher Zeit.

2. Falls $J \neq \emptyset$, dann gibt es, da alle Aktivitätsinstanzen in J in endlicher Zeit enden, einen Zeitpunkt t, an dem eine Aktivitätsinstanz j aus J endet und keine andere Aktivitätsinstanz aus J mehr in einem anderen Ausführungszustand als *beendet* ist. Da das Geschehnis das erwartete Resultat hat und Aktivitätsinstanz i und damit auch Prozessinstanz I noch nicht beendet ist,

wird die Aktion der Fortsetzungsregel ausgeführt: die Prozedur 6 *starteInstanzen*. Da jetzt keine nicht beendete Aktivitätsinstanz $j \in J$ mehr existiert, wird i gestartet und endet nach der Annahme in endlicher Zeit.

□

Theorem 5 Gegeben sei eine Prozessinstanz I, die ϕ-Instanz einer Prozessdefinition P ist. Die Prozedur *starteInstanzen(P, I, ϕ)* sei ausgeführt. Der Eintritt eines Geschehnisses habe immer das erwartete Resultat. Alle Aktivitätsinstanzen, die gestartet werden, enden in endlicher Zeit. Dann folgt, dass I in endlicher Zeit beendet ist.

Beweis Nach Definition 8 ist eine Prozessinstanz genau dann beendet, wenn alle Aktivitätsinstanzen beendet sind. Daher ist zu zeigen: $\forall\ i \in I : i$ endet in endlicher Zeit. Beweis durch Widerspruch: Entsprechend ist die Annahme die Negation der obigen Aussage: $\exists\ i_1 \in I : i_1$ endet nicht in endlicher Zeit. Nach Lemma 4 muss dann eine Aktivitätsinstanz $i_2 \in I$ existieren, für die gilt $(\phi(i_2), \phi(i_1)) \in \mathit{con}_P$ und die nicht in endlicher Zeit endet, da ansonsten i_1 in endlicher Zeit enden würde. Entsprechend muss es eine Aktivitätsinstanz $i_3 \in I$ existieren, für die gilt $(\phi(i_3), \phi(i_2)) \in \mathit{con}_P$ und die nicht in endlicher Zeit endet und so weiter. Zusammenfassend muss es eine Folge $F = (i_1, i_2, i_3, ...)$ von Aktivitätsinstanzen aus I geben, die nicht in endlicher Zeit enden. Dabei gilt jeweils, dass eine Aktivitätsinstanz $i_{n+1} \in I$ existieren muss, für die gilt $(\phi(i_{n+1}), \phi(i_n)) \in \mathit{con}_P$ und die nicht in endlicher Zeit endet, da ansonsten i_n in endlicher Zeit enden würde. Die Folge F kann nicht endlich sein, da ansonsten das letzte Element aus F nach Lemma 4 in endlicher Zeit enden würde. Aussage A sei: $\forall\ i_x, i_y \in F : x < y \Rightarrow i_x \neq i_y$. Fallunterscheidung:

- Aussage A gilt. Daraus folgt, dass I unendlich ist. Da mit ϕ eine totale, bijektive Funktion von I nach rep_P existiert, ist damit auch rep_P unendlich. Dies steht im Widerspruch zu der Aussage, dass P eine Prozessdefinition ist.

- Aussage A gilt nicht. Daraus folgt: $\exists\ i_x, i_y \in F : x < y \wedge i_x = i_y$. Da ϕ eine totale, bijektive Funktion von I nach rep_P ist, existiert demnach auch eine Folge W von Aktivitätsrepräsentanten aus rep_P mit $W = (\phi(i_y), ..., \phi(i_x))$, so dass:

  - $\forall\ k \in \{x, ..., y-1\} : (\phi(i_{k+1}), \phi(i_k)) \in \mathit{con}_P$, das heißt W einen gerichteten Weg in G_P darstellt.
  - $\phi(i_y) = \phi(i_x)$, das heißt Start- und Endknoten von W identisch sind.

  Somit ist G_P zyklisch. Dies steht im Widerspruch zu der Aussage, dass P eine Prozessdefinition ist.

□

Zusammenfassend ist entscheidend, dass der Graph einer Prozessdefinition endlich und azyklisch ist. Ein Zyklus in dem Graph würde dazu führen, das die Aktivitätsinstanzen die dem Zyklus zuzuordnen sind, bei der Ausführung gegenseitig auf ihre Beendigung warten. Somit würde die Prozessinstanz nicht in endlicher Zeit enden. Aufbauend auf den obigen Sätzen betrachten wir nun die gesamte Bearbeitung eines Geschäftsfalls durch die Prozedur 1 *bearbeiteGeschäftsfall* und zeigen, dass diese den Zustand des Geschäftsfalls in einen Zielzustand überführt.

Theorem 6 Existiert zu dem Planungsproblem T zum Geschäftsfall c eine gültige Prozessdefinition und haben alle Geschehnisse das erwartete Resultat, dann überführt die Ausführung der Prozedur 1 *bearbeiteGeschäftsfall(c)* den Zustand des Geschäftsfalls in endlicher Zeit in einen Zielzustand.

Beweis Da zu dem Planungsproblem T zum Geschäftsfall c eine gültige Prozessdefinition existiert, liefert die Prozedur *planeProzess(T, P, gefunden)* eine gültige Prozessdefinition P zum Planungsproblem T. Nach Lemma 2 erzeugt dann die Prozedur *instantiiereProzess(*P, I, ϕ*)* zur Prozessdefinition P eine Prozessinstanz I und eine Funktion ϕ, so dass I eine ϕ-Instanz von P ist. Anschließend wird die Prozedur *starteInstanzen(*P, I, ϕ*)* ausgeführt. Folglich ist nach Theorem 5 die Prozessinstanz I in endlicher Zeit beendet. Nach Lemma 3 ist die beendete Prozessinstanz I noch immer eine ϕ-Instanz von P. Da P gültig ist und I eine Instanz von P ist und alle Geschehnisse das erwartete Resultat haben, ist nach Definition 11 auch die beendete Prozessinstanz I gültig. Damit besitzt I nach Definition 10 einen finalen Zustand der ein Zielzustand von T und damit auch des Geschäftsfalls c ist. □

5.3 Automatisierte Prozessanpassung

Dieser Abschnitt erweitert das formale Modell der integrierten Prozessplanung und -ausführung, so dass eine dynamische Anpassung der Ausführung durch Umplanung der Prozessdefinition zur Laufzeit möglich wird. Hierbei wird der in Abschnitt 4.4.3 vorgestellte Lösungsansatz der *Domänenerweiterung um Stellvertreteraktivitäten* umgesetzt. Die Gliederung entspricht den in Abschnitt 4.2 vorgestellten, grundlegenden Schritten: Überwachung, Umplanung und Fortsetzung. Zur Überwachung der Prozessausführung wird das Modell in Abschnitt 5.3.1 um eine weitere Ereignis-Bedingungs-Aktions-Regel zu Auslösung der Umplanung erweitert. Die Umplanung unter Berücksichtigung von Stellvertreteraktivitäten für laufende Aktivitätsinstanzen wird in Abschnitt 5.3.2 beschrieben. Abschnitt 5.3.3 erläutert die Fortsetzung der Ausführung basierend auf einer neuen Prozessinstanz.

5.3.1 Überwachung

Die Grundidee bei der Modellierung der Prozessüberwachung ist, diese ebenso wie die oben vorgestellte Prozessausführung in Form von Ereignis-Bedingungs-Aktions-Regeln darzustellen. Somit erfolgt sowohl die Prozessausführung als auch

die Prozessüberwachung durch Reaktion auf Ereignisse. In Abschnitt 4.2 wurden bereits verschiedene Ereignisse diskutiert, die eine Umplanung erforderlich machen. Bei dem formalen Modell werden wir uns zunächst auf den Fall beschränken, dass ein Geschehnis nicht das erwartete Resultat hat. Später wird das Modell dann um Regeln erweitert, die auch Fälle wie die Änderung von Zustand oder Ziel eines Geschäftsfalls oder der Domäne berücksichtigen. Damit auf ein Geschehnis reagiert werden kann, das nicht das erwartete Resultat hat, wird das Modell um folgende Regel erweitert:

Umplanungsregel U_1
Ereignis: Eine Aktivitätsinstanz endet.
Bedingung: Der Eintritt des Geschehnisses hat nicht das erwartete Resultat.
Aktion: Führe Prozedur *anpassenAusführung* aus.

Bei jeder Beendigung einer Aktivitätsinstanz wird somit die Bedingung sowohl der Umplanungsregel als auch der in Abschnitt 5.2 eingeführten Fortsetzungsregel und Beendigungsregel geprüft. Falls das Geschehnis nicht das erwartete Resultat hat, ist nur die Bedingung der Umplanungsregel erfüllt. In diesem Fall ist eine Umplanung notwendig, da die Ausführbarkeit für die Prozessinstanz nach Definition 10 nicht mehr sicher gestellt ist. Daher kann auch nicht mehr gewährleistet werden, dass eine weitere Ausführung auf Basis der derzeitigen Prozessdefinition den Zustand der Geschäftsfalls in einen Zielzustand überführt. Als Reaktion auf die Gefährdung der Zielerreichung wird daher die Prozedur *anpassenAusführung* aufgerufen, die im folgenden Abschnitt spezifiziert wird.

Beispiel 10 Zur Illustration der von der Umplanungsregel abgedeckten unerwarteten Resultate eines Geschehnisses betrachten wir die in Ausführung befindliche Prozessinstanz I einer Angebotsanfrage aus Beispiel 6 auf Seite 64. Die Endzeitsequenz enthält die Endzeiten der beiden beendeten Aktivitätsinstanzen: t_0 = 2 und t_1 = 6. Das Geschehnis H_0 zum Zeitpunkt $t_0 = 2$ habe wie in Beispiel 6 das erwartete Resultat. Somit überführt das Geschehnis H_0 den Zustand $S_0 = (\{A_{anfrage}\}, 0)$ in den Folgezustand $S_1 = (\{A_{anfrage}, A_{erfasst}\}, 2)$. Das erwartete Resultat des Geschehnisses $H_1 = \{i_2\}$ zum Zeitpunkt t_1 = 6 ist dann der Zustand $S_2 = (\{A_{anfrage}, A_{erfasst}, A_{exp\_deu}\}, 6)$. Angenommen H_1 hat nicht das erwartete Resultat und für den Folgezustand S_2 gilt $S_2 = (\{A_{anfrage}, A_{erfasst}, A_{exp\_eng}\}, 6)$. Das heißt, die Bestimmung der Exportbestimmungen hat statt einem deutschsprachigen ein englischsprachiges Dokument ergeben. Somit kann die momentane Prozessinstanz nicht weiter ausgeführt werden, da die Aktivität $d_{antworten}$, wie in Beispiel 2 spezifiziert, die Existenz eines deutschsprachigen Dokuments als Vorbedingung hat. ◇

5.3.2 Umplanung

In diesem Abschnitt wird beschrieben, wie eine neue Prozessdefinition auf Basis des aktuellen Zustands und Ziels des Geschäftsfalls geplant wird. Wie in Abschnitt 4.4.3 beschrieben, werden zum Umplanungszeitpunkt laufende Aktivitätsinstanzen in Form von stellvertretenden Aktivitätsdefinitionen bei der Umplanung berücksichtigt. Hierzu wird ein Umplanungsproblem definiert, das eine Erweiterung eines Planungsproblems aus Definition 5 darstellt.

Definition 12 Umplanungsproblem Ein *Umplanungsproblem* N ist ein Paar N = (*aufgabe*$_N$, *proxy*$_N$), wobei

- *aufgabe*$_N$ ein Planungsproblem und
- *proxy*$_N$ eine Menge von Aktivitätsdefinitionen ist.

◇

Die Prozedur 7 *anpassenAusführung* wird ausgeführt, sobald eine Geschehnis nicht das erwartete Resultat hat. Sie beschreibt sämtliche Schritte zur Anpassung der Ausführung mittels einer Umplanung der Prozessdefinition. Zunächst werden alle

Prozedur 7 anpassenAusführung(c, I, P, ϕ)

Eingabe: Geschäftsfall c, Prozessinstanz I, Prozessdefinition P, Funktion ϕ

Variablen:
Umplanungsproblem N
Prozessdefinition P'
Prozessinstanz I'
Funktion ψ
Funktion ϕ'
Boolean *gefunden*

deaktiviereProzessinstanz(I)
erzeugePlanungsproblem(c, *aufgabe*$_N$)
erzeugeStellvertreter(I, P, ϕ, *proxy*$_N$, ψ)
planeProzessNeu(N, P', *gefunden*)
if *gefunden* **then**
  reinstanziiereProzess(P', ψ, I', ϕ')
  starteInstanzen(P', I', ϕ')
else
  manueller Eingriff
end if

Aktivitätsinstanzen, die sich noch im Zustand *aktiviert* befinden, deaktiviert. Dies geschieht in Prozedur 8 *deaktiviereProzessinstanz*. Wie bei der initialen Planung

Prozedur 8 deaktiviereProzessinstanz(I)

Eingabe: Prozessinstanz I

for all $i \in I$ **do**
  if *exec*$_i = aktiviert$ **then**
    exec$_i := deaktiviert$
  end if
end for

wird anschließend durch die Prozedur 2 *erzeugePlanungsproblem* ein Planungsproblem basierend auf dem aktuellen Zustand des Geschäftsfalls erzeugt. Dieses Planungsproblem bildet dann einen Teil des Umplanungsproblems. Anschließend werden in der Prozedur 9 *erzeugeStellvertreter* für die noch laufenden Aktivitätsinstanzen stellvertretende Aktivitätsdefinitionen erzeugt und ihnen zugeordnet. Die in der Prozedur erzeugten stellvertretenden Aktivitätsdefinitionen haben die gleichen Vorbedingungen und Effekte wie die ursprünglichen Aktivitätsdefinitionen, weil die Vorbedingungen für die gesamte Dauer der Aktivitätsinstanzen gelten müssen und alle Effekte am Endzeitpunkt der Aktivitätsinstanzen erwartet werden. Hingegen muss die erwartete Ausführungsdauer neu berechnet werden, da Ausführung bereits begonnen hat. Diese erwartete Rest-Ausführungsdauer ist wichtig, damit der Planer eine für die Situation optimale Prozessdefinition finden kann. Wesentlich ist hierbei, dass die erwartete Ausführungsdauer zwar eine Bedeutung für die Optimierung hat, aber die Gültigkeit einer Prozessdefinition nicht von der Einhaltung der erwarteten Ausführungsdauer abhängig ist. Somit muss eine stellvertretende Aktivitätsdefinition auch für eine Aktivitätsinstanz gebildet werden, die voraussichtlich beim Ende des Umplanungsvorgangs bereits beendet ist. Diese besitzt dann eine erwartete Ausführungsdauer von null Zeiteinheiten.

Beispiel 11 Führen wir hierzu das Beispiel 10 fort. Mit der Beendigung der Aktivitätsinstanz i_2 ist das Ereignis für die Umplanungsregel eingetreten. Die Prüfung der Bedingung fällt positiv aus, da das entsprechende Geschehnis H_1 nicht das erwartete Resultat hat. Somit wird als Aktion der Umplanungsregel die Prozedur 7 *anpassenAusführung* ausgeführt: Zunächst erfolgt die Deaktivierung aller aktivierten Aktivitätsinstanzen durch die Prozedur 8 *deaktiviereProzessinstanz*. Hierdurch ändert sich der Ausführungszustand der Aktivitätsinstanz i_4 von *aktiviert* auf *deaktiviert*. In den folgenden zwei Schritten wird ein Umplanungsproblem N erzeugt, das sich nach Definition 12 aus einem Planungsproblem *aufgabe*$_N$ und einer Menge stellvertretender Aktivitätsdefinitionen *proxy*$_N$ zusammensetzt. Im ersten Schritt erzeugt die Prozedur 2 *erzeugePlanungsproblem* ein Planungsproblem *aufgabe*$_N$ basierend auf dem aktuellen Zustand $S_2 = (\{A_{anfrage}, A_{erfasst}, A_{exp\_eng}\}, 6)$ des Geschäftsfalls. Für das Planungsproblem gilt dann:

$$\textit{aufgabe}_N = (D, (\{A_{anfrage}, A_{erfasst}, A_{exp\_eng}\}, 6), \{A_{beantwortet}\})$$

Prozedur 9 erzeugeStellvertreter(I, P, ϕ, Q, ψ)

Eingabe: Prozessinstanz I, Prozessdefinition P, Funktion ϕ
Ausgabe: Menge von Aktivitätsdefinitionen Q, Funktion ψ
Variablen:
Aktivitätsdefinition d, d^*
Aktivitätsinstanz i
Zeit *elapsed*

$Q := \emptyset$
$\psi := \emptyset$
for all $i \in I$ **do**
  if $exec_i =$ laufend **then**
    $d := \delta_P(\phi(i))$
    erzeuge neue Aktivitätsdefinition d^*
    $pre_{d^*} := pre_d$
    $eff^-_{d^*} := eff^-_d$
    $eff^+_{d^*} := eff^+_d$
    $elapsed := (t_{jetzt} - t_{start}(i)) + dur_{UMPLANUNG}$
    if $dur_d > elapsed$ **then**
      $dur_{d^*} := dur_d - elapsed$
    else
      $dur_{d^*} := 0$
    end if
    füge d^* zu Q hinzu
    füge tupel (d^*, i) zu ψ hinzu
  end if
end for

Im zweiten Schritt erzeugt die Prozedur 9 *erzeugeStellvertreter* für die noch laufende Aktivitätsinstanz i_3 eine stellvertretende Aktivitätsdefinition d^* und ordnet diese über die Funktion ψ der Aktivitätsinstanz i_3 zu. Die stellvertretende Aktivitätsdefinition d^* wird auf Basis der Aktivitätsdefinition $d_{kalkulieren}$ erzeugt, da i_3 eine Instanz von $d_{kalkulieren}$ ist. Hierbei werden die Effekte und Vorbedingungen übernommen. Lediglich die erwartete Ausführungsdauer wird neu berechnet: Zunächst wird die Zeitdauer *elapsed* bestimmt, die sich die Aktivitätsinstanz i_3 voraussichtlich beim Ende des Umplanungsvorgangs in Ausführung befindet. Hierbei wird zunächst die momentan verstrichene Zeit als Differenz von $t_{jetzt} = 6$ und $t_{start}(i_3) = 2$ berechnet. Die Aktivitätsinstanz i_3 befindet sich also momentan seit 4 Zeiteinheiten in Ausführung. Hierzu wird die Dauer $dur_{UMPLANUNG} = 1$ für die Generierung einer neuen Prozessdefinition addiert. Folglich sind seit dem Start der Aktivitätsinstanz i_3 beim Ende des Umplanungsvorgangs *elapsed* = 5 Zeiteinheiten verstrichen. Die erwartete Ausführungsdauer der Aktivitätsdefinition $d_{kalkulieren}$ beträgt 8 Zeiteinheiten. Demnach wird die Aktivitätsinstanz i_3 sich nach Fertigstellung der neuen Prozessdefinition noch $dur_d -$ *elapsed* = 3 Zeiteinheiten in Ausführung befinden, falls sie ausführungsdauer-treu ist. Entsprechend bekommt die stellvertretende Aktivitätsdefinition d^* eine erwartete Ausführungsdauer von 3 zugewiesen. Zusammenfassend gilt für die Ausgabe der Prozedur 9 *erzeugeStellvertreter*: $Q = \{d^*\}$ mit $d^* = (\{A_{erfasst}\}, \emptyset, \{A_{kalkuliert}\}, 3)$ und $\psi = (d^*, i_3)$. Die Menge Q bildet dann den zweiten Teil $proxy_N$ des Umplanungsproblems N = (*aufgabe*$_N$, $proxy_N$). Somit gilt für dieses Beispiel N = $(D, (\{A_{anfrage}, A_{erfasst}, A_{exp\_eng}\}, 6), \{A_{beantwortet}\})$, $\{d^*\})$. $\diamond$

Es folgt nun ein Lemma zur Prozedur *erzeugeStellvertreter*, das später benötigt wird, um die Korrektheit des gesamten Umplanungsvorgangs zu zeigen.

Lemma 7 Gegeben sei eine Prozessinstanz I, die eine ϕ-Instanz einer Prozessdefinition P ist. Bezeichne Ω die Menge aller Aktivitätsinstanzen aus I, die sich im Ausführungszustand *laufend* befinden. Dann liefert die Ausführung der Prozedur *erzeugeStellvertreter(*I, P, ϕ, Q, ψ*)* eine Menge von Aktivitätsdefinitionen Q und eine totale, bijektive Funktion ψ von Q nach Ω, wobei für jede Aktivitätsdefinition $d \in Q$ gilt, dass die Aktivitätsinstanz $\psi(d)$ eine Instanz von d ist.

Beweis Der Beweis erfolgt in drei Schritten:

- ψ ist linkstotal und linkseindeutig, da nach jedem Einfügen einer Aktivitätsdefinition d^* in Q genau ein Paar $(d^*, .)$ in ψ eingefügt wird.
- ψ ist rechtstotal und rechtseindeutig, da für jede Aktivitätsinstanz $i \in I$ im Ausführungszustand *laufend* die Schleife genau einmal durchlaufen wird und bei jedem Durchlauf genau ein Paar $(., i)$ in ψ eingefügt wird.
- Betrachten wir für jeden Schleifendurchlauf die Beziehungen zwischen i, d und d^*. Da laut Annahme die Prozessinstanz I eine ϕ-Instanz der Prozessdefinition P ist, ist nach der in Definition 8 geforderten Vollständigkeit die

Aktivitätsinstanz i eine Instanz der Aktivitätsdefinition $d = \delta_P(\phi(i))$. Da d^* jeweils die gleichen Vorbedingungen und Effekte hat wie d, muss i entsprechend Definition 7 auch eine Instanz der Aktivitätsdefinition d^* sein. Somit gilt für jedes Paar (d^*, i), das in ψ eingefügt wird, dass i eine Instanz der Aktivitätsdefinition d^* ist.

□

Als nächstes werden Eigenschaften für eine Prozessdefinition in Bezug auf ein Umplanungsproblem definiert. Darauf basierend erfolgt anschließend eine deskriptive Spezifikation der Prozedur 10 *planeProzessNeu*, die zu einem gegebenen Umplanungsproblem eine Prozessdefinition generiert.

Definition 13 Gültigkeit und Optimalität einer Prozessdefinition für ein Umplanungsproblem Eine Prozessdefinition P = (rep_P, con_P, δ_P) für ein Umplanungsproblem N = ($aufgabe_N$, $proxy_N$) ist *gültig*, falls

- P gültig für das Planungsproblem $aufgabe_N$ ist und
- bezüglich der Menge $proxy_N$ folgendes gilt:
  1. die Bildmenge von δ_P ist eine Obermenge von $proxy_N$
  2. δ_P ist linkseindeutig für die Teilmenge $proxy_N$ der Bildmenge
  3. $\forall\, r \in rep_P : \delta_P(r) \in proxy_N \Rightarrow \nexists\, (., r) \in con_P$

P ist *optimal* genau dann, wenn P gültig ist und es keine andere gültige Prozessdefinition mit einer kürzeren minimalen Ausführungsdauer gibt.

◇

Prozedur 10 planeProzessNeu(N, P, *gefunden*)

Eingabe: Umplanungsproblem N
Ausgabe: Prozessdefinition P, Boolean *gefunden*

Falls gültige Prozessdefinitionen für das Umplanungsproblem N existieren, wird eine gültige Prozessdefinitionen P und *gefunden* = true zurückgeliefert. Ansonsten ist P undefiniert und *gefunden* = false.

Existiert keine gültige Prozessdefinition für das übergebene Umplanungsproblem, so ist das System nicht mehr in der Lage, selbständig den Geschäftsfall in einen Zielzustand zu überführen. Nach Prozedur 7 *anpassenAusführung* erfolgt in diesem Fall ein manueller Eingriff. Dies kann eine Änderung des Ziels, eine Spezifikation neuer Aktivitäten oder eine Entscheidung zum Prozessabbruch sein. Wird dagegen eine neue Prozessdefinition gefunden, folgt die Anpassung der Ausführung, die im folgenden Abschnitt 5.3.3 beschrieben wird.

Beispiel 12 In Anknüpfung an das Beispiel 11 betrachten wir das Umplanungsproblem $N = (\textit{aufgabe}_N, \textit{proxy}_N)$ mit
$\textit{aufgabe}_N = (D, (\{A_{anfrage}, A_{erfasst}, A_{exp\_eng}\}, 6), \{A_{beantwortet}\})$ und
$\textit{proxy}_N = \{d^*\}$.
Folgende Prozessdefinition ist nach Definition 13 für dieses Umplanungsproblem gültig:
$P' = ((\textit{rep}_{P'}, \textit{con}_{P'}), \delta_{P'})$, wobei
$\textit{rep}_{P'} = \{r_5, r_6, r_7\}$
$\textit{con}_{P'} = \{(r_5, r_7), (r_6, r_7)\}$
$\delta_{P'} = \{(r_5, d^*), (r_6, d_{uebersetzen}), (r_7, d_{antworten})\}$
Der entsprechende Graph $G_{P'}$ ist in Abbildung 5.2 dargestellt. Die Knoten stellen Aktivitätsrepräsentanten dar, wobei in Klammern für jeden Aktivitätsrepräsentanten r auch die Aktivitätsdefinition $d = \delta_P(r)$ angegeben ist. Betrachten wir entsprechend der Definition 13 die Gültigkeit von P' für das Umplanungsproblem N: Die Prozessdefinition P' ist offensichtlich gültig für das Planungsproblem $\textit{aufgabe}_N$, da alle beendeten Prozessinstanzen, die eine Instanz von P' sind und bei denen der Eintritt jedes Geschehnisses das erwartete Resultat hat, gültig sind. Für jede dieser Prozessinstanzen gilt, dass sie entweder die Spur $< S_0^*, S_1^*, S_2^*, S_3^* >$ hat, wobei
S_0^* = $(\{A_{anfrage}, A_{erfasst}, A_{exp\_eng}\}, 6)$
S_1^* = $(\{A_{anfrage}, A_{erfasst}, A_{exp\_deu}\}, t_6)$
S_2^* = $(\{A_{anfrage}, A_{erfasst}, A_{exp\_deu}, A_{kalkuliert}\}, t_5)$
S_3^* = $(\{A_{anfrage}, A_{erfasst}, A_{exp\_deu}, A_{kalkuliert}, A_{beantwortet}\}, t_7)$
oder die Spur $< S_0^{**}, S_1^{**}, S_2^{**}, S_3^{**} >$ hat, wobei
S_0^* = $(\{A_{anfrage}, A_{erfasst}, A_{exp\_eng}\}, 6)$
S_1^* = $(\{A_{anfrage}, A_{erfasst}, A_{exp\_eng}, A_{kalkuliert}\}, t_5)$
S_2^* = $(\{A_{anfrage}, A_{erfasst}, A_{exp\_deu}, A_{kalkuliert}\}, t_6)$
S_3^* = $(\{A_{anfrage}, A_{erfasst}, A_{exp\_deu}, A_{kalkuliert}, A_{beantwortet}\}, t_7)$
Hierbei kennzeichnet der Zeitpunkt t_5 (t_6, t_7) den Endzeitpunkt der Aktivitätsinstanz, die dem Aktivitätsrepräsentanten r_5 (r_6, r_7) zugeordnet ist. Welche der Spuren für eine bestimme Prozessinstanz zutrifft, hängt davon ab, ob die Instanz von d^* oder die Instanz von $d_{uebersetzen}$ zuerst endet. Bezüglich der Gültigkeit von P' für das Umplanungsproblem N ist des Weiteren festzustellen, dass die Kriterien bezüglich der Menge $\textit{proxy}_N$ aus Definition 13 ebenfalls erfüllt sind:

1. Die Bildmenge $\{d^*, d_{uebersetzen}, d_{antworten}\}$ von $\delta_{P'}$ ist Obermenge von $\textit{proxy}_N = \{d^*\}$.
2. $\delta_{P'}$ ist linkseindeutig für die Teilmenge $\textit{proxy}_N = \{d^*\}$ der Bildmenge.
3. Nur für das Element r_5 aus $\textit{rep}_{P'}$ gilt, dass $\delta_{P'}(r_5) \in \textit{proxy}_N$. Für dieses Element ist die Aussage $\nexists\ (., r_5) \in \textit{con}_{P'}$ wahr.

Die minimale Ausführungsdauer von P' beträgt 13 Zeiteinheiten. In diesem Fall gilt t_5 = 10, t_6 = 9, t_7 = 13. Anmerkung: Die minimale Ausführungsdauer der

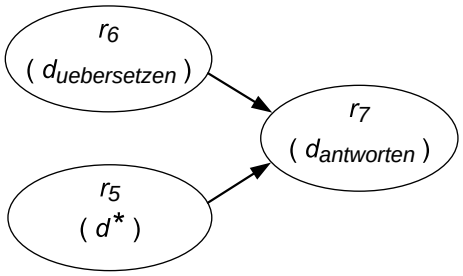


Abbildung 5.2: Graph der Prozessdefinition P'

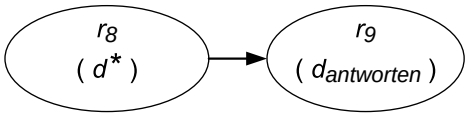


Abbildung 5.3: Graph der Prozessdefinition P''

Prozessdefinition P' ist nach Definition 11 als kürzeste Ausführungsdauer aller beendeten, ausführungsdauer-treuen Prozessinstanzen, die Instanz von P' sind, definiert.

Betrachten wir nun eine weitere Prozessdefinition P'', die für das Umplanungsproblem N gültig ist:
$P'' = ((rep_{P''}, con_{P''}), \delta_{P''})$, wobei
$rep_{P''} = \{r_8, r_9\}$
$con_{P''} = \{(r_8, r_9)\}$
$\delta_{P''} = \{(r_8, d^*), (r_9, d_{antworten2})\}$
Der entsprechende Graph $G_{P''}$ ist in Abbildung 5.3 dargestellt. Statt einer vorhergehenden Übersetzung erfolgt hier eine direkte Beantwortung der Angebotsanfrage auf Basis der Exportbestimmungen auf Englisch. Die minimale Ausführungsdauer von P'' beträgt 14 Zeiteinheiten. In diesem Fall endet die Aktivitätsinstanz, die dem Aktivitätsrepräsentanten r_8 zugeordnet ist, zum Zeitpunkt 10 und die Aktivitätsinstanz, die dem Aktivitätsrepräsentanten r_9 zugeordnet ist, endet zum Zeitpunkt 14. Damit ist P'' nicht optimal für das Umplanungsproblem N, da mit P' eine Prozessdefinition mit einer kürzeren minimalen Ausführungsdauer existiert. $\diamond$

In diesem Abschnitt wird beschrieben, wie die Ausführung an die neue Prozessdefinition angepasst und anschließend fortgesetzt wird. Die Situation ist die folgende: Alle aktivierten Aktivitätsinstanzen wurden vor der Umplanung deaktiviert. Für die laufenden Aktivitätsinstanzen wurde eine Menge $proxy_N$ von stellvertretenden Aktivitätsdefinition spezifiziert, die nun jeweils genau einen Aktivitätsrepräsentanten der neuen Prozessdefinition haben. Die Anpassung der Ausführung ist im Wesentlichen eine besondere Form der Instanziierung und wird in der Prozedur 11 *reinstanziiereProzess* beschrieben. Vergleichen wir diese Prozedur mit der

Prozedur 11 reinstanziiereProzess(P, ψ, I, ϕ)

Eingabe: Prozessdefinition P, Funktion ψ
Ausgabe: Prozessinstanz I, Funktion ϕ
Variablen: Aktivitätsinstanz i

$I := \emptyset$
$\phi := \emptyset$
for all $r \in rep_P$ **do**
  if $\delta_P(r)$ ist Element des Definitionsbereichs von ψ **then**
    $i := \psi(\delta_P(r))$
  else
    instanziiereAktivität($\delta_P(r)$, i)
  end if
  füge i zu I hinzu
  füge (i, r) zu ϕ hinzu
end for

initialen Instanziierung in Prozedur 5 *instanziiereProzess*: In beiden Prozeduren werden alle Elemente aus rep_P durchlaufen. Für den Fall, dass die Bedingung der IF-Anweisung bei einem Schleifen-Durchlauf zutrifft, findet bei beiden Prozeduren eine einfache Instanziierung statt. Einen Unterschied gibt es für den Fall, dass die Bedingung „$\delta_P(r)$ ist Element des Definitionsbereichs von ψ“ erfüllt ist. Hierbei werden diejenigen Aktivitätsrepräsentanten erfasst, die auf stellvertretende Aktivitätsdefinitionen verweisen. Für diese Aktivitätsrepräsentanten darf keine einfache Instanziierung stattfinden, da bereits gestartete Aktivitätsinstanzen existieren, für die eben diese stellvertretenden Aktivitätsdefinitionen erzeugt wurden. In diesem Fall findet anstelle einer Instanziierung nur eine Zuweisung der entsprechenden Aktivitätsinstanz statt.

Beispiel 13 Aufbauend auf Beispiel 12 betrachten wir nun die Anpassung der Ausführung. Die ersten vier Prozeduren *deaktiviereProzessinstanz*, *erzeugePlanungsproblem*, *erzeugeStellvertreter* und *planeProzessNeu* der Prozedur 7 *anpassenAusführung* sind ausgeführt. Es gilt
gefunden = true,
$P' = ((rep_{P'}, con_{P'}), \delta_{P'})$, wobei
$rep_{P'} = \{r_5, r_6, r_7\}$
$con_{P'} = \{(r_5, r_7), (r_6, r_7)\}$
$\delta_{P'} = \{(r_5, d^*), (r_6, d_{uebersetzen}), (r_7, d_{antworten})\}$ und
$\psi = (d^*, i_3)$.
Bei Ausführung der anschließenden Unterprozedur *reinstanziiereProzess* geschieht Folgendes: Für jeden der drei Aktivitätsrepräsentanten aus $rep_{P'}$ wird die Schleife durchlaufen und die Bedingung der IF-Anweisung geprüft. Für r_5 ist die Bedingung erfüllt, da $\delta_{P'}(r_5) = d^*$ Element des Definitionsbereichs $\{d^*\}$ von ψ ist. So-

mit wird die Aktivitätsinstanz $\psi(\delta_P(r)) = i_3$ der Menge I hinzugefügt und das Tupel (i_3, r_5) der Menge ϕ. Im Gegensatz zum Aktivitätsrepräsentanten r_5 ist die Bedingung der IF-Anweisung für die Aktivitätsrepräsentanten r_6 und r_7 nicht erfüllt und es wird jeweils eine neue Aktivitätsinstanz durch Ausführung der Prozedur 4 *instanziiereAktivität* erzeugt. Beim Durchlauf mit dem Aktivitätsrepräsentanten r_6 (r_7) wird von der Aktivitätsdefinition $\delta_P(r_6) = d_{uebersetzen}$ ($\delta_P(r_7) = d_{antworten}$) die Instanz i_5 (i_6) erzeugt. Somit werden die Aktivitätsinstanzen i_5 und i_6 der Menge I hinzugefügt und die Tupel (i_5, r_6) und (i_6, r_7) der Menge ϕ. Zusammenfassend gilt für die Ausgabe der Prozedur 4 *instanziiereAktivität*: $I = \{i_3, i_5, i_6\}$ und $\phi = \{(i_3, r_5), (i_5, r_6), (i_6, r_7)\}$. Es gilt I ist eine ϕ-Instanz der Prozessdefinition P'. Die hierfür nach Definition 8 geforderte Vollständigkeit und Ausführungskonsistenz sind gegeben:

- Vollständigkeit: ϕ ist offensichtlich eine totale, bijektive Funktion von I nach $rep_{P'} = \{r_5, r_6, r_7\}$. Des Weiteren gilt für alle $i \in I$, dass i eine Instanz der Aktivitätsdefinition $\delta_{P'}(\phi(i))$ ist: Die Aktivitätsinstanz i_5 (i_6) ist wie beschrieben direkt von $\delta_P(r_6) = d_{uebersetzen}$ ($\delta_P(r_7) = d_{antworten}$) instanziiert. Die Aktivitätsinstanz i_3 wurde ursprünglich von $d_{kalkulieren}$ instanziiert und ist somit nach Lemma 1 Instanz dieser Aktivitätsdefinition. Da $d_{kalkulieren}$ die gleichen Vorbedingungen und Effekte hat wie d^* ist i_3 auch Instanz von d^*.

- Ausführungskonsistenz: Nach Definition 8 ist müssen für alle $i, j \in I$ mit $(\phi(i), \phi(j)) \in con_P$ die beiden Bedingungen für die Ausführungskonsistenz gelten. In diesem Beispiel sind demnach die Fälle $i = i_3$, $j = i_6$ und $i = i_5$, $j = i_6$ zu prüfen. Da sich die Aktivitätsinstanz i_6 im Zustand *aktiviert* befindet, kann weder die Bedingung „$exec_i \notin$ {beendet} $\Rightarrow$ $exec_j \notin$ {laufend, beendet}“ noch die Bedingung „$exec_i \in$ {beendet} $\wedge$ $exec_j \in$ {laufend, beendet} $\Rightarrow t_{end}(i) \leq t_{start}(j)$“ verletzt sein.

$\diamond$

In dem Beispiel wurde gezeigt, wie durch den Ausführung der Prozedur 5 *instanziiereProzess* eine Instanz I erzeugt wird, die ϕ-Instanz der übergebenen Prozessdefinition ist. Das folgende Lemma zeigt diesen Zusammenhang in allgemeiner Form.

Lemma 8 Gegeben sei eine Prozessdefinition P, die für ein Umplanungsproblem N gültig ist. Des Weiteren sei gegeben eine Menge Ω von Aktivitätsinstanzen im Ausführungszustand *laufend* und eine totale, bijektive Funktion ψ von $proxy_N$ nach Ω, wobei für jede Aktivitätsdefinition $d \in proxy_N$ gilt, dass die Aktivitätsinstanz $\psi(d)$ eine Instanz von d ist. Dann liefert die Ausführung der Prozedur *reinstanziiereProzess(*P, ψ, I, ϕ*)* eine Prozessinstanz I und eine Funktion ϕ, so dass I eine ϕ-Instanz von P ist.

Beweis Es wird gezeigt, dass I eine ϕ-Instanz von P ist, indem wir analog zu Lemma 2 die in Definition 8 geforderte Vollständigkeit und Ausführungskonsistenz nachweisen:

- Vollständigkeit: Zum Nachweis der Vollständigkeit ist zu zeigen, dass ϕ eine totale, bijektive Funktion von I nach rep_P ist, so dass für alle $i \in I$ gilt: i ist eine Instanz der Aktivitätsdefinition $\delta_P(\phi(i))$.
  - ϕ ist linkstotal und linkseindeutig, da nach jedem Einfügen einer Aktivitätsinstanz i in I genau ein Paar (i, r) in ϕ eingefügt wird.
  - ϕ ist rechtstotal und rechtseindeutig, da für jeden Aktivitätsrepräsentanten $r \in \mathit{rep}_P$ die Schleife genau einmal durchlaufen wird und bei jedem Durchlauf genau ein Paar (i, r) in ϕ eingefügt wird.
  - Es bleibt zu zeigen, dass für alle Paare $(i, r) \in \phi$ gilt, dass die Aktivitätsinstanz i eine Instanz der Aktivitätsdefinition $\delta_P(r)$ ist. Hierzu unterteilen wir die Menge der Aktivitätsrepräsentanten rep_P in die Mengen $\mathit{rep}_P^{proxy} = \{r \in \mathit{rep}_P : \delta_P(r) \in \mathit{proxy}_N\}$ und $\mathit{rep}_P^{\neg proxy} = \mathit{rep}_P \setminus \mathit{rep}_P^{proxy}$. Fallunterscheidung:
    - Für alle $r \in \mathit{rep}_P^{\neg proxy}$ gilt analog zu Lemma 2, dass ausschließlich Paare (i, r) in ϕ eingefügt werden, bei denen die Aktivitätsinstanz i durch Ausführung der Prozedur *instanziiereAktivität($\delta_P(r)$, i)* erzeugt wurde. Folglich gilt nach Lemma 1 für alle Paare $(i, r) \in \phi$ mit $r \in \mathit{rep}_P^{\neg proxy}$, dass i eine Instanz der Aktivitätsdefinition $\delta_P(r)$ ist.
    - Für alle $r \in \mathit{rep}_P^{proxy}$ wird jeweils genau ein Paar (i, r) mit $i = \psi(\delta_P(r))$ in ϕ eingefügt. Laut Annahme gilt für jede Aktivitätsdefinition d aus dem Definitionsbereich proxy_N von ψ, dass die Aktivitätsinstanz $\psi(d)$ eine Instanz von d ist. Daraus folgt, dass nur Paare (i, r) in ϕ eingefügt werden, für die gilt, dass i eine Instanz der Aktivitätsdefinition $\delta_P(r)$ ist.

- Ausführungskonsistenz: Beweis durch Widerspruch: Angenommen die Ausführungskonsistenz ist nicht gegeben. Dann müssen nach Definition 8 zwei Aktivitätsinstanzen $i, j \in I$ mit $(\phi(i), \phi(j)) \in \mathit{con}_P$ existieren, für die zumindest eine der beiden Bedingungen für die Ausführungskonsistenz verletzt ist.
  - Fall 1: $j \in \Omega$. Betrachten wir den Aktivitätsrepräsentanten $r = \phi(j)$. Das Paar (j, r) wird durch die Prozedur *reinstanziiereProzess* in ϕ eingefügt. Aus $j \in \Omega$ folgt, dass j nicht neu instanziiert wurde. Es gilt demnach $j = \psi(\delta_P(r))$. Damit ist $\delta_P(r)$ Element des Definitionsbereichs proxy_N von ψ. Da laut Annahme P für N gültig ist, gilt nach Definition 13: $\forall\, r \in \mathit{rep}_P : \delta_P(r) \in \mathit{proxy}_N \Rightarrow \nexists\, (., r) \in \mathit{con}_P$. Dies steht im Widerspruch zur Aussage $(\phi(i), \phi(j)) \in \mathit{con}_P$.

– Fall 2: $j \notin \Omega$. Hieraus folgt, dass j durch die Prozedur *reinstanziiereProzess* neu instanziiert wurde. Damit befindet sich j im Ausführungszustand aktiviert. Somit kann offensichtlich keine der beiden Bedingungen für die Ausführungskonsistenz nach Definition 8 verletzt werden.

□

5.3.3 Fortsetzung

In diesem Abschnitt wird beschrieben, wie die Ausführung entsprechend der neuen Prozessinstanz fortgesetzt wird und analysiert, inwieweit hierdurch der Zustand des Geschäftsfalls in einen Zielzustand überführt wird.

Die Ausgangssituation bei der Fortsetzung der Ausführung ist die folgende: Durch die Prozedur 11 *reinstanziiereProzess* wurde eine neue Prozessinstanz erzeugt. Die laufenden Aktivitätsinstanzen der ursprünglichen Prozessinstanz wurden in die neue Prozessinstanz übernommen. Diese befinden sich weiterhin im Zustand *laufend* oder bereits im Zustand *beendet*. Diese Aktivitätsinstanzen müssen folglich nicht erneut gestartet werden. Des Weiteren kann es aber in der neuen Prozessinstanz zusätzliche Aktivitätsinstanzen geben, die entsprechend des Kontrollflusses nun gestartet werden müssen. Entsprechend wird als letzter Schritt der Prozedur 7 *anpassenAusführung* die neue Prozessinstanz, analog zur Prozedur 1 *bearbeiteGeschäftsfall*, durch die Prozedur 6 *starteInstanzen* gestartet. Die weitere Ausführung und Überwachung geschieht auf Basis der in den Abschnitten 5.2 und 5.3.1 beschriebenen Ereignis-Bedingungs-Aktions-Regeln.

Beispiel 14 Im Fall der Angebotsanfrage aus Beispiel 13 enthält die neue Prozessinstanz die laufende Aktivitätsinstanz i_3, die aus der ursprünglichen Prozessinstanz übernommen wurde. Die Aktivitätsinstanz i_5 hingegen wird in der Prozedur 11 *reinstanziiereProzess* von der Aktivitätsdefinition $d_{uebersetzen}$ instanziiert und befindet sich somit im Zustand *aktiviert*. Die Ausführung der Prozedur 6 *starteInstanzen* dient im diesem Fall dazu, Aktivitätsinstanz i_5 zu starten, damit sie nebenläufig zur Aktivitätsinstanz i_3 ausgführt wird. ◇

Im folgenden Theorem 9 wird gezeigt, dass nach Umplanung der Prozessdefinition und Anpassung der laufenden Prozessinstanz die Fortsetzung der Ausführung den Zustand eines Geschäftsfalls in einen Zielzustand überführt. Der entscheidende Schritt dies zu zeigen, wurde bereits in Lemma 8 geleistet. Der Beweis läuft analog zum Beweis des Theorems 6.

Theorem 9 Gegeben seinen ein Geschäftsfall c und eine Prozessinstanz I, die eine ϕ-Instanz der Prozessdefinition P ist. Existiert zu dem entsprechenden Umplanungsproblem N eine gültige Prozessdefinition und haben alle folgenden Geschehnisse das erwartete Resultat, dann überführt die Ausführung der Prozedur *anpassenAusführung(*c, I, P, ϕ*)* den Zustand des Geschäftsfalls in endlicher Zeit in einen Zielzustand.

Beweis Bezeichne Ω die Menge aller Aktivitätsinstanzen aus I, die sich im Ausführungszustand *laufend* befinden. Da laut Annahme die Prozessinstanz I eine ϕ-Instanz der Prozessdefinition P ist, liefert die Prozedur *erzeugeStellvertreter(*I, P, ϕ, Q, ψ*)* nach Lemma 7 eine Menge von Aktivitätsdefinitionen *proxy*$_N$ und eine totale, bijektive Funktion ψ von *proxy*$_N$ nach Ω, wobei für jede Aktivitätsdefinition $d \in$ *proxy*$_N$ gilt, dass die Aktivitätsinstanz $\psi(d)$ eine Instanz von d ist. Da laut Annahme zu dem Umplanungsproblem N eine gültige Prozessdefinition existiert, liefert die Prozedur *planeProzessNeu(N, P', gefunden)* eine gültige Prozessdefinition P' zum Umplanungsproblem N. Somit liefert die Prozedur *reinstanziiereProzess(*P', ψ, I', ϕ'*)* nach Lemma 8 eine Prozessinstanz I' und eine Funktion ϕ', so dass I' eine ϕ'-Instanz von P' ist. Die restliche Argumentation läuft analog zu Theorem 6: Die Prozedur *starteInstanzen(*P', I', ϕ'*)* wird ausgeführt. Folglich ist nach Theorem 5 die Prozessinstanz I' in endlicher Zeit beendet. Nach Lemma 3 ist die beendete Prozessinstanz I' noch immer eine ϕ'-Instanz von P'. Da P' gültig ist und I' eine Instanz von P' ist und alle Geschehnisse das erwartete Resultat haben, ist nach Definition 11 auch die beendete Prozessinstanz I' gültig. Damit besitzt I' nach Definition 10 einen finalen Zustand der ein Zielzustand von N und damit auch des Geschäftsfalls c ist. □

5.3.4 Erweiterte Prozessüberwachung

Das in Abschnitt 5.3.1 vorgestellte Konzept zur Prozessüberwachung basiert auf den grundlegenden Annahmen A11 und A12, die zu Beginn dieses Kapitels auf Seite 56 getroffen wurden:

- A11: Der Zustand eines Geschäftsfalls wird nur durch die Ausführung von Aktivitäten verändert.
- A12: Während der Bearbeitung eines Geschäftsfalls finden keine Änderungen des Ziels oder der Domäne statt.

Aufgrund dieser Annahmen wird nur eine einzige, bereits in Abschnitt 5.3.1 vorgestellte Umplanungsregel U_1 zur Auslösung der Umplanung benötigt, da keine Eingriffe von außen auf Zustand, Ziel und Domäne berücksichtigt werden müssen. In diesem Abschnitt wird das Konzept zur Prozessüberwachung für den Fall erweitert, dass die Annahmen A11 und A12 nicht gelten. Hierzu wird eine umfassendere Menge von Regeln aufgestellt, die zum Beispiel Regeln für Ereignisse wie die Veränderung des Ziels oder der Aktivitätsdefinitionen oder Effekte von Außen auf den Zustand des Geschäftsfalls enthält.

Wie auch die bereits beschriebene Umplanungsregel U_1, besitzen alle Regeln folgende Form:

- *Ereignis:* Jede Umplanungsregel basiert auf einem Ereignis, das potentiell die Optimalität beziehungsweise Gültigkeit der Prozessdefinition gefährdet.

- *Bedingung:* Die Bedingung schränkt die Fälle in denen das Ereignis die Optimalität beziehungsweise Gültigkeit der Prozessdefinition gefährdet möglichst eng ein, um eine unnötige Umplanung zu vermeiden.

- *Aktion:* Führe Prozedur *anpassenAusführung* aus.

Alle Regeln führen zur der Aktion, dass die Prozedur *anpassenAusführung* ausgeführt wird. Somit wird unabhängig von dem auslösenden Ereignis immer eine neue Prozessdefinition auf Basis der aktuellen Werte von Zustand, Ziel und Domäne generiert, falls die Bedingung der Regel erfüllt ist.

Bezüglich der relevanten Ereignisse und Bedingungen ist zwischen einer Gefährdung der Optimalität und einer Gefährdung der Gültigkeit der Prozessdefinition zu unterscheiden, da es Fälle gibt, in denen eine zuvor optimale Prozessdefinition aufgrund eines Ereignisses zwar weiterhin zum Ziel führt, das heißt gültig ist, aber potentiell nicht mehr den optimalen Weg zur Zielerreichung darstellt. Für diese Fälle muss entschieden werden, ob eine Umplanung ausgelöst werden soll, um gegebenenfalls eine bessere Prozessdefinition zu finden. Dagegen spricht der Aufwand für die erneute Planung und Prozessanpassung. Falls der Prozess auch manuell überwacht wird, wirkt sich eine Prozessanpassung zusätzlich negativ aus, da sich der Benutzer auf eine neue Prozessdefinition einstellen muss. Da potentielle Vor- und Nachteile abzuwägen sind, lässt sich keine generelle Aussage treffen, ob bei einer Gefährdung der Optimalität die Umplanung der Prozessdefinition auszulösen ist. Daher werden im Folgenden zwei Mengen von Regeln vorgestellt: Eine Menge von Regeln U_1 bis U_m, die eine Umplanung bereits auslösen, falls die Optimalität der Prozessdefinition gefährdet ist und eine Menge von Regeln U_1^* bis U_n^*, die eine Umplanung nur auslösen, falls die Gültigkeit der Prozessdefinition gefährdet ist.

Gefährdung der Optimalität einer Prozessdefinition

In diesem Abschnitt werden Regeln vorgestellt, die eine Umplanung auslösen, falls die Optimalität der Prozessdefinition gefährdet ist. Hierzu gehört die bereits in Abschnitt 5.3.1 vorgestellte Umplanungsregel U_1:

Umplanungsregel U_1
Ereignis: Eine Aktivitätsinstanz endet.
Bedingung: Der Eintritt des Geschehnisses hat nicht das erwartete Resultat.
Aktion: Führe Prozedur *anpassenAusführung* aus.

In dem Beispiel 10 auf Seite 10 führt ein fehlender Effekt einer Aktivität (Exportbestimmungen in Deutsch wurden nicht ermittelt) dazu, dass das Ziel ohne Umplanung nicht mehr erreicht werden kann. Allerdings kann auch ein zusätzlicher Effekt die Optimalität der Prozessdefinition gefährden, da der zusätzliche Effekt potentiell einen neuen, besseren Weg zum Ziel eröffnet. Daher kann die Bedingung der Umplanungsregel U_1 nicht weiter eingeschränkt werden.

Durch die Aufhebung der Annahme A12 müssen Änderungen des Ziels während der Bearbeitung eines Geschäftsfalls berücksichtigt werden. Dies geschieht durch die Umplanungsregel U_2:

Umplanungsregel U_2
Ereignis: Das Ziel des Geschäftsfalls ändert sich.
Bedingung: Wahr
Aktion: Führe Prozedur *anpassenAusführung* aus.

Enthält das neue Ziel zusätzliche Aussagen, sind diese unter Umständen im Zielzustand des Prozesses nicht enthalten. Daher ist in diesem Fall die Gültigkeit der Prozessdefinition gefährdet. Ist das neue Ziel *goal'* eine Teilmenge des ursprünglichen Ziels *goal*, so ist ebenfalls eine Umplanung notwendig, da es hierdurch möglicherweise besseren Weg zum Ziel, als die aktuelle Prozessdefinition gibt.

Neben Änderungen des Ziels müssen durch die Aufhebung der Annahme A12 auch Änderungen der Domäne während der Bearbeitung eines Geschäftsfalls berücksichtigt werden. Dies geschieht durch die Umplanungsregel U_3:

Umplanungsregel U_3
Ereignis: Die Domäne wird geändert.
Bedingung: Wahr
Aktion: Führe Prozedur *anpassenAusführung* aus.

Wird die Domäne geändert, so ist in jedem Fall der Gültigkeit beziehungsweise die Optimalität der Prozessdefinition gefährdet. Während die Erweiterung der Domäne um neue Aktivitäten lediglich die Optimalität der Prozessdefinition gefährdet, können nicht mehr verfügbare Aktivitäten auch die Gültigkeit einer Prozessdefinition gefährden, wenn die Aktivitäten Teil der Prozessdefinition sind.

Wird die Annahme A11 aufgehoben, so kann sich der Zustand des Geschäftsfalls auch durch Einwirkung von außen ändern. Die folgende Umplanungsregel U_4 berücksichtigt diesen Fall:

Umplanungsregel U_4
Ereignis: Der Zustand des Geschäftsfalls ändert sich.
Bedingung: Wahr
Aktion: Führe Prozedur *anpassenAusführung* aus.

Gilt durch die Änderung des Zustands eine Aussage nicht mehr, so ist unter möglicherweise die Vorbedingung einer Aktivitätsinstanz oder eine Aussage des Ziels nicht mehr erfüllt. Ein zusätzliche, gültige Aussage kann die Optimalität der Prozessdefinition gefährden, da die zusätzliche Aussage potentiell einen neuen, besseren Weg zum Ziel eröffnet.

Gefährdung der Gültigkeit einer Prozessdefinition

In diesem Abschnitt werden Regeln vorgestellt, die eine Umplanung nur auslösen, falls die Gültigkeit der Prozessdefinition gefährdet ist. Die Bedingungen dieser Regeln sind teilweise restriktiver als die im vorhergehenden Abschnitt vorgestellten Regeln. Dies liegt darin begründet, dass in Fällen, in denen zwar die Optimalität aber nicht die Gültigkeit der Prozessdefinition gefährdet ist, keine Umplanung notwendig ist.

An die Stelle der Umplanungsregel U_1 tritt die Umplanungsregel U_1^*:

Umplanungsregel U_1^*
Ereignis: Eine Aktivitätsinstanz endet.
Bedingung: Der Eintritt des Geschehnisses hat nicht das erwartete Resultat S, sondern das Resultat S', wobei eine Aussage $A \in \mathit{prop}_S - \mathit{prop}_{S'}$ existiert und ((es existiert eine Aktivitätsinstanz i mit $\mathit{exec}_i \neq$ beendet wobei i Element einer Prozessinstanz I ist, die ϕ-Instanz einer Prozessdefinition P ist und $\delta_P(\phi(i)) = d$ und $A \in \mathit{pre}_d$) oder ($A \in$ *goal*))
Aktion: Führe Prozedur *anpassenAusführung* aus.

Die Bedingung der Umplanungsregel U_1^* bringt zu Ausdruck, dass nur Aussagen, die im neuen Zustand entgegen der Erwartung ungültig sind, die Zielerreichung gefährden. Des Weiteren ist dies nur kritisch, wenn das Ziel oder eine Vorbedingung einer noch nicht beendeten Aktivität betroffen ist.

Da sich durch Aufhebung der Annahme A12 das Ziel während der Bearbeitung eines Geschäftsfalls ändern kann, muss gegebenenfalls eine Umplanung ausgelöst werden. Dies geschieht durch die Umplanungsregel U_2^*:

Umplanungsregel U_2^*
Ereignis: Das Ziel des Geschäftsfalls ändert sich von *goal* auf *goal'*.
Bedingung: *goal'* $\not\subseteq$ *goal*
Aktion: Führe Prozedur *anpassenAusführung* aus.

Ändert sich das Ziel des Geschäftsfalls, so ist eine Umplanung nur dann notwendig, wenn das neue Ziel *goal'* keine Teilmenge des ursprünglichen Ziels *goal* ist. Ist dagegen *goal'* eine Teilmenge von *goal*, so ist keine Umplanung notwendig, da die Zielerreichung nicht gefährdet ist. Dies zeigt sich wie folgt: Angenommen eine Prozessinstanz hat einen finalen Zustand S_{final}, der ein Zielzustand ist. Dann bedeutet dies, dass das Ziel *goal* eine Teilmenge von $\mathit{prop}_{S_{final}}$ ist. Wenn das neue Ziel *goal'* eine Teilmenge von *goal* ist, ist somit *goal'* auch eine Teilmenge von $\mathit{prop}_{S_{final}}$. Folglich ist der finale Zustand S_{final} auch ein Zielzustand bezüglich des neuen Ziels *goal'*.

Änderungen der Domäne während der Bearbeitung eines Geschäftsfalls werden durch die Umplanungsregel U_3^* berücksichtigt:

Umplanungsregel U_3^*
Ereignis: Eine Aktivitätsdefinition d der Domäne wird entfernt oder geändert.
Bedingung: Es existiert eine Aktivitätsinstanz i mit *exec*$_i \neq$ beendet, wobei i Element einer Prozessinstanz I ist, die ϕ-Instanz einer Prozessdefinition P ist und $\delta_P(\phi(i)) = d$
Aktion: Führe Prozedur *anpassenAusführung* aus.

Eine Erweiterung der Domäne um zusätzliche Aktivitätsdefinitionen gefährdet die Gültigkeit einer Prozessdefinition nicht. Wird eine Aktivitätsdefinition der Domäne wird entfernt oder geändert, so ist eine Umplanung nur dann notwendig, wenn es eine entsprechende Aktivitätsinstanz gibt, die noch nicht beendet ist. Ansonsten ist die Ausführung der Prozessinstanz nicht von der Änderung der Domäne betroffen.

Wird die Annahme A11 aufgehoben, so kann sich der Zustand des Geschäftsfalls auch durch Einwirkung von außen ändern. Die folgende Umplanungsregel U_4^* berücksichtigt diesen Fall:

Umplanungsregel U_4^*
Ereignis: Der Zustand S des Geschäftsfalls ändert sich ohne Beendigung einer Aktivitätsinstanz zu dem Zustand S'.
Bedingung: Es existiert eine Aussage $A \in$ *prop*$_S -$ *prop*$_{S'}$ und ((es existiert eine Aktivitätsinstanz i mit *exec*$_i \neq$ beendet wobei i Element einer Prozessinstanz I ist, die ϕ-Instanz einer Prozessdefinition P ist und $\delta_P(\phi(i)) = d$ und $A \in$ *pre*$_d$) oder ($A \in$ *goal*))
Aktion: Führe Prozedur *anpassenAusführung* aus.

Die Bedingung der Umplanungsregel U_4^* bringt zu Ausdruck, dass nur Aussagen, die ungültig werden, die Zielerreichung gefährden. Diese gefährden die Gültigkeit einer Prozessdefinition auch nur dann, wenn das Ziel oder eine Vorbedingung einer noch nicht beendeten Aktivität betroffen ist.

5.4 Zusammenfassung

In diesem Kapitel wurde ein formales Modell einer integrierten Prozessplanung und -ausführung formuliert und untersucht. Die in den vorhergehenden Kapiteln entwickelten Konzepte wurden präzisiert und die involvierten Artefakte formalisiert, wodurch sie einer analytischen Betrachtung zugänglich wurden. Die Spezifikation der Funktionalität des Planers erfolgte deskriptiv, indem definiert wurde, unter welchen Bedingungen eine Prozessdefinition für ein Planungsproblem gültig ist. Damit wurde von einem konkreten Planungsalgorithmus abstrahiert, so dass die aus anschließenden Analyse gewonnenen Erkenntnisse unabhängig von dem verwendeten Planungsalgorithmus sind. Die Funktionalität der Prozesssteuerung und der Prozessüberwachung wurde demgegenüber durch Prozeduren und auf Basis von Ereignis-Bedingungs-Aktions-Regeln modelliert. Aufbauend auf diese For-

malisierung wurden die Zusammenhänge und Abläufe analysiert und eine Reihe von Eigenschaften eines Systems zur integrierten Prozessplanung und -ausführung abgeleitet. Es wurde gezeigt, unter welchen Bedingungen der Zustand eines Geschäftsfalls durch Planung und Ausführung eines passenden Prozesses in einen Zielzustand überführt wird. Des Weiteren wurde nachgewiesen, dass die spezifizierte Vorgehensweise eine konsistente Anpassung eines Prozesses zur Laufzeit ermöglicht.

Kapitel 6

Realisierung

In diesem Kapitel wird mit dem Prototypen *Plaengine* die softwaretechnische Realisierung der in den Kapiteln 3 und 4 vorgestellten Konzepte zur integrierten Prozessplanung und -ausführung vorgestellt. Plaengine dient somit der Validierung der softwaretechnischen Umsetzbarkeit dieser Arbeit. Insbesondere wird die Möglichkeit zur Verknüpfung von Planer und Prozesssteuerung nachgewiesen und gezeigt, wie Prozessdefinitionen individuell für Geschäftsfälle geplant, ausgeführt und gegebenenfalls angepasst werden. Entwurf und Implementierung von Plaengine fand im Rahmen eines Projekts im Fachgebiet Business Process Technology des Hasso-Plattner-Instituts für Software-Systemtechnik an der Universität Potsdam statt. Die Projektergebnisse sind im Detail in [47], [72] und [79] beschrieben. Bezüglich der Realisierung von Plaengine ist momentan der zweite Prototyp implementiert. Im Gegensatz zum ersten Prototypen kommt dabei ein selbstentwickelter KI-Planer zum Einsatz, der insbesondere auf die Planung in einer Geschäftsdomäne zugeschnitten ist.

Abschnitt 6.1 beschreibt die zentralen Entwurfsziele von Plaengine. In Abschnitt 6.2 wird darauf aufbauend die Systemarchitektur von Plaengine dargestellt. Anschließend zeigt Abschnitt 6.3 die statische Struktur der relevanten konzeptionellen und technischen Artefakte aus Planung und Ausführung. Abschnitt 6.4 illustriert die Bearbeitung eines Geschäftsfalls in Plaengine am Beispiel der Angebotsanfrage, das bereits aus den verhergehenden Kapiteln bekannt ist. Schließlich erfolgt eine Zusammenfassung dieses Kapitels in Abschnitt 6.5.

6.1 Entwurfsziele

Zentrales Ziel des Plaengine-Projekts, ist die Realisierung eines prototypischen Softwaresystems zur integrierten Planung und Ausführung von Geschäftsprozessen. Als Kernfunktionalität soll Plaengine automatisch für jeden Geschäftsfall eine individuelle Prozessdefinition planen und damit die Bearbeitung des Geschäftsfalls steuern. Des Weiteren ist die Möglichkeit zur automatischen Prozessanpassung eine wesentliche Anforderung. Neben diesen zentralen Anforderungen verfolgt die

Entwicklung von Plaengine weitere Ziele, die sich im Entwurf des Systems widerspiegeln:

- **Komponentenbasierte Entwicklung:** Bei der Realisierung von Plaengine wird eine komponentenbasierte Entwicklung [24] verfolgt. Zentrale Komponenten wie Planer und Prozesssteuerung sind gekapselt und interagieren durch klar definierte Schnittstellen. Dadurch werden einzelne Komponenten leichter wiederverwendbar oder andererseits durch alternative Komponenten ersetzbar. Besonders bedeutend hierbei ist die Möglichkeit, durch Austausch der Planungskomponente alternative Planungsalgorithmen einsetzen zu können. Des Weiteren wird durch die Kapselung das Problem der Entwicklung von Plaengine in handhabbare und personell gut verteilbare Teilprobleme zerlegt.
- **Dienstorientierung:** Plaengine ist für den Einsatz in einer dienstorientierten Umgebung (Service Oriented Environment) [20] konzipiert. Das System soll in der Lage sein, Dienste anhand ihrer veröffentlichten Dienstspezifikation zu finden und aufzurufen. Die Dienstspezifikationen werden in einem Dienstverzeichnis gespeichert und verwaltet. Hierbei wird zwischen einem atomaren Dienst, der einer Aktivität entspricht, und einem zusammengesetzten Dienst, der einem Prozess entspricht, unterschieden. Als konkrete Technologie für eine dienstorientierte Umgebung unterstützt Plaengine Web-Services [54, 107] und insbesondere zusammengesetzte Web-Services [10, 32].
- **Persistenz und Wiederverwendung:** Zusammengesetzte Dienste, die für einen bestimmten Geschäftsfall generiert wurden, sollen im Dienstverzeichnis gespeichert und wiederverwendet werden können. Aus diese Weise wird das in Abschnitt 7.4.2 vorgestellte Konzept zur Wiederverwendung von Prozessdefinitionen umgesetzt.

6.2 Systemarchitektur

Das Komponentendiagramm in der Unified-Modeling-Language in Abbildung 6.1 zeigt die grobe Systemarchitektur von Plaengine bestehend aus vier wesentlichen Komponenten. Die beiden Komponenten Planer und Prozesssteuerung wurden bereits in Abbildung 3.1 auf Seite 24 vorgestellt. Eine weitere Komponente ist das Dienstverzeichnis, dass die Domänenbeschreibung verwaltet. Des Weiteren kommt durch die softwaretechnische Umsetzung die Komponente Geschäftsfallsteuerung hinzu. Im Folgenden werden die Komponenten kurz vorgestellt.

6.2.1 Dienstverzeichnis

Entsprechend dem Entwurfziel der Dienstorientierung werden die zur Verfügung stehenden Aktivitäten als *Dienste* aufgefasst und in dem Dienstverzeichnis regis-

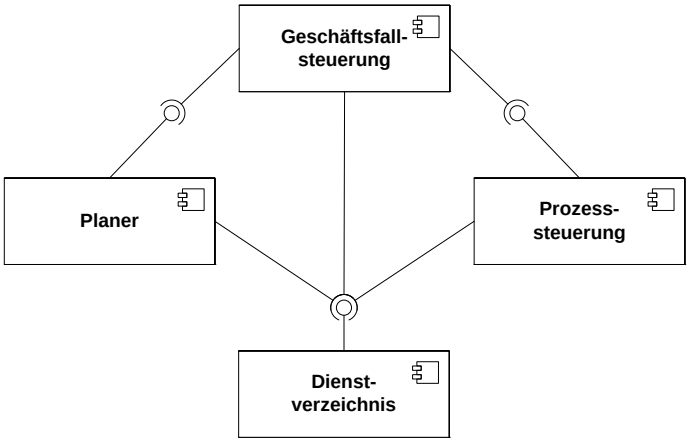


Abbildung 6.1: Komponentendiagramm: Systemarchitektur Plaengine

triert. Das Dienstverzeichnis verwaltet die Dienstspezifikationen aller registrierten Dienste. Neben den für die Planung benötigten Informationen über Vorbedingungen und Effekte der Dienste enthält eine *Dienstspezifikation* Informationen darüber, wie an den Dienst gebunden werden kann und welche Ein- und Ausgabe-Parameter dieser besitzt. Das Dienstverzeichnis speichert diese Daten persistent und erlaubt einen gezielten und effizienten Zugriff auf die Dienstspezifikationen. Hierzu gehört insbesondere die Möglichkeit einer auf logischen Ausdrücken basierenden Suche nach zu einer Anfrage passenden Dienstspezifikationen. Hierzu müssen die Vorbedingungen und Effekte der registrierten Dienste auf Äquivalenz zu einer Anfrage untersucht werden.

Neben atomaren Diensten werden in diesem Dienstverzeichnis auch *zusammengesetzte Dienste* verwaltet, die vom Planer aus mehreren Diensten zusammengestellt wurden. Ein zusammengesetzter Dienst kann dabei sowohl atomare als auch weitere zusammengesetzte Dienste umfassen. Zusammengesetzte Dienste besitzen wie jeder Dienst eine Dienstspezifikationen, die Informationen über Vorbedingungen und Effekte des Dienstes enthält. Hiermit lassen sich zusammengesetzte Dienste, genau wie atomare Dienste, über logische Ausdrücke suchen. Die Dienstspezifikation eines zusammengesetzten Dienstes umfasst des weiteren eine Prozessdefinition, die beschreibt, in welcher Reihenfolge die enthaltenen Dienste bei einer Ausführung aufzurufen sind. Auf dieser Basis wird das Entwurfsziel der Wiederverwendung von Prozessdefinitionen in Form von zusammengesetzten Diensten wie folgt realisiert: Falls bei der Suche nach einem Dienst für die Bearbeitung eines gegebenen Geschäftsfalls kein passender Dienst gefunden wird, versucht der Planer verfügbare Dienste so zu kombinieren, dass ein neuer, zusammengesetzter Dienst entsteht, der die Aufgabe löst. Gelingt dies, wird der Geschäftsfall auf Basis des zusammengesetzten Dienstes bearbeitet. Nach einer erfolgreichen Bearbeitung des Geschäftsfalls, wird der neue, zusammengesetzte Dienst im Dienstverzeichnis

registriert. Falls später ein strukturgleicher Geschäftsfall bearbeitet werden soll, kann der zusammengesetzte Dienst wiederverwendet werden.

6.2.2 Planer

In dem ersten Prototypen von Plaengine wurde der von Hoffmann entwickelte KI-Planer Metric-FF [52] gekapselt als Planungskomponente eingesetzt. Dieser Planer erfüllt die in Abschnitt 3.3 diskutierten Anforderungen an einen Planer in einer Geschäftsdomäne jedoch nur zum Teil. Insbesondere fehlt bei diesem Planer die Möglichkeit die Ausführungsdauer von Aktivitäten bei der Planung zu berücksichtigen. Aus diesem Grunde wurde für den zweiten Plaengine Prototypen ein eigener Planer entwickelt, der speziell auf die Anforderungen einer Geschäftsdomäne zugeschnitten ist. Wesentliche Eigenschaften dieses Planers sind die Planung nebenläufiger Prozesse, die Berücksichtigung von nicht-deterministischen Diensten und die Möglichkeit Objekterzeugung in die Planung einzubeziehen. Die Ausgabe der Planers ist ein temporaler Plan der als partiell geordnete Menge von Diensten dargestellt wird. Wie in Abschnitt 3.1.1 diskutiert bildet damit die Ausgabe dieses Planers das Äquivalent zu einer graphbasierten Prozessdefinition im Workflow-Management. Im Gegensatz dazu bieten die meisten anderen existierenden temporalen KI-Planer startzeitgebundene Pläne als Ausgabe, womit die Integration mit der Prozessausführungs-Komponente weitaus aufwendiger wäre. Für die Planung greift der Planer auf die im Dienstverzeichnis abgelegten Dienstspezifikationen zu, da hier die benötigten Informationen über Vorbedingungen und Effekte der verfügbaren Dienste zu finden sind.

6.2.3 Prozesssteuerung

Die Prozesssteuerung hat insbesondere die Aufgabe die Ausführung von zusammengesetzten Diensten zu steuern. Hierzu müssen die atomaren Dienste, aus denen ein zusammengesetzter Dienst besteht, entsprechend der Prozessdefinition der Dienstspezifikation ausgeführt werden. Ein atomarer Dienst ist dabei entweder ein Web-Service oder eine Aktivität, die über eine Arbeitsliste einem Benutzer zur Bearbeitung angeboten wird. Die Zuordnung geschieht in diesem Fall über eine dynamische Rollenauflösung. Zu den Aufgaben der Prozesssteuerung gehört der Aufruf eines Dienstes, die Übergabe der benötigten Daten und gegebenenfalls die Entgegennahme des Ergebnisses. Darüber hinaus hat die Prozesssteuerung die Aufgabe, die Ausführung eines Dienstes zu überwachen. Die Grundlage für diese Überwachung stellen Fehlermeldungen der einzelnen Dienste dar. Diese geben Auskunft darüber, welche Effekte die Ausführung des Dienstes tatsächlich hatte. Diese Überwachung ist eine Voraussetzung für die in Kapitel 4 vorgestellte automatische Prozessanpassung. Sollen Dienste eingebunden werden, die keine Informationen über die tatsächlich erzielten Effekte liefern, so müssen sie derart gekapselt werden, dass nach der Ausführung des Dienstes das Ergebnis überprüft wird.

6.2.4 Geschäftsfallsteuerung

Die Geschäftsfallsteuerung nimmt Geschäftsfälle entgegen und steuert deren Fluss durch das System. Wie die Interaktion der Komponenten im System für den Fall einer erfolgreichen Bearbeitung eines Geschäftsfalls aussieht, zeigt das UML-Sequenzdiagramm in Abbildung 6.2. Im ersten Schritt überprüft die Geschäftsfallsteuerung, ob es schon existierende Dienste gibt, die den Geschäftsfall bearbeiten können. Hierzu befragt sie das Dienstverzeichnis, das die Dienstspezifikationen verwaltet. Findet sich dort kein passender atomarer oder zusammengesetzter Dienst, so tritt der Planer in Aktion. Er lädt die notwendigen Domäneninformationen aus dem Dienstverzeichnis und sucht eine Prozessdefinition die verfügbare Dienste so kombiniert, dass ein neuer zusammengesetzter Dienst entsteht, der den Geschäftsfall bearbeiten kann. Danach wird der Geschäftsfall von der Geschäftsfallsteuerung an die Prozesssteuerung übergeben, die den zusammengesetzten Dienst ausführt. Dabei werden die enthaltenen Dienste entsprechend der Prozessdefinition aufgerufen und ihre Effekte überwacht. Treten bei der Ausführung Probleme auf, die nicht innerhalb der Prozesssteuerung gelöst werden können, so liefert diese den Geschäftsfall an die Geschäftsfallsteuerung zurück, die eine automatische Prozessanpassung veranlasst. Hierzu kann gegebenenfalls der Planer mit dem aktuellen Zustand des Geschäftsfalls erneut aufgerufen werden, um eine neuen zusammengesetzten Dienst zu erzeugen. Zusammenfassend steuert die Geschäftsfallsteuerung das Zusammenwirken der Komponenten Planer und Prozesssteuerung. Damit entfällt die Notwendigkeit für eine direkte Interaktion von Planer und Prozesssteuerung, wodurch Abhängigkeiten zwischen diesen Komponenten vermieden werden. Auf diese Weise wird das Entwurfsziel einer komponentenbasierten Entwicklung umgesetzt und einzelne Komponenten werden leichter austauschbar und wiederverwendbar.

6.3 Integriertes Domänenmodell

Eine wesentliche Voraussetzung für die präzise Verknüpfung der einzelnen Systemkomponenten ist eine Integration der Strukturmodelle der Komponenten. Hierbei wird die in Abschnitt 3.1.1 beschrieben Integration gemeinsamer Konzepte aus Workflow-Management und KI-Planung softwaretechnisch umgesetzt.

In Abbildung 6.3 wird das Domänenmodell als UML-Klassendiagramm dargestellt. Geschäftsfälle werden durch die zentrale Klasse `Case` repräsentiert. In einem `Case` werden unter anderem Informationen über den Ausgangszustand, das Ziel, die Bearbeitungsphase und die zugeordnete Domäne gespeichert. Die Domäne wird durch die Klasse `Domain` beschrieben und enthält alle planungsrelevanten Informationen für eine bestimmte Organisation. Dazu gehört die Definition von unterschiedlichen Typen von Ausdrücken und darauf aufbauende Aktivitätsdefinitionen. Aktivitätsdefinitionen werden durch die Klasse `ActivityDefinition` repräsentiert und stellen ein zentrales Element sowohl für den Planer als auch für die Prozesssteuerung dar. Prozessdefinitionen werden als zusammengesetzte

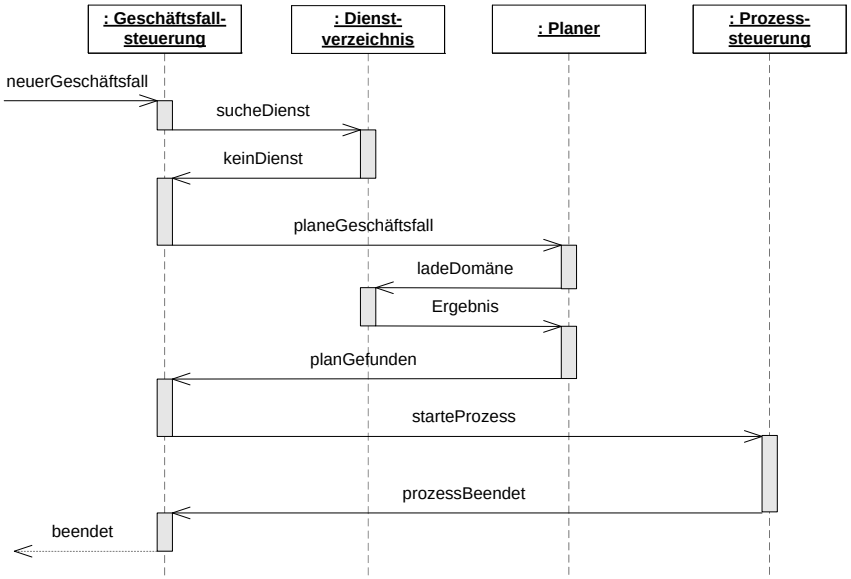


Abbildung 6.2: Sequenzdiagramm: Bearbeitung eines Geschäftsfalls

Dienste aufgefasst und in der Klasse `ComplexActivityDefinition` als Spezialisierung einer `ActivityDefinition` spezifiziert. Jeder atomare Dienst ist entweder ein Web-Service (Klasse `WebServiceOperation`) oder eine manuelle Aktivität (Klasse `WorklistActivityDefinition`). Beide Klassen stellen eine Spezialisierung der abstrakten Klasse `AtomicActivityDefinition` dar. Eine manuelle Aktivität wird über Rollen (Klasse `Role`) Personen (Klasse `Person`) zugeordnet. Eine Organisation umfasst eine Menge von Personen und Rollen und ist einer Domäne zugeordnet, die die Dienste der Organisation spezifiziert. Die konkrete Ausführung eines Dienstes beziehungsweise eines zusammengesetzten Dienstes wird durch die Klassen `ActivityInstance` beziehungsweise `ComplexActivityInstance` abgebildet. Für die Planung müssen Vorbedingungen und Effekte von Aktivitäten spezifiziert werden. Dies geschieht auf Basis logischer als auch numerischer Ausdrücke (Klasse `Expression`) die über die Klasse `ActivityProfile` den Aktivitätsdefinitionen zugeordnet werden. Die Typen die hierbei verwendet werden können, sind in der Domäne festgelegt und werden durch die Klasse `Type` verwaltet. In Bezug auf numerische Ausdrücke kann für die Planung ein Maximal- oder Minimalwert angegeben werden (Klasse `MetricByLimit`), beziehungsweise ein Optimierungsziel festgelegt werden (Klasse `MetricByOptimization`).

Das gemeinsame Domänenmodell ist ein wichtiger Schritt zur softwaretechnischen Integration der Komponenten Planer und Prozesssteuerung. Zentrale Klassen wie `Case`, `ActivityDefinition`, `ComplexActivityDefinition` werden sowohl vom Planer als auch von der Prozesssteuerung verwendet und bil-

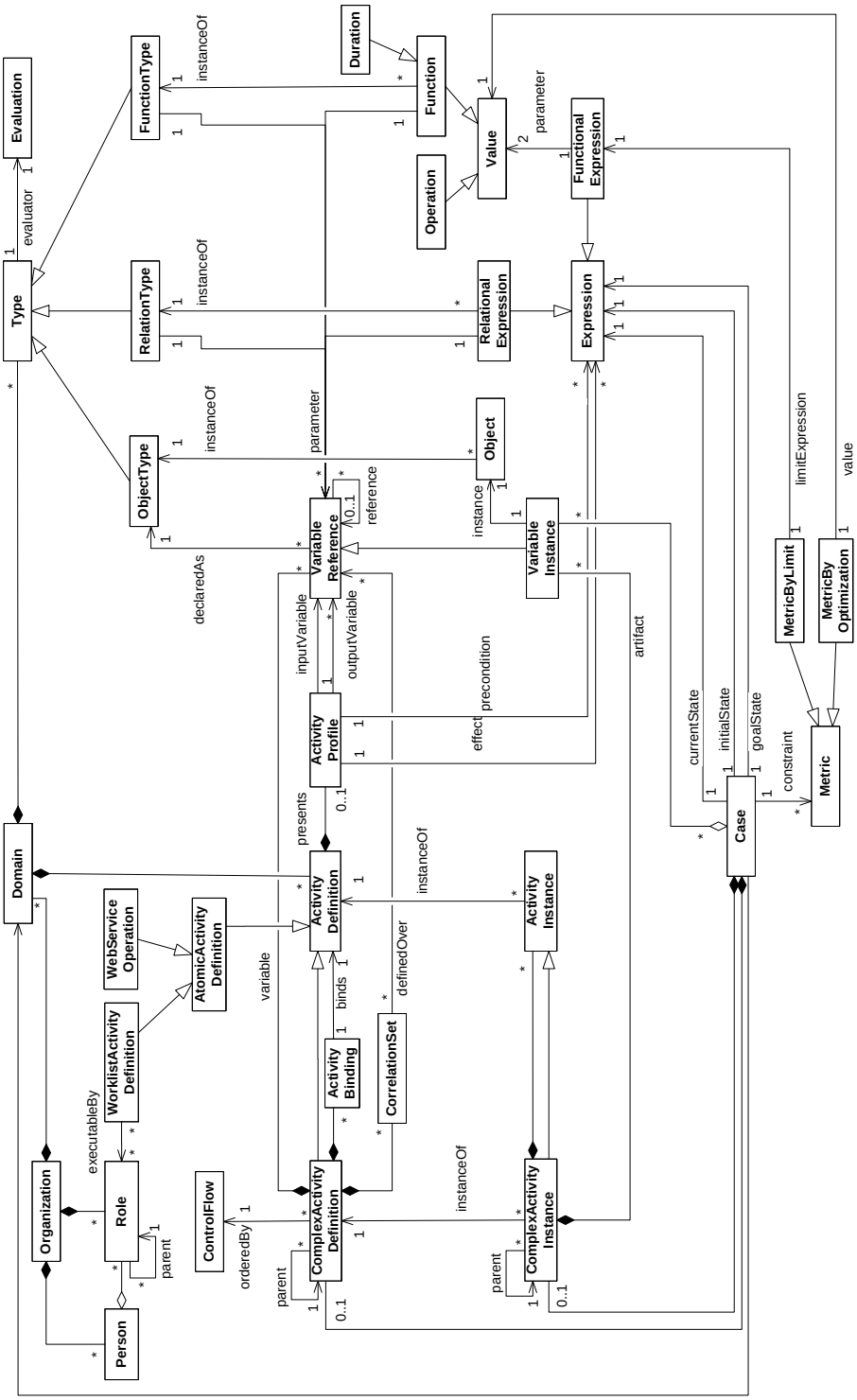


Abbildung 6.3: Klassendiagramm: Domänenmodell von Plaengine

den so bezügliche der Daten die Verbindung zwischen den Komponenten. Andere Klassen wie `ActivityInstance` oder `ComplexActivityInstance` werden nur von der Prozesssteuerung verwendet, stehen aber mit obigen zentralen Klassen in Beziehung. Ebenso verhält es sich mit Klassen wie beispielsweise `ActivityProfile` oder `Expression`, welche nur vom Planer verwendet werden. Zusammenfassend erfolgt die Integration auf der Ebene des Domänenmodells über eine gemeinsame Schnittmenge von Klassen zwischen Planer und Prozesssteuerung.

6.4 Beispiel

In diesem Abschnitt wird die Bearbeitung eines Geschäftsfalls durch Plaengine an dem in Abschnitt 3.1.3 eingeführten Beispiel einer Angebotsanfrage illustriert. Es wird davon ausgegangen, dass die Domäne auf Basis des Domänenmodells von Plaengine spezifiziert ist. Betrachten wir hierzu die Aktivität *kalkulieren* aus Abbildung 3.4. Hierbei handelt es sich um eine manuelle Aktivität und somit um ein Objekt der Klasse `WorklistActivityDefinition`. Dieses *kalkulierenWAD* benannte Objekt ist in dem Objektdiagramm in Abbildung 6.4 dargestellt. Dieses Objektdiagramm bezieht sich auf das in Abbildung 6.3 dargestellte Klassendiagramm von Plaengine. Da die Klasse `WorklistActivityDefinition` eine Spezialisierung der Klasse `ActivityDefinition` ist, existiert ein zugeordnetes Objekt *kalkulierenAP* der Klasse `ActivityProfile`. Dem Objekt *kalkulierenAP* ist ein Objekt *kalkRE1* der Klasse `RelationalExpression` als Vorbedingung zugeordnet. Dieser Zusammenhang wird durch die Beziehung `precondition` zum Ausdruck gebracht. Das Objekt *kalkRE1* ist einem Objekt *erfasstRT* der Klasse `RelationType` zugeordnet. Damit wird die in Abbildung 3.4 dargestellte Vorbedingung *erfasst* der Aktivität *kalkulieren* in Plaengine spezifiziert. Da es sich bei der Vorbedingung um einen aussagenlogischen Ausdruck handelt, ist die Relation 0-stellig und das Objekt *erfasstRT* keinem Parameter zugeordnet.

Analog zu dieser Vorbedingung ist der Effekt *kalkuliert* spezifiziert. Die Ausführungsdauer der Aktivität *kalkulieren* wird ebenfalls als Effekt aufgefasst. Hierbei wird jedoch kein Objekt der Klasse `RelationalExpression` verwendet, sondern ein Objekt der Klasse `FunctionalExpression`. Neben diesen für die Planung benötigten Informationen über Vorbedingungen und Effekte der Aktivität sind auch für die Ausführung der Aktivität benötigte Informationen, wie zum Beispiel die zugeordnete Rolle, spezifiziert. Hier wird davon ausgegangen, dass für die Ausführung der Aktivität *kalkulieren* ein kaufmännischer Sachbearbeiter benötigt wird. Zu diesen Ausführungsinformation gehören auch Ein- und Ausgabeparameter der Aktivität, die zur Vereinfachung in dem Objektdiagramm nicht dargestellt sind. Vor der Bearbeitung einzelner Geschäftsfälle durch die Geschäftsfallsteuerung müssen alle Aktivitäten der Domäne, wie hier anhand der Aktivität *kalkulieren* dargestellt, spezifiziert sein.

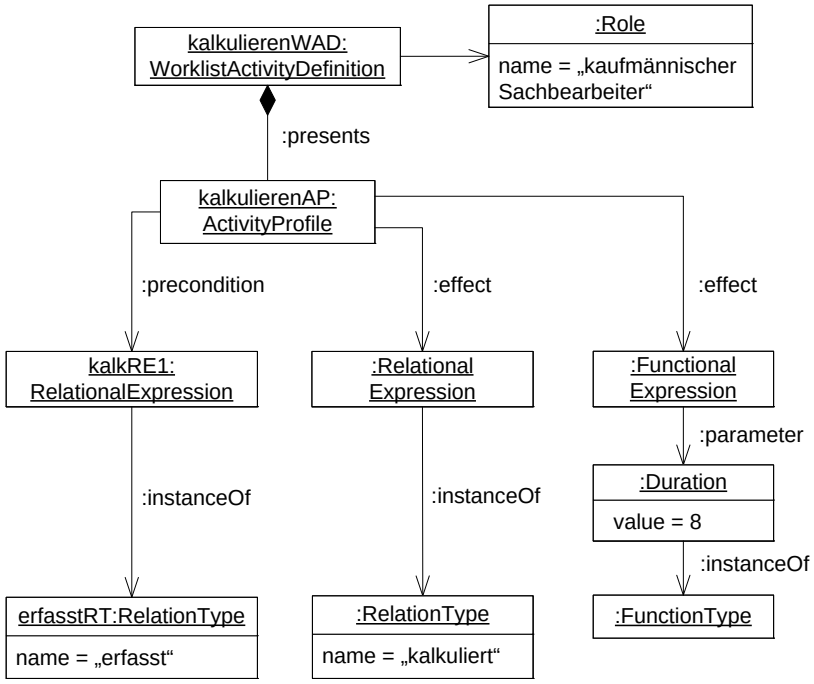


Abbildung 6.4: Objektdiagramm: Modellierung der Aktivität *kalkulieren*

Betrachten wir nun die Bearbeitung eines konkreten Geschäftsfalls am Beispiel einer eingehenden Angebotsanfrage, die von der Geschäftsfallsteuerung entgegengenommen wird. Der aktuelle Zustand des Geschäftsfalls sei, dass die Aussage *anfrage* gilt. Das Ziel des Geschäftsfalls sei, dass die Aussage *beantwortet* gilt. Es wird ein entsprechendes Objekt *case1* der Klasse `Case` erzeugt. Über die Beziehungen `initialState` und `goalState` werden dem Objekt *case1* der aktuelle Zustand und das Ziel des Geschäftsfalls zugeordnet. Hierbei handelt es sich um Objekte der Klasse *RelationalExpression*. Wenn wir die Ausführungsdauer des zusammengesetzten Dienstes als Optimierungskriterium annehmen, wird dies durch ein Objekt der Klasse `MetricByOptimization` zum Ausdruck gebracht, das dem Objekt *case1* zugeordnet ist.

Die Geschäftsfallsteuerung überprüft zunächst, ob es schon existierende Dienste gibt, die den Geschäftsfall bearbeiten können. Das heißt, es wird ein atomarer oder zusammengesetzter Dienst gesucht, der als Vorbedingung den aktuellen Zustand und als Effekt das Ziel des Geschäftsfalls hat. Hierzu befragt sie das Dienstverzeichnis, das die Dienstspezifikationen verwaltet. An dieser Stelle wird davon ausgegangen, dass noch kein passender Dienst verzeichnet ist. Daher ruft die Geschäftsfallsteuerung den Planer mit dem aktuellen Zustand und Ziel des Geschäftsfalls auf. Der Planer lädt die notwendigen Domäneninformationen aus dem Dienstverzeichnis und sucht eine Prozessdefinition, die verfügbare Dienste so kombiniert, dass ein neuer zusammengesetzter Dienst entsteht, der den Geschäftsfall bearbeiten

kann. In dem Beispiel der Angebotsanfrage bedeutet dies, dass ein neues Objekt der Klasse `ComplexActivityDefinition` erzeugt wird, dem analog zu Abbildung 3.5 die Aktivitätsdefinitionen *erfassen*, *pruefen*, *kalkulieren* und *antworten* zugeordnet werden. Nach der erfolgreichen Planung wird der Geschäftsfall von der Geschäftsfallsteuerung an die Prozesssteuerung übergeben, die den zusammengesetzten Dienst ausführt. Dabei werden die enthaltenen Dienste entsprechend der Prozessdefinition aufgerufen. Diese Aufrufe können Web-Services sein oder wie bei der manuellen Aktivität *kalkulieren* der Eintrag auf den Arbeitslisten der kaufmännischen Sachbearbeiter.

6.5 Zusammenfassung

In diesem Kapitel wurde mit dem Prototypen Plaengine eine softwaretechnische Realisierung der in den vorhergehenden Kapiteln vorgestellten Konzepte zur integrierten Prozessplanung und -ausführung beschrieben. Neben einer weiteren Konkretisierung dieser Konzepte dient Plaengine dabei der Validierung der softwaretechnischen Umsetzbarkeit einer Verknüpfung von Planungs- und Prozesssteuerungskomponenten. Zunächst wurden mit der komponentenbasierten Entwicklung, der Dienstorientierung, sowie der Persistenz und Wiederverwendung von Diensten die wesentlichen Entwurfsziele bei der Entwicklung von Plaengine dargestellt. Die komponentenbasierte Entwicklung verfolgt dabei das Ziel, zentrale Komponenten wie Planer und Prozesssteuerung durch klar definierte Schnittstellen zu kapseln und damit die Möglichkeit offen zu halten, einzelne Komponenten konsistent wiederverwenden zu können beziehungsweise leicht gegen alternative Implementierungen austauschen zu können. Anschließend wurde die Systemarchitektur von Plaengine mit ihren vier zentralen Komponenten vorgestellt: der Planer, die Prozesssteuerung, das Dienstverzeichnis und die Geschäftsfallsteuerung. Im Anschluss an die Systemarchitektur wurde die statische Struktur der konzeptionellen und technischen Artefakte von Plaengine und ihre Verknüpfungen in einem Klassendiagramm in der Unified-Modeling-Language beschrieben. Dabei stellen die gemeinsamen Klassen der Prozessplanung einerseits und der Prozessausführung andererseits einen wichtigen Teil der softwaretechnischen Umsetzung der Integration von Prozessplanung und -ausführung dar. Schließlich wurde die die Bearbeitung eines Geschäftsfalls in Plaengine an dem bereits in Kapitel 3 eingeführten Beispiel einer Angebotsanfrage illustriert.

Kapitel 7

Diskussion

In diesem Kapitel werden die Annahmen, Konzepte und Lösungen dieser Arbeit diskutiert. Abschnitt 7.1 zeigt auf, inwieweit die bei der Modellbildung getroffenen Annahmen eine adäquate Abstraktion beziehungsweise Einschränkung darstellen und welche Auswirkungen die Aufhebung einzelner Annahmen hätte. Des Weiteren wird in Abschnitt 7.2 die Verknüpfung von Workflow-Management und KI-Planung auf konzeptioneller Ebene und auf softwaretechnischer Ebene reflektiert. Anschließend identifiziert Abschnitt 7.3 Anwendungsgebiete, in denen sich aufgrund ihrer jeweiligen Charakteristika der Einsatz einer integrierten Prozessplanung und -ausführung als besonders vorteilhaft darstellt. Schließlich werden in Abschnitt 7.4 sinnvolle Erweiterungen der vorgestellten Konzepte skizziert und diskutiert.

7.1 Annahmen der Modellbildung

Das in dieser Arbeit vorgestellte Konzept zur integrierten Prozessplanung und -ausführung basiert auf den Annahmen A1 bis A7, die auf Seite 28 beschrieben werden. Für die Formulierung des formalen Modells wurden darüber hinaus die Annahmen A8 bis A12 aufgestellt, die auf Seite 56 beschrieben werden. Diese Annahmen dienen der Vereinfachung des formalen Modells auf die für eine Integration von Planung und Ausführung zentralen Aspekte. Auf viele dieser Annahmen kann bei entsprechender Erweiterung des Modells verzichtet werden, ohne dass die wesentlichen Aussagen ihre Gültigkeit verlieren. In Abschnitt 5.3.4 wird mit der erweiterten Umplanung eine derartige Erweiterung für den Verzicht auf die Annahmen A11 und A12 vorgestellt. Die Annahmen A8 und A9 besagen, dass sich Zustände, Vorbedingungen und Effekte mit Aussagenlogik beschreiben lassen. Auch hier kann das Modell erweitert werden, so dass auch numerische Werte wie zum Beispiel Kosten für die Ausführung von Aktivitäten spezifiziert und bei der Planung berücksichtigt werden können. Hierzu ist eine entsprechende Erweiterung der Definitionen von Aktivitäten und Zuständen, sowie der Gültigkeit und Optimalität einer Prozessdefinition notwendig. Die in Kapitel 6 vorgestellte prototypische Realisie-

rung *Plaengine* geht in diesem Punkt über das formale Modell hinaus, indem auch numerische Werte bei der Planung berücksichtigt werden können. Beispielsweise lassen sich Kosten für die Ausführung einzelner Aktivitäten spezifizieren und die Gesamtkosten eines Prozesses als Optimierungskriterium festlegen.

Annahme A5 besagt, dass die Ausführung von Aktivitäten im Allgemeinen nicht unterbrochen werden kann. Entsprechend sind im Zustandsautomaten in Abbildung 3.3 auch keine Ausführungszustände oder Zustandsübergänge für eine Unterbrechung vorgesehen. Diese Annahme passt zu den ständig an Bedeutung gewinnenden dienstbasierten Architekturen, da insbesondere bei einer losen Kopplung zwischen Diensten nicht davon ausgegangen werden kann, dass ein aufgerufener Dienst jederzeit unterbrochen werden kann.

In Bezug auf Unterstützung unterschiedlicher Arten von Kontrollfluss wird im formalen Modell von sequentiellen und nebenläufigen Abläufen ausgegangen, die sich als Graph modellieren lassen. Eine Erweiterung um beispielsweise alternative Ausführungspfade hätte jedoch keine wesentlichen Auswirkungen auf die Vorgehensweise zur Integration von Planung und Ausführung beziehungsweise auf die Vorgehensweise zur automatisierten Prozessanpassung. Hierfür muss in erster Linie die Ausdruckskraft der Domäne erweitert werden, damit Aktivitäten mit alternativen Effekten modelliert werden können. Bei der Umplanung spielen alternative Ausführungspfade hingegen keine Rolle, da lediglich der aktuelle Zustand und die zur Zeit laufenden Aktivitäten relevant sind. In der prototypischen Realisierung Plaengine können alternative Ausführungspfade geplant und ausgeführt werden.

Der Datenfluss in einem Prozess wurde bei der Integration von Prozessplanung und -ausführung nicht explizit betrachtet. Allerdings wurde in Abschnitt 3.1.1 die Analogie zwischen Datenkonnektoren einer Prozessdefinition und kausalen Verknüpfungen eines Plans, die auf der Verfügbarkeit von Daten basieren, gezeigt. Die Überlegung dabei ist, dass eine derartige kausale Verknüpfung zum Ausdruck bringt, dass eine Aktion als Effekt Daten bereitstellt, deren Verfügbarkeit eine Vorbedingung einer anderen Aktion ist. In Plaengine werden auf diese Weise Datenkonnektoren aus kausalen Verknüpfungen abgeleitet.

Es wird an in dieser Arbeit entsprechende Annahme A10 davon ausgegangen, dass die Dauer des Umplanungsvorgangs bekannt ist. Dies ist bei Planungsalgorithmen im Allgemeinen nicht der Fall, da die Planungsdauer stark von dem zu lösenden Planungsproblem abhängt. Allerdings zeigt sich in vielen Fällen, dass insbesondere informationsverarbeitende Prozesse, die insofern einfach strukturiert sind, dass keine Aktivitäten existieren, die die Effekte anderer Aktivitäten negieren, sehr kurze Planungsdauern erreicht werden. Diese können im Vergleich zu den Ausführungsdauern der Aktivitäten vernachlässigt werden. Andererseits wurden sogenannte Anytime-Planer [26] entwickelt, bei denen der Planungsvorgang zu jedem Zeitpunkt abgebrochen werden kann und dann die bis dahin beste, gefundene Prozessdefinition ausgegeben wird. Diese entsprechen genau der Annahme A10, da die Planungsdauer im Voraus festgelegt werden kann.

Neben der Dauer des Umplanungsvorgangs wird auch zumindest die erwartete Ausführungsdauer von Aktivitäten als in der Domäne modellierbar angenommen.

Dies ist kritisch zu sehen, da die Ausführungsdauer insbesondere bei informationsverarbeitende Prozessen typischerweise eng mit der zu bearbeitenden Datenmenge zusammenhängt. Diese Abhängigkeit ist jedoch schwer modellierbar. Allerdings haben Abweichungen von der erwarteten Ausführungsdauer nur Auswirkungen auf die Optimalität einer Prozessdefinition, nicht jedoch auf ihre Gültigkeit. In Kapitel 5 konnte gezeigt werden, dass bei der in dieser Arbeit vorgestellten Lösung auch unabhängig von der tatsächlichen Ausführungsdauer der Aktivitäten Prozessanpassungen korrekt sind und das Ziel erreicht wird.

7.2 Verknüpfung der Wissenschaftsdisziplinen

Wesentliche Voraussetzung für die Realisierung eines integrierten Systems zur Prozessplanung und -ausführung ist die Verknüpfung von Workflow-Management und KI-Planung auf konzeptioneller Ebene (Abschnitt 3.1.1) und darauf aufbauend auf softwaretechnischer Ebene (Abschnitt 6.3). Verknüpfungspunkte hierfür sind einander entsprechende Entitäten der jeweiligen Disziplinen. Die zentralen Entitäten stellen dabei die Prozessdefinition im Workflow-Management und der Plan in der KI-Planung dar. In Abschnitt 3.1.1 wurde diskutiert, inwieweit sich diese Entitäten einander zuordnen lassen. Dabei wurde eine Entsprechung zwischen einer Prozessdefinition in einer graphbasierten Prozessbeschreibungssprache im Workflow-Management und einem Plan als partiell geordnete Menge von Aktionen in der KI-Planung festgestellt. Demgegenüber gibt es eine Reihe von anderen Arten von Prozessdefinitionen beziehungsweise Plänen, die keine Entsprechung in der jeweils anderen Disziplin besitzen. Dies ist im Wesentlichen in den unterschiedlichen Domänen begründet, in denen Workflow-Management beziehungsweise KI-Planung ihren Ursprung haben. Während Workflow-Management die Prozesssteuerung und -überwachung in Organisationen zum Gegenstand hat, ist ein klassisches Anwendungsfeld der KI-Planung beispielsweise eine Robotersteuerung. Während ein Roboter in der Regel Aktionen nur sequentiell ausführen kann, laufen Aktivitäten eines Prozesses in einer Organisation häufig nebenläufig.

Des Weiteren besitzen die Disziplinen Workflow-Management und KI-Planung einen anderen Fokus, wenn es um die Ausdrucksmächtigkeit der Modellierung geht. Während im Workflow-Management die Mächtigkeit der Prozessbeschreibungssprache im Vordergrund steht, befindet sich der Fokus bei der KI-Planung auf der Mächtigkeit der Sprache zur Spezifikation von Planungsproblemen. Bezüglich der Ausdrucksmächtigkeit der Modellierung lässt sich des Weiteren feststellen, dass es in soziotechnischen Systemen wie zum Beispiel einer betriebswirtschaftlichen Organisation viele unterschiedliche Einflussfaktoren gibt, die letztendlich die Struktur eines Prozesses bestimmen. Es spielen zum Beispiel Ausführungsdauern von Aktivitäten, Kosten und Wahrscheinlichkeiten eine entscheidende Rolle bei der Wahl welcher Prozess optimal ist. Hierbei ist anzumerken, dass die KI-Planung schon seit über 35 Jahren [39] in der Entwicklung ist, aber erst in den letzten Jahren eine Ausdrucksmächtigkeit erreicht hat, die eine Modellierung dieser Ein-

flussfaktoren erlaubt. Während die ersten Planer stets vollständig geordnete Pläne erzeugten und immer die Anzahl der Aktionen im Plan das Optimierungskriterium war, erlauben heutige Planer Nebenläufigkeit und alternative Ausführungspfade, berücksichtigen die Ausführungsdauer von Aktivitäten und können numerische Effekte handhaben. Erst hierdurch wurde die KI-Planung auf den Bereich des Workflow-Managements sinnvoll anwendbar.

Der unterschiedliche Fokus der Disziplinen Workflow-Management und KI-Planung auf die Prozessdefinition beziehungsweise das Planungsproblem kommt auch in den Standardisierungs-Bemühungen zum Ausdruck. Während im Workflow-Management Standards für Prozessbeschreibungssprachen entwickelt wurden, existiert in der KI-Planung zwar seit 1998 mit PDDL ein Ansatz zur Standardisierung einer Sprache zur Spezifikation von Planungsproblemen. Für die Spezifikation von Plänen existieren in der KI-Planung jedoch noch keine Standards. Dies erschwert die Einbettung unterschiedlicher Planer in Plaengine, da für jeden neuen Planer eine spezifische Abbildung seiner Ausgabe-Pläne auf BPEL4WS realisiert werden muss. Eine Standardisierung würde hier die Austauschbarkeit wesentlich erleichtern.

7.3 Anwendungsgebiete

Das in dieser Arbeit vorgestellte System zur integrierten Prozessplanung und -ausführung bietet gegenüber klassischen Workflow-Management-Systemen insbesondere in Anwendungsgebieten Vorteile, die wie folgt charakterisiert sind:

- **Stark unterschiedlich strukturierte Geschäftsfälle:** Die Möglichkeit für jeden einzelnen Geschäftsfall eine eigene, individuelle Prozessdefinition zu erzeugen kommt besonders dann zum tragen, falls die Geschäftsfälle sehr unterschiedlich strukturiert sind.

- **Prozessanpassung zur Laufzeit notwendig:** Treten häufig unerwartete Ereignisse ein, bietet die Integration einer Planungskomponente die Möglichkeit einer konsistenten, automatisierten Prozessanpassung. Insbesondere in Fällen, in denen eine Prozessanpassung sehr schnell, oder möglichst ohne manuelle Eingriffe erfolgen muss, ist eine automatisierte Prozessüberwachung und -anpassung hilfreich.

- **Dynamische Domäne:** Ändert sich die Domäne häufig, so ist eine geschäftsfall-individuelle Planung unmittelbar vor Ausführung des Prozesses vorteilhaft, da hierbei aktuellere Informationen über die Domäne vorliegen, als bei der Planung einer Prozessdefinition für einen Geschäftsfalltyp, wie beim Workflow-Management üblich. Durch eine Planung unmittelbar vor Ausführung des Prozesses werden neu verfügbare oder nicht mehr verfügbare Aktivitäten sofort berücksichtigt. Wie in Abschnitt 5.3.4 beschrieben kann eine Änderung der Domäne sogar noch während der Ausführung eines Prozesses durch eine automatisierte Prozessanpassung berücksichtigt werden.

Aufgrund des letzten Punktes bietet sich als Anwendungsgebiet für die integrierte Prozessplanung und -ausführung beispielsweise eine dienstorientierte Umgebung an. Ein wesentliches Charakteristikum einer dienstorientierten Umgebung ist, dass ständig Dienste neu angeboten, geändert oder zurückgezogen werden können. Dies entspricht einer sich ständig ändernden Domäne. Eine automatische geschäftsfallindividuelle Planung eröffnet hier die Möglichkeit neue Dienste sofort bei der Planung für den nächsten Geschäftsfall zu berücksichtigen. Aus diesem Grunde wurde Plaengine, wie in Abschnitt 6.1 erläutert, für den Einsatz in einer dienstorientierten Umgebung konzipiert.

Wesentliche Voraussetzung für eine automatische Prozessplanung ist die präzise Modellierung von Domäne und Ziel. Sind Domäne oder Ziel nicht klar strukturiert oder nur unvollständig modellierbar, so führt eine automatische Prozessplanung unter Umständen zu einer inadäquaten Prozessdefinition. Einen Kompromiss ist hier eine teilautomatische Planung wie in Abschnitt 7.4.1 beschrieben. Auf diese Weise kann unvollständig oder fehlerhaft modelliertes Domänenwissen bei der Modellierung einer konkreten Prozessdefinition noch manuell ergänzt beziehungsweise korrigiert werden. Eine weitere Einschränkung für den Einsatz eines Systems zur integrierten Prozessplanung und -ausführung kann die explizite Anforderung sein, Prozesse manuell zu planen. Gründe hierfür sind zum Beispiel gesetzliche Vorschriften oder die organisatorische Notwendigkeit die Verantwortung für die Erstellung einer Prozessdefinition eindeutig einer Person zuordnen zu können. Auch in diesen Fällen bietet sich unter Umständen eine teilautomatische Planung als Lösung an.

7.4 Weiterführende Konzepte

Dieser Abschnitt skizziert und diskutiert voneinander unabhängige Konzepte zur Erweiterung der vorgestellten integrierten Prozessplanung und -ausführung. Zunächst werden in Abschnitt 7.4.1 als Alternativen zur rein manuellen oder vollständig automatisierten Prozessplanung unterschiedliche Grade einer Teilautomatisierung spezifiziert und verglichen. Des Weiteren wird in Abschnitt 7.4.2 eine Möglichkeit zur automatischen Wiederverwendung von erfolgreich ausgeführten Prozessdefinitionen vorgestellt, die auf Case-Based-Reasoning-Techniken basiert. Abschließend zeigt Abschnitt 7.4.3 eine Möglichkeit auf, wie durch die Integration von Prozessplanung und -ausführung die Atomarität von Prozessen sichergestellt werden kann.

7.4.1 Automatisierungsgrade

Im Gegensatz zur manuellen Planung von Geschäftsprozessen, wie sie bei klassischen Workflow-Management-Systemen üblich ist, wurde in dieser Arbeit eine Möglichkeit zur vollständigen Automatisierung der Planung vorgestellt. Hieraus ergab sich eine Reihe von Vorteilen, die, wie oben diskutiert, neue Anwendungs-

gebiete für eine informationstechnische Prozessunterstützung eröffnen. Allerdings gibt es auch Argumente, die gegen eine vollständige Automatisierung der Planung sprechen:

- Eine automatische Planung kann eine Prozessdefinition ergeben, die die gewünschten Ziele nicht erreichen kann. Gründe hierfür können sein:
  - unvollständige oder fehlerhafte Modellierung der Domäne
  - unvollständige oder fehlerhafte Modellierung von Zustand oder Ziel des Geschäftsfalls
  - fehlerhafter Planungsalgorithmus

  Derartige Fehler werden unter Umständen erst bei der manuellen Prüfung einer konkreten Prozessdefinition für einen Geschäftsfall erkannt.
- Es ist denkbar, dass eine manuelle oder zumindest teilweise manuelle Planung von Prozessen explizit gewünscht ist. Gründe hierfür können sein:
  - Es existieren (beispielsweise in Behörden) gesetzliche Vorschriften, dass ein informationstechnisches System einem Mitarbeiter nicht vorschreiben darf, welche Tätigkeiten in welcher Reihenfolge zu erledigen sind. Dies wäre bei einer vollautomatischen Prozessplanung und -ausführung, die manuelle Aktivitäten umfasst, der Fall.
  - Eine Anforderung an die Prozessplanung kann sein, dass die Verantwortung für die Korrektheit einer Prozessdefinition eindeutig einer Person zugeordnet werden kann. Hierzu muss diese Person an der Planung zumindest beteiligt sein.

Zwischen einer manuellen und einer vollständig automatischen Planung lassen sich weitere alternative Automatisierungsgrade identifizieren, die es erlauben die Vorteile einer automatischen Planung zu nutzen und dennoch eine manuelle Prüfung der Prozessdefinition beziehungsweise einen manuellen Eingriff bei der Planung ermöglichen:

- **Manuelle Planung:** Prozesse werden manuell geplant.
- **Manuelle Planung mit automatischer Prüfung:** Prozesse werden manuell geplant. Vor der Ausführung findet eine automatische Prüfung statt.
- **Teilautomatische Planung:** Es findet eine interaktive Planung der Prozesse statt. Die Initiative für die Gestaltung eines Prozesses wechselt dabei zwischen Mensch und Maschine.
- **Automatische Planung mit manueller Prüfung:** Prozesse werden automatisch geplant. Vor der Ausführung findet eine manuelle Prüfung statt.
- **Automatische Planung:** Prozesse werden automatisch geplant.

Eine manuelle Planung mit automatischer Prüfung unterstützt den Prozessmodellierer, indem Inkonsistenzen in dem modellierten Prozess vom System erkannt und aufgezeigt werden. Eine rein formale Prüfung (zum Beispiel auf die Konsistenz des Kontrollflusses oder die Übereinstimmung der Datentypen beim Datenfluss) ist bei heutigen Workflow-Management-Systemen üblich. Darüber hinaus erlaubt eine semantische Spezifikation von Aktivitäten auch eine Prüfung, ob die modellierte Prozessdefinition den Zustand des Geschäftsfalls in einen Zielzustand überführt. Wesentlich ist hierbei, dass der Benutzer möglichst genaue Hinweise erhält, warum ein Zielzustand gegebenenfalls nicht erreicht wird. Beispielsweise ist eine Vorbedingung einer bestimmten Aktivität nicht erfüllt.

Eine automatische Planung mit manueller Prüfung erlaubt es die Vorteile einer schnellen, automatischen Planung zu realisieren, ohne auf die Möglichkeit zu verzichten durch eine manuelle Prüfung Fehler in der Prozessdefinition, die beispielsweise ans einer unvollständigen oder fehlerhaften Modellierung der Domäne resultieren, vor der Ausführung zu korrigieren.

Besonders interessant erscheint die in diesem Zusammenhang die teilautomatische Planung. Einerseits kann mit einer informationstechnischen Unterstützung manuell geplant werden. Dies bedeutet zum Beispiel, dass zu einer gegebenen Stelle in der Prozessdefinition automatisch Vorschläge generiert werden, welche Aktivitäten unter Berücksichtigung des Zustands des Geschäftsfalls passend eingefügt werden können. Andererseits kann für Teile der Planungsaufgabe auch auf automatisch generierte Vorschläge in Form von Teilprozessen zurückgegriffen werden. Das Potential dieser teilautomatischen Planung liegt darin, dass auch dann eine wertvolle, informationstechnische Unterstützung möglich ist, wenn die Domäne unvollständig oder teilweise fehlerhaft modelliert ist. Demgegenüber wäre eine vollständig automatisierte Planung unter diesen Voraussetzungen nicht sinnvoll.

In diesem Bereich existieren bereits verwandte Arbeiten: Kim et al. stellen in [59, 60] das Composition Analysis Tool (CAT) vor. Dieses Tool unterstützt den Benutzer bei der Modellierung von Workflows. Die Autoren definieren eine Korrektheitseigenschaft namens *well formed* für Workflows, die eine Reihe wünschenswerter Eigenschaften zusammenfasst. Das Composition Analysis Tool schlägt darauf aufbauend jeweils Ergänzungen und Modifikationen für einen gegebenen Workflow vor, die dazu dienen die *well formed* Eigenschaft zu erreichen. In [12] präsentieren Anzböck und Dustdar ein Konzept zur teilautomatischen Generierung von BPEL-Prozessen. Es wird ein der Designzeit zugeordneter Modellierungsprozess spezifiziert. Dieser beschreibt, welche Artefakte modelliert werden, um ausgehend von Standard-Dokumenten über eine schrittweise Verfeinerung in verschiedenen Diagrammformen schließlich alle Informationen in einer BPEL-Spezifikation, einer WSDL-Beschreibung der Kommunikationsendpunkte und einer WS-policy-basierten Beschreibung der relevanten Sicherheits- und Transaktionseigenschaften zusammenzuführen. Die Transformationsschritte finden dabei teilweise automatisiert statt. Weitere Ansätze zur teilautomatischen Planung von Geschäftsprozessen finden sich in [73, 50, 96].

Die oben dargestellten Automatisierungsgrade lassen sich auch geschäftsfallindividuell wählen. Denkbare Kriterien sind hier zum Beispiel der erwartete Gewinn beziehungsweise Umsatz durch den Geschäftsfall oder die zur Bearbeitung zur Verfügung stehende Zeit. Dabei beziehen sich die Automatisierungsgrade nicht nur auf die initiale Planung, sondern auch auf eine gegebenenfalls notwendige Umplanung eines Geschäftsfalls.

7.4.2 Wiederverwendung von Prozessdefinitionen

Beim dem in dieser Arbeit vorgestellten Konzept zur Integration von Planung und Ausführung, wurde bislang davon ausgegangen, dass für jeden neuen Geschäftsfall eine individuelle Planung erfolgt. Bei einer voll automatisierten Planung, Ausführung und Überwachung erscheint dies auch angemessen. Lediglich der Rechenaufwand für die Planung einer neuen Prozessdefinition muss in Kauf genommen werden. Ganz anders sieht dies bei einem geringeren Automatisierungsgrad aus. Falls ein Bearbeiter eine automatisch generierte Prozessdefinition überprüft oder sogar teilweise selbst modelliert, so bedeutet dies eine erhebliche Investition von Arbeitszeit und Kosten, die nach Bearbeitung eines Geschäftsfalls verloren ist, da Prozessdefinitionen nicht wie beim Workflow-Management wiederverwendet werden. Um diesen Verlust zu vermeiden, erweitert dieser Abschnitt das Konzept zur Integration von Planung und Ausführung um einen Ansatz zur systematischen Wiederverwendung von Prozessdefinitionen.

Eine Möglichkeit zur Realisierung einer systematischen Wiederverwendung von Prozessdefinitionen ist der Einsatz einer weiteren KI-Technik, die als fallbasiertes Schließen (Case-Based Reasoning) [62, 95] bezeichnet wird. Hierbei handelt es sich um ein Verfahren zur Problemlösung durch Analogieschluss. In einer *Fallbasis* werden Fälle als Paar von Problembeschreibung und einer zugehörigen Problemlösung abgelegt. Zur Lösung zukünftiger Probleme wird dann die Fallbasis auf ähnliche Probleme durchsucht und gegebenenfalls die zugeordnete Lösung wiederverwendet. Dabei kann die Wiederverwendung auch eine Anpassung oder Erweiterung der Lösung umfassen. Schließlich wird der Fall nach der Bearbeitung wieder als Problem-Lösungs-Paar in Fallbasis eingetragen. Somit ahmt das fallbasierte Schließen die menschliche Fähigkeit zur Erinnerung an frühere ähnliche Problemstellungen nach. Eine Anwendung des fallbasierten Schließens zur Erweiterung der Techniken des Workflow-Managements wird von Madhusudan und Zhao in [71] vorgestellt. Die Übertragung des fallbasierten Schließens auf das in dieser Arbeit vorgestellte Konzept zur Integration von Planung und Ausführung, stellt sich wie folgt dar. Jeder Fall der Fallbasis ist ein Paar aus einem Planungsproblem, dass die Problembeschreibung darstellt, und einer Prozessdefinition, die die Problemlösung darstellt. Wird ein neuer Geschäftsfall erfasst, so wird vor der Planung geprüft, ob für das Planungsproblem bereits ein Fall in der Fallbasis existiert. Falls ein entsprechender Fall gewunden wird kann die zugeordnete Prozessdefinition auf den neuen Geschäftsfall angewendet werden. Eine Planung ist nur noch erforderlich, wenn kein passender Fall in der Fallbasis gefunden wird.

Eine Problematik bei diesem Vorgehen ist, dass die Planungsprobleme sehr spezifisch sind, da jeder Geschäftsfall individuell geplant wird. Insbesondere nichtfunktionale Anforderungen schränken die Möglichkeiten ein, ein identisches Planungsproblem in der Fallbasis zu finden. Tritt beispielsweise der Geschäftsfall ein, dass eine Versicherung einen Schadensfall mit einem Volumen von 1422 € zu bearbeiten ist, so stellt sich die Frage, ob dafür eine Prozessdefinition wiederverwendet werden kann, die für einen Schadensfall mit einem Volumen von 1513 € geplant wurde. Dabei ist es durchaus denkbar, dass zum Beispiel alle Schadensfälle mit einem Volumen über 1500 € genauer geprüft werden müssen als kleinere Schadensfälle. Ansätze zur Lösung dieser Problematik sind eine Abstraktion des Planungsproblems und der dazugehörigen Prozessdefinition, sowie die Möglichkeit zur Suche nach ähnlichen Planungsproblemen anstelle gleicher Planungsprobleme. Die Herausforderung hierbei ist die Bestimmung einer angemessenen Abstraktion, beziehungsweise die adäquate Festlegung eines Ähnlichkeitsmaßes.

7.4.3 Transaktionale Prozesse

In bestimmten Anwendungsgebieten sind transaktionale Eigenschaften von Prozessen gewünscht. Ein Beispiel hierfür ist die Umbuchung eines bestimmten Betrages von einem Konto auf ein anderes Konto. Hierbei ist die Atomarität des Prozesses bestehend aus den Aktivitäten Abbuchung und Gutschrift erforderlich. Falls die Gutschrift fehlschlägt, muss die Abbuchung wieder zurückgenommen werden. Die Transaktionalität von Geschäftsprozessen wurde intensiv im Kontext von Datenbankmanagementsystemen untersucht [9, 33, 66, 88, 87]. Aktuelle Forschungsansätze [37, 69, 80, 93], greifen diese Problematik im Hinblick auf Web-Services und zusammengesetzte Dienste wieder auf.

Die Möglichkeit zur automatischen, zielgerichteten Planung von Prozessdefinitionen eröffnet eine neue Möglichkeit die Atomarität von Prozessen zu erreichen. Anstelle des typischen Ansatzes eine kompensierende Aktivität für jede einzelne bisher ausgeführte Aktivität auszuführen, kann ein kompensierender Prozess durch eine erneute Planung automatisch ermittelt werden. Bei einem Fehlschlagen einer Aktivität wird zunächst, wie in Kapitel 4 beschrieben, eine Umplanung angestoßen, die nach einer neuen Prozessdefinition sucht, um das Ziel zu erreichen. Kann eine solche Prozessdefinition nicht mehr gefunden werden, müssen zur Gewährleistung der Atomarität alle bisherigen Zustandsänderungen durch den Prozess zurückgenommen werden. Hierzu wird der Ausgangszustand als neues Ziel definiert und ausgehend vom aktuellen Zustand erneut geplant. Eine Prozessdefinition die für diese Planungsaufgabe gültig ist, überführt den aktuellen Zustand wieder in den Ausgangszustand. Diese kompensierende Prozessdefinition besteht nicht notwendigerweise aus kompensierenden Aktivitäten für jede einzelne der bisher ausgeführten Aktivitäten. Stattdessen kann die kompensierende Prozessdefinition eine komplett andere Struktur besitzen.

Betrachten wir als Beispiel hierfür den LKW-Transport eines Gutes von Ort A zu Ort D. Angenommen es existieren Straßen zwischen den Orten A und B, B

und C, C und D, sowie eine Einbahnstraße von C nach A. Eine Prozessdefinition zur Lösung des Problems ist folglich eine Sequenz der drei Aktivitäten Transport von A nach B, Transport von B nach C und schließlich Transport von C nach D. Nehmen wir nun an, dass dieser Prozess atomar ausgeführt werden soll und bei seiner Ausführung die Verbindung zwischen C nach D nach erreichen des Ortes C überraschenderweise nicht mehr zur Verfügung steht. Zur Wahrung der Atomarität muss nun der LKW zum Ort A zurück gebracht werden. Eine erneute Planung mit dem Ziel A ergibt eine Prozessdefinition mit einer Aktivität Transport von C nach A unter Nutzung der direkten Einbahnstraßen-Verbindung. Nach dem Konzept der kompensierenden Aktivitäten für jede bisher ausgeführte Aktivität hätte der Rücktransport über die gleichen Verbindungen wie auf dem Hinweg, das heißt über den Ort B, stattgefunden und die Abkürzung von C direkt nach A bliebe ungenutzt.

Kapitel 8

Verwandte Arbeiten

Dieses Kapitel stellt verwandte Arbeiten vor und grenzt sie von der in dieser Arbeit vorgestellten integrierten Prozessplanung und -ausführung ab. Hierzu werden die verwandten Arbeiten in vier Gebiete unterteilt: In Abschnitt 8.1 werden Konzepte zur dynamischen Adaption von Workflows vorgestellt. Abschnitt 8.2 beschreibt mit dem Case-Handling und dem Serviceflow zwei Ansätze die Workflow-Management-Techniken in der Richtung erweitern, dass die Prozesssteuerung nicht nur durch den Kontrollfluss festgelegt ist, sondern sich stärker an vorhandenen Daten beziehungsweise dem Kunden orientiert. In Abschnitt 8.3 werden Ansätze vorgestellt, die die Planung von Geschäftsprozessen automatisieren. Schließlich werden in Abschnitt 8.4 Konzepte zur automatisierten Prozessüberwachung vorgestellt.

8.1 Dynamische Adaption von Workflows

In der Einleitung dieser Arbeit wurde dargestellt, dass bei der Ausführung von Prozessen unerwartete Ereignisse auftreten können, die eine Anpassung der Prozessdefinition notwendig machen. Werden Workflow-Management-Systeme eingesetzt, um die Prozessausführung zu unterstützen, müssen daher technische Möglichkeiten gefunden werden, Änderungen von Prozessdefinition zur Laufzeit von Prozessen durchzuführen und diese Änderungen konsistent auf die laufenden Prozessinstanzen zu übertragen. Dies wird als *dynamische Adaption von Workflows* bezeichnet. Entsprechende Konzepte werden in diesem Abschnitt vorgestellt. Sie beschreiben technische Realisierungsmöglichkeiten und spezifizieren darüber hinaus Kriterien für die Prüfung einer formalen, strukturellen Korrektheit der Anpassung. Im Kontext der in Kapitel 4 beschriebenen automatisierten Prozessanpassung konzentrieren sich diese Arbeiten auf den letzten Schritt bei der Umplanung eines Geschäftsfalls. Die vorhergehenden Schritte, wie das Überwachen der Prozessinstanz, die Auslösung der Umplanung und die Planung einer neuen Prozessdefinition bleiben bei diesen Ansätzen eine manuelle Aufgabe. Da diese Ansätze in erster Linie das in dieser Arbeit vorgestellte Konzept motivieren, aber bezüglich des Lö-

sungswegs wenig Überschneidungen existieren, werden die meisten Ansätze nur kurz skizziert.

Shazia Sadiq identifiziert in [91, 92] unterschiedliche Arten von Prozessänderungen. Darauf aufbauend wird generisches Konzept für Anpassungen von Prozessdefinitionen entwickelt, die auf aktuelle Prozessausführungen übertragen werden.

Ellis, Keddara und Rozenberg untersuchen in [34, 35] dynamische, strukturelle Änderungen von Prozessdefinitionen auf der Basis von Flow Nets. Flow Nets sind Petri-Netze mit einer dedizierten Eingabestelle und Ausgabestelle. Es wird eine bestimmte Klasse von Prozessänderungen names *synthetic-cut-over* identifiziert, die die Korrektheit der resultierenden Prozessdefinition sicherstellt. Die Idee hierbei ist, dass sowohl die ursprüngliche als auch die modifizierte Region einer Prozessdefinition Bestandteil der neuen Prozessdefinition werden. Auf diese Weise können laufende Prozessinstanzen entsprechend der ursprünglichen Region der Prozessdefinition gesteuert werden, falls die Bearbeitung schon zu weit fortgeschritten ist. Auf diese Weise wird sichergestellt, dass die Änderungen sich ganz oder gar nicht auf einer Prozessinstanz auswirken.

Van der Aalst et al. stellen in [1, 3] Prozessdefinitionen als WF-nets dar. WF-nets sind eine eingeschränkte Form von Stellen/Transitions-Netzen. Erfüllen sie darüber hinaus bestimmte Korrektheitseigenschaften, so werden sie als *sound* [1] bezeichnet. Dieser Formalismus wird benutzt, um Aussagen über die Korrektheit von Änderungen einer Prozessdefinition zur Laufzeit zu treffen.

MILANO ist ein auf Petri-Netzen basierender Ansatz, der von Agostini und De Michelis in [5, 6] vorgestellt wird. Ziel ist die kontrollierte Änderungen von Prozessdefinitionen unter Berücksichtigung laufender Prozesse.

In WASA_2 [105, 109] werden von Mathias Weske et al. Prozessdefinitionen und Prozessausführungen als Graphen modelliert. Anpassungen von Prozessdefinitionen können auf laufende Prozessausführungen übertragen werden. Dabei entscheidet der Ausführungszustand der einzelnen Aktivitäten darüber, welche Änderungen noch übertragbar sind, ohne die Korrektheit der Ausführung zu gefährden.

Mit WIDE [21] werden von Fabio Casati et al. formale, strukturelle Anforderungen an die Korrektheit einer Prozessdefinition spezifiziert. Diese Korrektheitseigenschaften werden dann auf die Anpassung von Prozessdefinitionen zur Laufzeit übertragen.

ADEPT_{flex} [83] ist ein Reichert und Dadam entwickelter Formalismus zur dynamischen Anpassung von Prozessdefinitionen, die in der Sprache ADEPT spezifiziert sind. Zur Anpassung von Prozessdefinitionen sind nur wohldefinierter Transformationsoperationen zugelassen. Hierdurch wird die Wahrung von strukturellen Korrektheitseigenschaften sichergestellt.

Das von Kradolfer und Geppert vorgestellte TRAMs [63] basiert auf einem Versionierungskonzept für Prozessdefinitionen. Der Übergang zwischen Versionen geschieht wie bei ADEPT_{flex} nur auf Basis wohldefinierte Transformationsoperationen.

Diese und weitere Forschungsansätze [49, 53, 56, 76] für die Prüfung der Kor-

rektheit der Anpassung von Prozessdefinitionen zur Laufzeit, beschränken sich auf die Prüfung einer rein formalen, strukturellen Korrektheit einer Anpassung. [84] Die Überwachung von Prozessen sowie die Planung und Modellierung neuer Prozessdefinitionen zur Berücksichtigung von unerwarteten Ereignissen bleibt eine manuelle Aufgabe. Im Gegensatz hierzu unterstützt und automatisiert die in dieser Arbeit vorgestellte integrierte Prozessplanung und -ausführung alle Phasen einer Prozessanpassung von der Überwachung der Prozessinstanz, über die Planung einer neuen Prozessdefinition bis hin zu Anpassung der Prozessinstanz.

8.2 Daten- und kundenorientierte Prozesssteuerung

In diesem Abschnitt werden mit dem *Case-Handling* und dem *Serviceflow* zwei zum Workflow-Management alternative Ansätze zur informationstechnischen Unterstützung von Geschäftsprozessen beschrieben, die den Fokus bei der Prozesssteuerung vom Kontrollfluss auf die zu verarbeitenden Daten, beziehungsweise auf den Kunden verschieben.

Case-Handling Das Case-Handling wird van der Aalst, Weske und Grünbauer in [2] vorgestellt. Ebenso wie das Workflow-Management dient das Case-Handling der Steuerung, Überwachung und Dokumentation von Geschäftsprozessen. Damit besitzt das Case-Handling gemeinsame Ziele mit dem Workflow-Management, wobei allerdings eine andere Vorgehensweise gewählt wird. Der zentrale Unterschied liegt in dem Übergang von einer kontrollflussbasierten zu einer datenbasierten Prozesssteuerung. Betrachtet man in diesem Zusammenhang die Existenz von Datenobjekten als Vorbedingung für die Ausführung von Aktivitäten, so bildet das Case-Handling einen konzeptionellen Zwischenschritt von vordefinierten Prozessdefinitionen im Workflow-Management zu semantisch beschriebenen Aktivitäten in der KI-Planung. Zusammenfassend ist das Case-Handling der Hintergrund für die Idee zur Integration von Prozessplanung und -ausführung in dieser Arbeit. Im Folgenden werden die wesentlichen Aspekte vorgestellt, in denen sich das Case-Handling vom Workflow-Management unterscheidet.

Case-Handling und Workflow-Management unterscheiden sich grundlegend in der Art und Weise, wie der Ablauf eines Prozesses gesteuert wird. Während beim Workflow-Management die Steuerung ausschließlich über den Kontrollfluss erfolgt, ist bei der Case-Handling diese *Steuerung datenbasiert*. Die Entscheidung, ob eine Aktivität ausgeführt werden kann, wird auf Basis der vorhandenen Datenobjekte, anstatt der ausgeführten Aktivitäten getroffen. Dieser Paradigmenwechsel ermöglicht eine einfachere und natürlichere Prozessmodellierung. Bestehen zwischen diesen Aktivitäten Abhängigkeiten in dem Sinne, dass für die Ausführung einer Aktivität Informationen oder Ergebnisse einer anderen Aktivität benötigt werden, so wird dies als Datenabhängigkeit bezeichnet. Beim Workflow-Management werden Datenabhängigkeiten als Datenfluss modelliert. Bei der Modellierung des Kontrollflusses werden Datenabhängigkeiten zwar berücksichtigt, aber nicht expli-

zit modelliert. Im Gegensatz dazu werden beim Case-Handling alle Datenabhängigkeiten explizit und direkt modelliert. Das heißt, die Bedingung, dass bestimmte Informationen vor dem Start einer Aktivität vorhanden sein müssen, wird Bestandteil der Prozessdefinition. Dagegen findet keine Modellierung eines Kontrollflusses statt. Neben einer einfachen Prozessmodellierung wird damit auch ein höherer Grad an Parallelität in der Prozessausführung möglich: Während beim Workflow-Management eine Aktivität nicht starten kann, bis alle dem Kontrollfluss nach vorhergehenden Aktivitäten beendet sich, kann beim Case-Handling eine Aktivität starten, sobald alle benötigten Daten vorhanden sind.

Beim Case-Handling können Datenobjekte auch vor und nach der Ausführung der in der Prozessdefinition dafür vorgesehenen Aktivität eingegeben, modifiziert und gelesen werden. Dies ist insbesondere vor dem Hintergrund des Business Process Redesign [51] von großer Bedeutung: Bei dem Übergang von einer arbeitsteilig organisierten Unternehmung zu prozessorientierten Organisationsformen ergibt sich ein neues Berufsbild eines Benutzers. An die Stelle des Spezialisten für ein bestimmtes, klar abgegrenztes Aufgabengebiet tritt der Generalist. Er hat ein weitaus breiteres Betätigungsfeld, das sich nicht mehr klar von dem der anderen Mitarbeitern eines Unternehmens abgrenzen lässt. Vielmehr tragen alle Prozessbeteiligten gemeinsam die Verantwortung für den Prozess und die Grenzen der Zuständigkeit werden unscharf. Der *freie Zugriff auf alle Informationen*, die mit der Bearbeitung eines Prozesses in Zusammenhang stehen, ist charakteristisch für das Case-Handling. Demgegenüber bieten Workflow-Management-Systeme für eine Aktivität immer nur die zum Zeitpunkt der Modellierung als relevant erachteten Dokumente zur Bearbeitung oder Einsichtnahme an. Es ist zwar unter Umständen möglich, das System zu umgehen und auf anderen Wegen an die gewünschten Informationen zu gelangen, jedoch ist dies nicht vorgesehen und wird nicht vom Workflow-Management-System unterstützt. Insofern spiegelt Workflow-Management im Umgang mit Informationen die traditionelle Sicht arbeitsteilig organisierter Unternehmen wieder, in denen der Arbeiter nur mit den für seine Aufgabe vorhersehbar notwendigen Informationen versorgt wird. Dieses Vorgehen wird auch als *Information Tunneling* [2] bezeichnet. In dieser Denkweise hat der Benutzer als Spezialist weder die Kompetenz noch den Überblick um weitere Informationen sinnvoll zu verwerten. Demgegenüber ist für einen Generalisten der freie Zugriff auf alle den Prozess betreffenden Daten Voraussetzung für eine effektive Arbeit.

In diesem Zusammenhang ist auch die *Trennung von Distribution und Autorisierung* zu sehen, die das Case-Handling vom Workflow-Management unterscheidet. Beim Workflow-Management können nur die Aktivitäten gesehen und ausgeführt werden, die auf der Arbeitsliste des Bearbeiters liegen. Es wird nicht unterschieden zwischen Aktivitäten, die er ausführen darf und Aktivitäten die er plangemäß ausführen soll. Das Konzept des Case-Handlings ist hier flexibler: Für jede Aktivität ist festgelegt, welche der Ausführungsrechte der Bearbeiter besitzt. Es gibt beim Case-Handling aber keine vordefinierte Arbeitsliste, die alle Aktivitäten enthält, auf denen zurzeit die Ausführung einer dieser Operationen möglich

ist. Vielmehr stellt der Bearbeiter selbst Anfragen, um die Aktivitäten zu finden, die er als nächstes bearbeiten möchte. Entsprechend seiner routinemäßigen Aufgaben wird er dabei häufig die gleichen Anfragen stellen. Die Menge der ermittelten Aktivitäten ist mit denen vergleichbar, die sich auf der Arbeitsliste eines Workflow-Management-Systems befinden. Die Stärke des Konzepts liegt in seiner Flexibilität: Wenn es der Einzelfall erfordert, kann der Bearbeiter Operationen auf Aktivitäten ausführen, die standardmäßig nicht in sein Aufgabengebiet fallen. Dazu muss er lediglich über eine entsprechend formulierte Anfrage die Aktivität finden und die benötigten Rechte besitzen, um die gewünschte Operation auszuführen. Ein Bearbeiter kann beim Case-Handling weitaus mehr Rechte besitzen, als er zur Ausführung seiner routinemäßigen Aufgaben benötigt. Hierdurch erhält er den Freiraum, seine Arbeit flexibel und der Situation angepasst zu gestalten.

Serviceflow Ralf Klischewski und Ingrid Wetzel stellen mit *Serviceflow* [61, 111] ein neues Konzept für eine informationstechnische Prozessunterstützung vor, bei dem der Kunde im Mittelpunkt organisationsübergreifender Prozesse steht. Ähnlich dem Case-Handling ist der freie und integrierte Zugriff auf prozessrelevante Daten ein zentraler Aspekt. Ein Kunde, der eine Dienstleistung in Anspruch nimmt, die von mehreren Leistungsanbietern abhängt, soll keine Angaben mehrfach machen müssen. Hierzu wird das Konzept des Servicepunkts eingeführt. Ein *Servicepunkt* ist mit den Anwendungen eines Unternehmens verbunden und definiert den Ort, an dem Kunde und Unternehmen aufeinander treffen. Ein Serviceflow ist darauf aufbauend eine Aneinanderreihung von Servicepunkten. Dabei sollen aus der Perspektive eines sich zeitlich und räumlich bewegenden Kunden Serviceleistungen nahtlos und räumlich transparent angeboten werden. Die Angaben, die der Kunde an einem Servicepunkt macht, werden in einem sogenannten *Servicefloat* gespeichert. Der Servicefloat ist einem Serviceflow zugeordnet. Auf ihn kann von jedem Servicepunkt des Serviceflows aus zugegriffen werden. Jedem Servicepunkt sind Vor- und Nachbedingungen zugeordnet, die eine korrekte Kommunikation mit dem Kunden spezifizieren.

Der Verlauf eines Prozesses ist beim Serviceflow stark vom Verhalten des Kunden abhängig und kann jederzeit von ihm grundlegend beeinflusst werden. Dies ist ein zentraler Unterschied zum Workflow-Management. Die Idee ist, dass Kunden sehr unterschiedliche Anforderungen an ein Unternehmen stellen. Es ist daher kaum praktikabel, wie bei Workflow-Management-Systemen üblich, alle möglichen Abläufe im Voraus zu antizipieren und zu modellieren. Der Lösungsansatz beim Serviceflow ist daher die Identifikation zentraler, wiederkehrender Abläufe und darauf aufbauend die Modellierung entsprechender Servicepunkte. Anschließend wird ein Muster für eine typische Abfolge der Servicepunkte definiert. Von diesem Muster kann jedoch im Gegensatz zum Workflow-Management bei Bedarf abgewichen werden. In dieser Arbeit wird in Abschnitt 3.2 mit der kundenindividuellen Planung von Prozessdefinition ein alternativer Lösungsansatz zur Berücksichtigung sehr unterschiedlicher Kundenanforderungen vorgeschlagen.

8.3 Automatisierte Prozessplanung

In diesem Abschnitt werden Ansätze vorgestellt, welche die Planung von Prozessen automatisieren. Einige der Ansätze erweitern klassische Workflow-Management-Techniken, um die Möglichkeit zur automatisierten Planung von Prozessdefinition. Andere Ansätze haben ihren Ursprung in dienstbasierten Architekturen und verfolgen das Ziel der automatisierten Komposition von Diensten. Ein Aspekt dieser Komposition ist die Planung eines Prozesses, der die Ausführungsreihenfolge der komponierten Dienste spezifiziert.

In [22] wird von Chun, Atluri und Adam ein Projekt der Rutgers University vorgestellt, das die KI-Technik des *Hierarchical Task Network Planning* auf die Planung von Unternehmensprozessen überträgt. Dieser Ansatz basiert auf einer hierarchischen Dekomposition der Domäne. Ausgangspunkt ist ein abstrakter, initialer Plan, der schrittweise verfeinert wird, bis der Plan nur noch atomare Aktionen enthält. Voraussetzung hierfür ist eine Bibliothek vordefinierter Pläne, die für die Zerlegung nicht-atomarer Aktionen verwendet werden. Dieser Ansatz wird in [13] um ein Konzept namens DWFCM (Decentralized Workflow Change Management) für ein dezentralisiertes Management von Workflow-Änderungen erweitert. DWFCM basiert auf Ontologien von Diensten und Domänenwissen, die zur Formulierung von Workflow-Integrationsregeln benutzt werden. Diese Workflow-Integrationsregeln erlauben eine erneute Planung zur Anpassung von Unternehmensprozessen. Dieser Ansatz wird in [23] auf dienstbasierte Architekturen erweitert. Im Gegensatz zu dem in dieser Arbeit vorgestellten Ansatz setzt DWFCM die Existenz einer Bibliothek vordefinierter Pläne voraus. Daher müssen alle möglichen (Teil-)Prozesse vor der Planung bekannt sein. Dies schränkt die möglichen Anwendungsszenarien stark ein.

Das PLM_{flow}-System [115] ist ursprünglich als Teil eines umfangreichen IBM-Projekts namens *Product Lifecycle Management* von Liangzhao Zeng et al. entwickelt worden. Ziel von PLM_{flow} ist die automatische Komposition von Geschäftsprozessen und der Verzicht auf vordefinierte Prozessdefinitionen. Grundlage für die automatische Komposition bilden eigenständige Geschäftsregeln, welche die Funktionalität von Aktivitäten beschreiben. Die Argumentation hierbei ist, dass einzelne Aktivitäten mit ihren Eingabe- und Ausgabeparametern zwar auch bei klassischen Workflow-Management-Systemen wiederverwendet werden können, die Funktionalität der Aktivitäten aus dem Prozess jedoch nicht ersichtlich ist. Daher werden Geschäftsregeln modelliert, die die Möglichkeiten zur Komposition von Aktivitäten festlegen. Die Planung eines Geschäftsprozesses basiert auf Inferenz in Form von Rückwärts- und Vorwärtsverkettung dieser Geschäftsregeln.

Das DY_{flow}-System [114] ist eine Weiterentwicklung von PLM_{flow} und überträgt die Komposition von Aktivitäten auf die Komposition von Web-Services. Eine wesentliche Einschränkung von PLM_{flow} und DY_{flow} ist, dass Aktivitäten keine negierten Effekte anderer Aktivitäten haben können. Dieses vereinfacht einerseits die Planung stark, da kausale Verknüpfungen nicht geschützt werden müssen. Andererseits sind viele Problemstellungen folglich nicht formulierbar. Des Weiteren

werden die Ausführungsdauer und die Kosten von Aktivitäten nicht berücksichtigt. Daher sind die Einsatzmöglichkeiten dieses Ansatzes zur Planung von Geschäftsprozessen eingeschränkt.

Ausgehend von einer Prozessstruktur eines zusammengesetzten Dienstes werden in [113] von Liangzhao Zeng et al. Techniken der Linearen Programmierung eingesetzt, um auf optimale Weise an konkrete Dienstimplementierungen zu binden. Die Argumentation hierbei ist, dass nur die gleichzeitige Berücksichtigung der Qualitätsmerkmale aller Dienste eines zusammengesetzten Dienstes die Erreichung eines globalen Optimums ermöglicht. Einen ähnlichen Ansatz verfolgen Stanley Su et al. in [99]. Sie stellen eine Infrastruktur für eine dienstbasierte Architektur vor, die Dienstaufrufe in einer Prozessdefinition dynamisch an verteilte Dienstimplementierungen bindet. Hierzu werden Dienste in einem *Broker Server* registriert. Dieser bestimmt bei einem Dienstaufruf die passenden Dienste und wählt auf Basis einer Kosten/Nutzen-Abwägung den besten Dienst aus.

In [11] wird von Stanislaw Ambroszkiewicz die auf XML basierende, vollständig deklarative Sprache *enTish* zur Beschreibung und Komposition von heterogenen Diensten in offenen und verteilten Umgebungen vorgestellt. Aufbauend auf dieser Sprache wird ein Protokoll spezifiziert und prototypisch implementiert, das den Nachrichtenaustausch zwischen Agenten und Web Services bei einer Komposition beschreibt. Basierend auf der Beschreibung eines Ausgangszustands und eines Ziels kommen KI-Planungstechniken zum Einsatz, um für eine konkrete Anfrage Dienste individuell zusammenzufügen.

Narayanan und McIlraith präsentieren in [74] ein Konzept, dass semantische Beschreibungen von Diensten basierend auf DAML-S [103] nutzt, um verschiedene Aspekte, wie Simulation, Test, Verifikation und Komposition von Diensten informationstechnisch zu unterstützen. Zur Komposition von Diensten kommt eine prädikatenlogische Sprache zum Einsatz, deren Semantik auf einer Teilmenge von DAML-S [103] basiert. Darauf aufbauend werden KI-Planungstechniken eingesetzt, um passende Prozessdefinitionen zu planen.

Therani Madhusudan et al. stellen in [70, 71] einen Ansatz vor, der zu einem gegebenen Geschäftsfall automatisch eine passende Prozessdefinition finden kann. Hierzu wird die Verknüpfung einzelner Arbeitsschritte oder Teilgraphen in ein fallbasiertes Konzept zur Wiederverwendung von Prozessen integriert. Zur Bearbeitung von Geschäftsfällen werden ähnliche, bereits bearbeitete Geschäftsfälle gesucht und die zugehörigen Prozessdefinitionen für den aktuellen Fall übernommen. Hierbei wird auf Case-Based Reasoning-Techniken [62, 95] zurückgegriffen. Wird kein passender Geschäftsfall gefunden, werden KI-Planungstechniken eingesetzt, um bestehenden Arbeitsschritte oder Teilgraphen bedarfsgerecht zusammenzufügen.

In [18] stellen Blythe, Deelman und Gil einen Ansatz zur automatischen Berechnung von wissenschaftlichen Aufgaben auf einem Grid vor. Hierbei werden KI-Planungstechniken zur Generierung von Workflows eingesetzt, die die Reihenfolge der notwendigen Berechnungen festlegen. Die Aktivitäten dieser Workflows werden von der *Pegasus*-Komponente [27] auf das Grid abgebildet. Der für die KI-

Planung benötigte Ausgangszustand wird von einem *Current State Generator* aus den vorhandenen Datenobjekten abgeleitet. Zukünftige Arbeiten [46] richten sich auf die Berücksichtigung von verfügbaren Ressourcen und Anwendungskomponenten bei der Planung. Die Notwendigkeit einer Möglichkeit zur Umplanung von Prozessen wird in [18] angesprochen und motiviert, aber keine Lösung vorgestellt.

In [85, 86] schlagen Moreno und Kearney die Anwendung von Contingent-Planern in bestehenden Workflow-Management-Systemen vor. Konkret wird der Cassandra-Planer als Planungskomponente mit dem System *Customer Oriented System for the Management of Special Services (COSMOSS)* integriert. Die Autoren argumentieren, dass der Cassandra-Planer besonders für die Planung von Geschäftsprozessen geeignet ist. Besonderes Augenmerk wird auf die Techniken zur Modellierung der Domäne gerichtet, da insbesondere die Modellierung der Domäne für Benutzer, die wenig Hintergrundwissen über KI-Planung haben, schwierig und fehleranfällig ist. Ziel ist eine möglichst einfache und intuitive Modellierung.

Eine Architektur zur Komposition von Web-Services für konkrete Anfragen wird von Lazovik, Aiello and Papazoglou in [65] vorgestellt. Diese Architektur baut auf dem *Planning as Model Checking*-Ansatz [17] auf und verwendet den MBP-Planer. Anfragen in der XML-Service-Request-Language [7] werden mit den Kommunikationsprotokollen der beteiligten Web-Services zu einem Ausführungsplan in Form eines deterministischen Zustandsautomaten verknüpft. Ziele werden dabei in der Sprache *EaGLe* [81] formuliert. Zur Berücksichtigung des undeterministischen Verhaltens von Web-Services und der unvollständigen Beobachtbarkeit alternieren Planungs- und Ausführungsphasen. Sobald bei der Ausführung neue Informationen gewonnen werden, wird wieder in eine Planungsphase gewechselt und ein neuer Plan generiert. Die Planungsphasen dienen hierbei allerdings nur zu Auswahl von im Ausführungsplan festgelegten Alternativen und nicht der Behandlung von unerwarteten Ereignissen. Entsprechend müssen alle möglichen Zustände im Voraus bekannt sein.

In [112] stellt Dan Wu et al. einen Ansatz zur automatischen Komposition von Web-Services vor. Hierzu werden semantische DAML-S-Beschreibungen [103] von Web-Services in ein Format für den SHOP2-Planer transformiert. SHOP2 [75] ist ein domänenunabhängiger Hierarchical Task Network Planer, der aus diesen Beschreibungen einen Plan generiert, der als zusammengesetzter Dienst interpretiert wird. Die semantische Entsprechung zwischen den DAML-S-Beschreibungen und dem generierten Plan wird gezeigt und prototypische Implementierung geliefert.

Das *ACT!*-Planungssystem, dass von Beckstein und Klausner in [15] und [16] vorgestellt wird, ist Teil einer Architektur für Workflow-Management-Systeme, die auf eine intelligente Behandlung von Ausnahmesituationen bei der Prozessausführung abzielt. Hierzu werden Reparaturtechniken für Pläne aus der Künstlichen Intelligenz eingesetzt. Diese erlauben die Berücksichtigung von nebenläufigen Ausführungen und nicht-deterministischen Aktivitäten. Grundlage für den Einsatz von KI-Techniken bilden semantisch beschriebene Prozessdefinitionen. Neben der Behandlung von Ausnahmesituationen dienen diese semantischen Informationen der Unterstützung des Benutzers bei der Prozessmodellierung. Prozessdefi-

nitionen werden in einem Dienstverzeichnis abgelegt. Auf das Dienstverzeichnis greift sowohl die Prozess-Engine als auch die Planungskomponente zu, die gegebenenfalls erfolgreich ausgeführte Prozessdefinitionen oder Prozessfragmente bei der Planung wiederverwendet.

Einige der obigen Ansätze verwenden zur Erzeugung von Prozessbeschreibungen Planungsalgorithmen aus der Künstlichen Intelligenz, während andere wie zum Beispiel DY_{flow} auf speziell entwickelten Algorithmen beruhen. Bezüglich der Reaktion auf unerwartete Ereignisse beschreiben nur die in [65] und [13] vorgestellten Ansätze Konzepte zur automatischen Anpassung von Prozessdefinitionen zur Laufzeit. Hingegen werden bei keinem der vorgestellten Ansätze laufende Aktivitätsinstanzen berücksichtigt, wie es bei der in dieser Arbeit vorgestellten integrierten Prozessplanung und -ausführung der Fall ist.

8.4 Automatisierte Prozessüberwachung

In diesem Abschnitt werden Konzepte zur automatisierten Prozessüberwachung vorgestellt. Im Gegensatz zur der in Kapitel 4 beschriebenen automatisierten Prozessanpassung wird hier mit der Überwachung nur der erste Schritt einer Prozessanpassung betrachtet. Eine automatisierte Anpassung einer Prozessdefinition findet nicht statt.

Akram et al. beschreiben in [8] einen Algorithmus zum Management von dynamischen, dienstbasierten Umgebungen. Die Grundlage für eine gezielte Organisation und Auswahl von Diensten bilden dabei Ontologien. Es wird eine Request-Broker-Architektur dargestellt, die der Verwaltung von Änderungen in der Domäne dient. Zu diesen Änderungen gehören beispielsweise neu registrierte Dienste oder nicht mehr verfügbare Dienste. Zur Überwachung der Ausführung von Diensten werden Agenten eingesetzt, die den Zustand der Dienste beobachten und gegebenenfalls andere Komponenten der Request-Broker-Architektur benachrichtigen, um auf Ereignisse zu reagieren. Hierzu werden periodisch Anfragen an die Dienste gesendet.

In [48] wird von Greiner und Rahm das *WebFlow*-System vorgestellt, das die automatische Überwachung von Ausführungsbedingungen erlaubt. Dabei sollen insbesondere die Herausforderungen kooperativer Workflows berücksichtigt werden, bei denen Aktivitäten durch Dienste unterschiedlicher Organisationseinheiten ausgeführt werden. Ziel ist die Robustheit und Flexibilität kooperativer Workflows. Die Unterstützung bei der Erkennung von Ausnahmesituationen basiert sowohl auf expliziten Fehlernachrichten, als auch auf Timeouts und semantischer Prüfung von Zwischen- und Endergebnissen. Eine Ausnahmebehandlung in Form einer automatischen Umplanung ist nicht vorgesehen. Hierfür können jedoch einfache Ereignis-Bedingungs-Aktions-Regeln formuliert werden.

Zeng et al. stellen in [116] ein Konzept zur Prozessüberwachung bei zusammengesetzten Diensten vor. Ein zentraler Aspekt hierbei ist die Trennung von Geschäftslogik und Ausnahmebehandlung. Ziel ist die Wiederverwendung von Ver-

fahren zur Ausnahmebehandlung bei verschiedenen zusammengesetzten Diensten. Zur Laufzeit werden automatisch die Geschäftslogik eines konkreten zusammengesetzten Dienstes mit allgemein formulierten Verfahren zur Ausnahmebehandlung integriert. Hierdurch entstehen Prozessdefinitionen, die sowohl den erwarteten Ablauf eines Prozesses als auch die Erkennung und Behandlung von Ausnahmesituationen eines Prozesses spezifizieren.

Kapitel 9

Zusammenfassung und Ausblick

9.1 Zusammenfassung

Ausgehend von Schwachpunkten klassischer Workflow-Management-Systeme im Hinblick auf eine adäquate, informationstechnische Unterstützung von Geschäftsprozessen wurde in dieser Arbeit ein Konzept zur integrierten Prozessplanung und -ausführung vorgestellt. Durch Integration einer Planungskomponente kann für jeden einzelnen Geschäftsfall eine eigene Prozessdefinition erzeugt werden. Somit lassen sich individuelle Besonderheiten besser berücksichtigen. Des Weiteren wird eine automatische Prozessüberwachung und -anpassung möglich.

Zunächst wurden in Kapitel 2 die Grundlagen der Wissenschaftsdisziplinen Workflow-Management und KI-Planung erläutert, die für die Integration von Prozessplanung und -ausführung relevant sind. In Kapitel 3 wurde aufgezeigt, wie die Prozessplanung durch den Einsatz von KI-Planungstechniken automatisiert und mit der Prozessausführung und -überwachung integriert werden kann. Ausgangspunkt für die Integration auf konzeptioneller Ebene war die Identifikation und Analyse einander entsprechender Entitäten der Disziplinen Workflow-Management und KI-Planung. Darauf aufbauend wurde ein Konzept für das Zusammenspiel von Planer und Prozesssteuerung entwickelt und die Anwendbarkeit von KI-Planungstechniken für die Planung von Geschäftsprozessen analysiert. In Kapitel 4 wurde beschreiben, wie Prozesse automatisch überwacht und zur Berücksichtigung von unerwarteten Ereignissen zur Laufzeit angepasst werden können. Es wurde die Problematik einer Umplanung zur Laufzeit dargestellt und hierfür alternative Lösungen entwickelt und verglichen. In Kapitel 5 wurde zur Präzisierung und Validierung des Ansatzes ein formales Modell der integrierten Prozessplanung und -ausführung formuliert und untersucht. Mit diesem Modell ließ sich die korrekte Funktionsweise der in den vorhergehenden Kapiteln entwickelten Lösungen nachweisen. Es wurde gezeigt, unter welchen Voraussetzungen das Geschäftsziel eines Geschäftsfalls erreicht wird. Darüber hinaus konnte nachgewiesen werden, dass die in Kapitel 4 spezifizierte Vorgehensweise eine konsistente Anpassung einer Prozessdefinition zur Laufzeit ermöglicht. Mit *Plaengine* wurde ein System zur

integrierten Prozessplanung und -ausführung entworfen und implementiert. Hierdurch konnte auch die softwaretechnische Umsetzbarkeit der Konzepte und Verfahren aus den Kapiteln 3 und 4 nachgewiesen werden. Zusammenfassend wurden somit die zu Beginn dieser Arbeit aufgestellten Ziele erreicht.

9.2 Ausblick

Zukünftige Arbeiten konzentrieren sich auf den Einsatz der integrierten Prozessplanung und -ausführung in realen, industriellen Szenarien. Ein wichtiger Schritt in diese Richtung ist das Adaptive Services Grid (ASG) Projekt [4], das durch das sechste Forschungsrahmenprogramm der Europäischen Union gefördert wird. Ein wesentlicher Aspekt dieses Projekts ist die semantische Repräsentation von Diensten und darauf aufbauend ihre automatische Auswahl und Komposition [64]. Hierbei kommen Konzepte, Techniken und Teilkomponenten aus der Entwicklung von Plaengine zum Einsatz.

Es hat sich gezeigt, dass die für eine Automatisierung der Planung notwendige Modellierung einer Domäne sehr aufwendig ist. Ist die Modellierung auch nur in Teilen unvollständig oder fehlerhaft, können gegebenenfalls Prozessdefinitionen erzeugt werden, die das Ziel nicht erreichen. Daher ist ein weiterer zukünftiger Forschungsschwerpunkt die Fragestellung, inwieweit die in Abschnitt 7.4.1 beschriebene teilautomatische Planung in solchen Situationen einen praktischen Mehrwert bei der Prozessmodellierung bietet. Die Idee hierbei ist, auf eine vollständige Automatisierung zu verzichten, und dafür auch aus teilweise unvollständig oder fehlerhaft modellierten Domänen einen Nutzen bei der Modellierung von Prozessdefinitionen zu ziehen.

9.3 Fazit

In dieser Arbeit wurde eine Möglichkeit aufgezeigt durch die Integration von KI-Planungstechniken die Funktionalität von Workflow-Management-Systemen um die automatische Planung und Anpassung von Prozessen zu erweitern. Voraussetzung hierfür ist die vollständige und konsistente Modellierung der Domäne. Hierzu sollte diese eng begrenzt und klar strukturiert sein. Die Fähigkeit automatisch Prozesse zu planen und gegebenenfalls zur Laufzeit anzupassen, ist insbesondere in Anwendungsgebieten von Vorteil, die durch eine dynamische Domäne, häufige unerwartete Ereignisse und stark unterschiedlich strukturierte Geschäftsfälle charakterisiert sind. Sind diese Charakteristika gegeben, erscheint der Aufwand für die Modellierung der Domäne im Verhältnis zum Nutzen angemessen.

Literaturverzeichnis

[1] AALST, W.M.P. van der ; BASTEN, T.: Inheritance of workflows: an approach to tackling problems related to change. In: *Theoretical Computer Science* 270 (2002), Nr. 1–2, S. 125–203

[2] AALST, W.M.P. van der ; WESKE, Mathias ; GRÜNBAUER, Dolf: Case Handling: A New Paradigm for Business Process Support. In: *Data & Knowledge Engineering* 53 (2005), S. 129–162

[3] AALST, W.M.P. van der ; WESKE, Mathias ; WIRTZ, Guido: Advanced Topics in Workflow Management: Issues, Requirements, and Solutions. In: *Journal of Integrated Design and Process Science* 7 (2003), Nr. 3

[4] ADAPTIVE SERVICES GRID PROJECT: `http://asg-platform.org`

[5] AGOSTINI, Alessandra ; MICHELIS, Giorgio D.: Improving Flexibility of Workflow Management Systems. In: *Business Process Management, Models, Techniques, and Empirical Studies*. London, UK : Springer-Verlag, 2000. – ISBN 3–540–67454–3, S. 218–234

[6] AGOSTINI, Alessandra ; MICHELIS, Giorgio D.: A Light Workflow Management System Using Simple Process Models. In: *Computer Supported Cooperative Work* 9 (2000), Nr. 3/4, S. 335–363

[7] AIELLO, Marco ; PAPAZOGLOU, Mike P. ; YANG, Jian ; CARMAN, M. ; PISTORE, Marco ; SERAFINI, Luciano ; TRAVERSO, Paolo: A Request Language for Web-Services Based on Planning and Constraint Satisfaction. In: *TES '02: Proceedings of the Third International Workshop on Technologies for E-Services*, Springer-Verlag, 2002, S. 76–85

[8] AKRAM, M. S. ; MEDJAHED, Brahim ; BOUGUETTAYA, Athman: Supporting Dynamic Changes in Web Service Environments. In: *First International Conference on Service-Oriented Computing (ICSOC)*, Springer, 2003, S. 319–334

[9] ALONSO, Gustavo ; AGRAWAL, Divyakant ; ABBADI, Amr E. ; KAMATH, Mohan ; GÜNTHÖR, Roger ; MOHAN, C.: Advanced Transaction Models in

Workflow Contexts. In: *Proceedings of the 12th International Conference on Data Engineering, New Orleans, February 1996.*, 1996, S. 574–581

[10] ALONSO, Gustavo ; CASATI, Fabio ; KUNO, Harumi ; MACHIRAJU, Vijaj: *Web Services. Concepts, Architectures and Applications.* Springer, 2004

[11] AMBROSZKIEWICZ, Stanislaw: enTish: An Approach to Service Composition. In: *4th International Workshop on Technologies for E-Services*, Springer, 2003

[12] ANZBÖCK, Rainer ; DUSTDAR, Schahram: Semi-automatic Generation of Web Services and BPEL Processes - A Model-Driven Approach. In: *Lecture Notes in Computer Science* 3649 (2005), S. 64 – 79

[13] ATLURI, Vijayalakshmi ; CHUN, Soon A.: Handling Dynamic Changes in Decentralized Workflow Execution Environments. In: *Proceedings of the 14th International Conference on Database and Expert Systems Applications*, Springer, 2003 (LNCS 2736)

[14] BACCHUS, Fahiem ; ADY, Michael: Planning with Resources and Concurrency: A Forward Chaining Approach. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-2001)*, Morgan Kaufmann, 2001, S. 417–424

[15] BECKSTEIN, Clemens ; KLAUSNER, Joachim: A Meta Level Architecture for Workflow Management. In: *Proceedings of the Third Biennial World Conference on Integrated Design & Process Technology*, 1998

[16] BECKSTEIN, Clemens ; KLAUSNER, Joachim: A Planning Framework for Workflow Management. In: *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999

[17] BERTOLI, Piergiorgio ; CIMATTI, Alessandro ; PISTORE, Marco ; TRAVERSO, Paolo: A Framework for Planning with Extended Goals under Partial Observability. In: *ICAPS'03*, 2003, S. 215–225

[18] BLYTHE, Jim ; DEELMAN, Eva ; GIL, Yolanda: Planning for Workflow Construction and Maintenance on the Grid. In: *ICAPS'03 Workshop on Planning for Web Services*, 2003

[19] BOOCH, Grady ; RUMBAUGH, James ; JACOBSON, Ivar: *The Unified Modeling Language User Guide*. Addison Wesley, 1999

[20] BURBECK, Steve: *The Tao of e-business services*. Version: October 2000. `http://www-106.ibm.com/developerworks/library/ws-tao/`. IBM Corporation. – Online–Ressource

[21] CASATI, Fabio ; CERI, Stefano ; PERNICI, Barbara ; POZZI, Giuseppe: Workflow Evolution. In: *International Conference on Conceptual Modeling*, 1996, S. 438–455

[22] CHUN, Soon A. ; ATLURI, Vijayalakshmi ; ADAM, Nabil R.: Domain Knowledge-Based Automatic Workflow Generation. In: *Proceedings of the 13th International Conference on Database and Expert Systems Applications*, Springer-Verlag, 2002 (LNCS 2453), S. 81–92

[23] CHUN, Soon A. ; ATLURI, Vijayalakshmi ; ADAM, Nabil R.: Using Semantics for Policy-based Web Services Composition. In: *Journal of Distributed and Parallel Databases* 18 (2005), Nr. 1

[24] COUNCILL, William T. ; HEINEMAN, George T.: *Component-Based Software Engineering*. Addison-Wesley, 2001

[25] DAYAL, Umeshwar ; BLAUSTEIN, Barbara T. ; BUCHMANN, Alejandro P. ; CHAKRAVARTHY, Upen S. ; HSU, M. ; LEDIN, R. ; MCCARTHY, Dennis R. ; ROSENTHAL, Arnon ; SARIN, Sunil K. ; CAREY, Michael J. ; LIVNY, Miron ; JAUHARI, Rajiv: The HiPAC Project: Combining Active Databases and Timing Constraints. In: *SIGMOD Record* 17 (1988), Nr. 1, S. 51–70

[26] DEAN, Thomas ; BODDY, Mark: An analysis of time dependent planning. In: *Proceedings of AAAI-1988*, 1988, S. 49–54

[27] DEELMAN, Ewa ; BLYTHE, James ; GIL, Yolanda ; KESSELMAN, Carl ; MEHTA, Gaurang ; PATIL, Sonal ; SU, Mei-Hui ; VAHI, Karan ; LIVNY, Miron: Pegasus: Mapping Scientific Workflows onto the Grid. In: *Second European Across Grids Conference*, Springer, 2004 (LNCS 3165), S. 11–20

[28] DEMING, William E.: *Out of the Crisis: Quality, Productivity and Competitive Position*. Cambridge : Cambridge University Press, 1992

[29] DO, Minh B. ; KAMBHAMPATI, S.: SAPA: A Multi-objective Metric Temporal Planner. In: *Journal of Artificial Intelligence Research* Bd. 20, Morgan Kaufmann, 2003, S. 155–194

[30] DO, Minh B. ; KAMBHAMPATI, Subbarao: SAPA: A Domain-Independent Heuristic Metric Temporal Planner. In: *Proceedings of 6th European Conference on Planning (ECP-01)*, 2001

[31] DO, Minh B. ; KAMBHAMPATI, Subbarao: Improving Temporal Flexibility of Position Constrained Metric Temporal Plans. In: *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS)*, AAAI, 2003, S. 42–51

[32] DUSTDAR, Schahram ; SCHREINER, Wolfgang: A Survey on Web services Composition. In: *International Journal of Web and Grid Services* 1 (2005), Nr. 1, S. 1 – 30

[33] EDER, Johann ; LIEBHART, Walter: Workflow Recovery. In: *Conference on Cooperative Information Systems*, 1996, S. 124–134

[34] ELLIS, Clarence ; KEDDARA, Karim ; ROZENBERG, Grzegorz: Dynamic Change Within Workflow Systems. In: *Proceedings of conference on Organizational computing systems*, ACM Press, 1995, S. 10–21

[35] ELLIS, Clarence ; MALTZAHN, Carlos: The Chautauqua Workflow System. In: *HICSS '97: Proceedings of the 30th Hawaii International Conference on System Sciences*, IEEE Computer Society, 1997

[36] ELLIS, Clarence A. ; NUTT, Gary J.: Office Information Systems and Computer Science. In: *ACM Computing Surveys* 12 (1980), Nr. 1, S. 27–60

[37] FAUVET, Marie-Christine ; DUARTE, Helga ; DUMAS, Marlon ; BENATALLAH, Boualem: Handling Transactional Properties in Web Service Composition. In: *Proceedings of the 6th International Conference on Web Information Systems Engineering*, 2005, S. 273–289

[38] FEILER, Peter H. ; HUMPHREY, Watts S.: Software Process Development and Enactment: Concepts and Definitions. In: *Proceedings of the Second International Conference on Software Process*, IEEE CS Press, 1993, S. 28–40

[39] FIKES, Richard E. ; NILSSON, Nils: STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. In: *Artificial Intelligence* 2 (1971), S. 189–208

[40] FOERSTER, Heinz von: *Observing Systems: Selected Papers of Heinz von Foerster*. Seaside (CA) : Intersystems Publications, 1981

[41] FOX, Maria ; LONG, Derek: PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. In: *Journal of Artificial Intelligence Research* 20 (2003), S. 61–124

[42] GEORGAKOPOULOS, Dimitrios ; HORNICK, Mark F. ; SHETH, Amit P.: An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. In: *Distributed and Parallel Databases* 3 (1995), Nr. 2, S. 119–153

[43] GEREVINI, Alfonso ; SERINA, Ivan: LPG: a Planner based on Local Search for Planning Graphs. In: *Proceedings of the Sixth Int. Conference on AI Planning and Scheduling (AIPS'02)*, AAAI Press, 2002

[44] GHALLAB, Malik ; HOWE, Adele ; KNOBLOCK, Craig A. ; MCDERMOTT, Drew ; RAM, Ashwin ; VELOSO, Manuela ; WELD, Daniel ; WIKINS, David: PDDL – The Planning Domain Definition Language. AIPS-98 / Yale Computer Science Department. 1998. – Forschungsbericht

[45] GHALLAB, Malik ; NAU, Dana ; TRAVERSO, Paolo: *Automated Planning: Theory and Practice*. Morgan Kaufmann, 2004

[46] GIL, Yolanda ; DEELMAN, Ewa ; BLYTHE, Jim ; KESSELMAN, Carl ; TANGMUNARUNKIT, Hongsuda: Artificial Intelligence and Grids: Workflow Planning and Beyond. In: *IEEE Intelligent Systems* 19 (2004), Nr. 1, S. 26–33

[47] GÜNTHER, Christian: *Konzeption und Implementierung einer Integrationsumgebung für Prozessplanung und -ausführung*. 2004. – Masterarbeit, Hasso-Plattner-Institut für Softwaresystemtechnik, Universität Potsdam

[48] GREINER, Ulrike ; RAHM, Erhard: WebFlow: Ein System zur flexiblen Ausführung webbasierter, kooperativer Workflows. In: *10. Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW)*, 2003

[49] HAGEN, Claus ; ALONSO, Gustavo: Exception Handling in Workflow Management Systems. In: *IEEE Transactions on Software Engineering* 26 (2000), Nr. 10, S. 943–958

[50] HAKIMPOUR, Farshad ; SELL, Denilson ; CABRAL, Liliana ; DOMINGUE, John ; MOTTA, Enrico: Semantic Web Service Composition in IRS-III: The Structured Approach. In: *CEC*, IEEE Computer Society, 2005, S. 484–487

[51] HAMMER, Michael ; CHAMPY, James: *Reengineering the corporation*. New York : Harper Collins Publishing, 1993

[52] HOFFMANN, Jörg: The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. In: *Journal of Artificial Intelligence Research* 20 (2003), S. 291–341

[53] HORN, Stefan ; JABLONSKI, Stefan: An Approach to Dynamic Instance Adaption in Workflow Management Applications. In: *Conference on Computer Supported Cooperative Work (CSCW)*, 1998

[54] HÜNDLING, Jens ; WESKE, Mathias: Web Services: Foundation and Composition. In: *EM - Electronic Markets* 13 (2003), Nr. 2, S. 108–119

[55] JABLONSKI, Stefan ; BUSSLER, Christoph: *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, 1996

[56] JOERIS, Gregor ; HERZOG, Otthein: Managing Evolving Workflow Specifications. In: *Proceedings of the 3rd International Conference on Cooperative Information Systems*, 1998

[57] KARTAM, Nabil Abdul-Fattah: *Investigating the utility of artificial intelligence techniques for automatic generation of construction project plans*, Stanford University Department of Civil Engineering, Diss., 1989

[58] KIFER, Michael ; LAUSEN, Georg ; WU, James: Logical foundations of object-oriented and frame-based languages. In: *Journal of the ACM* 42 (1995), Nr. 4

[59] KIM, Jihie ; BLYTHE, Jim: Supporting Plan Authoring and Analysis. In: *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*. New York, NY, USA : ACM Press, 2003, S. 109–116

[60] KIM, Jihie ; SPRARAGEN, Marc ; GIL, Yolanda: An Intelligent Assistant for Interactive Workflow Composition. In: *IUI '04: Proceedings of the 9th international conference on Intelligent user interface*. New York, NY, USA : ACM Press, 2004, S. 125–131

[61] KLISCHEWSKI, Ralf ; WETZEL, Ingrid: Serviceflow Management. In: *Informatik Spektrum* 23 (2000), Nr. 1, S. 38–46

[62] KOLODNER, Janet: *Case-Based Reasoning*. Morgan Kaufman, 1993

[63] KRADOLFER, Markus ; GEPPERT, Andreas: Dynamic Workflow Schema Evolution Based on Workflow Type Versioning and Workflow Migration. In: *Proceedings of the Fourth International Conference on Cooperative Information Systems*, 1999, S. 104–114

[64] KUROPKA, Dominik ; WESKE, Mathias: Die Adaptive Services Grid Plattform: Motivation, Potential, Funktionsweise und Anwendungsszenarien. In: *Emisa Forum* 26 (2006), Nr. 1, S. 13–25

[65] LAZOVIK, Alexander ; AIELLO, Marco ; PAPAZOGLOU, Mike: Planning and Monitoring the Execution of Web Service Requests. In: ORLOWSKA, Maria (Hrsg.) ; WEERAWARANA, Sanjiva (Hrsg.) ; PAPAZOGLOU, Mike (Hrsg.) ; YANG, Jian (Hrsg.): *1st international conference on service-oriented computing (ICSOC'03)*, Springer-Verlag, 2003 (LNCS 2910)

[66] LEYMANN, Frank: Supporting Business Transactions Via Partial Backward Recovery In Workflow Management Systems. In: *Datenbanksysteme in Büro, Technik und Wissenschaft (BTW)*, 1995, S. 51–70

[67] LEYMANN, Frank: Web Services Flow Language (WSFL 1.0) / IBM Corporation. 2001. – Technical report

[68] LEYMANN, Frank ; ROLLER, Dieter: *Production Workflow: Concepts and Techniques*. Prentice Hall, 2000

[69] LIMTHANMAPHON, Benchaphon ; ZHANG, Yanchun: Web service composition transaction management. In: *CRPIT '04: Proceedings of the fifteenth conference on Australasian database*, Australian Computer Society, Inc., 2004, S. 171–179

[70] MADHUSUDAN, Therani ; ZHAO, Leon: A Case-Based Framework for Workflow Model Management. In: *Proceedings of the Business Process Management Conference* Bd. 2678, Springer, 2003, S. 354–369

[71] MADHUSUDAN, Therani ; ZHAO, Leon ; MARSHALL, Byron: A Case-based Reasoning Framework for Workflow Model Management. In: *Data & Knowledge Engineering* 50 (2004), Nr. 1, S. 87–115

[72] MEYER, Harald: *Entwicklung und Realisierung einer Planungskomponente für die Komposition von Diensten*. 2005. – Diplomarbeit, Hasso-Plattner-Institut für Softwaresystemtechnik, Universität Potsdam

[73] MYERS, Karen L. ; TYSON, Mabry ; WOLVERTON, Michael J. ; JARVIS, Peter A. ; LEE, Thomas J. ; DESJARDINS, Marie: PASSAT: A User-centric Planning Framework. In: *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*, AAAI, 2002

[74] NARAYANAN, Srini ; MCILRAITH, Sheila: Simulation, Verification and Automated Composition of Web Services. In: *11th International World Wide Web Conference*, 2002

[75] NAU, Dana S. ; AU, Tsz-Chiu ; ILGHAMI, Okhtay ; KUTER, Ugur ; MURDOCK, J. W. ; WU, Dan ; YAMAN, Fusun: SHOP2: An HTN Planning System. In: *Journal of Artificial Intelligence Research (JAIR)* 20 (2003), S. 379–404

[76] NGUYEN, Minh N. ; CONRADI, Reidar: Towards a Rigorous Approach for Managing Process Evolution. In: *EWSPT '96: Proceedings of the 5th European Workshop on Software Process Technology*, Springer-Verlag, 1996, S. 18–35

[77] NORDSIECK, Fritz: *Betriebsorganisation. Lehre und Technik (Textband)*. 2nd enhanced edition. Stuttgart : C. E. Poeschel Verlag, 1972

[78] ORGANIZATION FOR THE ADVANCEMENT OF STRUCTURED INFORMATION STANDARDS (OASIS): *Web Services Business Process Execution Language Version 2.0*. Version: December 2005. `http://www.oasis-open.org/committees/download.php/16024/wsbpel-specification-draft-Dec-22-2005.htm`. – Online–Ressource. – Committee Draft, 21th Dec. 2005

[79] OVERDICK, Hagen: *Entwurf und Implementierung von Prozesssteuerungskomponenten in service-orientierte Umgebungen*. 2005. – Masterarbeit, Hasso-Plattner-Institut für Softwaresystemtechnik, Universität Potsdam

[80] PAPAZOGLOU, Michael P.: Web Services and Business Transactions. In: *World Wide Web* 6 (2003), Nr. 1, S. 49–91

[81] PISTORE, Marco ; BARBON, Fabio ; BERTOLI, Piergiorgio ; SHAPARAU, Dmitry ; TRAVERSO, Paolo: Planning and Monitoring Web Service Composition. In: *ICAPS'04 Workshop on Planning and Scheduling for Web and Grid Services*. Canada, 2004

[82] PRYOR, Louise ; COLLINS, Gregg: Planning for Contingencies: A Decision-based Approach. In: *Journal of Artificial Intelligence Research* 4 (1996), S. 287–339

[83] REICHERT, Manfred ; DADAM, Peter: ADEPTflex - Supporting Dynamic Changes of Workflows Without Losing Control. In: *Journal of Intelligent Information Systems* 10 (1998), Nr. 2, S. 93–129

[84] RINDERLE, Stefanie ; REICHERT, Manfred ; DADAM, Peter: Correctness criteria for dynamic changes in workflow systems: a survey. In: *Data and Knowledge Engineering* 50 (2004), Nr. 1

[85] RODRIGUEZ-MORENO, Maria D. ; BORRAJO, Daniel ; MEZIAT, Daniel: Proces Modelling and AI Planning Techniques: A New Apporach. In: *Proceedings of the 2nd International Workshop on Information Integration and Web-based Applications Services*, 2000

[86] RODRIGUEZ-MORENO, Maria D. ; KEARNEY, Paul: Integrating AI planning techniques with workflow management system. In: *Knowledge Based Systems* 15 (2002), Nr. 5-6, S. 285–291

[87] RUSINKIEWICZ, Marek ; KLAS, Wolfgang ; TESCH, Thomas ; WASCH, Jurgen ; MUTH, Peter: Towards a Cooperative Transaction Model - The Cooperative Activity Model. In: *The VLDB Journal*, 1995, S. 194–205

[88] RUSINKIEWICZ, Marek ; SHETH, Amit P.: Specification and Execution of Transactional Workflows. In: *Modern Database Systems: The Object Model, Interoperability, and Beyond.* 1995, S. 592–620

[89] RUSSELL, Stuart ; NORVIG, Peter: *Artificial Intelligence: A Modern Approach*. 2nd edition. Prentice-Hall, Englewood Cliffs, NJ, 2003

[90] SACERDOTI, Earl D.: The Nonlinear Nature of Plans. In: ALLEN, J. (Hrsg.) ; HENDLER, J. (Hrsg.) ; TATE, A. (Hrsg.): *Readings in Planning*. Kaufmann, 1990, S. 162–170

[91] SADIQ, Shazia: Handling Dynamic Schema changes in Workflow Processes. In: *Proceedings of the 11th Australian Database Conference*, Australian Computer Society, 2000

[92] SADIQ, Shazia ; MARJANOVIC, Olivera ; ORLOWSKA, Maria E.: Managing Change and Time in Dynamic Workflow Processes. In: *International Journal of Cooperative Information Systems* 9 (2000), Nr. 1

[93] SCHMIT, Benjamin A. ; DUSTDAR, Schahram: Model-driven Development of Web Service Transactions. In: *International Journal of Enterprise Modelling and Information Systems Architecture* 1 (2005), Nr. 1

[94] SHETH, Amit P. ; GEORGAKOPOULOS, Dimitrios ; JOOSTEN, Stef ; RUSINKIEWICZ, Marek ; SCACCHI, Walt ; WILEDEN, Jack C. ; WOLF, Alexander L.: Report from the NSF Workshop on Workflow and Process Automation in Information Systems. In: *SIGMOD Record* 25 (1996), Nr. 4, S. 55–67

[95] SHIU, Simon C. K. ; PAL, Sankar K.: Case-Based Reasoning: Concepts, Features and Soft Computing. In: *Applied Intelligence* 21 (2004), Nr. 3, S. 233–238

[96] SIRIN, Evren ; PARSIA, Bijan ; HENDLER, James: Filtering and Selecting Semantic Web Services with Interactive Composition Techniques. In: *IEEE Intelligent Systems* 19 (2004), S. 42–49

[97] SMITH, David E. ; WELD, Daniel S.: Temporal Planning with Mutual Exclusion Reasoning. In: *IJCAI*, 326-337

[98] STEFIK, Mark: Planning and Meta-Planning. In: WEBBER, B. L. (Hrsg.) ; NILSSON, N. J. (Hrsg.): *Readings in Artificial Intelligence*. Tioga, 1981, S. 272–286

[99] SU, Stanley Y. W. ; MENG, Jie ; KRITHIVASAN, Raja ; DEGWEKAR, Seema ; HELAL, Sumi: Dynamic Inter-Enterprise Workflow Management in a Constraint-Based E-Service Infrastructure. In: *Electronic Commerce Research* 3 (2003), Nr. 1-2, S. 9–24

[100] TATE, Austin: INTERPLAN: A plan generation system which can deal with interactions between goals / University of Edinburgh. 1974. – Forschungsbericht

[101] TATE, Austin: Generating Project Networks. In: ALLEN, J. (Hrsg.) ; HENDLER, J. (Hrsg.) ; TATE, A. (Hrsg.): *Readings in Planning*. Kaufmann, 1990, S. 291–296

[102] THATTE, Satish: XLANG - Web Services for Business Process Design / Microsoft Corporation. 2001. – Initial Public Draft

[103] THE DAML SERVICES COALITION: DAML-S: Web Service Description for the Semantic Web. In: *The First International Semantic Web Conference (ISWC)*, 2002

[104] TSAMARDINOS, Ioannis ; MUSCETTOLA, Nicola ; MORRIS, Paul: Fast transformation of temporal plans for efficient execution. In: *Proceedings of the fifteenth national conference on Artificial intelligence*, American Association for Artificial Intelligence, 1998, S. 254–261

[105] VOSSEN, Gottfried ; WESKE, Mathias: The WASA2 object-oriented workflow management system. In: *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, ACM Press, 1999, S. 587–589

[106] WASTELL, David ; WHITE, P. ; KAWALEK, Peter: A Methodology for Business Process Re-Design: experiences and issues. In: *Journal of Strategic Information Systems* 3 (1994), Nr. 1, S. 23–40

[107] WEERAWARANA, Sanjiva ; CURBERA, Francisco ; LEYMANN, Frank: *Web Services Platform Architecture*. Prentice Hall, 2005

[108] WESKE, Mathias: *Workflow Management Systems: Formal Foundation, Conceptual Design, Implementation Aspects*. Universität Münster, 2000 (Habilitationsschrift Fachbereich Mathematik und Informatik)

[109] WESKE, Mathias ; HÜNDLING, Jens ; KUROPKA, Dominik ; SCHUSCHEL, Hilmar: Objektorientierter Entwurf eines flexiblen Workflow-Management-Systems. In: *Informatik Forschung und Entwicklung* 13 (1998), Nr. 4, S. 179–195

[110] WESKE, Mathias ; VOSSEN, Gottfried ; PUHLMANN, Frank: Workflow and Service Composition Languages. In: BERNUS, P. (Hrsg.) ; MERTINS, K. (Hrsg.) ; SCHMIDT, G. (Hrsg.): *Handbook on Architectures of Information Systemst*, Springer, 2005, S. 369–390

[111] WETZEL, Ingrid ; KLISCHEWSKI, Ralf: Serviceflow beyond workflow? IT support for managing inter-organizational service processes. In: *Information Systems* 29 (2004), Nr. 2, S. 127–145

[112] WU, Dan ; PARSIA, Bijan ; SIRIN, Evren ; HENDLER, James ; NAU, Dana: Automating DAML-S Web Services Composition Using SHOP2. In: *2nd International Semantic Web Conference (ISWC'03)*, 2003

[113] ZENG, Liangzhao ; BENATALLAH, Boualem ; DUMAS, Marlon ; KALAGNANAM, Jayant ; SHENG, Quan: Quality driven web services composition. In: *Proceedings of the twelfth international conference on World Wide Web*, ACM Press, 2003, S. 411–421

[114] ZENG, Liangzhao ; BENATALLAH, Boualem ; LEI, Hui ; NGU, Anne Hee H. ; FLAXER, David ; CHANG, Henry: Flexible Composition of Enterprise Web Services. In: *Electronic Markets - Web Services* 13 (2003), Nr. 2

[115] ZENG, Liangzhao ; FLAXER, David ; CHANG, Henry ; JENG, Jun-Jang: PLMflow - Dynamic Business Process Composition and Execution by Rule Inference. In: *TES '02: Proceedings of the Third International Workshop on Technologies for E-Services*, Springer-Verlag, 2002 (LNCS 2444), S. 141–150

[116] ZENG, Liangzhao ; LEI, Hui ; JENG, Jun jang ; CHUNG, Jen-Yao ; BENATALLAH, Boualem: Policy-Driven Exception-Management for Composite Web Services. In: *Proceedings of the 7th IEEE International Conference on E-Commerce Technology*, IEEE Computer Society, 2005, S. 355–363

[117] ZISMAN, Michael: *Representation, Specification, and Automation of Office Procedures*, University of Pennsylvania, PhD Thesis, 1977

[118] ZUR MUEHLEN, Michael: *Advances in Information Systems and Management Science*. Bd. 6: *Workflow-based Process Controlling. Foundation, Design, and Implementation of Workflow-driven Process Information Systems.* Berlin : Logos, 2004