Seminarreader

Prozessmodellierung

Editoren

Frank Puhlmann
Arnd Schnieders
Mathias Weske

Kontakt
Hasso Plattner Institut für Softwaresystemtechnik
an der Universität Potsdam
Fachgebiet Business Process Technology
P.O. Box 90 04 60
D-14440 Potsdam

Editorial

Dieser Seminarreader zum Thema Prozessmodellierung ist das Ergebnis eines Bachelor-Seminars, welches im Sommersemester 2004 am Hasso-Plattner-Institut für Softwaresystemtechnik an der Universität Potsdam durchgeführt wurde. Studenten des zweiten und dritten Studienjahres der Softwaresystemtechnik modellierten Prozesse aus verschiedenen Bereichen, erstellten und hielten Präsentation und verfassten Berichte, welche in diesem Seminarreader zusammengefasst sind. Das Seminar wurde im Rahmen des Drittmittelprojektes PESOA durchgeführt. Das PESOA-Projekt hat das Ziel, Softwareproduktlinien und Prozesse miteinander zu verbinden. Eine Teilaufgabe davon ist die Entwicklung so genannter „variantenreicher Prozessmodelle“, welche verschiedene Prozessvarianten in einem Modell zusammenfassen. Zur Erstellung einer Methodik für variantenreiche Prozessmodelle müssen in einem ersten Schritt existierende Notationen auf ihre Tauglichkeit zur Modellierung in einer bestimmten Domäne geprüft werden. Die erstellten Modelle und das erarbeitete Wissen können dann als Ausgangsbasis für weitere Schritte verwendet werden. Die Studenten erhalten die Möglichkeit, praxisnahe Prozesse in einer aktuell diskutierten Notation zu erstellen.

Die Präsentationen wurden im Rahmen von wöchentlichen Vorträgen durchgeführt, auf welche jeweils eine Diskussion folgte. Das Seminar wurde für drei Themenbereiche angelegt, ein weiteres Thema wurde durch einen Studenten eingebracht:

- Prozesse im Automobilbau
- Verwaltungsprozesse
- Prozesse im E-Commerce
- Softwareentwicklungsprozess bei SAP Berlin

Der Grundgedanke der Prozessmodellierung ist die explizite Repräsentation von dynamischen Aspekten eines Systems. Durch die explizite Repräsentation ist nicht nur eine verbesserte Wart- und Erweiterbarkeit durch Techniken wie automatische Codegenerierung gegeben, vielmehr wird dadurch erst die Erstellung von komplexen Systemen ermöglicht. So entfallen im Bereich Automobilbau schon heute mehr als 30 Prozent der Wertschöpfung auf elektronische Bauteile und die damit verbundene Software. An den entstehenden Kosten hat die Software einen immer höheren Anteil, der auch dadurch bedingt ist, dass sich der Umfang alle 2-3 Jahre verdoppelt. Viele neue Funktionalitäten, wie z.B. ESP werden erst durch Softwareunterstützung ermöglicht. Somit ist die effiziente Entwicklung qualitativ hochwertiger Software für einen Automobilbauer oder Zulieferer von entscheidender Bedeutung.

Im Rahmen des PESOA Projektes sollen Prozessbeschreibungen für verschiedene Varianten einer Motorsteuerung untersucht werden. Es wurden drei Aufgabenstellungen vergeben, welche, auf einer einheitlichen Datenbasis des Projektpartners DaimlerChrysler basierend, verschiedene Notationen zur Modellierung der Prozesse einer Motorsteuerung im Automobilbau analysieren sollten. Die erste Aufgabe umfasste die Modellierung der vorgegebenen Daten mittels UML2 Statecharts und Sequenzdiagrammen. Die zweite Aufgabe bezog sich auf die Modellierung mittels UML2 Activity Diagrams und berücksichtigte bereits erste Varianten. In der dritten Aufgabenstellung sollte analysiert werden, inwiefern sich Techniken aus dem Workflow-Bereich, hier im Besonderen YAWL (Yet Another Workflow Language) für technische Prozesse verwenden lassen.

Eine klassische Domäne der Prozessmodellierung befasst sich mit der Büroautomatisierung und Verwaltung. Dieser Bereich der Modellierung umfasst nicht nur die Prozesse, sondern im Besonderen auch die an den Prozessen beteiligten Personen und Daten. Einmal explizit erfasste und modellierte Verwaltungsprozesse können durch Standardsysteme unterstützt, geändert, optimiert und überwacht werden. Als Datengrundlage wurden Verwaltungsprozesse am Hasso-Plattner-Institut verwendet, welche durch die Studenten mittels Interviewtechniken zu erheben waren. Beispiele für solche Prozesse finden sich unter anderem in der Personalverwaltung, der Reisekostenabrechnung, der Studentenverwaltung, der Organisation und Planung wissenschaftlicher Kongresse, der Verwaltung der Raumbelegung sowie in der Beschaffung.

Es wurden drei Aufgabenstellung vergeben, welche die erhobenen Daten jeweils in der Workflow-Sprache YAWL abbilden sollten. Die erste Aufgabe im Bereich Verwaltungsprozesse umfasste Beschaffungsprozesse am HPI. Eine weitere Aufgabe bildete die Modellierung der Studien- und Prüfungsordnung als Prozess. Zusätzlich wurden Unterschiede und Gemeinsamkeiten zwischen der aktuellen und einer künftigen Version aufgezeigt. Eine dritte Aufgabe befasste sich mit der Modellierung der Studienprozesse am HPI.

Eine weitere wichtige Domäne des PESOA Projektes entspringt aus dem Bereich E-Commerce. Es gibt weltweit hunderttausende von kleinen und großen Internetshops. In all diesen Shops sind Geschäftsprozesse enthalten, welche durch eine entsprechende Software umgesetzt werden. Für das PESOA Projekt im Besonderen interessant ist, dass viele Prozesse in großen Teilen Ähnlichkeiten aufweisen und somit auch als variantenreiches Prozessmodell dargestellt werden könnten. Ein Hersteller von E-Commerce Software kann seine Produkte durch die Berücksichtigung der Gemeinsamkeiten und Unterschiede der Geschäftsprozesse seiner Kunden bei der Softwareentwicklung effizienter entwickeln.

Zur Modellierung der Geschäftsprozesse hinter führenden E-Commerce Webseiten sollte die neue Notation BPMN (Business Process Modeling Notation) verwendet und analysiert werden. Die Erhebung der Prozesse sollte durch Beobachtung erfolgen. Die Themen des Bereiches E-Commerce umfassten die Modellierung klassischer und moderner Online-Kaufhäuser wie Amazon.de, Otto.de und Dell.de. Als Online-Auktionshaus wurde Ebay.de betrachtet. Weitere Bereiche des E-Commerce finden sich im Dienstleistungssektor, welcher durch Hotelreservationssysteme, E-Mail Provider und die Fahrkartenreservierung und Bestellung der Deutschen Bahn AG repräsentiert wurden.

Eine zusammenfassende Auswertung des Seminars befindet sich in einem Report des PESOA Projektes[1]. Die Veranstalter möchten allen Studenten des Seminars für ihr Interesse und ihre Teilnahme danken. Im Besonderen die Diskussionen mit Studenten, sowohl bei Einzelterminen, als auch nach und während der Präsentationen, haben viele neue Ideen und Anregungen für die weitere Arbeit geliefert.

F.P., A.S., M.W.

Potsdam, September 2004

[1] Frank Puhlmann. Modeling Workflows in the E-Business Domain. PESOA Report No. 08/2004. Hasso-Plattner-Institut, Potsdam.

Inhaltsverzeichnis

Modellierung einer Motorsteuerung mit UML Statecharts und Sequenzdiagrammen

Philipp Sommer

Hasso-Plattner-Institut für Softwaresystemtechnik
philipp.sommer@hpi.uni-potsdam.de

Abstract. Die zunehmende Bedeutung von Software im Bereich der embedded systems und deren Komplexität machen es notwendig, wie bei konventionellen Softwaresystemen bereits üblich, die in den Systemen ablaufenden Prozesse zu modellieren. Aber nicht nur die Modellierung des Verhaltens der Software sondern auch die Flexibilität dieser Modelle um Funktionalität hinzuzufügen, zu ersetzen und zu entfernen stellt eine Herausforderung dar. Inwiefern UML 2.0 Statecharts und Sequenzdiagramme die Möglichkeit bieten diese Variabilität zu modellieren, soll Gegenstand dieser Ausarbeitung sein. Am Beispiel einer Motorsteuerung, modelliert mit Prozessdaten von DaimlerChrysler, sollen die Möglichkeiten und Grenzen der Sprache gezeigt werden. Die Evaluierung der Eignung der Modellierungssprache wird anhand ausgewählter Kriterien vorgenommen.

1 Einleitung

Nahezu überall im alltäglichen Leben verwenden wir heute technische Geräte, in denen Mikrocontroller diverse Steuerungsaufgaben übernehmen. Diese so genannten embedded systems wickeln in zunehmendem Maße komplexe Software ab. Ein Sektor, in dem diese Entwicklung in den letzten Jahren besonderes rasant verlief, ist die Automobilbranche. Systeme, die zum Teil von Zulieferern produziert werden, müssen hier zu einem funktionierenden Gesamtsystem zusammengefügt werden.

Dadurch dass diese Systeme in großem Maße reaktiven Charakter besitzen, was bedeutet, dass sie hauptsächlich auf Eingaben reagieren und diese zu entsprechenden Ausgaben in Abhängigkeit eines internen Zustands verarbeiten [13], bieten sich zur Modellierung Zustandsdiagramme an. Klassische Zustandsgraphen haben jedoch den Nachteil, dass sie weder Hierarchieebenen noch Parallelität darstellen können, so dass die Zahl der Zustände mit zunehmender Komplexität exponentiell anwächst. Dadurch wird die Lesbarkeit solcher Zustandsgraphen beinahe unmöglich. Um diesen Problemen zu begegnen, entwickelte David Harel in [11] eine umfangreiche Erweiterung dieser Notationen. Seine Statecharts können über beliebige Ebenen verfeinert werden und bieten die Möglichkeit zur Modellierung von Nebenläufigkeit. Diese Notation ist später mit einigen kleinen Erweiterungen für objekt-orientierte Systeme für die Unified Modeling Language (UML) übernommen worden.

Daher im folgenden Kapitel zuerst eine kurze Übersicht der Möglichkeiten der UML in der Version 2.0, Verhalten und Prozesse zu modellieren. Dabei sollen die

Basiselemente des Metamodells mit deren Syntax und Semantik erläutert werden. Das dritte Kapitel wird sich dann mit der Modellierung der Motorsteuerung anhand der Prozessdaten von DaimlerChrysler widmen. Dabei werden zuerst die grobe statische Struktur der beteiligten Komponenten und deren Kommunikation mit einem Modell dargestellt. In den anschließenden Abschnitten folgen Statecharts zum Starten und Beenden sowie zum Betrieb. Zur Veranschaulichung der ablaufenden Kommunikation zwischen den beteiligten Objekten dient in den einzelnen Abschnitten jeweils ein Sequenzdiagramm. Das vierte Kapitel beschäftigt sich dann mit der Problematik der Modellierung von Varianten. Anhand zweier Beispiele sollen die Möglichkeiten, aber auch die Einschränkung der UML Statecharts zur Modellierung erörtert werden. Im letzten Teil dieser Ausarbeitung soll die Modellierungssprache anhand einiger für die Domäne des Automobilbaus relevanten Kriterien bewertet und die Ergebnisse noch einmal zusammengefasst werden.

2 Die Modellierungssprache

Seit ihrer Entwicklung 1995 durch Grady Booch [4], James Rumbaugh [19] und Ivar Jacobsen hat die UML im Bereich der Softwaremodellierung weite Verbreitung erreicht. Inzwischen weiterentwickelt durch die Object Management Group ist die Spezifikation der Version 2.0 [17] seit Anfang 2004 aktuell. Im Folgenden sollen zuerst die Statecharts und deren Rolle in der UML eingeordnet werden und anschließend deren Syntax und Semantik erklärt werden. Der zweite Teil widmet sich dann den Sequenzdiagrammen.

Ein zentraler Begriff in der UML ist das Verhalten. Verhalten kann sowohl für Objekte als auch für Operationen spezifiziert werden, dabei ist es dem Systemmodellierer überlassen, sich für einen der Untertypen zur Verhaltensbeschreibung zu entscheiden. Die unterschiedlichen Möglichkeiten haben nicht alle dieselbe Ausdrucksfähigkeit und sind in Abhängigkeit der Anwendungsdomäne mehr oder weniger geeignet. Die wichtigsten Varianten sind Statecharts und Aktivitätsdiagramme, aber auch Interaktionsdiagramme und Use-Cases sind möglich.

In Abbildung 5, die Anhang zu finden ist, ist das Metamodell der Statecharts zu sehen. Zentrale Elemente sind Zustand, Transition, Trigger und Aktivität. Zu Beginn der Abwicklung ist der durch den Anfangsknoten, repräsentiert durch einen ausgefüllten Kreis, markierte Zustand aktiv. Die graphische Repräsentation eines Zustands ist ein abgerundetes Rechteck (Notation siehe Abbildung 1). Das System befindet sich außer beim Schalten einer Transition immer in einem Zustand, dem so genannten aktiven Zustand. Der Prozess ist beendet, wenn das System den Endzustand, dargestellt durch einen ausgefüllten Kreis mit Umrundung, erreicht.

Transitionen zwischen Zuständen stellen die möglichen Übergänge dar. Meist werden sie durch einen Trigger ausgelöst, der wiederum durch ein Signal, ein zeitliches Ereignis oder eine Änderung in der Belegung von Attributen angestoßen wird. Besitzt eine Transition keine Trigger, schaltet sie sobald der Ursprungszustand aktiv ist. Jeder Transition kann eine Aktivität zugewiesen werden, die beim Schalten ausgeführt wird. Diese kann wiederum nur aus einer elementaren Aktion bestehen oder bei komplexeren Aktivitäten durch ein Aktivitätsdiagramm spezifiziert werden.

Zusätzlich besteht die Möglichkeit, das Schalten einer Transition an eine Bedingung zu knüpfen, so genannte Guards, die in eckigen Klammern an die Transition geschrieben werden (siehe Abbildung 1).

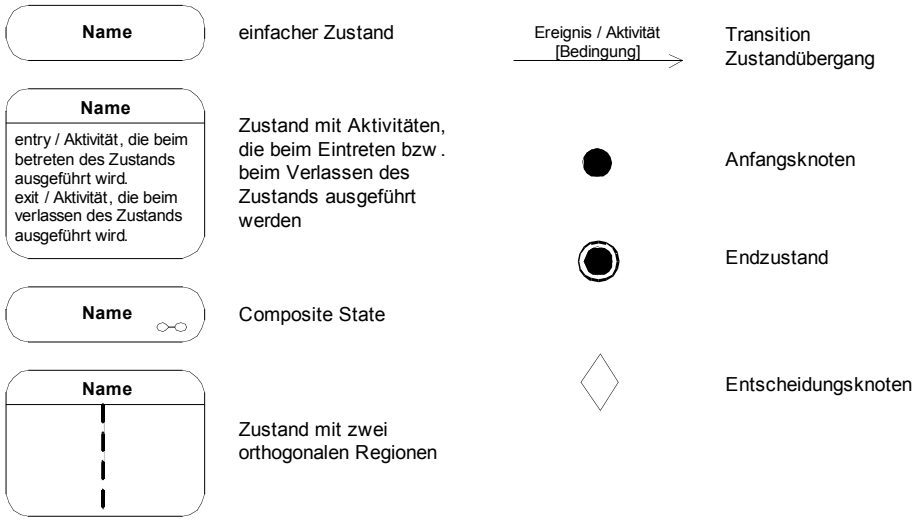


Abb. 1. Graphische Elemente der Statecharts

Pseudozustände sind Knoten, die jedoch keine aktiven Zustände sein können, sondern meistens lediglich einfache Transitionen zu komplexeren zusammenfassen und dadurch den Kontrollfluss regeln. Neben dem Anfangsknoten gibt es den Entscheidungsknoten, der eine eingehende und diverse ausgehende Transitionen haben kann, an denen nicht überschneidende Bedingungen notiert sein müssen. Von Pseudozuständen ausgehende Transitionen dürfen außerdem keine Trigger haben, damit sichergestellt ist, dass eine komplexe Transition nur einen Auslöser hat. Grundsätzlich reagiert das modellierte System nur auf Ereignisse, die an Transitionen stehen, deren Ursprungszustand aktiv ist. Das System arbeitet nach der „run-to-completion“ Semantik, d. h. ein Ereignis aus dem Ereignispool wird bearbeitet indem möglicherweise eine Transition schaltet, Guards ausgewertet und alle mit ihr verbundenen Aktivitäten abgearbeitet werden. Anschließend befindet sich das System wieder in einem aktiven Zustand. Dieser Zustand kann aber auch der vorherige sein, wenn zum Beispiel Bedingungen nicht erfüllt waren. Erst nachdem sich das System wieder in einem aktiven Zustand befindet, wird das nächste Ereignis aus dem Ereignispool entfernt und bearbeitet.

Zur Lösung der bereits in der Einleitung angesprochenen Probleme mit Hierarchie und Nebenläufigkeit bietet die UML zum einen mit den Composite States Zustände, die wiederum einen Statechart also eine Menge an Zuständen und Transitionen enthalten können. Zusätzlich kann ein solcher Zustand beliebig viele orthogonale Regionen besitzen, die jeweils durch einen Statechart mit einem ausgesprochenen initialen Zustand verfeinert werden. Wird ein solcher Composite State aktiv, so wird in jeder orthogonalen Region der initiale Zustand aktiviert. Diese Regionen sind die

einzige Ausnahme, in denen mehrerer Zustände gleichzeitig aktiviert sein können, damit das System nebenläufig arbeiten kann.

Sequenzdiagramme beschreiben wie auch Kommunikations-, Interaktions- und Zeitdiagramme die Interaktion zwischen Objektinstanzen, wobei bei Sequenzdiagrammen der Fokus auf dem Austausch der Nachrichten in der zeitlichen Abfolge liegt. Mit Sequenzdiagrammen kann zum einen die Interaktion vollständig beschrieben werden, zum anderen besteht aber auch die Möglichkeit, nur ausgewählte Szenarien darzustellen. In dieser Ausarbeitung wurde die zweite Variante gewählt, da die vollständige Spezifikation des Systems mit den Statecharts bereits beschrieben ist, so dass nur noch an ausgewählten Stellen, an denen die zeitliche Abfolge der Kommunikation eine entscheidende Rolle spielt, der Sachverhalt durch ein Sequenzdiagramm verdeutlicht werden soll.

3 Die Modellierung

Hintergrund für das folgende Beispiel einer Motorsteuerung ist das PESOA (Process Family Engineering for Service-Oriented Applications) Forschungsprojekt. PESOA ist ein Verbundprojekt, das vom Bundesministerium für Bildung und Forschung (BMBF) unterstützt wird. Ziel des Projekts sind der Entwurf und die prototypische Implementierung einer Plattform für Process Family Engineering und ihre Anwendung in den Bereichen e-Business und Telematikdienste [21]. Der Kooperationspartner im Bereich der Telematik, DaimlerChrysler, stellte die zugrunde liegenden Prozessdaten für folgende Modelle zur Verfügung. Zunächst soll in diesem Kapitel der Aufbau des Systems modelliert werden und anschließend die darin ablaufenden Prozesse. In Bezug auf die Modellierung von Prozessfamilien und die darin vorkommenden Varianten wird in Kapitel 4 näher eingegangen.

Zur Steuerung des Motors sind Sensoren und Aktoren notwendig. Um einen Überblick über den Aufbau des Systems zu gewinnen, sind in Abbildung 2 im wesentlichen alle in den Prozessmodellen relevanten Sensoren und Aktoren dargestellt und deren Möglichkeit zu kommunizieren in der FMC-Notation [1]. Es ist zu sehen, dass die Menge der Sensoren in zwei Gruppen eingeteilt werden kann, auf der linken Seite die Sensoren, die vom Fahrer beeinflusst werden, auf der rechten Seite diejenigen, die der Motorsteuerung Messwerte liefern. Dadurch dass Kanäle, dargestellt durch kleine Kreise, in beiden Richtungen zwischen der Motorsteuerung und der Menge der Sensoren vorhanden sind, ist es möglich, dass zum einen ein Sensor ein Ereignis melden kann, zum anderen aber auch die Motorsteuerung einzelne Messwerte jederzeit abfragen kann. Im folgenden Abschnitt ist erläutert, wie sich dieser Sachverhalt in UML darstellt. Hierzu ist anzumerken, dass zwischen Steuerung und dem eigentlichem Sensor ein Treiber die kontinuierlichen Spannungen in elektronisch codierte Werte umrechnet. Dieser Treiber ist in der Lage zu erkennen, wenn ein Grenzwert überschritten wird und sendet dann ein Signal an die Motorsteuerung. Im unteren Teil des Modells sind die wichtigsten Aktoren des Systems, die die berechneten Parameter am Motor einstellen.

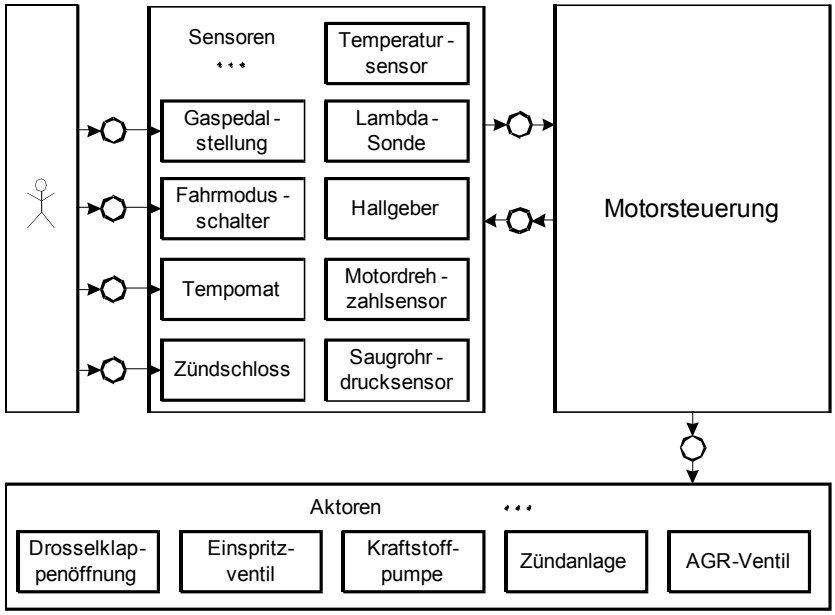


Abb. 2. Aufbau des System: Einteilung in Sensoren und Aktoren

In UML werden diese Sensoren und Aktoren sowie die Motorsteuerung selbst durch Objekte repräsentiert, die über Signale miteinander kommunizieren. In den folgenden Diagrammen ist jedoch nur das Verhalten des Objekts Motorsteuerung modelliert, wobei Signale von den Sensoren verarbeitet werden und Signale ausgegeben werden, um die Aktoren zu steuern.

3.1 Übersichtsdiagramme

Zuerst sollen hier nun die Prozesse auf höchster Ebene betrachtet werden, bevor im nächsten Abschnitt der eigentliche Betrieb der Motorsteuerung genauer modelliert wird. Das System wird gestartet, wenn der Zündschlüssel von der Position 0 in die Position I gedreht wird (siehe Abbildung 6 im Anhang). Die einzige Ausnahme, in der von Pseudozuständen ausgehenden Transitionen durch Trigger ausgelöst werden dürfen, ist wenn sie dem Stereotyp „<<create>>“ zugeordnet sind. Auf dieses erste Ereignis reagiert deshalb die Motorsteuerung nicht im eigentlichen Sinne, da die Stromzufuhr zur Motorsteuerung durch den Zündschlüssel hergestellt wird. Im Sequenzdiagramm (siehe Abbildung 7 im Anhang) wird dieser Sachverhalt dadurch dargestellt, dass das Rechteck, das die Objektinstanz der Motorsteuerung symbolisiert, später in der zeitlichen Abfolge erscheint als der Fahrer.

Sobald der Zustand „Stopp (Zündung II)“ aktiviert wird, wird die Aktivität „Init“ ausgeführt. Wie im Sequenzdiagramm zu sehen ist, besteht diese Aktivität aus fünf elementaren Aktionen. Zuerst werden die Instanzen der Sensoren und Aktoren, bzw. deren Treibersoftware, die aus Gründen der Übersichtlichkeit jeweils zu einem Objekt zusammengefasst wurden, erzeugt. Anschließend werden die relevanten Prozesse der

Motorsteuerung und der Sensoren zur Überwachung von Messdaten gestartet und schließlich werden die Sensoren und Aktoren überprüft. Die meisten Aktivitäten in der Motorsteuerung bestehen meist nur aus einer elementaren Aktion und es ist deshalb wenig sinnvoll, diese in separaten Aktivitätsdiagrammen darzustellen. Im Beispiel der Initialisierung des Systems wird ein Prozess ausgeführt, der an keiner Stelle auf externe oder interne Ereignisse reagiert und daher auch keine Zustandsänderungen nötig sind. Es ist an dieser Stelle also sinnvoll, den Ablauf mit einer komplexeren Aktivität zu beschreiben. Diese Aktivität könnte besser in einem separaten Aktivitätsdiagramm dargestellt werden.

Wird der Zündschlüssel von II nach III bewegt, wird der Bewegungszustand des Motors auf Start gesetzt und der aktive Zustand ist anschließend „Startup“, falls der von den Aktoren zurückgelieferte Fehlercode (siehe Abbildung 6) kleiner oder gleich dem Fehler-Threshold ist. Nachdem der Zündschlüssel zurückgedreht wurde, wird der Drehzahlsensor überprüft, ob der Motor erfolgreich gestartet werden konnte und in den Composite State „Betrieb“ gewechselt. Auffällig ist hier, dass die vom Zustand „Betrieb“ ausgehenden Transition zum Zustand „Fehler“ die Abfrage eines Wertes nicht wie ausgehend von Zustand „Startup“ durch den Guard „Motordrehzahl = 0“, sondern durch ein gleichlautendes Ereignis ausgelöst wird. Dies ist durch die Semantik der Statecharts zu erklären, denn besitzt eine ausgehende Transition keinen Trigger, so wird der Guard beim Betreten des Zustands überprüft und gegebenenfalls schaltet die Transition. Ist jedoch der Guard nicht erfüllt, so wird er bis der Zustand verlassen und neu betreten wird nicht wieder überprüft. Deshalb muss an dieser Stelle der Drehzahlsensor mit einem entsprechenden Ereignis melden, dass der Motor stehen geblieben ist. Diese Problematik, dass zum einen die Motorsteuerung nicht aktiv ununterbrochen Messwerte überprüft, sondern die Sensoren entsprechende Ereignisse melden, zum anderen aber auch die Steuerung die Möglichkeit hat, Werte abzufragen, wenn sie benötigt werden, tritt an diversen Stellen im Modell auf.

Der Zustand „Fehler“ wurde eingeführt, um mögliche Schwierigkeiten beim Starten und beim Betrieb des Motors abzufangen. Hier werden bis zum Auftreten des Ereignisses „Zündschloss II>I“ alle weiteren Ereignisse ignoriert, um nach dem Beenden der relevanten Prozesse den Motor erneut starten zu können. Zusammen mit dem Zustand „Shutdown“ ist somit sichergestellt, dass diese Prozesse beendet werden.

3.2 Verfeinerung des Zustands „Betrieb“

Nachdem der Motor gestartet wurde, ist der Zustand „Betrieb“ aktiv, der aus zehn orthogonalen Regionen besteht, d. h. das System befindet sich zu jeder Zeit in zehn verschiedenen Zuständen, die unabhängig voneinander auf Eingaben reagieren können (siehe Abbildung 3). Wird der Zustand über eine der zwei ausgehenden Transitionen verlassen (siehe oben), so werden alle Prozesse in den orthogonalen Regionen ebenfalls abgebrochen. Der folgende Abschnitt soll die Aufgaben dieser Regionen erläutern und deren Abbildung im Statechart.

Grundsätzlich können die Regionen in zwei unterschiedliche Kategorien aufgeteilt werden. In den fünf Regionen auf der linken Seite werden Informationen von den Sensoren durch den jeweils aktiven Zustand gespeichert. Auf der rechten Seite

dagegen werden in Bezug auf die aktiven Zustände auf der linken Seite Parameter für die Aktoren des Motors berechnet.

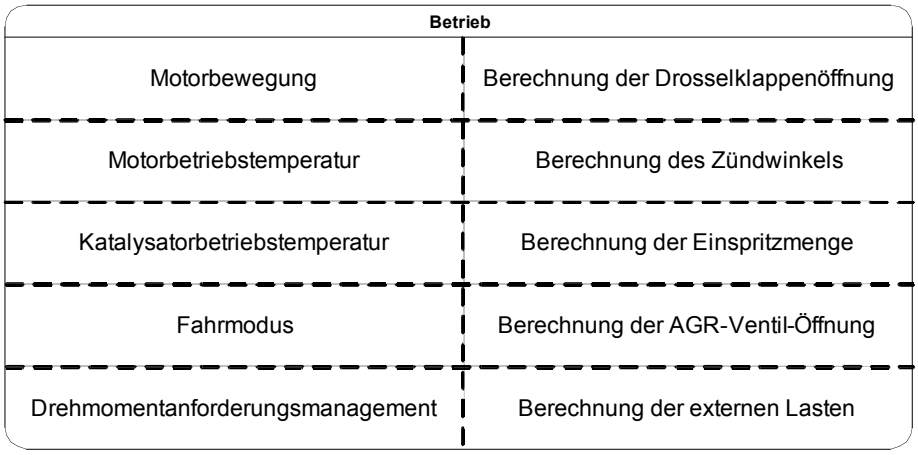


Abb. 3. Zustand „Betrieb" im Überblick

Zunächst aber zu den Regionen, die für die Sensoren zuständig sind. In der Region Motorbewegung (siehe Abbildung 8 im Anhang) ist zu Beginn der Zustand „Nachstart" aktiv. Beim Betreten des Zustands wird ein gradueller Übergang von Initialisierungsbedingungen zu Laufzeitbedingungen vollzogen. Nachdem die Nachstartzeit abgelaufen ist wechselt das System in den Zustand „Laufen", da die ausgehende Transition schaltet, sobald alle Aktivitäten abgearbeitet sind, da ihr keine Trigger zugewiesen sind. Im Composite State „Laufen", dessen Verfeinerung in diesem Fall graphisch innerhalb derselben Abbildung dargestellt ist und nicht durch zwei verbundene Kreise gekennzeichnet ist, ist zunächst der Zustand „Leerlauf" aktiv. Zum einen kann nun der Fahrer durch Einlegen eines Ganges und Schließen der Kupplung in den Schubbetrieb wechseln und das Fahrzeug setzt sich in Gang. Hierbei ist jedoch wichtig, sich im Klaren zu sein, dass nicht das Wechseln des Zustands in der Motorsteuerungssoftware das Fahrzeug in Gang setzt. Vielmehr liegt die Ursache beim Fahrer, der einen Gang einlegt und dabei die Steuerung des Getriebes, die hier nicht Gegenstand der Betrachtung ist, ein entsprechendes Signal an die Motorsteuerung liefert. Anschließend wechselt diese in den entsprechenden Zustand, dadurch wird bei der Berechnung der Parameter für die Aktoren auf andere Weise vorgegangen (siehe unten). Allgemein ist anzumerken, dass im gesamten Zustand „Laufen" keine einzige Aktivität ausgeführt wird, in dieser orthogonalen Region wird lediglich auf Sensoren des Gaspedals, bzw. des Tempomats und des Getriebes reagiert. Zusätzlich werden im Zustand „Teillast" die Messwerte des Saugrohrdrucks ausgewertet, um den Zustand in die Zustände „Normalbetrieb", „Beschleunigung und „Verzögerung" zu verfeinern. In den Zustand „Volllast" wechselt das System, sollte die Drosselklappe durch den entsprechenden Aktor vollständig geöffnet werden.

In den nächsten drei orthogonalen Regionen des Zustands „Betrieb" werden die Temperaturen des Motors und des Katalysators überwacht (siehe Abbildung 9). Überhitzt der Motor, erkennt ein Temperatursensor, dass ein zulässiger Grenzwert

überschritten wurde und die Motorsteuerung startet beim Betreten des Zustands „Überhitzung“ den Ventilator. Zusätzlich hat der Fahrer die Möglichkeit, den Fahrmodus über einen Schalter zu wählen.

In der Region „Drehmomentanforderungsmanagement“ (siehe Abbildung 10) werden diverse Sensoren ausgewertet, um die Prioritäten der Anforderungen zu regeln. In der Regel hat der Fahrer bzw. der Tempomat die Kontrolle über das Drehmoment. In einigen Ausnahmefällen, wie zum Beispiel das Überschreiten der Drehzahlbegrenzung, hat jedoch die Motorsteuerung die Möglichkeit, dem Fahrer die Kontrolle zu entziehen und entsprechend weniger Leistung am Motor einzustellen, um Schaden am Motor zu verhindern. Bei jeder Anforderung von Drehmoment wird der Composite State „Priorisierung“ verlassen und der Zustand eingestellt, der die höchste Priorität hat. Dies wird mit Hilfe einer Reihe von Entscheidungsknoten realisiert. Zuerst wird der Sensor mit der höchsten Priorität, die Anti-Schlupf-Regelung (ASR) überprüft. Meldet der ASR-Sensor ein Signal, so wird der „Schlupfzustand“ aktiv und den Berechnungsregionen wird gemeldet, dass die Priorisierung eingestellt ist. Ansonsten wird der nächste Sensor überprüft. Wird immer der Pfad mit dem Guard „else“ gewählt, ist am Ende der Zustand „Drehmomentdefault“ aktiv, was bedeutet, dass der Fahrer das Drehmoment bestimmt. Dadurch dass die Eingaben der Sensoren wieder mit Guards modelliert wurden und nicht mit Triggern, reagiert das System zum Beispiel nicht sofort auf das Überschreiten der Drehzahlbegrenzung, sondern erst bei der nächsten Anforderung von Drehmoment. Dies ist aber unwesentlich, da das Ereignis „Anforderung Drehmoment“ in so geringen Zeitabständen auftritt, wie man in den Regionen zur Berechnung sehen wird.

In den folgenden Regionen wird die eigentliche Hauptaufgabe einer Motorsteuerung ausgeführt. In jeder der folgenden Regionen wird ein Parameter für die Aktoren des Motors berechnet. In Abbildung 11 sind die Regionen „Berechnung der Drosselklappenöffnung“ und „Berechnung des Zündwinkels“ dargestellt. Die Berechnung der Einspritzmenge und der AGR-Ventil-Öffnung (siehe Abbildung 12) laufen nach demselben Schema ab und im Folgenden sollen diese Prozesse beispielhaft anhand der Berechnung der Drosselklappenöffnung erklärt werden.

Meldet ein Sensor, dass der Wert neu berechnet werden muss, wird zuerst in der Region „Drehmomentanforderungsmanagement“ eine Priorisierung vorgenommen. Da diese Berechnungen in der Regel einmal pro Motorumdrehung ausgeführt werden muss, wird der aktive Zustand in der Region regelmäßig aktualisiert. Danach wird der gesuchte Parameter, wie im Grundzustand berechnet, d. h. „Teillast“, „Drehmomentdefault“ und „Betriebstemperatur“. Anschließend wird der berechnete Wert in Abhängigkeit der Zustände in den anderen orthogonalen Regionen korrigiert. Um Bezug auf andere Regionen zu nehmen, wird in den Guards der ISIN-Operator verwendet. Dieser gibt den booleschen Wert wahr zurück, wenn der angegebene Zustand aktiv ist. Der Kreis schließt sich, wenn der Sensor erneut meldet, dass die Drosselklappenöffnung berechnet werden muss. Die Region „Berechnung der externen Lasten“ in Abbildung 3 wird in den Prozessdaten nicht näher spezifiziert und ist deshalb hier auch nicht modelliert.

Abbildung 13 zeigt ein Sequenzdiagramm zu einem beispielhaften Ablauf einer solchen Berechnung. Es ist zu sehen, dass diese Berechnung ohne Zutun des Fahrers im Hintergrund abläuft. Bei dem Versuch zu den Regionen, in denen die

Überwachung der Sensoren abläuft, ein Szenario darzustellen, wird deutlich, dass das Darstellen von Szenarios nicht immer sinnvoll sein muss. Da zu jedem Zeitpunkt beinahe jedes Ereignis auftreten kann und keine zeitliche Abfolge der Ereignisse spezifiziert werden kann, ist ein Szenario willkürlich und besteht nur aus nicht zusammenhängenden Nachrichten.

4 Modellierung von Varianten

„People can have the Model T in any color – so long as it's black."
(Henry Ford)

Was in den Anfängen des Automobilbaus und der Fließbandproduktion um 1920 üblich war ist heutzutage undenkbar. Beim Kauf eines Autos kann man nicht nur für eine Farbe entscheiden, sondern hat auch die Auswahl aus einer riesigen Palette an verschiedenen Ausstattungsmerkmalen. Viele dieser Ausstattungsmerkmale machen eine Veränderung der Motorsteuerungssoftware nötig. Zwar bleibt die Struktur des Gesamtprozesses erhalten, jedoch können meist recht kleine Änderungen an vielen Stellen im Prozessmodell auftreten. Ziel der Modellierung sollte es sein, das System mit all seinen Varianten vollständig zu beschreiben. Da beim Aufstellen von Modellen für jede mögliche Kombination an Ausstattungsmerkmalen die Zahl der Modelle exponentiell im Verhältnis zu den Varianten wächst, ist klar, dass dies nur eine unzureichende Lösung des Problems darstellt. Die verursachten Kosten würden in keinem Verhältnis zu den resultierenden Vorteilen stehen. Da der Gesamtprozess in einer Produktfamilie in großen Teilen erhalten bleibt, ist eine effizientere Lösung, die Varianten in den Gesamtprozess zu integrieren. Damit ein Modell eine Variante vollständig beschreiben kann, müssen nach Jacobson [14] und Coriat [7] folgende drei Aspekte abgedeckt sein:

- Die Lokalität der Teile des Modells, in denen sich die Variante auswirkt, d. h. an welcher Stelle eventuell Teile des Modells ersetzt oder ergänzt werden müssen, muss spezifiziert sein. Des Weiteren ist es notwendig, eine klare Schnittstelle zwischen den betroffenen Stellen und dem Rest des Modells zu definieren, die so genannten Variationspunkte.
- Alle möglichen Varianten, die an dieser Stelle eingesetzt werden können, müssen beschrieben werden. Dabei ist es möglich, die Modelle zum Beispiel durch Vererbung zu strukturieren. Schließlich wird jedoch nur eines der erstellten Modelle in den Gesamtprozess eingesetzt.
- Zuletzt muss es möglich sein, Bedingungen zu formulieren, an die die Auswahl bestimmter Varianten geknüpft ist. Dies kann zum einen die Auswahl einer oder mehrerer Varianten an anderer Stelle im Modell sein.

Inwiefern die Unified Modeling Language in der Version 2.0 Möglichkeiten bietet diese drei Aspekte zu beschreiben, soll im folgenden Abschnitt evaluiert werden. Hierzu werden Elemente des Metamodells vorgestellt, die in der Version 2.0

hinzugefügt wurden. Anschließend werden diese Neuerungen in zwei Beispielen angewendet.

4.1 Erweiterung der Notation in UML 2.0

Zur Beschreibung des ersten Aspekts, der Lokalität von Varianten, sind die in der Version 2.0 hinzugekommenen Submachine States geeignet. Dies sind Zustände, die im Gegensatz zu Composite States keine Subzustände oder Transitionen beinhalten, sondern eine Referenz auf einen Zustandsautomaten, der an dieser Stelle eingefügt wird. Die Abwicklungssemantik ist dieselbe wie bei einem Composite State, nur syntaktisch sind beide durch zwei kleine verbundene Kreise gekennzeichnet. Lediglich am Namen in der Titelzeile eines Zustandes sind sie zu unterscheiden. Beim Submachine State steht in der Titelzeile zuerst der Name des Zustands und anschließend, durch einen Doppelpunkt, getrennt der Name des referenzierten Zustandsautomats (siehe Abbildung 4).

Aber noch etwas unterscheidet sie von den Composite States. Während diese durch Transitionen außerhalb des Zustands zu beliebigen Zuständen innerhalb eines Composite States verbunden werden können, so ist dies bei einer Referenz zu einem Zustandsautomaten nicht möglich. Um diesem Problem zu begegnen, wurden zwei weitere Arten von Pseudozuständen eingeführt. Diese benannten Ein- und Austrittspunkte definieren die Stellen, an denen ein Zustandautomat betreten und verlassen werden kann. Dem Submachine State ist im Metamodell eine so genannte „ConnectionPointReference“ zugeordnet, die mit der Menge an Ein- und Austrittspunkten des referenzierten Zustands übereinstimmen muss. Damit ist es nun möglich, die Variante in einem separaten Zustandsautomaten zu kapseln und im Gesamtprozess verweist nur noch eine Referenz darauf. Durch die „ConnectionPointReference“ besteht eine Art Schnittstelle, die jede Variante, die an dieser Stelle eingesetzt werden soll, erfüllen muss. Die graphische Repräsentation dieser Pseudozustände, die an den Rändern des Zustands notiert werden ist in Abbildung 4 dargestellt, wobei die Austrittspunkte durch ein Kreuz von den Eintrittspunkten unterschieden werden kann.

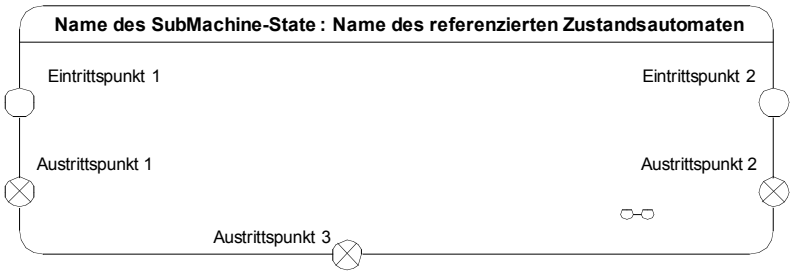


Abb. 4. Graphische Repräsentation eines Submachine States mit Ein- und Austrittspunkten

Der zweite Aspekt zur vollständigen Beschreibung von Varianten ist die Beschreibung aller möglichen Varianten. Hierzu können beliebig viele Zustandsautomaten modelliert werden mit der einzigen Bedingung, dass die Menge

der Ein- und Austrittspunkte mit der im Gesamtprozess übereinstimmt. Zusätzlich kann aber auch eine Struktur durch Vererbungsbeziehungen aufgebaut werden. So ist, wie viele Elemente in UML, auch der Zustandsautomat „generalizable“. D. h. ein Zustandsautomat für eine neue Variante kann die vorherige erweitern und somit zum Beispiel neue Zustände, weitere orthogonale Regionen oder Transitionen hinzufügen. Die Erweiterungsmöglichkeiten reichen bis zu Veränderungen vorhandener Zustände, vorausgesetzt, das Attribut „final“ des Zustands enthält den Wert „falsch“. Das Problem hierbei ist, dass die Spezifikation keine graphische Repräsentation für diesen Vererbungssachverhalt enthält. Erweiterte Zustandsautomaten erhalten lediglich das Schlüsselwort „extended“.

UML sieht die bei der Formulierung von Bedingungen die Object Constraint Language (OCL) vor, lässt dem Modellierer aber auch die Freiheit andere Sprachen bis hin zu natürlichen Sprachen zu wählen. OCL [18] wurde geschaffen, um Geschäftslogik, technische Anforderungen und Bedingungen in UML-Modellen zu formulieren [22]. Im Folgenden soll diese Sprache zur Beschreibung der Bedingungen, die die Auswahl der Varianten einschränkt, benutzt werden. Zur Begründung ist zu sagen, dass es wohl eine Frage des Verwendungszwecks der Modelle ist, ob eine formale oder natürliche Sprache zur Formulierung verwendet wird. Werden die Modelle nur von Menschen betrachtet, ist sicherlich die informelle Form sinnvoller, müssen die Diagramme jedoch von Maschinen gelesen werden, ist eine formale Spezifikation wohl unumgänglich.

4.2 Beispiel Wegfahrsperre

Nachdem nun die Syntax und Semantik der Zustandsautomaten erläutert wurde, soll nun eine beispielhafte Anwendung folgen. Eine mögliche Variante beim Starten des Motors ist die zusätzliche Überprüfung der Wegfahrsperre. Hierzu sind zwei neue Zustände notwendig, nämlich „Wegfahrsperre aktiviert“ und „Wegfahrsperre deaktiviert“. Diese sind unabhängig von den restlichen Zuständen beim Starten und wirken sich im bestehenden Zustandsautomaten nur beim Übergang von „Stopp (Zündung I)“ zu „Stopp (Zündung II)“. Um also das Verhalten zwischen diesen beiden Zuständen zu kapseln, wird der Submachine State „Stopp“ eingeführt. Dieser Zustand hat drei Eintritts- und zwei Austrittspunkte, die direkt aus dem Ursprungsmodell ergeben (siehe Abbildung 14). Der erste Aspekt einer Variante, die Lokalität und die Schnittstellen sind damit beschrieben.

Im nächsten Schritt werden die einzusetzenden Zustandsautomaten modelliert. Der in Abbildung 15 zu sehende Automat „StoppVariant“ entspricht dem Ursprungsmodell. Der Automat in Abbildung 16 erweitert den Automaten „StoppVariant“ um eine orthogonale Region und fügt der Transition zwischen „Stopp (Zündung I)“ und „Stopp (Zündung II)“ einen Guard hinzu. Um anzudeuten welche Elemente bei der Erweiterung hinzugekommen sind, werden diese mit gestrichelten Linien dargestellt. Nun kann der Name des referenzierten Zustandsautomaten beliebig gewählt werden, zudem ist sichergestellt, dass alle Automaten, die „StoppVariant“ erweitern, die entsprechenden Ein- und Austrittspunkte besitzen. Damit ist auch die Formulierung von Bedingungen, die die Auswahl einer Variante einschränken nicht nötig.

4.3 Beispiel Drehmomentanforderung

Nach der relativ problemlosen Modellierung nun zu einem komplexeren Beispiel, der Variabilität beim Drehmomentanforderderungsmanagement. Im Ursprungsmodell spielen diverse Sensoren eine Rolle, die nicht alle zwangsweise vorhanden sein müssen. Deshalb kann auch hier wieder mit Vererbung gearbeitet werden und die Variante, die nur die notwendigen Sensoren enthält, systematisch zu der Variante mit allen Sensoren erweitert werden. Hier soll jedoch der Schwerpunkt auf die Vielzahl der Stellen, an denen sich eine Variante auswirkt, liegen und deshalb sollen nur zwei Varianten entwickelt werden. Dies sind zum einen die Variante des Ausgangsmodells mit allen Sensoren und zum anderen eine Variante, in der die Sensoren für die Anti-Schlupfregelung und die Geschwindigkeitsbegrenzung nicht vorhanden sind.

Zunächst wird in der Region „Drehmomentanforderderungsmanagement“ ein Submachine State eingeführt, der die Priorisierung realisiert (siehe Abbildung 17). An dieser Stelle können die Automaten aus Abbildung 18 und 19 eingesetzt werden. Zusätzlich wirkt sich das Entfernen bestimmter Zustände in der zweiten Variante aber auch auf die zustandsabhängige Korrektur in zwei Berechnungszuständen aus. Wie in Abbildung 20 zu sehen ist, müssen hier ebenfalls Submachine States eingeführt werden. Die jeweils einzusetzenden Automaten unterscheiden sich hier lediglich in leicht modifizierten Guards (siehe Abbildung 21 und 22).

Bei dieser Art der Modellierung ist es nun notwendig, die Selektion der Varianten im Gesamtprozess zu beschränken, damit keine falschen Modelle entstehen können. Diese Bedingungen sind im Folgenden mit OCL notiert.

(Priorisierung.submachine = PrioVariant1) **implies** (**1**)
(KorrekturDrosselklappe.submachine = KorDKVariant1
and KorrekturZündwinkel.submachine = KorZWVariant1)

(Priorisierung.submachine = PrioVariant2) **implies** (**2**)
(KorrekturDrosselklappe.submachine = KorDKVariant2
and KorrekturZündwinkel.submachine = KorZWVariant2)

Durch die Ausdrücke (1) und (2), die für alle Modelle gelten müssen, wird sichergestellt, dass wenn im Submachine State „Priorisierung“ der Automat „PriorisierungVariant1“ referenziert wird, dass in den Berechnungsregionen der entsprechende Automat gewählt werden muss. Das Schlüsselwort „implies“ ist äquivalent zur Implikation als booleschen Operator.

In diesem Beispiel wurden nur zwei mögliche Varianten modelliert. Bei vollständiger Betrachtung dieses Sachverhalts wird aber klar, dass die drei Sensoren ASR, Geschwindigkeitsbegrenzung und der Klopfsensor unabhängig voneinander vorhanden oder nicht vorhanden sein können. Da sich aber all diese Sensoren an der gleichen Stelle im Statechart auswirken, muss an dieser Stelle für jede Kombination an Sensoren eine eigene Variante modelliert werden. Bei drei Sensoren sind dies bereits sieben verschiedene Varianten.

5 Bewertung der Sprache

Inwiefern sich Statecharts nach der Spezifikation 2.0 eignen, um technische Prozesse zu modellieren, soll Gegenstand dieses Kapitels sein. Zudem soll besonders auf die Eindeutigkeit der Syntax und Semantik sowie auf die Möglichkeit, Echtzeitanforderungen an das System zu modellieren, eingegangen werden. Im Allgemeinen sind Statecharts zur Beschreibung dieser Prozesse, die an vielen Stellen abhängig von externen Ereignissen sind, gut geeignet. Dadurch dass die Steuerungssoftware gleichzeitig auf diverse Ereignisse reagieren muss, sind vor allem die orthogonalen Regionen ein nützliches Konstrukt, wie an der Modellierung des Zustands „Betrieb“ deutlich wird.

Da bei der Modellierung der Schwerpunkt auf dem Objekt Motorsteuerung liegt und nicht die an das System angeschlossenen Sensoren und Aktoren, sind an einigen Punkten aussagekräftige Sequenzdiagramme nicht immer möglich. An Stellen, an denen die Abfolge der Kommunikation unter den Objekten im Vordergrund steht, ermöglichen ausgewählte Szenarios die Interaktion zu beschreiben. Dies ist insbesondere dann der Fall, wenn das Verhalten der anderen Objekte nicht mit einem Statechart spezifiziert ist. Problematisch ist die Modellierung von Szenarien an Stellen, an denen Zustände beteiligt sind, die eine Vielzahl von orthogonalen Regionen besitzen wie der Zustand „Betrieb“. Dadurch dass in jeder Region auf unterschiedliche Ereignisse gewartet wird, die meist weder kausal noch zeitlich zusammenhängen, ist die Auswahl der Ereignisse für ein Szenario willkürlich. Die Aussagekraft solcher Szenarios ist folglich unbedeutend.

5.1 Eindeutigkeit der Syntax und Semantik

Damit aus den erstellten Diagrammen eventuell Programmcode erzeugt werden kann, ist eine klar definierte Syntax und Semantik nötig. Die Syntax ist durch das in Abbildung 1 gezeigte Metamodell formal definiert. Die Semantik jedoch ist nicht formal, sondern nur in textlicher Form beschrieben. Dies lässt an manchen Stellen Uneindeutigkeiten bezüglich der Abwicklung solcher Diagramme zu. Um neben der Semantik der Spezifikation der OMG einen formalen Weg bei der Definition der Semantik zu gehen, gibt es in der Literatur diverse Ansätze [16][20][15]. Lattela et al. [16] wandelt die hierarchische Struktur in flache Strukturen um, um die Zustandsautomaten zu analysieren. Schäfer et al. [20] modellieren jeden Zustand mit einem eigenen Prozess. Diese Prozesse kommunizieren über Kanäle um die schaltbereiten Transitionen zu finden. Diese Variante erzeugt jedoch einen enormen Mehraufwand an Kommunikation. Jin [15] übersetzt den Zustandsautomaten in Object Mapping Automata, eine formale Sprache. Auf Basis dieser Sprache wird eine operationelle Semantik spezifiziert, die Transitionskonflikte löst und das Ausführen einer run-to-completion Semantik ermöglicht.

5.2 Echtzeitkriterien

Da die Motorsteuerung zum Teil Parameter berechnet, die je nach Drehzahl des Motors circa hundert Mal pro Sekunde aktualisiert werden müssen, müssen die Echtzeitanforderungen spezifiziert werden. Dies kann entweder separat geschehen oder aber im vorhandenen Prozessmodell mit modelliert werden. Während die Statecharts der UML keine Möglichkeit bieten, zeitliche Bedingungen zu formulieren, ist es in Sequenzdiagrammen möglich, einfache Aussagen über die Dauer bestimmter Abschnitte in der Sequenz anzugeben. Wie in Kapitel 2 bereits erwähnt, wurden in dieser Ausarbeitung die Sequenzdiagramme nur zur Darstellung ausgewählter Szenarien verwendet, deshalb ist die allgemeine Formulierung dieser zeitlichen Kriterien nicht möglich. Zur detaillierteren Modellierung von zeitlichen Beschränkungen und Bedingungen sieht die UML Timingdiagramme vor. Hier kann Bezug auf die Zustände eines Systems genommen werden und die zeitlichen Einschränkungen zum Beispiel beim Schalten einer Transition.

6 Zusammenfassung

Generell sind die Statecharts eine sehr ausdrucksstarke Modellierungssprache, mit der alle Aspekte einer Motorsteuerung modelliert werden können. Mit der Möglichkeit zur Dekomposition bleiben die Modelle übersichtlich und nach kurzer Einführung in die Syntax und Semantik leicht verständlich. Es wurde gezeigt, dass die Modellierung von Varianten bei einfachen Veränderungen dank der in UML der Version 2.0 neu eingeführten Submachine States und den zugehörigen Pseudozuständen problemlos möglich ist. Bei komplexeren Varianten müssen eventuell Auswahlbedingungen formuliert werden. Dies wurde am Beispiel des Drehmomentanforderungs-management mit der Object Constraint Language der OMG dargestellt. An diesem Beispiel ist aber auch zu sehen, dass Varianten, die sich an den gleichen Zuständen auswirken, nicht mehr unabhängig voneinander modelliert werden können und die Komplexität, vor allem aber auch die Quantität an Zustandsautomaten enorm zunehmen kann. Um dieses Problem zu bewältigen, ist Werkzeugunterstützung unumgänglich. Im letzten Kapitel wurde zudem gezeigt, dass die Eindeutigkeit in Syntax und Semantik der Zustandsautomaten durchaus gegeben ist. Dies ermöglicht Werkzeugunterstützung nicht nur bei der Verwaltung der einzelnen Varianten, sondern auch bei der Generierung von Quellcode direkt aus den Modellen. Bei dieser Art der Softwareentwicklung („model-driven development“) werden im Idealzustand nur noch Modelle erstellt und editiert. Das hat zur Folge, dass der Quellcode durch die automatische Generierung wesentlich weniger fehleranfällig ist, und somit eine Kostenreduktion erreicht werden kann.

Literaturverzeichnis

[1] Rémy Apfelbacher und Anne Rozinat, „Fundamental Modeling Concepts (FMC) Notation Reference“, Internet-URL: http://fmc.uni-potsdam.de, 2003.

[2] Joachim Bayer, Stefan Kettemann und Dirk Muthig, „Principles of Software Product Lines and Process Variants”, PESOA-Report No. 03/2004, 2004.

[3] Henrik Behrens, „Requirements Analysis and Prototyping using Scenarios and Statecharts”, 2002.

[4] Grady Booch, Jim Rumbaugh und Ivar Jacobson, „Unified Modeling Language User Guide”, Addison Wesley, 1997.

[5] Marc Born, Eckhardt Holz und Olaf Kath, „Softwareentwicklung mit UML 2”, Addison-Wesley, 2003.

[6] Matthias Clauß, „Untersuchung der Modellierung von Variabilität in UML”, Diplomarbeit, University of Dresden, 2001.

[7] Michel Coriat, Jean Jourdan und Fabien Boisbourdin, „The SPLIT method: building product lines for software-intensive systems”, Proceedings of the First International Software Product-Line Conference (SPLC-1), August 2000.

[8] Bruce Powel Douglass, „UML Statecharts”, Embedded Systems Programming, 1999.

[9] Carolyn Duby, „Implementing UML Statechart Diagrams”, Embedded Systems Conference, 2003.

[10] Martin Fowler, „UML Distilled”, Addison-Wesley, 2003

[11] David Harel, „Statecharts: A Visual Formalism for Complex Systems”, Science of Computer Programming 8, 1987.

[12] David Harel und Eran Gery, „Executable Object Modeling with Statecharts”, 18th International Conference on Software Engineering, Berlin, IEEE Press, 1996.

[13] David Harel und Michal Politi, „Modeling Reactive Systems with Statecharts: The Statemate Approach”, 1999.

[14] Ivar Jacobson, Martin Griss und Patrik Jonsson, „Software Reuse: Architecture, Process and Organization for Business Success”, Addison-Wesley, 1997

[15] Yan Jin, Robert Esser und Jörn W. Janneck, „Describing the Syntax and Semantics of UML Statecharts in a Heterogeneous Modelling Environment“, Proceedings of the 2nd International Conference on Theory and Application of Diagrams“, 2002

[16] Diego Latella, Istvan Majzik und Mieke Massink, „Towards a formal operational semantics of UML statechart diagrams“, In Proceedings of IFIP TC6/WG6.1 Third International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS’99), 1999.

[17] Object Management Group, „UML 2.0 Superstructure Specification”, Internet-URL: http://www.omg.org/cgi-bin/doc?ptc/2003-08-02, 2003.

[18] Object Management Group, „UML 2.0 OCL Specification”, Internet-URL: http://www.omg.org/cgi-bin/doc?ptc/2003-10-14, 2003.

[19] Jim Rumbaugh, Ivar Jacobson und Grady Booch, „Unified Modeling Language Reference Manual”, Addison Wesley, 1997.

[20] Timm. Schäfer, Alexander Knapp und Stephan Merz, „Model Checking UML State Machines and Collaborations“, In Electronic Notes in Theoretical Computer Science, volume 47, Elsevier Science B. V., 2001.

[21] Arnd Schnieders Frank Puhlmann und Mathias Weske, „Process Modeling Techniques”, PESOA-Report No. 01/2004, 2004.

[22] Jos Warmer und Anneke Kleppe, „The object constraint language: precise modeling with UML”, Addison-Wesley, 1998.

Anhang

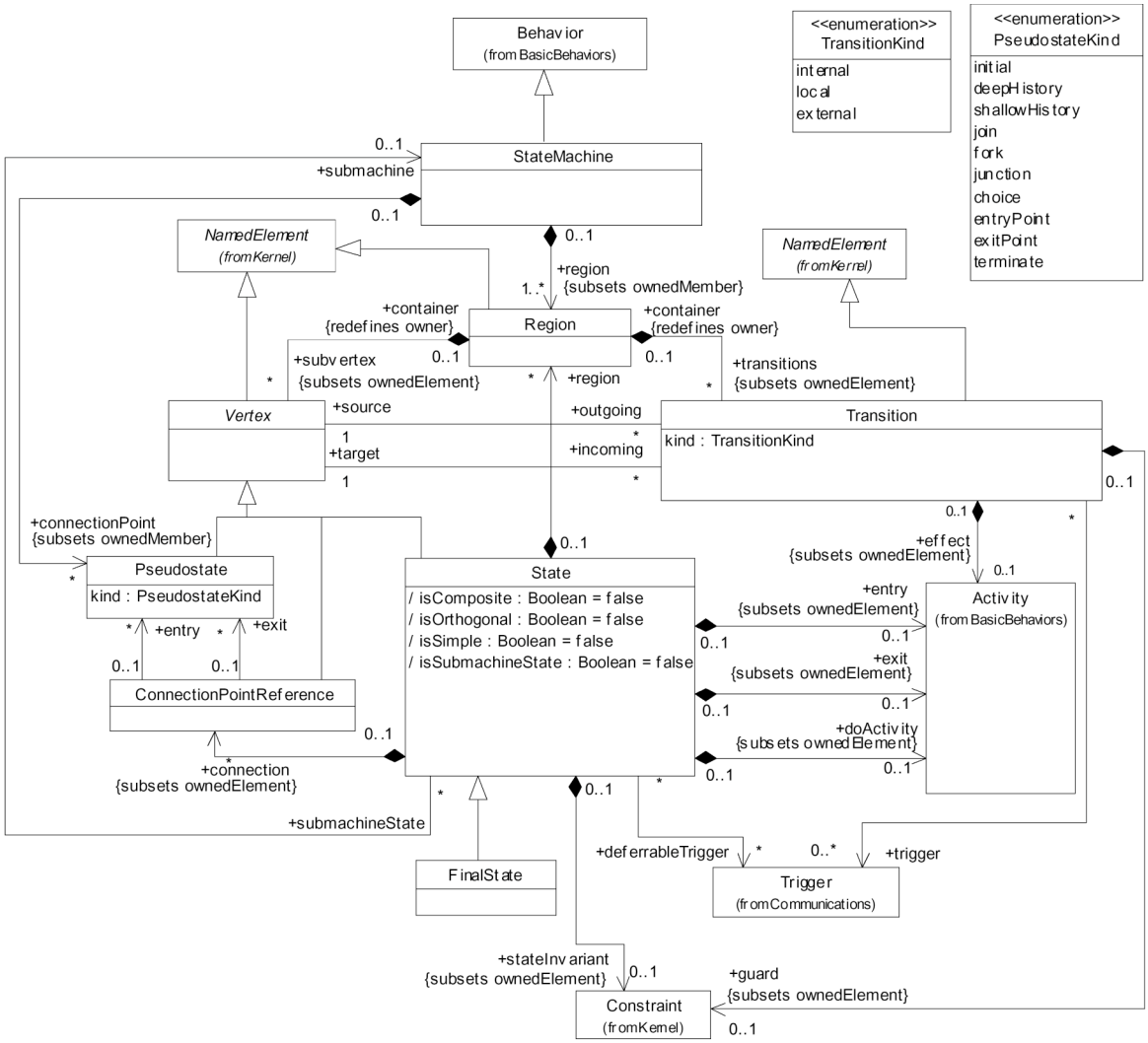


Abb. 5. Metamodell der Statecharts nach UML 2.0 (entnommen aus [OMG 03a])

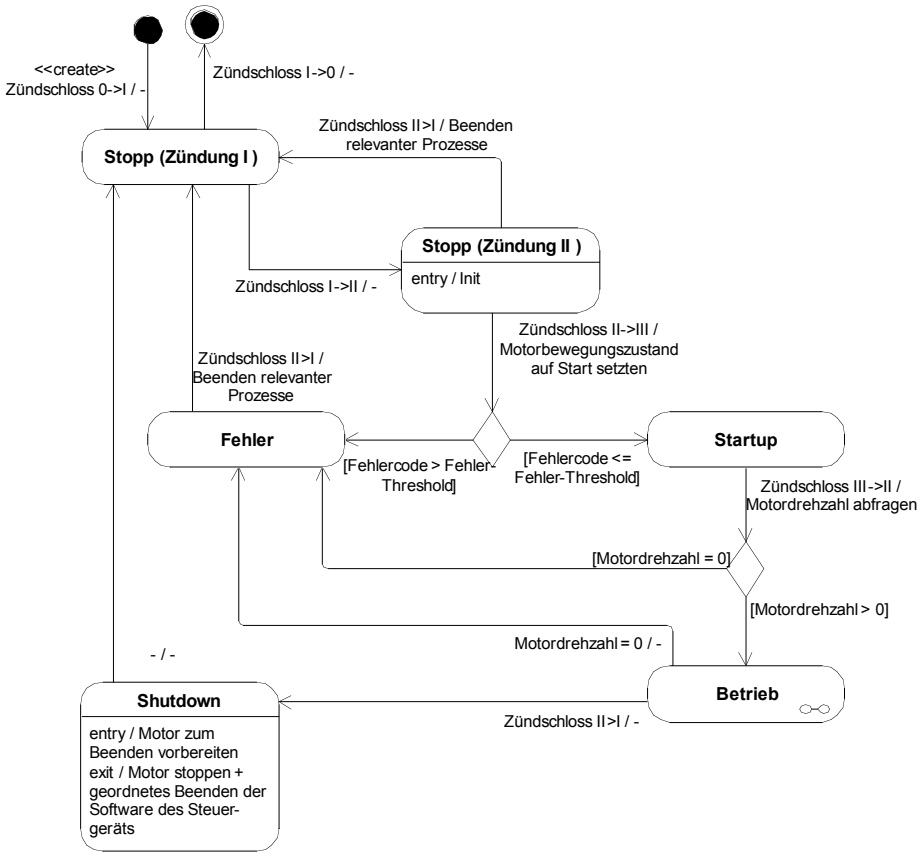


Abb. 6. Toplevel-Prozess der Motorsteuerung

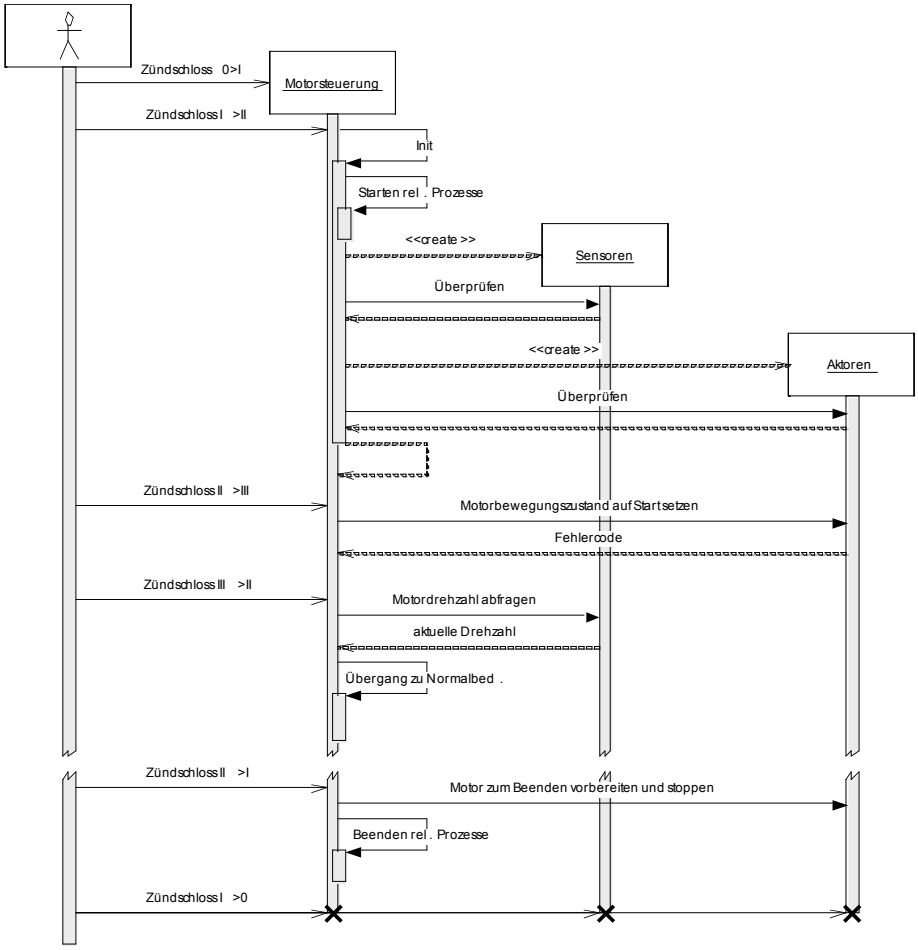


Abb. 7. Starten und Beenden der Motorsteuerung im Sequenzdiagramm

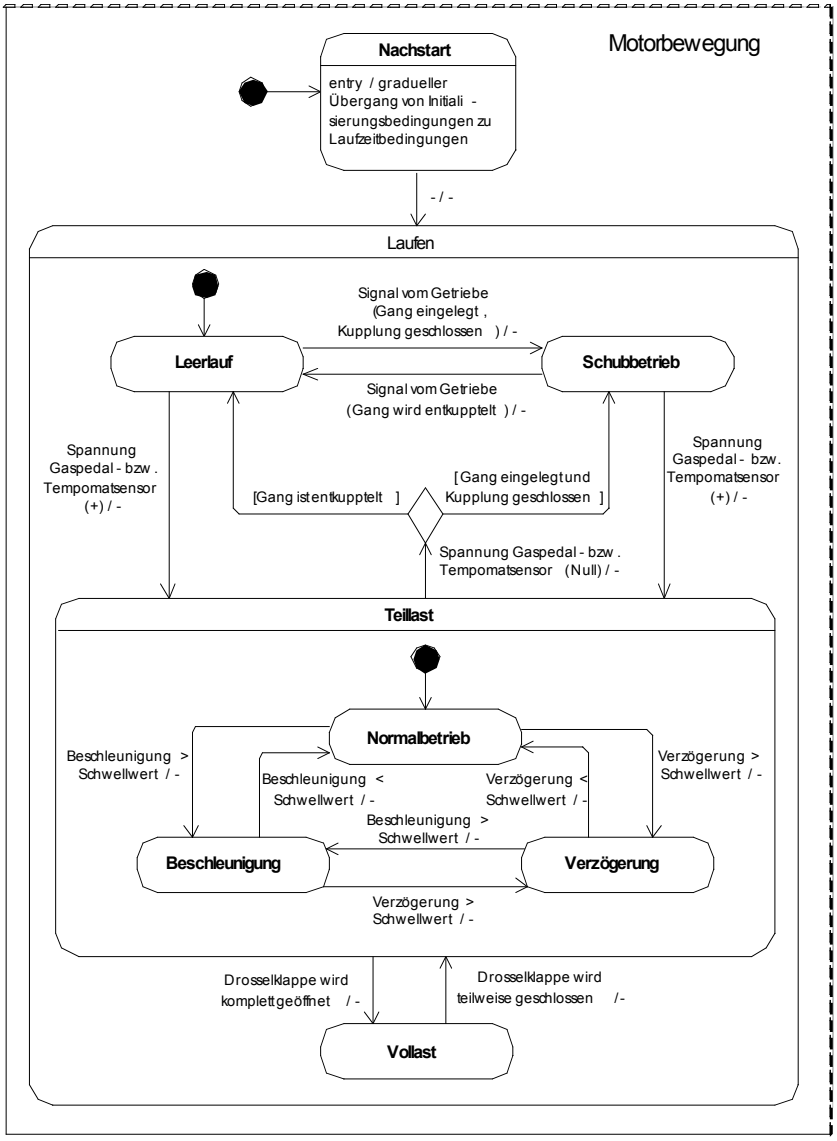


Abb. 8. Region „Motorbewegung" des Zustands „Betrieb"

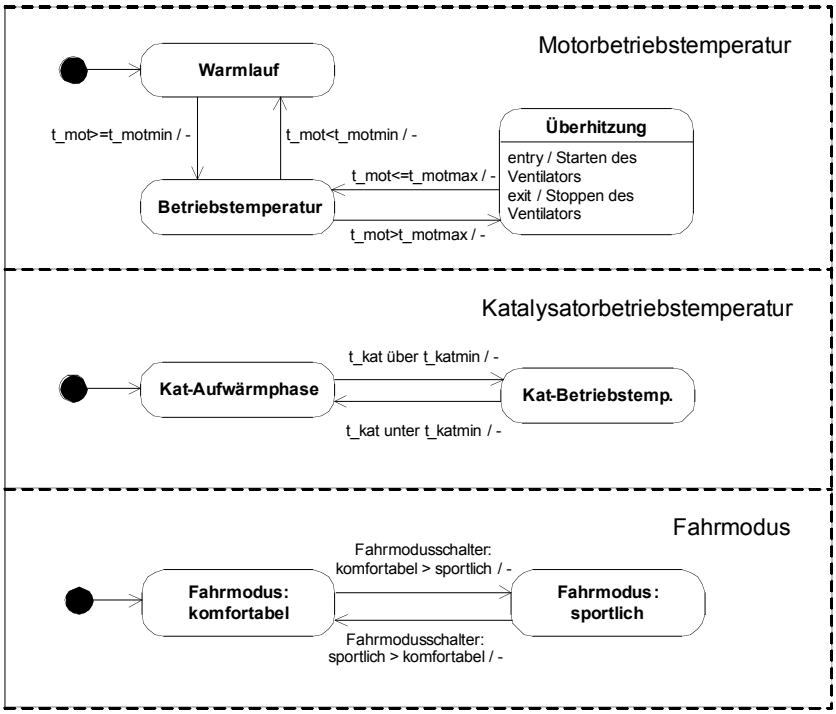


Fig. 9. Regionen „Motorbetriebstemperatur", „Katalysatortemperatur" und „Fahrmodus"

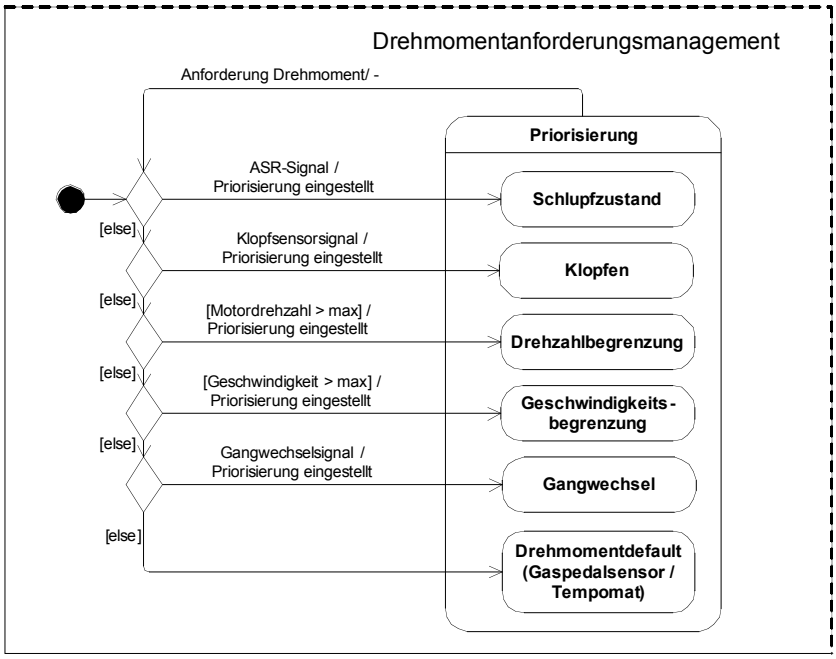


Abb. 10. Region „Drehmomentanforderungsmanagement" des Zustands „Betrieb"

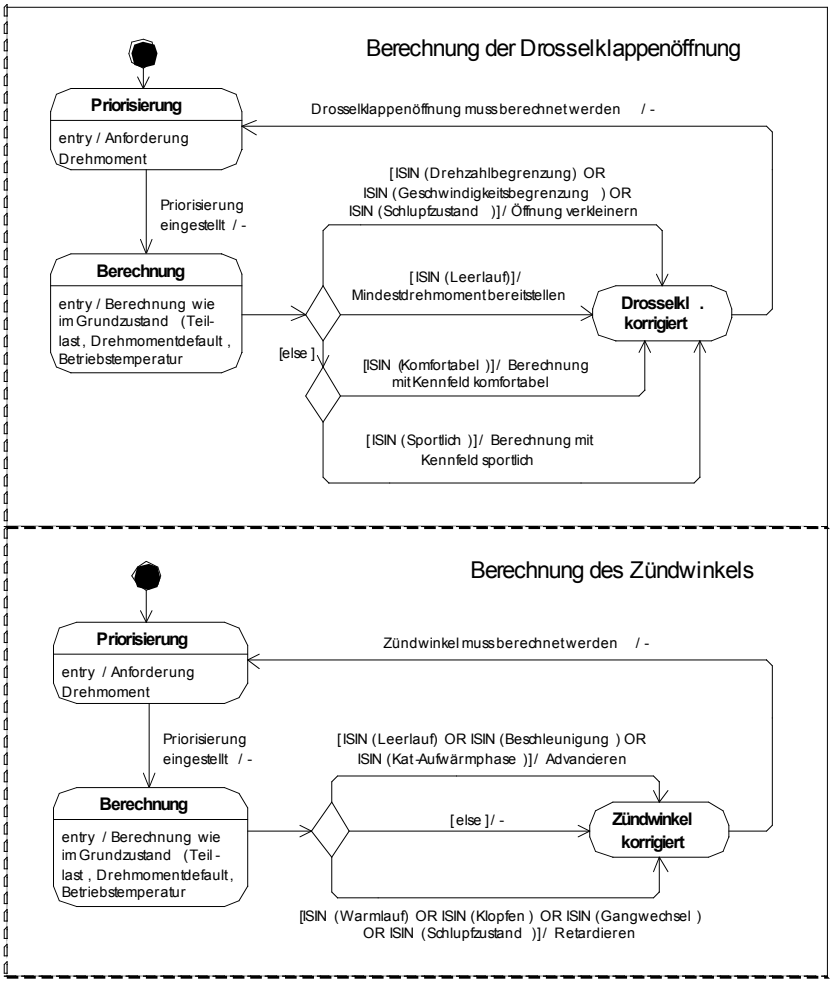


Abb. 11. Regionen zur Berechnung der Drosselklappenöffnung und des Zündwinkels

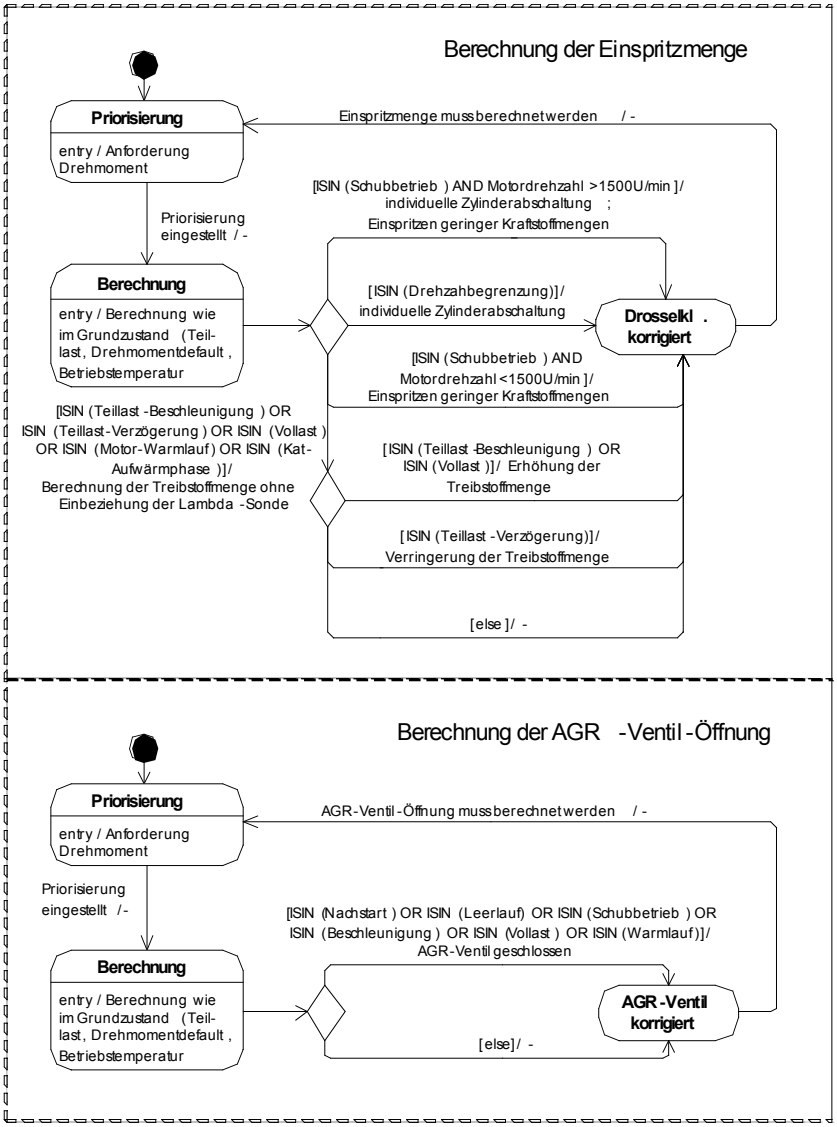


Abb. 12. Regionen zur Berechnung der Einspritzmenge und der AGR-Ventil-Öffnung

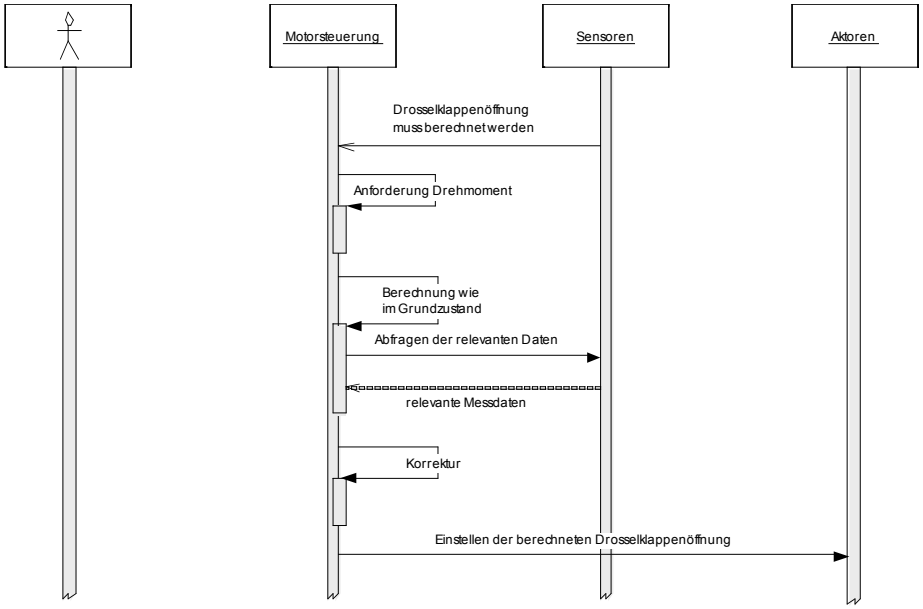


Abb. 13. Berechnung der Drosselklappenöffnung im Sequenzdiagramm

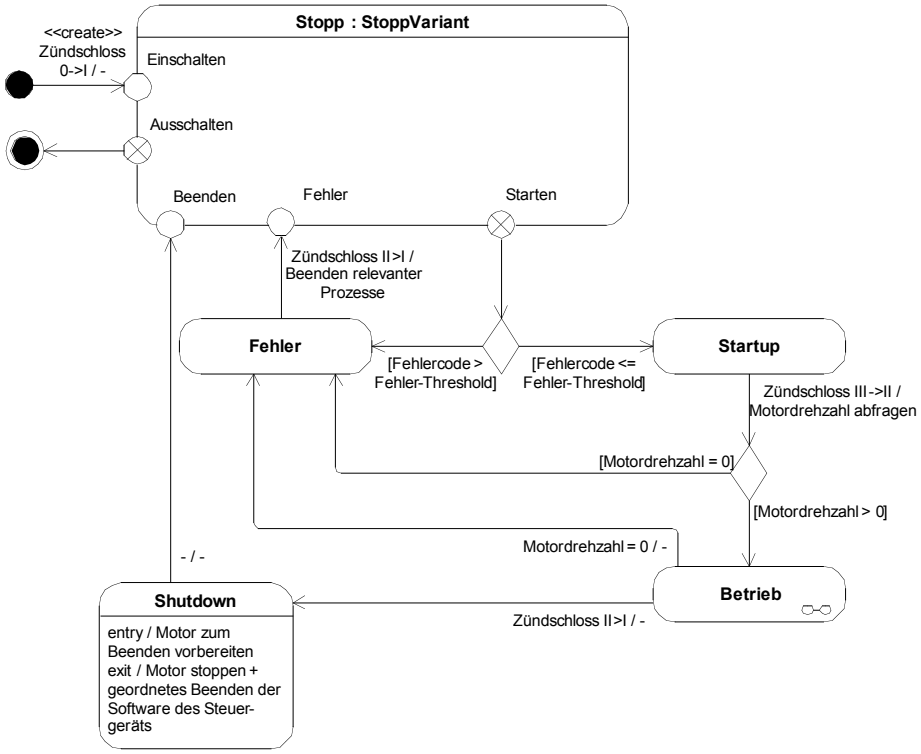


Abb. 14. Toplevel-Prozess der Motorsteuerung mit Submachine State „Stopp“

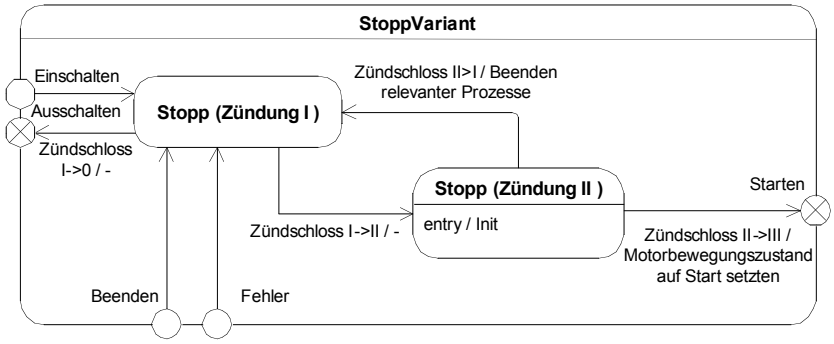


Abb. 15. Statechart für Variante ohne Wegfahrsperre

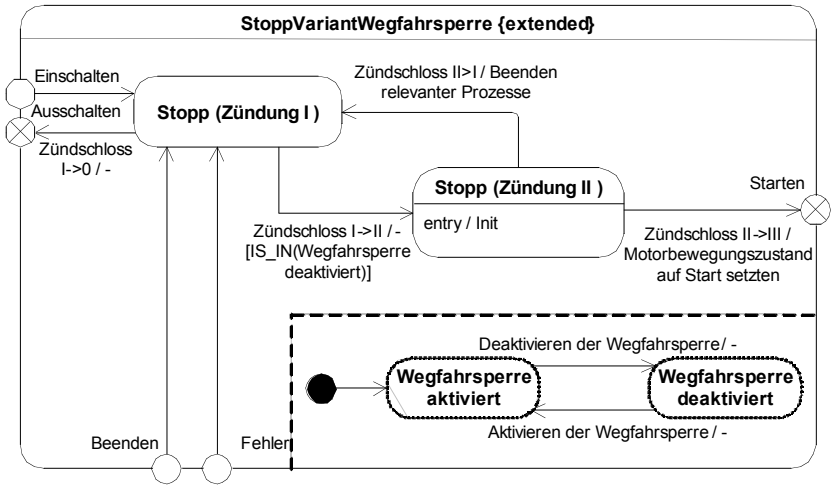


Abb. 16. Statechart für Variante mit Wegfahrsperre

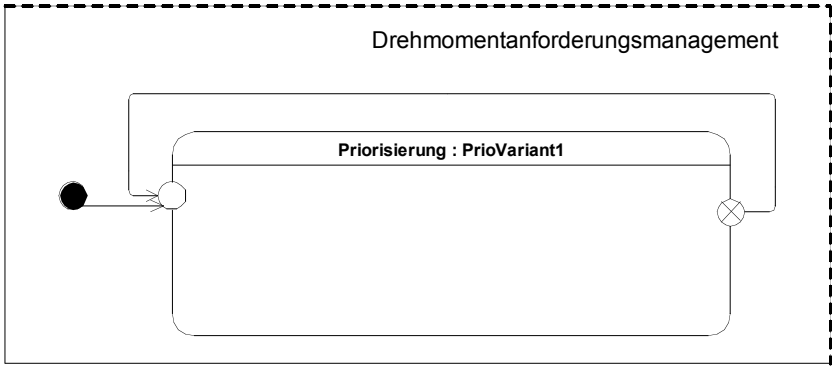


Abb. 17. Region „Drehmomentanforderungsmanagement" mit Submachine State „Priorisierung"

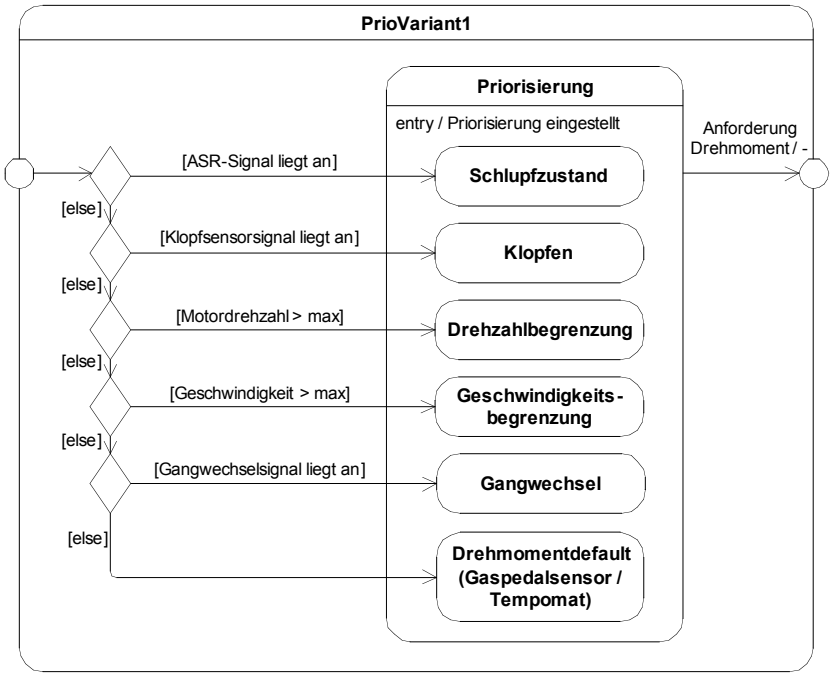


Abb. 18. Statechart für ursprünglicher Variante allen Sensoren

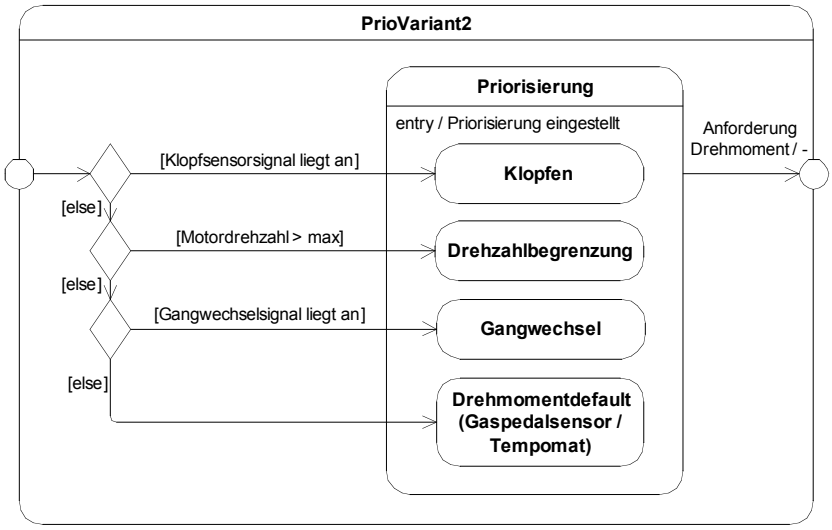


Abb. 19. Statechart für Variante ohne ASR und Geschwindigkeitsbegrenzung

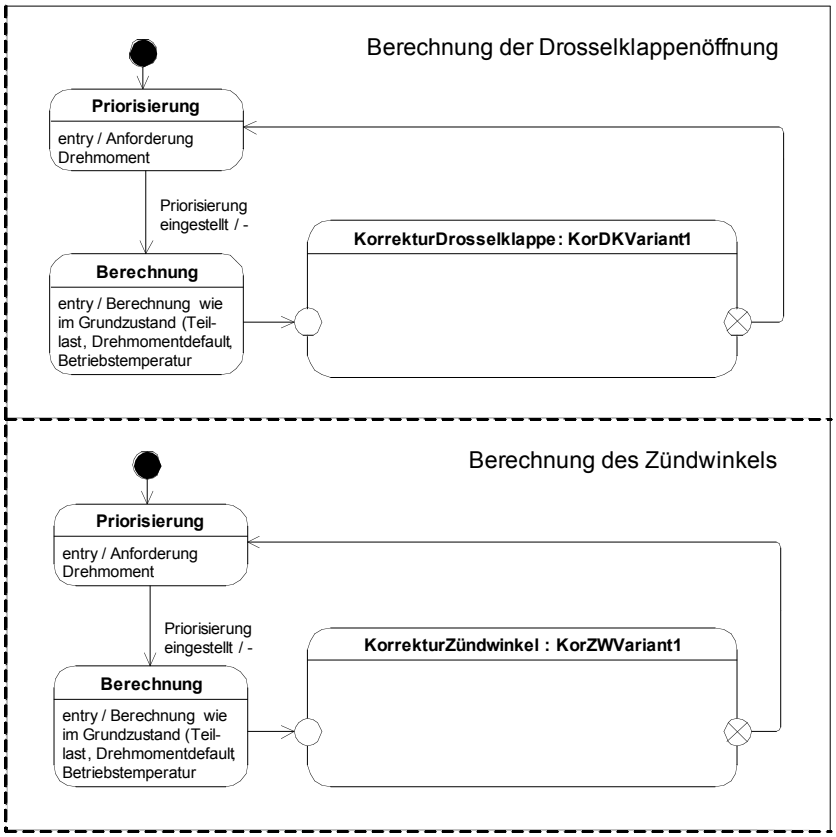


Abb. 20. Regionen zur Berechnung der Drosselklappenöffnung und des Zündwinkels mit Submachine State zur Korrektur der Berechnung

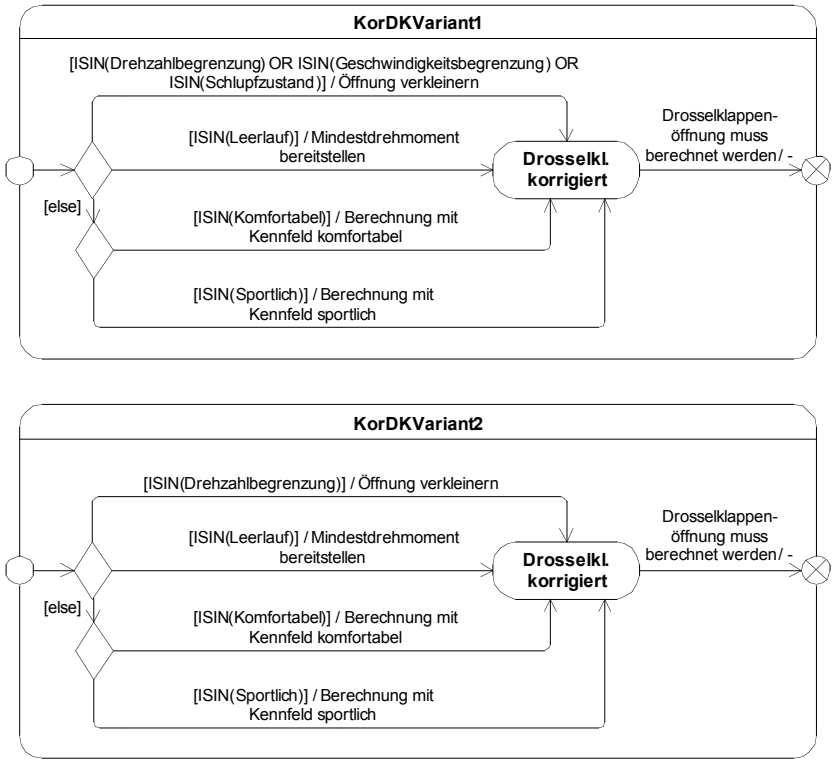


Abb. 21. Statecharts der unterschiedlichen Varianten für die Korrektur der Drosselklappenöffnung

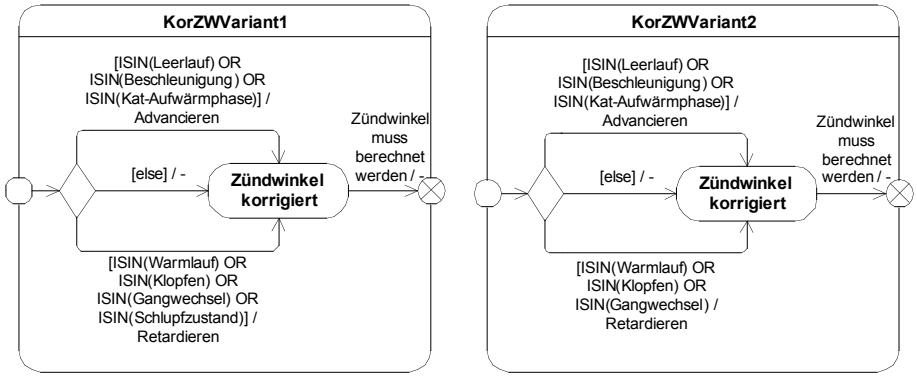


Fig. 22. Statecharts der unterschiedlichen Varianten für die Korrektur des Zündwinkels

Technische Prozesse – Motorsteuerung 2 (UML 2.0 Activity Diagrams)

Anja Bog

Anja.Bog@hpi.uni-potsdam.de

Abstract. Diese Ausarbeitung befasst sich mit der Modellierung von technischen Prozessen mit UML 2.0 Aktivitätsdiagrammen. Anhand eines Beispiels, welches vollständig modelliert wird, soll die Eignung der Aktivitätsdiagramme für eine solche Aufgabe festgestellt werden. Fortführend soll nicht nur die Modellierung der reinen Prozesse betrachtet werden, sondern auch die Fähigkeit der Sprache eventuelle Varianten dieser Prozesse geeignet darzustellen. Es werden zwei verschiedene Möglichkeiten zur Darstellung von Prozessvarianten vorgestellt und bewertet.

1 Einleitung

Die Unified Modeling Language (UML) ist ein Standard, um die Struktur und das Verhalten eines Softwaresystems zu beschreiben. Aktivitätsdiagramme sind ein Teil der UML Spezifikation. Ihre Aufgabe ist es Features bereitzustellen, mit denen die dynamischen Aspekte eines Systems dargestellt werden können. Sie eignen sich folglich zur Darstellung von Prozessen aus den verschiedensten Bereichen. Dazu zählen Prozesse aus der Produktion, Verwaltung und im E-Business. In dieser Ausarbeitung wird die Modellierung eines technischen Prozesses, genauer der Prozess zur Steuerung eines Automotors, vorgestellt. Anhand dieses Prozesses wird analysiert, inwieweit Aktivitätsdiagramme für die Modellierung entsprechender Prozesse geeignet sind. Im Einzelnen werden auch Varianten in dem genannten Prozess lokalisiert und Versuche unternommen, diese zu modellieren. Insbesondere sollen die Aktivitätsdiagramme aus der Version UML 2.0 erforscht werden. Dabei stellt sich heraus, dass in dieser Version, im Vergleich zu der alten Version, grundlegende Änderungen vorgenommen wurden. Im ersten Abschnitt erfolgt somit eine Einleitung in jene Änderungen, die sich ergeben haben und teilweise für diese Ausarbeitung von Belang sind. Anschließend wird der darzustellende Prozess kurz charakterisiert, sowie die Herangehensweise beschrieben, diesen im Endeffekt in Diagrammen abzubilden. Darauf folgen die Vorstellung der fertig modellierten Prozesse und die Beschreibung der Varianten, welche in diesen Prozessen versteckt sind. An geeigneten Stellen wird auf die Vor- und Nachteile dieser Sprache eingegangen und erläutert welche Schlussfolgerung sich daraus für die weitere Modellierung von Prozessen dieser Art ergeben.

2 UML 2.0 Aktivitätsdiagramme

Wie bereits erwähnt haben, sich die neuen Aktivitätsdiagramme im Vergleich zu ihrer alten Version stark gewandelt. Um im Weiteren möglichen Verwirrungen aus dem Weg zu gehen, werden in diesem Kapitel einige zum Teil für diese Ausarbeitung relevante und ein paar weitere interessante Änderungen beschrieben. Ein umfassenderer Überblick findet sich in [10].

2.1 Abstraktion

Die erste große Veränderung, welche Aktivitätsdiagramme durchlaufen haben ist die, dass die atomaren Knoten in den Diagrammen, ursprünglich bekannt als Aktivitäten, umbenannt wurden in Aktionen. Aktionen sind die atomaren Bestandteile in einem Aktivitätsdiagramm, sie spiegeln Tätigkeiten wider, welche nicht weiter verfeinert werden können. Der altbekannte Begriff der Aktivitäten bezeichnet nunmehr Strukturen auf einem höheren Abstraktionslevel, die sich wiederum aus Sequenzen von Aktionen und weiteren Aktivitäten zusammensetzen.

2.2 Automatisierung von Prozessen

Um die Automatisierung von Prozessen zu unterstützen, wurden neue Notationselemente, welche die Diagramme an einigen Stellen um diverse Attribute ergänzen, in den Sprachumfang aufgenommen. Diese Elemente beschreiben beispielsweise die für eine Aktion nötigen Vor- und Nachbedingungen (local pre-/postconditions), welche gelten müssen, bevor beziehungsweise nachdem die annotierte Aktion ausgeführt wird/wurde. Das folgende Beispiel soll den Sachverhalt demonstrieren:

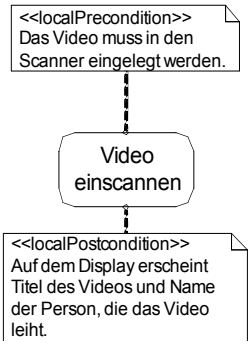


Fig. 1. Vor- und Nachbedingungen einer Aktion

Hier wird angenommen, dass das zu scannende Video in den Scanner eingelegt sein muss, bevor die Aktion „Video einscannen“ starten kann. Danach muss der Titel des Videos und der Name der Person auf einem Display angezeigt werden, damit die Aktion erfolgreich ausgeführt wurde.

2.3 Time Signal

Ein weiteres neues Notationselement ist das „Time Signal“. Hier kann eine Zeit spezifiziert werden, die ablaufen muss, bevor eine nachfolgende Aktivität starten kann. Im Prinzip wird der Kontrollfluss solange aufgehalten, bis die Zeit abgelaufen ist und dann wird das Kontrollflusstoken weitergereicht.

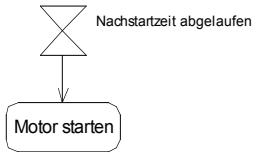


Fig. 2. Time Signal Notation

Hat ein Time Signal wie in diesem Beispiel keine eingehenden Kontrollflusskanten, so läuft die Zeit ab und die nachfolgende Aktivität wird gestartet, ohne dass erst auf einen Kontrollfluss gewartet werden muss, der das Ablaufen der Zeit anstößt.

2.4 Implizite Joins

Die letzte und wahrscheinlich tiefgreifendste Änderung, welche an dieser Stelle besprochen werden soll, ist die Interpretation von mehrfach eingehenden Kontrollflusskanten in eine Aktion, siehe Abbildung 3 (a). In der alten Version der Aktivitätsdiagramme, zu sehen in Abbildung 3 (b), war dieses gleichzusetzen mit einem „Or-Join“ am Eingang der Aktion, was bedeutet, dass nur eine der eingehenden Kanten einen Kontrollfluss aufweisen musste, damit die Aktion ausgeführt werden konnte.

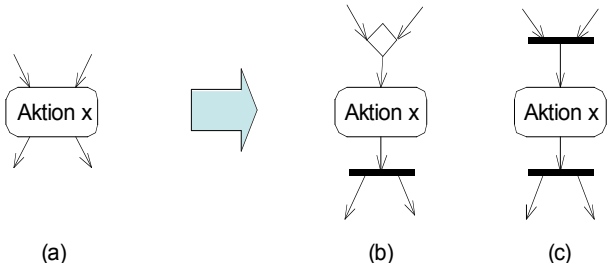


Fig. 3. Implizites Or-Join / And-Join

In der neuen Version der Aktivitätsdiagramme (Abbildung 3 (c)) wird ein solches Konstrukt nun aber als „And-Join” interpretiert, das heißt alle eingehenden Kanten müssen einen Kontrollfluss aufweisen, bevor die Aktion ausgeführt werden kann. Vor allem diese Änderung ist nicht sonderlich gelungen, da an dieser Stelle eine Rückwärtskompatibilität zu den Aktivitätsdiagrammen der alten Version, welche die implizite Notation nutzen, ausgeschlossen wird. Da an Diagrammen zumeist nicht

angemerkt ist, mit welcher Version von UML sie modelliert wurden kann dies zu schwerwiegenden Problemen führen, denn ausgehend vom Interpretationsstandpunkt der neuen Version weisen nun die alten Diagramme an einigen Stellen Deadlocks auf. Das nachfolgende Diagramm verdeutlicht eine solche Deadlock-Situation:

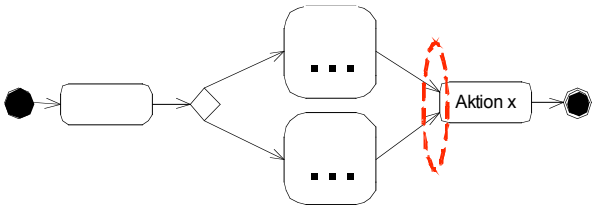


Fig. 4. Deadlock aus Sicht der Aktivitätsdiagramme 2.0

Der Vollständigkeit halber sei gesagt, dass mehrfach aus einer Aktion ausgehende Kanten in beiden Versionen der Aktivitätsdiagramme als „And-Split“, interpretiert werden.

Im Allgemeinen ist es wahrscheinlich besser, die implizite Notation überhaupt nicht zu verwenden, da sie, wie hier zu sehen war, zu Missverständnissen führen kann. Außerdem kann die inkonsistente Verwendung von sowohl impliziter, als auch expliziter Notation in einem Diagramm zu Interpretationsschwierigkeiten führen. Aus diesem Grund wurde bei der Modellierung der folgenden Prozesse darauf geachtet, nur die explizite Notation zu verwenden. (Dies ist auch dahingehend sinnvoll, falls der Standard nochmals überarbeitet und der impliziten Notation wieder eine vollkommen neue Interpretation zugeordnet wird.)

3 Modellierung der Motorsteuerung

In den nachfolgenden Sektionen dieser Ausarbeitung werden die modellierten Abläufe der Motorsteuerung vorgestellt. Dazu wird zunächst kurz die allgemeine Herangehensweise an das Gesamtproblem geschildert. Davon ausgehend werden danach die Motorsteuerungsprozesse beschrieben.

3.1 Herangehensweise

Die Informationen zu dem Prozess der Motorsteuerung waren gegeben als Zustandstabellen [1]. Ein logischer Ansatz, welcher sich aus dieser Form von Information ergibt, war es zuerst die Zustände in welchen sich die Motorsteuerung befinden kann, grob zu identifizieren. Nachdem die Zustände erforscht waren, war der nächste Schritt, diesen einzelnen Zuständen in einem Zustandsdiagramm Zusammenhang zu geben, das heißt die Bedingungen, welche zu Zustandsübergängen führen, mussten analysiert werden. Aus diesen Schritten ergab sich schließlich ein

Zustandsdiagramm, welches den Gesamtprozess auf höchster Ebene darstellt. Dieses Vorgehen entspricht dem Top-Down Ansatz.

Allerdings bestand nun das Problem, ausgehend von einem Zustandsdiagramm ein Aktivitätsdiagramm zu erstellen. Den Zuständen beziehungsweise Zustandsübergängen mussten nun Aktionen zugeordnet werden, die an den entsprechenden Stellen ausgeführt wurden. Da UML Zustandsdiagramme und Aktivitätsdiagramme eng miteinander verwoben sind, gibt es die Möglichkeit diese beiden Diagrammtypen miteinander zu verbinden [2, S.502 ff.]. Als Schlussfolgerung ergab sich daraus, dass die Aktionen und Signale, die bei Zustandübergängen stattfinden, beziehungsweise empfangen werden, in das Zustandsdiagramm einfügbar sind. Das daraus entstandene Diagramm befindet sich im Anhang A „Zustandsdiagramm: Motor starten und beenden“.

Im Groben sind in diesem Diagramm fünf Zustände zu erkennen „Motor aus“, „Stopp(1)“, „Stopp(2)“, „Startup“ und „Betrieb“. An den Transitionen sind die entsprechenden Signale und Aktionen kenntlich gemacht, welche zu Zustandsübergängen führen. Beispielsweise geht der Motor aus dem Zustand „Motor aus“ in den Zustand „Stopp(1)“ über, wenn das Signal eintrifft, dass der Zündschlüssel von Position 0 auf Position 1 gedreht wurde. Um ein Beispiel anzuführen, wo auch Aktionen bei Zustandsübergängen durchgeführt werden, betrachte man den Zustand „Startup”. Dieser Zustand wird verlassen, wenn die Aktion „Motor starten“ ausgeführt wird. Danach wird auf das Eintreffen eines Signals gewartet. Trifft das Signal ein und der Motor ist bei dem gesamten Vorgang nicht abgestorben, so wird in den Zustand „Betrieb“ übergegangen. Anderenfalls ist der Folgezustand von „Startup“ der Zustand „Stopp(2)“.

Aktionen können allerdings nicht nur an den Transitionen zwischen zwei Zuständen stattfinden, es gibt Außerdem noch Aktivitäten und Aktionen, die innerhalb eines Zustands ausgeführt werden. Diese sind in dem besprochenen Diagramm nicht gekennzeichnet, es sei aber gesagt, dass der Zustand Betrieb ein komplexer Zustand ist und während sich der Motor in diesem Zustand befindet, gewisse Aktionen ausgeführt werden. Diese werden in weiteren Diagrammen klarer.

3.2 Prozesse, Aktivitätsdiagramme und Varianten

Aus dem eben besprochenen Zustandsdiagramm kann nun fast 1:1 ein Aktivitätsdiagramm abgeleitet werden. Zu beachten ist an dieser Stelle nur, dass nun auch die Aktionen und Aktivitäten, welche in Zuständen stattfinden, berücksichtigt werden müssen. Das daraus entstandene Diagramm ist im Anhang B „Aktivitätsdiagramm: Motor starten und beenden (Variante ohne Wegfahrsperre)“ zu finden.

3.2.1 Aktivität - Motor starten und beenden (Variante ohne Wegfahrsperre)

Im Folgenden soll der dargestellte Prozess kurz beschrieben werden. Die gesamte Aktivität wird aktiviert, sobald ein Signal „Drehen des Zündschlüssels von Position 0 auf 1“ empfangen wird. Nun kann der Fahrer den Schlüssel wieder zurückdrehen und der Prozess endet. Oder der Schlüssel wird weitergedreht (auf Position 2) und eine

Aktivität „Motorstart vorbereiten“ wird durchgeführt. In dieser Aktivität werden drei Aktionen parallel ausgeführt. Sind diese beendet, so ist auch „Motorstart vorbereiten“ abgeschlossen. Der Kontrollfluss verweilt an dieser Stelle bis ein weiteres Signal eintrifft. Durch das Weiterdrehen des Schlüssels auf Position 3 wird die Aktion „Motorbewegungszustand auf Start setzen“ durchgeführt. Nachfolgend wird abgefangen, ob eine gewisse Fehlerschwelle nicht überschritten wurde. Ist dies der Fall, so wird der Motor letztendlich gestartet. Wurde der Schlüssel nun wieder auf Position 2 zurückgedreht und ist der Motor dabei nicht irgendwann abgestorben, so befindet sich das System im Zustand „Betrieb“. Hier ist eine Aktivität erkennbar, die im Zustandsdiagramm keinen Platz fand, die Aktivität „Motor betreiben“. Diese Aktivität ist verfeinerbar und wird im Folgenden noch tiefergehend besprochen. Hier sei nur angemerkt, dass sie eine Endlosschleife ist, die auch irgendwann einmal wieder unterbrochen werden muss, da der Motor ansonsten ewig laufen würde. Unterbrechungen wären an dieser Stelle einerseits das Signal, dass der Fahrer den Schlüssel von Position 2 auf 1 gedreht hat und den Motor ausschalten will oder andererseits das Signal, dass der Motor im Betrieb abgestorben ist (beispielsweise ist der Sprit alle, oder der Fahrer hat die Kupplung springen lassen). Annotiert ist dieses Verhalten mit Hilfe der „Interruptable Region“. Tritt eines der Signale ein, so wird die Aktivität abgebrochen und der Kontrollfluss wird nach außen geleitet, wo er weiterverarbeitet werden kann.

An dieser Stelle ist vielleicht schon ein Nachteil dieser Notation erkennbar. Die gerade laufende Aktivität wird unterbrochen, egal in welchem Zustand sie sich gerade befindet. Bei sicherheitsrelevanten Systemen („System Security“) sollte darauf geachtet werden, dass nach dem Abbruch eventuelle Rollback-Maßnahmen im Sinne von einer „Ganz oder Garnicht“-Durchführung der unterbrochenen Aktivität durchgeführt werden, um die Sicherheit des Systems zu erhalten. Auch die Motorsteuerung führt entsprechend den Abbruchsignalen unterschiedliche Aktionen aus, um wieder in einen konsistenten Zustand zu gelangen. Die Sicherheitsmaßnahmen muss sich der Designer oder Entwickler an dieser Stelle selbst „bauen“, da die Sprache keine eigenen Konstrukte dafür anbietet.

Der in dem Diagramm grau unterlegte Bereich stellt einen Punkt dar, in dem der dargestellte Prozess variieren kann. Ein Auto kann eine Wegfahrsperre haben oder nicht. Der Fall, dass diese nicht vorhanden ist, wurde in dem eben besprochenen Diagramm gezeigt. Hier wurde bereits eine Kapselung der Aktionen, in denen sich die beiden Varianten unterscheiden, vorgenommen. Im Allgemeinen stellt die Kapselung ein spracheigenes Mittel der Aktivitätsdiagramme dar, um Prozessteile austauschbar zu halten. Wie allerdings den folgenden Ausführungen zu entnehmen sein wird, ist die Kapselung kein in allen Fällen geeignetes Mittel zur Darstellung von Prozessvarianten.

3.2.2 Aktivität – Motor starten und beenden (Variante mit Wegfahrsperre)

Zuvor soll zum Vergleich allerdings noch kurz der Prozess mit Wegfahrsperre vorgestellt werden. Das dazugehörige Diagramm befindet sich im Anhang C „Aktivitätsdiagramm: Motor starten und beenden (Variante mit Wegfahrsperre)“. Im Großen und Ganzen unterscheidet sich dieser Prozess von dem vorher besprochenen

nur in dem grau unterlegten Bereich. Es ist noch eine weitere Aktion hinzugekommen „Veranlassen der Prüfung der Wegfahrsperre“. Daraufhin wird entschieden, ob die Wegfahrsperre deaktiviert ist oder nicht. Ist sie deaktiviert, so verläuft der Prozess genau wie der Vorherige. Ist die Wegfahrsperre an dieser Stelle allerdings noch aktiviert, so kann der Motor nicht gestartet werden und der Folgezustand ist „Stopp(1)“. Genau hier ergibt sich das Problem, dass diese Variante nicht mehr so einfach gekapselt werden kann, wie die Vorherige, denn die vorhergehende gekapselte Aktivität hatte genau einen Ausgang. Die neue braucht zwei, nämlich einen für den Fall, dass die Wegfahrsperre aktiviert ist und einen für den Fall, dass sie deaktiviert ist.

3.2.3 Darstellung von Prozessvarianten via Kapselung

Die oben vorgeschlagene minimale Kapselung ist demnach nicht realisierbar, da einerseits die Schnittstellen der beiden gekapselten Aktivitäten unterschiedlich sind und es andererseits nicht möglich ist, eine Aktivität auf tieferer Ebene an unterschiedlichen Stellen zu verlassen und auf höherer Ebene noch festzustellen, wo genau sie auf unterer Ebene verlassen wurde, um darauf aufbauend „Routing-Entscheidungen“ zu treffen. Eine Aktivität kann zwar mehrere ausgehende Transitionen haben, aber in diesem Fall soll ja entschieden werden, welchen Weg das Kontrollflusstoken nehmen soll, das heißt, es müsste genau eine ausgehende Transition, welche zu einem XOR-Split führt, geben. Das XOR-Split müsste nun anhand einer Bedingung, die sich darauf bezieht, an welcher Stelle die vorherige Aktivität verlassen wurde, entscheiden, wo das Kontrollflusstoken entlang geschickt werden soll.

Trotz dieser Probleme soll hier ein möglicher Kapselungsvorschlag gegeben und diskutiert werden. Dieser Vorschlag ist in den Diagrammen im Anhang D – „Aktivitätsdiagramm: Motor starten und beenden (Kapselung der Varianten)“ modelliert. Zu sehen ist, dass der Zustand „Stopp(1)“ ebenfalls in die Kapselung eingebunden werden musste, um den Fall der nicht deaktivierten Wegfahrsperre und den darauf folgenden Zustandsübergang modellieren zu können. Das Problem, welches sich nun darstellt, ist, wie das Signal (Schlüssel 1 => 0), welches eigentlich den Gesamtprozess auf höherer Ebene terminiert, zurück nach außen geleitet werden soll. Die Lösung wird an dieser Stelle über Exception Edges getroffen: Jenes Signal löst auf unterer Ebene eine Exception aus, welche nach außen geleitet wird und deren Handler Body den Gesamtprozess letztendlich terminiert. Aus verschiedenen Gründen ist diese Lösung jedoch nicht optimal. Erstens ist sie nicht allgemeingültig, denn natürlich nicht jedes Problem kann auf diese Art und Weise mit Exceptions gelöst werden. Für jedes Problem muss eine individuelle Lösung gefunden werden. Zweitens wird hier ein Missbrauch der Exceptions betrieben, denn das Eintreffen des Signals ist eigentlich ein regulärer Teil des Programmablaufs und kein Fehlerfall, welcher via Exception behandelt werden muss. Diese Notation ist demzufolge vielleicht ein wenig irreführend. Als letzter Punkt ist anzuführen, dass die verwendete Kapselung keinesfalls minimal ist, es werden auch Prozessteile gekapselt, die zu keiner Variante gehören, sondern immer anzutreffen sind.

3.2.4 Darstellung von Prozessvarianten – Zweiter Ansatz

Da die Kapselung nicht unbedingt einen guten Ansatz zur Darstellung von Prozessvarianten bietet, soll hier noch eine zweite Möglichkeit angeboten werden. Für diesen Ansatz ist es nicht notwendig, die Aktivitätsdiagramme um Sprachelemente zu erweitern, sondern es werden vorhandene Elemente auf eine neue Art genutzt. Diese Elemente sind die Notiz/Kommentar-Notationselemente auch „tagged values“ genannt. Die in [3] beschriebene Möglichkeit Varianten zu Modellieren stützt sich auf die Anwesenheit beziehungsweise Nichtanwesenheit von Merkmalen, genau wie es in diesem Prozess benötigt wird. Es werden neue „tagged values“ `<<variant>>`/`<<optional>>` eingeführt. Des Weiteren haben diese „tagged values“ zwei Attribute. Das Attribut `Binding time` gibt den Zeitpunkt an, wann entschieden wird, ob die Variante im System vorhanden ist oder nicht und `condition` gibt an, unter welchen Bedingungen die Variante vorkommt. Im Anhang E „Aktivitätsdiagramm: Teilprozess Wegfahrsperre“ befindet sich das zugehörige Diagramm.

Es ist nur ein Ausschnitt aus dem Gesamtprozess zu sehen. Markant an dieser Lösung ist, dass beide Varianten des Prozesses in einem Diagramm nebeneinander dargestellt werden können. Mit Hilfe der eben besprochenen Notation wird nun gekennzeichnet, welche Elemente im System bei bestimmten Varianten vorhanden sind und welche nicht. Im Falle der Wegfahrsperre wird beispielsweise zur Bauzeit des Systems (`binding time = development`) entschieden, welche Aktionen und Transitionen in dieser Variante vorkommen. Abhängig ist die Entscheidung weiterhin davon, ob eine Wegfahrsperre eingebaut ist, oder nicht (`condition = Wegfahrsperre (nicht) vorhanden`).

Im Allgemeinen stellt diese Modellierung von Varianten eine echte Alternative zur Kapselung dar, allerdings gibt es auch hier einige Kritikpunkte. Einerseite wird die Notation schnell recht unübersichtlich, sobald viele Varianten vorkommen, denn alle Elemente die zu einer Variante gehören müssen entsprechend gekennzeichnet sein. Auch in dem hier besprochenen Diagramm mussten Kürzungen zur Übersichtlichkeit vorgenommen werden. Weiterhin besteht die Möglichkeit, dass durch das Weglassen von Varianten vielleicht Deadlocks im System entstehen, die vorher nicht da waren. Der Systemdesigner muss dementsprechend mehr Aufwand in das Design stecken, damit solch ein Fall nicht eintritt. Ferner müssen die Bedingungen für die Varianten genau spezifiziert sein, das heißt auch eventuelle Abhängigkeiten zum Vorhandensein von anderen Varianten müssen berücksichtigt werden.

In den folgenden Prozessdiagrammen wird nur noch diese Notation für Variabilitätspunkte genutzt.

3.2.5 Aktivitäten „Motor betreiben“, „Laufzeitzustände überwachen“, „Drehmoment korrigieren“ und „Teillastzustände überwachen“

Das weitere Vorgehen zur Modellierung der restlichen Prozesse war nun die Verfeinerung der Aktivitäten. Wie in dem Gesamtprozess zu erkennen ist, gibt es nur eine Aktivität, welche verfeinert werden kann, die Aktivität „Motor betreiben“. Das zugehörige Diagramm ist in Anhang F „Aktivitätsdiagramm: Motor betreiben“ zu

finden. Hier ist nun die bereits besprochene Endlosschleife zu erkennen, da diese Aktivität keinen Terminationsknoten besitzt. „Motor betreiben“ gliedert sich in 3 Aktivitäten auf, welche parallel ausgeführt werden, zum einen ist das die Überwachung der Motorbetriebstemperatur und des Katalysators und zum anderen müssen die Laufzeitzustände des Motors überwacht werden. Ein Laufzeitzustand ist dabei beispielsweise der Zustand, dass der Fahrer auf das Gaspedal tritt und keinen Gang eingelegt hat. In diesem Fall befindet sich der Motor im Teillastzustand, allerdings bewegt sich das Auto dabei keinen Zentimeter durch eigenen Antrieb. Ist ein Gang eingelegt und der Fahrer tritt auf das Gaspedal, so kann sich der Motor in demselben Zustand befinden. Alle jene Aktivitäten und die dazugehörigen Abhängigkeiten sind in dem Diagramm im Anhang G dargestellt. Auffällig ist hier weiterhin, dass in den verschiedenen Zuständen wiederholt ein und dieselbe Aktivität „Drehmoment korrigieren“ ausgeführt werden kann, sobald das Signal für eine Drehmomentanforderung eintrifft.

Auf diese Aktivität soll ein wenig näher eingegangen werden, da sie weitere Prozessvarianten enthält, welche mit Hilfe der Notation aus [3] modelliert worden sind. Das Diagramm zu dieser Aktivität befindet sich im Anhang H. Dargestellt sind hier zwei verschiedene Arten von Varianten. Zur ersten Art gehören diejenigen, deren Vorhandensein bereits zur Bauzeit des Systems entschieden wird (`binding time = development`). Dazu zählt beispielsweise die Aktivität „Klopfen überwachen“, da sie nur im System vorhanden sein kann, wenn auch ein Klopfsensor eingebaut wurde. Der Prozess enthält allerdings nicht nur Varianten, die sich zur Bauzeit des Systems äußern, denn wie bereits erwähnt, kann die Aktivität „Drehmoment korrigieren“ in verschiedenen Motorzuständen durchgeführt werden. Dann muss natürlich unterschieden werden, welche Aktionen in welchem Zustand überhaupt Sinn machen. Es macht im Zustand „Leerlauf“ zum Beispiel keinen Sinn, die Aktion „Geschwindigkeit begrenzen“ durchzuführen, da das Auto in diesem Zustand nicht angetrieben wird und somit auch keine Höchstgeschwindigkeit überschreiten kann. Diese Aktion ist mit dem „tagged value“ `optional` markiert, da hier zur Laufzeit (`binding time = runtime`) entschieden werden muss, ob die Aktion in dem Zustand, in welchem sich der Motor gerade befindet, ausführbar ist (`condition = IsIn Teillast || Volllast`).

Auf die anderen Varianten soll nicht weiter eingegangen werden, da sie nach analogem Schema aufgebaut sind. Allerdings sticht an dieser Stelle ein Kritikpunkt an der verwendeten Notation hervor. Einerseits ist die Darstellung aller Varianten in einem Diagramm zwar recht schön, da die Varianten nebeneinander zu sehen sind, andererseits wirkt es jedoch relativ komplex und ein grobes Verständnis des Ablaufs ist auf den ersten Blick nicht unbedingt gegeben, da recht viel Text/Pseudocode gelesen werden muss, um den Sachverhalt zu erkennen.

Um die Liste der Prozesse, welche zur Motorsteuerung gehören, zu vervollständigen, bleibt zu sagen, dass in der Aktivität „Laufzeitzustände überwachen“ noch eine Aktivität verfeinert werden kann. Diese Aktivität beschreibt das Überwachen der Teillastzustände (Anhang I). Auch hier gibt es noch eine Prozessvariante, die Wahl des Fahrmodus. Da dieser Teilprozess allerdings keine neuen Diskussionspunkte oder Probleme aufwirft, soll nicht weiter darauf eingegangen werden.

4 Bewertung und Zusammenfassung

Die Diagramme zu dem Prozess der Motorsteuerung wurden in den wichtigsten Punkten beschrieben und analysiert. Des Weiteren sind Probleme, die bei der Modellierung auftraten erläutert und Lösungsmöglichkeiten angegeben worden. In dieser Sektion soll nun darauf aufbauend eine Bewertung beziehungsweise kurze Zusammenfassung der Sprache der Aktivitätsdiagramme zur Modellierung von technischen Prozessen, sowie bezüglich allgemeinen Auffälligkeiten angeführt werden.

Beginnend ist ein natürlich nicht zu vernachlässigender Pluspunkt die weite Verbreitung von UML. Aktivitätsdiagramme als ein Teil dieser Modellierungssprache genießen folglich ebenfalls einen gewissen Bekanntheitsgrad. Die Bekanntheit von UML bringt eine breite Palette von Tools mit sich, welche UML nutzen. Dies wären beispielsweise Rational Rose, Microsoft Visio oder Telelogic Tau.

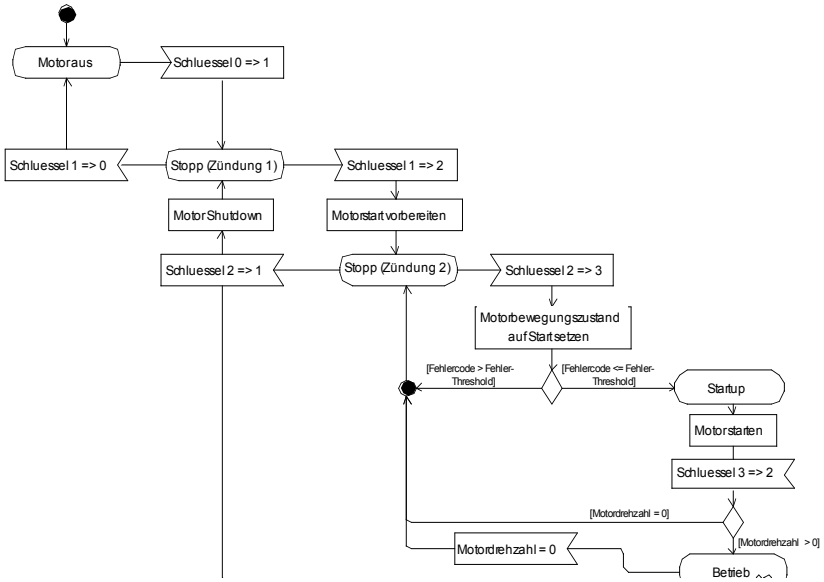
Bezüglich der Modellierung des technischen Prozesses stellt sich heraus, dass die Aktivitätsdiagramme eine recht gute Wahl sind. Es wird ein umfangreiches Repertoire von Sprachmitteln angeboten, um die verschiedensten Probleme darzustellen und zu lösen, vor allem das „Signal Handling“ war für den Prozess der Motorsteuerung ein guter Ansatz. In Kombination mit der Abstraktion, der Möglichkeit Aktivitäten auf tieferen Ebenen zu verfeinern, bieten die Signale einen leistungsfähigen Ansatz zur Modellierung von „Interruption Handling“. Nicht nur in diesem Zusammenhang ist die Einführung der Abstraktion ein Erfolg, sondern auch bezüglich der Übersichtlichkeit der Diagramme. Aktionen und Aktivitäten, die in einem „groben“ Sachverhalt auf hoher Ebene zu spezifisch erscheinen, können so auf eine tiefere Ebene gekapselt werden. Die Kapselung stellt sogar eine spracheigene Möglichkeit zur Modellierung von Prozessvarianten dar, indem einzelne Teilprozesse in Aktivitäten gekapselt und so austauschbar gehalten werden können. Jedoch ist dies, wie an dem Prozess der Motorsteuerung zu sehen war, nicht unbedingt die „einzig wahre“ Lösung, vor allem dann nicht, wenn sich die Teilprozesse nicht nur im Innenleben, sondern auch an den Schnittstellen unterscheiden. In diesem Fall muss auf eine andere Lösung zurückgegriffen werden, wie zum Beispiel die Möglichkeit der Variantenbeschreibung über „tagged values“ [3]. Dennoch hat auch dieser Ansatz seine Nachteile, denn das Darstellen der Variationspunkte kann aufgrund der Beschreibungen im Pseudocode ab einer bestimmten Anzahl von Varianten recht unübersichtlich werden. Des Weiteren ist keine genaue/umfassende Beschreibung dieser Notation gegeben. UML ist ebenfalls größtenteils in natürlicher Sprache verfasst und nur geringfügig formal spezifiziert, was die Beweisbarkeit der Modelle zum Teil verhindert. Weiterhin bietet UML selbst keine Modellierungsmöglichkeiten für Echtzeit- oder Sicherheitskritische Systeme an. Diesbezüglich muss der Entwickler/Designer auf die Tools vertrauen, welche zur Modellierung verwendet werden, denn einige davon bieten Erweiterungen an, um zum Beispiel Echtzeitsysteme zu modellieren (Rational Rose Real Time).

Letzten Endes bleibt zu sagen, dass die Aktivitätsdiagramme der Version 2.0 eine sehr ausdrucksstarke Sprache darstellen, aber an einigen Stellen doch Eigenschaften und Lücken (zum Beispiel die Darstellung von Varianten) aufweisen, die in einer nächsten Version Beachtung finden könnten/sollten.

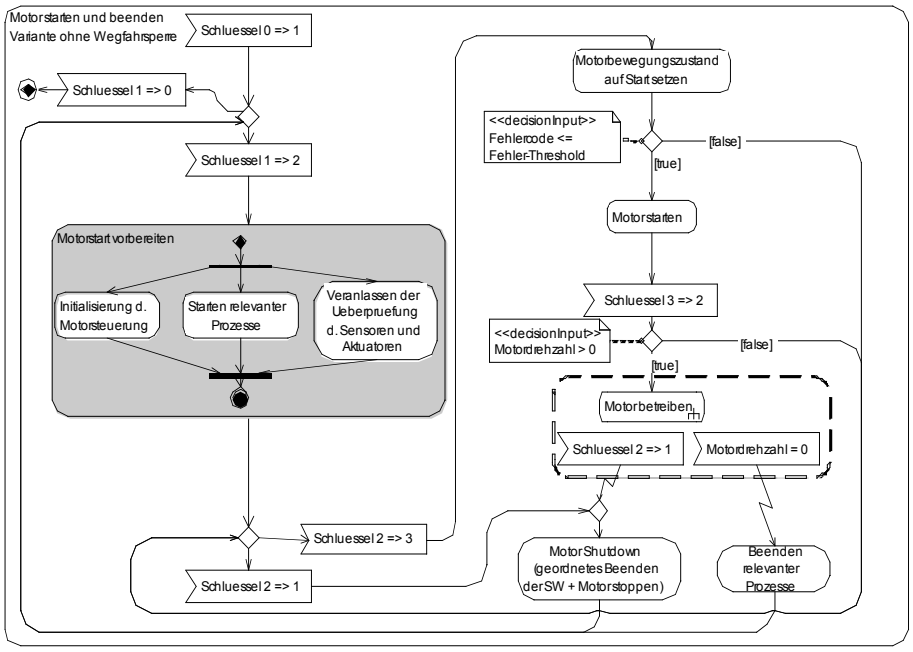
Referenzen

1. Forschungsprojekt PESOA – Phase 1, Anwendungsszenario Benzinmotorsteuerung, Erschienen: n/a
2. UML 2.0 Superstructure Specification, OMG Adopted Specification, Version 2.0, August 2003
3. Diplomarbeit, Untersuchung der Modellierung von Variabilität in UML, Mathias Clauß, Technische Universität Dresden, Fakultät Informatik, 24. Juli 2001
4. Journal of Object Technology, UML 2 Activity and Action Models, Conrad Bock, US National Institute of Standards and Technology, Vol. 2, No. 4, July-August 2003
5. PESOA, Process Modeling Techniques, Arnd Schnieders, Frank Puhlmann, Mathias Weske, Hasso-Plattner-Institut für Softwaresystemtechnik GmbH, PESOA-Report No. 01/2004, February 6, 2004
6. UML Activity Diagrams as a Workflow Specification Language, Marlon Dumas and Arthur H.M. ter Hofstede, Cooperative Information Systems Research Centre, Queensland University of Technology, In Proceedings of the UML'2001 Conference
7. Telelogic Tau, http://www.telelogic.com/, Informationen u.A. zu diesem Tool für UML
8. Rational Rose, http://www.rational.com/, Informationen u.A. zu diesem Tool für UML
9. Microsoft Visio, http://www.microsoft.com/office/visio/, Informationen u.A. zu diesem Tool für UML
10. What's New in UML. 2.0? Activity, class, communication, and use case diagrams: Part one in a three-part series, A Borland White Paper, Granville Miller, Senior Manager, Developer Relations, Dezember 2003
11. Using UML 2.0 in Real-Time Development, A Critical Review, Kirsten Berkenkötter, University of Bremen, P.O.B. 330 440, D-28334 Bremen, kirsten@tzi.de, Erschienen: n/a

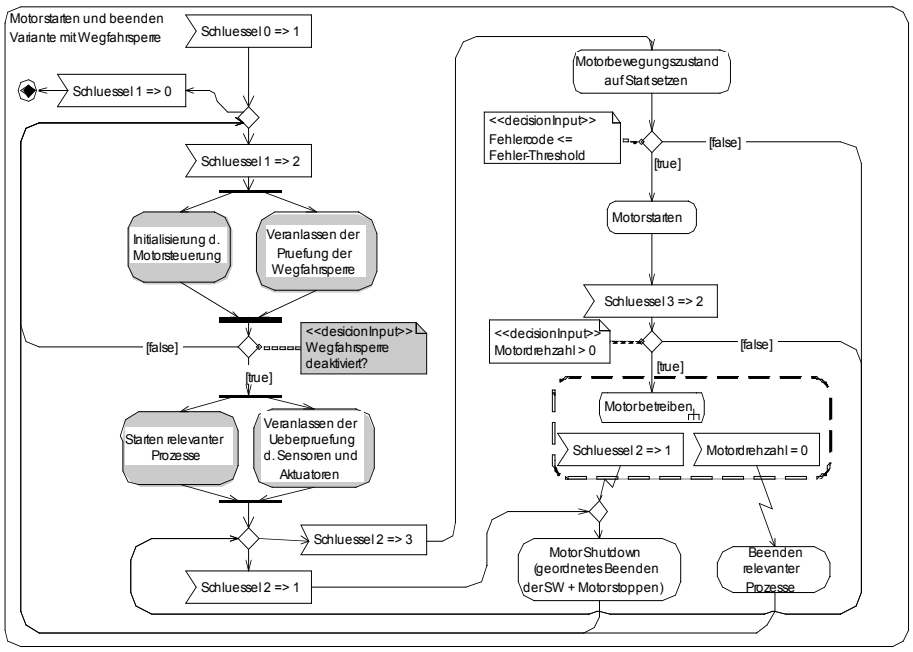
Anhang A: Zustandsdiagramm, Motor starten und beenden



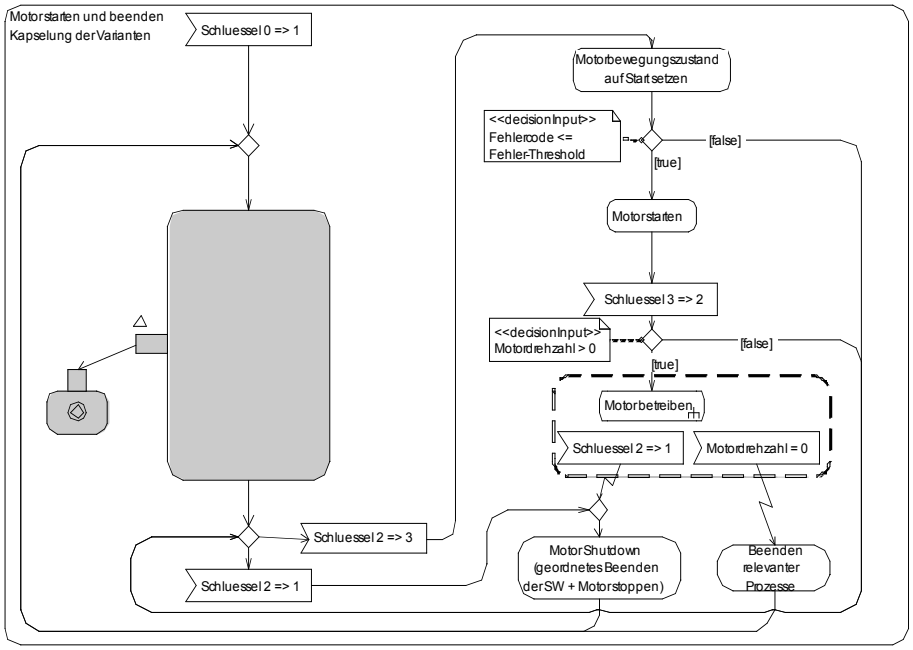
Anhang B: Aktivitätsdiagramm, Motor starten und beenden (Variante ohne Wegfahrsperre)



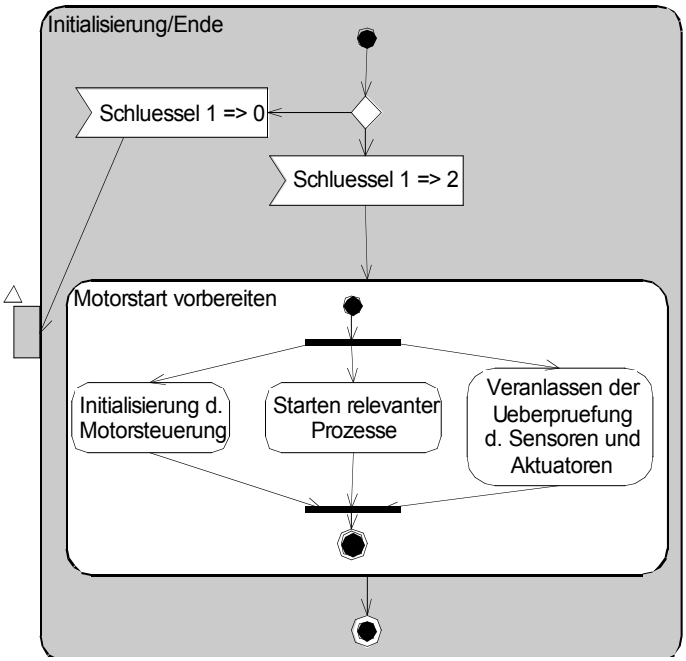
Anhang C: Aktivitätsdiagramm, Motor starten und beenden (Variante mit Wegfahrsperre)



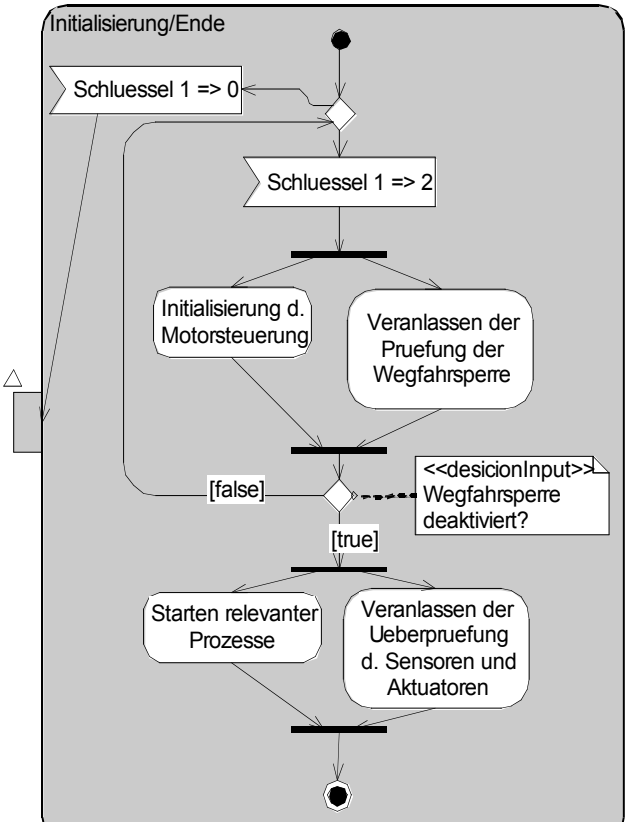
Anhang D: Aktivitätsdiagramm, Motor starten und beenden (Kapselung der Varianten)



(a) Gesamtprozess mit Kapselungsbereich

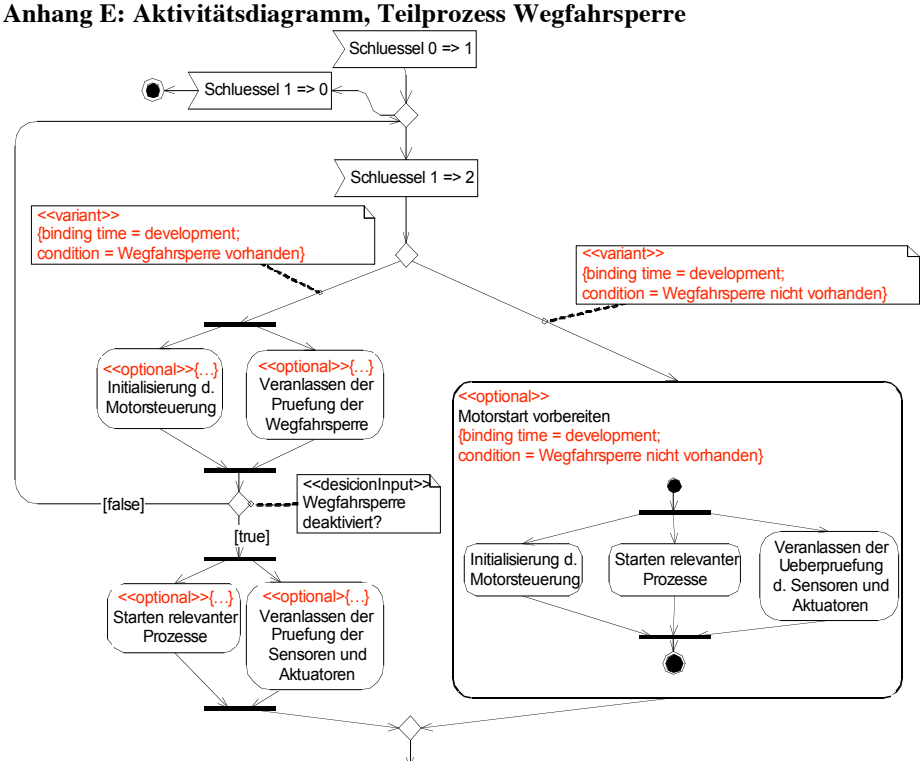


(b) Teilprozess ohne Wegfahrsperre

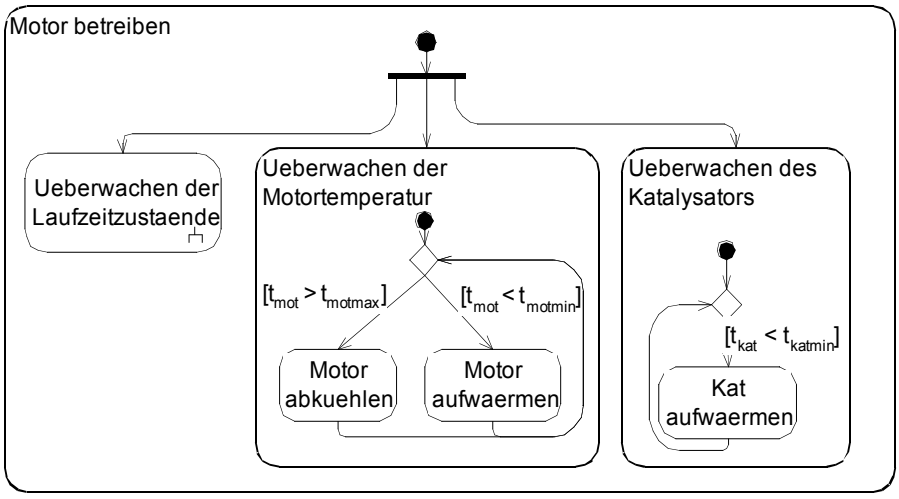


(c) Teilprozess mit Wegfahrsperre

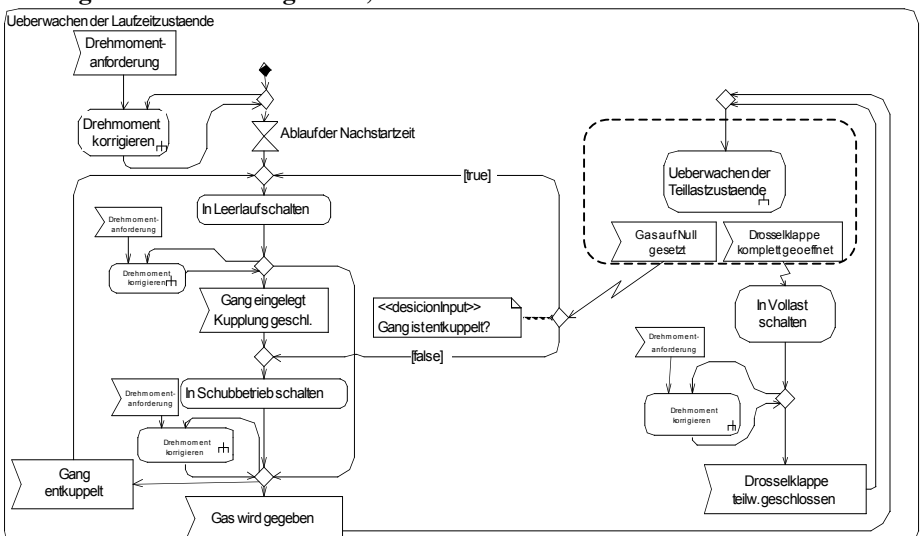
Anhang E: Aktivitätsdiagramm, Teilprozess Wegfahrsperre



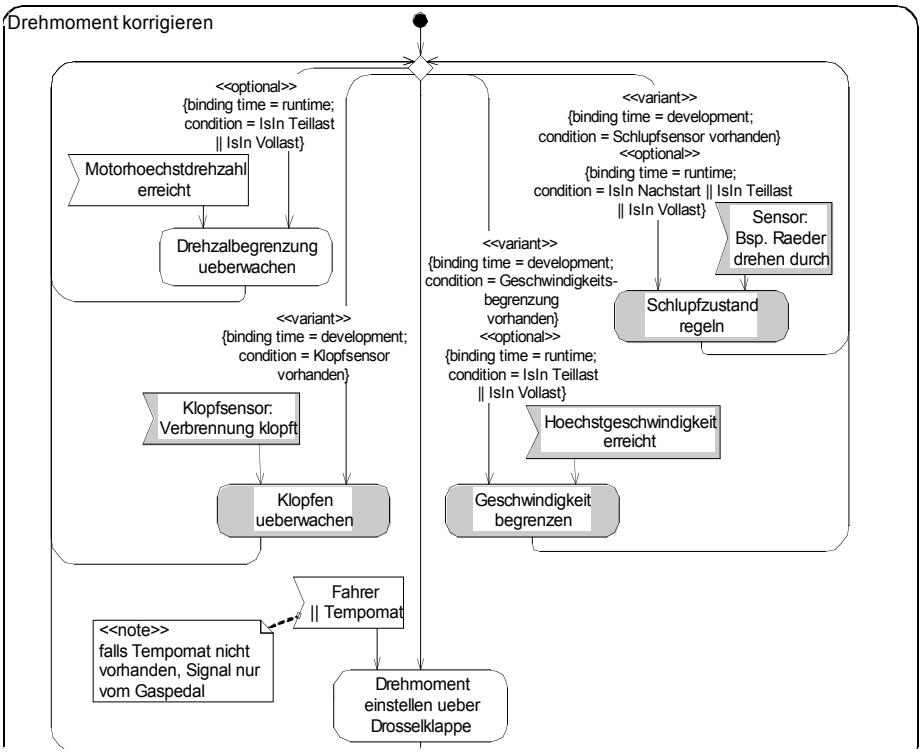
Anhang F: Aktivitätsdiagramm, Motor betreiben



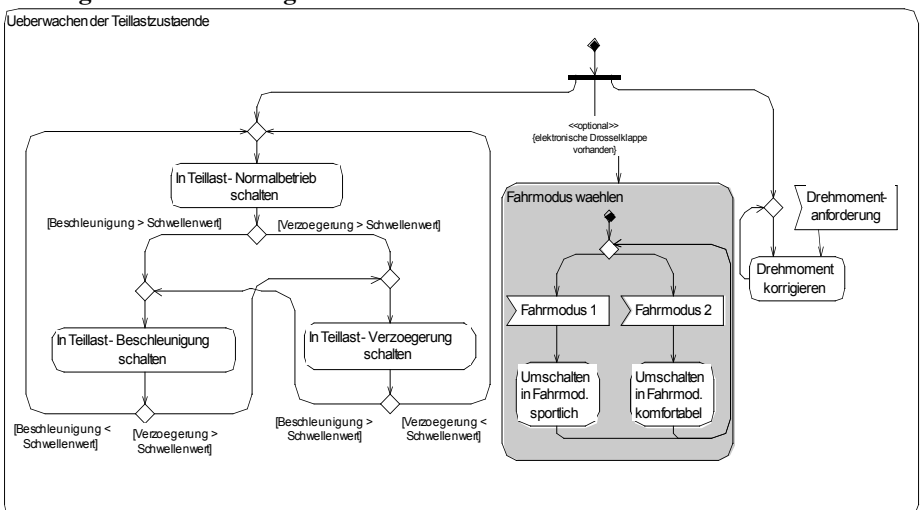
Anhang G: Aktivitätsdiagramm, Überwachen der Laufzeitzustände



Anhang H: Aktivitätsdiagramm, Drehmoment korrigieren



Anhang I – Aktivitätsdiagramm: Überwachen der Teillastzustände



Prozesse im Automobilbau

Alexander Saar

Hasso Plattner Institute für Softwaresystemtechnik
Lehrstuhl für Business Process Technology
Postfach 900460, D-14440 Potsdam

alexander.saar@hpi.uni-potsdam.de

Die vorliegende Arbeit befasst sich mit der Modellierung von Automatisierungsprozessen aus dem Bereich des Automobilbaus. Die Prozesse werden im Verlauf der Arbeit mit der Workflow-Modellierungssprache *Yet Another Workflow Language* (YAWL) modelliert, wobei deren Eignung für die Modellierung solcher Prozesse untersucht wird.
Da auf Software-Entwicklung im Bereich des Automobilbaus ein erheblicher und ständig wachsender Anteil der Gesamtentwicklungskosten entfällt, werden anschließend an die Modellierung die Prozesse, bzw. deren verschiedene Varianten, hinsichtlich wiederverwendbarer Teilprozesse untersucht.
Abschließend werden die Ergebnisse der Modellierung mit anderen Modellierungstechniken wie Zustandsdiagrammen oder Aktivitätsdiagrammen verglichen.

1. Einführung

Aufgrund der sehr hohen und ständig weiter zunehmenden Komplexität der Software und dem damit verbundenen hohen Anteil an den Gesamtentwicklungskosten wird in der Automobilbranche sowie in den meisten anderen Bereichen der Software-Entwicklung verstärkt auf die Modellierung von Prozessen gesetzt.

Diese Entwicklung soll vor allem zu einer Verbesserung der Qualität der Prozesse führen und eine kontrollierte Ausführung der Prozesse ermöglichen. Prozesse lassen sich dabei in fast allen Bereichen eines Unternehmens finden, z.B. Entwicklungs-, Software- oder Geschäftsprozesse.

Der Focus dieses Dokuments liegt auf der Modellierung von Software-gesteuerten Automatisierungsprozessen. Es sollen Prozesse modelliert werden, die im Rahmen der Automatisierung im Automobilbereich entstanden und die von Software-Komponenten ausgeführt werden. Die Modellierung dieser Prozesse soll mit Hilfe einer Modellierungssprache für Geschäftsprozesse wie z.B. Workflow-Netzen oder YAWL erfolgen. Dabei soll untersucht werden, ob ein Workflow-basierter Modellierungsansatz für die Modellierung von Software-Prozessen geeignet ist.

Ein Beispiel für einen solchen Automatisierungsprozess ist das automatisierte Abbremsen eines Fahrzeuges, welches von einem Sensor, der ein sich näherndes Hindernis entdeckt, angestoßen wird. Der Prozess des Bremsens besteht dabei aus dem Er-

kennen eines Hindernisses sowie dem Einleiten des Bremsvorgangs. Früher konnte der Bremsprozess nur vom Fahrer des Fahrzeugs ausgeführt werden. Heutzutage kann er jedoch in bestimmten Situationen durch eine Software-gesteuerte Komponente ausgeführt werden, welche Signale von einem Sensor erhält und nach entsprechender Interpretation dieser Signale den Bremsvorgang einleitet.

Dieses Beispiel zeigt die Ähnlichkeit der in dieser Arbeit betrachteten Prozesse mit den von Menschen ausgeführten Prozessen. Doch auch wenn Automatisierungsprozesse den von Menschen durchgeführten Prozessen ähnlich sind – ursprünglich wurden sie ja von diesen ausgeführt – sind bei ihrer Modellierung einige Besonderheiten zu beachten.

Diese Besonderheiten beziehen sich im Automobilbereich vor allem auf die Aspekte *Echtzeitfähigkeit (Real Time)* und *Sicherheit (Safety)* [10], denn schließlich steht bei einem Fahrzeug die Sicherheit der Fahrzeuginsassen immer im Vordergrund[1]. Im angeführten Beispiel des automatisierten Bremsprozesses ist es z.B. notwendig dafür zu sorgen, dass der Bremsvorgang rechtzeitig eingeleitet wird um das Fahrzeug noch vor dem Hindernis zum Stehen zu bringen.

Der Aspekt *Sicherheit (Safety)* bezieht sich hierbei auf die Sicherheit der Fahrzeuginsassen und darf nicht mit dem englischen Begriff *Security*, also der Sicherheit von Daten und Kommunikationsmedien, verwechselt werden. Der Aspekt *Echtzeitfähigkeit* bezieht sich auf das rechtzeitige Reagieren der Bremskomponente bzw. das rechtzeitige Reagieren der zuständigen Steuerungskomponenten.

Weiterhin ist zu beachten, dass sich beim Ausfallen der Bremskomponente diese entsprechend abschaltet und dies dem Fahrer angezeigt wird. Beim Ausfall von Komponenten ohne die ein sicheres Führen des Fahrzeugs nicht mehr möglich ist, muss das gesamte System (in diesem Fall das Fahrzeug) abgeschaltet oder in einen sicheren Zustand gebracht werden. Dieses Verhalten wird im Allgemeinen als *Fail-Safe* bezeichnet.

Komponenten wie z.B. die Bremskomponente, die auf einen Sensor reagiert und wenn notwendig den Bremsvorgang einleitet, werden als *Electronic Control Units* (ECU) bezeichnet. Eine solche ECU für die Steuerung eines Benzinmotors soll in dieser Arbeit als Beispiel für die Modellierung dienen. Abbildung 1 zeigt beispielhaft den Aufbau einer ECU zur Steuerung eines Benzinmotors. Die ECU verfügt über Sensoren, mit denen sie auf Signale ihrer Umwelt reagieren kann, sowie über Aktoren, mittels derer sie auf ihre Umwelt einwirken und Aktionen auslösen kann.
Die als Vorlage dienenden Prozesspezifikationen einer ECU zur Steuerung eines Benzinmotors wurden im Rahmen des PESOA Projektes [9] von der *Daimler/Chrysler AG* zur Verfügung gestellt.

[1] Da keine Angaben über Sicherheit, Echtzeitfähigkeit oder Fail-Safe in den PESOA-Prozessspezifikationen vorhanden sind, treten derartige Betrachtungen nicht in den Modellen auf. Jedoch gehören diese Aspekte i. A. dazu, wenn Komponenten aus dem Automobilbau betrachtet werden.

Abbildung 1: ECU zur Steuerung eines Benzinmotors

Um ein leichteres Verständnis und eine Fokussierung der Leser auf die Aspekte der Modellierung der Automatisierungsprozesse zu ermöglichen, wird im weiteren Verlauf des Dokuments zuerst die Modellierung mit Workflow-Netzen [5] und YAWL ([7], [1]) eingeführt. Danach werden in Kapitel 3 einige ausgewählte Prozesse beschrieben und modelliert.

Kapitel 4 befasst sich mit der Analyse verschiedener Varianten der im 3. Kapitel entwickelten Prozess-Modelle hinsichtlich der Wiederverwendung von Teilprozessen. Abschließend werden im 5. Kapitel die Ergebnisse der Modellierung aus Kapitel 3 mit anderen Modellierungstechniken wie den Automaten-, Sequenz- und Aktivitätsdiagrammen der *Unified Modeling Language* (UML 2.0, [4]) verglichen.

2. Workflow-Modellierung

Im Folgenden werden die in dieser Arbeit verwendeten Modellierungstechniken eingeführt und beschrieben. Dazu wird zum besseren Verständnis der für die Modellierung gewählten Sprache YAWL zuerst deren zugrunde liegende Konzepte aus den Workflow-Netzen und Workflow-Pattern erläutert. Im Anschluss werden die einzelnen Modellierungselemente vorgestellt und anhand eines Beispiels verdeutlicht.

2.1 Modellierung mit Workflow-Netzen

Ein bekanntes Mittel zur Modellierung von Geschäfts- und Entwicklungsprozessen sind *Workflow-Netze* [5]. Workflow-Netze sind sogenannte High-Level Petri-Netze, also Petri-Netze welche um zusätzliche Elemente erweitert wurden.

Eine Kenntnis der Petri-Netze wird beim Leser vorausgesetzt und kann in [6] nachgelesen werden. Die Erweiterungen der Workflow-Netze umfassen nachfolgend aufgelistete Aspekte:

Funktionale Dekomposition:
Durch das Einführen von hierarchischen Transitionen wurde ein Aufteilen der modellierten Prozesse in mehrere Teilprozesse möglich. Dies hilft vor allem beim Verständnis und der Strukturierung eines Prozesses sowie bei der Identifikation und Wiederverwendung von Teilprozessen.

Trigger:
Trigger ermöglichen die Modellierung externer Ereignisse und Nachrichten. Somit kann die Reaktion eines Prozesses auf die Umgebung modelliert werden.

Zeitliche Erweiterung:
Die Modellierung von Warte- und Bearbeitungszeiten durch die Erweiterung der Token und Kanten um Zeitattribute ermöglicht unter anderem Performanzanalysen oder die Definition und Einhaltung von zeitlichen Standards.

Farbliche Erweiterung:
Mittels Erweiterung der Petri-Netze um farbliche Kennzeichnung von Token kann diesen eine eindeutige Identifikationen bzw. entsprechende Daten zugewiesen werden. Mit Hilfe dieser können so verschiedene Instanzen eines Prozesses sowie die Transformation von Daten bei der Verarbeitung in Aktivitäten modelliert werden.

Grafische Erweiterungen:
Grafische Erweiterungen für Routing-Konstrukte wie z.B. Splits oder Joins ermöglichen eine komprimierte Darstellung und somit eine bessere Übersichtlichkeit und Verständlichkeit der Modelle.

2.2 Workflow-Pattern

Basierend auf den Erfahrungen aus der Arbeit mit Workflow-Netzen und anderen Modellierungssprachen für Unternehmensprozesse, entwickelte *W.M.P. van der Aalst* eine Reihe von häufig auftretenden *Workflow-Pattern*[2].

Mit Hilfe dieser Workflow-Pattern kann die Ausdrucksfähigkeit verschiedener Workflow-Sprachen verglichen werden. Dazu werden die einzelnen Pattern mit den entsprechenden Sprachen modelliert und die Tauglichkeit der Sprache beziehungs-

[2] Der Begriff *Pattern* wurde erstmals von Erich Gamma [3] verwendet, um häufig wiederkehrende Muster in der objekt-orientierten Programmierung zu beschreiben.

weise Probleme bei der Modellierung bewertet. Eine genaue Beschreibung der einzelnen Pattern kann in [2] nachgelesen werden.

Mittels der so gewonnenen Erkenntnisse untersuchte *van der Aalst* fünfzehn verschiedene Workflow-Systeme verschiedener Anbieter [1]. Die meisten dieser Systeme implementierten eine eigene Lösung einer Workflow-Sprache oder erweiterten zumindest eine standardisierte Sprache entsprechend ihren Bedürfnissen.

Dabei machte *van der Aalst* die Erfahrung, dass die meisten Systeme nur für weniger als die Hälfte der Pattern direkte Unterstützung zur Implementierung anbieten. Oft konnten die restlichen Pattern zwar modelliert werden, jedoch nur auf sehr umständliche Weise und mit großem Aufwand.

Ausgehend von dieser Erkenntnis wurde von *van der Aalst* eine eigene Workflow-Sprache vorgeschlagen, welche volle Pattern-Unterstützung ermöglicht. Diese wird im nächsten Kapitel vorgestellt.

2.3 Modellierung mit YAWL

Die Workflow-Sprache *Yet Another Workflow Language* (YAWL) wurde im Jahr 2000 von *van der Aalst* als Modellierungssprache für Workflows vorgeschlagen ([7], [1]). Diese soll in dieser Arbeit als Modellierungssprache für die Prozesse dienen.

YAWL basiert auf den als Workflow-Netzen bekannten High-Level Petri-Netzen, die in Kapitel 2.1 schon kurz beschrieben wurden. Da einige der Workflow-Pattern mit Workflow-Netzen nicht ohne großen Aufwand zu modellieren sind, wurden diese in YAWL um Elemente zur Modellierung der Workflow-Pattern erweitert.

Abbildung 2 zeigt eine Übersicht über die in YAWL verwendeten Notationssymbole. Diese werden im Folgenden kurz beschrieben.

Condition:
Repräsentiert einen Zustand in einem Prozess.

Input Condition:
Eingangsbedingung bei welcher der Prozess startet.

Output Condition:
Ausgangsbedingung bei welcher der Prozess beendet wird.

XOR-Join:
Wird jedes Mal aktiviert, wenn eine eingehende Kante aktiviert wird.

AND-Join:
Wird aktiviert, wenn alle eingehenden Kanten aktiviert sind.

OR-Join:
Wird aktiviert, wenn eine oder mehrere eingehende Kanten aktiviert werden.

XOR-Split:
Nach der Beendigung wird eine ausgehende Kante aktiviert.

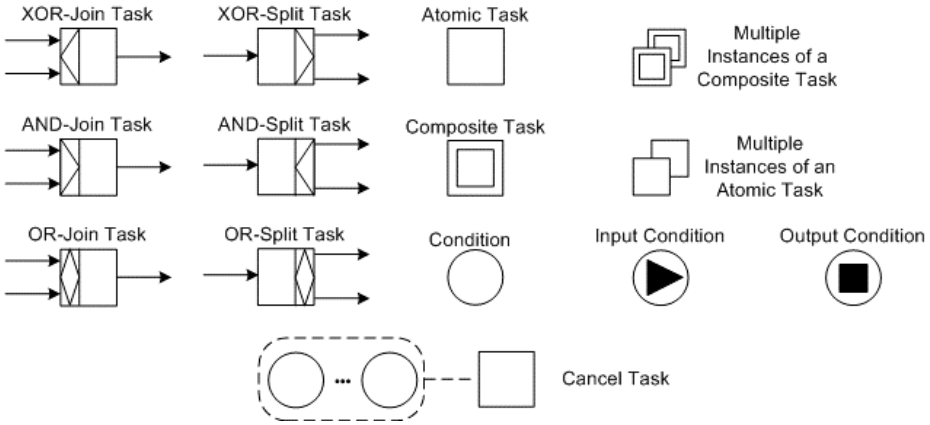


Abbildung 2: YAWL Notationsübersicht

AND-Split:
Nach der Beendigung werden alle ausgehenden Kanten aktiviert.

OR-Split:
Nach der Beendigung werden eine oder mehrere ausgehende Kanten aktiviert.

Atomic Task:
Atomare Aktivität die von einem Menschen oder einer externen Anwendung ausgeführt wird.

Composite Task:
Repräsentiert einen Container für einen weiteren YAWL-Prozess und ermöglicht somit funktionale Dekomposition.

Multiple Instances of an Atomic Task:
Ermöglicht mehrere Instanzen eines Atomic Tasks, welche nebenläufig arbeiten. Es ist dabei möglich die minimale und maximale Anzahl der Tasks festzulegen sowie einem Schwellwert. Haben entsprechend diesem Schwellwert genügend Tasks ihre Arbeit beendet, wird der gesamte Task mit allen noch laufenden Instanzen beendet. Weiterhin können weitere Instanzen zur Laufzeit erzeugt werden.

Multiple Instances of a Composite Task:
Selbe Funktion wie bei mehreren Instanzen des Atomic Task, jedoch für Composite Tasks.

Cancel Task:
Bei der Aktivierung dieses Tasks werden alle anderen Tasks oder Conditions, die sich in der markierten Region befinden, deaktiviert bzw. alle Token werden gelöscht.

Mittels dieser Elemente lässt sich jedoch nur der Kontrollfluss eines Prozesses modellieren. Für die Anbindung von externen Anwendungen zur Ausführung von atomaren Aktivitäten werden sogenannte YAWL-Services verwendet, welche auf der *Web-Service Description Language* (WSDL) basieren. Web-Services arbeiten mittels XML-Darstellung von Objekten und Schnittstellen, so dass Datenobjekte in XML-Format übertragen werden können. Sowohl die operationellen als auch die Datenaspekte werden in YAWL mittels XML und Web-Services dargestellt und verarbeitet.

Abbildung 3 zeigt ein Beispiel eines YAWL-Prozessmodells. In dem Beispiel wird nach der initialen Ausführung von Task A der Timeout-Task B aktiviert und der Zustand C_1 erreicht.

Danach wird Task C aktiviert und nach dessen Beendigung wird der Zustand C_2 erreicht. Anschließend werden mehrere Instanzen von Task D gestartet, welche aus einer weiteren YAWL-Prozesspezifikation mit zwei Tasks I und J bestehen.

Läuft der Timer in Task B ab bevor D gestartet wird, so löscht er die entsprechenden Token in C_1, C_2 oder C und beendet somit die Ausführung. Der Composite Task D wird nach der Ausführung von Task I entsprechend dessen Ergebnis entweder sofort beendet oder es wird vorher noch Task J ausgeführt.

Bei der Modellierung mit YAWL ist gegenüber der Modellierung mit Workflow-Netzen zu beachten, dass YAWL-Prozessdiagramme nicht bipartit sein müssen. Unnötige Conditions müssen nicht modelliert werden, vielmehr werden nur explizit definierte Zustände des Prozesses modelliert. Dies entspricht aber nur dem Zusammenfassen von Sequenzen und hat keine Auswirkungen auf die Modellierung oder die formale Spezifikation.

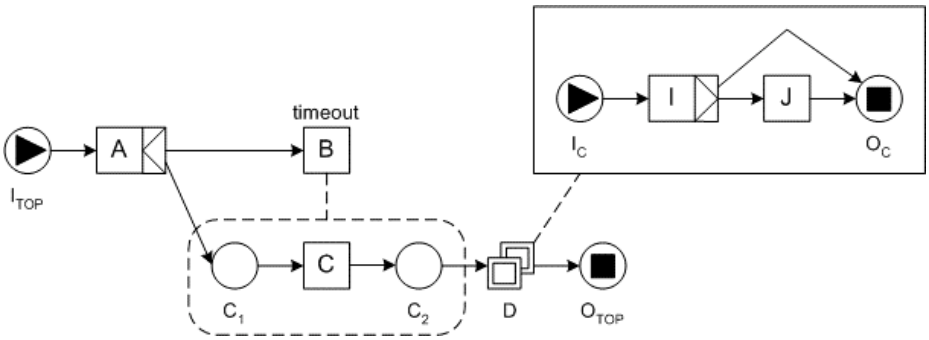


Abbildung 3: Beispiel eines YAWL-Prozessmodells

Ein Vorteil von YAWL gegenüber anderen Prozessmodellierungstechniken wie z.B. der *Buissness Process Modeling Language* (BPMN) oder den UML 2.0 Aktivitätsdiagrammen ist die vollständige formale Spezifikation, welche durch das Ableiten YAWLs von den Petri-Netzen ermöglicht wurde.

Im Rahmen dieser Arbeit wird die formale Spezifikation jedoch keine Anwendung finden, da der Focus auf der Modellierung und nicht auf der Analyse von Prozessen liegen soll. Genauere Informationen bzgl. der formalen Beschreibung sowie der beweisbaren Eigenschaften von YAWL können in [1] nachgelesen werden.

3. Modellierung der Prozesse

In diesem Kapitel werden die Vorgehensweise bei der Modellierung der Prozesse und die Abbildung der Komponenten der betrachteten ECU auf YAWL-Modellierungselemente beschrieben. Weiterhin werden die aus den PESOA-Prozessspezifikationen entwickelten Prozessmodelle vorgestellt und erklärt.

3.1 Abbildung der ECU-Komponenten auf YAWL

Bei der Modellierung der gegebenen Prozessspezifikationen war es zuerst notwendig, die Sensoren und Aktoren der betrachteten ECU sowie die Eigenschaften der PESOA Prozesse auf YAWL abzubilden.

Die in den Prozessbeschreibungen enthaltenen Zustände und Aktivitäten werden auf Conditions und Tasks abgebildet. Die Abbildung der Sensoren erfolgte mittels der im Kapitel über Workflow-Netze beschriebenen Trigger. Trigger sind zwar nicht Bestandteil von YAWL, sie lassen sich jedoch ohne Probleme verwenden, da YAWL auf Workflow-Netzen basiert und ein Trigger nur jeweils eine zusätzliche Condition repräsentieren auf die von außen, also von einem anderen Prozess, ein Token gelegt wird.

Abbildung 4 zeigt die beispielhafte YAWL-Modellierung eines Sensorprozesses, welcher, entweder bei Bedarf oder kontinuierlich, Token erzeugt, die dann von einem anderen Prozess konsumiert werden können. Durch das Empfangen eines externen Signals (Nachricht wird mit einem kleinen Briefumschlag gekennzeichnet) kann der Sensor abgeschaltet werden. Sensoren treten in den modellierten Prozessen als Trigger auf.

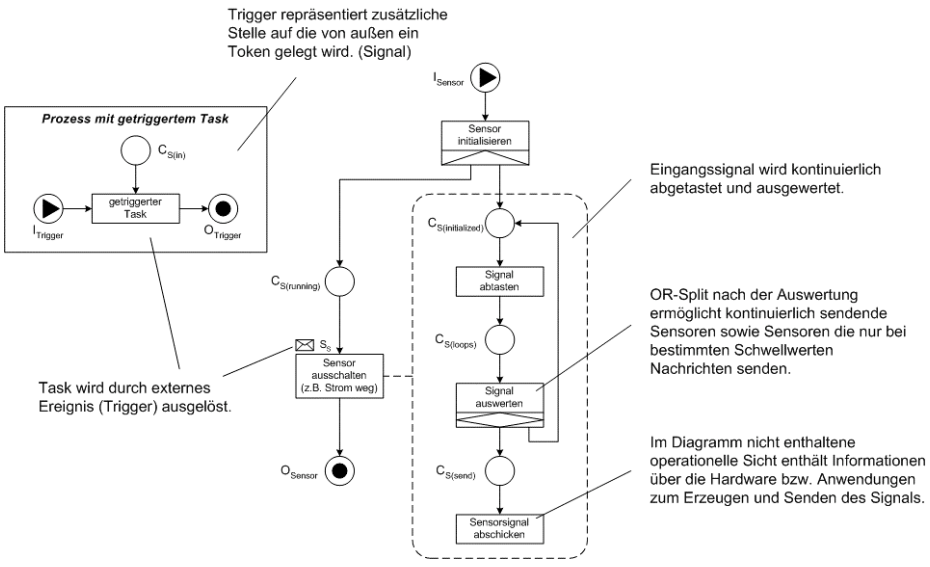


Abbildung 4: Abbildung von Sensoren auf Trigger

Das Beispiel zeigt weiterhin ein häufig zu findendes Muster der zu modellierenden Prozesse. So laufen viele Prozesse in Endlosschleifen und können sich nicht selbst beenden. Ein Beenden des Prozesses kann nur von außen durch einen dritten Akteur oder einen Ausfall anderer Systeme (z.B. Stromversorgung) geschehen. Die Modellierung dieses Musters stellt jedoch mit Hilfe des Cancel Tasks kein Problem dar.

Die Abbildung der Aktoren der ECU ist in den modellierten Diagrammen nicht enthalten, da es sich hierbei um Elemente aus dem operationellen Bereich der Prozesse handelt. Sie müssen also zusätzlich in XML beschrieben werden. Jedoch ist die Verwendung von Web-Services im Bereich der eingebetteten Automatisierungsprozesse nicht zu empfehlen, da der Verarbeitungsaufwand von XML sehr hoch ist und somit auch die Bearbeitungszeiten der durchzuführenden Aktionen steigen.

Z.B. müssen bei einer Drehzahl von 3000 U/min die Einspritzmenge 3000-mal pro Minute berechnet und entsprechende Korrekturen an der Einspritzung vorgenommen werden. Somit ist eine effiziente, schnelle und echtzeitfähige Kommunikation notwendig.

Der Einsatz binär arbeitender Bus-Systeme wie z.B. *Controller-Area-Network* (CAN) ist in der Automobiltechnik weit verbreitet. Dies ermöglicht eine schnellere Kommunikation zwischen den beteiligten Akteuren, da solche Bus-Systeme auf der 2. Schicht des OSI-Referenzmodells arbeiten und nicht wie die XML-basierten Web-Services fünf Schichten des Modells implementieren (siehe Abbildung 5).

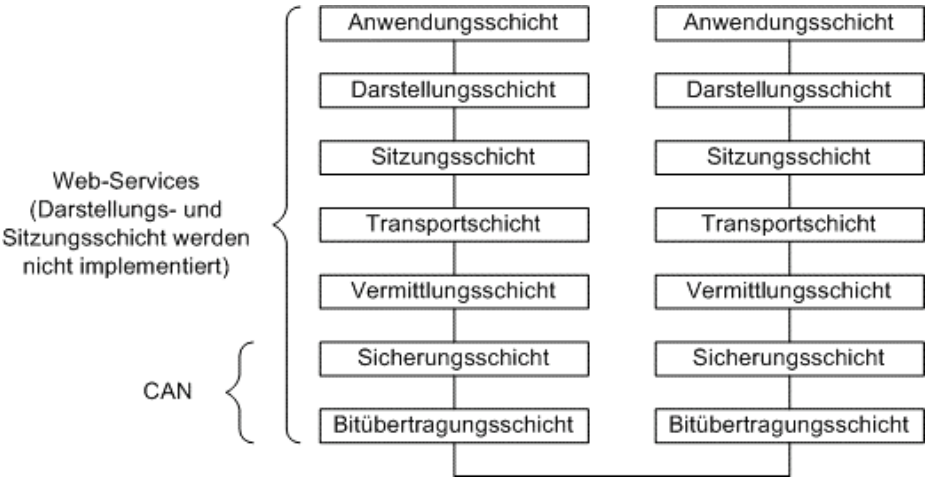


Abbildung 5: OSI-Referenzmodell

Im Bereich der eingebetteten Automatisierungsprozesse ist im Gegensatz zu Unternehmens- oder Entwicklungsprozessen eine On-Demand Erzeugung von Prozessen im Allgemeinen jedoch nicht notwendig. Somit könnten eine Modellierung der Prozesse mittels YAWL und eine automatische Umsetzung der XML-Spezifikationen in Bus-basierte Kommunikationsprotokolle denkbar sein.

3.2 Top-Level Prozess

Der Top-Level Prozess beschreibt die übergeordnete Steuerung des Benzinmotors, also das Starten und Beenden des Motors und der für die Steuerung notwendigen Prozesse. Er kann vom Fahrer des Fahrzeugs direkt über das Zündschloss beeinflusst und gesteuert werden. Abbildung 6 zeigt das YAWL-Modell des Top-Level Prozesses.

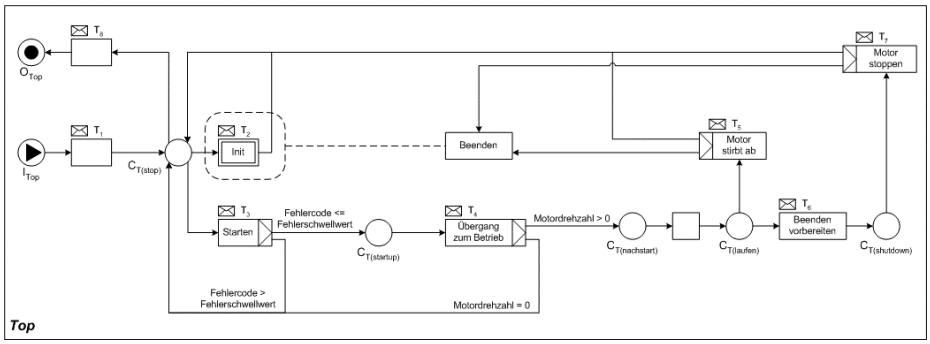


Abbildung 6: Top-Level Prozess

Die Trigger der einzelnen Tasks beschreiben im Top-Level Prozess die Signale vom Zündschloss, über welches der Prozess gesteuert werden kann. Dreht z.B. der Fahrer den Zündschlüssel von Stellung 0 auf 1, so wird aus dem Initialzustand in den Zustand $C_{T(stop)}$ gewechselt.

Eine Änderung der Zündschlüsselstellung von 1 auf 2 aktiviert den in Abbildung 7 dargestellten Teilprozess *Init*, welcher die relevanten Teilprozesse der ECU startet und eine Überprüfung der Sensoren und Aktoren durchführt. Danach befindet sich der Prozess weiterhin im Zustand $C_{T(stop)}$.

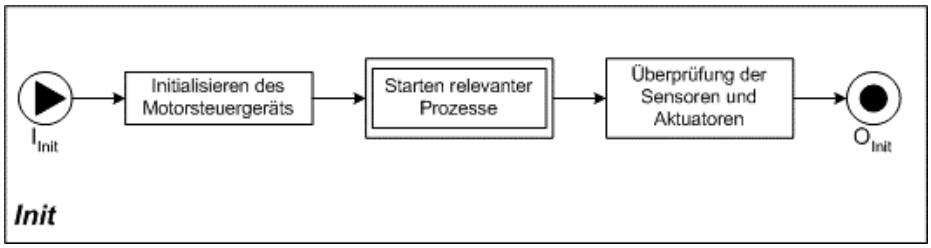


Abbildung 7: Teilprozess Init

Das in Abbildung 7 als Composite Task dargestellte Starten der relevanten Prozesse wird in Abbildung 8 gezeigt. Bei den zu startenden Prozessen handelt es sich um Prozesse zur Überwachung und Abbildung von Motorzuständen sowie um Prozesse zur direkten Steuerung des Motors bzw. des Drehmoments.

Dadurch ist eine Unterteilung der Prozesse in die Kategorien *Überwachungszustände* und *Steuerungszustände* möglich. Im Folgenden werden einige ausgewählte Prozesse aus den beiden Kategorien veranschaulicht und beschrieben.

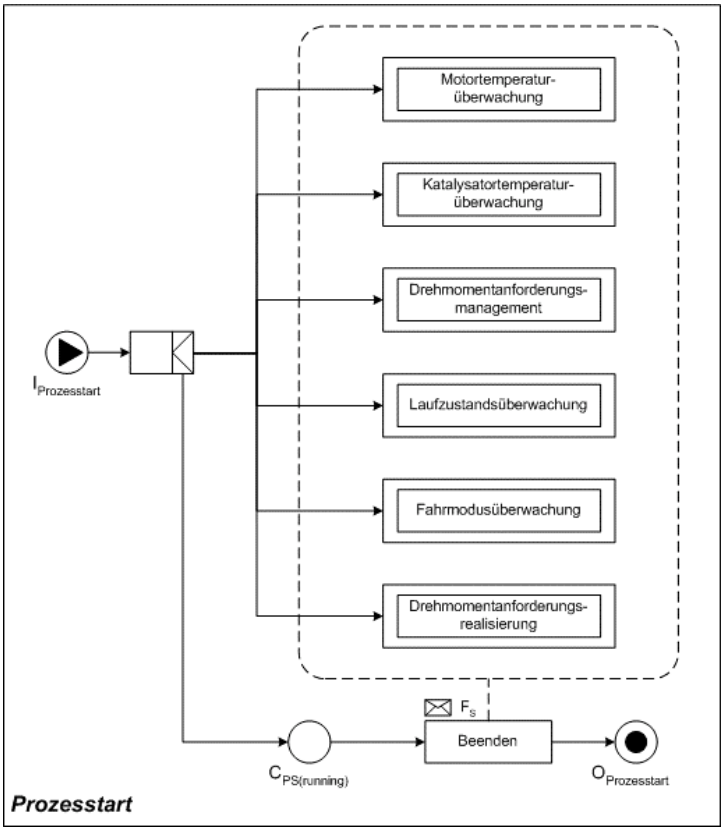


Abbildung 8: Starten relevanter Prozesse

3.3 Überwachungsprozesse

Überwachungsprozesse empfangen hauptsächlich Signale von den Sensoren und bilden aktuelle Motorgegebenheiten oder -einstellungen intern auf Motorzustände[3] ab.

Sie sind im Allgemeinen nicht von den Zuständen anderer Prozesse abhängig und besitzen meist keine steuernden Aufgaben. Folgende Überwachungsprozesse wurden in den PESOA Prozesspezifikationen identifiziert (siehe auch Abbildung 15-18 im Anhang).

Motortemperaturüberwachung:
Bildet die kontinuierlichen Temperaturzustände des Motors auf diskrete Motortemperaturzustände ab, z.B. 2000°C auf den Zustand $C_{M(\text{überhitzung})}$. (Abbildung 9)

[3] Eine genaue Beschreibung der Zustände der Überwachungsprozesse sowie der zugehörigen Zustandsübergänge und Sensoren kann in [9] nachgelesen werden.

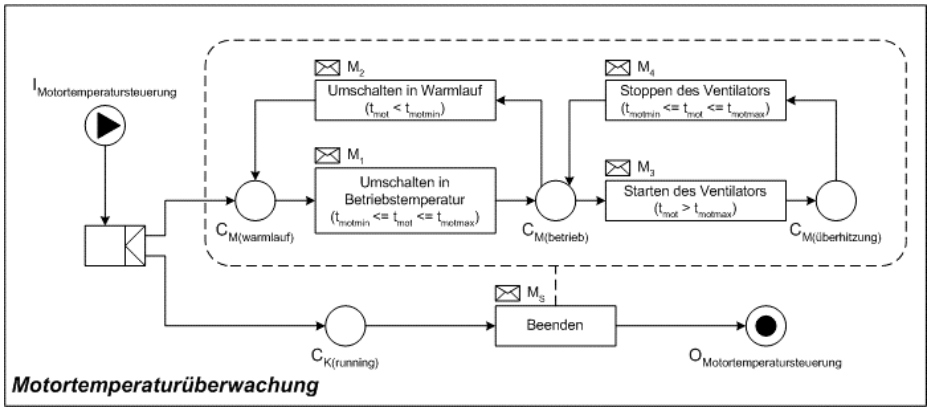


Abbildung 9: Motortemperaturüberwachung

Katalysatortemperaturüberwachung:
Bildet die kontinuierlichen Temperaturzustände des Katalysators, z.B. 2000°C, auf diskrete Katalysatortemperaturzustände, z.B. $C_{K(betrieb)}$, ab. (siehe Abbildung 15 im Anhang)

Fahrmodusüberwachung:
Bildet die vom Fahrer gewählte Fahrmoduseinstellung über von einem Schalter empfangene Signale auf die entsprechenden Zustände ab. (siehe Abbildung 16 im Anhang)

Laufzustandsüberwachung und Teillaststeuerung:
Bilden anhand von Signalen des Getriebes und des Drehzahlmessers aktuelle Laufzustände des Motors und Getriebes ab. (siehe Abbildung 17 und 18 im Anhang)

Drehmomentanforderungsmanagement:
Berechnet in Abhängigkeit von Signalen von z.B. Klopfsensor oder Drehzahlmesser die aktuell einzustellende Drehmomentanforderung und bildet diese auf die verschiedenen Drehmomentanforderungszustände ab. (siehe Abbildung 19 im Anhang)

3.4 Steuerungsprozesse

Steuerungsprozesse nehmen direkt Einstellungen am Motor bzw. am Zündwinkel, der Drosselklappe, der Einspritzung oder dem Abgasrückführventil (AGR-Ventil) vor. Sie steuern somit das vom Motor zur Verfügung gestellte Drehmoment.

Abbildung 20 (siehe Anhang) zeigt die entsprechenden Teilprozesse (Composite Tasks) als Bestandteile des Teilprozesses zur Realisierung der Drehmomentanforderung.

Steuerungsprozesse sind im Gegensatz zu Überwachungsprozessen bei ihren Berechnungen nicht nur von eingehenden Sensorensignalen, sondern auch von den aktuellen Zuständen verschiedener Überwachungsprozesse abhängig[4].

Diese Abhängigkeit wird in den Modellen durch einen XOR-Join realisiert, welcher in Abhängigkeit von der Eingangsbelegung verschiedene Variablen setzt, welche den weiteren Ablauf des Prozesses beeinflussen. Anschließend werden die entsprechenden Token durch einen XOR-Split wieder an ihre Überwachungsprozesse zurückgegeben. Diese Modellierung wird am Beispiel des Prozesses zur Steuerung des Zündwinkels in Abbildung 10 illustriert.

Bei der Betrachtung der Modelle der Steuerungsprozesse ist darauf zu achten, dass nur die Conditions innerhalb des Rahmens direkt dem Prozess zugeordnet sind. Die externen Conditions gehören zu anderen Teilprozessen und wurden zur Verdeutlichung der Prozessabhängigkeiten hinzugefügt.

Das Problem bei dieser Modellierung von Prozessabhängigkeiten ist zum einen die kurzzeitige Blockierung der Überwachungsprozesse, da deren Token von den Steuerungsprozessen verwendet werden. Ein Modellierungskonstrukt zur direkten Modellierung der Abfrage von Prozesszuständen ist in YAWL nicht vorhanden. Weiterhin ist bei dieser Art der Modellierung jegliche Information über gültige Eingangsbelegungen für die Steuerungsprozesse nicht mehr im Diagramm zu erkennen.

Nachfolgend werden die einzelnen Steuerungsprozesse der Benzinmotorsteuerung kurz beschrieben. (siehe auch Abbildung 21-23 im Anhang)

Zündwinkel:
Nimmt in Abhängigkeit von Zuständen der Überwachungsprozesse und Signalen von Sensoren periodisch Korrekturen am Zündwinkel vor. (siehe Abbildung 10)

Drosselklappe:
Nimmt in Abhängigkeit von Zuständen der Überwachungsprozesse und Signalen von Sensoren periodisch Korrekturen an der Drosselklappeeinstellung vor. (siehe Abbildung 21 im Anhang)

Einspritzmenge:
Nimmt in Abhängigkeit von Zuständen der Überwachungsprozesse und Signalen von Sensoren periodisch Korrekturen an der Einspritzmenge vor. (siehe Abbildung 22 im Anhang)

AGR-Ventilöffnung:
Nimmt in Abhängigkeit von Zuständen der Überwachungsprozesse und Signalen von Sensoren periodisch Korrekturen an der Öffnung des AGR-Ventils vor. (siehe Abbildung 23 im Anhang)

[4] Die entsprechenden Abhängigkeiten der Steuerungsprozesse von den Überwachungsprozessen werden als Tabelle in [9] genauer beschrieben.

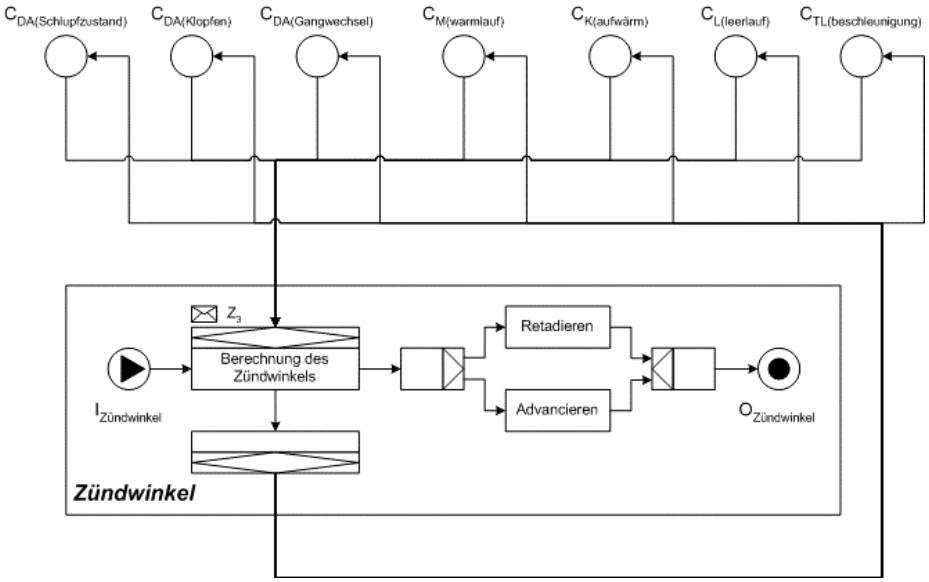


Abbildung 10: Zündwinkelsteuerung

4. Prozessvarianten und wiederverwendbare Teilprozesse

Aufgrund der zunehmenden Verwendung von Software-Komponenten verschiedener Zulieferer, müssen Entwickler solcher Software-Komponenten ihre Produkte möglichst flexibel gestalten. Dies ermöglicht eine Reduzierung von Entwicklungszeit und –kosten. In diesem Kapitel werden deshalb verschiedene Varianten der modellierten Prozesse hinsichtlich wiederverwendbarer Teilprozesse untersucht.

Die verschiedenen Prozessvarianten lassen sich dabei wieder in verschiedene Kategorien einteilen, da einige Varianten sich direkt auf die kontrollflussorientierten YAWL-Modelle auswirken und andere wiederum nicht.

So können z.B. sämtliche Sensoren und Aktoren Quellen der Variabilität darstellen, da sie eventuell von verschiedenen Drittanbietern bezogen werden. Ein Sensor kann z.B. mit einer Spannung von 0 bis 0,1 Volt arbeiten während ein anderer Sensor mit einer Spannung von 0 bis 0,5 Volt arbeitet. Solche Varianzen lassen sich durch den Einsatz verschiedener Treiber für Sensoren und Aktoren abfangen und treten dadurch nicht in den Modellen auf. Die Treiber müssen eine Umrechnung der entsprechenden Signale in physikalische Größen vornehmen, z.B. 0,1 Volt wird interpretiert als 1000° C.

Weiterhin kann die Motorsteuerung für verschiedene Motortypen verwendet werden. Dies wirkt sich jedoch nur auf diverse Funktionsparameter und nicht auf die kontrollflussorientierten Modelle aus.

Da der Focus der vorliegenden Arbeit sich auf die Modellierung richtet, werden im Folgenden einige Beispiele für Prozessvarianten vorgestellt, welche sich direkt auf die Modelle auswirken. Diese Prozessteile werden dazu als ausgelagerte Teilprozesse (Composite Task) modelliert und mittels sogenannter Varianzpunkte beschrieben. Das heißt, dass anhand der Beschreibung durch Varianzpunkte die entsprechenden Teilprozesse zu einem gültigen Prozessmodells zusammengesetzt werden können.

Varianzpunkte können sowohl in Tabellen als auch in XML dargestellt werden. Sie beschreiben die betroffenen Prozessteile, die Anzahl und Ausprägungen einer Variante sowie gültige Konfigurationen. Anhand der Konfigurationen können somit Prozessmodelle erzeugt werden, da sie varianzabhängige und systemweite Bedingungen für gültige Prozessmodelle beschreiben.

Weiterhin könnten durch die Darstellung mittels XML auch Abhängigkeiten zwischen Varianzpunkten sowie die Spezialisierung und Erweiterung von Varianzpunkten modelliert werden.

4.1 Varianzpunkt für Wegfahrsperre:

Tabelle 1 beschreibt einen Varianzpunkt für eine Wegfahrsperre, welche bei ihrer Aktivierung das Starten des Motors verhindert. Um dies zu realisieren, muss ein zusätzlicher Prozess für die *Wegfahrsperre* sowie eine Änderung des Prozesses *Init* zum Modell hinzugefügt werden. Diese wird als *Init_WFS* bezeichnet und überprüft vor dem Starten des Motors den aktuellen Zustand des Prozesses *Wegfahrsperre*.

Eine Beschreibung der gültigen Konfigurationen ist hier deshalb notwendig, weil der Einsatz von *Init_WFS* ohne die gleichzeitige Verwendung des Prozesses *Wegfahrsperre* keinen Sinn macht. Genauso wäre auch die Verwendung von *Init* und *Wegfahrsperre* innerhalb einer Konfiguration nicht sinnvoll.

Name		Wegfahrsperre
Anzahl		2
Ausprägungen (Varianten)		vorhanden
		nicht vorhanden
Auswirkungen (betroffene Prozesse)		Init
		Init_WFS
		Wegfahrsperre
Konfi-guratio-	vorhanden	Init_WFS
		Wegfahrsperre
	nicht vorhanden	Init

Tabelle 1: Varianzpunkt Wegfahrsperre

Folgendes Listing zeigt eine mögliche XML-Darstellung der Beschreibung des Varianzpunktes aus Tabelle 1.

```
<?xml version="1.0" encoding=" UTF-8"?>
<variant name=”Wegfahrsperre” number=”2”>
       <definition>
```

```
                <specifications>
                        <spec name=”vorhanden” />
                        <spec name=”nicht vorhanden” />
                </specifications>
                <effects>
                        <location name=“Init“ />
                        <location name=“Init_WFS“ />
                        <location name=“Wegfahrsperre“ />
                </effects>
        </ definition >
        <configurations>
                <config name=“vorhanden“>
                        <location name=“Init_WFS“ />
                        <location name=“Wegfahrsperre“ />
                </config>
                <config name=“nicht vorhanden“>
                        <location name=“Init“ />
                </config>
        </configurations>
</variant>
```

Listing 1: Varianzpunkt Wegfahrsperre in XML-Darstellung

Wird bei der Erzeugung des endgültigen Prozessmodells die Konfiguration *vorhanden* für den Varianzpunkt Wegfahrsperre gewählt, so wird das nachfolgend in Abbildung 11 dargestellte Modell des Toplevel-Prozesses mit *Init_WFS* und *Wegfahrsperre* erzeugt.

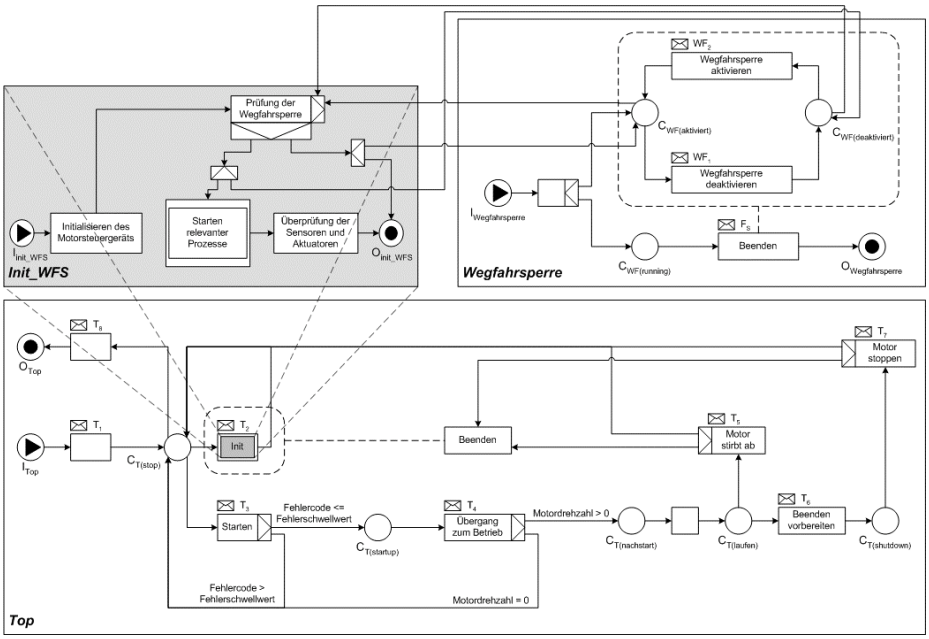


Abbildung 11: Wegfahrsperre

4.2 Varianzpunkt für Lambdasonde mit Heizmöglichkeit

Im Gegensatz zur Variante der Wegfahrsperre, bei welcher die Auslagerung des Teilprozesses *Init* schon im ursprünglichen Modell modelliert wurde, müssen bei der Variante einer Lambdasonde mit Heizmöglichkeit einige Änderungen am ursprünglichen Modell vorgenommen werden.

Wird eine Lambdasonde mit Heizfunktion verwendet, so ist die Berechnung der Einspritzmenge nicht mehr von der Temperatur des Katalysators abhängig, sondern von der Temperatur der Lambdasonde. Dafür muss ein zusätzlicher Überwachungsprozess eingeführt werden, welcher für die Temperatur der Lambdasonde zuständig ist.

Weiterhin muss der Prozess für die Berechnung der Einspritzmenge angepasst werden, um die wiederverwendbaren Prozessteile in einen eigenen Teilprozess auszulagern und die Abhängigkeiten von verschiedenen Überwachungsprozessen in einzelne Teilprozesse (Einspritzmenge_KAT/_LBD) zu kapseln. Tabelle 2 zeigt die Beschreibung dieses Varianzpunktes.

Name		Lambdasonde mit Heizmöglichkeit
Anzahl		2
Ausprägungen (Varianten)		vorhanden
		nicht vorhanden
Auswirkungen (betroffene Prozesse)		Einspritzmenge
		Einspritzmenge_KAT
		Einspritzmenge_LBD
		Lambdasondentemperaturüberwachung
		Katalysatortemperaturüberwachung
Konfigurationen	vorhanden	Einspritzmenge
		Einspritzmenge_LBD
		Lambdasondentemperaturüberwachung
	nicht vorhanden	Einspritzmenge
		Einspritzmenge_KAT
		Katalysatortemperaturüberwachung

Tabelle 2: Varianzpunkt Lambdasonde mit Wegfahrsperre

Die XML-Darstellung des Varianzpunktes wird im folgenden Listing dargestellt.

```
<?xml version="1.0" encoding="UTF-8"?>
<variant name=”Lambdasonde mit Heizmöglichkeit” number=”2”>
        <definition>
                <specifications>
                        <spec name=”vorhanden” />
                        <spec name=”nicht vorhanden” />
                </specifications>
                <effects>
                        <location name=“Einspritzmenge“ />
                        <location name=“ Einspritzmenge_KAT“ />
                        <location name=“ Einspritzmenge_LBD“ />
                        <location name=
                                “Lambdasondentemperaturüberwachung“ />
```

```
                        <location name=
                                "Katalysatortemperaturüberwachung" />
                </effects>
        </ definition >
        <configurations>
                <config name="vorhanden">
                        <location name="Einspritzmenge" />
                        <location name=" Einspritzmenge_LBD" />
                        <location name=
                                "Lambdasondentemperaturüberwachung" />
                </config>
                <config name="nicht vorhanden">
                        <location name="Einspritzmenge" />
                        <location name=" Einspritzmenge_KAT" />
                        <location name=
                                "Katalysatortemperaturüberwachung" />
                </config>
        </configurations>
</variant>
```

Listing 2: Varianzpunkt Lambdasonde mit Heizmöglichkeit in XML-Darstellung

Wird bei der Erzeugung eines Prozessmodells die Variante *vorhanden* gewählt, so wird ein Prozess nach dem in Abbildung 12 gezeigten Prozessmodell erzeugt. Zu beachten ist die Auslagerung der Nicht-Varianten Prozessteile in einen eigenen Teilprozess sowie die Abhängigkeit von einem Zustand des Prozesses zur Überwachung der Temperatur der Lambdasonde statt des Katalysators (grau markierte Condition).

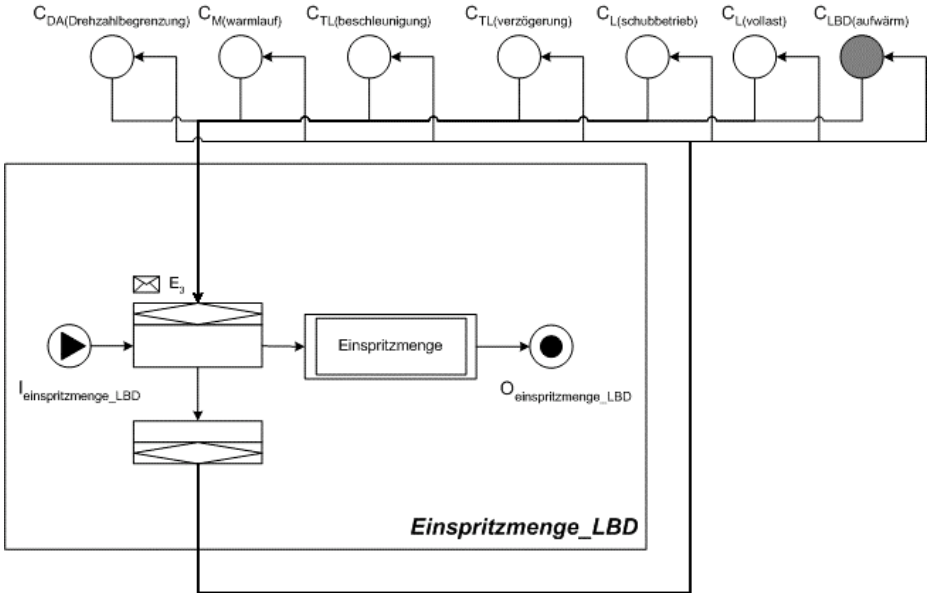


Abbildung 12: Einspritzmenge_LBD

Die zusätzlichen Prozessmodelle, die für die Realisierung dieses Varianzpunktes notwendig sind, können im Anhang (Abbildung 24-26) nachgeschlagen werden. Weitere ähnlich zu modellierende Prozessvarianten betreffen z.B. das Vorhandensein einer Fahrmodussteuerung, eines Turboladers, einer Abgasrückführkontrolle (AGR-Ventil), einer Geschwindigkeitsbegrenzung oder einer Antischlupfregelung.

5. Vergleich verschiedener Modellierungstechniken

Um eine qualitativ möglichst gute Aussage über die Eignung des in dieser Arbeit gewählten Ansatzes der Modellierung zu ermöglichen, sollen in diesem Kapitel die entwickelten Modelle mit den Modellierungsmöglichkeiten der *Unified Modelling Language* (UML) verglichen werden.

UML als technische Modellierungssprache besitzt gegenüber YAWL den Vorteil, dass sie weit verbreitet ist und ihr Einsatz als Modellierungssprache für technische Systeme allgemein als Standard angenommen wurde. Es ist mit UML möglich sowohl die Zustände (siehe Abbildung 13) als auch die Aktivitäten (siehe Abbildung 14) eines Prozesses zu modellieren.

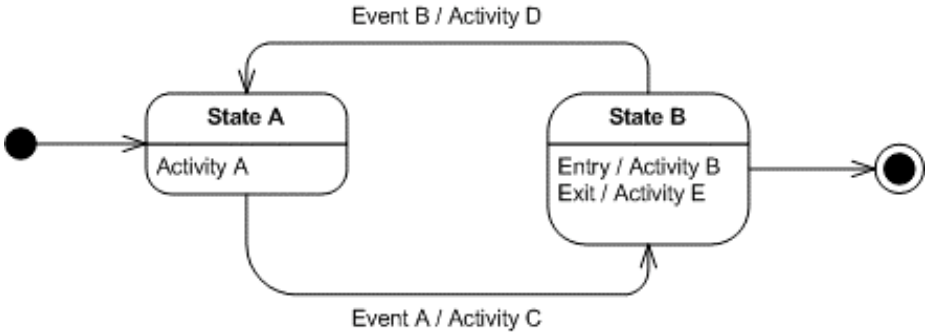


Abbildung 13: UML Zustandsdiagramm

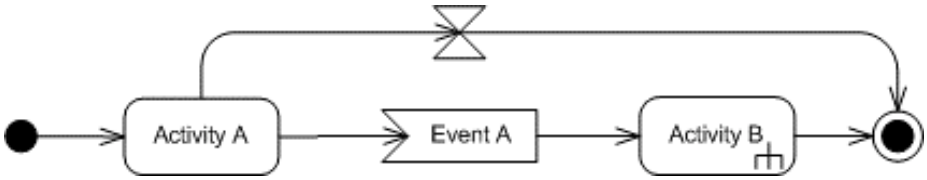


Abbildung 14: UML Aktivitätsdiagramm

Zustände und Aktivitäten können dabei sowohl in jeweils eigenen Diagrammtypen modelliert und betrachtet werden als auch innerhalb eines Diagramms kombiniert werden. Dies entspricht der Modellierung von Conditions und Tasks innerhalb eines YAWL-Diagrammes.

Zusätzlich können in UML nicht nur die Zustandsübergänge mit Aktivitäten versehen werden, sondern es können auch innerhalb eines Zustands Aktivitäten ausgeführt

werden (siehe Abbildung 13). Weiterhin ist auch eine Verfeinerung der Zustände möglich. Dies könnte zwar mit Hilfe von Konstrukten wie den Stellen-berandeten Teilnetzen der Petri-Netze modelliert werden, wird aber in YAWL nicht definiert oder unterstützt.

UML ist im Gegensatz zu YAWL nicht nur auf die grafische Modellierung des Kontrollflusses mit Zuständen und Aktivitäten beschränkt, sondern verfügt über diverse Diagrammtypen für unterschiedliche Aspekte technischer Systeme. Dazu gehören z.B. Elemente zur Modellierung von Kommunikationssequenzen (Sequenzdiagramme), statischen Datenstrukturen (Klassendiagramme) oder Deployment-Konfigurationen (Deploymentdiagramme).

Eine Modellierung der hier betrachteten ECU zur Steuerung eines Benzinmotors mittels UML findet sich in [11] und [12].

6. Zusammenfassung und Schlussfolgerungen

Im Verlauf der Arbeit wurde gezeigt, dass es möglich ist Software-gesteuerte Automatisierungsprozesse aus dem Automobilbereich mit YAWL zu modellieren.

Jedoch wird der Sinn einer solchen Modellierung angesichts etablierter und technisch orientierter Modellierungsstandards wie UML schnell fragwürdig. Dies trifft vor allem deshalb zu, weil einige grundsätzliche Vorstellungen wie z.B. die Abhängigkeit zwischen Prozessen nicht zufrieden stellend modelliert werden konnten. Auch konnten nur einige wenige Workflow-Pattern wie z.B. einfache Routing-Konstrukte oder Cancel Pattern in den PESOA-Spezifikationen wiedergefunden werden.

Weiterhin können bestimmte wichtige Aspekte von Softwareprozessen wie z.B. Deploymentbeschreibungen oder Kommunikationssequenzen aufgrund der Orientierung YAWLs auf Geschäfts- und Entwicklungsprozesse und der Abhängigkeit von den Petrinetzen nicht in die Modellierung eingehen.

Als Vorteil von YAWL gegenüber der UML könnte man den grundsätzlich sehr einfachen Aufbau von YAWL anführen, jedoch ist zurzeit nur die Modellierung des Kontrollflusses in YAWL möglich. UML dagegen bietet eine Reihe von Diagrammtypen, mit denen sich Softwaresysteme mit allen notwendigen Aspekten und in verschiedenen Abstraktionsebenen modellieren lassen. Somit sollte bei der Modellierung von Automatisierungsprozessen bzw. von Software-Prozessen weiterhin UML verwendet werden.

Literaturangaben

[1] W.M.P. van der Aalst, A.H.M. ter Hofstede: *YAWL: Yet Another Workflow Language* (revised version), QUT Technical report, 2003

[2] Department of Technology, Eindhoven University of Technology, *Workflow-Pattern,* http://www.workflowpatterns.com

[3] Gamma et al: *Design Pattern*. Addison-Wesley Verlag, 1996.

[4] Object Management Group, *Unified Modeling Language* (UML), http://www.omg.org/technology/documents/modeling_spec_catalog.htm

[5] W.M.P. van der Aalst, Kees van Hee: *Workflow Management*. MIT Press, Cambridge, Massachusetts / London, England, 2002

[6] Denmark University of Aarhus: *Petri Nets World,* http://www.daimi.au.dk/PetriNets/

[7] Centre for Infomation Technology Innovation, Queensland University of Technology, *Yet Another Workflow Language* (YAWL), http://www.citi.qut.edu.au/yawl

[8] Prof. Dr. Mathias Weske: Unterlagen zur *Vorlesung Workflow-Management* (HPI 03/04), http://www.hpi.uni-potsdam.de/intern

[9] PESOA Projekt, Daimler/Chrysler AG

[10] Prof. Dr. Peter Liggesmeyer: *Software-Qualität: Testen, Analysieren und Verifizieren von Software*, Spektrum Akademischer Verlag, Heidelberg / Berlin, 2002

[11] Phillip Sommer, Modellierung einer Motorsteuerung mit UML 2.0 Statecharts und Sequenzdiagrammen, Seminar Prozessmodellierung, Hasso Plattner Institut, SS2004

[12] Anja Bog, *Technische Prozesse – Motorsteuerung 2 (UML Activity Diagrams)*, Seminar Prozessmodellierung, Hasso Plattner Institut, SS2004

Anhang

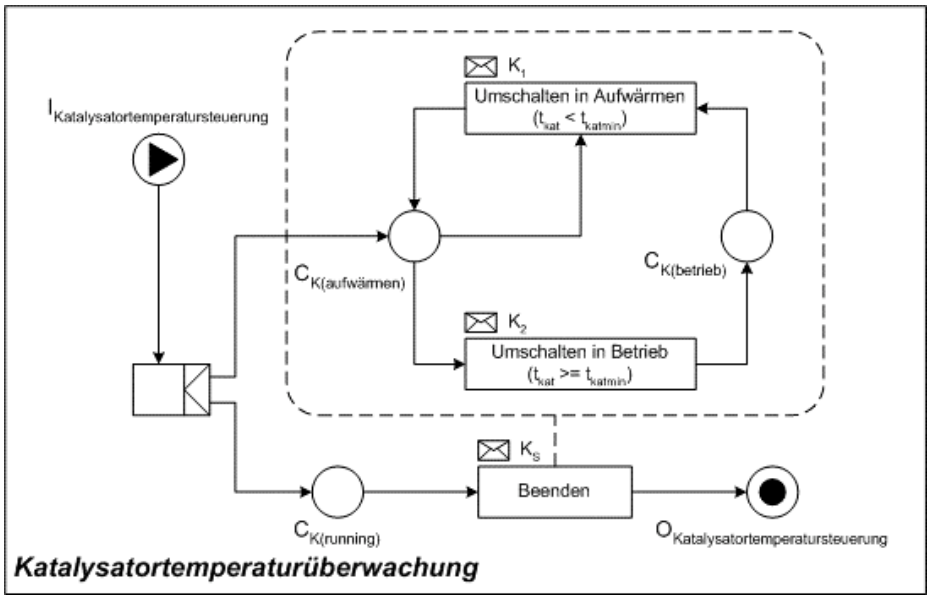


Abbildung 15: Katalysatortemperaturüberwachung

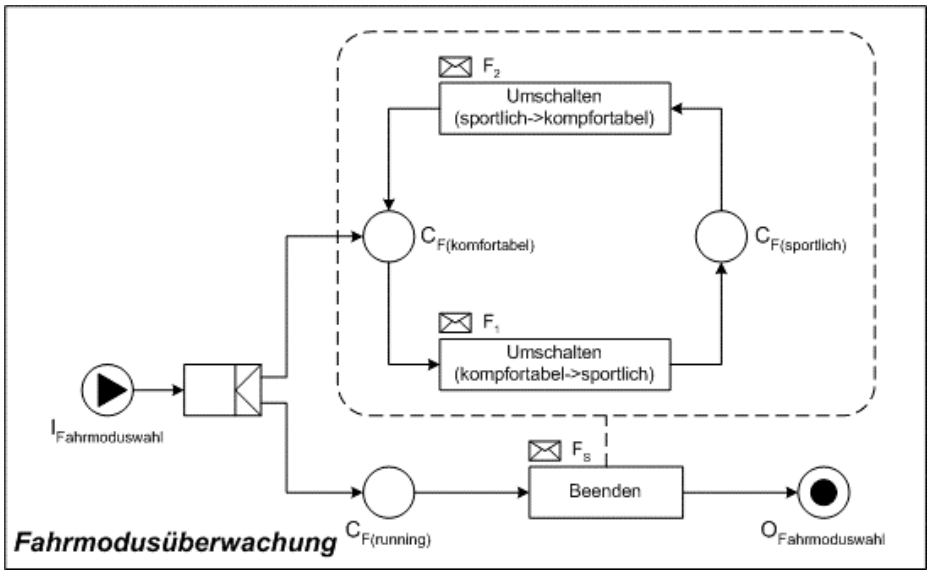


Abbildung 16: Fahrmodusüberwachung

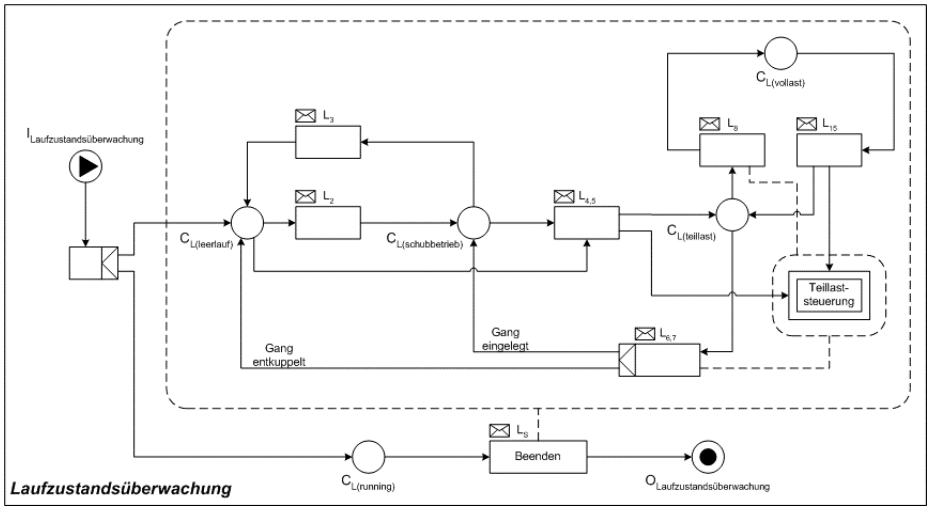


Abbildung 17: Laufzustandsüberwachung

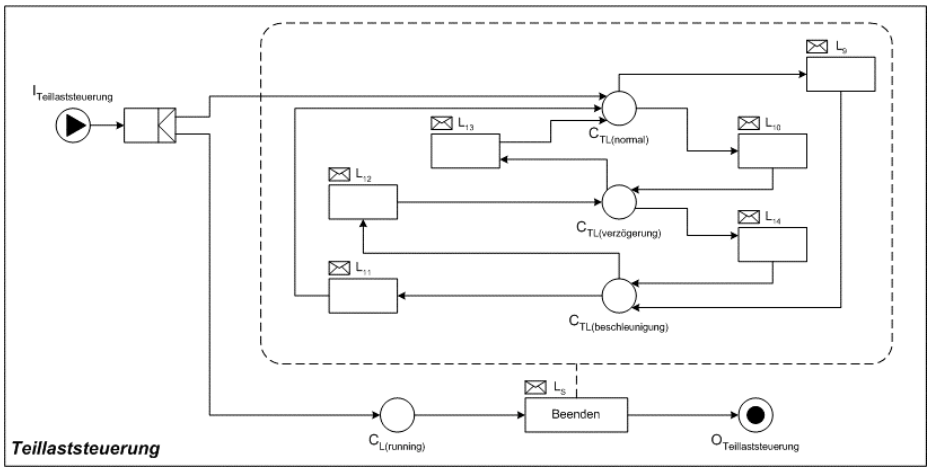


Abbildung 18: Teillaststeuerung

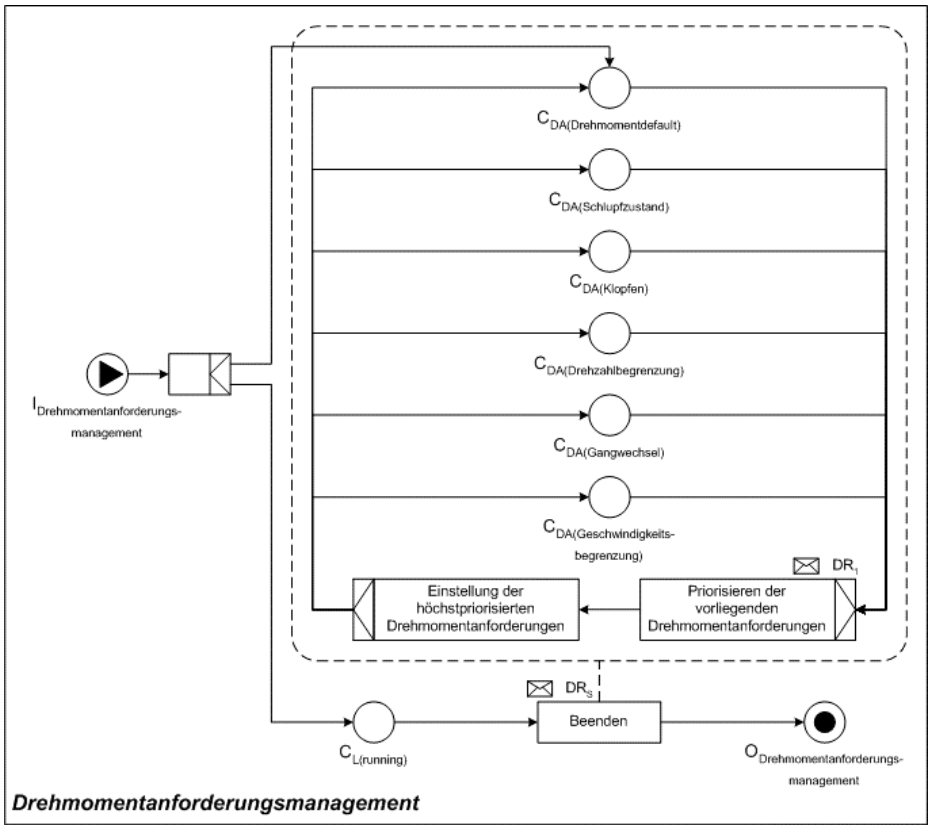


Abbildung 19: Drehmomentanforderungsmanagement

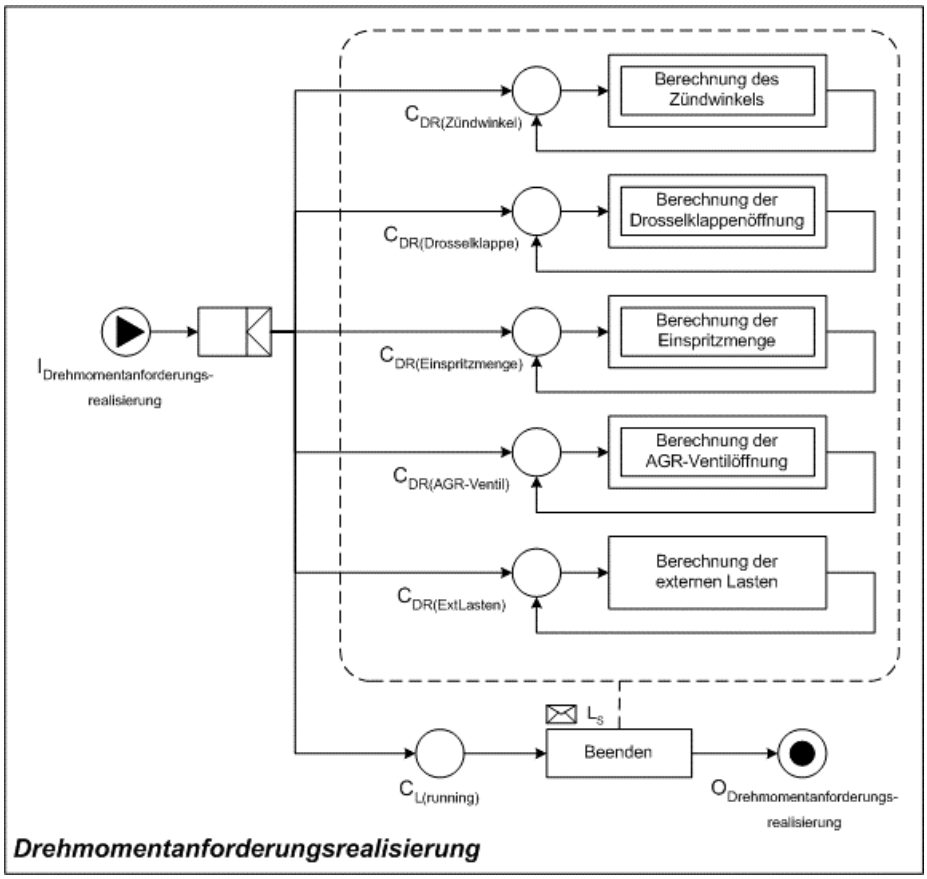


Abbildung 20: Drehmomentanforderungsrealisierung[5]

[5] Die Berechnung der externen Lasten wurde nur der Vollständigkeit halber mit modelliert, da hierzu keine genaueren Informationen in den PESOA Prozesspezifikationen aufgeführt sind.

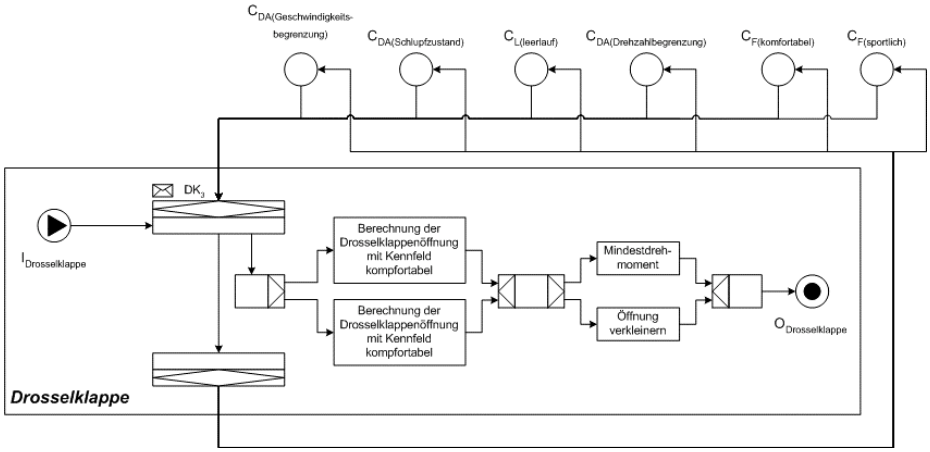


Abbildung 21: Drosselklappensteuerung

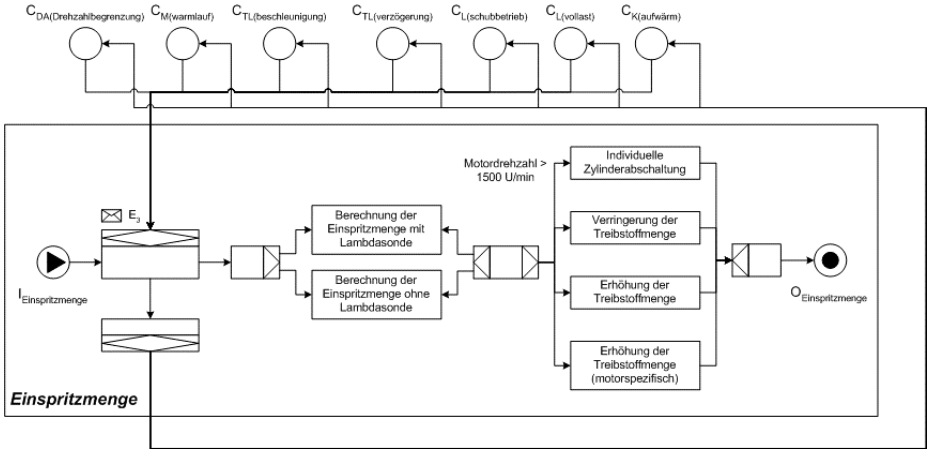


Abbildung 22: Einpritzmengensteuerung

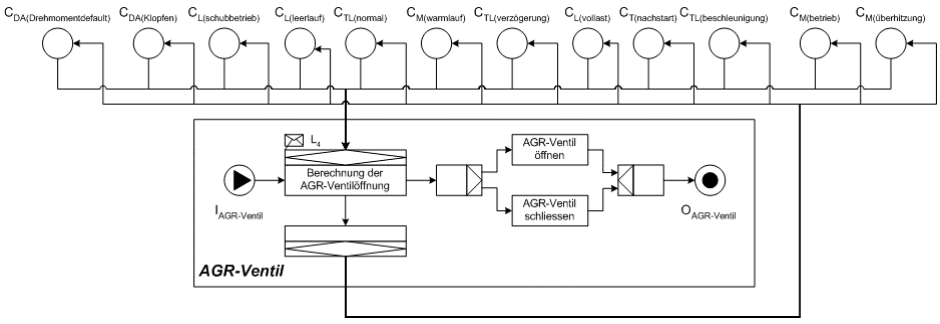


Abbildung 23: AGR-Ventilsteuerung

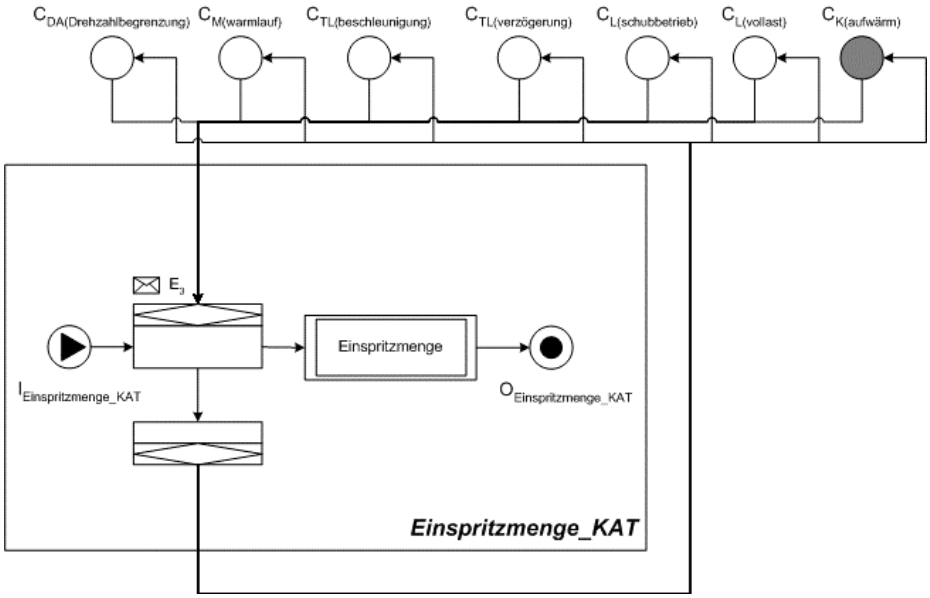


Abbildung 24: Einspritzmenge_KAT

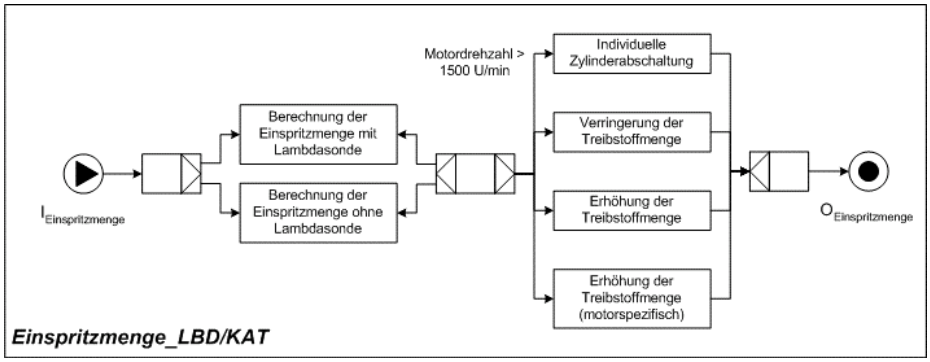


Abbildung 25: Einspritzmenge für Variante Lamdasonde mit Heizmöglichkeit

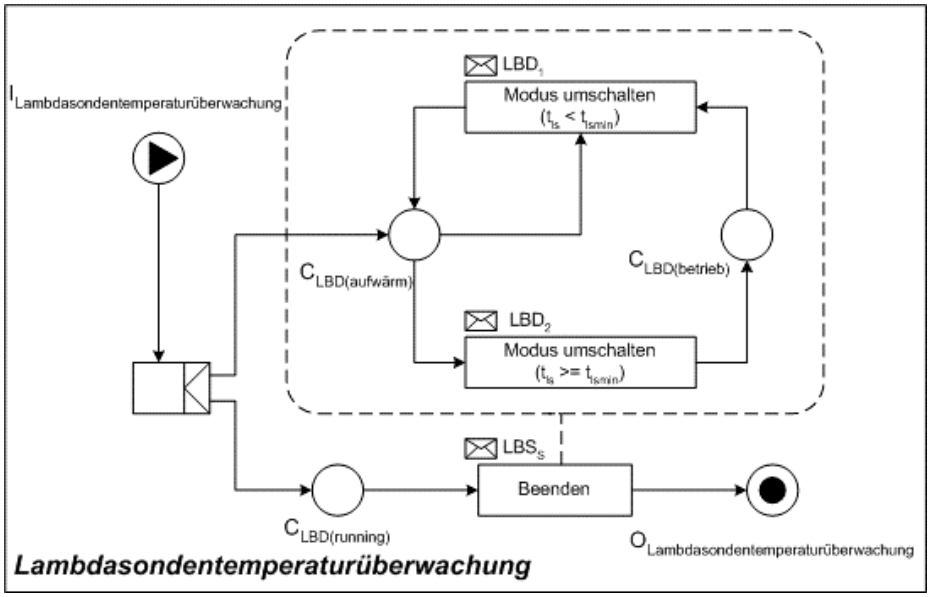


Abbildung 26: Lambdasondentemperaturüberwachung

Beschaffungsprozesse am HPI

Silvan T. Golega

Seminar Prozessmodellierung 2004
Hasso Plattner Institut für Softwaresystemtechnik
Universität Potsdam

Silvan.Golega@hpi.uni-potsdam.de

Zusammenfassung: In dieser Arbeit werden die Prozesse betrachtet, die am Hasso Plattner Institut in Potsdam den ordnungsgemäßen Ablauf von Beschaffungen sicherstellen. Eingebettet in das ARIS Modell werden diese Prozesse in der Workflow Modelliersprache YAWL dargestellt.

Während der Prozess auf mögliche Schwachstellen untersucht wird, soll insbesondere auf die Vorzüge und Nachteile von YAWL zur Modellierung von Geschäftsprozessen eingegangen sowie Verbesserungsvorschläge zu der noch jungen Sprache offeriert werden. Insbesondere wird überprüft, ob sich die verschiedenen ARIS Sichten in YAWL, als Modelliersprache für die Steuerungssicht, integrieren lassen.

Schlüsselwörter: Geschäftsprozessmodellierung, Beschaffungsprozesse, YAWL, HPI Verwaltung, ARIS;

1 Einleitung

Das Hasso Plattner Institut ist ein privat finanziertes Institut, das an die Uni Potsdam angegliedert ist. Mittlerweile sind über 60 Mitarbeiter in Wissenschaft und Verwaltung damit betraut, einen ordnungsgemäßen Lehrbetrieb sicherzustellen. Ständig werden Waren geordert und Dienstleistungen Dritter in Anspruch genommen. Der Prozess muss sicherstellen, dass nur Berechtigte solche Beschaffungen vornehmen und Budgets dabei nicht überschritten werden.

Das erste Kapitel dieser Arbeit „Vorgehensweise“ stellt die zur Informationsbeschaffung und -darstellung verwendeten Techniken vor: Interviews, ARIS und YAWL. Danach werden die zur Prozessmodellierung nötigen Daten vermittelt: die Organisation des HPI, beteiligte Rollen und zu verwaltende Datensätze werden geklärt. Der dritte Teil der Ausarbeitung beschäftigt sich mit dem Prozess an sich und seinen Schwachstellen, der vierte versucht, Optimierungsansätze aufzuzeigen. In diesem Kontext soll YAWL mit anderen Modelliersprachen verglichen werden, um so die Eignung von YAWL zu testen, Geschäftsprozesse darzustellen.

2 Vorgehensweise

Dieses Kapitel beschäftigt sich mit den Techniken, die verwendet wurden, um diese Arbeit zu erstellen. Da die behandelten Prozesse nur teilweise dokumentiert sind (siehe [1]), wurden die Informationen in Interviews mit an Beschaffungen beteiligten HPI Verwaltungsmitarbeitern gesammelt.

Die gesammelten Daten wurden nach dem so genannten ARIS Haus gegliedert, einer in der Wirtschaft sehr erprobten Vorgehensweise. Statt der zu ARIS gehörenden Notation zur Modellierung von Abläufen wurde jedoch YAWL für die Prozessdarstellung gewählt, einer relativ jungen Sprache, die in einigen Aspekten noch eher wissenschaftlichen denn professionellen Charakter zeigt.

2.1 Interviews

Es erwies sich als sinnvoll zunächst die Personen zu interviewen, die hauptsächlich mit Beschaffungen betraut sind. Am HPI sind das die Abteilungen für Buchhaltung, Controlling und Postannahme. Diese Mitarbeiter sind am stärksten in den Ablauf von Beschaffungen involviert, kennen den Ablauf im Regel- und in Sonderfällen und können so einen guten Überblick vermitteln. Es empfiehlt sich daher, den Interviewpartner möglichst viel über seine Arbeit mit Beschaffungen berichten zu lassen, unverfälscht durch zu viel Nachfragen durch den Interviewer. So wird man umfassend informiert, insbesondere auch in Aspekten, nach denen man sich als Unbedarfter nicht erkundigt hätte.

Erst nach dem ersten Modellierungsversuch sollten in weiteren Interviews gezielte Fragen gestellt werden, oder weitere an Beschaffungen beteiligte Personen herangezogen werden, um Detailfragen zu klären und auch Schwächen des Ist-Prozesses aufzudecken.

2.2 ARIS

Das ARIS Toolset ist ein Werkzeug der Firma IDS Scheer AG zur Unterstützung des Softwareentwicklungsprozesses, das sich in vielen Bereichen der Softwareindustrie etabliert hat. Es beruht auf einem Modell, das die Software in verschiedene Aspekte gliedert, dem so genannten ARIS Haus. Die in den Interviews gewonnenen Informationen wurden nach diesem ARIS Modell gegliedert, weshalb hier ein kurzer Überblick dazu geboten werden soll.

Das ARIS Haus (siehe Abb. 1) gliedert sich in fünf Sichten: Organisationssicht, Funktionssicht, Datensicht, Steuerungssicht und Leistungssicht. Jede dieser Sichten gibt das Modell des Geschäftsprozesses unter einem bestimmten Aspekt wieder:

In der Organisationssicht werden alle Ressourcen (Menschliche Arbeitskräfte, Maschinen, Hardware), d. h. alle Organisationseinheiten zusammengefasst und hierarchisiert. Die Organisationsschicht bildet somit das Dach des Hauses und sagt aus, wer an dem System beteiligt ist.

Funktions- und Datensicht bilden die Säulen und dekomponieren die Systemteile aus der funktionellen Perspektive, bzw. bringen alle generierte Daten und Umfelddaten sowie Schriftverkehr, Dokumente etc. in Relation.

Die Steuerungssicht stellt den Kern des Systems dar: sie integriert die vorangegangenen Sichten in einen logischen und zeitlichen Ablaufplan. Im ARIS Toolset gibt es hierzu eine eigene ereignisgesteuerte Notation. In dieser Arbeit wird jedoch YAWL als Workflow Sprache verwendet. Es liegt also nahe, YAWL auf die Integrationsfähigkeit der vorangegangenen Sichten zu überprüfen und, wo nötig und möglich, in diese Richtung zu erweitern. So werden die Funktionen durch die Tasks bestimmt, Ressourcen können als Rollen auftauchen und Daten als operationeller Zustand gesehen werden.

Die fünfte Sicht, die Leistungssicht stellt den Unterbau des Hauses dar und befasst sich mit der der Software zugrunde liegenden Architektur, wie Schichtenmodellen, Komponenten, Infrastruktur, etc. Von dieser Schicht wird hier abstrahiert, da nur auf die Geschäftsprozesse, nicht auf mögliche Implementierungen dieser in Software eingegangen werden soll.

Des Weiteren wird jede Sicht im ARIS Haus in drei Ebenen aufgeteilt. Das Fachkonzept strukturiert Prozesse mit informatikfremden Beschreibungsmodellen (hier wurden z.B. Organigramme und Funktionsbäume verwendet), das DV-Konzept setzt das Fachkonzept mit informatiknahen Beschreibungen (hier z.B.: E/R Diagramme, YAWL) um. Diese Arbeit bewegt sich in diesen beiden ersten Ebenen, ohne dazwischen zu unterscheiden. Die dritte, die Implementierungsebene wird hier nicht betrachtet. Sie stellt die technische Realisation des Systems dar.

Die orthogonalen Sichten des ARIS Modells führen einerseits sehr gut an den Prozess heran und beschreiben seine Umstände auf bewährte Art und Weise. Daher eignet sich ARIS auch sehr gut, um dieser Ausarbeitung einen gegliederten Rahmen zu geben. Stellt die Steuerungssicht nur kausale Abhängigkeiten von Ereignissen dar, bieten die ARIS Pläne ebenfalls einen guten Überblick. Bei der Integration aller Aspekte jedoch scheinen riesige, unübersichtliche Pläne zu entstehen. Dieses Dokument versucht, dieses Problem mit YAWL zu lösen.

2.3 YAWL

Eine geeignete Sprache zur Modellierung von Geschäftsprozessen sollte insbesondere zwei Kriterien erfüllen: sie sollte einerseits so einfach sein, dass auch Laien ohne lange Einarbeitungszeit die Bedeutung von Graphen verstehen und andererseits doch mächtig genug um auch komplexe System mit vertretbarem Aufwand darzustellen.

YAWL ist eine Workflow Sprache die auf einer Weiterentwicklung von Petrinetzen beruht. Ihre wenigen Elemente und die Aufteilung in Tasks und Conditions machen sie somit sehr verständlich. Dennoch bietet die Sprache Unterstützung für fast alle Workflowpattern und gewährleistet somit die nötige Ausdrucksstärke. Die Existenz einer Engine, die YAWL Netze abwickeln kann, scheint zudem eine Integrationsfähigkeit der verschiedenen ARIS Sichten zu bedeuten. Dies bleibt jedoch noch zu verifizieren. Da in anderen Arbeiten dieser Vortragsreihe YAWL bereits eingeführt wurde, soll die Sprache hier nicht weiter erklärt werden. Stattdessen wird auf [3] verwiesen.

3 HPI Beschaffungen

Um die Beschaffungsprozesse modellieren zu können, müssen erst ihre Umstände, also die Organisation von Beschaffungen am HPI geklärt werden. Dieses Kapitel befasst sich daher mit dem statischen Umbau des ARIS Hauses. Die Funktionssicht klärt den Begriff der Beschaffung, zeigt die verschiedene Beschaffungsarten und deutet Unterfunktionen an. Daran beteiligte Rollen werden in der Organisationssicht identifiziert. Zuletzt beschreibt die Datensicht die nötigen Informationen, die zur Abwicklungszeit der Prozesse gehalten werden.

3.1 Funktionssicht: Arten von Beschaffungen und ihre Unterfunktionen

Beschaffungen sind „jede Form von Beauftragung eines fremden Unternehmens zur Lieferung von Gegenständen oder zur Erbringung von Leistungen (z.B. Dienstleistungen)“(Zitat aus [1], Seite 2).

Dazu gehören alle Arten von Warenbeschaffungen wie der IT Infrastruktur des HPIs und spezieller Projekte, wie der Materialbeschaffung von Bewirtungsbedarf, Hygieneartikeln und Büromaterialien. Weiter zählen hierzu Dienstleistungen wie die Wartung des Hauses, der Außenanlagen, der Technik und des Mobiliars. Ebenso werden Medien dazugerechnet, wie Strom, Wasser und Gas, die Personalbeschaffung, also Gehälter, aber auch Räumlichkeiten, und nötige Ausstattungen sowie Dienstreisen, Schulungen und Reisekosten sowie alle sonstigen Aufkommen und Ereignisse, die bezahlt werden müssen, wie die Organisation von Fachtagungen, von Öffentlichkeitsarbeit, Versicherungen und Absolventenfeiern.

Die vorliegende Arbeit konzentriert sich auf Warenbeschaffungen, die per Post geliefert werden. Hierbei lassen sich grob einige Subfunktionen herauskristallisieren: Beschaffungen werden genehmigt, Waren werden bestellt, empfangen und bezahlt. Des Weiteren unterliegen Warenbeschaffungen wie alle anderen regelmäßigem Controlling.

Die hier genannten Funktionen können leicht in Unterfunktionen aufgegliedert werden herab bis auf die Ebene, in der die einzelnen Schritte definiert werden müssen, die letztendlich die Tasks der Steuerungssicht bilden. Dies soll hier nicht geschehen.

3.2 Organisationssicht: Beteiligte Rollen

Aus diesen Subfunktionen ergeben sich direkt Rollen. Es muss Leute geben, die eine Beschaffung beantragen, die sie genehmigen und die sie kontrollieren. Ebenso muss jemand die Ware bestellen, liefern, empfangen, und bezahlen. [2], gegliedert dargestellt in Abb. 3, gibt Aufschluss über die Mitarbeiter des HPI in der Verwaltung und klärt so die ersten Rollen.

Die Genehmigung von Anträgen wird durch die jeweils zuständige Abteilung, beziehungsweise den Geschäftsführer durchgeführt, das Controlling selbstredend durch die Controllingabteilung. Waren werde durch den Antragsteller selbst bestellt, von Extern, dem so genannten Lieferanten geliefert, an der Rezeption von der hier Postab-

teilung bezeichneten Mitarbeitergruppe empfangen und im Regelfall durch die Buchhaltung bezahlt.

Ist der Antragsteller jedoch ein Wissenschaftlicher Mitarbeiter, so ist er Mitglied eines Lehrstuhls, wodurch der Professor die Aufgabe der zuständigen Abteilung übernimmt. In Abb. 4 sind diese Rollen zusammengefasst, wobei die Controllingabteilung einerseits zwar auch dem Geschäftsführer unterstellt ist, andererseits jedoch durch eine Notwendigkeit, die Geldmittelausgabe zu stoppen, jede Beschaffung unterbinden kann. Der Lieferant kann bei anderen Beschaffungsarten durch einen Dienstleister, Hersteller oder auch Angestellten ersetzt werden, im Falle, der in dieser Arbeit betrachtet wird, ist es ein Postunternehmen.

3.3 Datensicht

Prinzipiell kann jeder Mitarbeiter Beschaffungen beantragen. Je nach sinnvoller Zuordnung wird ein Antrag bei der Zuständigen Abteilung oder bei einem Professor beantragt. Bei der Ausgabe von HPI Geldmitteln liegt die letzte Entscheidung immer beim Geschäftsführer. Professoren können jedoch weitere Drittmittel zur eigenen Verfügung haben. Da die Hierarchie in Abb. 5: Genehmigungshierarchie von Beschaffungen unter dem Aspekt von Geldmitteln gesehen werden soll, sind Weisungslinien, die bei bestimmten Geldmitteln nicht bestehen, gestrichelt gezeichnet.

Jede Verwaltungsabteilung kann in ihrem Zuständigkeitsbereich über Geldmittel bis zu 1.000 € pro Beschaffung frei verfügen. Höhere Beträge müssen durch den Geschäftsführer genehmigt werden. Das Gleiche gilt für Professoren und Assistenzprofessoren, wenn es um HPI Mittel geht. Bei Drittmitteln ist der Professor alleine entscheidungsbefugt, muss aber durch die Controllingabteilung klären lassen, ob das Drittmittelprojekt noch genügend Geldkapazitäten aufweist und ob die geplante Beschaffung der Drittmittelbeschreibung genügt.

Aus [1] sind noch einige weitere Regeln zu entnehmen. So sind bei Beschaffungen über 2.500 € sind Preisverhandlungen zu führen, über 25.000 € müssen mindestens drei Angebote eingeholt werden. Diese und weitere Grenzen sind jedoch sehr weit gefasst, so dass fast jede Abteilung sich eigene, strengere Regeln aufstellt, die den Prozess an sich jedoch kaum verändern.

Unabhängig davon führt dieselbe Controllingabteilung für HPI und für Drittmittel regelmäßige Kontrollen durch und warnt Abteilungen und Lehrstühle, die nahe an den Grenzen ihres Budgets sind. Feste Grenzen gibt es hierbei nicht, jedoch Erfahrungswerte aus den letzten Jahren, die nicht überschritten werden sollten.

Einen Sonderfall bildet die Fachschaft. Auch Studenten können außerordentliche Anträge an den Geschäftsführer stellen. Sie verfügen aber auch über eigene Geldmittel. Diese entstammen nicht dem HPI, sondern werden wie allen Fachschaften der Universität von dem Allgemeinen Studien Ausschuss und der Vereinigung der Fachschaften zugeteilt. Über die zugeteilten Mittel kann frei verfügt werden, der Verwendungszweck muss jedoch strengen Regeln genügen, deren Einhaltung jährlich durch eine Studentenkommission kontrolliert wird. Der gesonderte Beschaffungsprozess bei Fachschaften wird hier nicht weiter betrachtet.

Bei den HPI Beschaffungsdaten in Abb. 6 ist der zentrale Datensatz die Beschaffung. Jede Beschaffung hat verschiedene Daten, wichtig ist hier, dass gespeichert werden muss, ob die Ware bezahlt wurde und ob sie defekt ist. Bestellungen tauchen auf in Controlling- und Buchungslisten. Die Controllingliste erfasst über einen Zeitraum alle Ausgaben einer Mittelherkunft, die Buchungsliste erfasst pro Abteilung oder Lehrstuhl und pro Mittelherkunft sämtliche Bestellungen. Jede Abeilung und jeder Lehrstuhl führt zudem Liste über die Lieferanten, um so in der Lage zu sein, sich positive und negative Erfahrungen mit bestimmten Lieferanten zu merken.

4 HPI Beschaffungsprozess

4.1 Steuerungssicht

Abb. 7 zeigt den Prozess, wie er im Moment am HPI praktiziert wird. Einzelne Sequenzen wurden dabei an einigen Stellen zusammengefasst, um die Größe nicht ausufern zu lassen. Dieser Prozess soll im Folgenden im Detail erklärt werden, wobei auch die Funktionen identifiziert werden, die in Kap. 2.1 ermittelt wurden. Bei der Diskussion werden Probleme des Ist Prozesses sowie der Integrationsfähigkeit der Funktionssicht in YAWL deutlich werden, denen im darauf folgenden Kapitel 4 Lösungen geboten werden sollen.

4.2 Integration der Funktionssicht

Die Subfunktionen der Warenbeschaffung aus Kapitel 2.1 lassen sich in Abb. 7 schon erahnen. Die Warenbestellung ist ein eigener Task, die Beschaffungsgenehmigung, der Warenempfang und ihre Bezahlung sind in den grau unterlegten Bereichen zu erkennen. Dazugekommen ist die Fehlerbehandlung, falls es Probleme mit dem Lieferanten gibt. Das Controlling taucht nicht auf, da es als eigenständiger Prozess, logisch und zeitlich unabhängig von dem ersten, gesehen werden muss.

Abb. 8 zeigt zwei Prozesse, notiert in YAWL, die nur aus den Subfunktionen bestehen. Wenn es gelingt, den Beschaffungsprozess so umzustrukturieren, dass der neue Prozess durch Abb. 8 hierarchisierbar ist, wäre die direkte Integration der Funktionssicht in die Steuerungssicht geglückt.

Im Folgenden sollen nacheinander die grau unterlegten Bereiche untersucht werden, ob sie zu den entsprechenden Subprozessen umformbar sind. Sowohl die Warenbestellung wie auch die Problemlösung sind hier atomare Tasks, die je nach Fall unterschiedlich behandelt werden müssen. Auf Controlling wird am Ende des Kapitels eingegangen.

4.2.1 Antragsgenehmigung

Die Einflussfaktoren der Antragsgenehmigung wurden bereits in Kap. 2.3.1 genannt, Abb. 9 bringt ihre Überprüfung in die richtige Reihenfolge. Der erste Task „Antrag stellen“ steht dabei auch für die Suche nach Angeboten und ihre Auswahl, die individuell vorgenommen werden.

Hier wird sogleich ein Problem der Hierarchisierung deutlich: in YAWL Netzen, auch in Subnetzen, ist immer nur eine definierte Endstelle erlaubt. Entscheidungsinformationen können also nicht durch Steuerungszustände in die übergeordneten Netze weitergereicht werden. In Abb. 7 jedoch verlassen Pfeile den grauen Bereich, der die Antragsgenehmigung umfasst, mit zwei verschiedenen Zuständen als Ziel: entweder wurde der Antrag genehmigt, oder abgelehnt, wobei im zweiten Falle der Prozess beendet werden kann.

Die Lösung besteht darin, zwei neue Tasks einzuführen: „Antrag genehmigen“ und „Antrag ablehnen“ speichern diese Entscheidung als externe Datensätze. Im übergeordneten Prozess (Abb. 8) kann das Or-Split am Ende des Tasks „Antrag genehmigen“ dann diesen Datensatz für seine Entscheidung benutzen. Auf gleiche Weise wird auch die Mittelherkunft, die im Task „Mittelherkunft prüfen“ bestimmt wird, im operationellen Zustand gespeichert, sodass Buchung und Controlling später darauf zurückgreifen können. Hier ist deutlich erkennbar, dass auch die Datensicht in die Darstellung der Steuerungssicht integriert werden sollte. Möglichkeiten hierzu werden im Kapitel 4.2 untersucht.

4.2.2 Warenempfang und Bezahlung

Auch beim Warenempfang werden Daten produziert, die den weiteren Prozess beeinflussen, jedoch nicht als Conditions modelliert werden können, wenn das Modell hierarchisierbar bleiben soll. Das HPI Reglement verlangt von sich aus, dass Nachnahme vermieden werden soll, wo immer es möglich ist, da in diesem Fall die Ware bezahlt werden muss, bevor sie geprüft werden kann. Nachnahme bringt aber noch weitere Probleme mit sich. Die Postabteilung muss vor Eintreffen der Ware über die Nachnahme informiert sein, um Geld besorgen zu können und die Ware überhaupt annehmen zu dürfen, die Buchungsabteilung muss über die Nachnahme informiert sein, um zu verhindern, dass die Ware später bei der Buchung ein zweites Mal bezahlt wird (siehe Abb. 10).

Ein großer Vorteil von YAWL ist es, fast alle Workflow Pattern zu unterstützen. Es böte sich daher an ein Konzept aus diesem Bereich zu übernehmen, in diesem Fall Meilensteine. Durch Meilensteine ist das hier betrachtete Problem jedoch nicht zu lösen, da der Prozess hierarchisierbar bleiben soll.

Die Frage ist berechtigt, warum die Subfunktionen „Warenempfang“ und „Warenbezahlung“ in getrennten Subprozessen dargestellt werden sollen. Stattdessen wäre ein Subprozess „Warenempfang“ denkbar, der zwei Tasks „Ware bezahlen“ enthält, einem im Falle der Nachnahme, einen im Falle von Bezahlung per Rechnung. In dieser, in Abb. 11 gezeigten Idee würden also ebenso beide Subfunktionen erkennbar bleiben und der Prozess würde die Notwendigkeit vermeiden, sich operationell zu merken, ob die Ware bereits gezahlt wurde.

Im Falle einer defekten Ware muss der Task Problemlösung aufgerufen werden, und zwar ohne eine etwaige Rechnung zu bezahlen oder die Beschaffung zu buchen. Da YAWL Netze nur eine definierte End-Condition besitzen, besteht das Problem also weiterhin: im operationellen Zustand muss nun nicht nur gespeichert werden, ob die Ware bezahlt wurde, sondern auch, ob die Ware defekt war. Wie man diesen Datenbezug modellieren könnte wird Kapitel 4.2 untersucht.

4.2.3 Controlling

Die Prozesse des vorigen Abschnitts endeten alle mit dem Task „Buchen“. Bei der Buchung wird die Beschaffung im System gespeichert, wobei Mittel und Kostenstelle vermerkt werden, um späteres Controlling zu ermöglichen. Controlling wird in einem vom Geschäftsführer bestimmten Rhythmus durchgeführt, zurzeit einmal im Monat. Sobald eine Abteilung die Zahlungsmittel zu überschreiten droht, die für einen Zeitraum vorgesehen waren, wird sie gewarnt und gehalten, nur noch die nötigsten Ausgaben zu machen. Drittmitteletats können nicht überschritten werden, da hier vor jeder Beschaffung die Mittelverfügbarkeit überprüft wird.

Zwischen Controlling- und Beschaffungsprozess werden also Daten ausgetauscht. Es macht daher Sinn, den Controllingprozess aus Abb. 12 mit dem hier diskutierten Beschaffungsprozess in ein Diagramm (Abb. 13) zusammenzufassen.

Controlling wird dabei zyklisch in einer Endlosschleife durchgeführt. Beliebig viele Beschaffungen können parallel dazu nebenläufig auftreten. Sollte es zu einem Zeitpunkt keine Beschaffungen geben, wird in diesem Teil des Prozesses einfach gewartet, bis der nächste Antrag gestellt wird. Das Procedere kann nur unterbrochen werden, wenn die eingebetteten Subprozesse geändert werden sollen, oder das Institut geschlossen wird. In diesem Fall müssen die noch laufenden Beschaffungen jedoch individuell geregelt werden (Task „Fehlerbehandlung“).

5 Optimierung

Das letzte Kapitel zeigte den Ist-Zustand der Beschaffungsprozesse am HPI. Dabei ist bereits auf Aspekte der Modellierung eingegangen worden, indem versucht wurde, die Prozessdarstellung so zu hierarchisieren, dass die in der Funktionssicht gefundenen Systemfunktionen auf einzelne, atomare oder zusammengesetzte Tasks der Steuerungssicht abgebildet werden können. Hierbei wurden Probleme zweierlei Art festgestellt.

Zum Einen Probleme des Prozesses: da es am HPI keinen globalen Ort des Datenaustausches bezüglich Beschaffungen gibt, stellt der Prozess nicht sicher, dass Ware nicht doppelt bezahlt wird oder im Falle von fehlerhafter Ware zurückverlangt wird. Ebenso sind gewisse Benachrichtigungen (z.B. der Postabteilung über kommende Ware) davon abhängig, dass Personen an sie denken. Es stellt sich also die Frage, ob ein automatisierter Prozess diese Probleme lösen kann, und wie er gestaltet werden soll.

Die zweite Art der aufgetretenen Probleme betreffen die Modellierung: Entscheidungen im Prozess sind abhängig von Daten. Daher wäre es sinnvoll, diese zu modellieren. Da jedoch die Funktionssicht mittels Hierarchisierung des Prozesses integriert werden soll, verbieten sich Ansätze, die wie z.B. Meilensteine Conditions benutzen. Also müssen die Daten als operationeller Zustand gesehen werden. Wird dieser in YAWL integriert, so kann der Bezug zwischen Datensicht und Steuerungssicht hergestellt werden. Als letzter Punkt bleibt die Integration der Organisationssicht.

5.1 Prozessoptimierung

Der Prozess an sich bietet wenig Optimierungspotenzial. Es reicht, die nötigen Benachrichtigungen sicherzustellen, um sicherzustellen, dass nur vollständig funktionsfähige angenommene Ware einmalig bezahlt wird. Dies kann am einfachsten durch Automatisierung geschehen. Dafür muss der Prozess definiert und mit einem Workflowsystem ausgeführt werden.

Der große Vorteil einer IT-gestützten Prozessabwicklung ist im Falle der HPI Beschaffungen, dass der Prozess an sich beibehalten werden kann, wie er bis jetzt von den Beteiligten ausgeführt wird. Dadurch dürften auch die Annahmehürden bei den Mitarbeitern leichter überwunden werden. Der zu implementierende Prozess schaut dabei im Wesentlichen so aus, wie er in Abb. 7 dargestellt wurde, ergänzt um die Sequenzen, die zusammengefasst wurden. Dadurch werden alle am Prozess Beteiligten automatisch benachrichtigt. So wird automatisch sowohl ausgeschlossen, dass Benachrichtigungen vergessen werden, und so z.B. die Postabteilung die Ware nicht annehmen kann, als auch, dass Benachrichtigungen doppelt erfolgen und Waren dadurch ungewollt mehrfach bezahlt werden.

Ein weiterer positiver Nebeneffekt ist, dass die Datenhaltung werkzeuggestützt und zentralisiert wird. Verschiedene Beteiligte müssen nicht mehr die gleichen Daten in ihr System speisen. Redundante Datenhaltung entfällt und dadurch auch die unwissentliche Bezahlung von Waren, die bereits per Nachnahme bezahlt wurden. Auch können so globale Lieferantenlisten eingeführt werden, in denen jeder Erfahrungen mir Lieferanten einsehen kann.

Will man den Prozess und alle mit ihm verbundenen Regeln jedoch komplett implementieren, so muss eine besondere Anforderung an das Workflow System gestellt werden. Da z.B. jede Abteilung eigene Regeln festsetzt, ab wann sie mehrere Angebote einholt, muss dies individuell, also zur Laufzeit konfigurierbar sein.

5.2 Integration der Datensicht

Die Entwickler von YAWL stellen neben einem Wergzeug zur Modellierung auch eine Engine zur Verfügung, die die erstellten Netze abwickeln kann. Soll ein Prozess jedoch gestartet werden, so sind neben der reinen Steuerungssicht noch weitere Angaben nötig: es muss definiert werden, wer die einzelnen Tasks auszuführen hat, und auf welchen Daten dabei gearbeitet wird.

Bei der YAWL Engine geschieht das nach dem Einlesen der Pläne in zusätzlichen, eigens dafür vorgesehenen Dialogen. Wie ARIS verschiedene Sichten kennt, wird YAWL um Perspektiven erweitert. YAWL selbst stellt die Kontrollperspektive dar, die Datenperspektive definiert Variablen, auf denen gearbeitet werden kann, die Ressourcenperspektive beteiligte Rollen und die Operationssicht die Funktionalitäten der einzelnen Tasks, und Schnittstellen zu Applikationen, die in den Tasks aufgerufen werden sollen.

Es wäre jedoch aus zwei Gründen schön, diese Informationen in die Ablaufpläne integrieren zu können[1]. Zum einen beeinflusst die Datenhaltung den Prozess und die Entscheidungen. Werden Datenänderungen nur informal zu der Bezeichnung der Task dazugeschrieben, kann die Abwicklerplattform diese Information nicht extrahieren, und auch der Mensch tut sich schwer, die daraus resultierenden Zusammenhänge zu erkennen. Eine definierte Syntax könnte da Abhilfe schaffen. Zum Anderen dient es generell dem Verständnis des Menschen, wichtige Informationen integriert in der selben Zeichnung zu finden, z.B. die Information, welche Abteilung eine gewisse Tätigkeit ausführt, und so können die Erklärungen der Tasks knapp gehalten werden.

5.2.1 Erweiterung von YAWL um Operationszustände

Ein Vorschlag für die Verknüpfung der Steuerungs- mit der Datensicht ist in Abb. 14. dargestellt. Die relevanten Datensätze sind seit Abb. 6 bekannt. Lesender Zugriff auf sie von Tasks aus wird dabei durch einen eingehenden, schreibender durch einen ausgehenden Pfeil symbolisiert. Das Symbol für den lesenden Zugriff kann man sich daher in der Vorstellung durch eine Condition ersetzen, die nur belegt ist, wenn die Variable den angegeben Zustand hat und die durch ein And-Join notwendige Ausführungsbedingung für den Task ist. Lesende XOr-Splits legen abhängig von der aktuellen Belegung der eingelesenen Variable nur eine Marke in die folgenden Conditions oder aktivieren nur einen der folgenden Task. Der schreibende Zugriff legt dann, wenn der Wert der Variablen geändert wird, in die entsprechenden imaginären Conditions Marken, beziehungsweise nimmt sie heraus.

Es ist jedoch gedacht, dass dies über die Grenzen der Subprozesse global für den gesamten Prozess funktioniert. Alternativ ist eine Deklaration oder Initialisierung der Variablen aus der Datensicht bei den Starting-Conditions denkbar, so dass die Variable nur in diesem Subprozess und in allen untergeordneten Subprozessen gültig wäre. Zur in ARIS gewählten Vorgehensweise siehe bitte Kap. 4.4.2. Die Kritik, die Netze würden durch die Hinzufügung von Datenzugriffen zu komplex und nicht mehr nach formalen Kriterien wie Soundness überprüfbar, sei hier zurückgewiesen: die Prozesse müssen ohnehin um die Variablen erweitert werden, unabhängig, davon, ob diese in den Netzen dargestellt werden, oder nicht.

Abb. 15 zeigt die Alternative, in der der Warenempfang sowie die Warenbezahlung zu einem Subprozess zusammengefasst wurden. Die Antragsgenehmigung ändert sich dabei nicht, weshalb sie nicht wieder aufgeführt wird.

Der am HPI aktuell benutze Prozess ist nicht fähig, auf den Fall zu reagieren, wenn bestellte Ware nicht geschickt wird. Die Mitarbeiter würden also vom Prozess abweichen um diesen Sonderfall zu behandeln. Ein maschinenabgewickelter Prozess jedoch muss für dieses Ereignis gerüstet sein. Die beiden Netze wurden daher um einen Timer erweitert, der sicherstellt, dass die Zeit, die auf die Ware gewartet wird, begrenzt ist. Die definierten Variablen aus der Datensicht ermöglichen es nun, die Information an die „Problemlösung“ weiterzugeben, ob die Ware angekommen ist und ob sie bereits bezahlt wurde.

[1] Auch die Entwickler von YAWL haben diese Möglichkeit bereits in Erwägung gezogen, wie [4] andeutet.

5.2.2 Weiterreichung von Events

Spätestens wenn Prozesse von firmenexternen Personen abhängen, entstehen Fehlerquellen wie hier die defekte oder nicht verschickte Ware. Ein Prozess, der in der Praxis benutzbar sein soll, muss diese Fehler abfangen. In YAWL ist dazu das Cancellation Pattern eingeführt worden. Oft sollen die Fehler, die nur an bestimmten Orten in Subprozessen abgefangen werden, jedoch erst im übergeordneten Prozess behandelt werden. Mit Variablen kann dies einfach modelliert werden. Als Beispiel ist in Abb. 16 das Ereignis aufgeführt, dass nach einer bestimmten Zeitspanne die Ware noch nicht empfangen wurde. Das Timeout-Event setzt in diesem Falle den Operationszustand so, dass die Problemlösung im übergeordneten Prozess ausgeführt werden kann, und dadurch die im Timeout übrig gebliebene Marke entfernt,

Diese Modellierung kann man als Erweiterung von YAWL um die Möglichkeit verstehen, Events auszulösen. Ob sie genutzt werden sollte, ist allerdings genau zu untersuchen. Immerhin wird dadurch Kontrollfluss über Subnetzgrenzen hinweg dargestellt und, wenn man so will, die bewusste Einschränkung von YAWL umgangen, nur eine End-Condition pro Teilnetz zu erlauben.

5.3 Integration der Organisationssicht

Die Integration der Organisationssicht ist nun kein Problem mehr. Daten, die eine passive, lesbare Ressource darstellen, sind nötig, damit ein Task schalten kann. Das Eintreten eines Events ist Vorraussetzung für die Ausführung. Ebenso muss sich ein Mensch oder eine andere aktive Ressource zuwenden, um die Aufgabe zu erledigen.

Man kann also das gleiche Symbol verwenden um darzustellen, welche der in der Organisationssicht dargestellten Rollen für die Ausführung eines Tasks verantwortlich ist. Die Analogie ist hier eine Condition, die belegt wird, wenn die Rolle freie Kapazitäten hat. Abb. 17 zeigt nun den endgültigen Prozess, der Abb. 14 aus Kap. 4.2.1 um Rollenangaben ergänzt und damit alle ARIS Sichten (außer der Leistungssicht, siehe Kap. 1.2) vereint. Ein Task ohne Rollenangabe wird vom System ausgeführt.

5.4 Weitere Modelliersprachen

5.4.1 Prozess als Petrinetz

Abb. 18 stellt einen Ausschnitt des Prozesses mit einem Petri Netz dar, wie sie nach FMC Notation dargestellt werden. Die Rollen werden hier sehr deutlich durch Swimlanes hervorgehoben, was einerseits deren Identifikation dient, andererseits die Prozessdarstellung in die Breite zieht und logisch Zusammengehöriges von einander entfernt.

FMC Petrinetze sind ausschließlich dazu gedacht, für das Verständnis von Abläufen zu sorgen. Deswegen sind sie bereits um operationelle Zustände erweiterbar und ihre Transitionen hierarchisierbar. Das Cancellation Pattern von YAWL entspricht bei FMC stellenberandeten Teilnetzen. Auch für Multiple Instance Tasks gibt es eine Entsprechung, die jedoch nicht ganz so komfortabel ist. Somit kann der in YAWL

modellierte Beschaffungsprozess eins zu eins in FMC Petrinetz übersetzt werden und unterscheidet sich dann (fast: siehe unten bezüglich Events) nur in der Formatierung und der Darstellung der Rollen.

Dadurch, dass Teilnetze jedoch nicht mit einer definierten Endstelle aufhören, sondern Marken „nach Außen“ legen, bleibt jedoch die Wahl frei, manche Informationen, statt als operationeller Zustand, als Stelle zu modellieren. Dies kann einerseits hilfreich sein, da bei der Überprüfung der Schaltbereitschaft von Transitionen nicht der abstrakte Operationszustand abgefragt werden muss, sondern die bekannten Schaltregeln gelten. Andererseits verleitet diese Möglichkeit dazu, zu komplizierte Netze zu entwerfen, bei der „üble Tricksereien“ angewendet werden, die zwar syntaktisch und semantisch korrekt sind, jedoch das Verständnis nur erschweren.

Will man einen Prozess nur darstellen, bieten FMC Petrinetze also deutlich mehr Möglichkeiten, auch da die hier für YAWL vorgeschlagene Erweiterung um den operationalen Zustand bereits integriert ist. Ein großes Manko jedoch bleiben jedoch die fehlenden Events. Man kann das Problem durch eine zusätzliche Rolle lösen, die die Events auslöst und Marken auf entsprechende Stellen legt. Dies ist jedoch eine nur wenig elegante Alternative zu den Events in YAWL.

Will man sein Modell jedoch von einem System abwickeln lassen, eignen sich FMC Petrinetze nur sehr bedingt: für die operationelle Erweiterung und die Hierarchisierung müsste erst eine formale Syntax definiert werden. Des Weiteren müsste ein Workflowsystem, das Petrinetze versteht, erst neu entwickelt werden.

5.4.2 Prozess als ARIS Steuerungssicht

In ARIS wird die Integration der verschiedenen Sichten in die Steuerungssicht dadurch erreicht, dass dem Graphen Knoten angefügt werden, die durch ihre Form aufzeigen, welchen Aspekt sie darstellen. In ARIS Steuerungsplänen wird jeder Schritt des Systems durch Ereignisse ausgelöst; anders als YAWL kennt ARIS keine Conditions oder Marken, komplexe Strukturen sind daher nur schwer darzustellen.

Zu jeder Aktivität werden sämtliche Informationen hinzugefügt, die das ARIS Modell verlangt: ihre Funktion, Daten die verwendet werden und Organisationseinheiten. Dadurch wird wesentlich mehr dargestellt. Allerdings entstehen riesige Pläne (Abb. 19), die an Übersichtlichkeit leiden. Es stellt sich die Frage, ob eine Reduzierung nur auf die wesentlichen Daten zu empfehlen ist. YAWL würde jedenfalls das Potential bieten, falls die Syntax in der hier vorgeschlagenen oder einer ähnlichen Weise erweitert wird, die Graphen um für das Verständnis relevante Angaben zu erweitern und in den bereits vorhandenen Dialogen die zuzüglich für die Abwicklung wichtigen Informationen anzugeben. So wird eine Ausführung durch Workflowsysteme möglich und gleichzeitig Verständlichkeit für die Semantik des Prozesses erhöht.

6 Fazit

Die Modellierung von Geschäftsprozessen ist ein Bereich der Informatik, der erst seit jüngerer Zeit seine Bedeutung entfaltet. Methodiken zur Informationserhebung sind bereits aus anderen Disziplinen bekannt, in der Informationsdarstellung für den Men-

schen jedoch konkurrieren aufgrund der spezifischen und neuen Anforderungen noch die verschiedensten Ansätze. Die Entwicklung von graphischen Sprachen und Modellen, aus denen sich Code erzeugen lässt oder die sich sogar automatisiert in maschinenausführbare Darstellungen transformieren lassen, ist ein allgemeiner Trend in der Softwareentwicklung; hier erhält er besondere Bedeutung. Es bleibt abzusehen, welche Verfahren sich dabei durchsetzen.

Für Firmen wird diese Entwicklung viele positive Effekte zur Folge haben. Definierte Prozesse können leichter optimiert werden, Workflow Systeme können flexibel und somit wartungsarm an Prozessveränderungen angepasst werden. Die Erfahrung wird zeigen, in welchen Einsatzgebieten und bei welchen Prozesskomplexitäten die Nutzung solcher Programme sinnvoll ist.

6.1 HPI Beschaffungsprozesse

Am HPI würde eine werkzeuggestützte Prozessabwicklung definitiv die Prozesse unterstützen, Fehlerquellen ausschließen und die Ausführung vereinfachen. Vielleicht würde es Sinn machen, nur die Buchungs-, Controlling-, und Postabteilung entsprechend auszurüsten, die alle täglich mit Beschaffungen befasst sind. Antragsteller würden dadurch nicht umlernen müssen, oder nur wenn sich auch bei ihnen eine Einführung aufgrund von häufigen Beschaffungen lohnt (beispielsweise bei den Netzwerkadministratoren).

Der Erfahrung nach treten bedingt durch die überschaubare Größe des HPIs andererseits nur selten wirkliche Probleme auf, wie die mehrfache Bezahlung der selben Ware, und diese können mittels der kurzen Kommunikationswege leicht gelöst werden. Der Sinn des Aufwandes, ein solches System zu entwickeln und einzuführen muss daher in Frage gestellt werden. Eine Komplettlösung, die auch die weiteren Beschaffungsprozesse wie Dienstleistungen, Personal und regelmäßige Ausgaben integriert, sowie Controlling, Inventur und Buchung unterstützt wäre jedoch prüfenswert.

6.2 YAWL

YAWL hat in dieser Ausarbeitung gezeigt welch gutes Ausdrucksmittel es trotz oder gerade durch seine Schlichtheit ist. Bei leichter Verständlichkeit für unausgebildete Betrachter und schneller Erlernbarkeit der knappen Syntax ist es doch mächtig und ausdrucksstark durch das Konzept der Conditions, die die aus Petrinetzen bekannten Stellen um Semantik erweitern, und die Unterstützung so unterschiedlicher Pattern, von denen hier beispielsweise das Multiple Instance Pattern genutzt wurde.

Probleme traten erst zu Tage, als versucht wurde, andere Softwareaspekte aus dem Ablauf her zu referenzieren. Hier bieten sich jedoch diverse Erweiterungsmöglichkeiten an, unter anderem die hier vorgestellten, so dass sich YAWL auch in bewährte Modelle wie das ARIS Haus integrieren lässt. Es bietet sich insbesondere an, eine Möglichkeit zur Verfügung zu stellen, Rollenbezeichnungen anzufügen, sei es durch Swimlanes oder wie hier durch ein Symbol, und den Zugriff auf Daten, die keinen

Steuerungszustand beschreiben, im Modell darstellbar zu machen. Dadurch können nicht nur die ARIS Sichten in das Modell integriert werden, auch das Hierarchisierungskonzept wird somit verbessert, da Interdependenzen zwischen Subprozessen aufgezeigt werden können.

Quellen

Interviews: Vielen Dank insbesondere an
Monika Zennig
Ute Langer-Lapalus
Monika Köllner

Beschaffungen:
[1] HPI Verfahrensanweisung für Beschaffungen
VA-0602 von Ute Langer-Lapalus
[2] HPI Organisation:
http://www.hpi.uni-potsdam.de/deu/personen/verwaltung.de.html

Modellierung mit YAWL:
Verschiedene Dokumente von
W.M.P. van der Aalst, A.H.M. ter Hofstede et al.
[3] YAWL: http://www.citi.qut.edu.au/yawl/index.jsp
[4] darunter: Design and implementation of the YAWL system
[5] Pattern:
http://tmitwww.tm.tue.nl/research/patterns/documentation.htm

ARIS:
[6] Wikipedia.de:
http://de.wikipedia.org/wiki/ARIS
[7] ARIS House of Business Engineering,
Prof. Dr. August-Wilhelm Scheer
Uni des Saarlandes:
http://www.iwi.uni-sb.de/Download/iwihefte/heft133.pdf
[8] Grundlagen betrieblicher Informationssysteme,
Prof. Dr.-Ing. Dr. h. c. Theo Härder
TU Kaiserslautern:
http://wwwdvs.informatik.uni-kl.de/courses/
GBIS/SS2004/Vorlesungsunterlagen/Kapitel.00.half.pdf

Alle Internetquellen beziehen sich auf Juni 2004

Anhang: Bilder

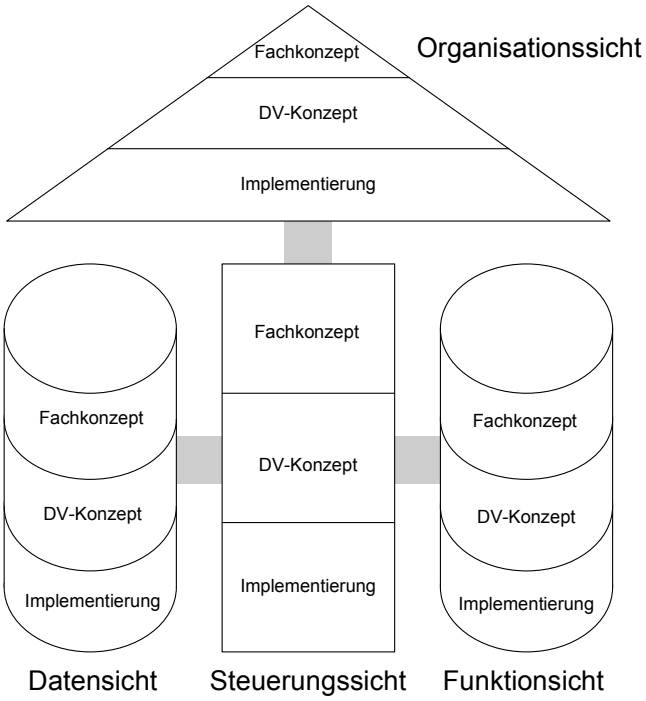


Abb. 1: Das ARIS Haus

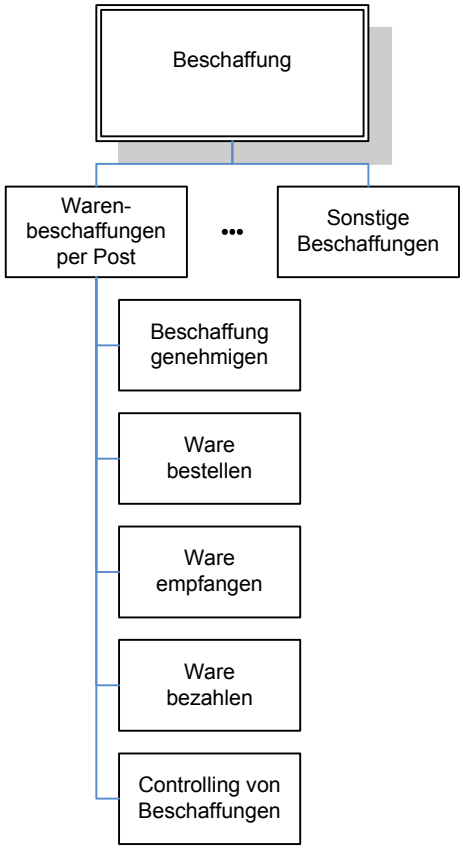


Abb. 2: Beschaffungsfunktionen

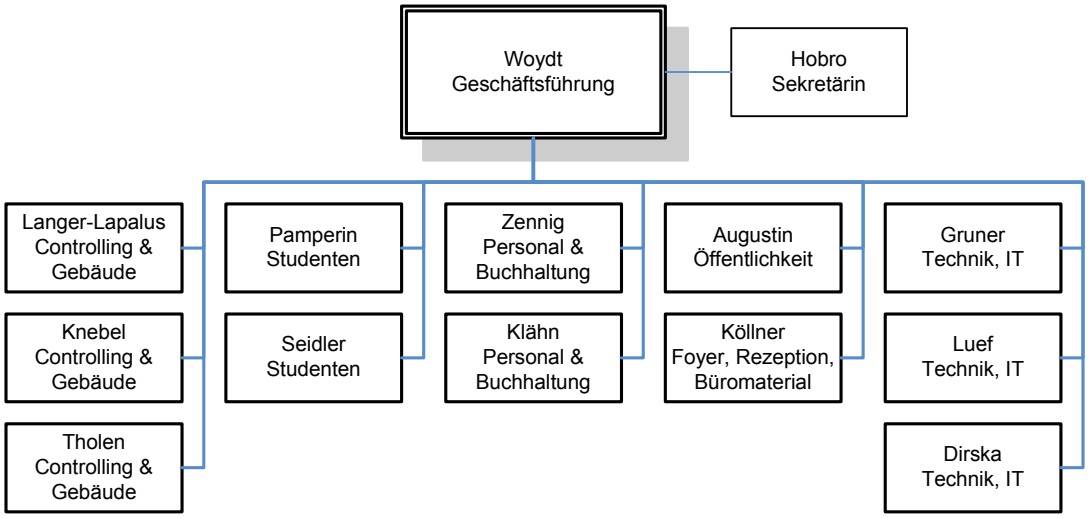


Abb. 3: Wesentliche Verwaltungsmitarbeiter am HPI

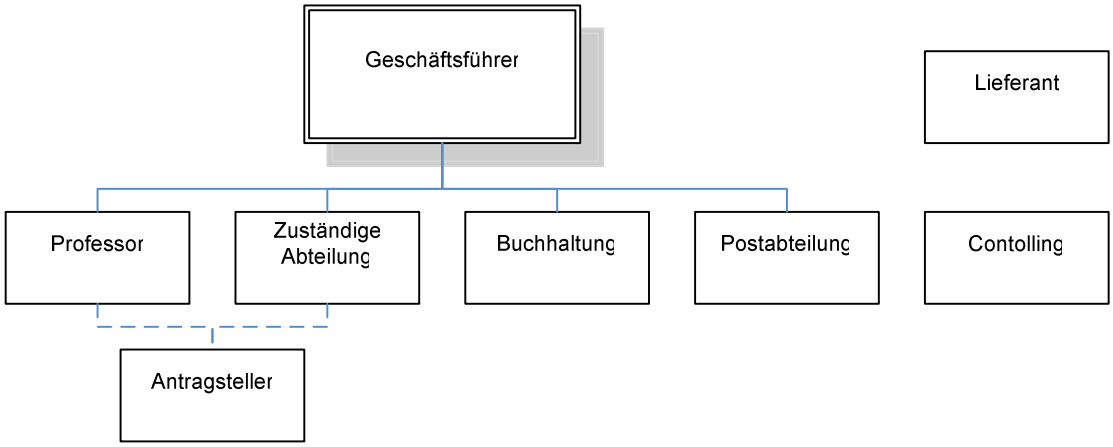


Abb. 4: An Beschaffungen beteiligte Rollen

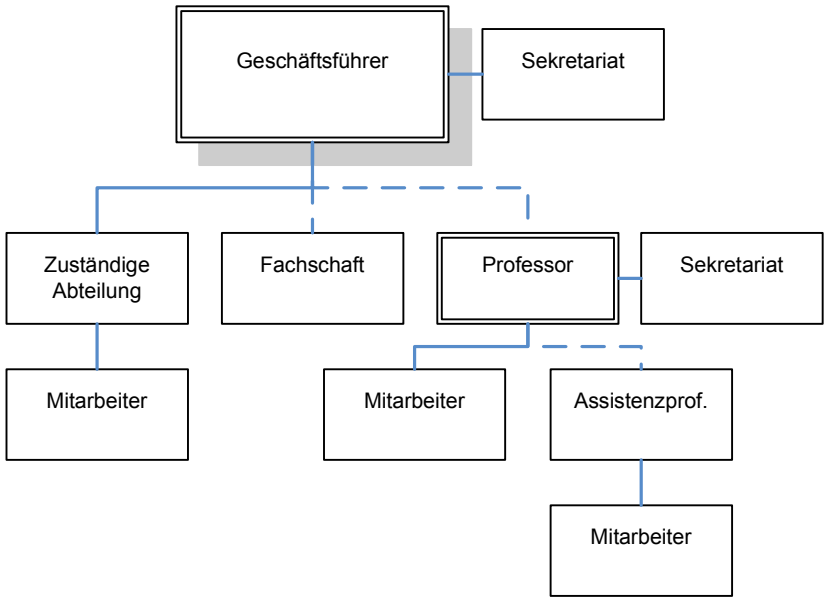


Abb. 5: Genehmigungshierarchie von Beschaffungen

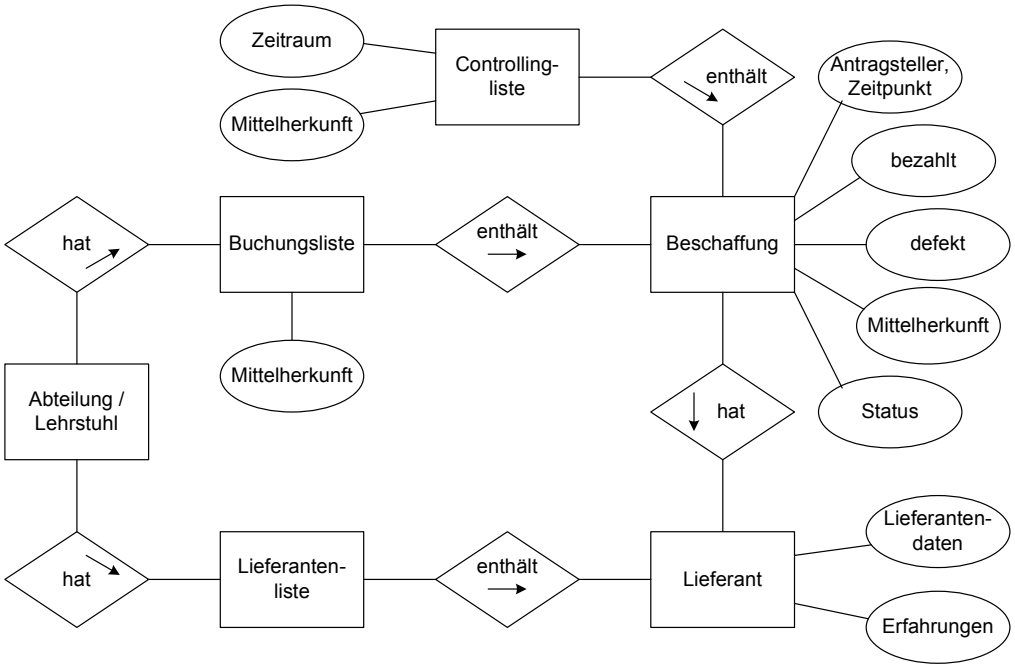


Abb. 6: Beschaffungsrelevante Daten

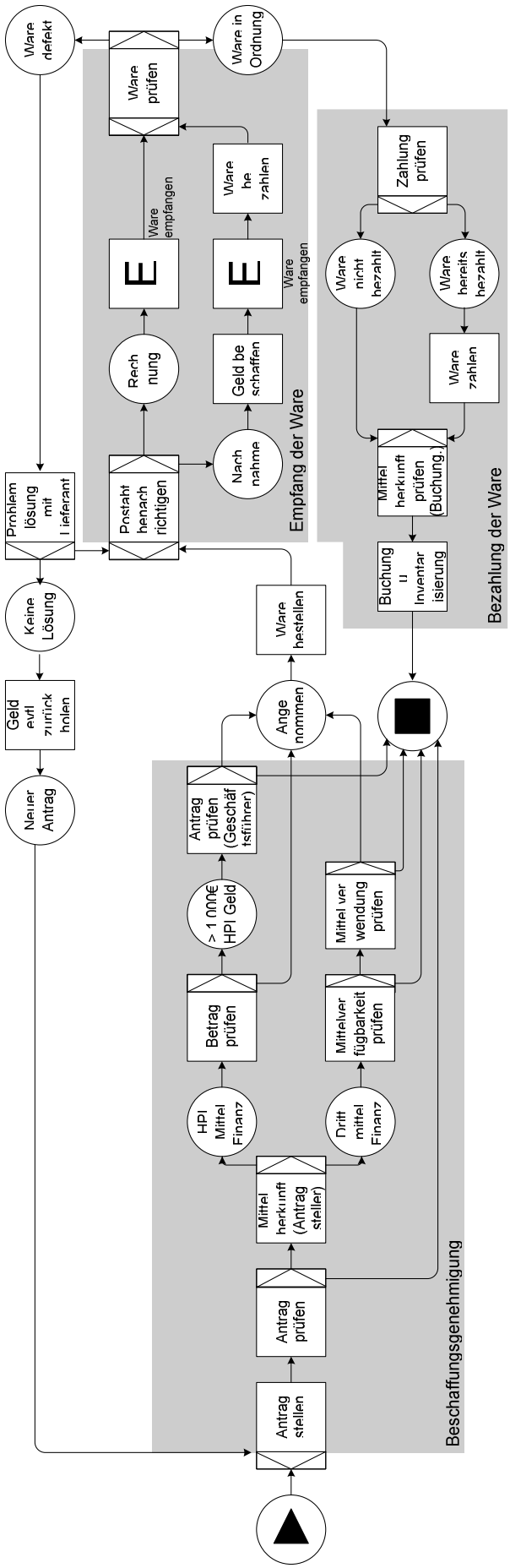


Abb. 7: Ist Prozess

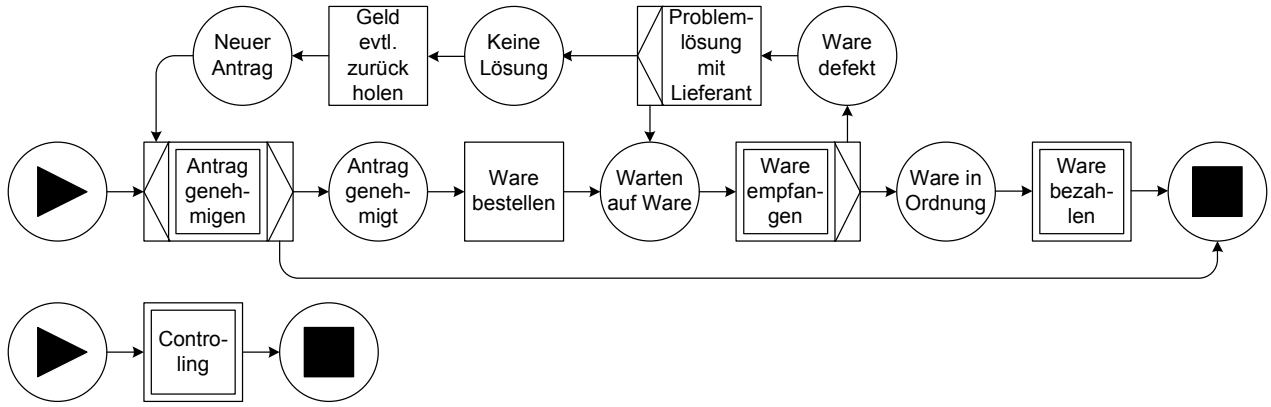


Abb. 8: Prozessüberblick

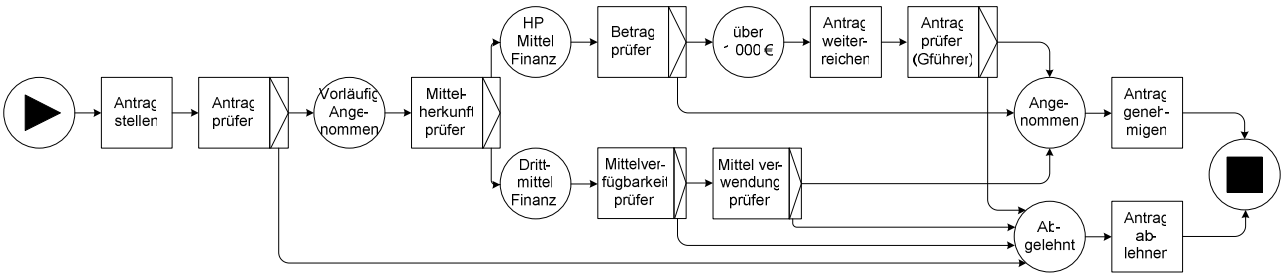


Abb. 9: Antragsprüfung

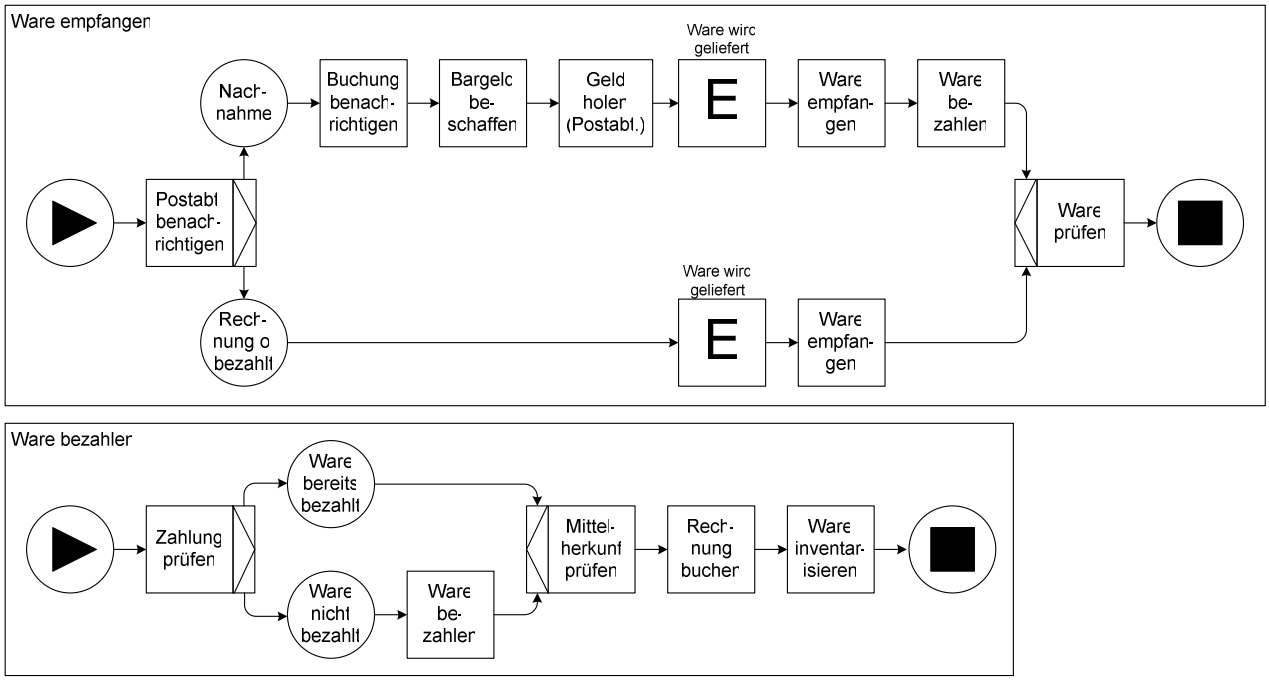


Abb. 10: Warenempfang und –bezahlung getrennt

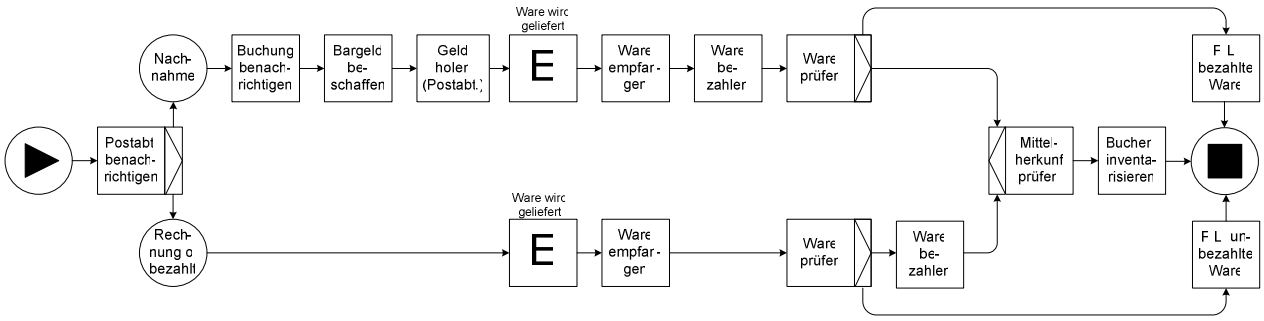


Abb. 11: Warenempfang und -bezahlung zusammen

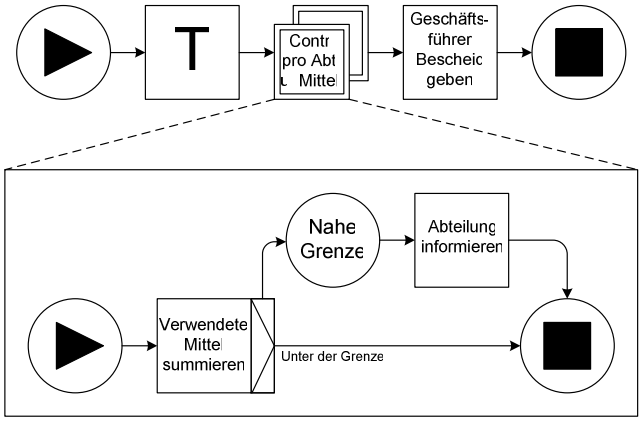


Abb. 12: Controlling

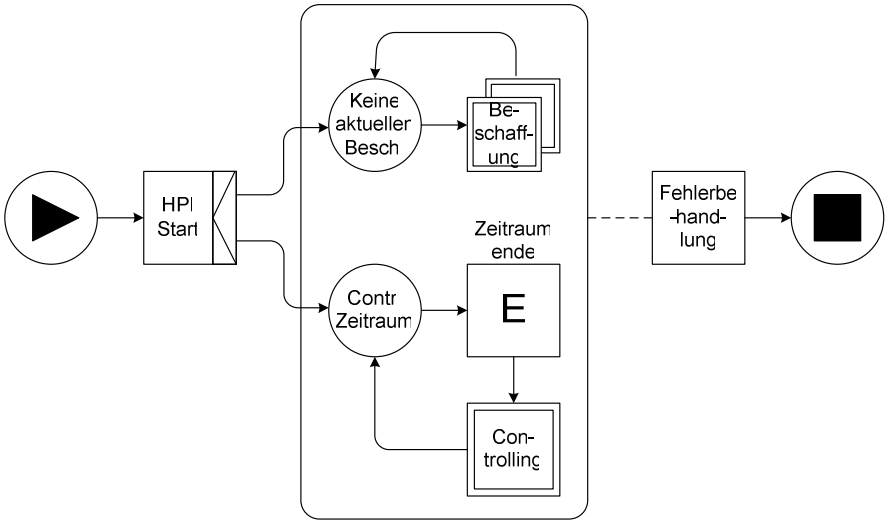


Abb. 13: Gesamtprozess

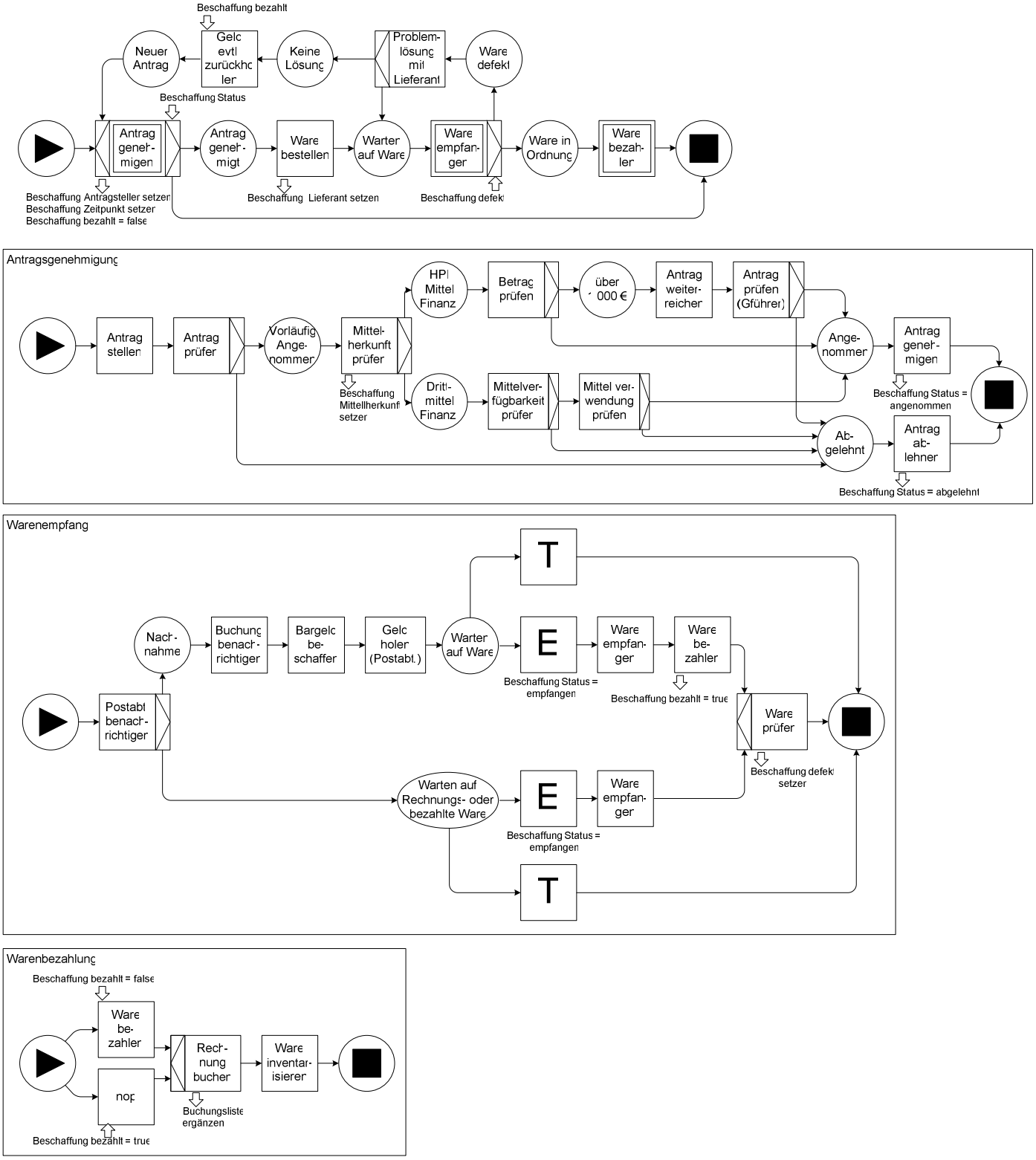


Abb. 14: Beschaffungsprozess, erweitert um Operationszustände

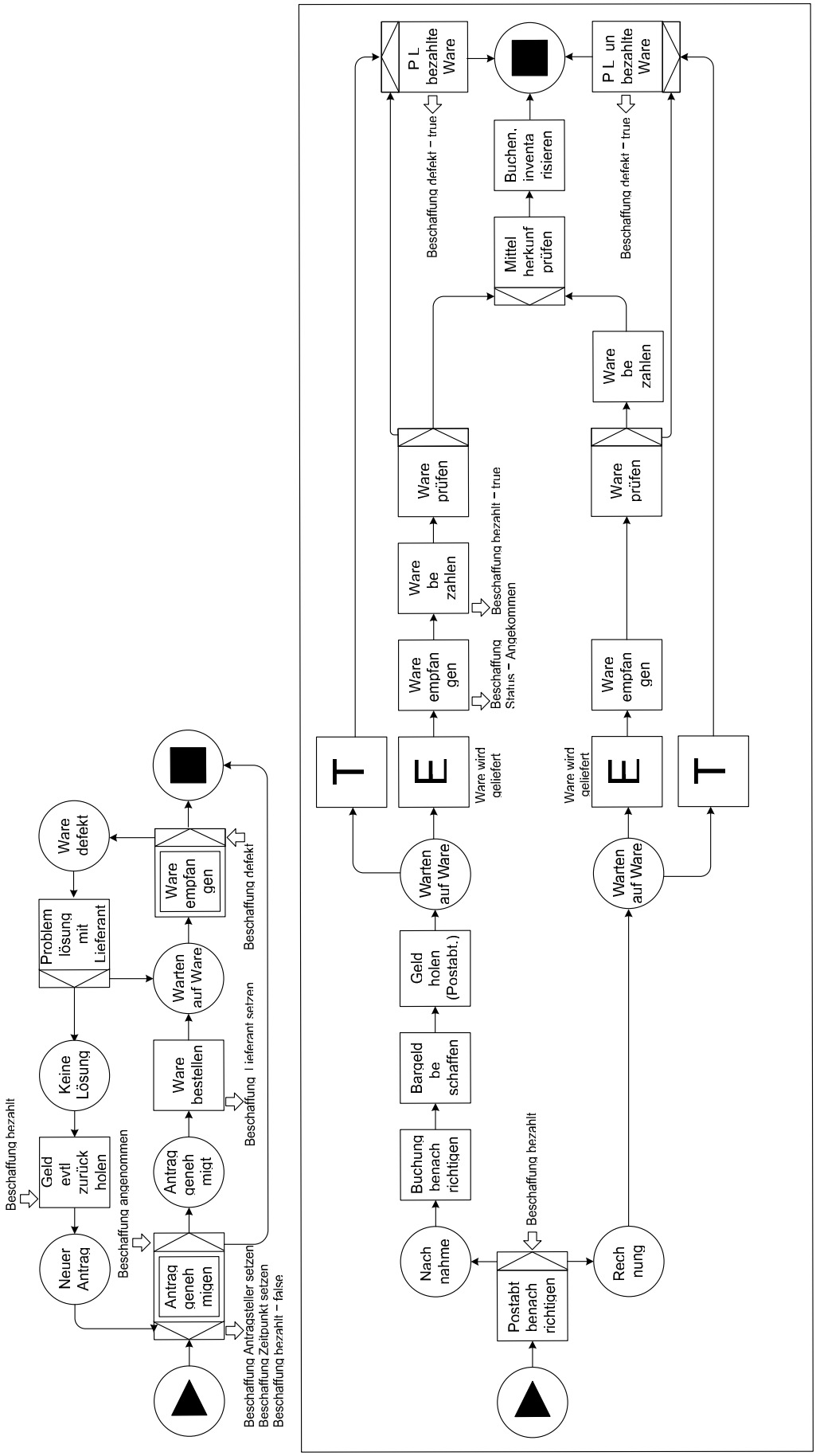


Abb. 15: Alternativer Beschaffungsprozess, erweitert um Operationszustände

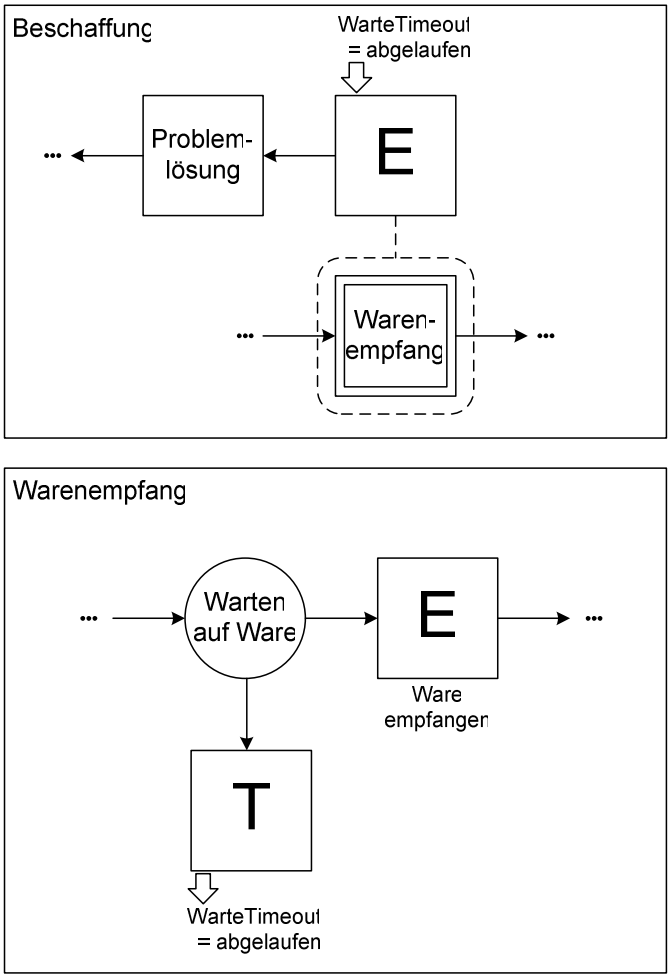


Abb. 16: Um Operationszustände erweiterte Events

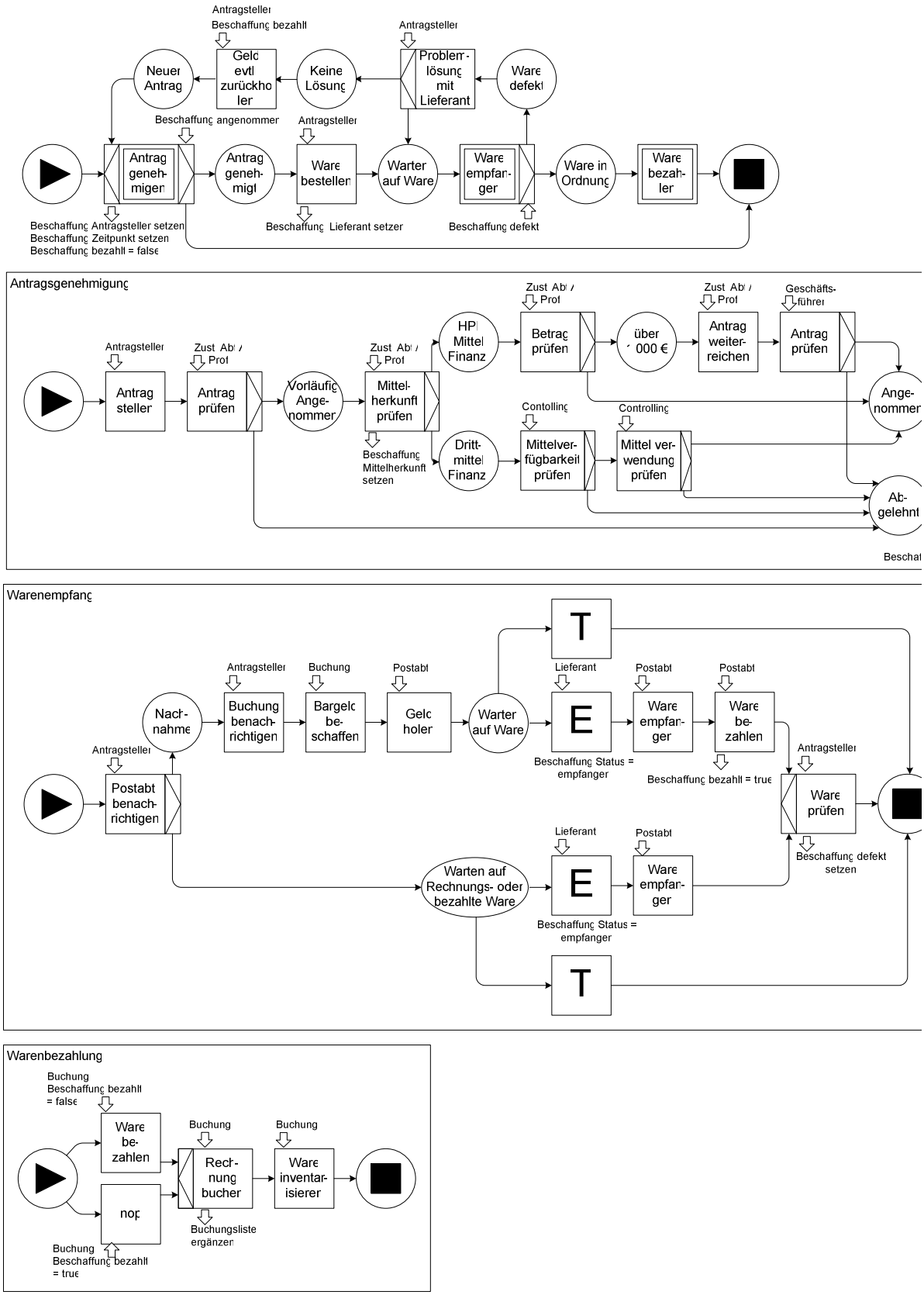


Abb. 17: Beschaffungsprozess, erweitert um Operationszustände und Rollenbezeichnungen

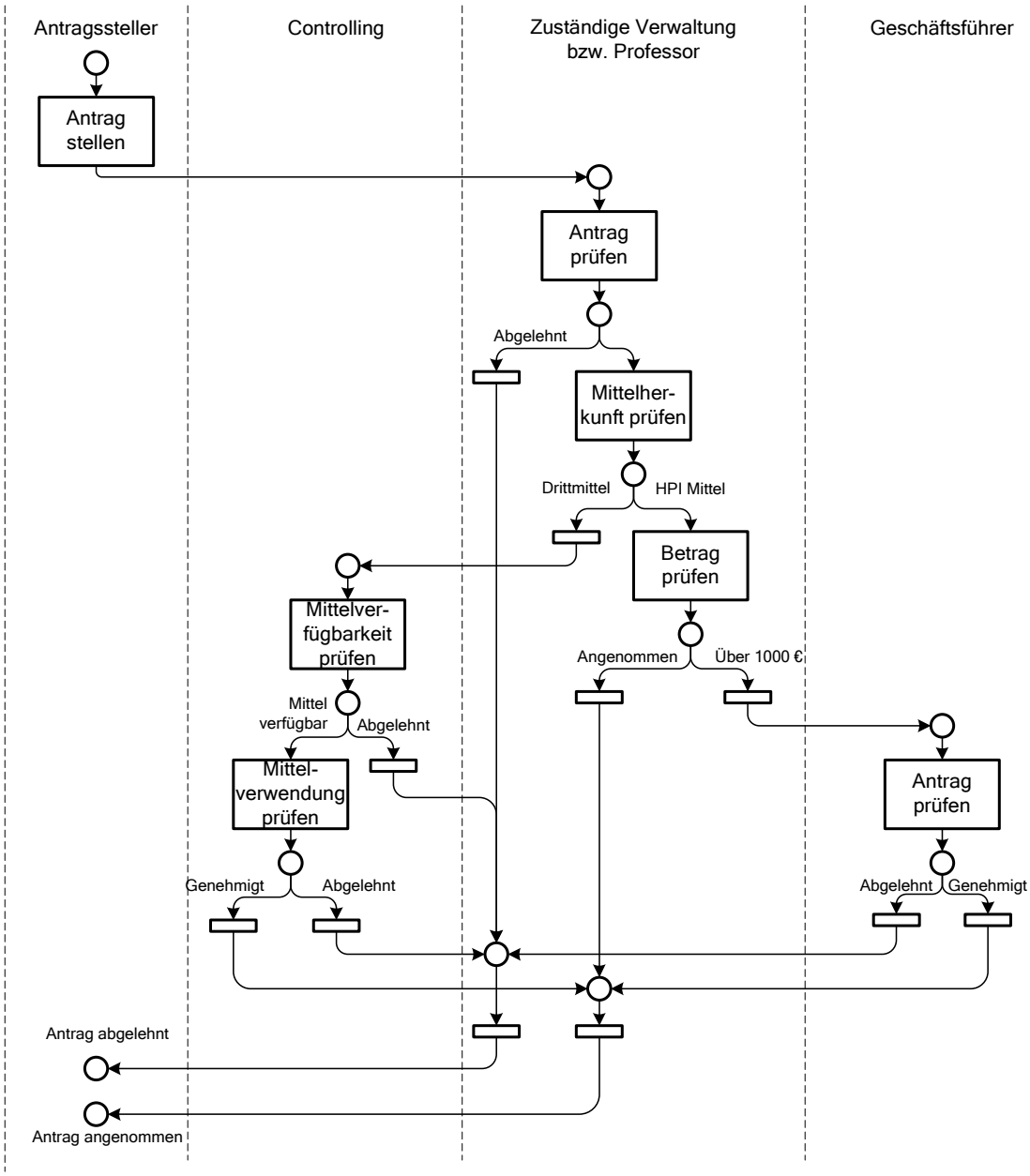


Abb. 18: Petrinetz zur Antragsgenehmigung

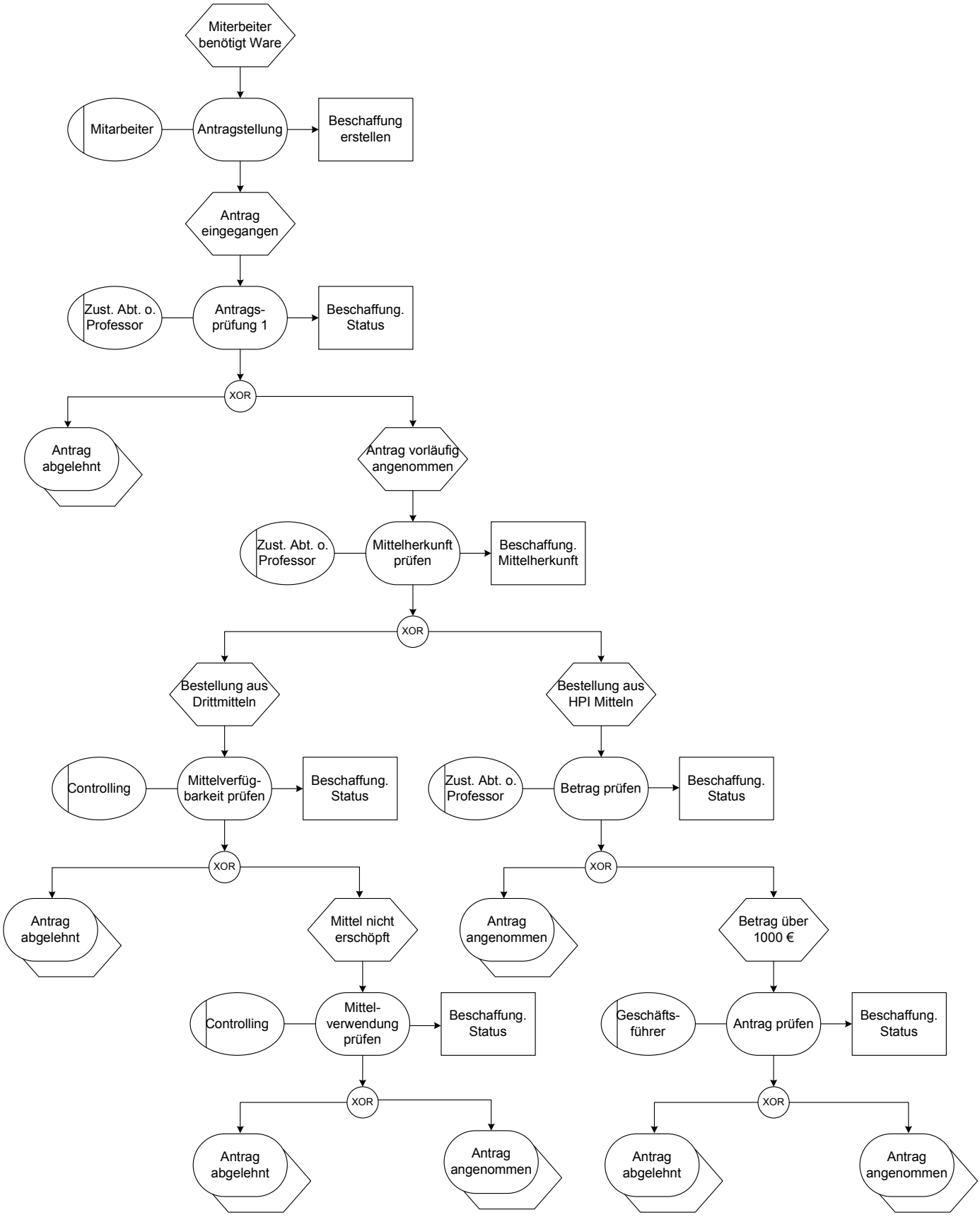


Abb. 19: Antragsgenehmigung in der ARIS Steuerungssicht

Modellierung der Studien- und Prüfungsordnungen

Kay Hammerl

Lehrstuhl Business Process Technology
Hasso-Plattner-Institut für Softwaresystemtechnik
kay.hammerl@student.hpi.uni-potsdam.de

Abstract. Diese Ausarbeitung, welche im Rahmen des Seminars „Prozessmodellierung“ entstand, beschäftigt sich mit der Modellierung der Studien- und Prüfungsordnungen des Hasso-Plattner-Instituts, welche in zwei unterschiedlichen Fassungen vorlagen. Als Hauptmodellierungssprache findet die noch recht junge Workflowsprache YAWL Verwendung. Neben den Ergebnissen, an welchen die Eignung von YAWL zur Modellierung festgestellt werden soll, wird auch die Herangehensweise angesprochen. Es werden weiterhin die Unterschiede der beiden Ordnungen aufgedeckt und anhand dieser die Möglichkeiten der Variabilitätsmodellierung in YAWL untersucht. Hierzu werden mehrere Vorschläge vorgestellt. Abschließend wird eine (subjektive) Bewertung zu YAWL gegeben.

1 Einleitung

Die Studien- in Verbindung mit der Prüfungsordnung des Hasso-Plattner-Instituts für Softwaresystemtechnik (HPI) regelt den allgemeinen Aufbau sowie die Gestaltung des dort angebotenen Bachelor- und Masterstudiengangs. Die beiden Ordnungen existieren in zwei unterschiedlichen Versionen: In der aktuell gültigen Fassung, welche im Jahr 2001 verabschiedet wurde, und in einer Neufassung vom April 2004, die die Alte in naher Zukunft ersetzen soll.

Die Aufgabe in diesem Seminar bestand nun darin, die in den Studien- und Prüfungsordnungen für Studenten der Softwaresystemtechnik beschriebenen Prozesse zu analysieren und in Diagrammform darzustellen. Weiterhin sollten statische Modelle der in den Prozessen verarbeiteten Daten erstellt, sowie die am Prozess beteiligten Rollen identifiziert werden. Abschließend sollten die beiden unterschiedlichen Fassungen auf Gemeinsamkeiten und Variabilitäten untersucht werden.

Das Hauptaugenmerk dieser Aufgabe lag auf den dynamischen Aspekten und deren Modellierung mit der relativ jungen Workflowsprache YAWL (Yet Another Workflow Language) [4].

Nach einer Beschreibung der Vorgehensweise, findet im Hauptteil die Präsentation der Ergebnisse statt, wobei dort an ausgewählten Punkten Probleme der Modellierungssprache diskutiert werden. Basierend auf der Variabilitätsuntersuchung der beiden Fassungen der Studien- beziehungsweise Prüfungsordnung wird am Ende analysiert, wie es grundlegend möglich ist, Variabilitäten in YAWL darzustellen.

Aufgrund der Absenz von wissenschaftlichen Auseinandersetzungen mit YAWL, werden dort eigene Vorschläge demonstriert.

2 Vorgehensweise

Nach dem Erhalt der Aufgabenstellung war zunächst eine intensive Beschäftigung mit der inhaltlichen Thematik, sprich den Studien- und Prüfungsordnungen des HPI, nötig. Die Informationserhebungsphase bestand somit vornehmlich aus dem Lesen der verschiedenen Ordnungen. Dazu sei gesagt, dass im Folgenden nur noch der Ausdruck „Studienordnung“ verwendet wird, welcher hier die Vereinigung der Studien- mit der Prüfungsordnung umschreibt. Der Grund dieser Verallgemeinerung liegt schlichtweg in der Tatsache, dass eine getrennte Analyse sich als nicht sinnvoll erwies, da ein Studiengang mit allen seinen Facetten nur durch die gemeinschaftliche Betrachtung beider Ordnungen beschreibbar ist. Die einzigen Unterscheidungen die im Weiteren getroffen werden, sind die in „Studienordnung 2001“ und „Studienordnung 2004“.

Während des Lesens der Studienordnungen fand sogleich eine Identifizierung der dynamischen sowie der statischen Aspekte statt. Das Resultat dieses Schrittes erfasst Tabelle 1.

Table 1. Statische und Dynamische Aspekte der HPI-Studienordnung

Statische Aspekte	Dynamische Aspekte
1) Aufbau Studienausschuss 2) Relation zwischen: Leistungspunkten, Lehrveranstaltungen, Semesterwochenstunden, Themenkomplex und Benotungsinformation 3) Belegungs- / Leistungspunktekonto 4) Antrag auf Graduierung (Bachelor/Master, 2001/2004) 5) Aufbau Zulassungskommission (Master) 6) Notenskala	1) Studienfachberatung 2) Anerkennung von Leistungen 3) Leistungserfassungsprozess 4) Belegung von Lehrveranstaltungen 5) Belegungen zurückziehen 6) Graduierung 7) Zulassung zum Masterstudium 8) Ablauf der verschiedenen Abschlussarbeiten (Bachelorprojekt etc.)

Im nächsten Stadium wurde versucht, sämtliche auftretenden Rollen zu identifizieren, um anschließend ihr Auftreten in den einzelnen dynamischen Aspekten zu untersuchen. Dieses Vorgehen mündete in einem mittelgroßen UML Use-Cases-Diagramm (siehe Anhang, Abbildung 5). An dieser Stelle wurden bereits erste Gesichtspunkte von der weiteren Betrachtung ausgeschlossen. Konkret handelt es sich hierbei um die Punkte 1 (Studienfachberatung) und 7 (Zulassung zum Masterstudium) der dynamischen Aspekte in Tabelle 1. Streichungsgrund war hier hauptsächlich die Tatsache, dass der Plan bestand, den Ablauf eines HPI-Studiums zu modellieren, aber die beiden aufgezählten Punkte Aktivitäten darstellen, welche *vor* dem Beginn eines

Studiums stattfinden. Ferner sei angemerkt, dass die Abläufe in diesen beiden Fällen nicht sonderlich komplex sind, und ein Fehlen dieser keinen schmerzlichen Verlust darstellt.

Die nächste unternommene Maßnahme war das Modellieren der statischen Aspekte, was unter Zuhilfenahme von Entity-Relationship-Diagrammen (ER-Diagrammen) geschah. Hierbei wurde auf die grafische Aufbereitung der Punkte 5 (Aufbau Zulassungskommission) und 6 (Notenskala) verzichtet. Das Auslassen beruhte bei Ersterem auf der oben geschilderten Entscheidung, die Zulassung zum Masterstudium nicht zu betrachten, und bei Letzterem auf Trivialität.

Den letzten und umfassendsten Schritt bildete freilich das Erstellen der dynamischen Diagramme mit YAWL. Dort wurden die Abläufe sämtlicher Use Cases dargestellt und deren Relationen untereinander in einem Gesamtprozess zusammengefasst.

3 Die HPI-Studienordnung

In diesem Abschnitt wird der Aufbau und Ablauf eines HPI-Studiums anhand der angefertigten Diagramme vorgestellt. Nebenläufig dazu wird auf Modellierungsprobleme sämtlicher Art eingegangen.

Aufgrund der Tatsache, dass ein weiterer Punkt dieser Ausarbeitung sich mit den Unterschieden zwischen den Studienordnungen beschäftigt, findet an dieser Stelle eine Darstellung statt, welche für beide Ordnungen und sogar für beide Studiengänge (Bachelor, Master) gültig ist.

Betrachten wir zum Einstieg das bereits im vorherigen Kapitel erwähnte Use-Cases-Diagramm (siehe Anhang, Abbildung 5). Es bedarf wahrscheinlich wenig erläuternder Worte. Man erkennt, dass nur vier Rollen an den Prozessen beteiligt sind. Namentlich genannt handelt es sich hierbei um den Studenten selbst, die Verwaltung, welche bei fast jedem Vorgang beteiligt ist, den Studienausschuss, welcher vor allem bei bestimmenden Aktivitäten involviert ist und den Professor, wobei diese Bezeichnung sämtliche Anbietungsberechtigte inkludiert. Was nicht auf Anhieb klar sein sollte, ist die Unterscheidung von „Belegung zurückziehen“ in einen regulären und einen irregulären Fall. Kurz zusammengefasst lässt sich dies so erklären, dass es bis zu einem gewissen Zeitpunkt (eine Woche vor Beginn des Leistungserfassungsprozesses) möglich ist, eine Belegung ohne Angabe von Gründen zurückzunehmen (= regulärer Fall). Nach Überschreitung dieses Zeitpunktes ist eine Rücknahme nur noch in begründeten Ausnahmefällen möglich und an einen an den Studienausschuss gestellten Antrag gebunden (= irregulärer Fall). Um den Begriff „Studienausschuss“ zu konkretisieren, findet der Leser im Anhang (Abbildung 6) ein ER-Diagramm, welches den Aufbau dieses Ausschusses zeigt.

Abbildung 7 im Anhang stellt das wichtigste Datum dar, auf dem fast alle Prozesse der Studienordnung arbeiten: das Punktekonto. Jeder HPI-Student besitzt genau ein Solches. Ein Punktekonto besteht aus jeweils einem Belegungspunktekonto und einem Leistungspunktekonto. Auf Ersterem befinden sich Belegungspunkte und auf Letzterem dementsprechend Leistungspunkte. Belegungspunkte werden im Laufe des Studiums durch erfolgreiches Besuchen von Lehrveranstaltungen in Leistungspunkte

umgewandelt. Daraus ergibt sich die Bedingung, dass die Summe aus Belegungs- und Leistungspunkten immer kleiner oder gleich der anfänglich vergebenen Belegungspunkte ist. Der „kleiner als“-Fall kommt dann zustande, wenn eine oder mehrere Veranstaltungen ohne Erfolg besucht wurden. Die für diese Veranstaltungen eingesetzten Belegungspunkte stellen somit verlorene Punkte dar. Die zu Beginn des Studiums vergebene Belegungspunkteanzahl ist bestimmt durch den Studiengang, beziehungsweise die entsprechende Ordnung. Die jeweils zutreffenden Werte sind in der Abbildung entsprechend gekennzeichnet.

In Abbildung 8 (siehe Anhang) wird der Leistungspunkt an sich näher betrachtet. Man sieht, dass er von einer, zu einem gewissen Themenkomplex gehörenden, Lehrveranstaltung vergeben wird und eine Benotungsinformation sowie die Auskunft über den Themenkomplex, in dem er erbracht wurde, enthält.

Kommen wir nun zu der Betrachtung des Top-Level-Prozesses (siehe Anhang, Abbildung 9), welcher ein Studium am Hasso-Plattner-Institut beschreibt. Man erkennt, dass nach Beginn des Studiums die Möglichkeit besteht mehrere Lehrveranstaltungen zu belegen und zu besuchen. Dies kann mehrere Semester lang wiederholt werden. Nach Abschließen der Abschlussarbeit (an dieser Stelle noch nicht näher spezifiziert) ist es möglich einen Antrag auf Graduierung zu stellen und, sollte dieser vollständig sein, auch zu graduieren und damit das Studium abzuschließen. Im unteren Viertel ist der negative Fall dargestellt, nämlich der, dass aus irgendeinem Grund die Graduierung als verwirkt angesehen wird und somit eine Exmatrikulation ansteht. Bei Ausführen dieser werden sämtliche Marken aus dem gekennzeichneten Bereich entfernt, um bei der nachfolgenden Beendigung des Gesamtprozesses mit einer „sauberen Terminierung“ (keine verbleibenden Marken im Netz) abzuschließen. Der mit dem großen „E“ gekennzeichnete Task[1] unten links findet sich so nicht in einer der Studienordnungen wieder, entspricht aber dem regulären Tatbestand. Jedes Semester wird nämlich eine Überprüfung durchgeführt, welche Studenten nicht mehr in der Lage sind, ausgehend von ihren noch vorhandenen Belegungspunkten, den Abschluss zu erreichen. Davon betroffene Studenten bekommen auf diesem Wege mitgeteilt, dass ihre Tage am HPI gezählt sind.

Nun entdeckt man in dieser Sicht auf den Prozess nur zwei Use-Cases wieder, nämlich „Abschlussarbeit durchführen“ und „Graduieren“. Die restlichen vier finden sich eine Ebene tiefer, in der detaillierten Betrachtung des zusammengesetzten Task (Composite Task) „Lehrveranstaltungen belegen und besuchen“ (siehe Anhang, Abbildung 10). In diesem Diagramm wird der Ablauf von der Belegung einer Veranstaltung bis zur Leistungspunktevergabe bei erfolgreicher Teilnahme beschrieben. Zu Beginn findet eine Fallunterscheidung statt, ob der Student sich nur extern erworbene Leistungen anerkennen lassen will, oder ob er auch am HPI etwas belegen möchte. In letzterem Fall besteht freilich später immer noch die Möglichkeit, sich externe Leistungen anrechnen zu lassen. Der Ablauf einer Belegung ist in Abbildung 11 (siehe Anhang) illustriert und nicht weiter spannend. Es wird dort nur überprüft, ob der betreffende Student noch in der Lage ist, den Abschluss zu erlangen und ob er über genügend Belegungspunkte verfügt, um die gewünschte Veranstaltung

[1] Das „E” steht hierbei für „Event”. Die Bedeutung ist die, dass ein so gekennzeichneter Task von einem externen Ereignis abhängt.

zu belegen. Nach einer Belegung, bis zu Beginn des Leistungserfassungsprozesses, hat der Student die Möglichkeit, seine Belegung auf regulärem Wege zurückzunehmen (siehe Anhang, Abbildung 12) und seine „bezahlten" Belegungspunkte wiederzuerhalten. Das Zeitintervall [Belegungszeitpunkt, Leistungserfassungsprozessbeginn] wird durch einen so genannten „timed Task" dargestellt (der Task mit dem großen „T"). Dieser wird direkt nach der Belegung aktiviert und erst nach Ablauf der Zeit beendet. Danach wird der Leistungserfassungsprozess gestartet und alle Marken aus dem gekennzeichneten Bereich (gestrichelte Linie) entfernt, so dass keine reguläre Rücknahme der Belegung mehr möglich ist. Andersherum würde nach erfolgreicher Zurücknahme der Timer abgebrochen, da in diesem Fall keine Teilnahme an der Leistungserfassung stattfinden soll.

Wie man leicht sieht, ist es zu jedem Zeitpunkt möglich, sich extern erbrachte Leistungen anerkennen zu lassen. Es wäre folglich kein Problem gewesen, diesen Prozess direkt auf oberster Ebene im Top-Level-Prozess (siehe Anhang, Abbildung 9) nebenläufig zu „LV belegen und besuchen" einzufügen. Der Grund für die Einbindung in den Sub-Prozess lag schlichtweg darin, dass bevor man sich Leistungen anrechnen lassen kann, diese erstmal erbracht werden müssen. Sprich, der Student muss extern Veranstaltungen belegt und besucht haben. Deswegen schien die Platzierung in dem Subprozess semantisch nahe liegend.

Auf den exakten Vorgang des Anerkennens von Leistungen, welcher mit einem Antrag an den Studienausschuss beginnt, werde ich nicht weiter eingehen. Der Leser findet jedoch eine Vorstellung dieses Prozesses im Anhang in den Abbildungen 13, 14 und 15, welche selbsterklärend sein sollten.

Wenden wir den Blick zurück auf Abbildung 10 und dort auf den OR-Join vor dem Endzustand. Das Konstrukt realisiert das Workflowpattern „Synchronizing Merge" [6]. An dieser Stelle wird auf sämtliche mögliche eingehende Marken gewartet. Erst nach Feststellung, dass keine Marke mehr kommen kann, wird der OR-Join ausgeführt und der Prozess beendet. Dies vereinfacht die Darstellung erheblich und erhält die Übersichtlichkeit des Diagramms, da eine Ersetzung des OR-Joins nur durch *mehrere* andere Routing-Konstrukte möglich ist (Eine mögliche Lösung würde hier zum Beispiel das Einfügen eines AND-Splits, eines XOR-Joins, eines AND-Joins sowie einer zusätzlichen Stelle erfordern.).

Bevor wir uns nun jedoch in der Hierarchie wieder nach oben bewegen, möchte ich den schon erwähnten Leistungserfassungsprozess näher vorstellen. Dieser findet sich detailliert in Abbildung 16 (siehe Anhang).

Ein Leistungserfassungsprozess kann aus mehreren Leistungserfassungsschritten (z.B. aus mehreren Klausuren) bestehen. Betrachten wir hier beispielhaft eine Veranstaltung mit zwei schriftlichen Klausuren, welche von einem Studenten namens Anton besucht wird[2]. Nach Eintreten in den Prozess, entscheidet er sich an dem ersten Leistungserfassungsschritt (= an der ersten Klausur) regulär teilzunehmen. Während die Klausur läuft, überkommt ihn jedoch plötzliches Übelgefühl, und er bricht die Klausur vorzeitig ab. Im Prozess befindet er sich nun an gleicher Stelle, als wenn er gar nicht erst zum Test hingegangen wäre. Anton hat nun eine gewisse Zeit (5 Werktage) zur Verfügung, bei der zuständigen Lehrkraft Gründe für sein Versäumnis

[2] Es sollte an dieser Stelle der Ablauf im Diagramm mitverfolgt werden!

einzureichen, was er auch macht. Nun liegt es an der Lehrkraft, ob sie den Grund akzeptiert und einen Termin für eine Ersatzprüfung anberaumt oder Anton eine 5 (= nicht bestanden) als Ergebnis für diese Klausur einträgt. Wir nehmen an, dass Anton ein glaubhaftes ärztliches Attest vorlegt und eine neue Chance erhält. Zu dieser Prüfung erscheint er pünktlich und bringt sie ordnungsgemäß zu Ende. Im Anschluss bewertet die Lehrkraft die Arbeit. Weiterhin gibt sie, da es sich um einen schriftlichen Leistungserfassungsschritt handelte, Anton die Möglichkeit, Einsicht in die korrigierte Klausur zu nehmen. Anton zeigt daran jedoch kein Interesse, und so wird das Ergebnis nach Ablauf einer gewissen Zeit eingetragen. Nun ist Anton mit dem Ergebnis nicht wirklich zufrieden und versucht die Belegung (irregulär) zurückzuziehen (siehe Abbildung 17 im Anhang). Der Studienausschuss lehnt seinen Antrag jedoch, mangels plausibler Gründe, ab, so dass Anton als Teilnehmer dieses Leistungserfassungsprozesses erhalten bleibt. Einige Wochen später steht nun die letzte Klausur an. Anton entscheidet sich daran teilzunehmen und das Beste zu versuchen. Aufgrund mangelnder Vorbereitung, versucht er allerdings seinem Glück nachzuhelfen, indem er einen Spickzettel mitnimmt (im Diagramm: „LES stören oder täuschen“). Er hofft, dass sein Schummeln nicht auffällt und er die Klausur ordnungsgemäß zu Ende bringen darf. Doch dieses Glück wird ihm nicht zu Teil: Anton wird erwischt, seine Arbeit mit „nicht ausreichend“ bewertet und das Ergebnis eingetragen. Da nun alle Leistungserfassungsschritte durchgeführt sind, wird abschließend die Gesamtnote berechnet und vergeben.

Nach dem beispielhaften Abwickeln dieses Prozesses, möchte ich die hier verwendeten „Cancellation“-Konstrukte jedoch noch einmal näher betrachten. Unten links im Diagramm erkennt man, dass sobald ein Grund eingereicht wurde der Timer beendet wird. Wir beachten, dass es sich hier („Grund einreichen“) um einen Vorgang der Länge Null handelt, welcher zu einem diskreten Zeitpunkt eintritt (oder auch nicht). Die Semantik ist also direkt ersichtlich: Entweder läuft der Timer ab oder ein Grund wird eingereicht. Oben rechts, dort wo ein Timer die Möglichkeit der Einsichtnahme zeitlich begrenzt, stellt sich auf den ersten Blick die Sache genauso dar, nämlich dass sobald die Einsicht durchgeführt wurde, der Timer beendet wird. Nun ist „Einsicht nehmen“ allerdings ein kontinuierlicher Vorgang. Das führt zu dem Problem, dass der Task „Einsicht nehmen“ am Laufen sein kann, wenn der Timer abläuft. Der Timer würde somit den laufenden Task beenden und weitermachen, als ob keine Einsicht stattgefunden hätte. Das wäre in diesem Kontext vielleicht kein Problem. Stellen wir uns jedoch einen Unternehmensfall vor, in dem der Task statt „Einsicht nehmen“ „Bezahle A-Z“ heißt, der zuständige Mitarbeiter schon das Geld an die Firmen A-F überwiesen hat, und dann der Timer die Bezahlung abbricht und so weitermacht, als ob überhaupt niemand bezahlt worden wäre. An dieser Stelle wäre es folglich erforderlich, einen Rollback durchzuführen und das überwiesene Geld wieder zurückzuholen. Der einfachste Weg dieses Problem zu umschiffen ist jedoch der, dass man vor dem kontinuierlichen Task einen zusätzlichen, keine Zeit beanspruchenden Task einfügt, dessen einzige Aufgabe es ist, den Timer zu beenden. Die Anwendung dieser Methode auf unseren Kontext zeigt nachstehende Abbildung.

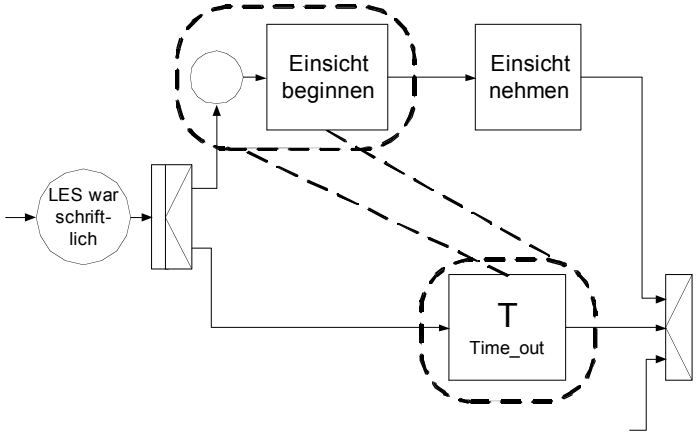


Fig. 1. Vorschalten eines keine Zeit beanspruchenden Tasks.

Nun ist es nicht mehr möglich, dass eine laufende Einsichtnahme abgebrochen wird.

Es sei an dieser Stelle darauf hingewiesen, dass auch ein Timer existieren müsste, welcher die rechtzeitige Abgabe eines schriftlichen Leistungserfassungsschrittes überwacht. Das Auslassen der Modellierung eines Solchen geschah aus dem simplen Grund, die Übersichtlichkeit erhalten zu wollen.

Springen wir nun wieder in den Top-Level-Prozess (Abbildung 9 im Anhang). Wir sehen, dass sobald die Abschlussarbeit erfolgreich durchgeführt wurde es möglich ist, einen Antrag auf Graduierung zu stellen. Sollte dieser vollständig sein, dann gelangen wir zu dem Punkt, auf den jeder Student hinarbeitet: die Graduierung. Der abschließende Prozess ist in Darstellung 18 (siehe Anhang) modelliert. Es handelt sich um keinen besonders komplizierten Vorgang, weshalb hier auch nur auf einen einzigen Aspekt ausführlicher eingegangen wird. Man erkennt, dass nach dem Anfertigen der Urkunde, diese sowohl vom Vorsitzenden des Studienausschusses als auch vom Dekan der Mathematisch-Naturwissenschaftlichen Fakultät unterschrieben werden muss. Dies kann freilich in beliebiger Reihenfolge geschehen, allerdings nicht gleichzeitig. Um diesen Sachverhalt auszudrücken, bietet sich das Workflowpattern „Interleaved Parallel Routing“ [6] an. Die Umsetzung des Patterns mit YAWL ist denkbar einfach und dem Diagramm entnehmbar. Was jedoch bitter aufstößt ist die Tatsache, dass die Semantik meiner Meinung nach nicht direkt herauslesbar ist, und man sich vorstellen kann, dass das Ganze leicht unübersichtlich wird. In diesem Fall sind nur zwei Tasks betroffen, also die kleinste Menge auf der dieses Pattern überhaupt definiert ist, aber es werden schon drei AND-Splits, drei AND-Joins sowie zehn Pfeile benötigt. Für jeden hinzukommenden Task müssten weitere vier Pfeile plus jeweils ein AND-Split und –Join ergänzt werden.

4 Variabilitäten

Dieses Kapitel beschäftigt sich im ersten Teil mit den Varianten, die sich aus den verschiedenen Studienordnungen ergeben. Im Zweiten Teil werden allgemeine Überlegungen zur Darstellung von Variabilitäten in YAWL behandelt.

4.1 Unterschiede der Studienordnungen 2001 / 2004

Nachdem im vorangegangenen Kapitel der allgemeingültige Ablauf eines Studiums am HPI, welcher sowohl für die beiden Studiengänge Bachelor und Master als auch die beiden Fassungen der Studienordnung von 2001 und 2004 gültig ist, besprochen wurde, werde ich in diesem Abschnitt auf die Unterschiede zwischen den beiden Ordnungen eingehen und dazu selbstverständlich die studiengangspezifischen Prozesse betrachten.

Die Leser, welche herausragende Abweichungen erwarten, muss ich leider schon im Vorfeld enttäuschen. Die Änderungen beziehen sich lediglich auf die Art beziehungsweise den Ablauf der Abschlussarbeit, auf die (benotet) einzubringenden Fächer sowie die zu Beginn verfügbaren Belegungspunkte. Richten wir unseren Blick erneut auf den Top-Level-Prozess (Anhang, Abbildung 9). Wir können nun drei Stellen ausmachen, an denen etwas geändert werden muss, um diese Darstellung in einen konkreten Studiengang zu wandeln:

1. Beim Eintritt in den Prozess („Studium beginnen“) wird die entsprechende Anzahl an Belegungspunkten vergeben.
2. Der zusammengesetzte Task „Abschlussarbeit durchführen“ wird durch einen konkreten „Composite Task“ ersetzt (z.B. Bachelorprojekt durchführen).
3. Je nach Studiengang und Fassung enthält der Antrag auf Graduierung andere Pflichtinhalte. Die Überprüfung auf Vollständigkeit eines Antrags beruht somit jeweils auf einem Vergleich mit andern Daten.

Auf den ersten Punkt wurde im letzten Kapitel schon hingewiesen, ich verweise an dieser Stelle erneut auf das ER-Diagramm in Abbildung 7 (Anhang), aus welchem sich die entsprechende Punktezahl herauslesen lässt. Auf die Aspekte zwei und drei werde ich jedoch genauer eingehen.

Sehen wir uns zuerst den Bachelorstudiengang an, so wie er in der Studienordnung von 2001 festgelegt ist. Wenn wir nun in den „Composite Task“ „Abschlussarbeit durchführen“ hereinschauen, würden wir den in Abbildung 19 (siehe Anhang) modellierten Prozess vorfinden. Hierin wird zuerst auf ein bestandenes Vorbereitungsseminar geprüft. Das Vorbereitungsseminar selbst befindet sich nicht in diesem Subprozess, da es sich bei diesem prinzipiell um eine ganz normale Lehrveranstaltung handelt (Die Belegung ist an Abgabe von Belegungspunkten gebunden, Abschlussarbeiten wie das Bachelorprojekt sind jedoch „kostenlos“). Kann der Student nun solch ein erfolgreich besuchtes Seminar vorweisen, darf er an einem Bachelorprojekt teilnehmen. Scheitert er im ersten Durchlauf, darf er einen erneuten Versuch unternehmen. Misslingt auch dieser, wird die Graduierung als verwirkt angesehen.

Nach erfolgreichem Abschluss des Bachelorprojekts wird der Student sicherlich einen Antrag auf Graduierung stellen. Der Aufbau desselbigen ist in Abbildung 20

(siehe Anhang) dargestellt. Man sieht dort, dass ein bestandenes Bachelorprojekt vorliegen muss, sowie eine benotete und eine unbenotete Liste. Diese Listen enthalten jeweils eine bestimmte Anzahl an Leistungspunkten, welche wiederum zu einem bestimmten Themenkomplex gehören. Die Minimum-Kardinalitätsangaben an letzterer Relation besagen, wie viele Punkte aus einem Themenkomplex mindestens eingebracht werden müssen (benotet oder unbenotet). Zum Beispiel müssen mindestens 24 Leistungspunkte aus dem Bereich Softwarekonstruktion auf den beiden Listen erscheinen.

Wechseln wir nun zur neueren Studienordnung und schauen, wie dort der Bachelorstudiengang aufgebaut ist. Der spezifische Ablauf der Abschlussarbeit ist in Abbildung 21 (siehe Anhang) gezeigt. Im Unterschied zur alten Ordnung existiert jedoch kein Vorbereitungsseminar mehr, sondern das Projekt wird direkt begonnen. Nach erfolgreichem Abschluss desselbigen, muss der Student eine schriftliche Bachelorarbeit anfertigen, für welche ihm eine Bearbeitungszeit von drei Monaten zur Verfügung steht. Die Arbeit ist benotet und wird für die Ermittlung der Abschlussnote äquivalent zu 15 benoteten Leistungspunkten gewichtet. Sowohl für das Projekt als auch für die Arbeit stehen dem Studierenden zwei Versuche zu. Grafisch hier nicht herauslesbar ist, dass das Bachelorprojekt, welches hier gezeigt wird, nicht identisch ist mit dem der Studienordnung von 2001. In der alten Ordnung handelt es sich nämlich um ein Vollzeitprojekt, wohingegen dieses einen reduzierten Umfang von 12 Semesterwochenstunden besitzt.

Schauen wir uns nun den Antrag auf Graduierung für den Fall Bachelor 2004 an (siehe Anhang, Abbildung 24). Das ER-Diagramm ist deutlich komplizierter und enthält auch mehr Bedingungen in schriftlicher Form. Der Grund hierfür ist denkbar einfach: Die neue Studienordnung weist einen restriktiveren Charakter auf, was die Wahl der einzubringenden Fächer angeht. Wir sehen zum Beispiel, dass die Summe der Leistungspunkte aller Kernfächer 108 beträgt. Diese müssen allesamt eingebracht werden. 78 Leistungspunkte davon müssen auf der benoteten Liste erscheinen, wovon bei 48 Punkten sogar fest vorgeschrieben ist, aus welchen Veranstaltungen sie hervorgehen müssen. Der großzügigen Wahlfreiheit aus der alten Ordnung wurde hiermit folglich ein Riegel vorgeschoben.

Der Masterstudiengang am HPI bekam keine wirklichen Veränderungen bei der Durchführung der Abschlussarbeit verordnet. Es fand lediglich eine Beschränkung der Bearbeitungszeit (2001: 12 Monate; 2004: 6 Monate) statt, welche sich mit der Verkürzung der Regelstudienzeit erklärt. Außerdem ist in der alten Ordnung noch von einem Masterprojekt und einer Masterarbeit die Rede[3], in der neuen Fassung findet dagegen nur noch letzterer Begriff Verwendung. Der reine Ablauf ist jedoch derselbe und in Abbildung 22 (siehe Anhang) vorzufinden. An dieser Stelle sei auf zwei Besonderheiten im Prozess der Masterabschlussarbeiten hingewiesen. Zum einen ist es dem Studenten einmal möglich im ersten Drittel der Bearbeitungszeit ein Thema zurückzugeben und ein Neues zu erhalten, zum anderen läuft die Bewertung nach einem bestimmten Prozedere ab, welches in Abbildung 23 (siehe Anhang) dargestellt wurde. Hierbei wird die entsprechende Arbeit von zwei Gutachtern benotet. Beträgt die Differenz in der Benotung 2,0 oder mehr oder ist ein Gutachter der Meinung, dass es sich um eine nicht ausreichende Arbeit handelt, so *kann* vom Studienausschuss ein

[3] Auf Nachfrage wurde mir erklärt, dass beide Begriffe synonym verwendet wurden.

dritter Gutacher bestellt werden. Liegen am Ende mindestens zwei Bewertungen mit Note vor, so ergibt sich die Note der Arbeit aus dem arithmetischen Mittel der Einzelnoten.

Was der Antrag auf Graduierung zur Erlangung des Mastergrades enthalten muss, ist den Abbildungen 25 (Studienordnung 2001) und 26 (Studienordnung 2004) zu entnehmen. Die ER-Diagramme sind ähnlich zu denen der Bachelorstudiengänge aufgebaut und sollten deshalb keiner weiteren Erklärung bedürfen.

Zusammenfassend lässt sich also sagen, dass der allgemeine Teil, welcher für alle Studiengänge und auch für beide Fassungen identisch ist, überwiegt. Unterschiede, welche den Prozessablauf betreffen, spiegeln sich allein in der Durchführung der verschieden Abschlussarbeiten wider. Die restlichen Unterschiede sind rein statischer Natur und betreffen die eingangs verteilten Belegungspunkte sowie den Inhalt des Antrags auf Graduierung.

4.2 Darstellung von Variabilitäten in YAWL

In diesem Abschnitt werde ich eigene Überlegungen zur geeigneten Darstellung von Variabilitäten in der Workflowsprache YAWL vorstellen.

Eine Möglichkeit haben wir bereits kennen gelernt: Man führt einen abstrakten „Composite Task“ ein und ersetzt diesen, in der jeweiligen Version, durch einen konkreten zusammengesetzten Task, welcher den individuellen Ablauf beinhaltet. Dieses Verfahren ist an die objektorientierte Programmierung angelehnt und wurde von mir verwendet, um die unterschiedlichen Abschlussarbeiten in den allgemeinen Top-Level Prozess zu integrieren.

Diese Methode funktioniert sehr gut und ist bei größeren Änderungen sicherlich der ratsamste Weg, um Variabilität darzustellen. Nun kam jedoch die Frage auf, wie mit kleineren Änderungen zu verfahren ist. Also mit Abweichungen, die so minimal sind, dass sie ein Auswechseln eines Subprozesses nicht rechtfertigen. Nehmen wir einfach einmal an, dass es eine zusätzliche Änderung in der Studienordnung von 2004 zu der von 2001 gäbe, nämlich, dass ein Student nach dem Belegen sofort ein Entgelt für die gewählte Lehrveranstaltung leisten muss. In diesem Fall wäre es stark übertrieben einen „Belegen 2001“- sowie einen „Belegen 2004“-Subprozess zu erzeugen, da der Unterschied ja nur in einem einzigen zusätzlichen Task läge. Für solche Fälle möchte ich im Folgenden drei andere Vorschläge präsentieren.

Abbildung 2 zeigt eine erste, grobe Überlegung. Hier sind die unterschiedlichen Varianten durch Annotierungen gekennzeichnet. Zur „Bauzeit“ des Systems werden dann alle nicht gewählten Varianten weggelassen. Dieser Ansatz ist jedoch aufgrund zweier Punkte mehr als problematisch. Zum einen beinhaltet YAWL keine Anmerkungselemente, es wäre also eine grafische Erweiterung der Sprache nötig, und zum anderen kann es eventuell bei unpassender Kombination mehrerer Varianten zu Deadlocks kommen.

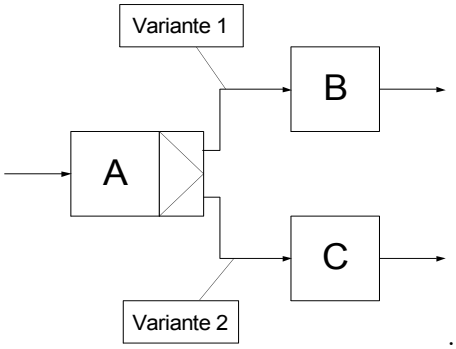


Fig. 2. Kennzeichnung von Varianten mit Hilfe von grafischen Erweiterungen

Ich möchte diese Methode auch gar nicht weiter verfolgen, sondern gleich eine weitere Alternative vorschlagen. Diese ist beispielhaft in Abbildung 3 gezeigt. Wir nehmen an, dass „A“ und „C“ zur Standardversion gehören und Task „B“ zu einer hier nicht näher spezifizierten „Variante 1“. „B“ ist in diesem Fall als multiple Instanz eines atomaren Tasks dargestellt. Die Werte zwischen den eckigen Klammern darüber geben folgende Informationen an (in ebendieser Reihenfolge):

1. Minimale Anzahl der Instanzen.
2. Maximale Anzahl der Instanzen.
3. Schwellwert; Anzahl der Instanzen die beendet sein müssen, bevor der Task als abgeschlossen angesehen wird.
4. Erzeugungsmodus (statisch/dynamisch); dynamisch: Während der Task läuft, können weitere Instanzen erzeugt werden; statisch: Die Anzahl der Instanzen wird entschieden wenn die Ausführung des Tasks startet.

Wir sehen, dass „B“ insgesamt n mal ausgeführt wird. Der Trick an dieser Stelle liegt nun an der Definition des Wertebereichs für n. Diesen setzen wir auf {0, 1}, was dazu führt, dass „B“ entweder null- oder einmal ausgeführt werden kann.

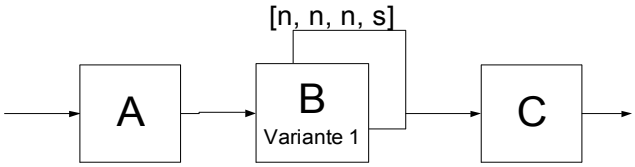


Fig. 3. Variabilitätsmodellierung unter Zuhilfenahme Multipler Instanzen.

Zur Systembauzeit ist es nun möglich durch entsprechende Belegung von n die „Variante 1“ zu inkludieren (n=1) oder diese auszulassen (n=0). Es ist weiterhin denkbar, auch eine dynamische Entscheidung zu ermöglichen. Hierzu bleibt n zur Bauzeit unbestimmt und wird erst zur Laufzeit ermittelt. Als Beispiel könnte man sich erneut den oben erwähnten, kostenpflichtigen Belegungsvorgang betrachten. Nehmen wir an, Abbildung 3 zeige einen Ausschnitt dieses Vorgangs, „A“ und „C“ gehörten somit zum regulären Ablauf und „B“ stünde für den Bezahlvorgang. Würde nun das System auf diese fiktive Studienordnung 2004 umgestellt, müssten alle Studenten, die

nach der neuen Ordnung studieren, bezahlen, die, die jedoch noch nach der Alten angefangen haben, aber nicht. So würde zur Laufzeit eine Ermittlung stattfinden (sehr Wahrscheinlich über eine XQuery-Anfrage), um was für einen Studenten es sich handelt und daraus der Wert von n bestimmt.

Auch diese Alternative Variabilitäten in YAWL zu modellieren ist nicht vollständig problemlos. Durch die Optionalität von „B“ kann Task „C“ nun sowohl die Ausgabe von Task „A“ als auch die von „B“ als Eingabe erhalten. Hier muss folglich auf Datenflussebene für Kompatibilität gesorgt werden. Des Weiteren müsste zur Anwendung dieser Modellierungsvariante auch eine Änderung in der Spezifikation von YAWL erfolgen: Der Wertebereich von n ist dort mit $\mathbb{N}$ angegeben, hier wäre dagegen № erforderlich. Nicht zu vergessen ist ebenso die Tatsache, dass die Anwendung dieser Methode einen starken Missbrauch der multiplen Instanzen darstellen würde, sind diese doch schließlich dafür gedacht, die (mögliche) gleichzeitig mehrfache Ausführung eines Tasks darzustellen und nicht dessen optionale Ausführung.

Abschließend stelle ich eine letzte Überlegung vor, welche auf dem XML-Charakter der YAWL-Implementierung [5] basiert.

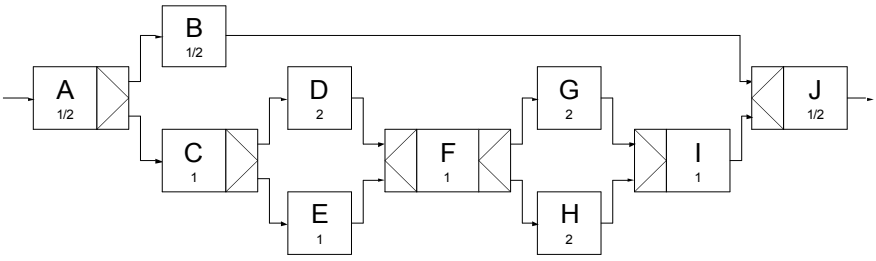


Fig. 4. Beispielhafter Prozessausschnitt.

Die Tasks in dem in Abbildung 4 gezeigten Prozessausschnitt tragen zu ihrem Namen noch eine Nummer. Diese soll angeben, zu welcher Version ein Task gehört. „A“ ist zum Beispiel in beiden Varianten enthalten, „C“ dagegen gehört nur zu Version 1. Die einzelnen Elemente sind nun im XML Format abgespeichert. Eine mögliche Speicherung des Tasks „A“ sähe folgendermaßen aus:

```
<element>
          <name>A</name>
          <version>1</version>
          <version>2</version>
            …
          <work_to_do>…</work_to_do>
</element>
```

Die Idee liegt jetzt darin, dass nur die zu der/den gewünschten Version(en) gehörenden Tasks *wirklich* ausgeführt werden. So werden mit Hilfe einer XQuery Anfrage zuerst sämtliche Tasks ermittelt, welche *nicht* die entsprechenden Versionsnummern tragen. Bei eben diesen wird nun die auszuführende Arbeit („work_to_do“ im Beispiel) auf „NULL“ gesetzt. Sie bleiben weiterhin im Prozess, laufen jedoch automatisiert, für den Nutzer transparent und ohne Daten zu verändern ab.

Nehmen wir als Beispiel den in Abbildung 4 gezeigten Prozessausschnitt. Es sei nur Version 1 gewünscht. Zuerst werden also alle „überflüssigen“ Tasks ermittelt. Dies wären hier „D“, „G“ und „H“. Danach werden die mit ihnen verbundenen Tätigkeiten gelöscht. Befinden wir uns nun zur Laufzeit in „F“, so findet der AND-Split auf jeden Fall statt, auch wenn beide nachfolgenden Tasks nur zur Version 2 gehören. Diese beiden Tasks („G“ und „H“) werden automatisch abgewickelt, führen jedoch keinerlei Arbeit durch und aktivieren Task „I“. Für den Nutzer hat es folglich den Anschein, als ob es sich einfach nur um eine Sequenz von „F“ und „I“ handelt.

Der Vorteil des Ganzen wird schnell deutlich: Dadurch, dass keine Löschung von Tasks oder Umleitung von Kontrollflüssen stattfindet, ist es bei beliebiger Variantenauswahl unmöglich einen Deadlock zu produzieren, sofern er nicht von vorne herein vorhanden war. Syntaktische Korrektheit ist also gewährleistet, semantisch stellt sich die Sache natürlich ganz anders dar. Das Verfahren kann zwar garantieren, dass eine beliebige Kombination der Varianten ausführbar ist, aber nicht, dass der entstehende Prozess sinnvoll ist.

Nachteilig an dieser Lösung ist vor allem die Tatsache, dass das entstehende Diagramm einen sehr komplexen Charakter erlangen kann, je nachdem wie viele und wie große Varianten eingebaut werden. Dies ist dann selbst für den Systemdesigner keine triviale Aufgabe mehr. Man muss sich allerdings auch immer überlegen, welchen Nutzen die Diagramme im Endeffekt haben sollen. Geht es darum, mit anderen Menschen über die gezeigten Sachverhalte zu kommunizieren, dann ist es sicher sinnvoll Varianten getrennt zu betrachten. Soll der Plan allerdings nur über ein Tool in eine Workflow-Engine exportiert und dort ausgeführt werden, dann ist es letzten Endes (fast) egal, wie komplex das Ausgangsdiagramm ist.

Letztendlich möchte ich aber noch einmal darauf hinweisen, dass es in den Überlegungen ohnehin um die Darstellung kleinerer Variabilitäten ging. Größere Änderungen lassen sich definitiv am Besten mit dem Austauschen eines Subprozesses lösen.

5 Zusammenfassung und Ausblick

Das letzte Kapitel meiner Ausarbeitung möchte ich mit einer (subjektiven) Bewertung von YAWL beginnen.

Positiv stellt sich vor allem dar, dass der (zahlenmäßige) Umfang der Elemente bei YAWL recht klein ist. Dies senkt nämlich nicht nur den Lernaufwand für den Designer, sondern auch für andere Betrachter. So sind mit YAWL angefertigte Diagramme auch für Laien nach nur wenigen erläuternden Worten verständlich. Auch in der Ausdrucksstärke kann YAWL punkten. Die Befürchtung, dass die

Beschränkung auf wenige Elemente zu einer verminderten Ausdrucksfähigkeit führt, ist definitiv ungerechtfertigt, was die Zahl der unterstützten Workflowpattern beweist[4].

Es gibt allerdings auch negative Aspekte an YAWL. Es ist in meinen Augen unverständlich, dass es nicht möglich ist, Bedingungen an Kanten anzufügen. Dies führt nämlich zu einer unnötig starken Benutzung der Zustände (Stellen). Wir erinnern uns zum Beispiel an den im Hauptteil behandelten Top-Level Prozess. Dort gab es den Task „Antrag auf Graduierung einreichen“. Hinter diesem befand sich ein XOR-Split, welcher zwei Wege ermöglichte: Erstens in den Zustand „Antrag vollständig“, zweitens in den Zustand „Antrag unvollständig“. Beide Informationen hätten meiner Meinung nach mehr Sinn als Kantenbeschriftung, da sie ja nicht unbedingt einen Zustand angeben sollen, sondern vielmehr eine Verzweigungsvorschrift. Weiterhin ist es störend, dass eine Ressourcenperspektive nicht modellierbar ist. Es soll zwar in der Implementierung irgendwann möglich sein, einen Task einer bestimmten Person/Gruppe zuzuordnen [5], aber es wäre sicherlich sinnvoll, dieses auch im Diagramm zu erkennen. Letztendlich bin ich der Meinung, dass Diagramme mit vielen Abbrüchen, durch eine Flut von gestrichelten Linien, leicht unübersichtlich werden können. Deswegen würde ich mir, zumindest für die Visualisierung am Computer, eine grafisch schönere Lösung wünschen. Vorstellbar wäre hier zum Beispiel, alle abbrechenden Tasks mit einem kleinen „c“ (wie „cancel“) zu kennzeichnen. Erst durch einen Klick auf einen solchen Task, würde dann die zugehörige Abbruchregion kenntlich gemacht (z.B. rot unterlegt).

Zusammenfassend lässt sich sagen, dass mit YAWL definitiv ein richtiger Weg eingeschlagen wurde. Die Frage ist jedoch, was die Zukunft bringen wird. YAWL sollte auf jeden Fall noch kleinere Änderungen, wie zum Beispiel die Hinzunahme von Kantenbeschriftungen, erfahren. Auch besteht weiterer Forschungsbedarf was die Darstellung von Variabilität angeht, wie ich im vorherigen Kapitel sicherlich gezeigt habe. Ob YAWL allerdings eine Zukunft hat, in dem Sinne, dass die Sprache Verwendung findet, hängt meiner Meinung nach stark von dem Fortschritt der Toolentwicklung ab. Im Moment sind die Tools noch nicht ernsthaft verwendbar (Die syntaktische Überprüfung funktioniert zum Beispiel nicht, sobald man ein XOR-Split einbaut; es sind keine Verzweigungsbedingungen angebbar; bisher erscheinen alle Tasks auf allen Arbeitslisten...). Sollte sich daran in absehbarer Zeit etwas ändern, halte ich es für gut möglich, dass YAWL in größerem Umfeld zum Einsatz kommt. Andernfalls, sollten die Tools keine großartigen Verbesserungen erfahren, sehe ich ehrlich gesagt keine große Zukunft für YAWL und gehe davon aus, dass die Sprache wieder in der Versenkung verschwinden wird. So Schade dies auch wäre.

Literatur

1. Studienbestimmungen für den Bachelor- und den Masterstudiengang der Softwaresystemtechnik an der Universität Potsdam in der Fassung vom 14. Juni 2001

[4] YAWL unterstützt sämtliche der 20 Workflowpattern, mit Ausnahme der „Implicit Termination” [4].

2. Prüfungsbestimmungen für die Studiengänge der Softwaresystemtechnik an der Universität Potsdam in der Fassung vom 14. Juni 2001
3. Studien- und Prüfungsbestimmungen für den Bachelor- und den Master-Studiengang der Softwaresystemtechnik an der Universität Potsdam in der Version vom 27.01.2004
4. W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language (Revised version). QUT Technical report, FIT-TR-2003-04, Queensland University of Technology, Brisbane, 2003.
5. W.M.P. van der Aalst, L. Aldred, M. Dumas, and A.H.M. ter Hofstede. Design and Implementation of the YAWL system. To appear in Proceedings of The 16th International Conference on Advanced Information Systems Engineering (CAiSE 04), Riga, Latvia, June 2004. Springer Verlag.
6. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. QUT Technical report. FIT-TR-2002-02, Queensland University of Technology, Brisbane, 2002

Appendix

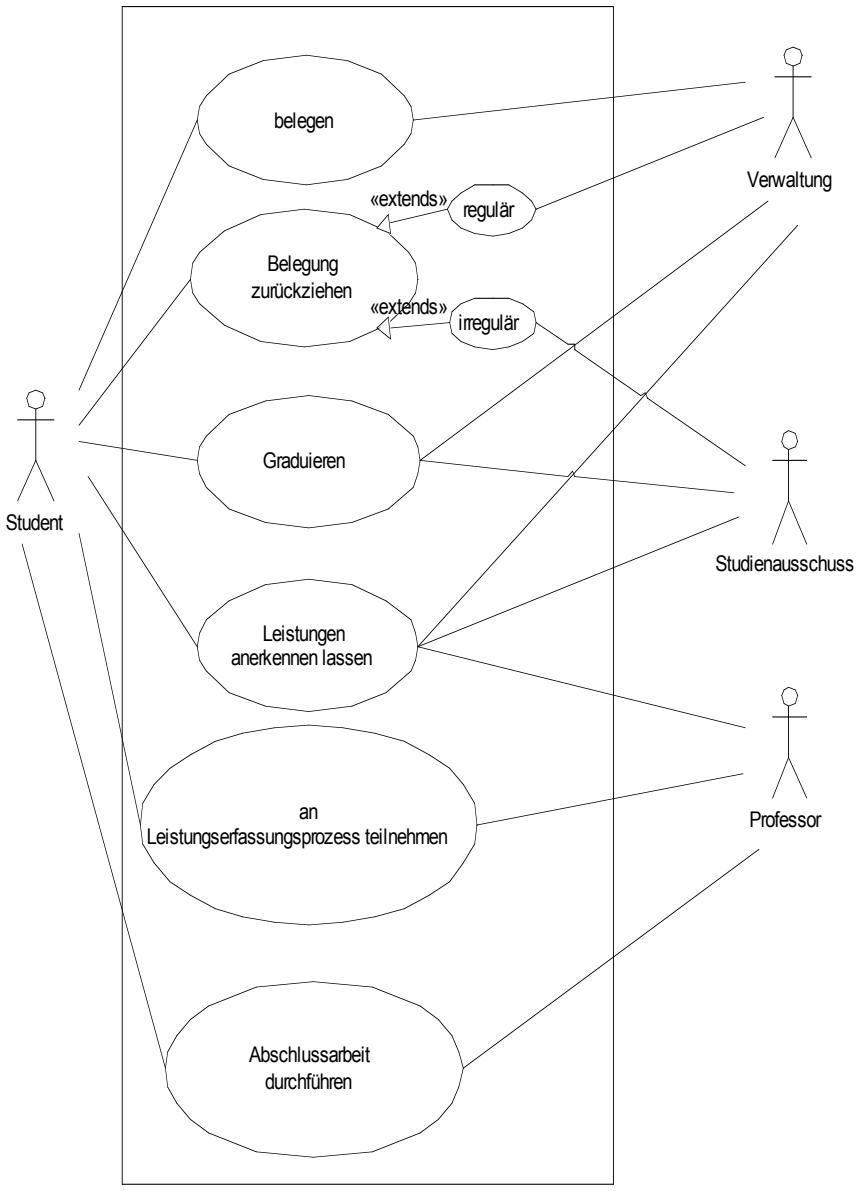


Fig. 5. Typische Use Cases eines HPI-Studenten.

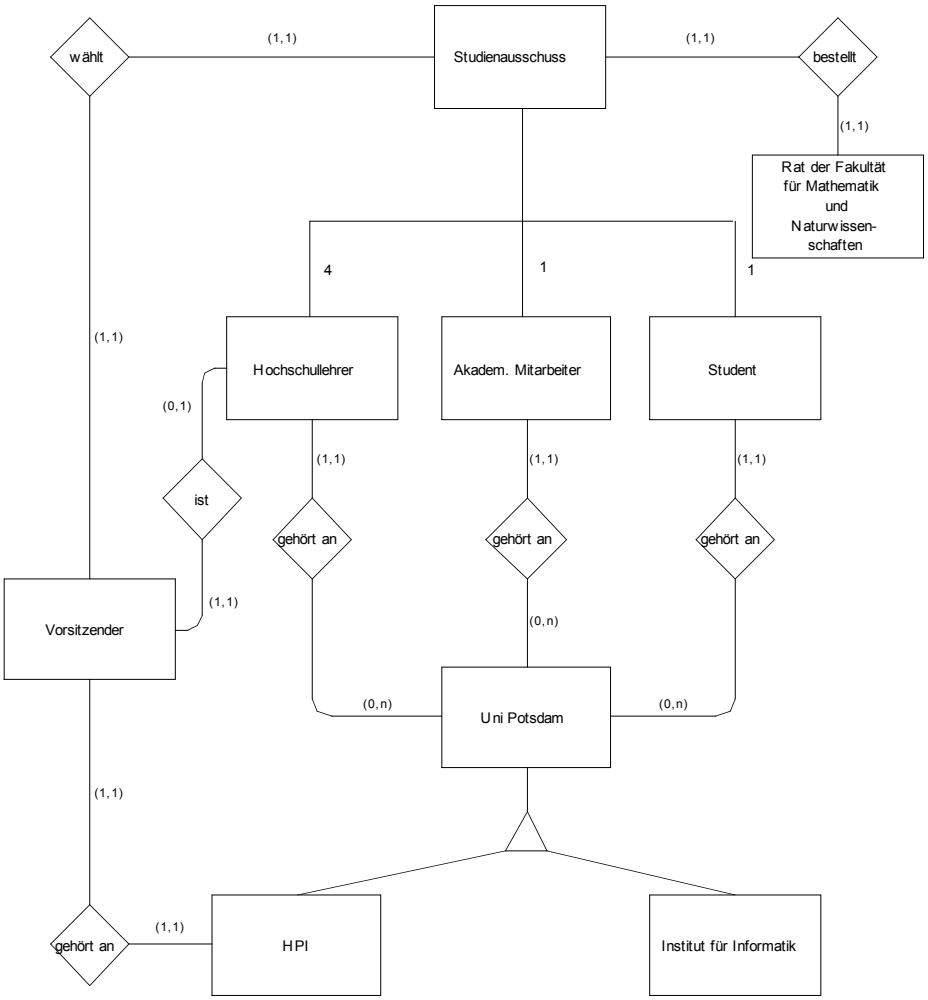
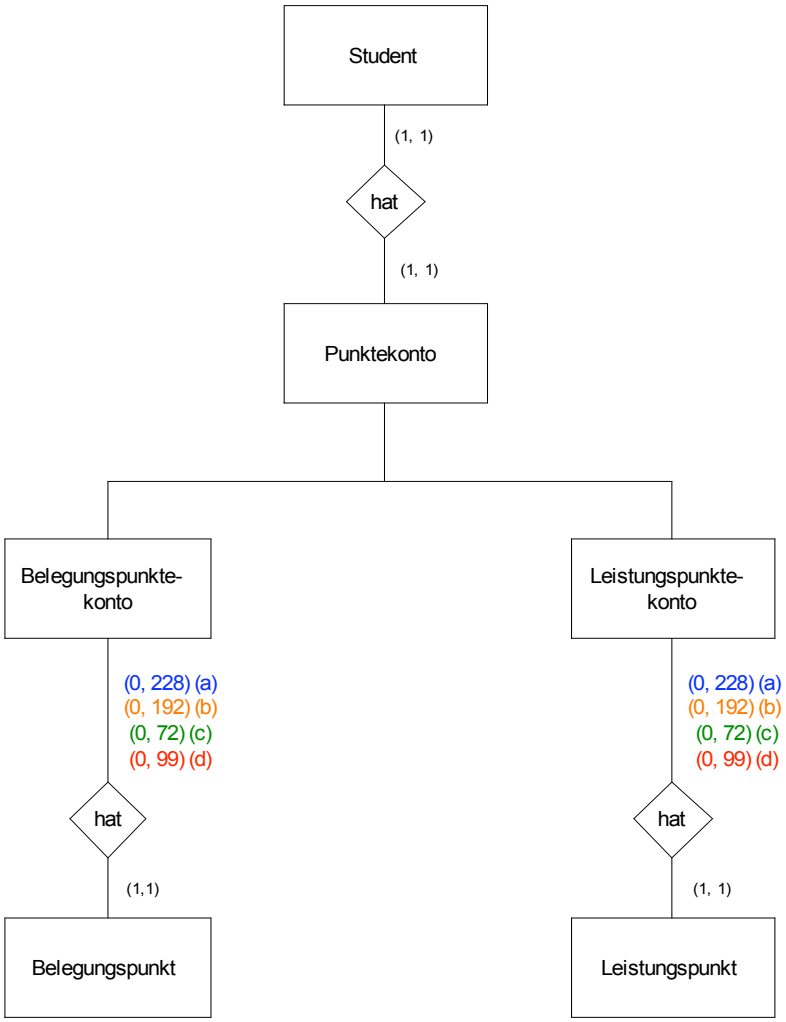


Fig. 6. Aufbau des Studienausschusses.



Wobei: #Leistungspunkte = 228 - #Belegungspunkte - x

x = „Verlorene“ Punkte

Legende:

(a) Bachelor 2001
(b) Bachelor 2004
(c) Master 2001
(d) Master 2004

Fig. 7. Das HPI-Punktekonto.

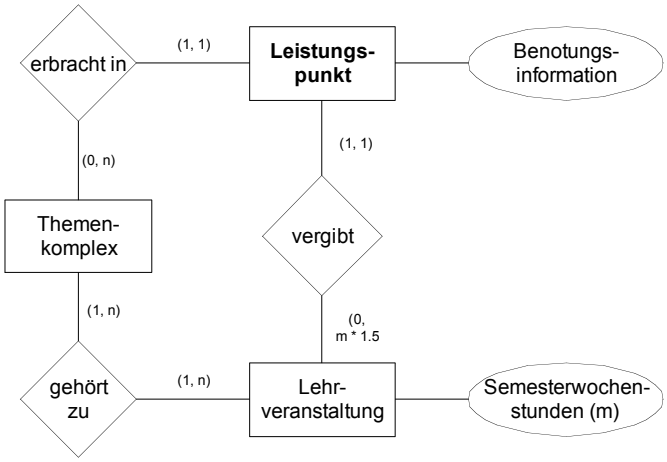


Fig. 8. Nähere Betrachtung des Leistungspunktes.

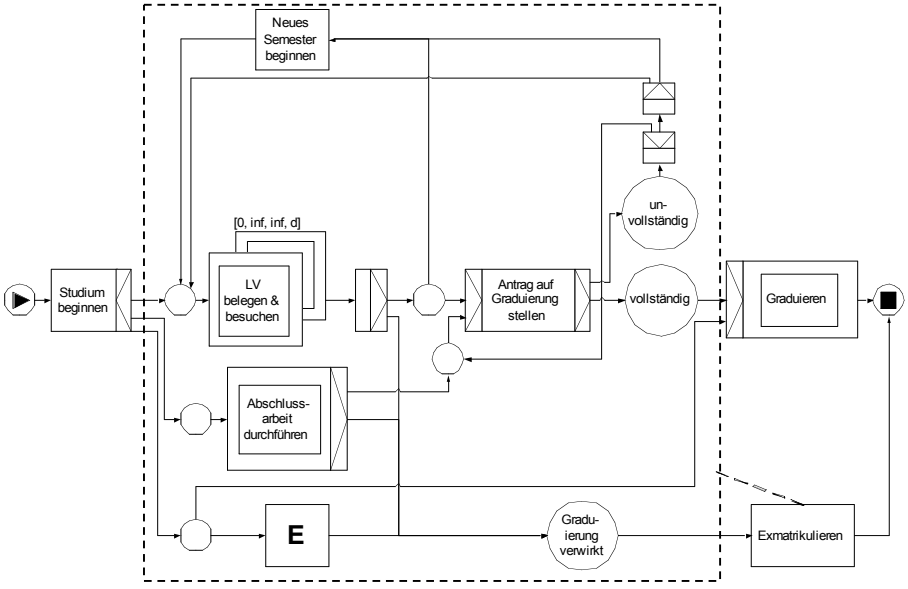


Fig. 9. Der Top-Level Prozess

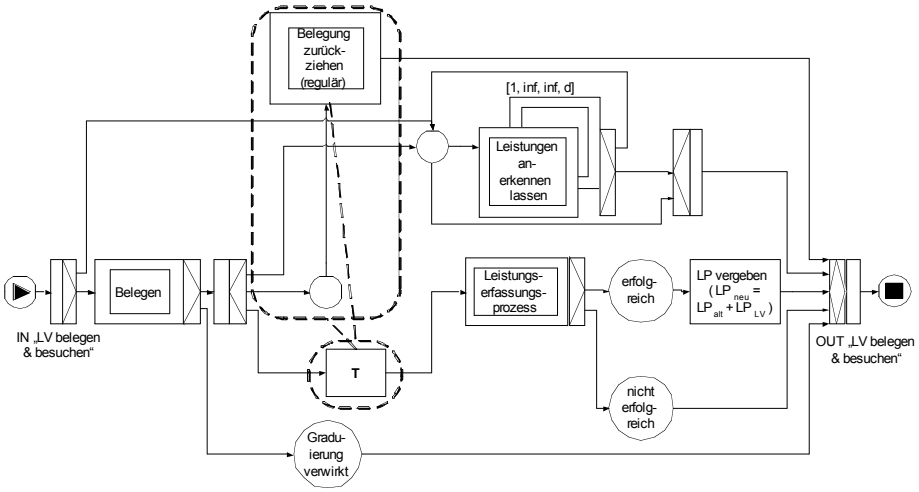


Fig. 10. Lehrveranstaltungen belegen und besuchen.

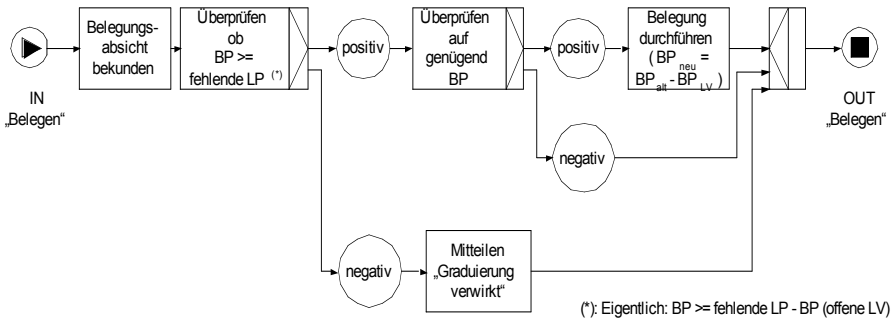


Fig. 11. Belegen einer Lehrveranstaltung.

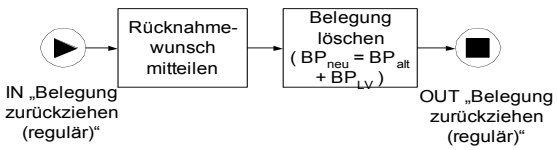


Fig. 12. Belegung zurückziehen (regulär).

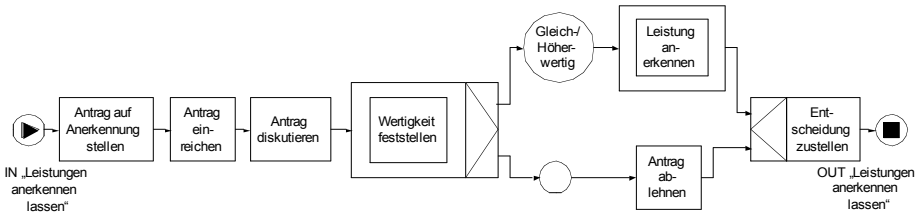


Fig. 13. Leistung anerkennen lassen.

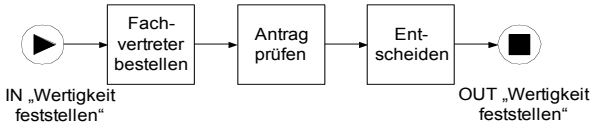


Fig. 14. Wertigkeit einer Leistung feststellen.

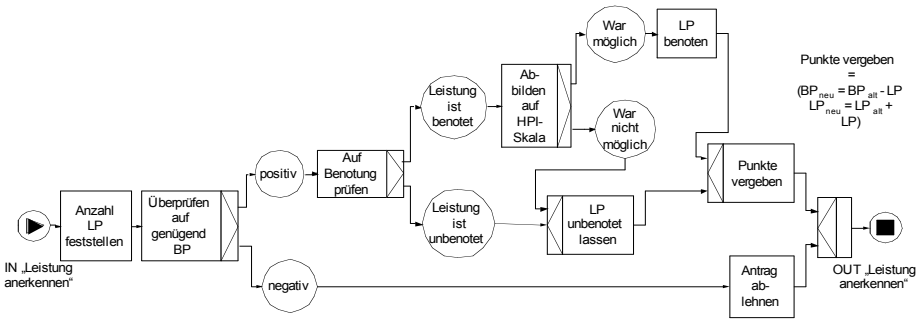


Fig. 15. Leistung anerkennen.

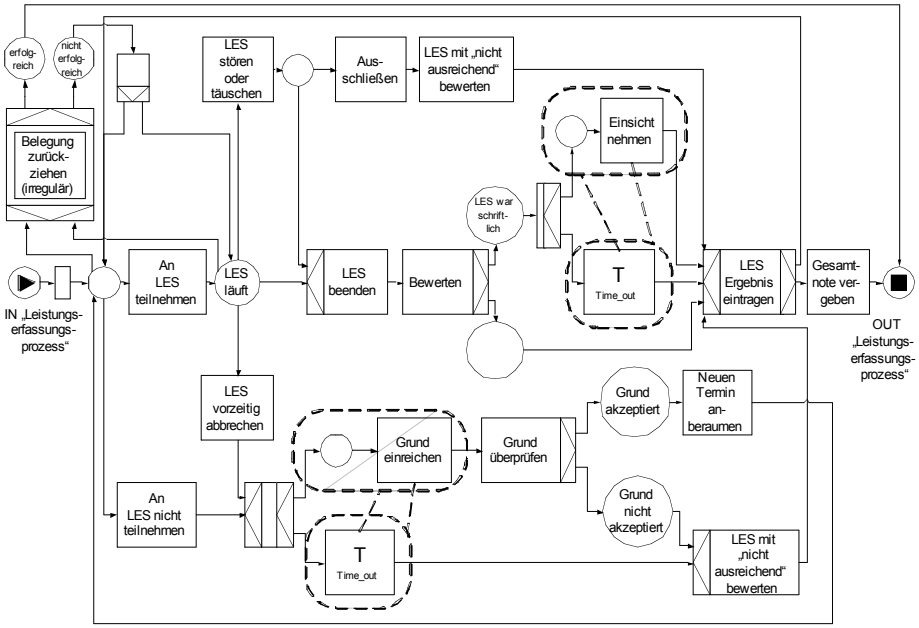


Fig. 16. Leistungserfassungsprozess.

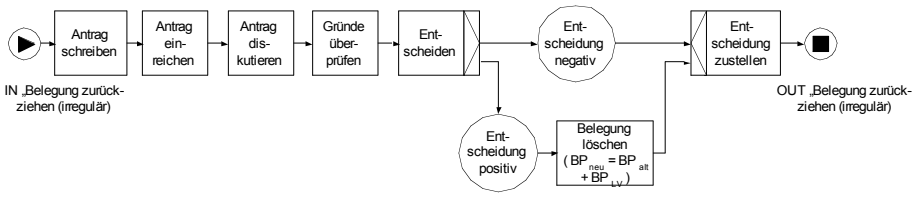


Fig. 17. Belegung zurückziehen (irregulär).

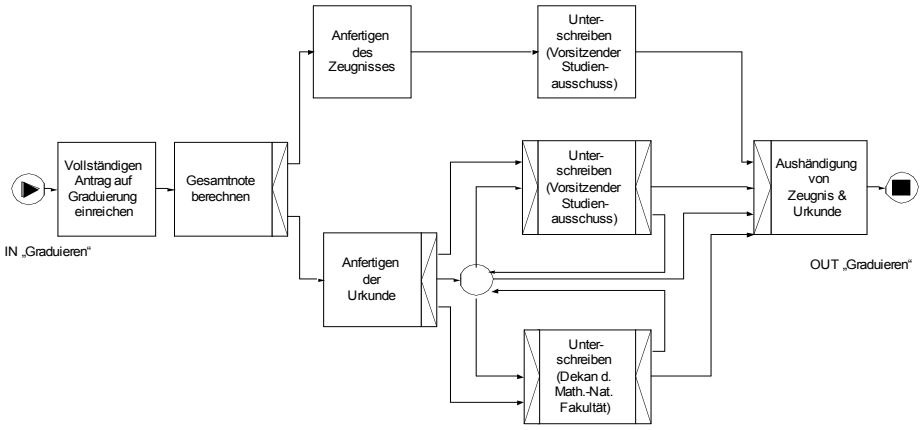


Fig. 18. Graduieren.

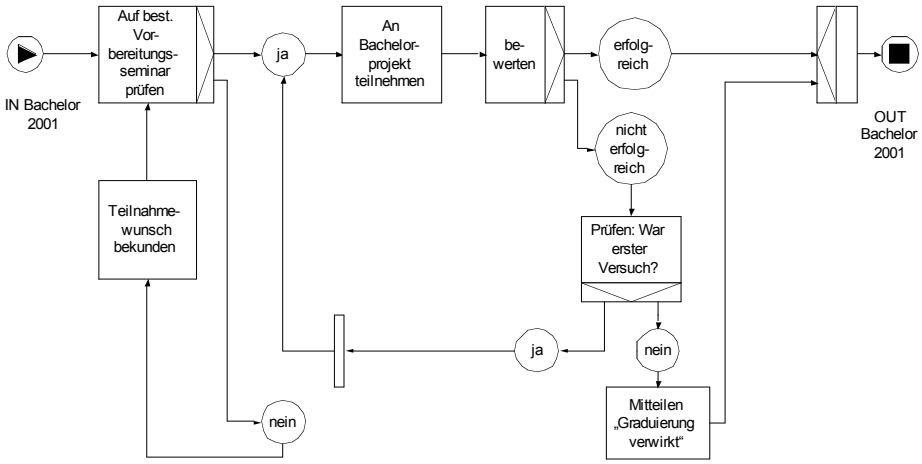
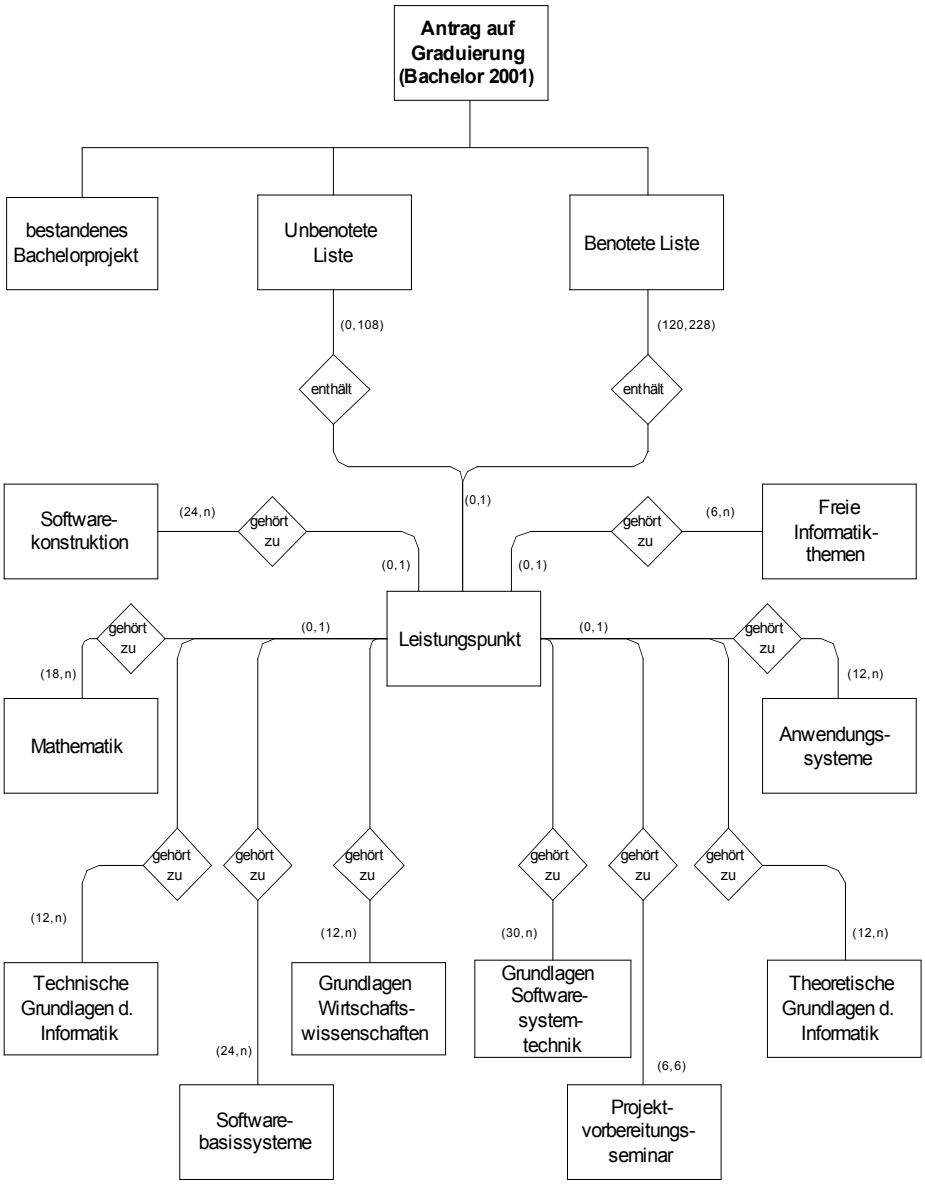


Fig. 19. Abschlussarbeit durchführen (Bachelor 2001).



Bed.: 228 >= #Leistungspunkte(unbenotete Liste) + #Leistungspunkte(benotete Liste) >= 168

Fig. 20. Antrag auf Graduierung (Bachelor 2001).

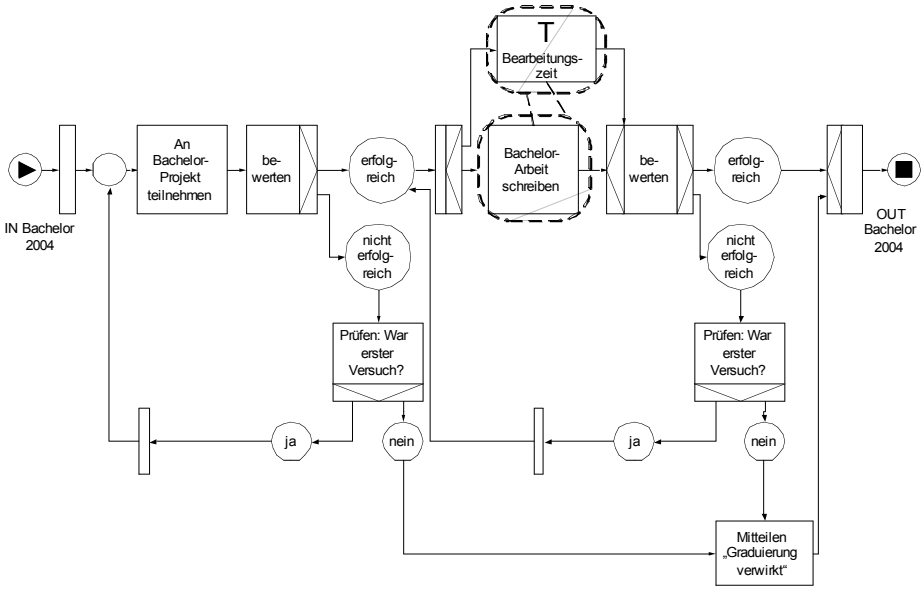


Fig. 21. Abschlussarbeit durchführen (Bachelor 2004).

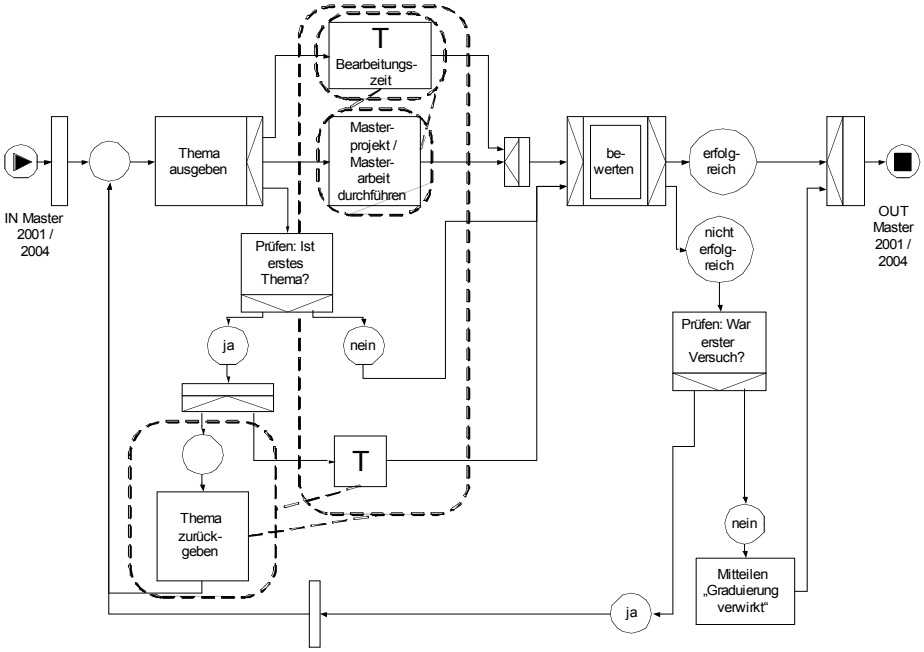


Fig. 22. Abschlussarbeit durchführen (Master 2001/2004).

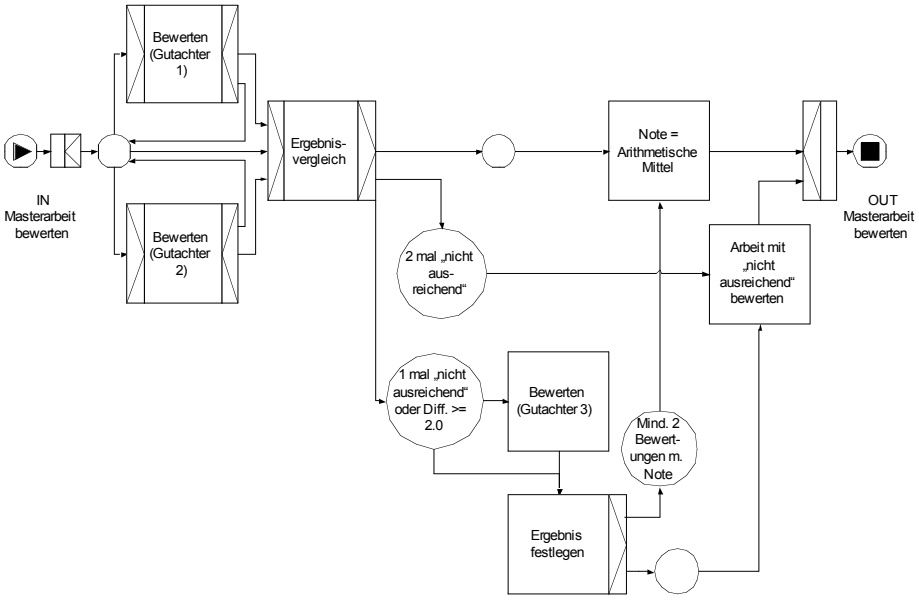
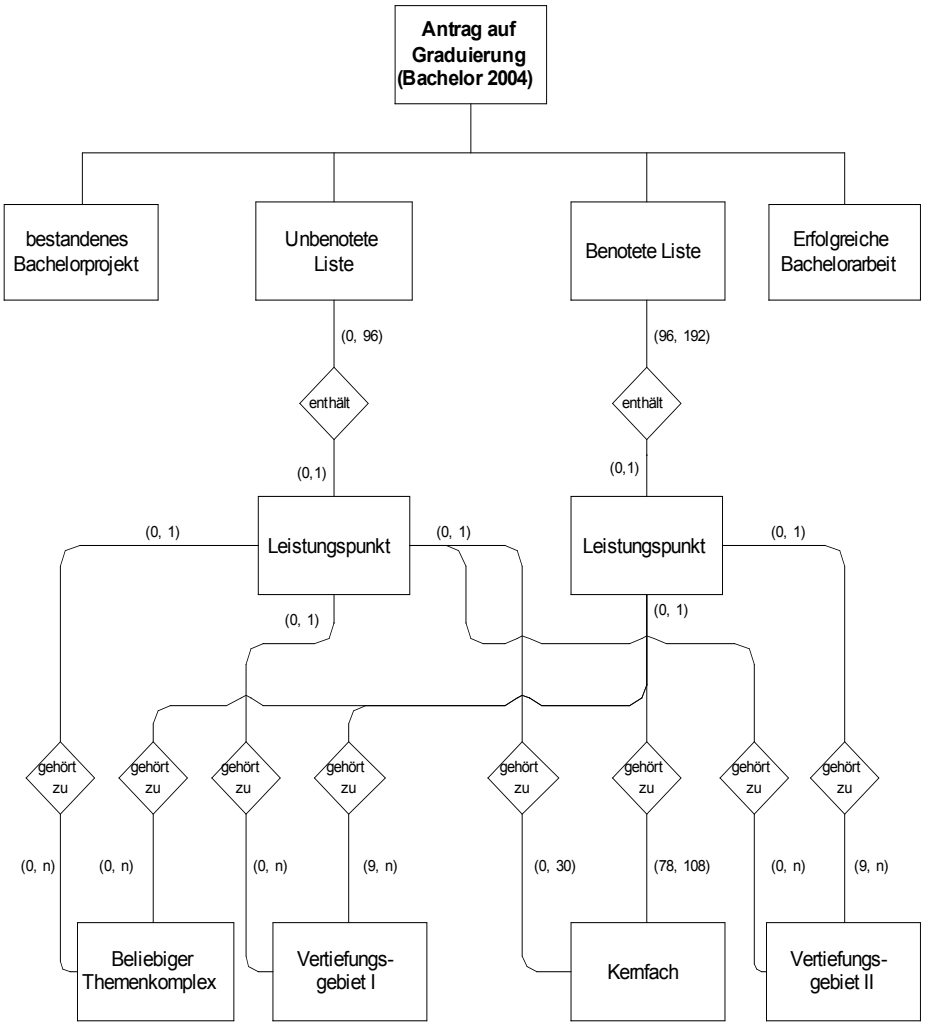
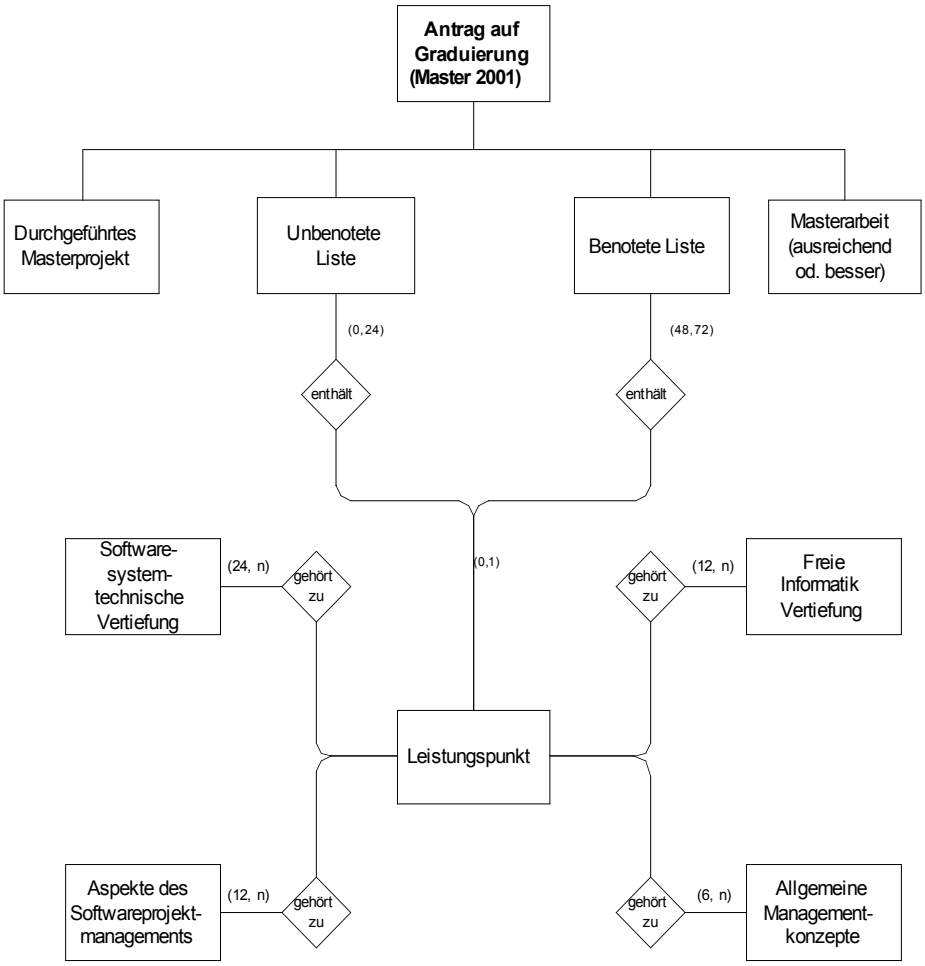


Fig. 23. Masterarbeit bewerten.



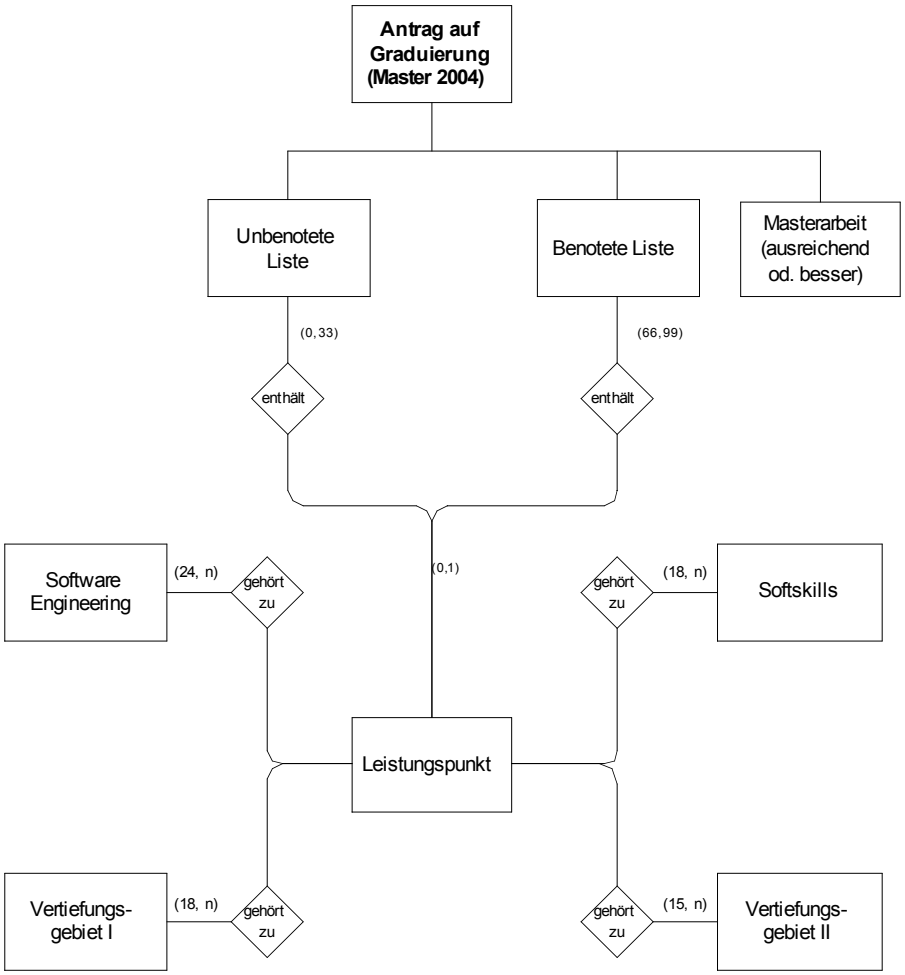
Bed.: 192 >= #Leistungspunkte(unbenotete Liste) + #Leistungspunkte(benotete Liste) >= 132
∑ Leistungspunkte(Kernfach) = 108
∑ Leistungspunkte(Vertiefungsgebiet I) + Leistungspunkte(Vertiefungsgebiet II) >= 24
Folgende Kernfächer müssen benotet eingebracht werden:
- Modellierung I und II,
- Softwaretechnik I und II,
- Softwarearchitektur,
- Betriebssysteme
- 2 ausgewählte Veranstaltungen im Umfang von 12 Leistungspunkten aus Softwarebasissysteme I bis III

Fig. 24. Antrag auf Graduierung (Bachelor 2004).



Bed.: 72 >= #Leistungspunkte(unbenotete Liste) + #Leistungspunkte(benotete Liste) >= 48

Fig. 25. Antrag auf Graduierung (Master 2001).



Bed.: 99 >= #Leistungspunkte(unbenotete Liste) + #Leistungspunkte(benotete Liste) >= 75

Benotete Liste ⊃ 12 LP _ Software Engineering + 9 LP _ Softskills
+ 9 LP _ Vertiefungsgebiet I + 9 LP _ Vertiefungsgebiet II

Fig. 26. Antrag auf Graduierung (Master 2004).

Modellierung von Verwaltungsprozssen
Studienverwaltungsprozesse am HPI

Udo Werner

Hasso-Plattner-Institut für Softwaresystemtechnik
Postfach 90 04 60, 14460 Potsdam
udo.werner@gmx.net

Zusammenfassung Immer mehr Unternehmen sind daran interessiert, detailliertes Wissen über ihre Geschäftsprozesse zu erlangen. Um einen Prozess tatsächlich bis ins Detail zu durchdringen, bedarf es der Kenntnisse über die verschiedenen Aspekte, die seine Ausprägung beeinflussen. Zur Darstellung dieser verschiedenen Prozesssichten bedient man sich immer öfter neben rein textuellen Beschreibungen auch verschiedener Modellierungstechniken.
Ziel dieser Ausarbeitung ist die Beschreibung der Studienverwaltungsprozesse der Hasso-Plattner-Institut für Softwaresystemtechnik GmbH, wobei zunächst die Erhebung der Prozessdaten und die dazu gemachten Vorüberlegungen erläutert werden. Im Anschluss werden die Prozesse in Form von Ist-Modellen zu den einzelnen Sichten beschrieben und Probleme, die bei der Modellierung aufgetreten sind, an Hand eines Beispiels erläutert. Dann werden Optimierungsmöglichkeiten ermittelt und in einem Soll-Modell erfasst. Abschließend werden die Ergebnisse der Modellierung einschließlich der Erkenntnisse über die Eignung der verwendeten Techniken vorgestellt.

Schlagworte: Prozessmodellierung, Verwaltungsprozesse, Studienverwaltung

1 Einleitung

Immer mehr Unternehmen sind daran interessiert, detailliertes Wissen über ihre Geschäftsprozesse zu erlangen. Die Motive dafür sind vielseitig und reichen von der Zurechenbarkeit von Kosten zu einzelnen Produkten über die Gewinnung von Daten zur Optimierung von Abläufen und der Verfügbarkeit von besseren Dokumentationen für Schulungszwecke bis hin zum Erreichen einer Zertifizierung des Unternehmens nach internationalen Qualitätssicherungsstandards.
Um einen Prozess tatsächlich bis ins Detail zu durchdringen, bedarf es der Kenntnisse über die verschiedenen Aspekte, die seine Ausprägung beeinflussen. Zu diesen Aspekten gehören die Ablaufbeschreibung eines Prozesses, seine Abhängigkeiten von anderen Prozessen und die Rollen in denen Personen an dem Prozess teilnehmen, sowie weitere, von der Art des Prozesses abhängige Aspekte. Bei Verwaltungsprozessen kommt beispielsweise den Daten (oder logischen Objekten), die innerhalb des Prozesses verarbeitet werden, eine besondere Bedeutung zu, während bei Fertigungsprozessen auch physikalische Objekte (z.B. ein konkretes Werkstück) von Interesse sind.
Zur Darstellung dieser verschiedenen Prozesssichten bedient man sich immer öfter neben rein textuellen Beschreibungen auch verschiedener Modellierungstechniken. Ziel dieser Ausarbeitung ist die Beschreibung der Studienverwaltungsprozesse der Hasso-Plattner-Institut für Softwaresystemtechnik GmbH (HPI). Die Aufgabenstellung legt fest, dass zu den Informationen, die für die Darstellung dieser Prozesse benötigt werden, eine Erhebung in Form eines Interviews durchzuführen ist. Anschließend sind die Studienverwaltungsprozesse in einem Ist-Modell darzustellen, welches die an den Prozessen beteiligten Personen, sowie die statischen und dynamischen Aspekte repräsentiert. Im Rahmen der Ausarbeitung wird dazu zunächst der Weg der Erhebung des Prozesses einschließlich dazu nötiger Vorüberlegungen beschrieben. Anschließend werden die verschiedenen Sichten auf das Ist-Modell der Prozesse vorgestellt und Probleme, die bei der Modellierung aufgetreten sind, an Hand eines Beispiels erläutert. Darauf aufbauend werden die Möglichkeiten zur Optimierung der Prozesse diskutiert und ein mögliches Ergebnis in Form eines Soll-Modells präsentiert. Abschließend werden die Ergebnisse der Modellierung zusammengefasst, speziell auch die Eignung der gewählten Modellierungstechniken für den jeweiligen Zweck.

2 Der Weg zu den Modellen

So wie Rom sprichwörtlich nicht an einem Tag erbaut wurde, wurden auch die Modelle der Studienverwaltungsprozesse nicht von heute auf morgen erstellt. Dieses Kapitel widmet sich deshalb den vorbereitenden Maßnahmen. Dazu wird zunächst grob die Vorgehensweise beschrieben. Anschließend werden Vorüberlegungen zu den Befragungs- und Analysemethoden, sowie zur Auswahl der Modellierungstechniken erläutert.

2.1 Vorgehensweise im Überblick

Die Studienverwaltung ist bei der HPI einer der Kernbereiche und dient der Unterstützung des Lehrbetriebs. Zu den in diesem Rahmen ablaufenden Prozessen existierten bislang keine detaillierten Beschreibungen. Um Modelle der verschiedenen Prozesssichten erstellen zu können, ist deshalb zunächst eine Erhebung der Prozessdaten in Form von Interviews erforderlich gewesen. Im Anschluss daran konnte dann nach Auswahl geeigneter Techniken die Modellierung der Ist-Prozesse erfolgen. Nach der ersten Entwurfsphase wurden im Rahmen einer Diskussionsrunde die Entwurfsergebnisse mit den Interview-Partnern erörtert und Überlegungen zu Verbesserungsmöglichkeiten am Ablauf zusammengetragen. Darauf aufbauend wurden zunächst die noch nicht übereinstimmenden Details am Modell des Ist-Prozesses korrigiert und dann eine intensive Analyse bezüglich der Optimierungsideen durchgeführt. Als Ergebnis der Analyse wurde ein Modell eines möglichen Soll-Prozesses erstellt und hinsichtlich der Umsetzbarkeit bewertet.

2.2 Erkenntnisse aus DIN EN ISO 9000-2000 ff.

Um eine effektive Befragung zu ermöglichen, wurde nach Verfahren gesucht, die in ähnlichem Kontext die Ermittlung, Darstellung und Analyse von Prozessdaten als Zielstellung haben. Dabei hat sich das nachfolgend näher beschriebene Vorgehen nach DIN EN ISO 9000-2000 ff. als untersuchenswert herausgestellt.

Grundlagen der DIN EN ISO 9000-2000 ff. Die Basis für eine erfolgreiche Umsetzung der Normen ist die Definition von Qualitätszielen. Diese Ziele beziehen sich einerseits auf die konkreten Produkte bzw. Dienstleistungen eines Unternehmens, andererseits aber auch auf die Prozesse, die zur Herstellung bzw. Erbringung und deren Unterstützung ablaufen. Zur Umsetzung dieser Qualitätsziele bedarf es der Implementierung eines so genannten Qualitätsmanagementsystems. Im Zusammenhang mit der DIN EN ISO 9000-2000 Familie wird daher auch von Prozessorientiertes Qualitäts-Management (PQM) bzw. einem PQM-System gesprochen. [1, 2]
Der Ablauf zum Aufbau eines PQM-Systems sieht im Wesentlichen die folgenden Schritte vor [1]:

- Identifikation der relevanten Prozesse
- Festlegung der Abfolge und Wechselwirkungen der Prozesse
- Definition von Kriterien und Methoden zur Kontrolle jedes Prozesses
- Prüfung der Verfügbarkeit von Informationen und Ressourcen zur Durchführung und Überwachung der Prozesse
- Durchführung von Messungen und Analysen zur Ermittlung von Optimierungspotentialen
- Anpassung der zu optimierenden Prozesse

Ein PQM-System hat dabei verschiedene Darstellungsebenen bezogen auf den Grad der Detaillierung. An oberster Stelle ist die Prozesslandschaft als Übersicht über das gesamte Unternehmen zu sehen. Eine Ebene tiefer spricht man von Prozessgruppen, die sich wiederum in einzelne Prozesse aufteilen. Auf unterster Ebene befinden sich die Prozessdetails. Dieser letzten Ebene werden beispielsweise konkrete Arbeits- und Prüfanweisungen für einen Prozess sowie Formulare und Checklisten zugeordnet. [1]
Die ersten Schritte zum Aufbau eines PQM-Systems befassen sich mit der Identifikation relevanter Prozesse und der Festlegung von deren Abfolge und Wechselwirkungen. Dazu ist in jedem Fall zunächst eine Erhebung der dafür benötigten Daten erforderlich. In der Regel wird diese Erhebung formularbasiert in Form von Interviews durchgeführt. Die zur Befüllung der Formulare erforderlichen Fragen werden dabei nach der so genannten „6W"-Methode formuliert: Was, Warum, Wie, Wer, Wann, Wo. Auf Grundlage der so gewonnenen Profile werden Ablaufpläne für die einzelnen Ebenen erstellt und um die zu erfüllenden Qualitätsziele ergänzt. Nach der Implementierung der entsprechenden Ablaufsteuerung und der zugehörigen Messeinrichtungen kann mit der Analyse der Prozesse begonnen werden. Dazu werden die gesetzten Qualitätsziele mit den Messwerten verglichen. Sollten Abweichungen auftreten oder durch Aktualisierung der Qualitätsziele die Anforderungen verändert sein, gilt es mögliche Optimierungspotentiale zu finden. [1, 2]
Dazu kann beispielsweise ein Ishikawa-Diagramm [3] zur Darstellung von Ursache-Wirkungs-Zusammenhängen verwendet werden. Beim Ishikawa-Diagramm wird die „6M"- (bzw. „7M"-) Methode verwendet um mögliche Ursachen zu kategorisieren: Mensch, Maschine, Methode, Material, Milieu, Messung, (Management). Gesucht werden die „3 Mu" (nach Kaizen), übernommen aus dem Japanischen für Verschwendung (Muda), Überlastung (Muri) und Abweichung (Mura). Als Beispiel einer Überlastung könnte bei einer Verpackungsmaschine festgestellt werden, dass sie 25000 Einheiten pro Tag verpacken müsste, nominal aber nur eine Kapazität von 1000 Einheiten pro Stunde hat und eine Verschwendung könnte dann ermittelt werden, wenn die nicht verpackten 1000 Einheiten einfach vernichtet würden, während es eine Abweichung wäre, wenn diese 1000 Einheiten einfach unverpackt weitergegeben würden. Für die konkrete Analyse und das Finden von Optimierungsansätzen ergeben sich also aus Kombination der 6W, 6M und 3 Mu wieder eine Reihe von Fragen zur Ermittlung von Ursachen oder Alternativen. Nachfolgend einige beispielhafte allgemein gehaltene Fragen:

- Wer macht es? Wer sollte es machen?
- Wer kann es noch machen? Wer sollte es noch machen können?
- Wann wird es gemacht? Wann sollte es gemacht werden?
- Wann kann es noch gemacht werden? Wann sollte es noch gemacht werden können?

Auswertung der Betrachtung von DIN EN ISO 9000-2000 ff. Im Gegensatz zur vorgestellten Norm gehörte die Einbettung in Qualitätsdefinitionen

und damit unmittelbar verbundener Maßnahmen wie die Implementierung von Methoden zur Messung von Prozesskennzahlen nicht zu den gesuchten Verfahrensweisen. Grundsätzlich ließe sich aber eine Einordnung der Aufgabenstellung in das Vorgehen zur Implementierung eines PQM-Systems vornehmen: Eine Prozessgruppe der Verwaltungsprozesse der HPI, nämlich die Studienverwaltungsprozesse, sollte nach der dazu notwendigen Datenerhebung modelliert und analysiert werden. Die entsprechenden Verfahren zur Erstellung der Darstellungen für die Ebenen 2 bis 4 eines PQM-Systems konnten also bei Weglassung der qualitätsbezogenen Aspekte verwendet werden.

2.3 Datenerhebung im Detail

In Umgebungen, in denen Prozesse von Menschen ausgeführt werden, ist die Durchführung von Interviews mit diesen Menschen das am besten geeignete Mittel zur Erhebung der für den Ist-Prozess relevanten Daten. Die Interviews müssen dabei gut vorbereitet werden. Dazu sind im Vorfeld folgende Fragen zu klären:

- Welche Informationen werden benötigt?
- Welche Personen können Aussagen dazu treffen?
- Welche Fragen liefern die notwendigen Antworten?

Im Fall der Studienverwaltungsprozesse mussten zunächst die Mitarbeiter identifiziert werden, die an den relevanten Prozessen beteiligt sind. Im Rahmen der Vorbereitung der Datenerhebung wurden diese Mitarbeiter über das Vorhaben informiert und es wurden Termine für die Interviews mit ihnen vereinbart. Die benötigten Informationen sind dabei:

- die einzelnen Aufgaben und Rollen, die von den Mitarbeitern wahrgenommen werden
- die Abfolge bestimmter Tätigkeiten zur Erfüllung dieser Aufgaben
- weitere Rollen bzw. Personen, mit denen sie im Rahmen der Ausführung der Tätigkeiten in Kontakt kommen
- die Daten, die im Rahmen der Ausführung von Bedeutung sind

Die in Kapitel 2.2 gewonnenen Erkenntnisse waren hilfreich bei der Zusammenstellung der Fragen, die im Interview gestellt wurden. Das Ergebnis dieser Fragen ist der im Anhang 6.1 aufgeführte Prozesskatalog, der diese Basisinformationen zu den einzelnen Aufgaben enthält.

2.4 Auswahl der Modellierungstechniken

Nach der Erstellung des Prozesskataloges kann man sich einen Überblick über die Prozessdaten verschaffen und Überlegungen zur Auswahl geeigneter Techniken für die Modellierung der Prozesse anstellen.

Datenmodell Von den gängigen Techniken zur Modellierung von Daten haben sich im vorliegenden Fall zwei zur Auswahl gestellt: Das UML-Klassendiagramm aus dem Bereich der Softwareentwicklung und das Entity-Relationship Diagram (ERD) aus dem Bereich der Datenbankentwicklung. Beide Diagrammtypen unterstützen die Zuordnung bestimmter (Daten-)Attribute zu Entitäten. Während Klassendiagramme vorrangig für die Darstellung von Hierarchie-Beziehungen von gleichartigen Entitäten, wie Vererbung oder Aggregation, verwendet werden und diese Entitäten auch als Aktoren auffassen, die verschiedene Methoden ausführen können, sind ERDs in erster Linie für die Beziehungen unterschiedlicher Entitäten gedacht, wobei die Entitäten passiv gesehen werden.
Das zu erstellende Modell soll den Betrachter in die Lage versetzen, die im Rahmen der Studienverwaltungsprozesse verarbeiteten Entitäten zu überblicken. Wie bereits aus dieser Formulierung deutlich wird, ist man an aktiven Fähigkeiten der Entitäten nicht interessiert, sondern nur an den zuzuordnenden Informationen und deren Zusammenhängen. Beispielsweise ist es für das Datenmodell unerheblich, ob ein Student eine Methode `schreibeKlausur()` besitzt oder nicht. Aus Sicht der Studierendenverwaltung reduziert sich der Student auf Eigenschaften wie seine Matrikelnummer und seine Beziehungen zu Lehrveranstaltungen. Deshalb sind Klassendiagramme für das zu erstellende Datenmodell nicht so gut geeignet wie ERDs. Es hat sich jedoch herausgestellt, dass auch in einem solchen Datenmodell zur Verbesserung der Übersichtlichkeit beispielsweise Generalisierungen sehr hilfreich sein können. Aus diesem Grund wurde die Notation für Enhanced Entity-Relationship Diagrams (EERD) nach [4] benutzt, die solche zusätzlichen Darstellungsmittel anbietet.

Rollenmodell Zur Darstellung von Rollen gibt es verschiedene Möglichkeiten. In den meisten Fällen wird im Zusammenhang mit Prozessmodellen ein Organigramm erstellt, das die Unternehmenshierarchie abbildet. Dabei kann man bei der Modellierung verschiedene Abstraktionsebenen auswählen und beispielsweise von einzelnen Unternehmenszweigen über Abteilungen und Teams bis hin zu detaillierten Stellen- bzw. Rollenbeschreibungen verfeinern. Da es sich bei der Studierendenverwaltung jedoch nur um einen Teil des gesamten Unternehmens handelt, der zudem eine sehr flache Hierarchisierung aufweist, wurde von der Verwendung von Organigrammen Abstand genommen. Stattdessen wurde eine zweite Möglichkeit, die UML Use Case Diagramme verwendet. Use Case Diagramme können unter anderem die Beteiligung von Rollen an einzelnen Prozessen bzw. Tätigkeiten darstellen und eignen sich daher, um diesen Aspekt einfacher und schneller zu überblicken, als es mit Hilfe des textuellen Prozesskatalogs möglich wäre.

Ablauf-Modell Hinsichtlich des Workflow-Modells gab es die Vorgabe, Yet Another Workflow Language (YAWL) [5, 6] zu verwenden. Aus diesem Grund wurden keine anderen Modellierungssprachen in Betracht gezogen. Es wurde jedoch zur Skizzierung der Abläufe in erster Instanz Gebrauch von den in YAWL vorhandenen Workflow-Pattern [7, 8] gemacht, ohne detailliert auf die Restrik-

tionen durch YAWL zu achten. Diese Skizzen wurden unter Berücksichtigung der entsprechenden Anforderungen nach YAWL überführt und dann weiter bearbeitet.

3 Modelle der Ist-Prozesse

Nach der Datenerhebung und den Vorüberlegungen zu den zu wählenden Techniken begann mit der Modellierung der zweite Abschnitt, der zu einer Erweiterung der in Form des Prozesskatalogs grob vorliegenden textuellen Beschreibung um graphische Darstellungen führen soll. Es werden nun die Ergebnisse der Modellierung der im Rahmen der Studienverwaltungsprozesse vorkommenden Rollen, der verarbeiteten Daten und der Abläufe vorgestellt.

3.1 Rollenmodell

Als erstes betrachtet wird das Rollen-Modell in Form von UML Use Case Diagrammen. Es wurden die Rollen der beteiligten Personen aus dem Prozesskatalog übernommen und jedem bei der Erhebung identifizierten Prozesse ein Anwendungsfall zugeordnet. Dann wurden die Rollen jeweils mit den Anwendungsfällen verknüpft, an denen sie beteiligt sind. Als zentrale Rollen können so die der StudentenverwalterIn und die der Lehrpersonen identifiziert werden. Da speziell diese beiden Rollen an sehr vielen Anwendungsfällen beteiligt sind, wurde zur Erhöhung der Übersichtlichkeit für die Use Case Diagramme eine Gruppierung der Prozesse vorgenommen, die sich in ähnlicher Form auch in den späteren Abläufen wiederfinden wird, wobei jede Gruppe in einem separaten Diagramm dargestellt ist:

- allgemeine Verwaltungstätigkeiten (Abb. 10 in Anhang 6.2)
- Planung und Vorbereitung von Lehrveranstaltungen (Abb. 11 in Anhang 6.2)
- Nachbereitung von Lehrveranstaltungen (Abb. 12 in Anhang 6.2)

3.2 Datenmodell

Als nächster Aspekt wird das Datenmodell durch ein EERD repräsentiert (siehe Abbildung 13 in Anhang 6.3). Zur Erstellung des EERDs wurden die verarbeiteten Daten aus dem Prozesskatalog ausgewertet. Dabei fiel auf, dass speziell für das Erkennen einiger Abhängigkeiten zwischen Entitäten nicht nur die direkt erkennbaren Daten, sondern auch die Ablaufbeschreibungen aus dem Prozesskatalog betrachtet werden müssen. Weiterhin waren einige Entitäten zunächst nicht als Datenelement erkennbar bzw. modellierbar. Ein Beispiel ist die Entität „Raumbelegung“, die in Abbildung 1 als Ausschnitt des EERDs zu sehen ist. Hierbei handelt es sich um eine Entität, die aus zwei weiteren Entitäten und deren Relation zueinander besteht. Der Rahmen um die Relation `findet statt in` und die beiden Entitäten `Veranstaltungsteil` und `Raum` stellt dabei als ein

Mittel der Erweiterung normaler ERDs die Hülle der zusammengesetzten Entität `Raumbelegung` dar. Diese Erweiterung wird auch Objektifizierung genannt. Dieser Sachverhalt könnte weniger übersichtlich auch durch zusätzliche Relationen mit der Notation des ERD dargestellt werden, aber bei mehrfachem Gebrauch wird das gesamte ERD dadurch deutlich schlechter zu erfassen und einige Informationen würden redundant dargestellt werden. Im vorliegenden Fall wäre es auch möglich, einfach nur die Relation `findet statt in` bereits als Raumbelegung zu betrachten. Dabei ginge jedoch der eigenständige Charakter verloren, den die Entität im Rahmen der Studienverwaltungsprozesse besitzt. Ähnlich ist es im Fall der `Belegung` (zu sehen in Abbildung 2), wo die zusammengesetzte Entität auch ein Attribut (`gewählter Themenkomplex`) besitzt.

Eine weitere hilfreiche Erweiterung ist die Generalisierung. Sie wird im EERD

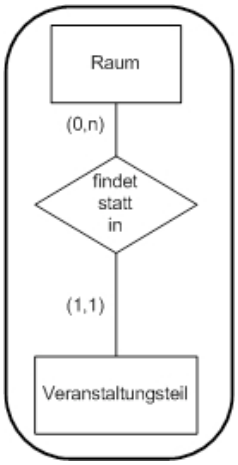


Abbildung 1. EERD: Zusammengefasste Entität Raumbelegung

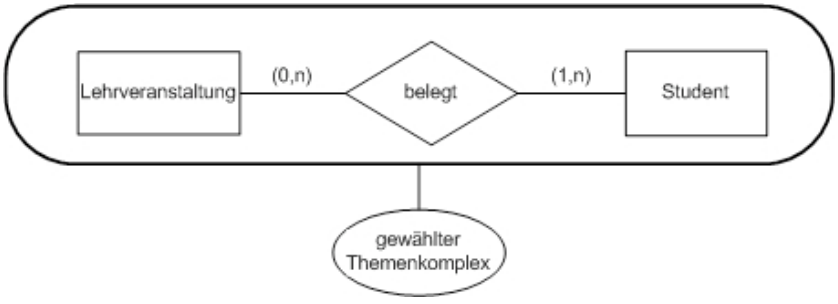


Abbildung 2. EERD: Zusammengefasste Entität Belegung

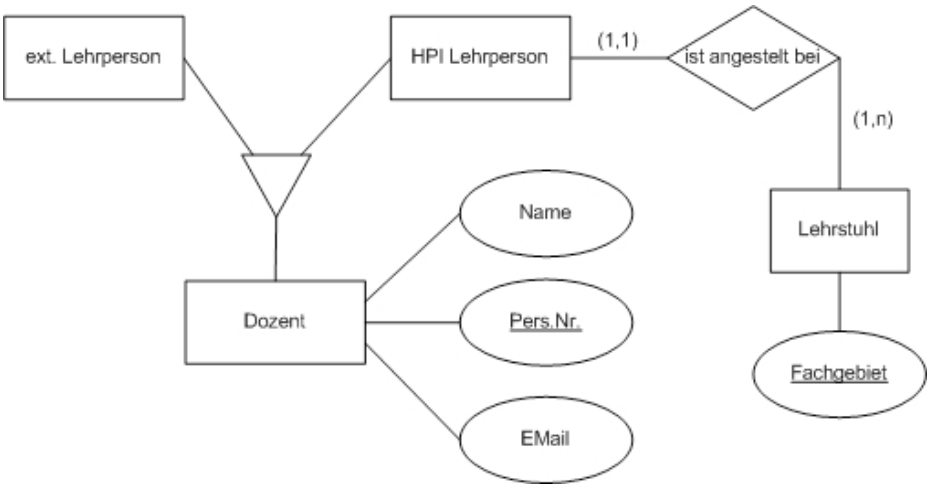


Abbildung 3. EERD: Generalisierung Dozent

bei der Entität `Dozent` verwendet. Die Entität `Dozent` besteht, wie im Ausschnitt des EERDs in Abbildung 3 dargestellt, nämlich sowohl aus HPI-eigenen Lehrpersonen, die zu einem der Lehrstühle gehören, und externen Lehrpersonen. Diese Unterscheidung ist für die Betrachtung der Studienverwaltungsprozesse wichtig, was man an den Ablaufbeschreibungen erkennen kann.
Abschließend muss erwähnt werden, dass zur Gewährleistung der Übersichtlichkeit die Attribute der verschiedenen Entitäten nicht immer vollständig im EERD modelliert wurden. Zu einer `Lehrveranstaltung` beispielsweise wird auch die Sprache erfasst, in der sie gehalten wird, ebenso wie die Information, ob die Lehrveranstaltung eine Kern-Lehrveranstaltung, eine Wahlpflicht-Lehrveranstaltung oder eine Master-Lehrveranstaltung ist. Diese Daten sind im Prozesskatalog vorhanden und finden sich in konkreten Formularen wieder, die jedoch einer detaillierteren Sichtebene zuzuordnen sind. Bei einem ERD bzw. EERD sind primär die Entitäten und ihre Relationen von Bedeutung.

3.3 Ablaufmodell

Der Schwerpunkt dieser Arbeit liegt bei der Modellierung der dynamischen Aspekte der Studienverwaltungsprozesse. Die Modellierung des Ablaufs der Ist-Prozesse erfolgte auf Grundlage der im Rahmen der Prozessdatenerhebung gesammelten textuellen Ablaufbeschreibungen in YAWL. Das Modell wurde dabei in drei Schichten unterteilt:

- die Gesamtübersicht eines Semesters als Abstraktion von den erkannten Hauptprozessen
- die Ablaufübersicht der Hauptprozesse

– die Detailsicht, in der der Ablauf einzelner Hauptprozesse ausführlich beschrieben wird

Gesamtübersicht Die Gesamtübersicht ist in Abbildung 14 im Anhang 6.4 zu sehen. Sie zeigt die Studienverwaltungsprozesse aus der Vogelperspektive. Zwei Timer-Events markieren jeweils den Beginn und das Ende eines Semesters. Darin eingeschlossen sind die allgemeinen Verwaltungsprozesse, die immer fortlaufend stattfinden. Bereits vor Beginn eines Semesters beginnt die Abwicklung der HPI-Veranstaltungen, die bis nach Semesterende andauert.

Ablaufübersicht In Abbildung 15 aus Anhang 6.4 wird die Ablaufübersicht der zur Abwicklung der HPI-Veranstaltungen gehörenden Prozesse dargestellt. Als erstes findet die Basis-Lehrveranstaltungsplanung statt, woran sich die Basis-Raumplanung anschließt. Parallel können aber weitere Lehrveranstaltungen nachgeplant werden. Nach der Basis-Raumplanung sind Raumbuchungen jederzeit möglich. Sie werden als Event verarbeitet, an das sich eine Konfliktprüfung zur Vermeidung von Überbuchungen anschließt. Jeweils wöchentlich werden die Raumbelegungspläne ausgehängt. Die Aktivierung erfolgt hier durch ein Timer-Event. Im Anschluss an die Raumplanung werden auch die Leistungserfassungsprozesse der einzelnen Lehrveranstaltungen veröffentlicht, was durch Verwendung des Multiple Instances Pattern von YAWL modelliert wurde. Zusätzlich hätte hier eine Parametrisierung der minimalen bzw. maximalen Anzahl von Instanzen in das Diagramm eingetragen werden können, worauf aber aus Gründen der Übersichtlichkeit verzichtet wurde. Es wird für jede angebotene Lehrveranstaltung genau ein Leistungserfassungsprozess veröffentlicht. Selbiges gilt für die danach parallel zur Klausurvorplanung stattfindende Belegungsbearbeitung - für jede Lehrveranstaltung wird dort eine Instanz gestartet und im Anschluss daran aus den verbuchten Belegungen eine Teilnehmerliste erstellt. Wenn dieser Prozess abgeschlossen ist, werden keine weiteren Lehrveranstaltungen mehr eingeplant. Die Beendigung der Endlosschleife für die Nachplanung von Lehrveranstaltungen wird durch Verwendung des Cancel Pattern realisiert. Im Anschluss daran kann die Klausurfeinplanung durchgeführt werden. Als nächstes findet dann die Begleitung der Leistungserfassungsprozesse statt. Diese Prozesskette wurde im Gesamtablauf zusammengefasst, weil mehrere Instanzen davon gestartet werden müssen. Die Begleitung beginnt mit der Bearbeitung von Krankmeldungen, die Einfluss auf den Leistungserfassungsprozess haben, sowie der Bearbeitung von Belegungsrücknahmen aus wichtigem Grund. Tagesaktuell werden dann die Teilnehmerlisten erstellt. Ist die Leistungserfassung abgeschlossen, werden die ermittelten Noten bzw. Leistungspunkte verbucht und eine Gesamtnotenübersicht für die jeweilige Veranstaltung erstellt. Im Fall von Nach- bzw. Ersatzprüfungen werden die dadurch ermittelten Noteninformationen nachträglich verbucht und die Notenübersicht aktualisiert. Die Leistungserfassungsbegleitung ist damit abgeschlossen und es können nun Leistungsnachweise für die Studenten sowie Auswertungen für die Lehrpersonen erstellt werden. Danach ist die Abwicklung der HPI-Lehrveranstaltungen eines Semesters abgeschlossen.

Detailsicht Als Beispiel einer detaillierteren Beschreibung eines bei der Erhebung identifizierten Prozesses dient die Basis-Lehrveranstaltungsplanung. Das entsprechende Diagramm ist im Anhang 6.4 in Abbildung 16 zu sehen. Dort werden zunächst die für das Semester vorgesehenen Kern-Lehrveranstaltungen ermittelt und jeweils einem Lehrstuhl zugeordnet. Sollten nicht alle zugeordnet werden können, müssen externe Lehrpersonen eingebunden werden. Dann kann das Personalverzeichnis an die Universitätsverwaltung übermittelt werden. Anschließend werden bei den einzelnen Lehrpersonen die Daten zu den angebotenen Lehrveranstaltungen mit Hilfe eines Formulars per Mail abgefragt. Dabei wird eine Frist gesetzt, die in Form eines Timers modelliert wurde. Wird das Formular ausgefüllt zurück geschickt, wird es ausgewertet und die jeweilige Veranstaltung ins Vorlesungsverzeichnis aufgenommen. Werden regelmäßig stattfindende Lehrveranstaltungen nicht innerhalb der festgelegten Frist gemeldet, wird zunächst der Stand vom vorhergehenden Studienjahr übernommen. Diese Unterscheidung wurde auf Grund der Event-Aktivierung nicht durch ein OR-Split sondern durch ein AND-Split und nachfolgendes Abbrechen des jeweils anderen Pfades bei Eintreffen eines Events modelliert. Als nächstes findet eine fristgebundene Übermittlung des Vorlesungsverzeichnisses an die Universitätsverwaltung statt. Danach sind noch Nachmeldungen oder Änderungen zu Lehrveranstaltungsangaben möglich. Das entsprechend korrigierte Vorlesungsverzeichnis wird dann wieder fristgebunden an die Universitätsverwaltung übermittelt und die bereits feststehenden Lehrveranstaltungen werden veröffentlicht. Trotzdem können weiterhin Lehrveranstaltungen eingeplant werden. Parallel dazu ist die Einpflege der Lehrinhalte und anderer Informationen zu einer Lehrveranstaltung im Veröffentlichungsbereich im Web möglich. Bis zu einem bestimmten Zeitpunkt kann sich diese Abfolge (Veröffentlichung, Einpflege der Inhalte und ggf. Einplanung zusätzlicher Lehrveranstaltungen) wiederholen. Dann werden alle für das Semester geplanten Lehrveranstaltungen in eine Liste eingetragen, woraufhin die Basis-Lehrveranstaltungsplanung endet.

Probleme bei der Modellierung Obwohl sich YAWL sehr gut zur Beschreibung von Abläufen eignet, gab es dennoch einige Schwierigkeiten bei der Umsetzung der Studienverwaltungsprozesse.
So ist eine explizite Modellierung der Zeitpunkte für Semesteranfang und -ende auf der Ebene der Ablaufübersicht nicht möglich, da einige Prozesse wie die Veröffentlichung der Leistungserfassungsprozesse jeweils über diese Grenzen hinweg verlaufen. Einige Instanzen davon werden vor Semesterbeginn ausgeführt, einige erst im laufenden Semester.
Ein ähnliches Problem ergibt sich bei der Modellierung unterschiedlicher Zeitabläufe für multiple Instanzen. Es besteht die Möglichkeit, dass einige Lehrveranstaltungen, z.B. Seminare, sehr frühe Belegungsfristen haben, während bei anderen auf Grund einer späten Leistungserfassung auch späte Belegungsfristen gelten, die dann auch hinter dem Ende der Leistungserfassung anderer Lehrveranstaltungen liegen können. Dies kann durch das vorliegende Modell so nicht dargestellt werden, da in YAWL der Kontrollfluss erst dann fortgesetzt wird, wenn

alle Instanzen eines Prozesses beendet sind. Eine Möglichkeit zur Umgehung dieses Problems wäre die Verwendung eines zusammenfassenden Multiple Instance Prozesses, der die zuvor einzeln aufgeführten Prozesse enthält. Innerhalb dieser Abstraktion wäre der zeitliche Ablauf zwischen einzelnen Lehrveranstaltungen somit entkoppelt, die Reihenfolge der Teilprozesse für jede Lehrveranstaltung bliebe aber korrekt erhalten. Hinderlich für die Umsetzung sind jedoch die Synchronisationen mit den Prozessen, die nicht in die Abstraktion integrierbar sind. So müsste beispielsweise die Annahme getroffen werden, dass die Klausurvorplanung in jedem Fall rechtzeitig aktiviert wird und zeitlich eher punktuellen Charakter hat, so dass auf die explizite Synchronisation verzichtet werden kann, und außerdem nur Seminare nachgemeldet werden. Ein entsprechend angepasstes Diagramm zeigt Abbildung 17 in Anhang 6.4. Es ergibt sich hier auch die Möglichkeit, die Klausurfeinplanung z.B. für Seminare nicht auszuführen. Insgesamt ist dieser Ansatz aber unpassend, da die Annahmen in YAWL nicht explizit abbildbar sind und ein Nichtzutreffen der Annahmen beispielsweise zur Folge haben kann, dass Lehrveranstaltungen nachgemeldet werden, obwohl die Leistungserfassungsbegleitung nicht mehr ausführbar ist.
Das nächste Problem taucht bei der Verwendung des Cancel Patterns auf. Abbildung 4 zeigt als Beispiel den Ausschnitt aus dem Ablauf der Abwicklung von HPI-Lehrveranstaltungen, bei dem die potentiell endlose Schleife zur Nachmeldung von Lehrveranstaltungen durch Erreichen eines bestimmten Punktes im Workflow abgebrochen wird. Grundsätzlich sind Endlosschleifen mit hartem Abbruch unschön und nach Möglichkeit zu vermeiden. Deshalb wurden verschiedene alternative Modellierungsmöglichkeiten ausprobiert, die nachfolgend vorgestellt werden.
Abbildungen 5 bis 8 zeigen Ansätze, die das Problem umgehen sollen. Im Ansatz in Abbildung 5 wird der Kontrollfluss nach der Entscheidung, ob weiter auf Nachmeldungen gewartet wird oder nicht, zum AND-Join geleitet. Das Problem hier ist der Cancel-Rahmen - wird gecancelt, während die Marke noch im Rahmen eingeschlossen ist, kann der AND-Join dahinter nicht schalten, weil der entsprechende Eingang keine Marke bekommen kann. Das Ergebnis wäre ein Deadlock.
In Abbildung 6 wird versucht, diesen Effekt zu vermeiden, indem das Cancel-Pattern gar nicht verwendet wird. Stattdessen wird nach der Nachplanung ein Zustand erreicht, in dem der AND-Join immer schalten könnte. Allerdings kann das Event nie verarbeitet werden, weil zu keinem Zeitpunkt alle Eingangsstellen eine Marke enthalten können. Auch hier ist ein Deadlock die Folge.
Im Ansatz in Abbildung 7 ist dieser Fehler behoben. Allerdings kann dann die Event-Schleife beliebig oft durchlaufen werden. Beliebig oft bedeutet unter Umständen auch, dass die Schleife deutlich vor oder nach Terminierung der anderen Prozesse, die für das Schalten des AND-Joins benötigt werden, verlassen wird, was nicht gewollt ist.
Man braucht also eine Signalisierung zwischen den beiden parallelen Kontrollflüssen, die das Terminieren der anderen zu synchronisierenden Prozesse anzeigt. Im fünften Ansatz, der in Abbildung 8 dargestellt ist, wird dies mittels eines

Events dargestellt. Dieses Event soll bei Abschluss des anderen Kontrollflusses aktiviert werden, wodurch ein vorzeitiges Verlassen der Schleife ausgeschlossen wird. Ein Zwang zum Verlassen der Schleife wird dadurch aber nicht bewirkt. Ohne Verwendung des Cancel-Patterns scheint die Abbildung des Sachverhaltes nicht möglich zu sein. Damit bleibt nur der harte Abbruch der Schleife aus dem ersten Ansatz als Lösung übrig.

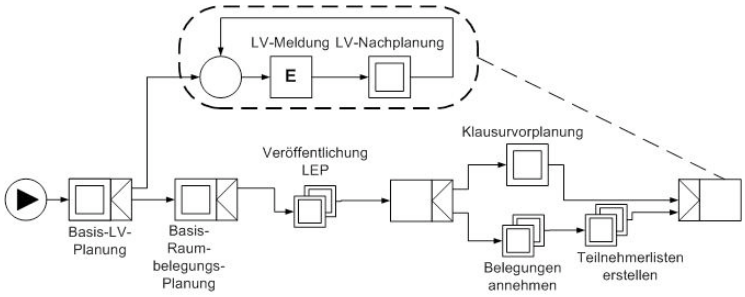


Abbildung 4. 1. Ansatz Problem „Sauberer Abbruch“ in YAWL

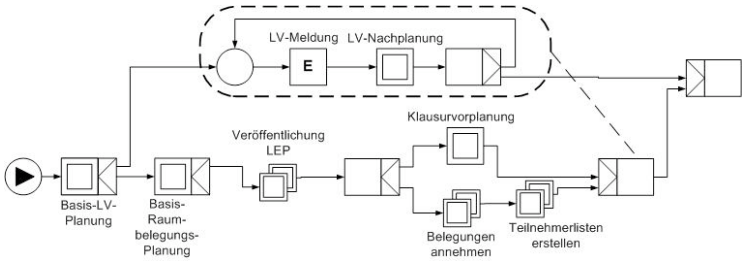


Abbildung 5. 2. Ansatz Problem „Sauberer Abbruch“ in YAWL

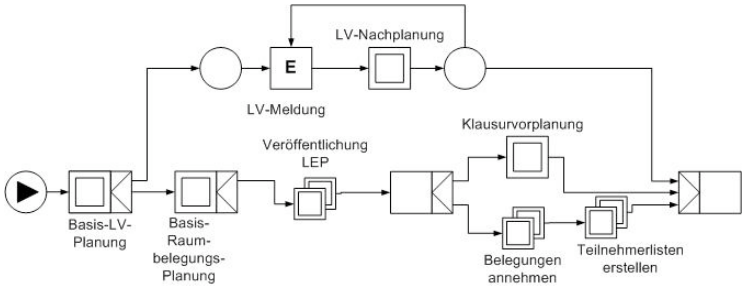


Abbildung 6. 3. Ansatz Problem „Sauberer Abbruch“ in YAWL

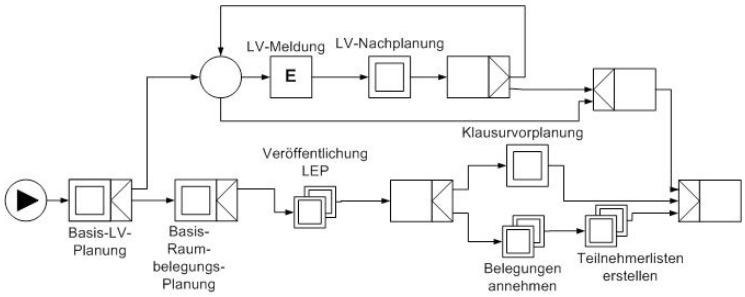


Abbildung 7. 4. Ansatz Problem „Sauberer Abbruch“ in YAWL

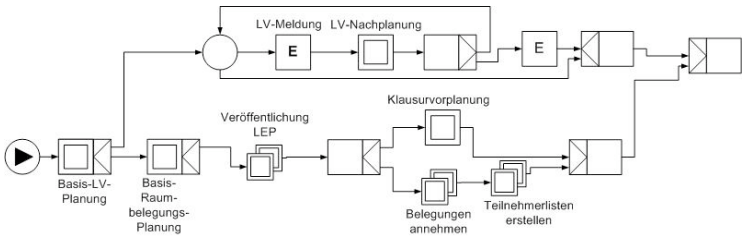


Abbildung 8. 5. Ansatz Problem „Sauberer Abbruch“ in YAWL

4 Entwicklung möglicher Soll-Prozesse

Im Anschluss an die Modellierung der Ist-Prozesse wurde nach Optimierungspotentialen mit dem Ziel gesucht, diese anhand von möglichen Soll-Prozess-Modellen zu veranschaulichen. Hilfreich hierfür waren die in Kapitel 2.2 beschriebenen Erkenntnisse zu dem Vorgehen bei PQM-Systemen. Mit Hilfe der dort verwendeten Methoden wurden sowohl die einzelnen Prozesse als auch der Gesamtprozess untersucht. Zur Beantwortung der Fragen nach 6 W und 3 Mu, die zu den Prozessen gestellt werden konnten, dienten einerseits die Erkenntnisse aus der Datenerhebungsphase, andererseits auch zusätzliche Informationsquellen, wie beispielsweise Prüfungs- und Studienordnung [9, 10] zu den am HPI angebotenen Studiengängen, die bestimmte Rahmenbedingungen für die Prozesse vorgeben.

4.1 Ergebnis der Analyse des Ist-Prozesses

Da Methoden aus dem Bereich des PQM verwendet werden, muss man zur Durchführung der Analyse auch die entsprechenden Voraussetzungen dafür beachten. Neben der Datenbasis sind hier auch die Qualitätsziele entscheidend. Für diese Arbeit spielt die Qualitätsorientierung jedoch keine Rolle. Deshalb wurde die Annahme getroffen, dass es erklärtes Ziel der Analyse der Studienverwaltungsprozesse ist, den Verwaltungsaufwand in der Studienverwaltung zu reduzieren.
Die Analyse der Abläufe einzelner Prozesse brachte keine nennenswerten Optimierungsmöglichkeiten zum Vorschein. Von außen bestimmte Anforderungen an einige Prozesse, wie die Setzung von Fristen durch die Universitätsverwaltung zur Lehrveranstaltungsplanung, oder Vorgaben zur Rechtsverbindlichkeit der Belegungsverwaltung, die sich auch auf Aspekte wie die Klausurplanung auswirken, lassen für alternative Abläufe keinen Spielraum. Die Abfolge einzelner Prozesse kann auch nicht vorteilhaft verändert werden, da logische Abhängigkeiten bzw. Datenabhängigkeiten zwischen den Prozessen vorhanden sind, die eine Umordnung verhindern. So besteht die einzige Möglichkeit Optimierungspotentiale zu finden darin, nach Abweichungen von Vorgaben an den Ist-Prozess zu schauen, die zu unnötigen Mehrbelastungen führen kann, oder nach (Teil-)Prozessen, die gemäß dieser Vorgaben auch verzichtbar sind, weil sie nicht verlangte Leistungen erbringen.
Bezüglich der ersten Kategorie wurde die Erstellung von Teilnehmerlisten durch Lehrpersonen als Verschwendung ermittelt. Es wird unnötiger Verwaltungsaufwand verursacht: Die Belegung muss in jedem Fall unabhängig von einer vorliegenden Teilnehmerliste aus Gründen der Rechtsverbindlichkeit von der Studierendenverwaltung vorgenommen werden. Das Verwaltungssystem ist in der Lage, mit vergleichsweise geringem Aufwand Teilnehmerlisten zu erstellen. Die Anlage und doppelte Verwaltung an anderer Stelle bedeuten damit zusätzlichen Aufwand. Gefunden wurde diese Möglichkeit durch die Fragengruppen `Wer` und `Wann`: Es gibt Abweichungen zwischen „Wer tut es?“ und „Wer sollte es eigentlich tun?“, sowie zwischen „Wann wird es getan?“ und „Wann sollte es eigentlich

getan werden?“, jeweils bezogen auf die Teilnehmerlistenerstellung.
In die zweite Kategorie eingeordnet werden kann die an verschiedenen Stellen auftauchende Möglichkeit zur Nachplanung von Lehrveranstaltungen. Zweifelsfrei ist die Nachplanung einer Lehrveranstaltung aufwendiger, als wäre diese bereits zusammen mit den anderen Lehrveranstaltungen eingeplant worden. Diese Nachplanung zieht beispielsweise zusätzliche Belegungsvorgänge nach sich. Unter Umständen kann es dadurch zu einer komplizierteren Raum- und Klausurplanung kommen, weil im Gegensatz zur Basisplanung, wo Zeitslots relativ flexibel und ohne weit reichende Konsequenzen verschoben werden können, im späteren Verlauf nur wenig Spielraum vorhanden ist und Änderungen an bereits laufenden Veranstaltungen immer mit zusätzlichem Aufwand verbunden sind. Diese Möglichkeit wurde durch die Fragengruppe `Wann` ermittelt. Die Alternative wird hier gefunden durch „Wann wird es getan?“ und „Wann sollte es noch getan werden können?“ bezüglich der Nachmeldung von Lehrveranstaltung bzw. zwischen „Wann wird es getan?“ und „Wann sollte es getan werden?“ bezogen auf die Einplanung von Lehrveranstaltungen insgesamt.
Die gefundenen Optimierungsansätze sind streitbar, da sie auf die Einsparung von Verwaltungsaufwand abzielen. Es ist aber nicht gesagt, dass das Qualitätsziel die unbedingte Minimierung des Verwaltungsaufwands ist. Statt dessen könnte beispielsweise auch größtmögliche Flexibilität beim Anbieten von Lehrveranstaltungen im Vordergrund stehen. Dann wäre das Weglassen der Möglichkeiten zur Nachplanung kein sinnvoller Ansatz. Bei anderen Zielfestlegungen könnten sich durchaus auch andere Ansätze finden lassen.

4.2 Soll-Modell

Entsprechend den Analyse-Ergebnissen wird nun der in den Abbildungen 18 und 19 dargestellte Soll-Ablauf als eine Möglichkeit zur Optimierung vorgestellt. Die Meldung der Lehrveranstaltungen wird nur bis zur Übermittlung des endgültigen Vorlesungsverzeichnisses an die Universitätsverwaltung ermöglicht. Alle nachfolgenden Prozesse haben dann volle Planungssicherheit und es gibt keinen zusätzlichen Aufwand durch eine Nachplanung von Lehrveranstaltungen. Die Veränderung bei der Teilnehmerlistenerstellung ist nicht sichtbar, da dieser Prozess nicht detailliert modelliert wurde.

5 Zusammenfassung der Ergebnisse

In den vorhergehenden Kapiteln wurden zunächst die Vorüberlegungen zur Modellierung der Studienverwaltungsprozesse der HPI und der grobe Ablauf bis zur Entstehung der verschiedenen Modelle erläutert. Anschließend wurden die Ist-Prozesse in Form von Rollen-, Daten- und Ablaufmodellen vorgestellt. Darauf aufbauend wurden diese Prozesse dann analysiert und die daraus abgeleiteten Soll-Modelle beschrieben. Nun werden die Ergebnisse und Einschätzungen zu den einzelnen Teilen vorgestellt.

5.1 Vorgehensweise

Die gewählte Vorgehensweise über Datenerhebung, Entwurfsmodellierung, Überprüfung der Modelle und anschließende Analyse hat sich als gangbarer Weg erwiesen. Die Erkenntnisse aus dem Bereich des PQM waren hilfreich, so dass sich der Aufwand der Betrachtung gelohnt hat. Allerdings fällt auf, dass das Fehlen des Qualitätsbezugs auch zu Problemen führte, die ohne diese Orientierung vermutlich nicht aufgetreten wären. Beispielsweise ist es sehr schwierig gewesen, Optimierungsmöglichkeiten zu finden, weil keine echten Qualitätsziele für die Prozesse vorgegeben waren. Weiterhin setzt die Analyse auch Messergebnisse zu den Prozessen voraus, die es im konkreten Fall nicht gab, weil Messungen weder gefordert noch möglich waren.

5.2 Modellierungstechnik

Die Nutzung von UML Use Case Diagrammen zur Darstellung der Beziehung von Rollen zu den im Rahmen der Erhebung identifizierten Prozessen verschafft einen sehr guten Überblick zu diesem Aspekt und bietet gegenüber der textuellen Beschreibung im Prozesskatalog den Vorteil, sehr schnell und übersichtlich erfassbar zu sein. Bei umfassenderer Modellierung wäre die zusätzliche Verwendung von FMC Aufbaubildern hilfreich gewesen. Mit ihnen hätte man die Möglichkeit, auch Datenabhängigkeiten und Kommunikationswege zwischen Rollen darzustellen und den jeweiligen Ort der Verarbeitung von bestimmten Daten zu modellieren. Ein Beispiel ist in Abbildung 9 zu sehen. Es beschreibt den Aufbau für die Prozesse zur Nachbereitung von Lehrveranstaltungen. Damit hätte man auch eine Brücke zum Datenmodell geschlagen. Für das Datenmodell wurde ein EERD verwendet. Die Notation eignete sich besonders für die detaillierte Modellierung der Zusammenhänge zwischen den im Prozess verarbeiteten Daten. Leider lassen sich diese in YAWL, der für die Ablaufbeschreibung verwendeten Modellsprache, nicht einbeziehen, da Datenflüsse bzw. -abhängigkeiten nicht beschrieben werden können. Ebenso fehlt es an Möglichkeiten zur Darstellung von Inter-Prozess-Kommunikation. Dafür ist eine Modellierung von Ablaufbeschreibungen mit YAWL durch die Verwendung von Pattern sehr einfach, wobei die Mächtigkeit der zu Grunde liegenden Petri-Netz-Notation weitestgehend erhalten bleibt. Dadurch werden auch gewisse Aussagen, beispielsweise zur Ablaufsicherheit des Workflows, formal beweisbar. [5]

5.3 Ergebnisse der Prozess-Modellierung

Insgesamt bilden die Modelle die tatsächlichen Prozesse gut ab. Das hat auch die Diskussionsrunde mit den Interview-Partnern bestätigt. Probleme bei der Modellierung gab es besonders bei Ausnahmen wie den Nachmeldungen.
Die Suche nach Optimierungspotentialen wurde durch äußere Bedingungen wie Termine bzw. Fristen, Studien- und Prüfungsordnung oder logische Zwänge bei der Abfolge von Prozessen sehr eingeschränkt. Durch Vorgabe von Zielen zur Optimierung wurden dann zwar Möglichkeiten gefunden, doch bleiben diese durchaus streitbar, da die Ziele in der Realität nicht in dieser Art definiert sind.

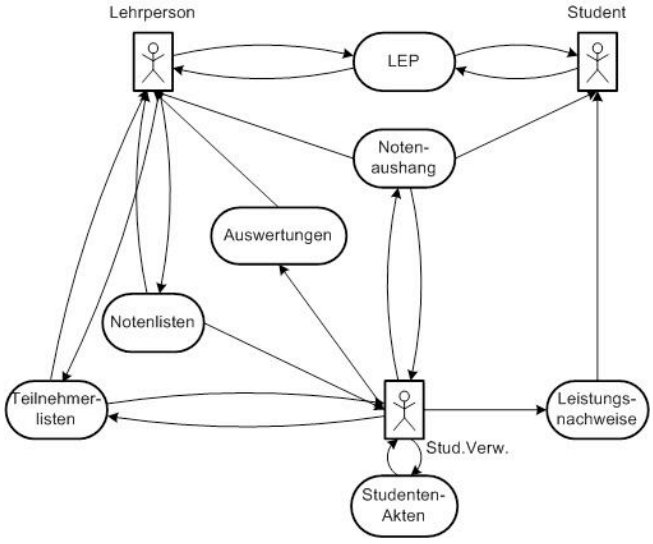


Abbildung 9. Beispiel für ein FMC Aufbaubild

6 Anhang

6.1 Prozesskatalog

Lehrveranstaltungsplanung

Personen:	Professoren/Lehrpersonen, HPI-LV-PlanerIn, Uni-Verwaltung, HPI-GL (GF, BSc- u. MSc-Beauftragte)
Daten:	Lehrperson(en), LV-Titel, Einordnung (BSc / MSc), Anzahl LP/SWS, Sprache, Aufteilung SWS (Übung, Vorlesung), Zuordnung (Kern/Wahl/Master), Benotung (benotet, unbenotet), mögliche Themenkomplexe, Raumbedarf, Terminwünsche, Zielgruppe (z.B. 1. Fachsem. SST), Personalverzeichnis (f. VV), Vorlesungsverzeichnis, fortlaufende Liste der durchgeführten LVen
Zeiten:	ab ca. 5 Monate vor Beginn des Folgesemesters bis in dessen Vorlesungszeit hinein
Orte:	HPI Stud.Verw., Lehrstühle, GL
Zweck:	Übermittlung des LV-Verzeichnisses an die Uni-Verwaltung, Zusammenstellung des LV-Katalogs für das HPI als Basis für Raumplanung, Aushang und Belegung
Ablauf:	Ermittlung der nach Studienordnung nötigen Kern-LV, Verteilung auf Lehrstühle bzw. Anfrage an GF bezüglich externer Lehrpersonen, Verträge mit und Koordination von Lehrpersonen (paral.), Abfrage der einzelnen Lehrstühle nach LV, Aufbereitung der Daten, Übermittlung an Uni (drei Termine: PV, VV, Korrekturen) und Raumplanung. Veröffentlichung bereits feststehender LVen. Parallel zur weiteren Planung werden Lehrinhalte und Details zur Leistungserf. von den Lehrpersonen eingepflegt bzw. aktualisiert, zum Abschluss wird Liste über durchgeführte LVen aktualisiert.

Raumplanung

Personen:	HPI-Raum-PlanerIn, FM, Informatik-Raum-PlanerIn
Daten:	LV-Daten (Kapazität=Anzahl Teiln., Dauer=Anzahl Blöcke, Besonderheiten (z.B. Block-LV, bereits festliegende Zeitslots)
Zeiten:	vor Beginn der Vorlesungszeit, nach Basis-LV-Planung
Orte:	HPI-Hörsaalgebäude, Seminarräume, Rechnerpools
Zweck:	Zusammenstellung des Stundenplans inkl. Raumverteilung
Ablauf:	Auswertung der LV-Planungsdaten, Prüfung besonderer Terminwünsche, Einplanung LV über Stundenplan/Hörsaalplan, Koordination gemeinsamer Termine mit Informatik-Planung, parallel: Einplanung LV über Seminarraumplan, nach endgültigem Gesamt-Stundenplan Vergabe freier Hörsaalkapazitäten an Informatik-Planung, weitere Raumvergabe über Outlook (allgemein einsehbar, Buchung für Lehrpersonen, Stud.Verw. und FM möglich, Vorrang: LVen, Klausurzeitraum pauschal geblockt; wöchentlich freitags wiederkehrend: Ausdruck der aktuellen Raumbelegungspläne für Folgewoche und deren Verteilung

Veröffentlichung der Leistungserfassungsprozesse

Personen:	HPI-LV-PlanerIn, Stud.Verw. BSc und MSc, Lehrpersonen
Daten:	LV-Daten, Raum-Daten
Zeiten:	vor Beginn der Vorlesungszeit
Orte:	HPI BSc- und MSc-Verwaltung und -Aushänge
Zweck:	Informationen zu angebotenen LV veröffentlichen
Ablauf:	Auswertung und Aufbereitung (z.B. Festlegung vorläufiger Belegungsfristen) der LV- und Raum-Planungsdaten, Aushang und Veröffentlichung im Web/Mail

Belegungen (inkl. Rücknahme)

Personen:	BSc- und MSc-Studenten, Stud.Verw. BSc und MSc
Daten:	LV-Daten (Bezeichnung, Nummer, Themenkomplex, Anzahl LP, Fristen), Studentendaten
Zeiten:	zu Beginn der Vorl.zeit, bis Ablauf d. einzelnen Belegungsfristen
Orte:	HPI BSc- und MSc-Verwaltung
Zweck:	Erfassung der Teilnehmer von LVen (HPI-Sicht), Erfassung der belegten LVen (Stud-Sicht)
Ablauf:	Überprüfung der Studentendaten, bei Master: Überprüfung der Zulassungsvoraussetzungen (MSc-Student oder gleichwertige Qualifikation?), Einbuchung von Belegungen, Stornierung von Belegungen durch Rücknahme oder mangels Zulassung durch Dozenten bei teilnehmerbeschränkten LV, jeweils {Beleg-Erstellung, Unterschrift, Ablage}; Timer: Master Fristsetzung auf Zeitraum 4-5 Wochen nach Beginn LV; Bachelor: 2 Wochen vor Beginn des LEP, spätestens 2 Wochen vor Ende Vorles.zeit; Ausnahmen: Blocklehrveranstaltungen (=zum Zeitfenster gehörender Termin) bzw. bei Seminaren unmittelbar nach Themenvergabe (=Beginn des LEP)

Teilnehmerlistenerstellung

Personen:	Stud.Verw. BSc und MSc, Lehrpersonen
Daten:	LV-Daten (Bezeichnung/Nummer, Lehrperson), Belegungsdaten
Zeiten:	nach Belegung bei Bedarf temporäre Erstellung, unmittelbar nach Ablauf der jeweiligen Frist endgültige TN-Liste
Orte:	HPI BSc- und MSc-Verwaltung
Zweck:	Teilnehmerlisten zur übermittlung an Lehrperson (Kontroll-Möglichkeit), Grundlage für Klausurplanung (Vermeidung von Überschneidungen bei Studenten)
Ablauf:	Auswertung der Belegungsdaten, Erstellung der Listen, Weiterleitung an Lehrperson

Klausurplanung

Personen:	Stud.Verw. BSc und MSc, Lehrpersonen
Daten:	LV-Daten (Bezeichnung, Fristen, Vorl.-Zeiten), Teilnehmerlisten
Zeiten:	Beginn nach Ablauf der ersten Belegungsfristen, Ende nach Ablauf aller Belegungsfristen
Orte:	HPI BSc- und MSc-Verwaltung
Ablauf:	Auswertung der Teilnehmerlisten und Auswertung der LEP-Daten, Iteration über mögliche Termine, vorl. Planung, nach Ablauf aller Fristen endgültige Planung, ggf. Rückfragen Lehrperson, Raumbuchung, Veröffentlichung der Termine

Rücknahme Belegungen aus wichtigem Grund/Verarbeitung von Krankmeldungen

Personen:	Lehrpersonen, Stud.Verw. BSc und MSc, Studenten
Daten:	Personendaten Student, Belegungsdaten, LV-Daten
Zeiten:	bis vor Abschluss der Leistungserfassung
Orte:	HPI BSc- und MSc-Verwaltung, Lehrstühle
Ablauf:	Auswertung von Krankmeldungen oder Anträgen auf Änderung aus wichtigem Grund, ggf. Weiterleitung Prüfungsausschuß, Anpassung Teilnehmerliste Klausr und Weiterleitung Lehrperson, Entscheidung Nachprüfung oder Rücknahme Belegung

Notenerfassung

Personen:	Lehrpersonen, Stud.Verw. BSc und MSc
Daten:	LV-Daten, Studentendaten (Matr.Nr.), Noteninformationen
Zeiten:	Nach Abschluss des LEP am Ende des Semesters
Orte:	HPI BSc- und MSc-Verwaltung, Lehrstühle
Ablauf:	Abfrage der Notenlisten der einzelnen Lehrstühle, Überprüfung auf Korrektheit (Übereinstimmung Teilnehmerliste), Verbuchung; ggf. bei Nachklausur o.ä. Korrektur der Buchung

Erstellung von Leistungsnachweisen

Personen:	Studenten, Stud.Verw. BSc und MSc, Vorsitzende/r Studienausschuß SST
Daten:	Noteninformationen, Studentendaten
Zeiten:	Nach Abschluss der Notenerfassung I bzw. II
Orte:	HPI BSc- und MSc-Verwaltung, Büro Studienausschußvorsitzenden
Ablauf:	Seriendruck der einzelnen Übersichten bzw. Ausdruck gewählter Informationen auf Anfrage, Unterschrift durch wissenschaftlichen Leiter, Abholung durch Student

Erstellung von Gesamtnotenübersichten

Personen:	Stud.Verw. BSc und MSc, Lehrpersonen
Daten:	Noteninformationen
Zeiten:	Nach Abschluss der Notenerfassung I, Korrektur ggf. nach Notenerfassung II
Orte:	HPI BSc- und MSc-Verwaltung, Aushänge
Ablauf:	Aufbereitung der Noteninformationen, Veröffentlichung der Übersichten im Web/Aushänge

Auswertungen erstellen

Personen:	Lehrpersonen, Stud.Verw. BSc und MSc
Daten:	Noteninformationen, Belegungsinformationen
Zeiten:	Nach Abschluss der Notenerfassung I, ggf. Update nach Notenerfassung II
Orte:	HPI BSc- und MSc-Verwaltung
Ablauf:	Auf Anfrage der Lehrperson können dieser verschiedene Auswertungen über Belegungs- und Leistungsdaten zur Verfügung gestellt werden.

Bescheinigungen/Nachweise erstellen

Personen:	Studenten, Stud.Verw. BSc und MSc, Vorsitzende/r Studienausschuß SST
Daten:	Studentendaten (einschl. Noten- und Belegungsinfos, etc.)
Zeiten:	jederzeit
Orte:	HPI BSc- und MSc-Verwaltung, Büro d. Studienausschußvorsitzenden
Ablauf:	Auf Anfrage des Studenten können diesem verschiedene Bescheinigungen/Nachweise anhand der vorhandenen Daten erstellt werden.

Anerkennung von Fremdlehrveranstaltungen

Personen:	Studenten, Stud.Verw. BSc und MSc, Lehrpersonen, Prüfungsausschuß
Zeiten:	jederzeit
Orte:	HPI BSc- und MSc-Verwaltung
Ablauf:	Auf Antrag des Studenten können Fremd-LV anerkannt werden. Student gibt Antrag (u.a. muss dieser einen Themenkomplex enthalten) bei Stud.Verw. ab. Weiterleitung an Lehrperson/Prüfungsausschuß. Entscheidung über Anerkennung. Mitteilung des Bescheids an Studenten und ggf. Verbuchung. / Unterscheidung: einzelne LVen oder Studiengangswechsel. Studiengangswechsel: Einstufung in ein Fachsemester über Prüfungsausschuß. LVen: bei Anerkennung erfolgt eine separate Aufführung der Leistungen bis einschließlich Abschlusszeugnis. Übernahme von Fremd-BP/LP:HPI-BP/LP -> 1:1, SWS:LP -> 1:1.5

6.2 Use Case Diagramme

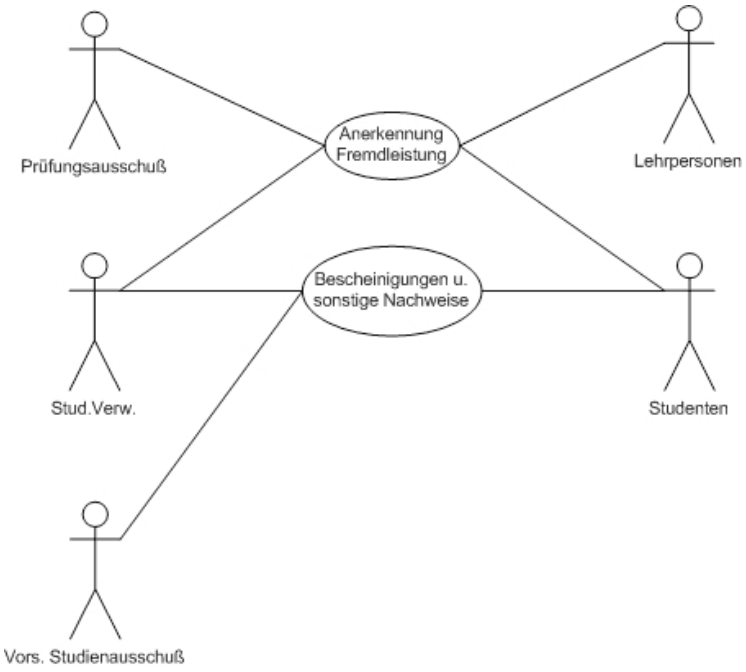


Abbildung 10. Use Cases: allgemeine Studienverwaltungsprozesse

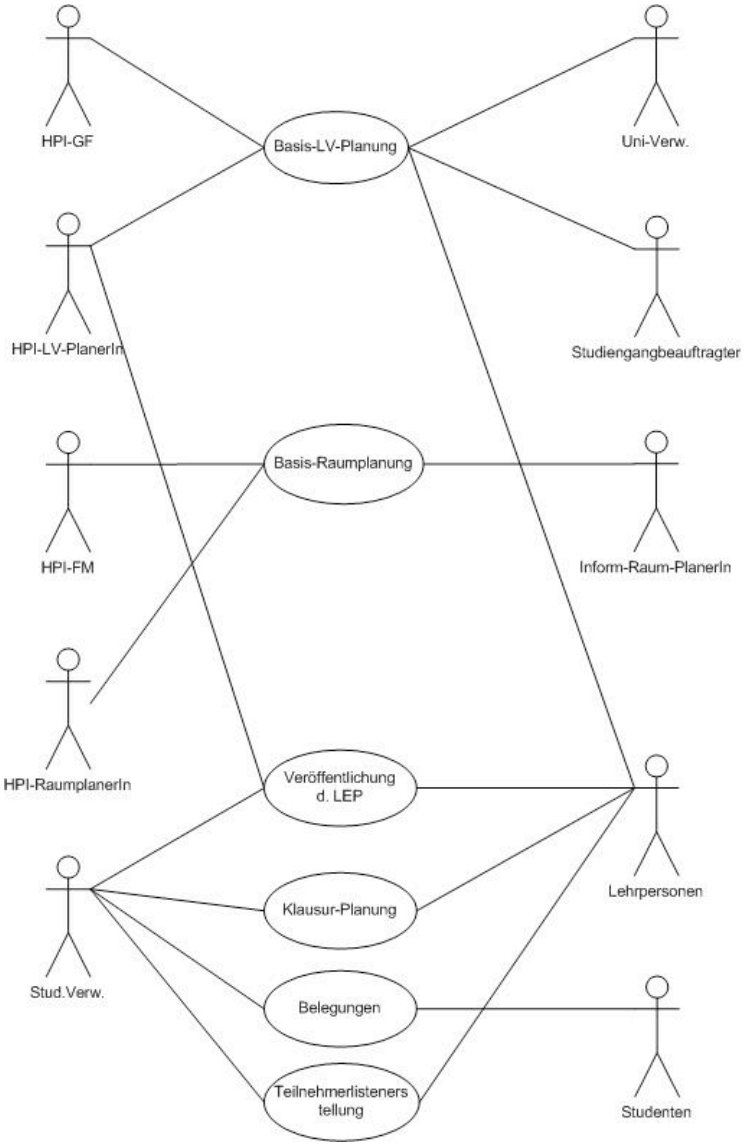


Abbildung 11. Use Cases: Lehrveranstaltungsvorbereitungsprozesse

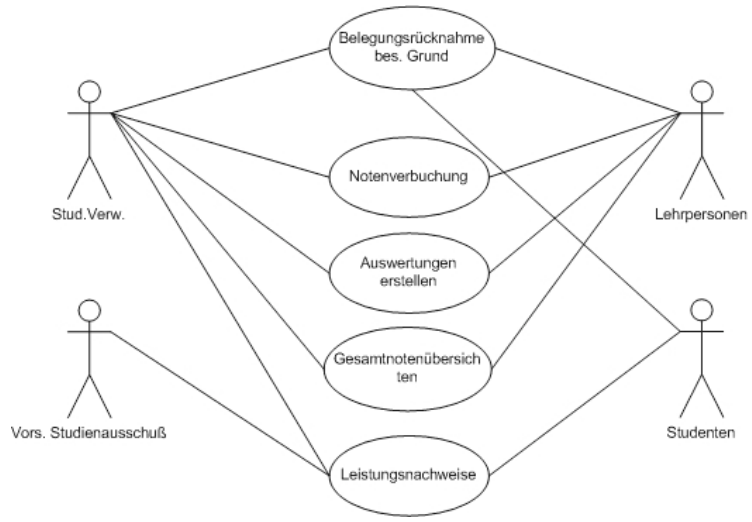


Abbildung 12. Use Cases: Lehrveranstaltungsnachbereitungsprozesse

6.3 EERD

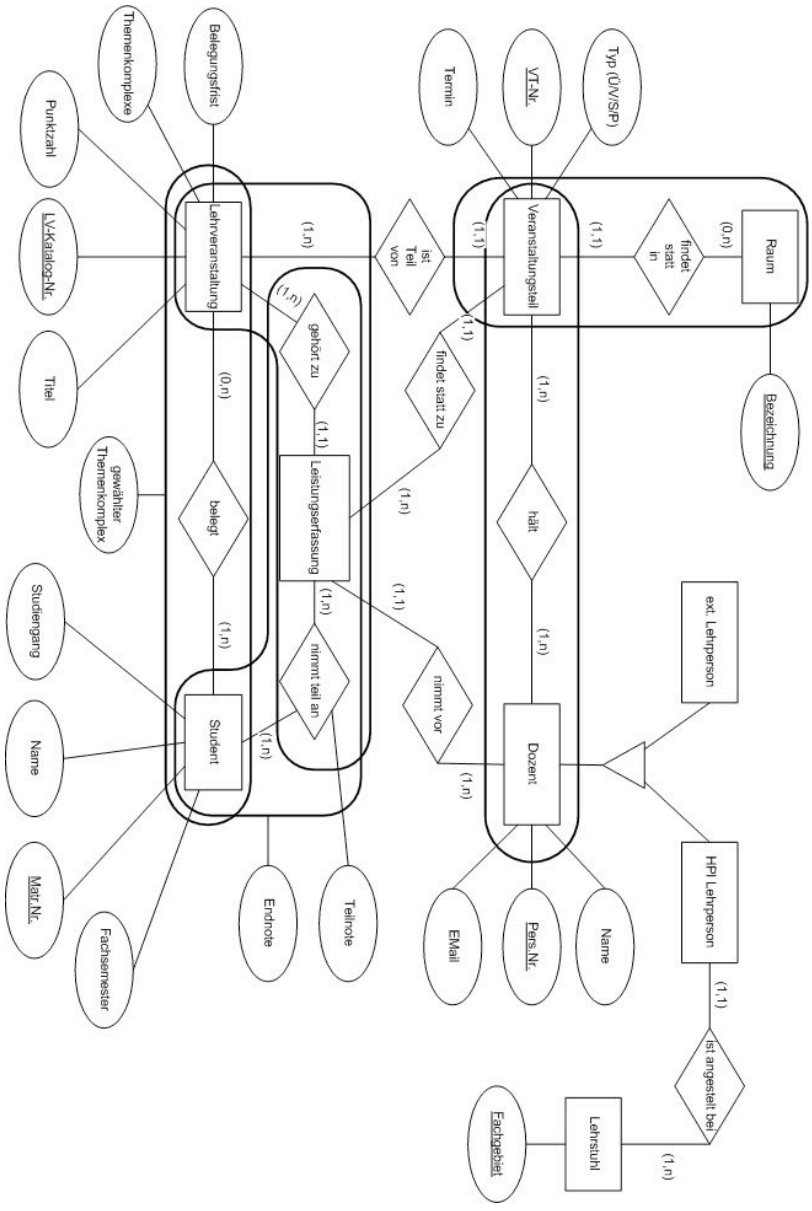


Abbildung 13. Datenmodell Studienverwaltungsprozesse

6.4 Ist-Workflow Diagramme

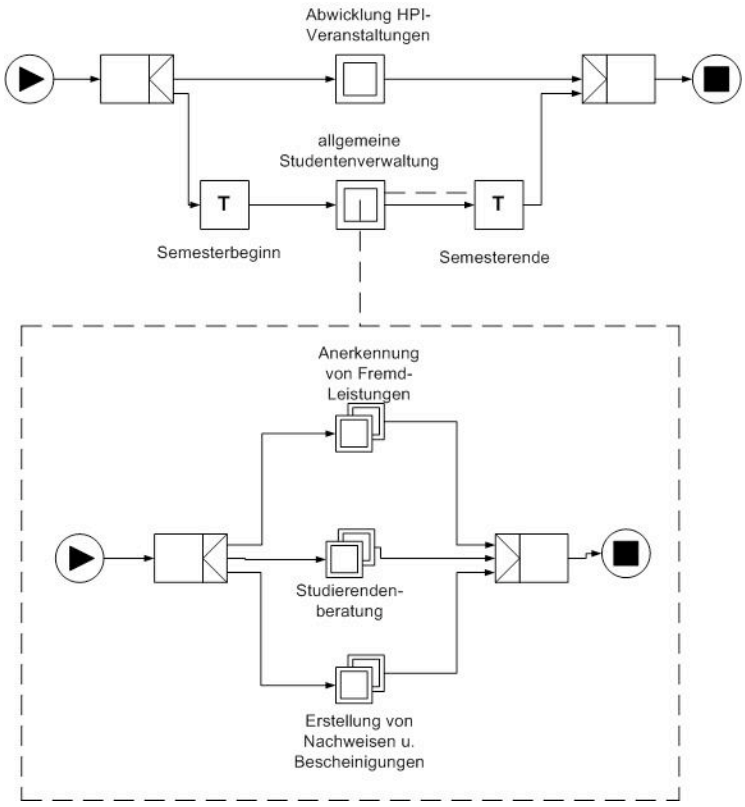


Abbildung 14. Ist-Prozess: Überblick Studienverwaltungsprozesse

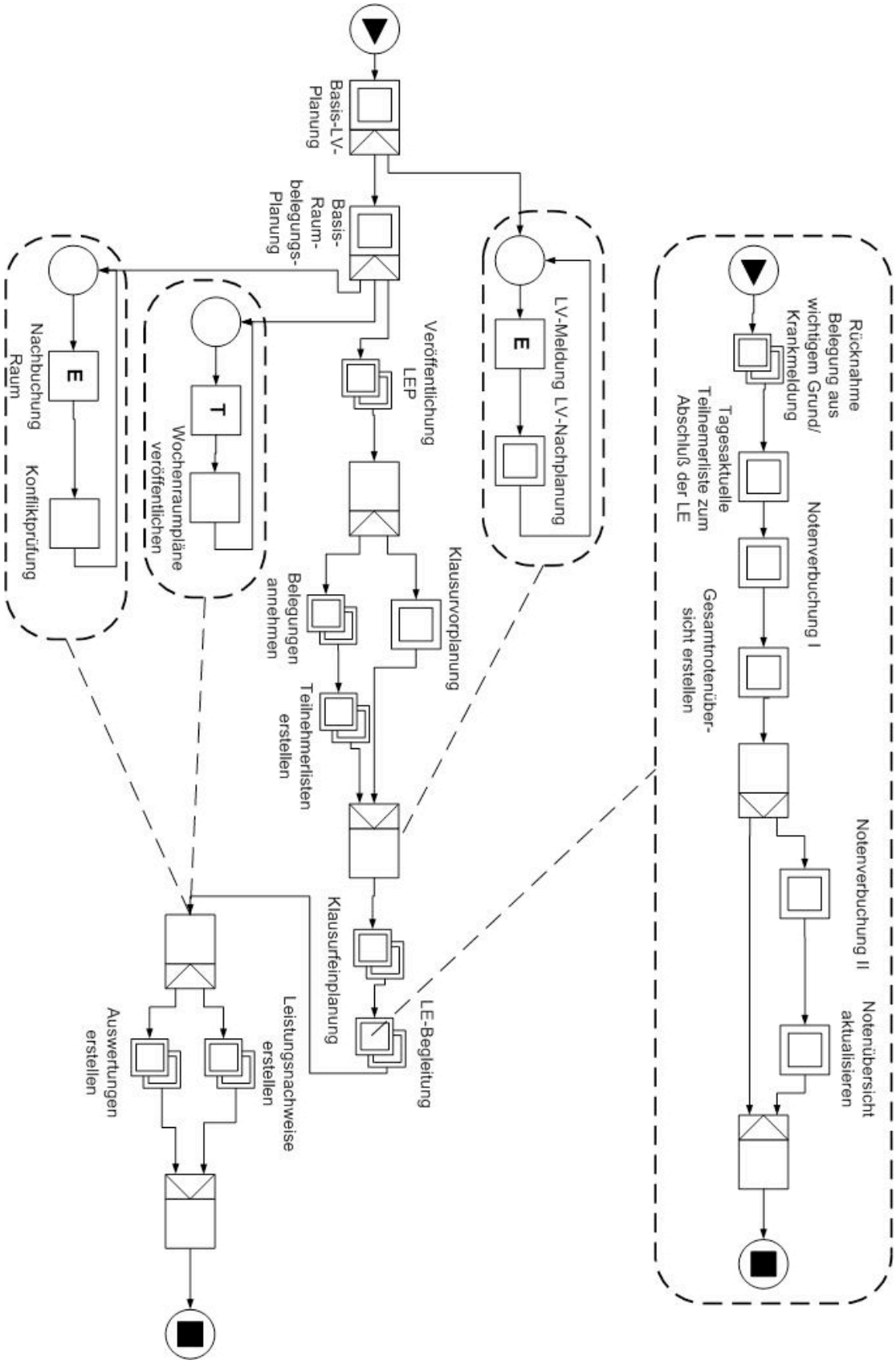


Abbildung 15. Ist-Prozess: Abwicklung HPI-Lehrveranstaltungen

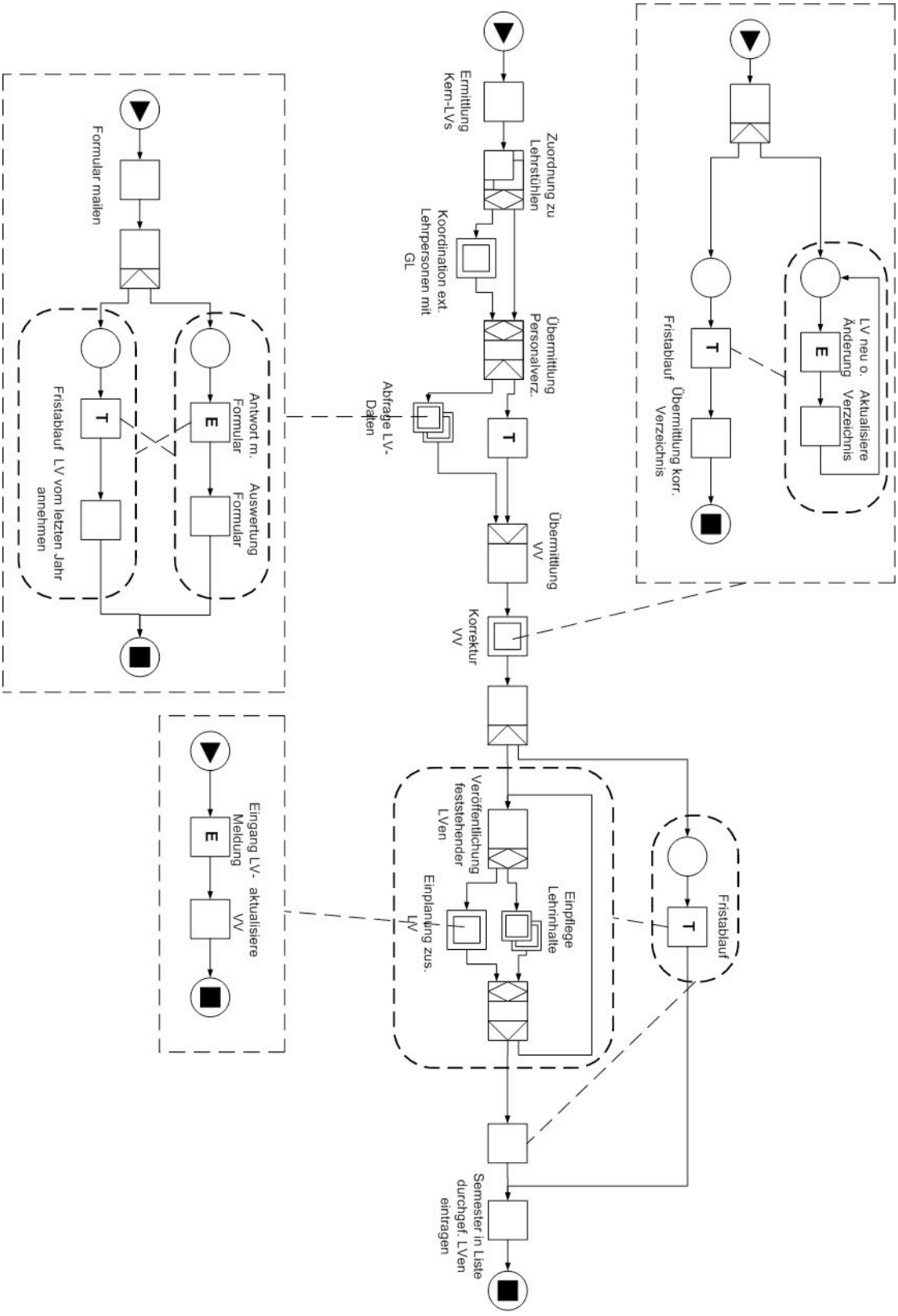


Abbildung 16. Ist-Prozess: Lehrveranstaltugnsplanung

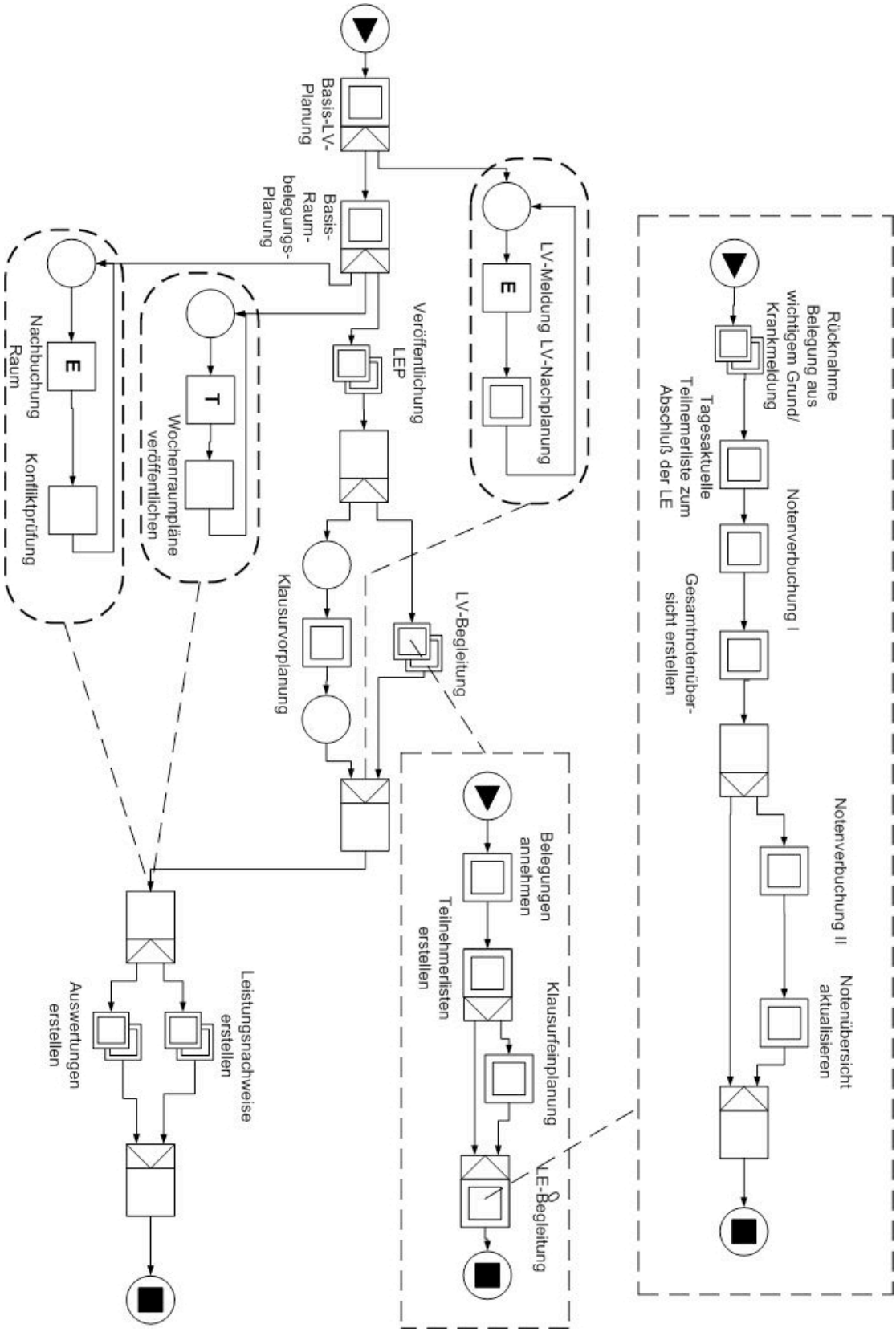


Abbildung 17. Ist-Prozess: Abwicklung HPI-Lehrveranstaltungen, Problemlösung

6.5 Soll-Workflow Diagramme

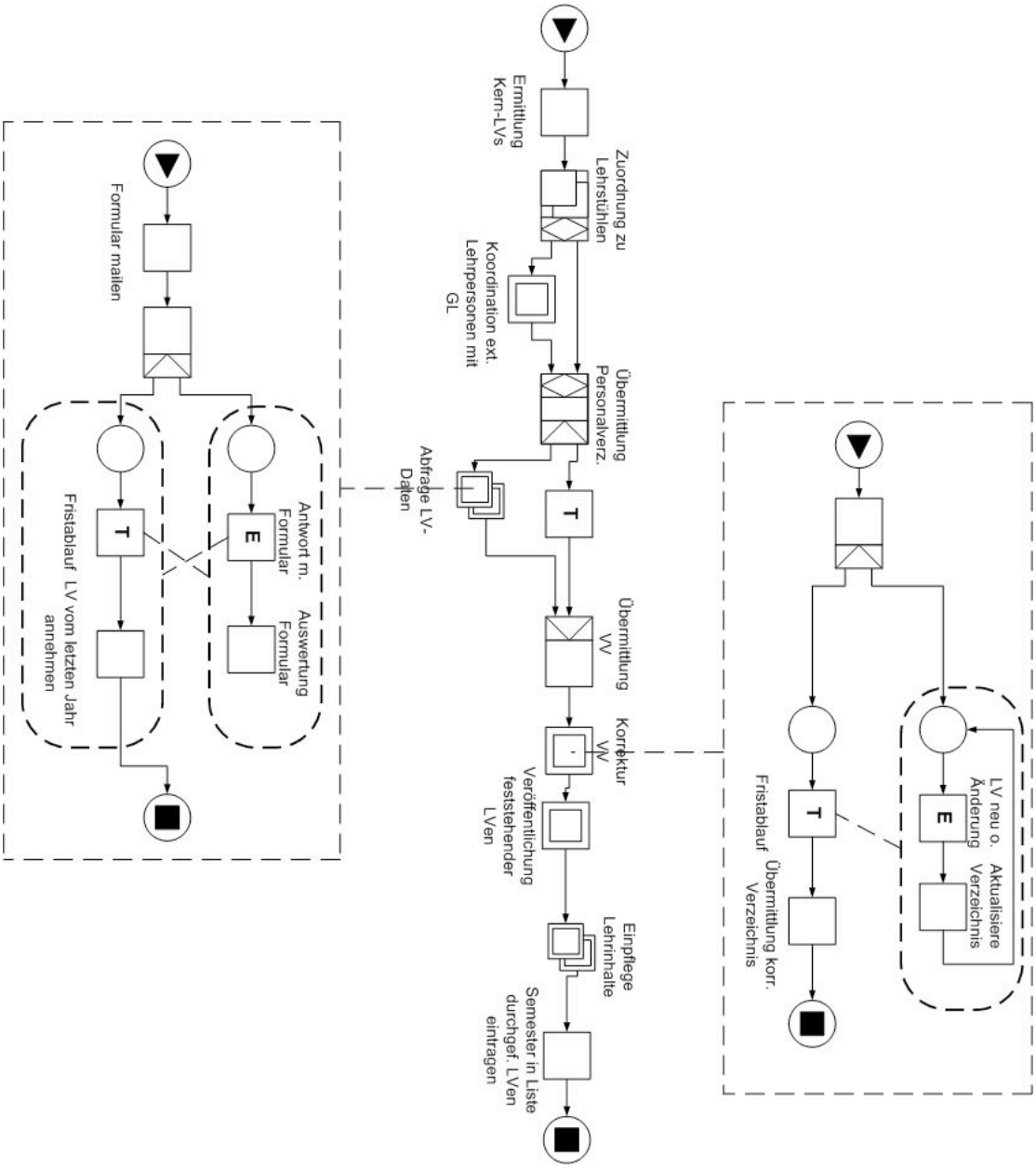


Abbildung 18. Soll-Prozess: Lehrveranstaltungsplanung

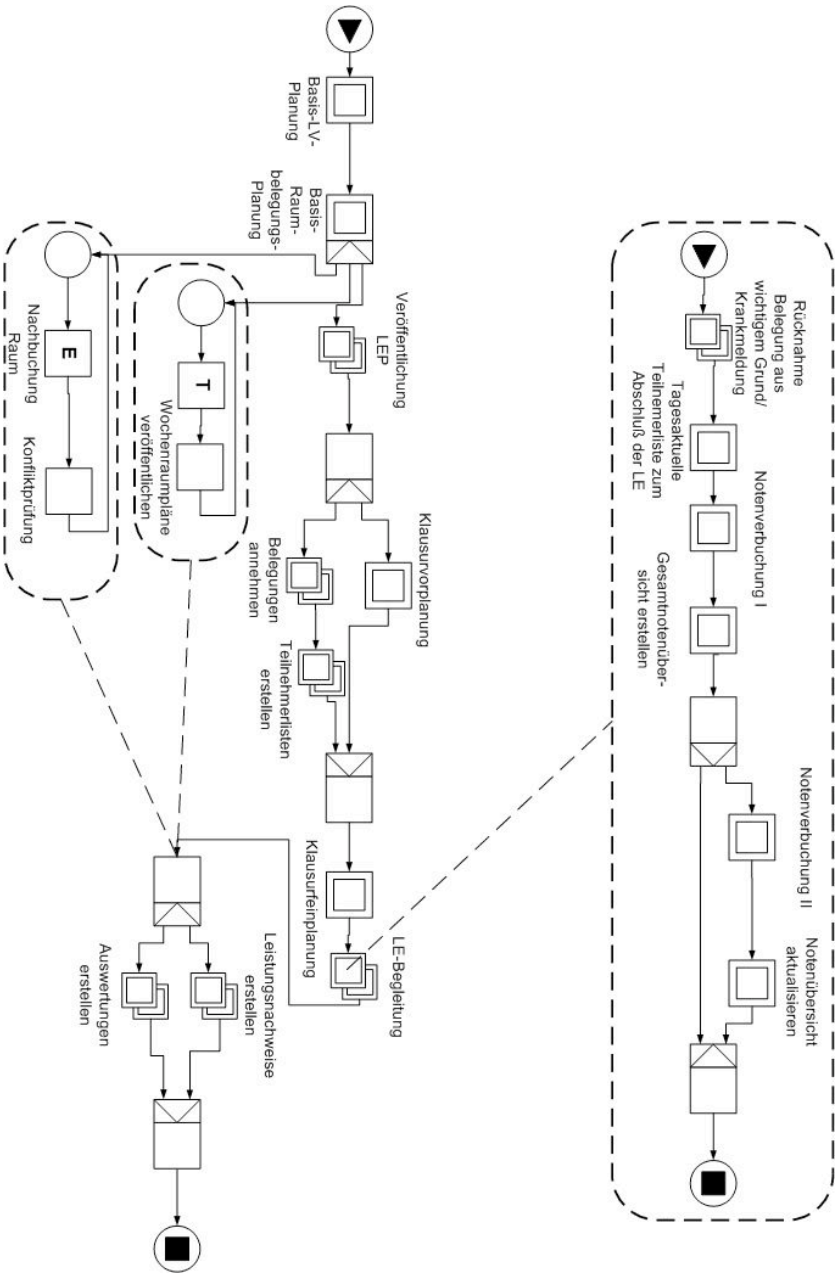


Abbildung 19. Soll-Prozess: Abwicklung HPI-Lehrveranstaltungen

6.6 Abkürzungsverzeichnis

EERD Enhanced Entity-Relationship Diagrams
ERD Entity-Relationship Diagram
HPI Hasso-Plattner-Institut für Softwaresystemtechnik GmbH
PQM Prozessorientiertes Qualitäts-Management
YAWL Yet Another Workflow Language

Literatur

[1] Wagner, „PQM Prozessorientiertes Qualitäts-Management“, Carl Hanser Verlag München Wien, 2003
[2] Brauer, „DIN EN ISO 9000-2000 ff. umsetzen“,Carl Hanser Verlag München Wien, 2002
[3] Liggesmeyer, „Software Qualität“, Spektrum Akademischer Verlag Heidelberg Berlin, 2002
[4] Weske, „Datenbanksysteme“, Vorlesung am Hasso-Plattner-Institut, Universität Potsdam, 2002
[5] van der Aalst et al., „YAWL: Yet Another Workflow Language (Revised Version)“, QUT Technical report, FIT-TR-2003-04, Queensland University of Technology, Brisbane, 2003
[6] van der Aalst et al., „Design and Implementation of the YAWL System“, Proceedings of The 16th International Conference on Advanced Information Systems Engineering (CAiSE 04), Riga, Latvia, Juni 2004
[7] van der Aalst et al., „Distributed and Parallel Databases: Workflow Pattern“, 14(3), Juli 2003
[8] van der Aalst et al., „Lecture Notes in Computer Science Vol. 1901: Advanced Workflow Patterns“, Springer-Verlag Berlin, 2000
[9] HPI, „Prüfungsordnung“, HPI, Universität Potsdam, Juni 2001
[10] HPI, „Studienordnung“, HPI, Universität Potsdam, Juni 2001

Amazon.com

Lars Trieloff

Hasso-Plattner-Institut für Softwaresystemtechnik
lars@trieloff.net

Amazon.com war einer der ersten Online-Shops, deren gesamtes Geschäftsmodell auf den Online-Vertrieb von Waren ausgerichtet war. Bis heute ist Amazon.com wegweisend für die Gestaltung von E-Business Webseiten.
In der Vorliegenden Ausarbeitung werden wichtige Geschäftsprozesse, die für Amazon.com charakteristisch sind analysiert und mithilfe der Business Process Modeling Notation (BPMN) abgebildet und eingeordnet.
Diese Notation ist verhältnismäßig neu und wurde bisher noch nicht systematisch wissenschaftlich untersucht. Auch diese Ausarbeitung wird am Beispiel Amazon.com lediglich ein exemplarisches Anwendungsbeispiel für die BPMN geben.

Einführung

In dieser Ausarbeitung, die meinen Vortrag vom 25. Mai 2004 im Hasso-Plattner-Institut für Softwaresystemtechnik, gehalten im Rahmen des Seminars Prozessmodellierung am Lehrstuhl für Business Process Management Technology, ergänzt werde ich Geschäftsprozesse, die für den Einkauf und die Abwicklung von Bestellungen bei Online-Shops von Bedeutung sind, modellieren.

Dabei wird die graphische Modellierungssprache BPMN, das steht für Business Process Modeling Notation verwendet. Diese Sprache wurde eingeführt, um eine einheitliche Notation für die Darstellung von Geschäftsprozessen zu haben, deren Instanzen leicht verständlich gut austauschbar sind.

Die Modellierung wird sich am Beispiel des Online-Shops Amazon.com orientieren.

Motivation

Das Ziel dieser Untersuchung ist es, die Modellierungssprache Business Process Modeling Notation im Praxiseinsatz zu testen, da sie erst im Mai 2004 veröffentlicht wurde und bisher bis auf einige Beispiele, die von der Business Process Management Initiative vorgestellt wurden, um die Bedeutung einiger Sprachkonstrukte zu erläutern, noch in keinen wissenschaftlichen oder industriellen Projekt, welches die Ergebnisse einer Evaluation veröffentlicht hätte, eingesetzt wurde.

Das Beispiel Amazon.com wurde gewählt, da es sich bei Amazon.com nicht nur einen der ersten reinen Online-Shops der Welt handelt, sondern da Amazon.com aufgrund seines hohen Marktanteils einem großen Teil der derzeitigen Benutzer des World-Wide-Web bekannt ist.

Von Besonderem Interesse ist die Möglichkeit eines Vergleichs zwischen einem reinen Online-Händler wie Amazon.com und einem Versandhaus wie Otto oder Quelle, für die der Online-Shop nur ein weiterer Vertriebskanal neben postalischer und telefonischer Bestellung ist. Nicht zuletzt positioniert Amazon.com sich auch in technischer Hinsicht als richtungweisend, mit Technologien wie dem One-Click-Checkout, der in dieser Untersuchung ebenfalls beleuchtet wird, werden dem Benutzer eines Online-Shops deutliche Erleichterungen der Benutzbarkeit zu teil.

Abriss

Im Verlaufe dieser Ausarbeitung wird zunächst von einer sehr abstrakten Sicht auf den Online-Shop ausgegangen. Es werden beteiligte Parteien der Geschäftsprozesse vorgestellt und Anwendungsfälle analysiert.

Anschließend werde ich auf Möglichkeiten zur Modellierung von Navigationsflüssen in Web-Applikationen eingehen und eine erste Ad-hoc-Modellierung einiger interessanter Geschäftsprozesse vornehmen.

Diese Ad-hoc-Modellierung wird von einer detaillierten, hierarchischen, und in einer Top-Down-Vorgehensweise ermittelten, Analyse der charakteristischen Geschäftsprozesse abgelöst.

Die dort betrachteten Geschäftsprozesse sind: Artikel kaufen, Artikel suchen, Artikel durchsuchen, Ähnliche Artikel betrachten, Wunschzettel verwalten, Warenkorb verwalten, Der One-Click-Checkout, Einloggen, Bestellinformationen verifizieren, Nach der Bestellung, Artikel erhalten, Beschwerden erhalten, Lagerhaltung verwalten, Beschwerden verwalten, Waren anbieten, Nachbestellungen verwalten, Beschwerden von Zulieferern verwalten, Bestellungen überprüfen, und Transaktion durchführen.

Verwandte Arbeiten

Da der Standard der Business Process Modeling Notation noch sehr neu ist, und erst am 3. Mai 2004, also kaum zwanzig Tage von meinem diesbezüglichem Vortrag veröffentlich wurde, liegen noch keine wissenschaftlichen Arbeiten vor, die sich auf die Business Process Modeling Notation beziehen.

Als weitere Möglichkeiten die Betrachtung der Business Process Modeling Notation zu vertiefen, empfehle ich die anderen Vorträge dieses Seminars, die sich mit der Modellierung von E-Business Geschäftsprozessen befassen.

Vorhergehende Betrachtungen

Das Grundlegende Vorgehensmodell bei der Erstellung dieser Ausarbeitung sah es vor, zunächst einen informellen Überblick über die Materie zu erhalten. Dabei analysierte ich die Geschäftsprozesse vorerst ohne Berücksichtigung der Business Process Modeling Notation, sondern betrachtete Anwendungsfälle, beteiligte Personen und Institutionen in verschiedenen Rollen und das Datenmodell.

Anschließend wurden einige wichtige Geschäftsprozesse in der Business Process Modeling Notation abgebildet. Das Ziel war es dabei, sich mit den Eigenheiten dieser Sprache vertraut zu machen und einen grundlegenden Überblick über wichtige Geschäftsprozesse machen zu können.

Erst danach kam es zu einer strukturierten und organisierten Untersuchung aller bedeutenden Geschäftsprozesse, auf die ich im nächsten Abschnitt eingehen möchte.

Beteiligte der Prozesse

Bei der Untersuchung der Beiligten der Geschäftsprozesse ist zunächst der Kunde und Amazon.com zu nennen. Diese tauschen über die Amazon.com Webseite und per Email Informationen über die von Amazon.com vertriebenen Artikel sowie Bestellungen und Bestellbestätigungen aus.
Amazon.com obliegt dabei die Verwaltung der eigenen Lagerbestände, auf die ein Versanddienst Zugriff hat, indem er bestellte Artikel zum Kunden bringt. Die Zahlung wird von einem Netz von Finanzdienstleistern abgewickelt, zu dem die Geschäftsbanken der Kunden und von Amazon.com gehören, aber auch Kreditkartenfirmen und Bewertungsagenturen wie die Schufa. Auf die Betrachtung dieses komplexen Netzes von Dienstleisten wird in dieser Ausarbeitung verzichtet, da das Ziel die Modellierung der Geschäftsprozesse von Amazon.com ist, nicht die von bestimmtem Finanzdienstleistern.

Als weitere Komponente hinzukommen die Zulieferer von Amazon.com. Dies können Buch- oder Musikverlage sein, aber auch Hersteller von anderen Produkten, die Amazon.com vertreibt.

Dieselbe Rolle nehmen auch externe Anbieter ein, die über den Amazon.com Marketplace, ein Customer to Customer-Portal Waren anbieten. Dieses Portal erlaubt es Anbietern, beliebige Produkte, die im Amazon.com-Produktkatalog aufgeführt sind über die Amazon.com Webseite zu vertreiben.

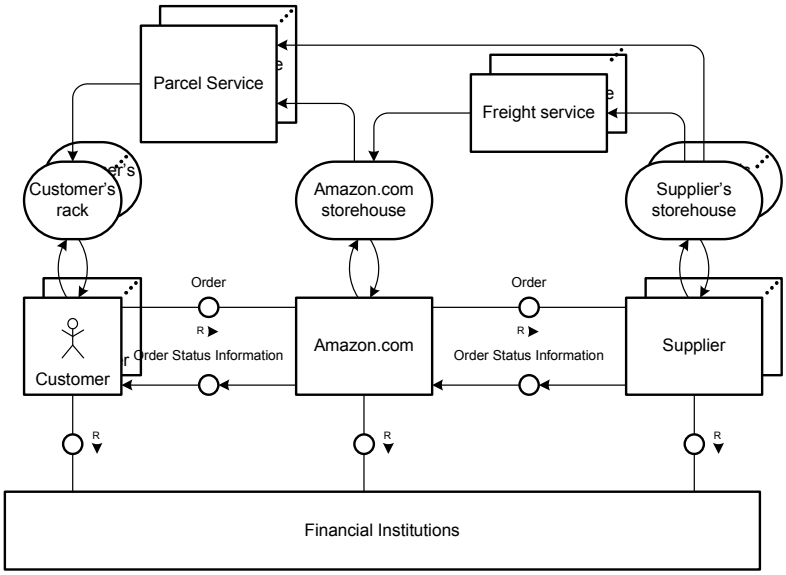
Produkte, die über den Amazon.com Marketplace verkauft werden, müssen von einem Paketdienst direkt vom Anbieter zum Käufer transportiert werden, da Amazon.com lediglich als Vermittler auftritt und keine Haftung und Kontrolle der Waren anbietet.

Lieferungen von Zulieferern an Amazon.com werde nicht über Paketdienste abgewickelt, sondern über Güterverkehrsbetriebe, da in diesem Fall größere Mengen von Produkten von einem in das andere Lage transferiert werden.

Das Ergebnis dieser Betrachtungen wurde von im in dem FMC-Aufbaubild, Fig. 1. zusammengefasst. Die Modellierungssprache FMC, Fundamental Modeling Concepts, wurde hier gewählt, da sie für die Darstellung von Systemstrukturen besser geeignet ist als Softwarespezifische Modellierungssprachen wie UML und gegenüber einer

Ad-hoc-Notation den Vorteil bietet, dass sie vielen Zuhören des Vortrags, namentlich Studenten und Mitarbeiten des Hasso-Plattner-Instituts für Softwaresystemtechnik bereits bekannt ist.

Fig. 1. Beteiligte der Geschäftsprozesse



Block diagram **Szenario Participants**	
Seminar Processmodeling	
Lars Trieloff	05-06-2004

Use Cases

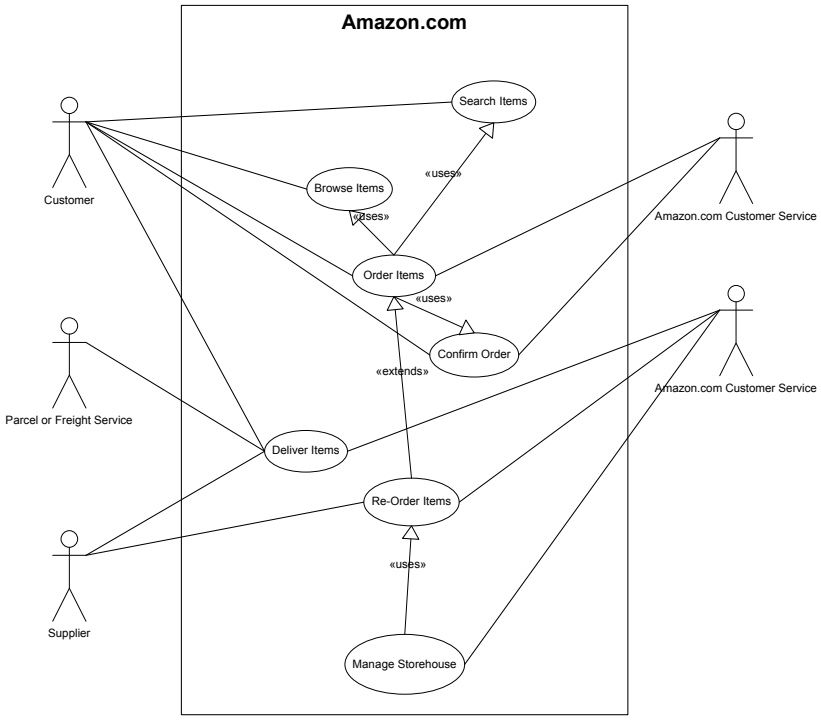
Aus diesem groben Aufbaubild lassen sich bereits einige Anwendungsfälle mit beteiligten Akteuren erkennen.

Wichtige identifizierte Anwendungsfälle sind im Kontext von Amazon.com

- Bestellen von Waren
- Anliefern von Waren
- Verwalten von Lagerbeständen
- Nachbestellen von Waren
- Bestätigen von Bestellungen
- Abwickeln von Zahlungen

Diese lassen sich in einem UML-Use-Case-Diagram wie folgt darstellen:

Fig. 2.



Datenmodell

Die Untersuchung des Datenmodells wurde mir von meinem Seminarbetreuer nahe gelegt und wurde von mir auch durchgeführt, es hat sich jedoch gezeigt, dass für das Reverse-Engineering von Geschäftsprozessen auf einem Abstraktionslevel, wie es bei dieser Ausarbeitung der Fall ist, die Analyse von Datenmodellen nicht von großer Bedeutung ist.

Die Gründe liegen darin, dass das Datenmodell nicht zwingend erforderlich ist, um die Geschäftsprozesse zu verstehen und dass die tatsächlich notwendigen Objekttypen wie Artikel, Bestellung, Ware selbsterklärend sind und keiner komplexen Analyse bedürfen.

Die Situation ist anders geartet, wenn es darum geht, Geschäftsprozesse selbst zu implementieren oder bestehende Geschäftsprozesse mit dem Ziel der Verwaltung dieser durch ein Workflowmanagementprogramm detailliert zu beschreiben. In diesem Fall muss das Datenmodell explizit ausformuliert sein, da andernfalls Inkonsistenzen oder Duplikationen von ausgetauschten Daten zwischen Prozessteilnehmern auftreten können.

Das vereinfachte Datenmodell der Amazon.com E-Business-Applikation finden sie im Anhang (Siehe: E/R Diagram Online Shop Data Model).

Modellierung von Navigationsstrukturen

Eine während des Seminars wiederholt aufgeworfene Frage war: „Wie lassen sich Navigationsstrukuren einer Webseite sinnvoll modellieren?“. Zur Modellierung wurden einige nicht standardisierte Notationen vorgeschlagen, darunter auch eine UML-Erweiterung, die UML mit Flussdiagrammen koppelte.

Die Entwicklung modernen Web-Applikationen greift oftmals auf das Model-View-Controller-Pattern zurück, welches zum einen eine Entkoppelung von Geschäftsobjekten und Geschäftsprozessen von der Darstellung dieser vorsieht, zum zweiten die Möglichkeit, dass eine zentrale Kontrollinstanz verschiedene Geschäftsobjekte kontrolliert und dass verschiedene Sichten auf eine Instanz eines Geschäftsobjekts parallel existieren können.

Zur Erleichterung der Umsetzung solcher MVC (Model-View-Controller) Webapplikationen gibt es eine Reihe von Softwareframeworks, die es Entwicklern erlauben mit geringem Aufwand die Vorteile des MVC-Patterns zu nutzen. Eines der bekanntesten Frameworks ist das Jakarta Struts Framework, welches von der Apache Software Foundation produziert wird.

Das Struts-Framework sieht im Model zum einen die tatsächlichen Geschäftsobjekte vor, auf denen Instanzen von `Action`-Klassen arbeiten. Diese `Action`-Klassen implementieren somit atomare Geschäftsprozesse.

Der Controller wird durch Klassen implementiert, die von `ActionServlet` erben. Ein `ActionServlet` ist ein Java-Servlet, also eine im Kontext eines Webservers ausführbare Java-Klasse, welche die HTTP-Anfrage lesen und die HTTP-Antwort schreiben kann.

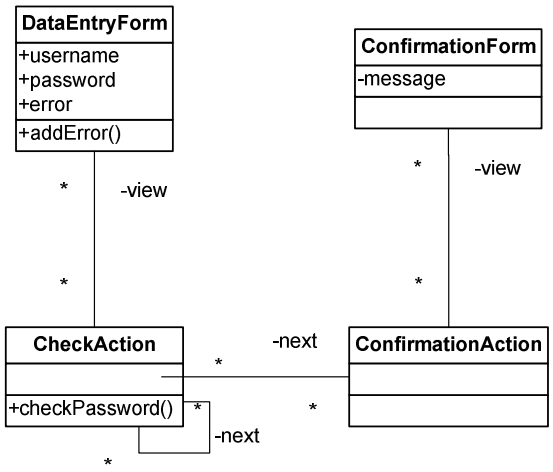
Dieses `ActionServlet` findet anhand eines `ActionMapping`s heraus, welche `Action` instanziiert und ausgeführt werden soll. In der Regel befüllt die entsprechende Action eine so genannte `FormBean` mit Daten, die anschließend durch eine JavaServer Page oder eine ähnliche Frontend-Technologie zur Darstellung gebracht werden.

Die `FormBean` und die zugeordnete JSP stellen damit den View auf die Applikation dar. Außerdem ist es möglich, dass eine Action keine direkte Anzeige hat, sondern die Anfrage an eine andere `Action` weitergibt, und damit Ketten von Aktionen gebildet werden können.

Mit dieser Grundlage lassen sich Navigationsprozesse mit Mitteln der UML darstellen. Als Beispiel sei ein einfacher Eingabedialog genannt, der den Benutzer auffordert, eine Eingabe zu machen, diese Eingabe überprüft und bei erfolgreicher Überprüfung den Benutzer auf eine Bestätigungsseite führt.

Wird hingegen keine korrekte Eingabe getroffen, muss der Benutzer die Eingabe wiederholen.

Fig. 3. Klassendiagramm zum Beispiel



Das Klassendiagramm zeigt, dass das Formular zur Dateneingabe mit einer Überprüfungsaktion kombiniert ist, d.h. die Check Action liest z.B. Benutzername und Passwort aus dem Formular. Bei erfolgreicher Überprüfung wird zur Bestätigen-Aktion weitergeleitet, bei Misserfolg, wird eine Fehlermeldung in das Dateneingabeformular eingetragen. Die `ConfirmationAction` setzt eine Nachricht im Bestätigungsformular und bringt dieses zur Anzeige.

Der Fluss der Aktionen hängt vom Zustand der Geschäftsobjekte und den Eingaben des Benutzers ab. Damit wird klar, dass man die Navigationsstruktur gut mittels UML-Zustandsautomaten darstellen kann, da nun keine Notwendigkeit mehr besteht, eine Webseite mit einer Aktion zu koppeln, da deutlich geworden ist, dass eine Webseite (ein Formular) lediglich Daten enthält, die von den Aktionen ausgewertet werden und die eigentliche Navigation durch das Anzeigen von Formularen oder weitergeben der Anfrage umgesetzt wird.

Ad-hoc-Modellierung von Prozessen

In diesem Arbeitsschritt ging es mir darum, einen Überblick über relevante Geschäftsprozesse möglich zu machen und zugleich sich mit der Business Process Modeling Notation vertraut zu machen.

Diese Diagramme sind in mehrerer Hinsicht nicht optimal. Zum einen greifen sie sich Teilaspekte heraus und sind nicht in ein globales „Helicopter-View“-Diagramm integriert.

Die sich daraus ergebende Folge ist die Inkonsistenz zwischen den Diagrammen. Die sich daraus ergebende Konsequenz, dass nur eine Top-Down-Modellierung zu

einem konsistenten Modell führen kann, wurde in der Hauptmodellierungsphase berücksichtigt.

Ein weiterer Nachteil der folgenden Diagramme ist, dass nicht auf Fehlerbedingen und Fehlerbehandlung eingegangen wird.

Im Folgenden werde ich die einzelnen Ad-hoc-Modellierten Prozessdiagramme kurz vorstellen, alle sind im Anhang zu finden.

Kassieren

Das Collaboration Diagram „Checkout Overview“ zeigt den grundlegenden Ablauf des Bestellvorgangs. Der Kunde stellt seinen Warenkorb zusammen und meldet sich beim System an. Die Anmeldung wird von Amazon.com durch einen Vergleich der Benutzerdaten mit der Datenbank als Teilprozess durchgeführt.

Erst wenn Warenkorb zusammengestellt sind und der Benutzer sich authentifiziert hat, kann mit dem Bestellvorgang fortgefahren werden. Der Besucher der Webseite wählt Zahlungsmethode und Liefer- bzw. Rechnungsadresse aus und sendet die entsprechenden Daten an Amazon.com, welches die Bestellung anschließend abwickelt.

Nach dem Kassieren

In diesem Prozess, dargestellt im Collaboration Diagram „Post-Checkout-Process“ wird an der Stelle fortgefahren, an der die Bestellung des Kunden empfangen wird. Es wird nun eine Bestellbestätigung an den Kunden verschickt und begonnen, die bestellten Waren zu verpacken. Die gepackten Waren werden dem Paketdienst zur Lieferung überstellt und eine Versandbestätigung wird an den Kunden übertragen.

Es kann allerdings vorkommen, dass bestellte Waren nicht im Lager vorhanden sind. In diesem Fall muss unterschieden werden, ob es möglich ist einen Teil der Bestellung zu versenden oder nicht. In jedem Fall wird die Nachlieferung der nichtvorrätigen Waren angestoßen und mit dem Packen der bestellten Waren fortgefahren, sobald die Nachbestellung angekommen ist.

Wenn es möglich ist, dass zunächst nur ein Teil der Bestellung ausgeliefert wird, so werden die vorhandenen Artikel verpackt und an den Lieferdienst überstellt und der Kunde wird per Email über die zu erwartenden Nachlieferung informiert.

Anbieten am Amazon.com Marketplace

Eine entscheidende Besonderheit von Amazon.com ist das Amazon.com Marketplace Portal. Es ist eine Consumer to Consumer Plattform (C2C), auf der Endkunden Waren verkaufen können und Amazon.com lediglich als Vermittler auftritt.

Der grundlegende Ablauf ist nach der Modellierung im Collaboration Diagram „Marketplace Offering“ folgender: Der Anbieter durchsucht den Amazon.com Katalog und findet Artikel, die er anzubieten hat. Amazon.com wird über den Angebotswunsch unterrichtet und aktualisiert seinen Datenbestand.

Der Kunde durchsucht ebenfalls den Amazon.com Datenbestand und fügt Artikel aus dem Amazon.com Marketplace seinem Warenkorb hinzu. Nach der Bestellung werden die Daten am Amazon.com übertragen, welches wiederum Anbieter und Käufer informiert.

Der informierte Anbieter muss jetzt die Waren versenden, womit der Prozess für ihn beendet ist. Der Käufer empfängt die Waren und der Prozess ist auch für den Käufer beendet.

Dazu parallel wickelt Amazon.com die Bezahlung ab. Dem Käufer wird ein entsprechender Betrag in Rechnung gestellt und dem Käufer gutgeschrieben. Damit es dabei nicht zu Inkonsistenzen kommt, werden beide Teilprozesse in Form einer Transaktion durchgeführt.

Modellierung der Geschäftsprozesse

In der Hauptmodellierungsphase, die auf die ersten Ad-hoc-Modellierungsschritte folgte, konnte ich auf die Erfahrungen aus der vorherigen Phase zurückgreifen. So erfolgte die weitere Modellierung einem Top-Down-Ansatz, indem zunächst ein globales Übersichtsmodell erstellt wurde, in dem alle wichtigen Geschäftsprozesse wiedergefunden werden können.

Die entsprechenden Teilprozesse wurden verfeinert, wobei die Verfeinerungen häufig erneut zusammengesetzte Prozesse enthielten, die erneut einer Verfeinerung bedurften.

Da es noch keine entsprechend mächtigen Werkzeuge zur Erstellung von Business Process Modeling Notation Diagrammen gab, was insbesondere der Novität dieses Standards geschuldet ist, gab es auch keine Möglichkeit, Diagramme zu validieren und eine Gruppe von Diagrammen auf Vollständigkeit zu überprüfen.

Zu diesem Zweck führte ich eine Liste über die erstellten Diagramme und über die enthaltenen Subprozesse beziehungsweise über Verweise auf zusammengefasste Subprozesse. Auf diese Weise konnte ich überprüfen, ob ich auf zusammengefasste Subprozesse verwiesen habe, die in keinem Diagramm komplett ausformiert wurden.

Auch die im Folgenden besprochenen Diagramme finden sich im Anhang.

Überblick

Den Gesamtüberblick über alle betrachteten Prozesse im Kontext der E-Business-Applikation Amazon.com bietet das Business Process Modeling Notation Collaboration Diagram „Overview“.

Es bietet sich an, für jeden Prozessbeteiligten, hier dargestellt durch einzelne Pools, die Abfolge der Oberprozesse einzeln darzustellen.

Der Kunde kauft Artikel bei Amazon.com, wartet anschließend auf die Bestellung, um anschließend die gelieferten Waren zu überprüfen und eventuell zu beanstanden.

Der Paketdienst ist damit beschäftigt, Artikel von Amazon.com oder dem Anbieter am Amazon.com Marketplace zum Käufer zu transportieren oder vom Käufer beanstandete Produkte zurück zu Amazon.com oder zum Amazon.com Marketplace Anbieter zu bringen.

Amazon.com beinhaltet drei wichtige Oberprozesse. Zum ersten die Verwaltung des Lagerbestandes. Dabei muss insbesondere auf Nachlieferungen von Zulieferern und Anforderungen auf nichtvorrätige Produkte geachtet werden. Weiterhin die

Abwicklung von Handelsgeschäften und die Abarbeitung von Beschwerden oder sonstigen auftretenden Problemen.

Der Güterverkehrsdienst transportiert Nachlieferungen vom Zulieferer zum Amazon.com Lager.

Der Anbieter am Amazon.com Marketplace oder der Zulieferer von Amazon.com nehmen in diesem Diagramm eine Rolle ein, da ihr Verhalten sehr ähnlich ist. An diesem Punkt wäre die Möglichkeit der Vererbung von Rollen/Prozessmodellierung von Vorteil, dies ist in der Business Process Modeling Notation allerdings nicht vorgesehen. In dieser Rolle werden zunächst Angebote gemacht, die auf die Gestaltung des Amazon.com Warenkatalogs Einfluss haben. Zum zweiten müssen Bestellungen und Nachbestellungen von Amazon.com sowie Amazon.com Marketplace-Bestellungen gehandhabt werden und eventuelle Beschwerden oder Problemfälle, insbesondere im Zusammenhang mit Angeboten des Amazon.com Marketplace abgewickelt werden.

Der Finanzdienstleister bietet die die Dienste der Zahlungsverifizierung und Zahlungsdurchführung an, die an diversen Stellen im Gesamtprozess aufgerufen werden.

Artikel kaufen

Der Prozess des Einkaufens wird durch den Expanded Sub-Process „Purchase Items“ beschrieben. Der Kunde füllt seinen Warenkorb, indem er in beliebig vielen Iterationen nach Artikeln sucht, den Katalog durchsucht, seinen Wunschzettel verwaltet und Empfehlungen von Amazon.com Beachtung schenkt. Jede Iteration wird dadurch beendet, dass der Kunde einen gefundenen Artikel in den Warenkorb legt.

Jederzeit während des Füllens des Warenkorbs kann der Kunde seinen Warenkorb verwalten und die Zahl von Artikeln korrigieren oder Artikel aus dem Warenkorb nehmen.

Danach gibt es für den Kunden zwei Möglichkeiten, zum ersten kann der Weg des One-Click-Checkout gewählt werden, oder der herkömmliche Bestellvorgang. Wenn nicht genügend Informationen für einen One-Click-Checkout vorliegen, so wird zum normalen Bestellvorgang übergegangen.

Dieser sieht so aus, dass der Kunde sich zunächst gegenüber Amazon.com authentifiziert, anschließend Zahlungsmethode, Rechnungsadresse und optional eine davon abweichende Lieferadresse angibt. Diese Informationen werden von Amazon.com überprüft, welches bei einem Fehler zu einer Änderung der Zahlungsbedingungen auffordert. So kann eine gesperrte Kreditkarte zurückgewiesen werden oder ein säumiger Zahler kann zur Zahlung per Vorkasse aufgefordert werden.

Die Bestellinformationen werden nach einem positiven Verifizierungsbescheid an Amazon.com übertragen und die Nachbehandlung der Bestellung kann beginnen.

Im den nächsten Unterabschnitten werde ich auf die einzelnen hier angesprochenen Unterprozesse eingehen.

Artikel suchen
Beschrieben im Expanded Sub-Process „Search Items“. Der Kunde durchsucht den Katalog durch Eingabe bestimmter Suchkriterien. Anschließend betrachtet er die Ergebnisliste und kann die Suche einschränken oder erweitern, beziehungsweise, wenn er einen interessanten Artikel auf dieser Liste findet, diesen zum Warenkorb oder zur Wunschliste hinzufügen oder die Detailansicht des Artikels aufrufen. Von hier aus kann man den Artikel ebenfalls zur Wunschliste oder zum Warenkorb hinzufügen oder zum One-Click-Checkout weitergehen, was den Suchprozess ebenso beendet wie das betrachten ähnlicher oder empfohlener Artikel.

Ist der Kunde hingegen an dem Artikel nicht interessiert, so kann er mit der Suche fortfahren.

Artikel durchsuchen
Dieser Teilprozess ist im Expanded Sub-Process „Browse Items“ dargestellt. Der Kunde kann sich hierbei eine Kategorie zum Durchsuchen auswählen oder Unterkategorien einer Kategorie betrachten.

Für die einzelnen Artikel stehen die gleichen Möglichkeiten zur Wahl, wie auch bei der Artikelsuche: Hinzufügen zum Warenkorb, Hinzufügen zum Wunschzettel, One-Click-Checkout und Betrachtung ähnlicher oder empfohlener Artikel.

Ähnliche Artikel betrachten
Zum Betrachten ähnlicher Artikel, dargestellt im Expanded Sub-Process „Related Items“ kann man von verschiedenen Startpunkten aus gelangen, als da wären Empfehlungen von Amazon.com, die anhand von Verkaufsmustern berechnet werden, Empfehlungen vom Amazon.com, die redaktionell erstellt werden und so genannten Guide-Lists, die von Amazon.com-Kunden erstellt werden und eine Reihe von Produkten zu einem bestimmten Thema enthalten.

Der Kunde kann den empfohlenen Artikel zum Warenkorb oder zur Wunschliste hinzufügen, aber auch detailliert betrachten. Ist der Kunde nicht interessiert, so wird der Teilprozess abgebrochen, bei Interesse kann er neben den zuvor genannten Optionen auch mit dem One-Click-Checkout fortfahren.

Wunschzettel verwalten
Die Verwaltung des Wunschzettels setzt voraus, dass der Benutzer sich authentifiziert hat und ein Wunschzettel vorhanden ist. Ist eine der beiden Voraussetzungen nicht erfüllt, so können sie durch einen Login oder die automatische Erstellung des Wunschzettels behoben werden.

Gegenstände auf dem Wunschzettel können in den Warenkorb gelegt werden oder vom Wunschzettel gelöscht werden oder en Detail betrachtet werden. In diesem Fall stehen wieder die Optionen One-Click-Checkout und Verfolgen von Empfehlungen zur Auswahl.

Warenkorb verwalten
Die Verwaltung des Warenkorbs, dargestellt durch den Expanded Sub-Process „Manage Shopping Cart“, sieht vor, dass der Benutzer die Inhalte seines Warenkorbs

betrachtet. Existiert kein Warenkorb für den aktuellen Benutzer so wird ein leerer Warenkorb initialisiert.

Für einzelne Posten im Warenkorb kann die Anzahl geändert werden oder die Detailansicht aufgerufen werden. Aus der Detailansicht kann zum One-Click-Checkout weitergegangen werden oder es können Empfehlungen zu diesem Artikel betrachtet werden.

Der One-Click-Checkout

Der One-Click-Checkout, dargestellt im Expanded Sub-Process „One-Click-Checkout“ setzt voraus, dass diese Möglichkeit Waren zu bestellen für den Benutzer aktiviert ist.

Ist sie das nicht, so kann der Benutzer sie aktivieren, was jedoch eine Authentifizierung voraussetzt. Nach der Aktivierung oder, für Accounts, bei denen der One-Click-Checkout bereits aktiviert ist werden die Bestellinformationen überprüft. Sind diese korrekt und vollständig ist der One-Click-Checkout erfolgreich abgeschlossen.

Bei falschen oder unvollständigen Informationen wird dieser Teilprozess mit einem Fehler verlassen.

Einloggen

Wesentlich komplexer als der One-Click-Checkout ist der Authentifizierungsprozess bei Amazon.com, was insbesondere an der Fehlerbehandlung liegt. Dieser Prozess wird im Expanded Sub-Process „Login“ dargestellt.

Ist der Benutzer bereits eingeloggt, so war die Authentifizierung erfolgreich und kann als abgeschlossen betrachtet werden, sofern keine kritische Transaktion folgen soll, bei der eine Aktualisierung der Login-Informationen erforderlich ist.

Betrachte man zunächst den Fall, dass es sich um einen wiederkehrenden Kunden handelt. In diesem Fall gibt er Email-Adresse und Passwort ein, um sich zu authentifizieren. Bei fehlerhaften Angaben hat der Benutzer die Wahl zwischen erneuter Eingabe und der Eingabe einer Email-Adresse zur Passwortwiederherstellung.

Bei der Passwortwiderherstellung wird dem Benutzer eine Email an die angegebene Adresse geschickt, die einen Bestätigungslink enthält. Folgt der Benutzer diesem Link, kann er ein neues Passwort wählen, welches durch Doppeleingabe bestätigt wird. Danach wird er als authentifiziert betrachtet.

Neue Besucher müssen zunächst ihre Email-Adresse angeben, anschließend Email-Adresse und Passwort wählen. Diese zweite Eingabeaufforderung bleibt so lange bestehen, bis Passwort und Kontrollpasswort zur Tippfehlervermeidung übereinstimmen und die angegebene Email-Adresse gültig ist.

Bestellinformationen verifizieren

Die Verifikation der Bestellinformationen ist im Collaboration Diagram „Verify Order Information“ dargestellt. In diesem Fall ist nicht nur Amazon.com mit zwei Unterabteilungen, sondern auch ein Finanzdienstleister, konkreter eine Kreditkartenfirma betroffen.

In diesem Beispiel konnte ich, das es sich lediglich um eine Black-Box-Modellierung handelte allerdings nur Mutmaßungen anstellen, was die Praktikabilität des Modells jedoch nicht einschränkt.

Zunächst wird nach der Zahlungsmethode unterschieden. Im Falle einer Zahlung per Rechnung oder per Lastschrifteinzug ist es von Bedeutung, die Zahlungsmoral des Kunden zu prüfen, da offene Rechnungen oder Lastschriftrückläufer ein finanzielles Risiko darstellen. Dazu wird zunächst die Zahlungshistorie des Kunden abgefragt und die Plausibilität der Bankverbindungsdaten überprüft. Im Falle einer negativen Prüfung wird der Teilvorgang mit einem Fehler beendet.

Bei einer Kreditkartenzahlung muss die Prüfung durch einen externen Dienstleister vorgenommen werden. Dieser externe Dienstleister überprüft zunächst die Validität von Kreditkartennummer und Ablaufdatum und stellt anschließend fest, ob die angeforderte Zahlung im Kreditrahmen des Kunden liegt. Ist dies der Fall, so wird der Transaktion zugestimmt, andernfalls wird sie abgelehnt.

Amazon.com wiederum wertet die von der Kreditkartenfirma erhaltene Information aus, um den Prozess erfolgreich oder mit einem Fehler zu beenden.

Der Standardfall, also die Zahlung per Nachnahme sieht keine weitere Prüfung vor, obwohl auch hier geprüft werden könnte, ob Adresse und Empfängername übereinstimmen.

Nach der Bestellung

Wie bereits angedeutet hat die Positionierung von Amazon.com als reiner Online-Shop gewisse Vorteile, die Versandhäuser, für die der Online-Shop nur ein weiterer Kanal zur Bestellaufnahme ist, nicht bieten können. Dies zeigt sich insbesondere in Prozesshinsicht, wenn es darum geht, die Bestellung zu handhaben, da es bei Amazon.com möglich ist, im Nachhinein noch Änderungen an der Bestellung vorzunehmen.

Dazu wird mit Abgabe der Bestellung ein Timer initialisiert, der nach zwei Stunden ausläuft. Nach Ablauf dieser zwei Stunden wird die Bestellung finalisiert und es können keine Änderungen mehr vorgenommen werden. Möglichkeiten noch weitere Änderungen vorzunehmen bestehen darin, weitere Artikel der Bestellung hinzuzufügen (über eine normale Bestellung oder One-Click-Checkout), die Zahlungsweise zu ändern, die Rechnungs- oder Lieferadresse zu ändern oder einzelne Artikel von der Bestellung auszuschließen. Werden alle Artikel von der Bestellung ausgeschlossen, so hat das den gleichen Effekt wie das Zurückziehen der Bestellung: Die Bestellung wird storniert und der Prozess ist effektiv beendet.

Artikel erhalten

Die Zustellung von Artikeln, dargestellt im Collaboration Diagram „Receive Items“, ist deshalb ein komplexerer Prozess weil die Übergabe bei Nachnahmezahlung an die Zahlung gekoppelt ist und der Kunde unter bestimmten Umständen die Annahme der Ware verweigern kann.

Der Prozess beginnt durch die Aufforderung seitens Amazon.com and den Lieferdienst einen oder mehrere Artikel an den Kunden auszuliefern. Hierbei muss

unterschieden werden, ob es sich um eine Nachnahmezahlung, eine Paketzustellung oder eine Päckchenzustellung handelt.

Bei der Nachnahmezahlung muss der Kunde die Ware erst bezahlen, um sie in Empfang zu nehmen. Lehnt der Kunde die Zahlung ab oder ist dauerhaft nicht zu erreichen, muss die Ware wieder zurück zu Amazon.com gebracht werden und der Zustellungsprozess wird abgebrochen oder fehlerhaft beendet. Zahlt der Kunde hingegen und nimmt er die Lieferung an, ist der Prozess erfolgreich abgeschlossen.

Bei einer Päckchenzustellung kann der Lieferant das Packchen im Briefkasten oder bei Nachbarn hinterlassen, ohne dass der Kunde die Möglichkeit hat, die Annahme abzunehmen. Der Prozess wird also geordnet beendet.

Bei einer Paketzustellung hat der Kunde wiederum die Möglichkeit, die Annahme des Pakets abzulehnen. In diesem Fall wird der Prozess mit einem Fehler beendet, nachdem der Zustelldienst die Ware zu Amazon.com zurückgebracht hat.

Beschwerden erhalten

Im Prozess „Beschwerden erhalten“, dargestellt im Collaboration Diagram „File Complaints“ wird der Kunde zunächst die erhaltene Ware überprüfen. Entspricht die Ware nicht seinen Wünschen, so informiert der Kunde Amazon.com und erhält bald darauf Anweisungen, wie die Ware zurück zu Amazon.com gelangen kann.

Ein Paketdienst wird dann die Waren entgegennehmen und zurück zu Amazon.com bringen.

Gibt es hingegen keinen Anlass zur Beschwerde, so wird der Prozess beendet.

Lagerhaltung verwalten

Im Expanded Sub-Process „Manage Storehouse“ wird dargestellt, wie die Lagerverwaltung bei Amazon.com erfolgen könnte. Anlass zur Lagerverwaltung geben zwei häufig auftretende Eingangsprozesse: Entweder wird ein Artikel von einem Kunden zurückgeschickt oder auch von der Versandabteilung zurückgestellt, da eine komplette Lieferung bislang noch nicht erfolgen kann oder ein Gegenstand wird dem Lager entnommen, weil er gekauft wurde.

In jedem Fall müssen die Lagerbestandszahlen für den jeweiligen Artikel aktualisiert werden und, wenn ein Schwellwert, der von der Popularität des Gegenstandes abhängt, unterschritten ist, Nachbestellungen getätigt werden.

Weitere Ausgangspunkte sind die Änderung der Anzahl der auf dem Amazon.com Marketplace angebotenen Artikel. Wenn ein Anbieter ein neues Angebot einstellt oder ein Angebot zurückzieht, so müssen die Listen, die die Amazon.com Marketplace Angebote enthalten aktualisiert werden.

Schließlich läuft ein weiterer Teilprozess ab, in dem neue Gegenstände dem Amazon.com Katalog hinzugefügt werden, zum Beispiel Neuerscheinungen von Produkten. Hierbei kann es vorkommen, dass Amazon.com Mitarbeiter sofort eine Rezension für den entsprechenden Artikel verfassen.

Beschwerden verwalten

Ein sehr interessanter Teilprozess ist im Collaboration Diagram „Manage Complaints“ dargestellt. Hierbei geht es um die Verwaltung von Beschwerden und Rücksendungen.

Wünscht ein Kunde eine Rücksendung oder möchte er die Bestellung Rückgängig machen, so setzt er sich mit Amazon.com in Verbindung. Dort muss unterschieden werden, ob die Artikel schon dem Versandunternehmen überstellt wurden, ob sie schon ausgeliefert wurden oder ob es sich um eine Bestellung beim Amazon.com Marketplace handelt.

Haben die Artikel Amazon.com noch nicht verlassen, so wird der Kauf einfach abgebrochen.

Wurden sie bereits zum Versand überstellt, jedoch noch nicht ausgeliefert, so wird der Versanddienst informiert, die Auslieferung abzubrechen und die Waren zurück zum Sender zu bringen.

Wenn die Waren bereits beim Kunden angekommen sind, müssen dem Kunden Anweisungen zur Durchführung der Rücksendung zu kommen. Dieser sendet die Gegenstände über den Paketdienst zurück.

Sobald der Paketdienst die Artikel zurückgebracht hat, werden sie auf Vollständigkeit und Widerverkaufbarkeit überprüft. Fällt diese Prüfung erfolgreich aus, wird die Kundenrechnung entlastet und der Prozess ist beendet.

Fällt diese Prüfung jedoch negativ aus, so muss sich die Kundendienstabteilung mit Kunde und Versanddienstleister in Verbindung setzen, um diesen Fall individuell zu klären.

Bei einem Einkauf im Amazon.com Marketplace müssen zuerst Anbieter und Käufer informiert werden, dass die Transaktion rückgängig zu machen ist. Verstreicht dabei eine gewisse Zeitspanne oder weist der Anbieter die zurückgeschickten Artikel zurück, so muss der Fall wiederum individuell von der Kundendienstabteilung geklärt werden.

Andernfalls wird die Transaktion rückgängig gemacht und eventuell geleistete Zahlungen werden erstattet.

Waren anbieten

Das Anbieten von Waren auf dem Amazon.com Marketplace wird im Expanded Sub-Process „Offer Items“ beschrieben. Die grundsätzliche Vorgehensweise ist Folgende: der Anbieter sucht den anzubietenden Artikel im Amazon.com-Katalog durch Nutzung der Funktion „Suchen“ oder „Durchsuchen“ kann dann die Preisempfehlung von Amazon.com beachten und gibt schließlich einige Informationen zu dem Zustand des angebotenen Artikel an.

Der Benutzer kann nun einen eigenen Preis für das Angebot festlegen und bestimmen, in welche Regionen ausgeliefert werden soll.

Damit ist der Artikel angeboten und der Anbieter erhält eine Bestätigungsnachricht von Amazon.com

Nachbestellungen verwalten

Die Verwaltung von Nachbestellungen, dargestellt im Collaboration Diagram „Manage Re-Order“, unterscheidet zwei Fälle. Zunächst werden die zu versendenden und bestellten Artikel verpackt und versandbereit gemacht. Handelt es sich um eine Amazon.com Marketplace-Bestellung, so werden die Artikel über den Paketdienst verschickt.

Handelt es sich hingegen um eine Nachbestellung direkt von Amazon.com, so wird die Lieferung über ein Gütertransportserviceunternehmen abgewickelt und nach der Übergabe der Waren an das Transportunternehmen wird die Lieferung Amazon.com in Rechnung gestellt.

Beschwerden von Zulieferern verwalten

Dieser einfache Prozess ist im Expanded Sub-Process „Manage Supplier Compaints“ dargestellt. Der Prozess beginnt damit, dass der Anbieter eine Nachricht erhält, dass es Beschwerden gegeben habe. Er wartet dann auf die Rücksendung der Ware, um diese zu prüfen. Fällt die Prüfung positiv aus, wird die Beschwerde akzeptiert, und Amazon.com informiert, andernfalls wird Amazon.com über die Probleme mit der Handhabung der Beschwerde informiert.

Gleichfalls wird verfahren, wenn nach Versteichen einer bestimmten Frist keine Rücksendung eingeht.

Bestellungen überprüfen

Diese Aktion wird vom Finanzdienstleister durchgeführt und ist im Expanded Sub-Process „Verify Order“ zu betrachten.

Der eingehende Prüfungsauftrag wird angenommen, es wird überprüft, ob der Anfragende die nötigen Berechtigungen hat, Bestellungen zu prüfen, zum Beispiel ob Amazon.com mit der jeweiligen Kreditkartenfirma einen Vertrag abgeschlossen hat.

Anschließend wird Rechnungsbetrag und Kreditrahmen des Kunden verglichen, um dann einen positiven oder negativen Bescheid abzugeben,

Transaktion durchführen

Die Abwicklung einer Zahlungstransaktion ist schematisch im Expanded Sub-Process „Handle Transaction“ aufgeführt. Nach Eingang der Aufforderung zur Transaktionsdurchführung wird erst das Konto des Zahlenden belastet, um den entsprechenden Betrag anschließend dem Konto des Empfängers gutzuschreiben.

Treten dabei Fehler auf, so wird die Überweisung rückgängig gemacht.

Zusammenfassung

Die vorliegende Analyse kann nur eine begrenzte Analyse der Prozesse im Amazon.com Online Shop sein, da Amazon.com als Black-Box-Modell betrachtet wurde und nur ein Reverse-Engineering möglich war.

Um tatsächliche E-Business Applikationen aus einem solchen Modell ableiten zu können muss die Analyse wesentlich umfassender und die Modellierung sehr viel genauer sein.

Ausblick

Da die Business Process Modeling Notation noch ein sehr neuer Standard ist, ist sie noch nicht in dem Maße verbreitet und eingeübt wie zum Beispiel die UML. Als Folge einer weitergehenden Adoption der Business Process Modeling Notation erwarte ich die Herausbildung von Standardvorgehensweisen und üblichen Modellierungsausdrücken, welche die Möglichkeit, ein und denselben Prozesssachverhalt in vielerlei Art und Weise auszudrücken zwar nicht unterbinden, jedoch eine Modellierungsvariante als die bevorzugte erscheinen lassen, so dass es leichter Möglich ist, Diagramme zu lesen und zu verstehen.

In diesen Bereich würde ich auch die Bildung von BPMN-Modellierungs-Pattern und Anti-Pattern einordnen.

Im Bereich der Erstellung von Diagrammen erwarte ich, dass es bald entsprechende Werkzeugprogramme gibt, mit denen Prozessmodellierer ihre Modelle auf Vollständigkeit, Konsistenz und Standardkonformität überprüfen lassen, so dass nur noch wohlgeformte Modelle überhaupt erstellt werden können.

Aus diesen vollständigen, wohlgeformten und standardkonformen Modellen ließe sich dann leicht Programmcode für Enterprise-Anwendungen oder Business Process Execution Language Code erstellen, der von geeigneten Abwicklern ausgeführt werden könnte.

Fazit

Die Business Process Modeling Notation stellt sich als sehr flexible und mächtige Prozessmodellierungssprache dar, die im Umfeld der Bildung nachrichtenflussorientierter Unternehmensanwendungen, die ein breites Feld von Legacy-Anwendungen und externer Dienste konsumieren, effektiv eingesetzt werden kann.

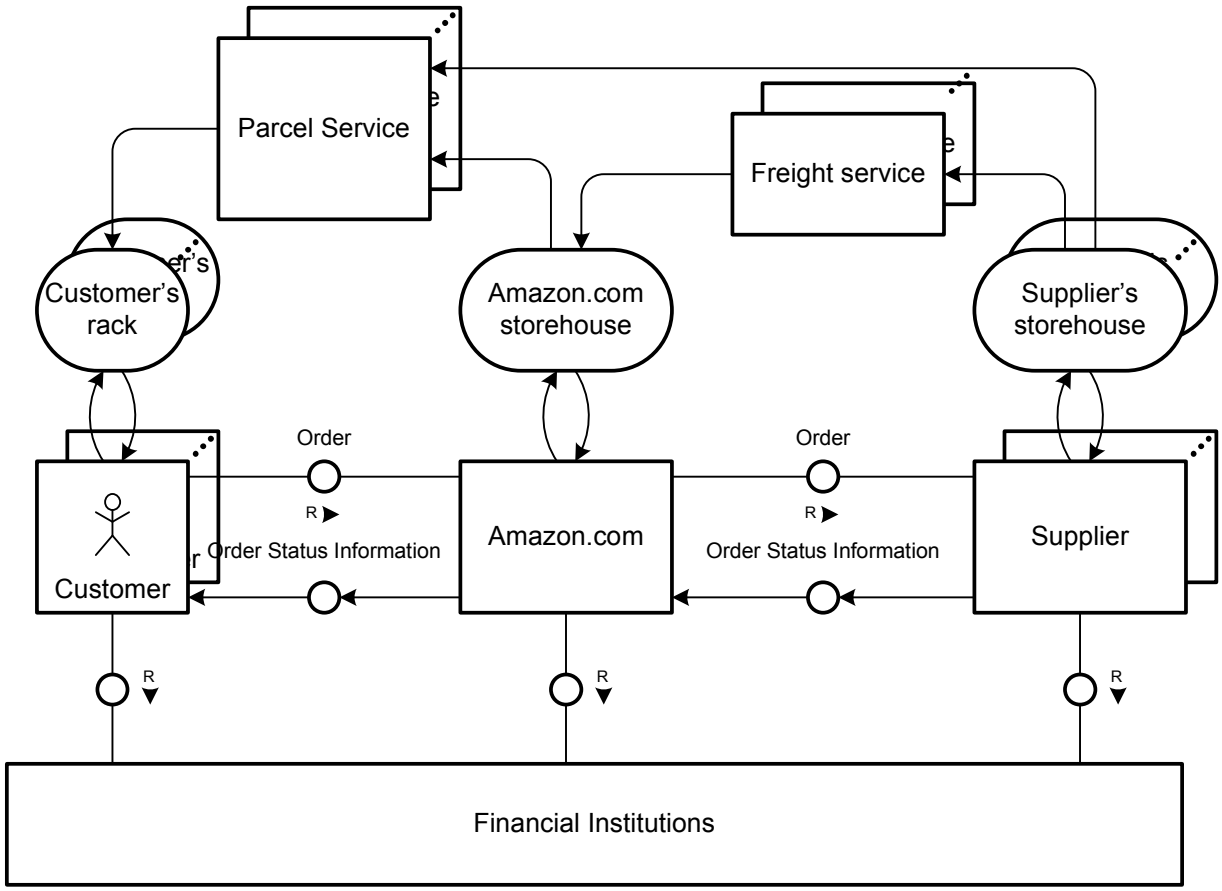
Die Möglichkeit des leichten Daten- und Modellaustauschs ist durch die Standardisierung sicherlich gegeben, auch wenn es noch Unklarheiten in der Spezifikation gibt, die nur durch aktive Anwendung und Auslegung des Standards gefunden und behoben werden können.

Das Ziel, eine Modellierungssprache zu entwickeln, die es ermöglicht auf den Einsatz eines Prozessmodellierers, der besondere Kenntnisse in der Workflowmodellierung und im Einsatz von Workflowmanangementsystemen hat, zu verzichten und die Modellierung durch Geschäftsverantwortliche allein

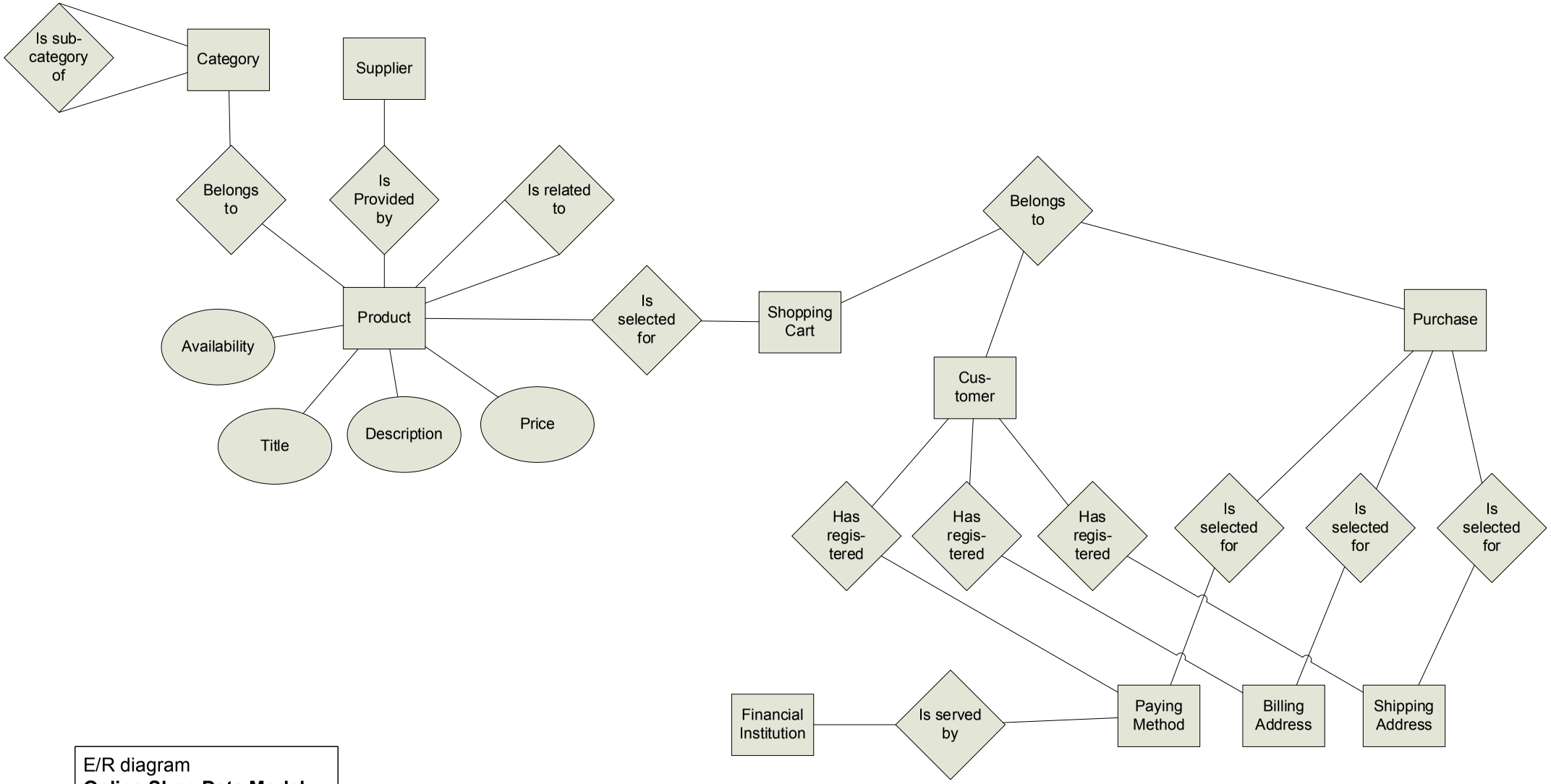
durchzuführen halte ich für verfehlt und ich bezweifele, dass die Vorliegende, nicht durch eine formale Spezifikation abgedeckte Definition des Business Process Modeling Notation Standards ausreicht, um eine Abbildung von BPMN auf BPEL zu gewährleisten.

Literaturverzeichnis

[1] BPMN 1.0 Specification , Business Process Management Initiative – 3. Mai 2004
[2]Introduction to BPMN, Stephen A. White – Mai 2004
[3] Workflow Patterns with BPMN and UML, Stephen A. White – Januar 2004
[4] BPMN and Business Process Management, Jog Roj, Martin Owen – September 2003
[5] Amazon.com Homepage http://www.amazon.com/
[6] Amazon Deutschland Homepage http://www.amazon.de/
[7] Apache Jakarta Struts http://struts.apache.org/
[8] Business Process Execution Language for Webservices 1.1 Specification, Microsoft, IBM, SAP, BEA, Siebel Systems – 5.Mai 2003

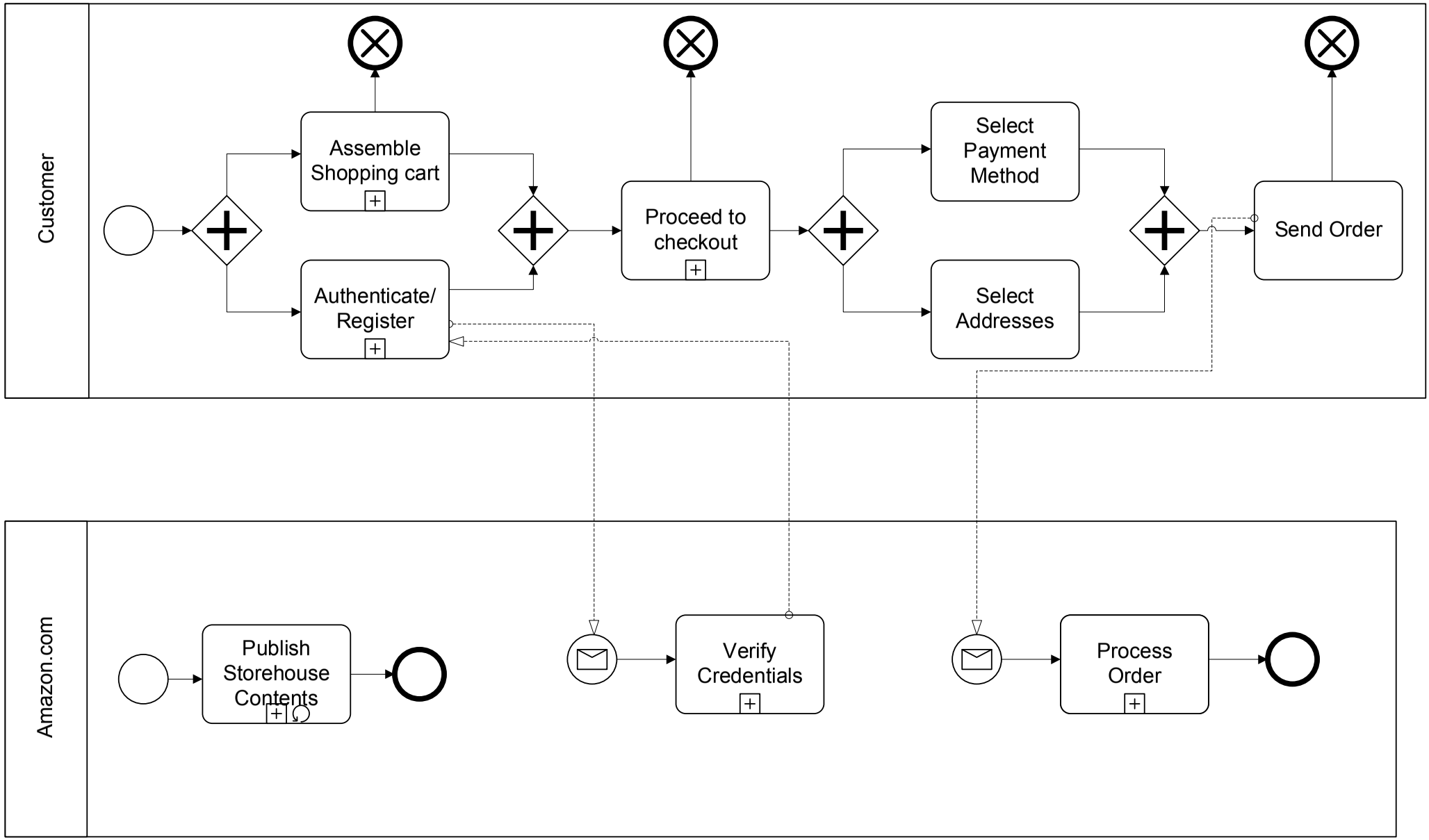


Block diagram **Szenario Participants**	
Seminar Processmodeling	
Lars Trieloff	05-06-2004

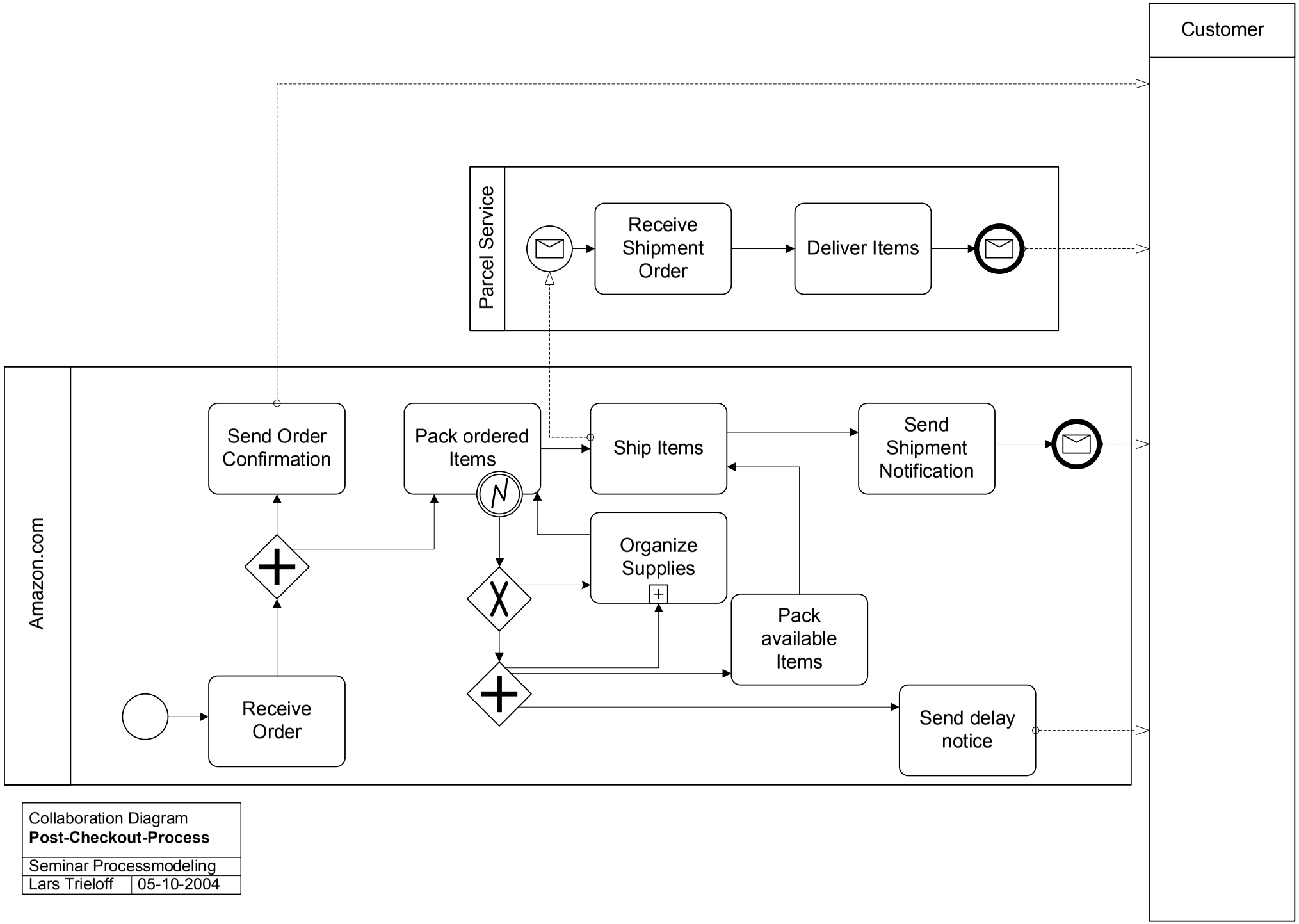


E/R diagram
Online Shop Data Model

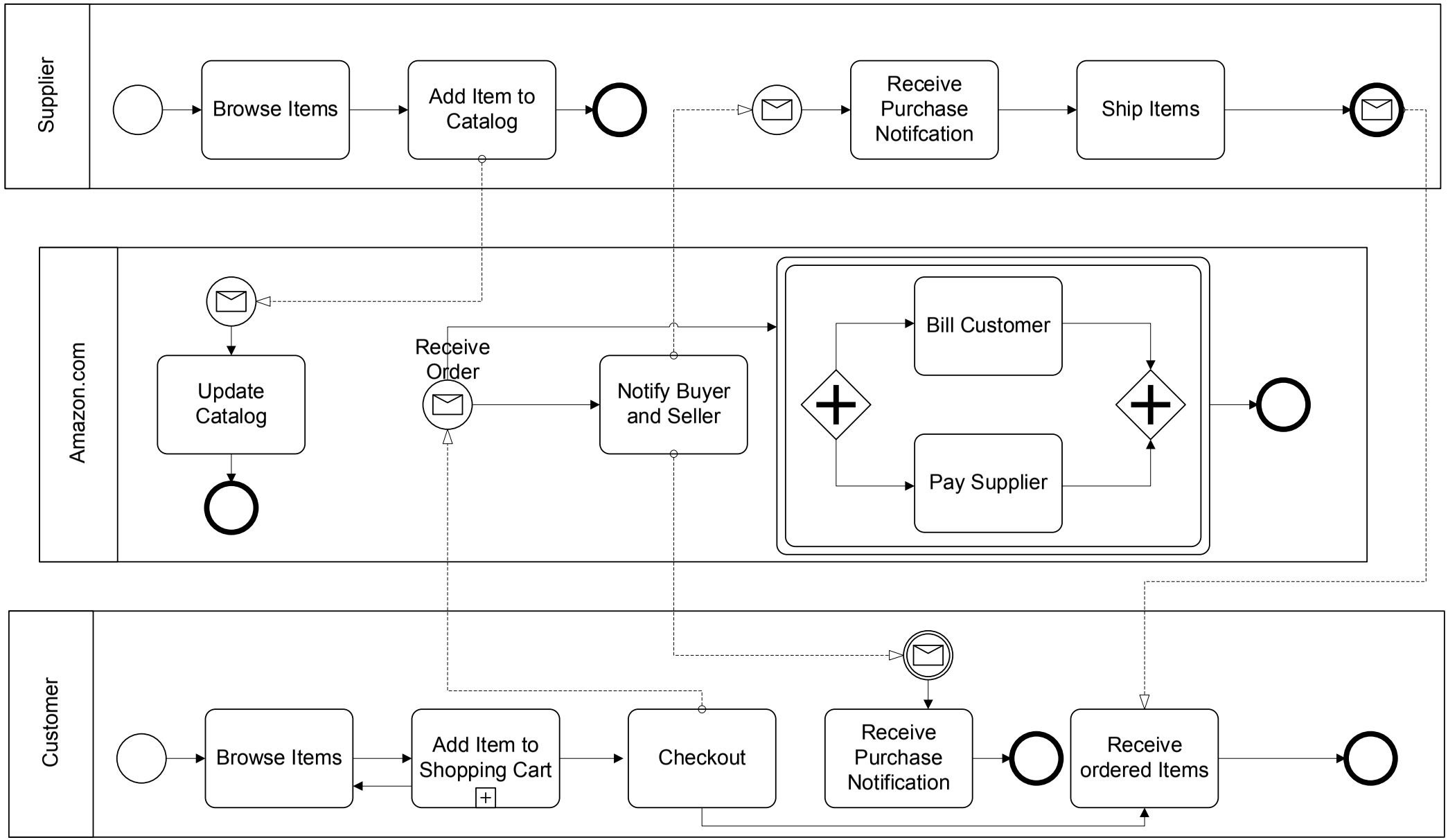
Seminar Processmodeling	
Lars Trieloff	05-07-2004



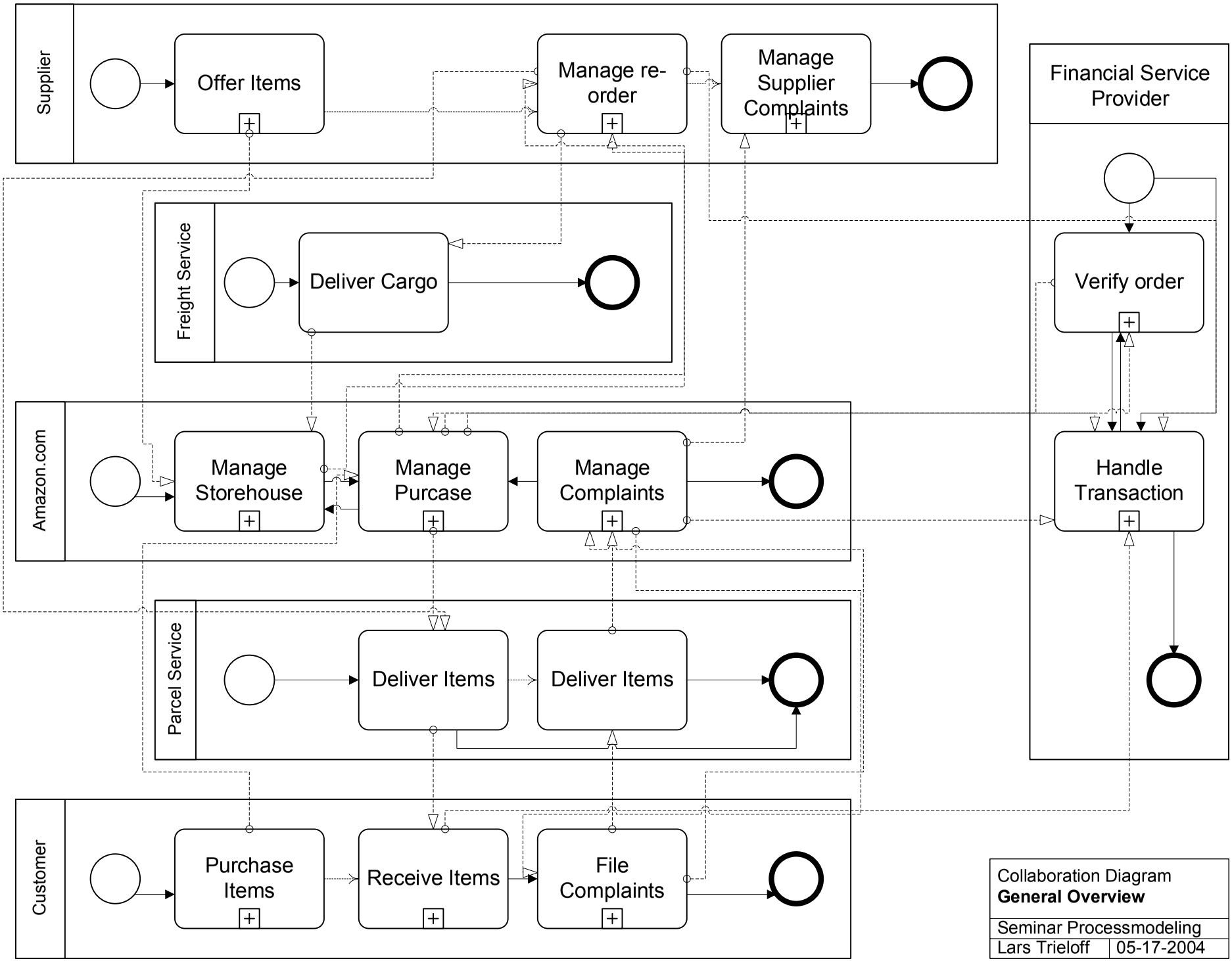
Collaboration Diagram **Checkout Overview**	
Seminar Processmodeling	
Lars Trieloff	05-09-2004

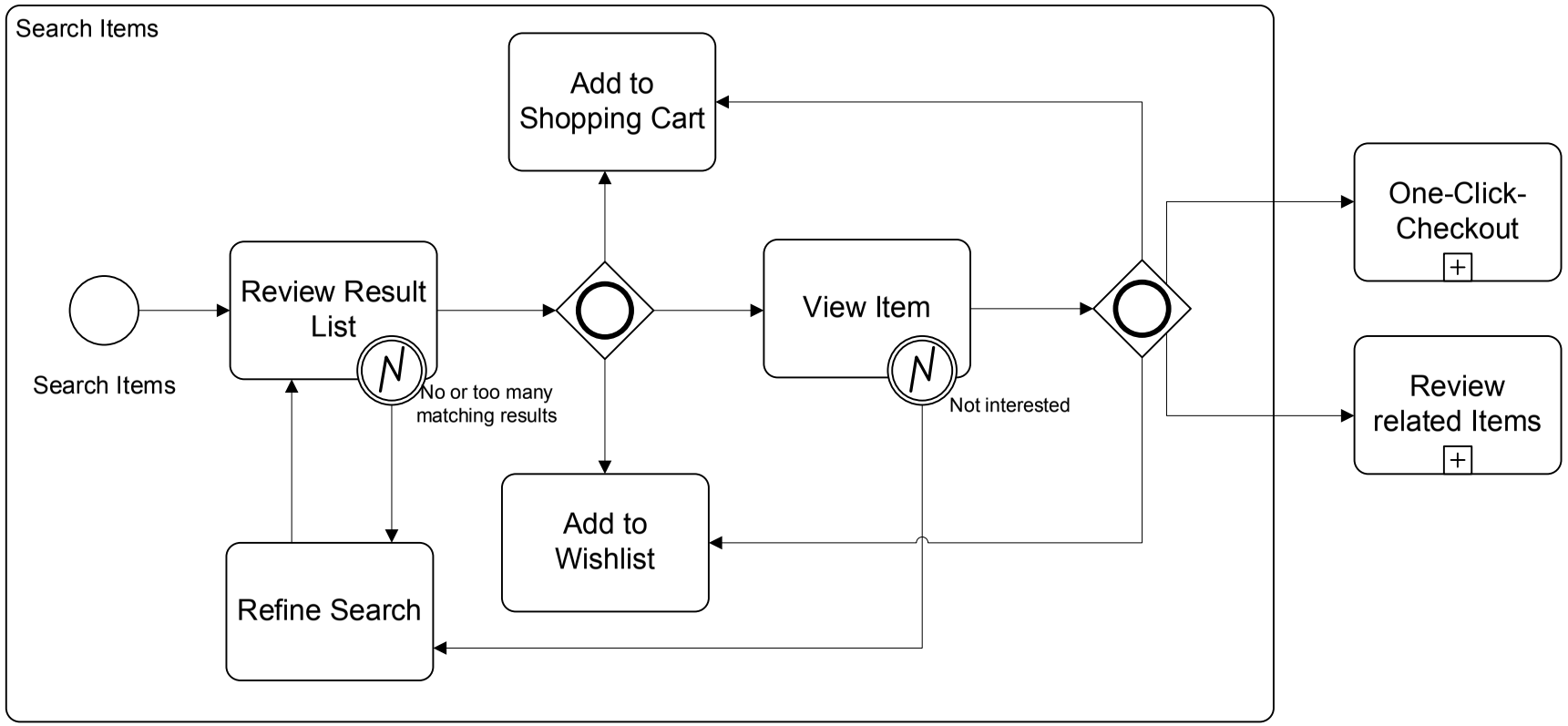


Collaboration Diagram **Post-Checkout-Process**	
Seminar Processmodeling	
Lars Trieloff	05-10-2004

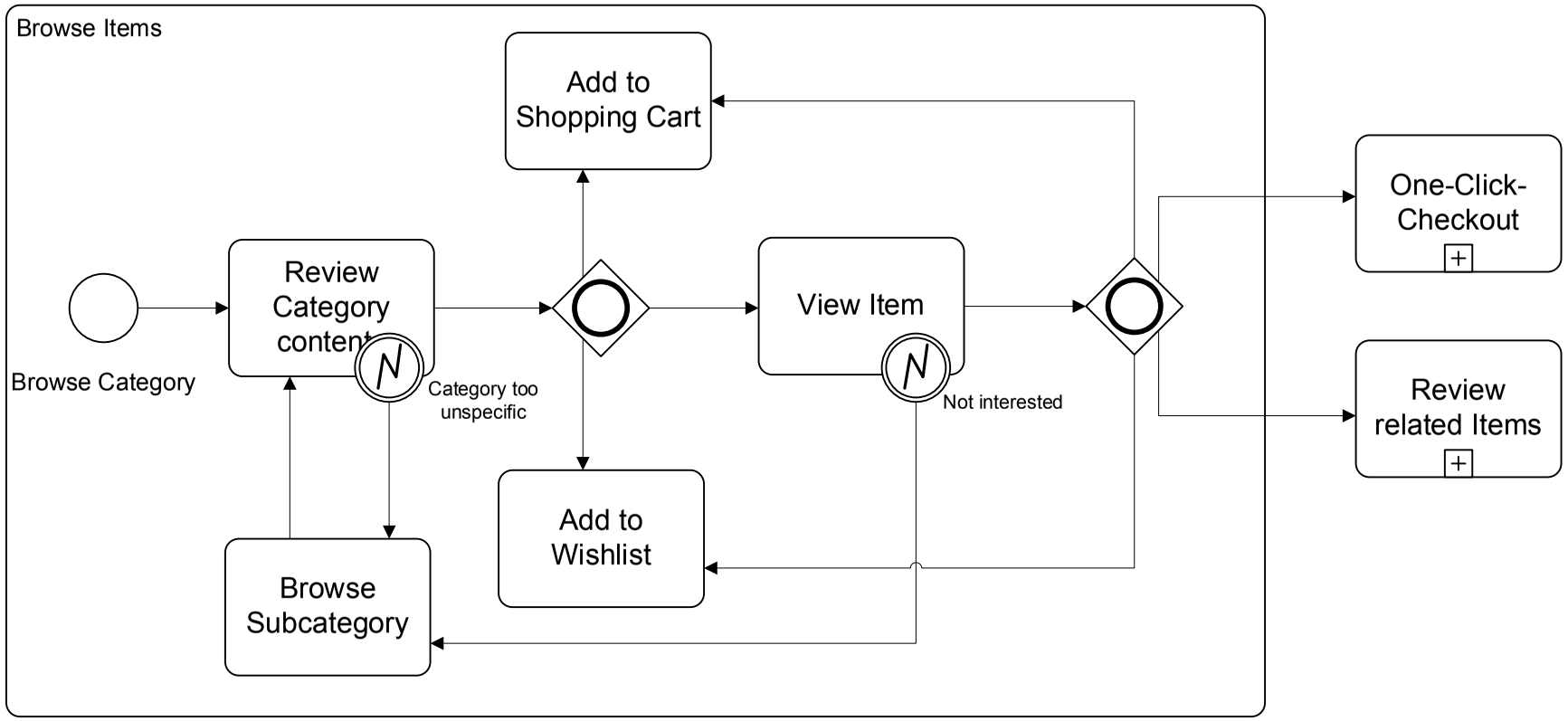


Collaboration Diagram **Marketplace Offering**	
Seminar Processmodeling	
Lars Trieloff	05-11-2004

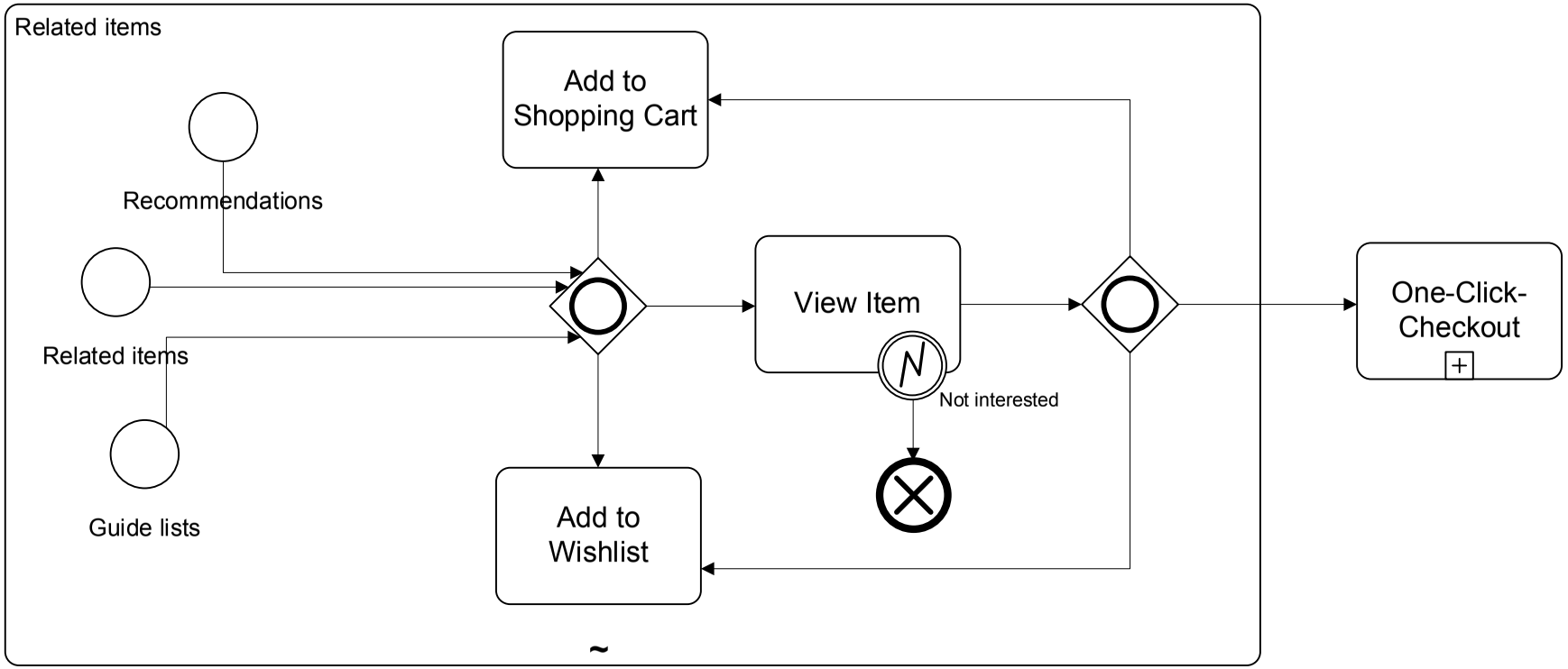

Supplier
Offer Items
Manage re-order
Manage Supplier Complaints
Financial Service Provider
Freight Service
Deliver Cargo
Verify order
Amazon.com
Manage Storehouse
Manage Purcase
Manage Complaints
Handle Transaction
Parcel Service
Deliver Items
Deliver Items
Customer
Purchase Items
Receive Items
File Complaints
Collaboration Diagram
General Overview
Seminar Processmodeling
Lars Trieloff
05-17-2004



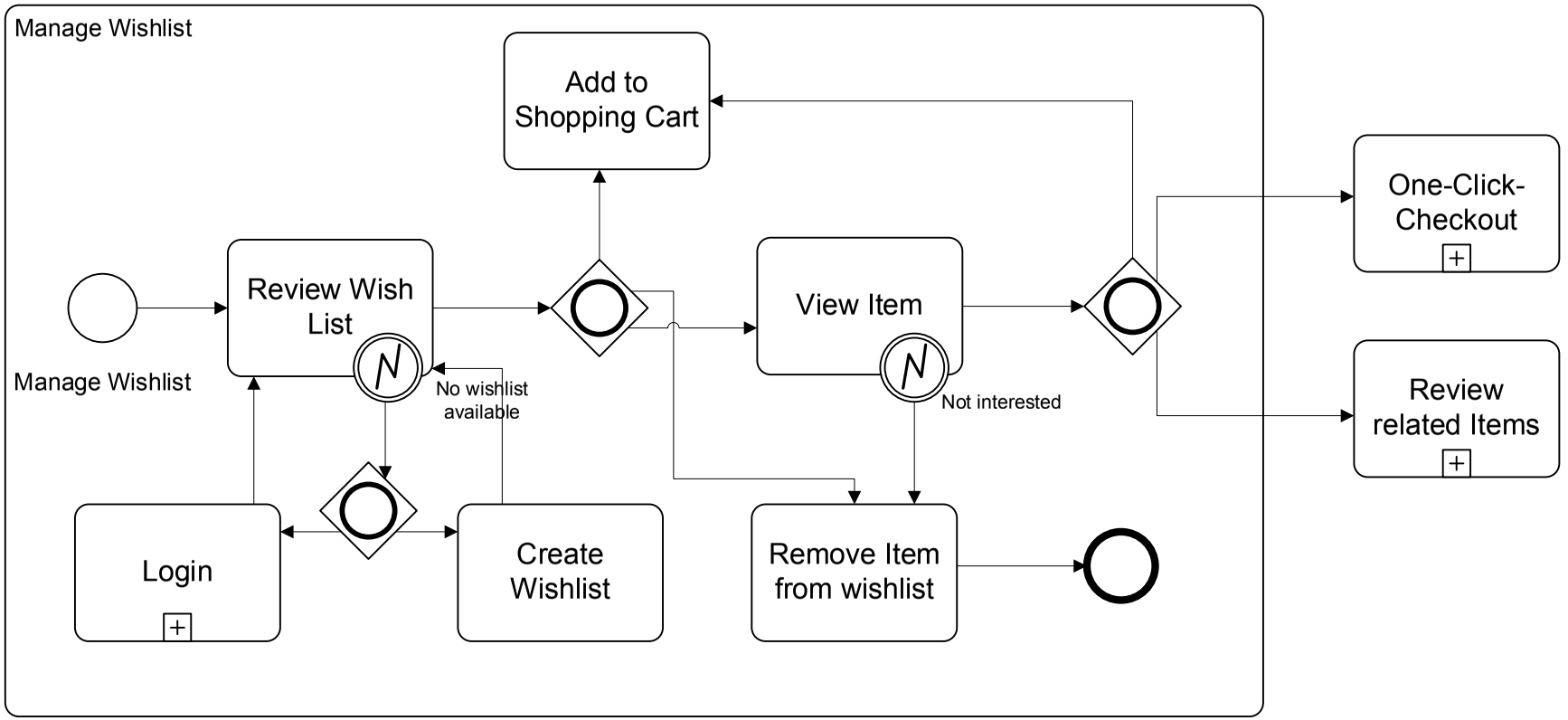
Expanded Sub-Process **Search Items**	
Seminar Processmodeling	
Lars Trieloff	05-17-2004



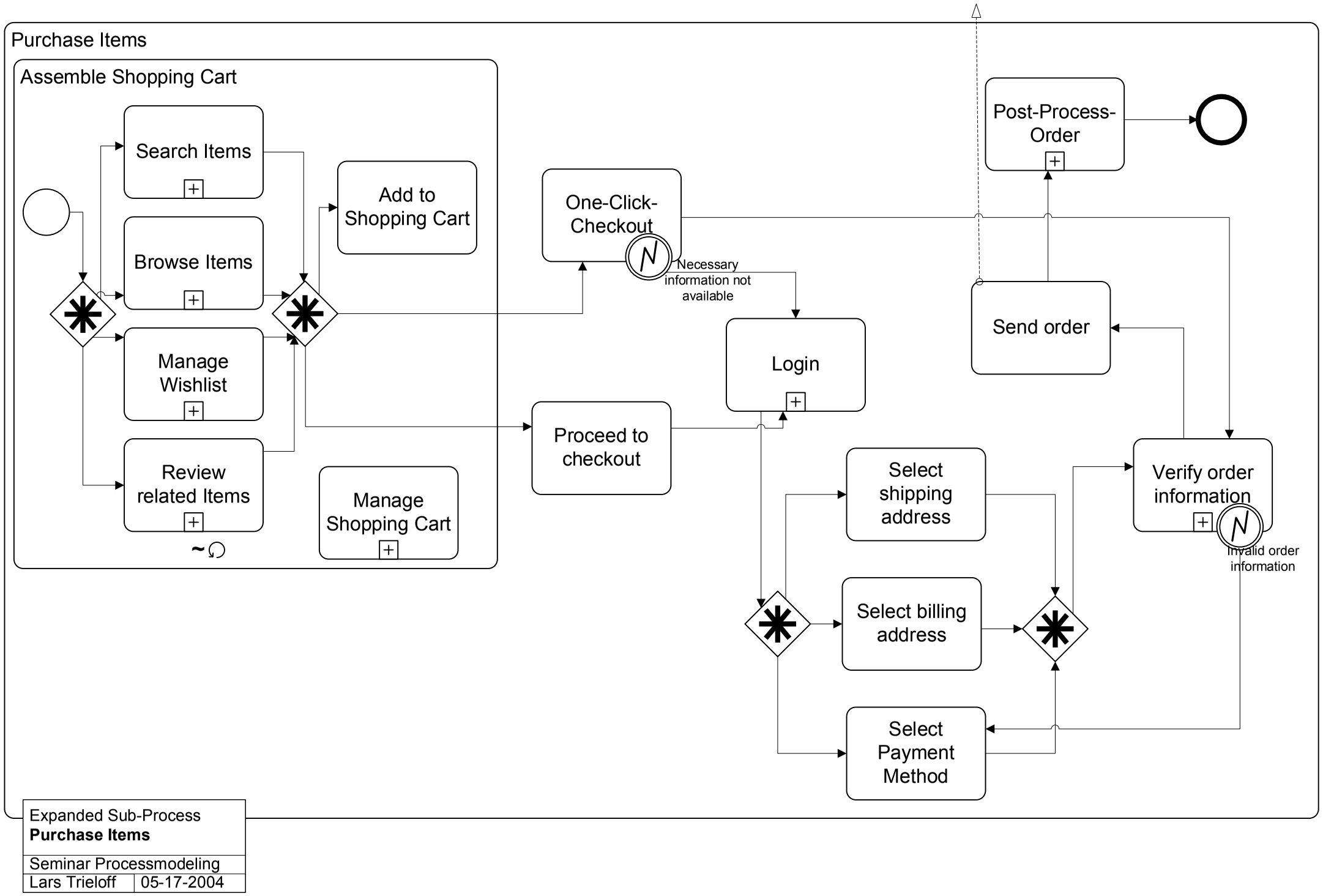
Expanded Sub-Process **Browse Items**	
Seminar Processmodeling	
Lars Trieloff	05-17-2004



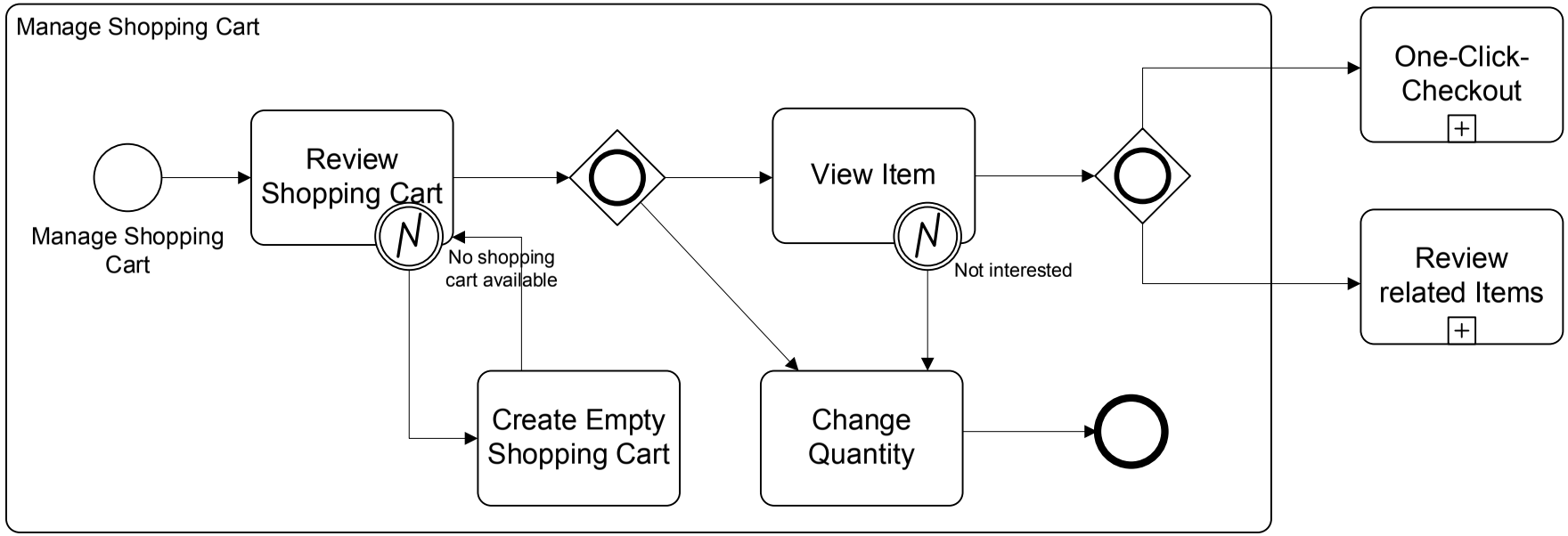
Expanded Sub-Process **Related Items**	
Seminar Processmodeling	
Lars Trieloff	05-17-2004



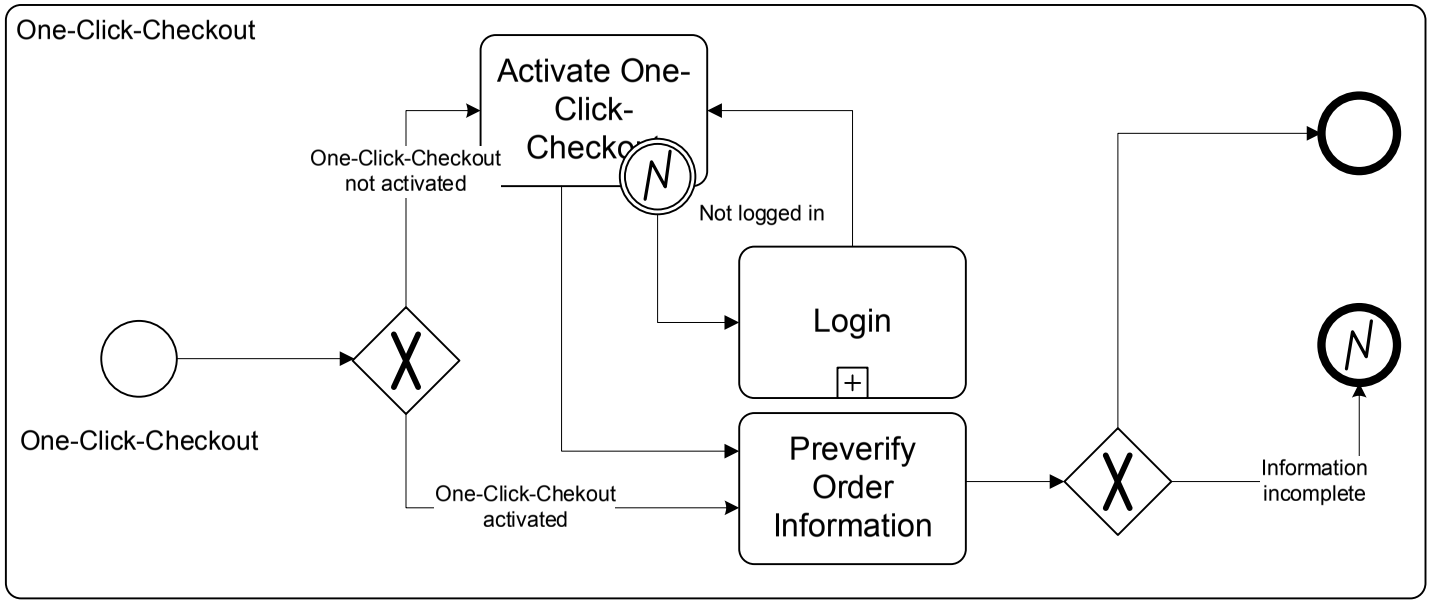
Expanded Sub-Process **Manage Wishlist**	
Seminar Processmodeling	
Lars Trieloff	05-17-2004



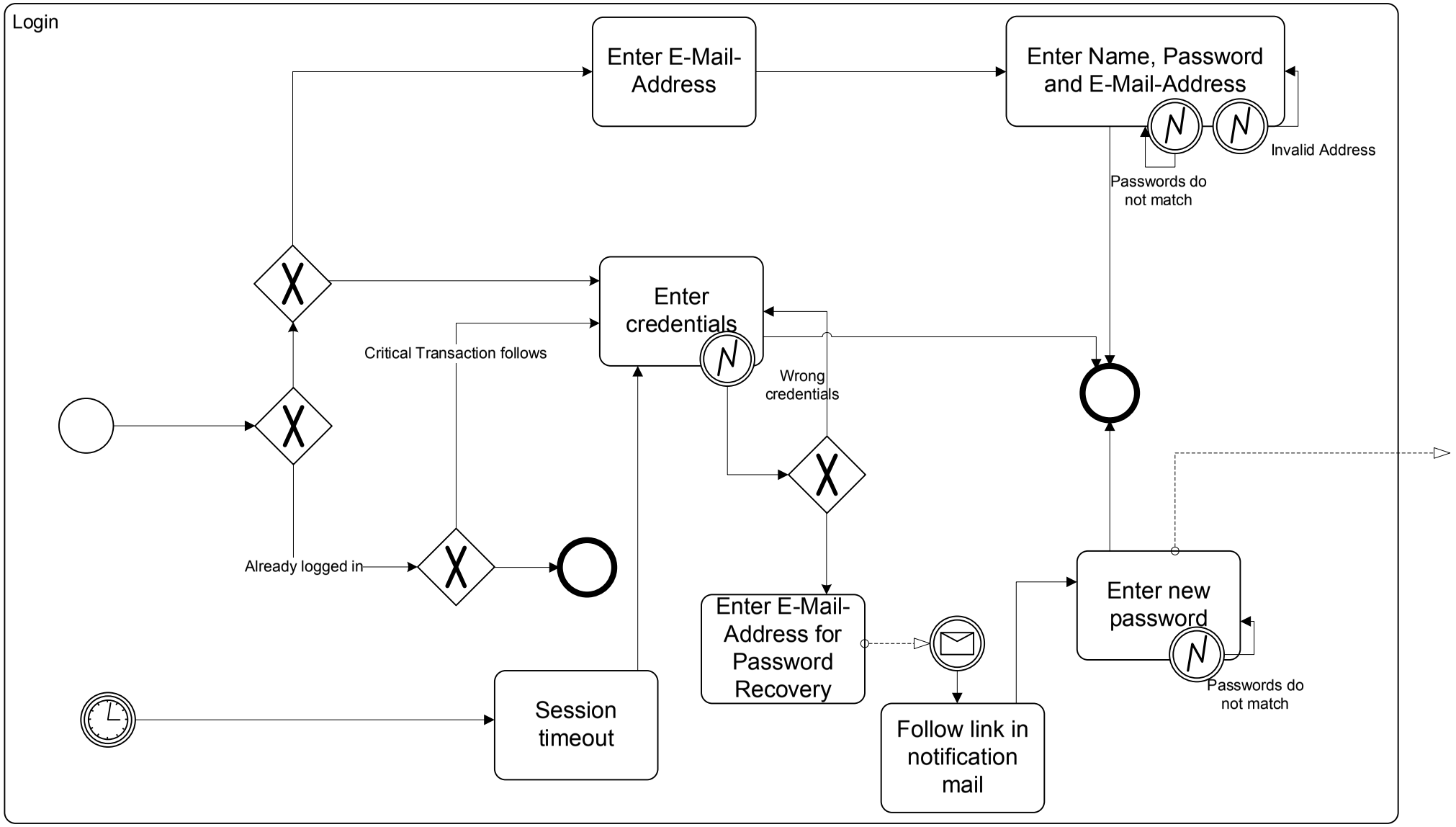
Expanded Sub-Process **Purchase Items**	
Seminar Processmodeling	
Lars Trieloff	05-17-2004



Expanded Sub-Process **Manage Shopping Cart**	
Seminar Processmodeling	
Lars Trieloff	05-18-2004



Expanded Sub-Process **One-Click-Checkout**	
Seminar Processmodeling	
Lars Trieloff	05-18-2004

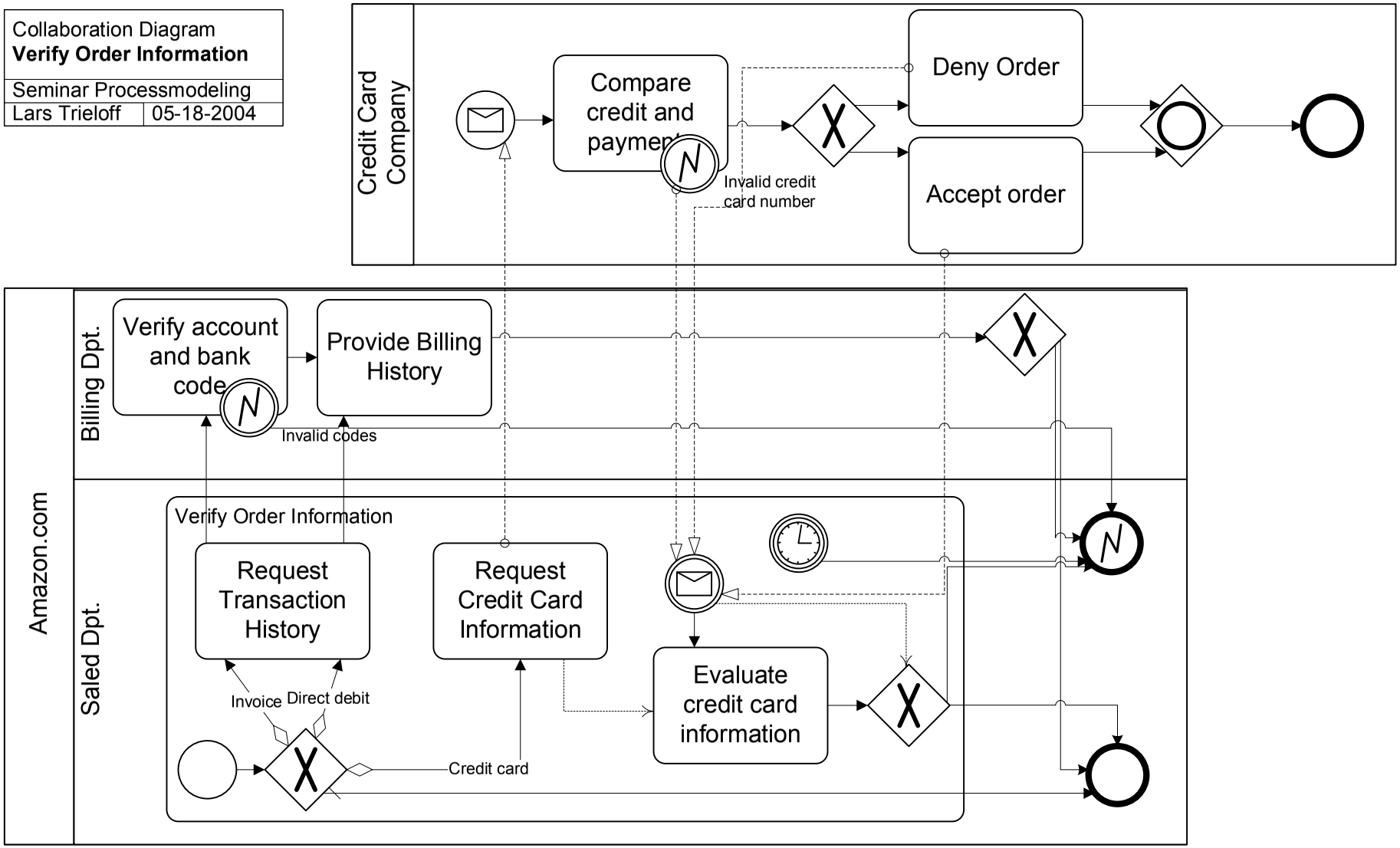


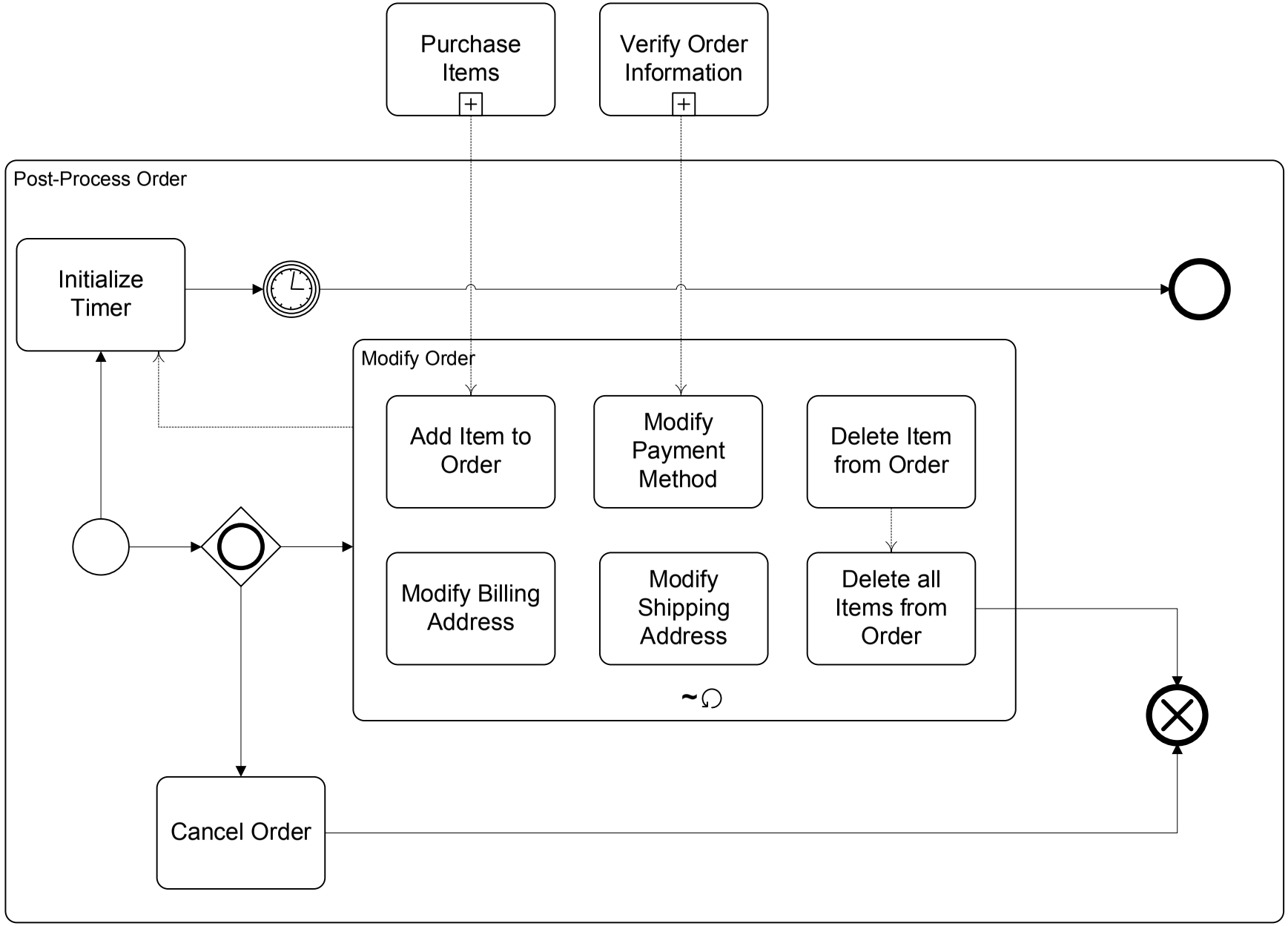
Expanded Sub-Process **Login**	
Seminar Processmodeling	
Lars Trieloff	05-18-2004

Collaboration Diagram
Verify Order Information

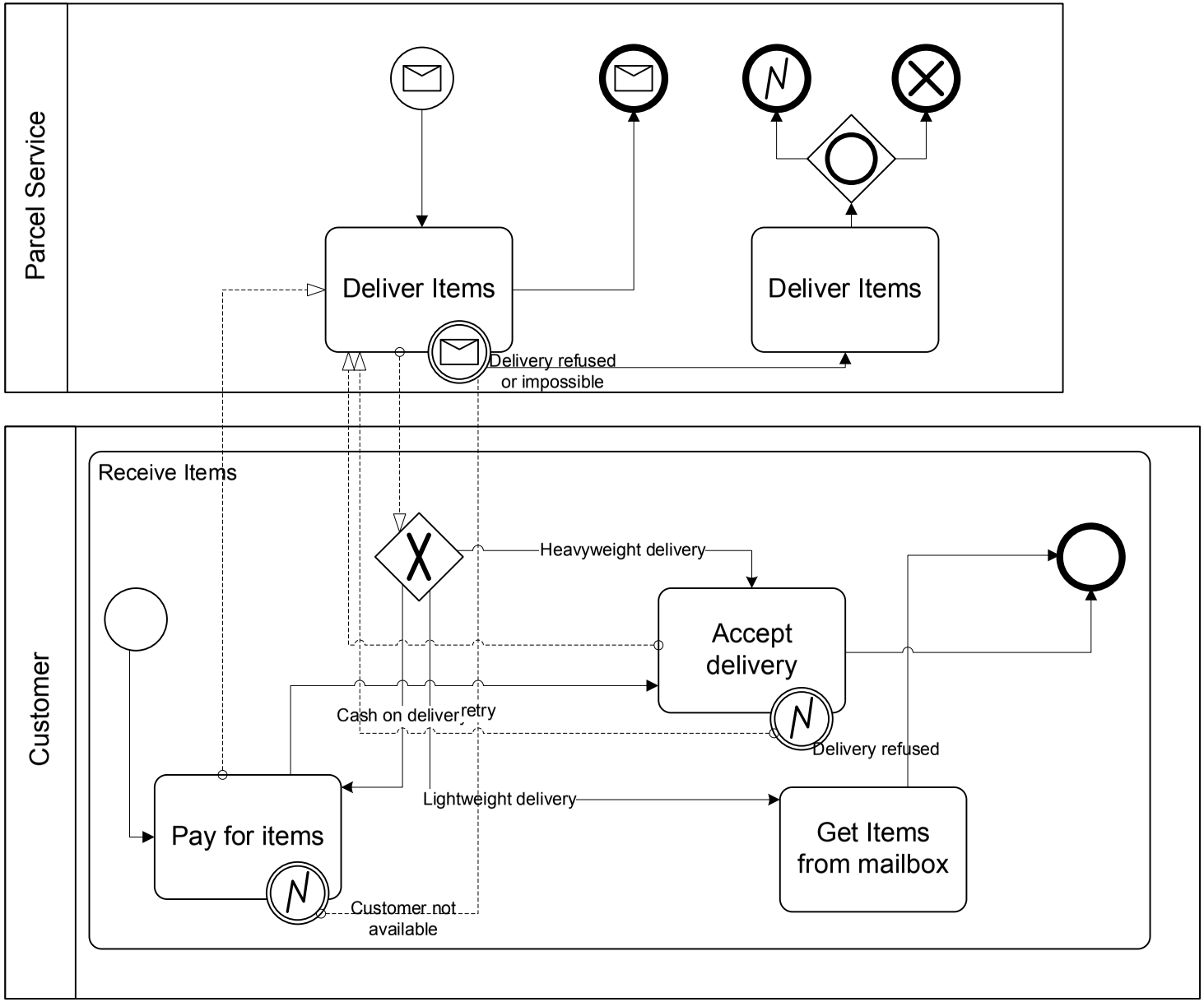
Seminar Processmodeling

Lars Trieloff | 05-18-2004

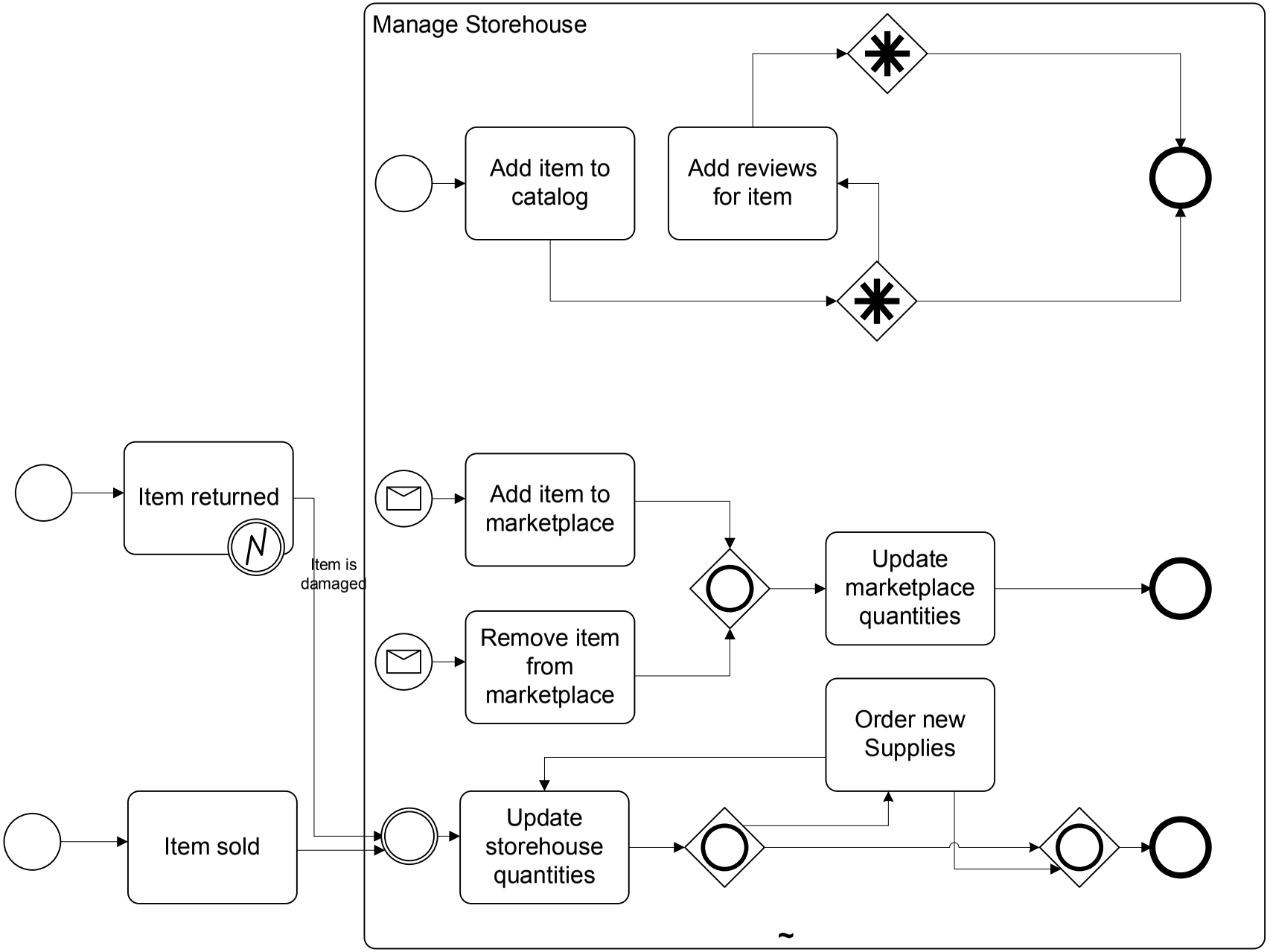




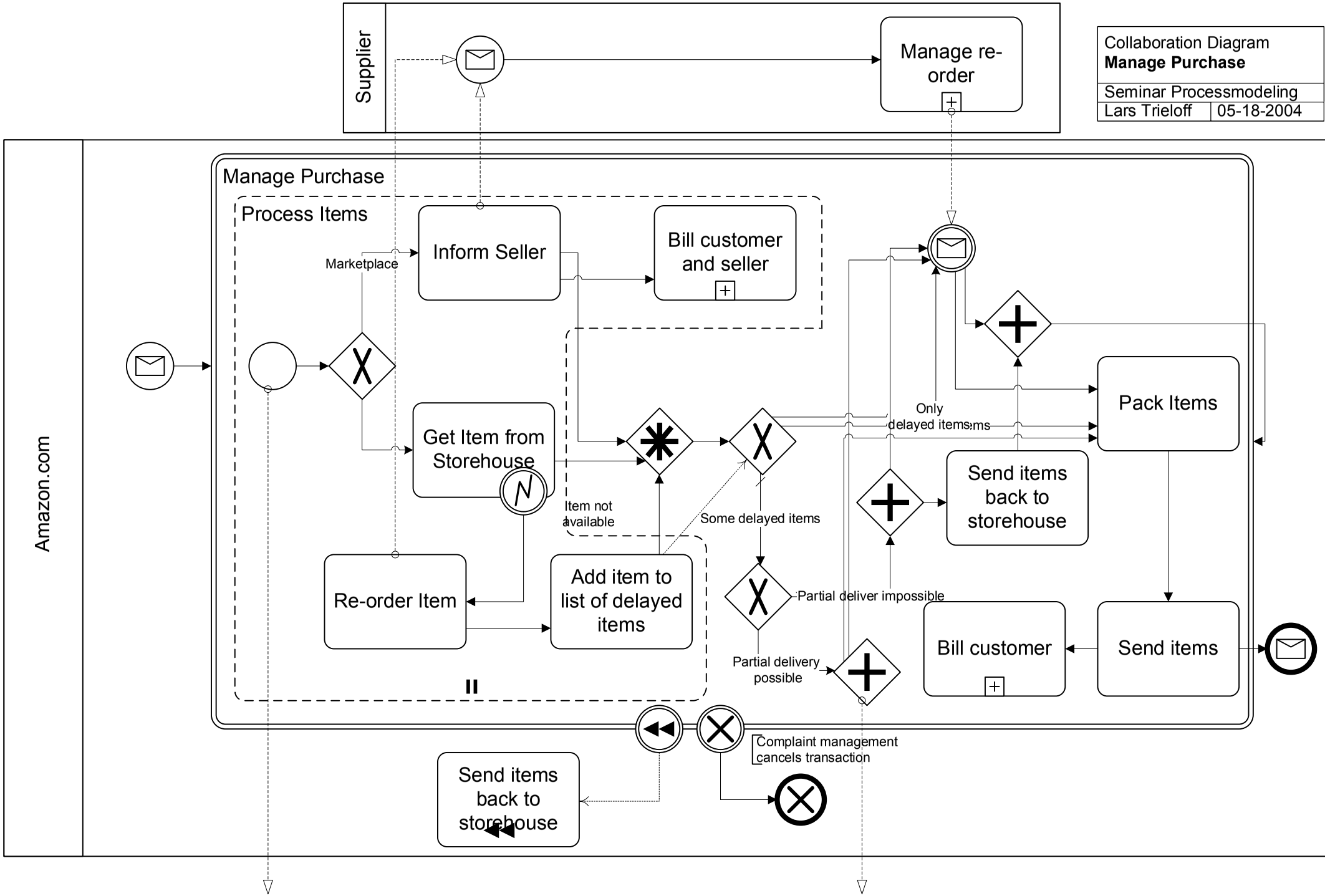
Expanded Sub-Process **Post-Process Order**	
Seminar Processmodeling	
Lars Trieloff	05-18-2004

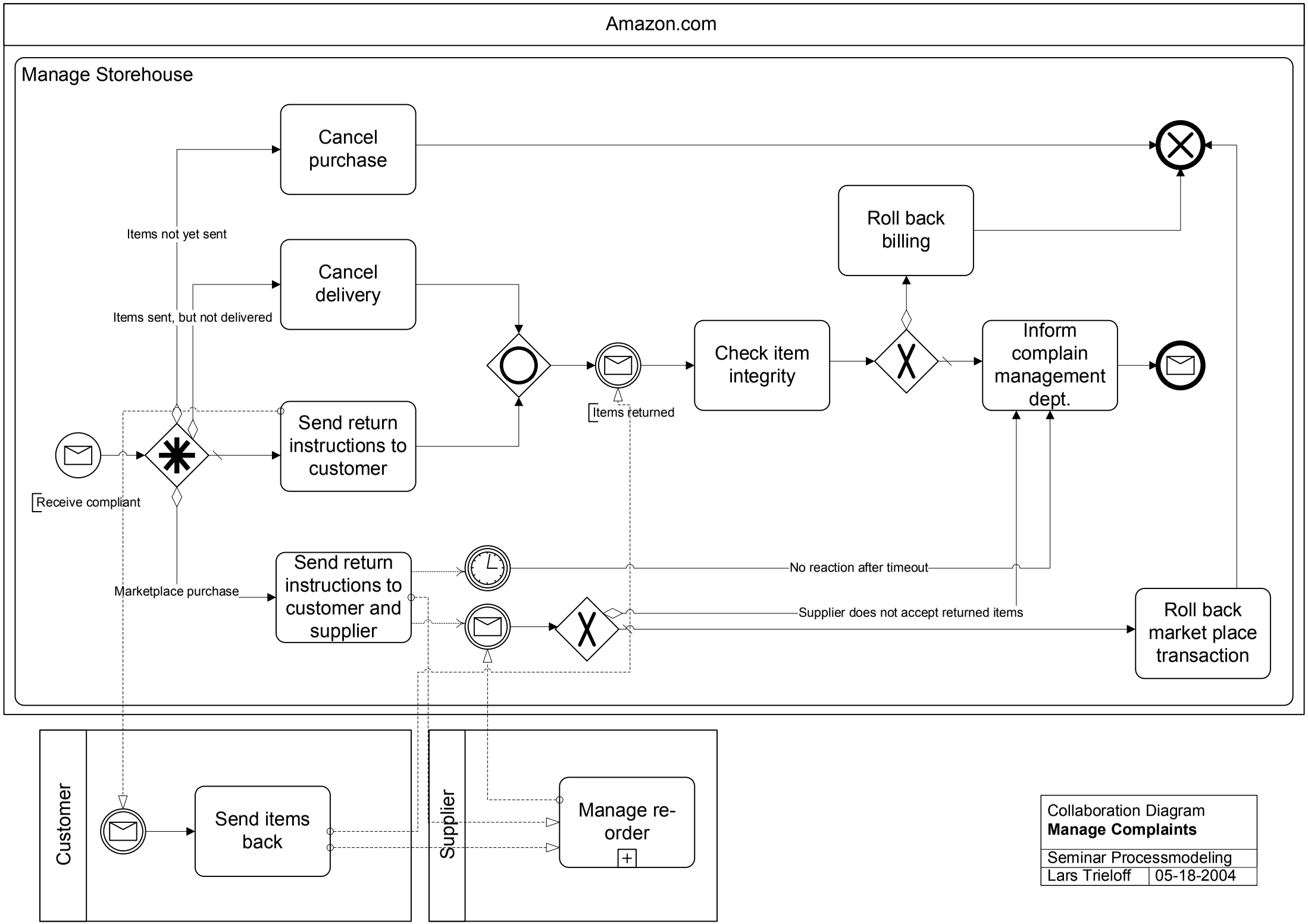


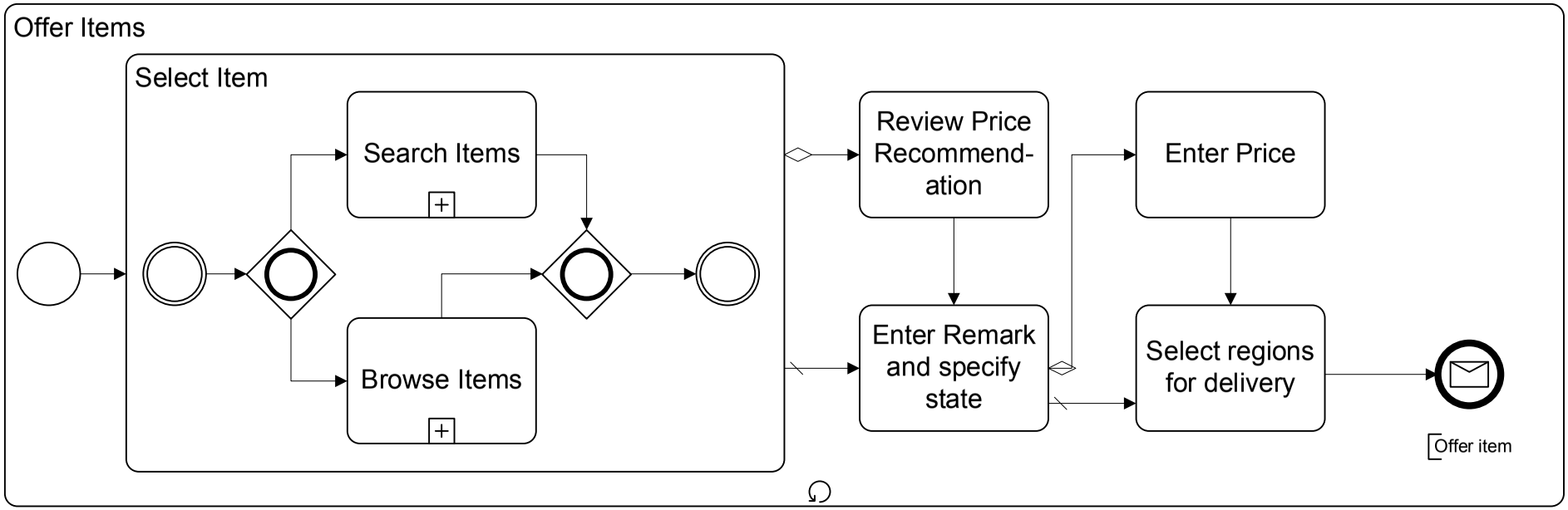
Collaboration Diagram **Receive Items**	
Seminar Processmodeling	
Lars Trieloff	05-18-2004



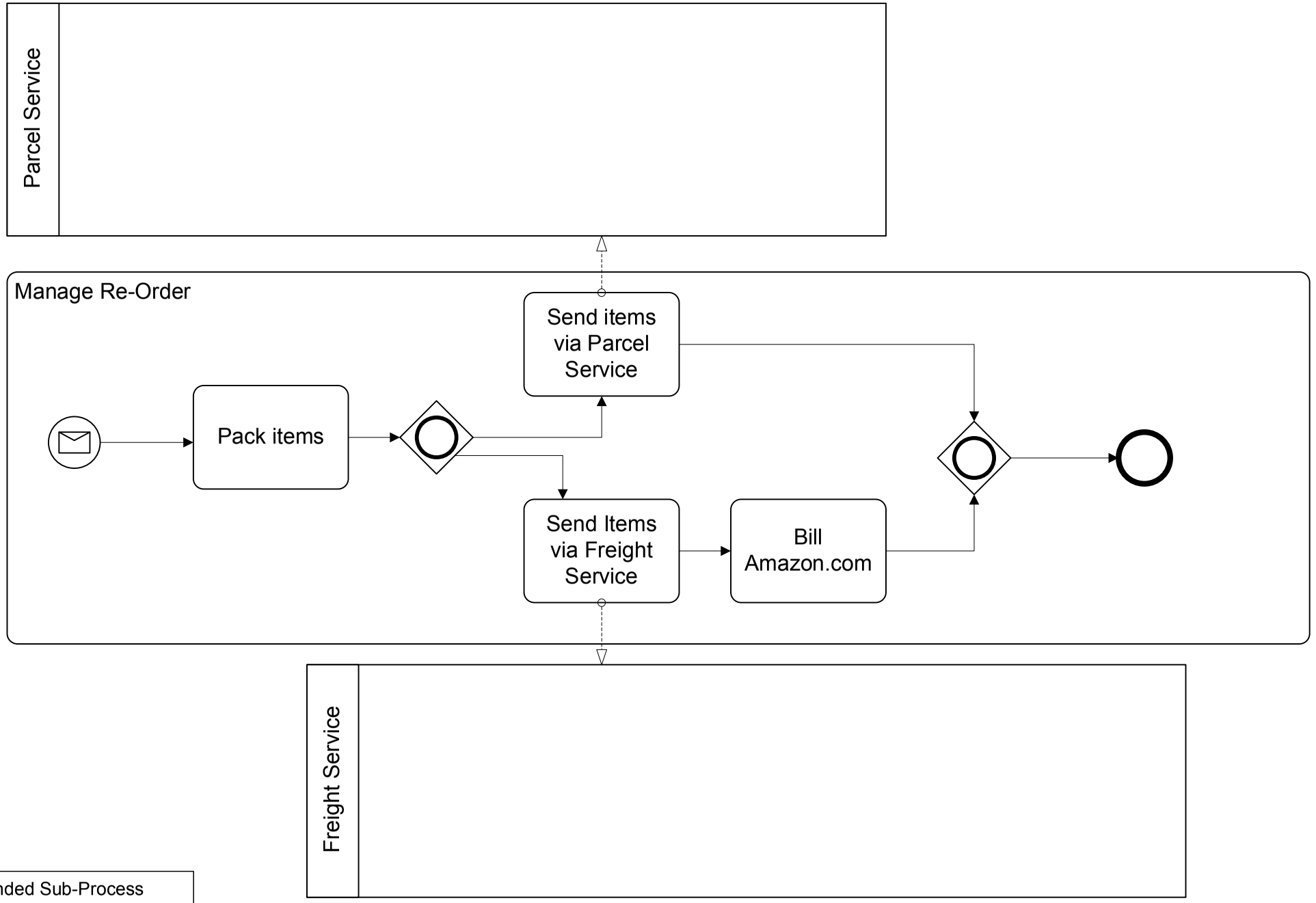
Expanded Sub-Process **Manage Storehouse**	
Seminar Processmodeling	
Lars Trieloff	05-18-2004


Supplier
Manage re-order
Collaboration Diagram
Manage Purchase
Seminar Processmodeling
Lars Trieloff
05-18-2004
Amazon.com
Manage Purchase
Process Items
Marketplace
Inform Seller
Bill customer and seller
Get Item from Storehouse
Item not available
Re-order Item
Add item to list of delayed items
Some delayed items
Only delayed items
Pack Items
Send items back to storehouse
Partial deliver impossible
Partial delivery possible
Bill customer
Send items
Complaint management cancels transaction
Send items back to storehouse

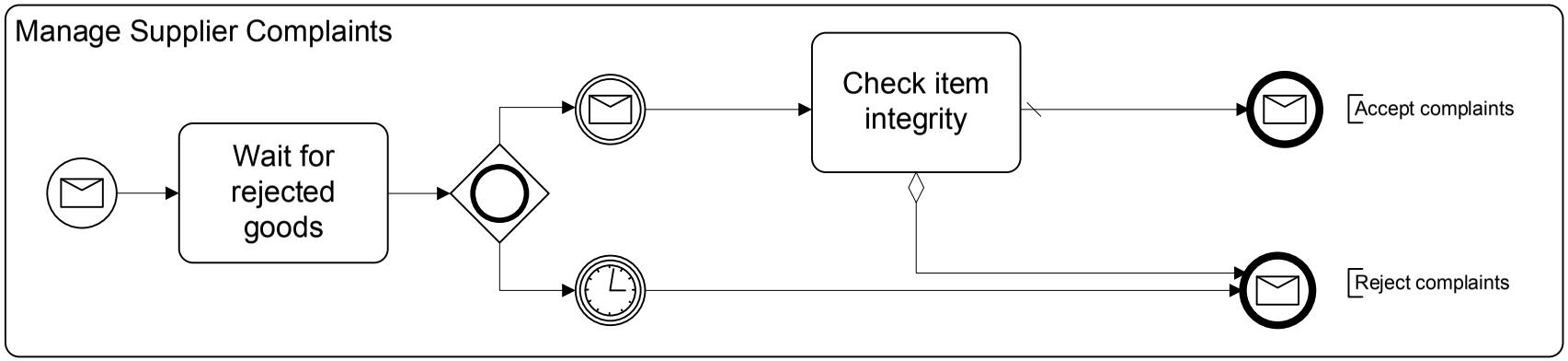

Amazon.com
Manage Storehouse
Cancel purchase
Items not yet sent
Cancel delivery
Items sent, but not delivered
Send return instructions to customer
Receive compliant
Marketplace purchase
Send return instructions to customer and supplier
Items returned
Check item integrity
Roll back billing
Inform complain management dept.
No reaction after timeout
Supplier does not accept returned items
Roll back market place transaction
Customer
Send items back
Supplier
Manage re-order
Collaboration Diagram
Manage Complaints
Seminar Processmodeling
Lars Trieloff
05-18-2004



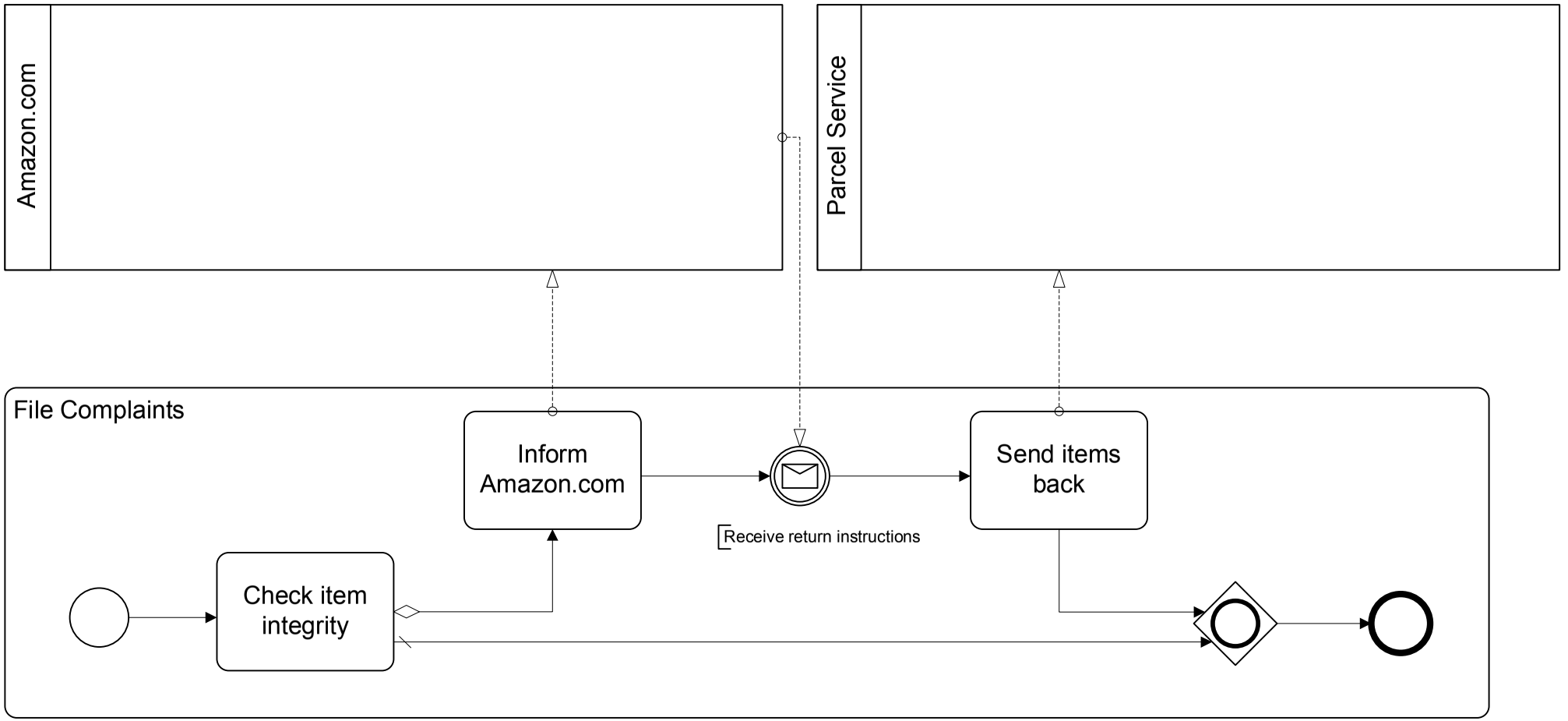
Expanded Sub-Process **Offer Items**	
Seminar Processmodeling	
Lars Trieloff	05-21-2004



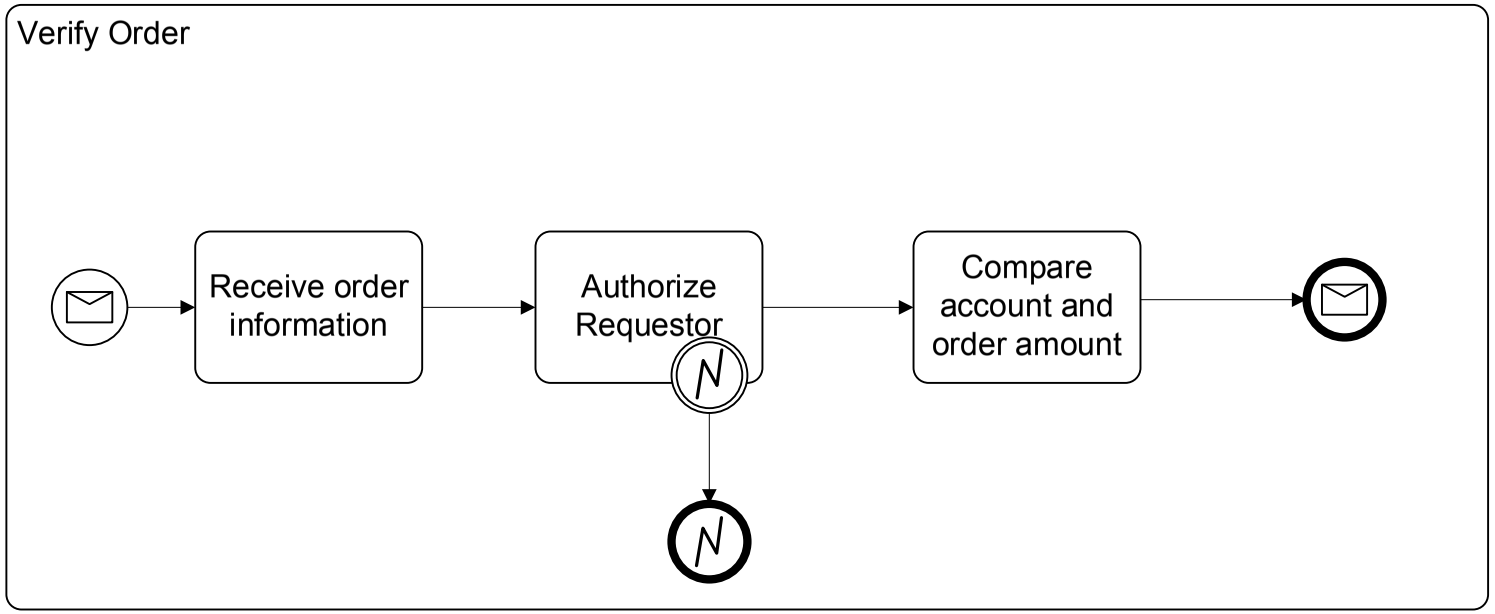
Expanded Sub-Process **Manage Re-Order**	
Seminar Processmodeling	
Lars Trieloff	05-21-2004



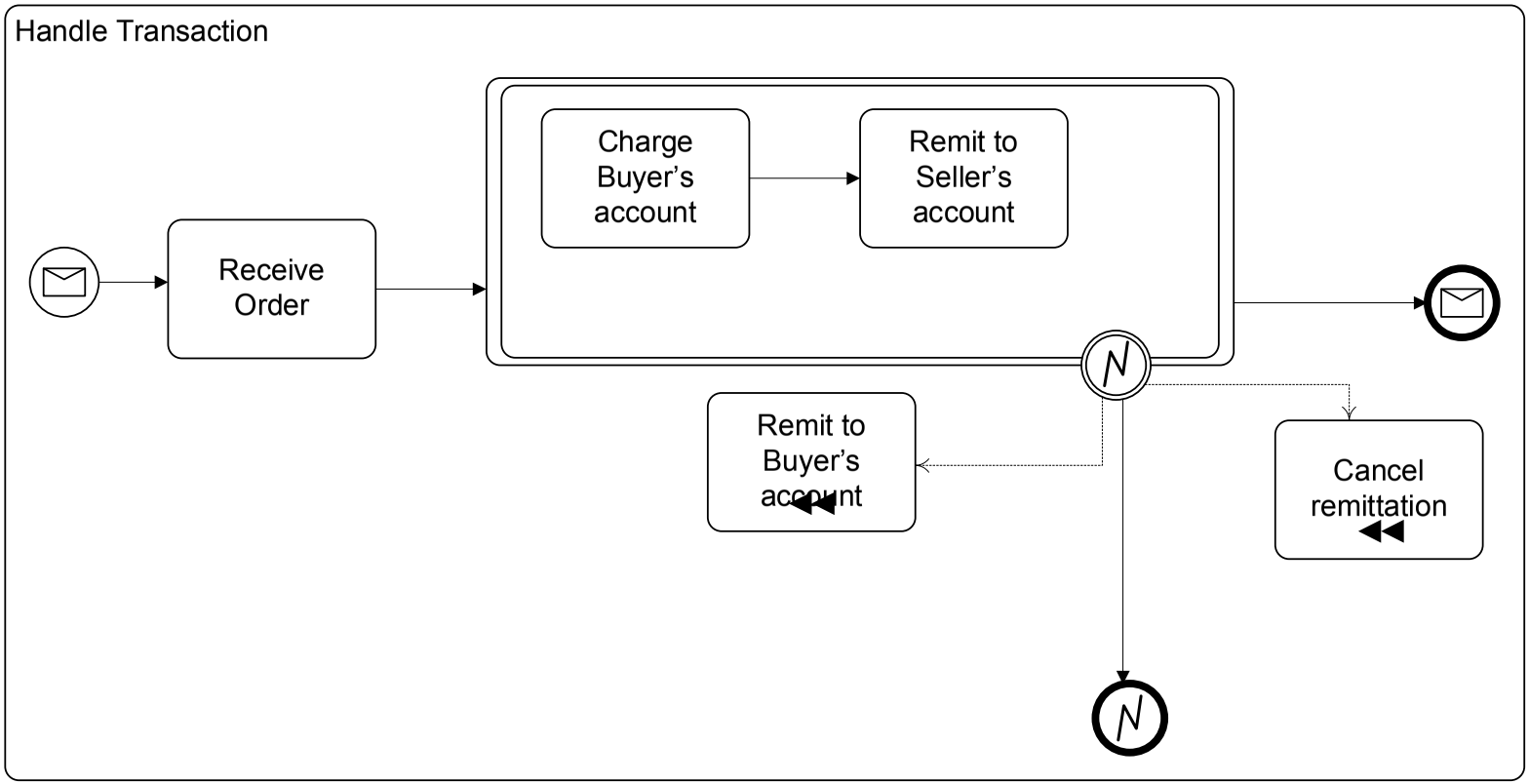
Expanded Sub-Process **Manage Supplier Complaints**	
Seminar Processmodeling	
Lars Trieloff	05-21-2004



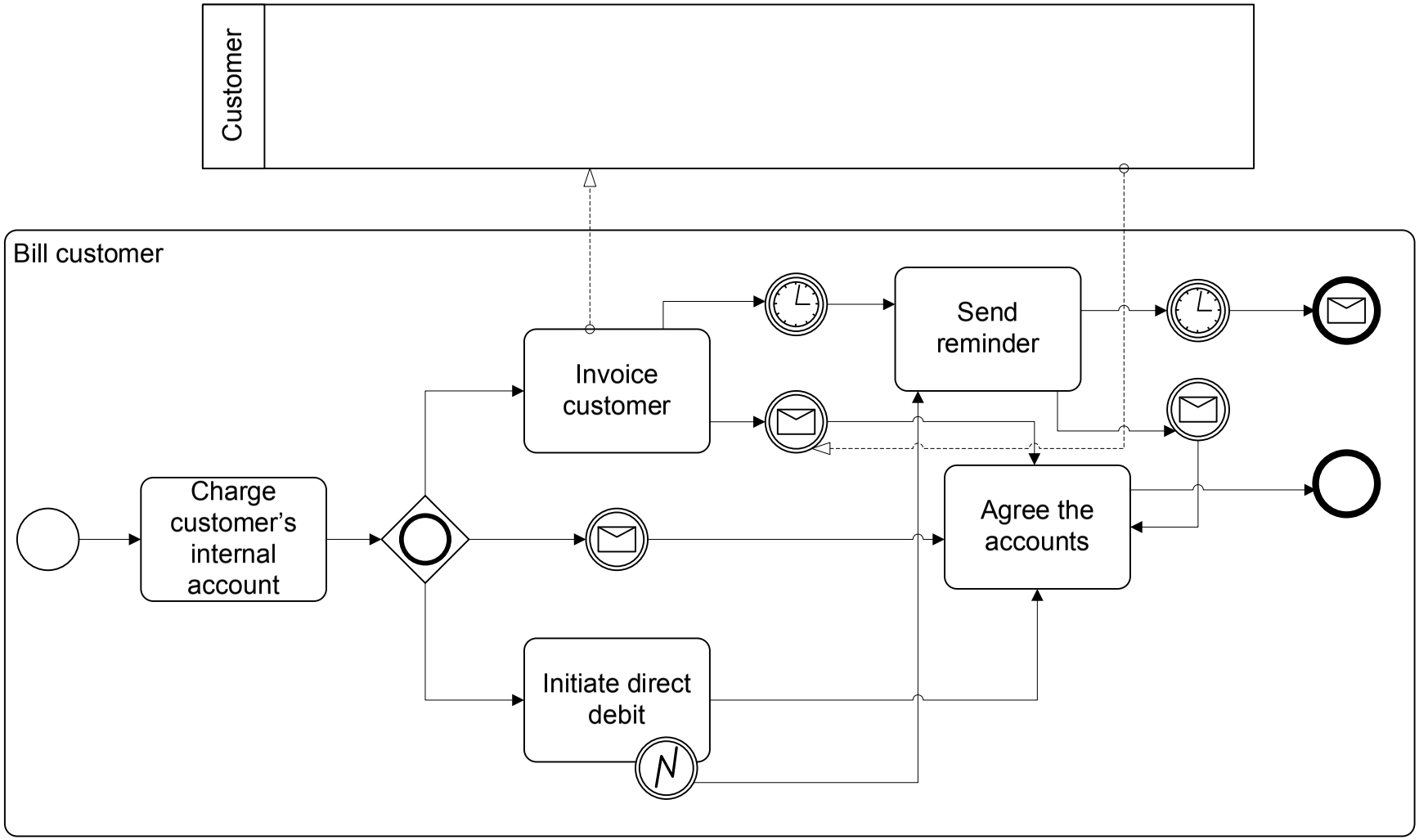
Collaboration Diagram **File Compliants**	
Seminar Processmodeling	
Lars Trieloff	05-21-2004



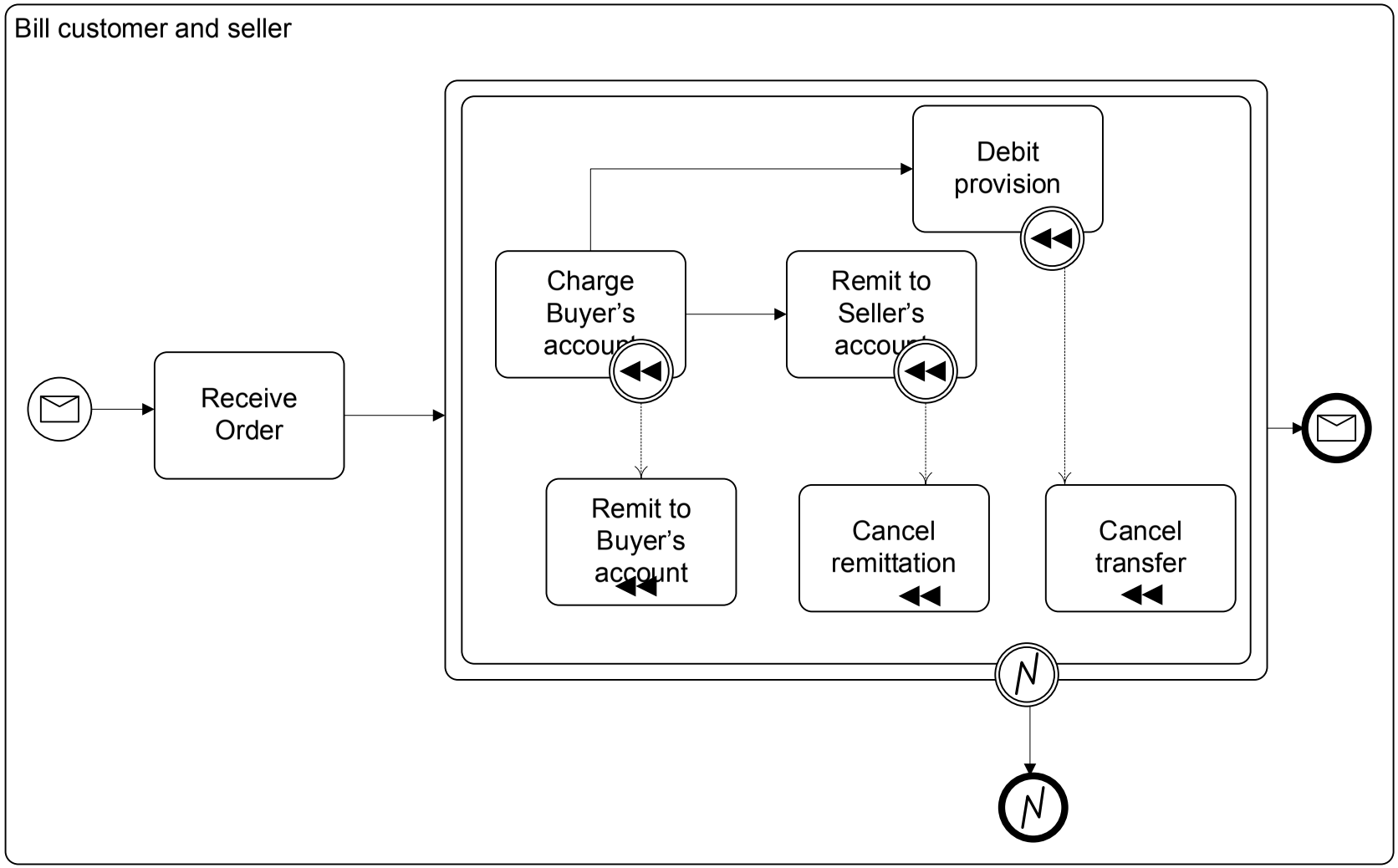
Expanded Sub-Process **Verify Order**	
Seminar Processmodeling	
Lars Trieloff	05-22-2004



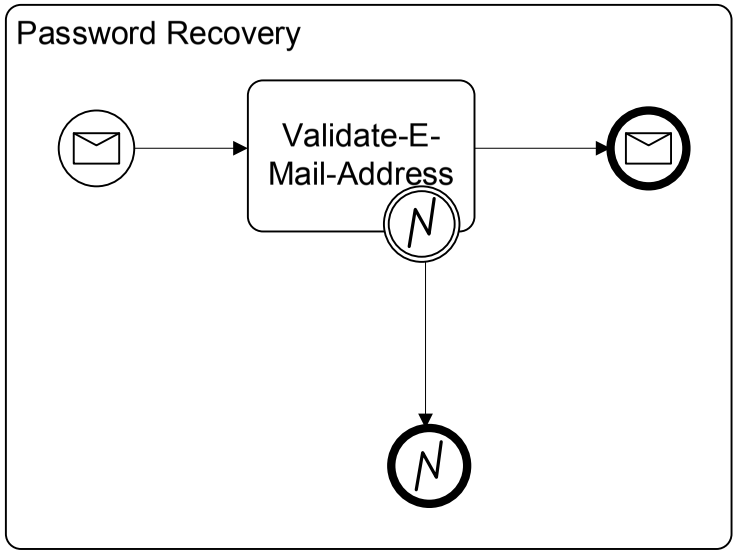
Expanded Sub-Process **Handle Transaction**	
Seminar Processmodeling	
Lars Trieloff	05-22-2004



Expanded Sub-Process **Bill Customer**	
Seminar Processmodeling	
Lars Trieloff	05-22-2004



Expanded Sub-Process **Bill customer and seller**	
Seminar Processmodeling	
Lars Trieloff	05-22-2004



Expanded Sub-Process **Password Recovery**	
Seminar Processmodeling	
Lars Trieloff	05-22-2004

Otto.de – Modellierung eines WebShops

Autor: Thomas Hille

Email: webmaster@thomas-hille.com

Die Modellierung eines Systems, insbesondere eines WebShops ist ein komplexer Vorgang, für den eine Vielzahl von Schritten notwendig ist. Am Beispiel von Otto.de soll erläutert werden, wieso die Kombination von Navigations-, Use Case- und BPMN-Diagrammen eine gute Möglichkeit darstellt dies zu erreichen. Es wird Aufschluss darüber gegeben, wo die Stärken der BPMN liegen und wieso die Modellierungssprache YAWL nicht geeignet ist, einen WebShop zu modellieren.

1 Einleitung

Hat man die Aufgabe ein komplexes System zu modellieren, steht man zunächst vor der Frage, wo man beginnen soll. Es gibt viele mögliche Ansätze. Man kann einen statischen Diagrammtyp wählen oder man entscheidet sich für prozessmodellierende Diagramme. Mit der Hilfe von statischen Diagrammen lässt sich der Aufbau des Systems darstellen. Es werden Akteure aufgezeigt und es ist möglich ihnen Rollen zuzuordnen oder ihre Schnittstellen zu definieren. Im Gegensatz dazu bieten prozessmodellierende Diagramme die Möglichkeit dynamische Aspekte zu modellieren. Man kann kausale oder temporale Zusammenhänge aufzeigen und somit das Verständnis für das zu modellierende System fördern.

Welcher Weg bietet sich nun an, um einen WebShop zu modellieren? Ein mögliches Vorgehen ist es, zunächst mit ein paar statischen Diagrammen zu beginnen und dadurch einen Überblick über den Webshop zu gewinnen. Sobald dies gelungen ist, sollte herausgefunden werden, wer welche Rollen im System Webshop übernimmt. Anschließend sollten die einzelnen ablaufenden Geschäftsprozesse näher untersucht werden. Dies ist mit Hilfe von den bereits erwähnten prozessmodellierenden Diagrammen möglich.

Um eine konkrete Vorstellung darüber zu bekommen, wie dieser Entscheidungs- und Modellierungsprozess abläuft, soll dies nun anhand der Modellierung des Webshops Otto.de nachvollzogen werden.

2 Navigationsdiagramm

Das Navigationsdiagramm [2] ist ein statischer Diagrammtyp, mit dessen Hilfe es möglich ist, den konzeptionellen Aufbau von Internetseiten insbesondere den von WebShops grafisch darzustellen. Wie in Abbildung 1.1 zu sehen ist, tritt hierbei jede

vom Browser aufrufbare Seite als einzelner Knoten auf. Die Form der Knoten ist rechteckig. Optional ist es möglich den einzelnen Knoten auch noch Attribute und Methoden zuzuweisen, wenn eine Interaktion mit dem Nutzer möglich ist. Im Diagramm 1.1 wurde jedoch darauf zu Gunsten der Übersichtlichkeit verzichtet. Bei jedem Internetportal oder WebShop ist es möglich mit Hilfe von Links von einer Seite zur Nächsten zu navigieren. Dies ist im Navigationsdiagramm in Form von Pfeilen dargestellt, welche die mögliche Navigationsrichtung verdeutlichen sollen. Hierbei ist der Aspekt des „Back-Buttons“ des Browsers ausser Acht gelassen, da es sich dabei um keine seiteninterne Navigation handelt.

2.1 Bewertung des Diagrammtyps

Zu Beginn der Erfassung eines Internetportals oder eines WebShops erweist sich ein Navigationsdiagramm als durchaus sinnvoller Ansatz um eine Übersicht über den Umfang und die Funktionalität des zu modellierenden Objektes zu erhalten. Die Stärken liegen vor allen Dingen in der einfachen, intuitiven Interpretierbarkeit des Diagramms. Einem Betrachter wird es leicht fallen das Betrachtete zu verstehen und es gegebenenfalls umzusetzen, d.h. zielgerichtet auf die gewünschten Seiten zu navigieren, sobald er sich im Bereich des WebShops befindet.

Das Navigationsdiagramm weist jedoch auch einige entscheidende Nachteile auf. Es ist nicht möglich Daten oder Zustände zu modellieren, welche von nicht unerheblicher Bedeutung, gerade bei WebShops, sind. Auch ist es nicht möglich, sogenannte PopUp-Windows zu modellieren. Diese befinden sich meistens nicht auf einem linearen Navigationsweg, sondern tauchen parallel zum eingentlichen Navigationsprozess auf. Besonders kritisch wird es dann noch, wenn eventuell in PopUp-Windows getroffene Entscheidungen zu einem anderen möglichen Navigationsverlauf führen. Ein weiterer nicht unerheblicher Nachteil ist die Größe und Komplexität eines solchen Diagramms. Aufgrund der Vielzahl von aufrufbaren Seiten von Internetportalen ist es, selbst wenn man Seiten zu Gruppen zusammenfasst, nicht möglich ein solches Diagramm schnell zu erfassen.

2.2 Modellierung von Otto.de

Wie bereits erwähnt, wurde bei der Modellierung des WebShops Otto.de mit Hilfe des Navigationsdiagramms(Abbildung 1.1) auf den Einsatz von Attributen und Funktionen verzichtet. Da die meisten Besucher von Otto.de zunächst auf der Startseite landen sollte diese auch als Ausgangspunkt zur Betrachtung des Diagramms dienen. Der Node der Startseite trägt die Bezeichnung „Homepagenode“. Von dort aus ist es möglich über die Topic_Nodes und Special_Shop_Nodes oder auch direkt zu den Artikeln zu browsen, welche man erwerben oder betrachten möchte. Über diese Artikel_Nodes oder auch direkt vom Homepage_Node aus, ist es möglich einen Artikel zu bestellen. Dies geschieht über einen komplexen Bestellvorgang, welcher direkt unter dem Homepage_Node verdeutlicht wurde. Desweiteren bietet Otto.de die Möglichkeit verschiedene Serviceleistungen in Anspruch zu nehmen, wie beispielsweise eine Farbberatung oder die „Verfolgung“ von bestellten Artikeln.

2.2.1 Probleme

Die Modellierung des WebShops Otto.de mit Hilfe des Navigationsdiagramms zeigt einige gravierende Probleme auf. Beispielsweise muss der Nutzer während des Bestellvorganges eine Lieferauskunft einholen. Dies geschieht jedoch nicht über ein separates Fenster. Es wird lediglich ein Button gedrückt und anschließend erscheint die gewünschte Information in einem Teilbereich der Seite. Dies ist nicht modellierbar, da es sich dabei um keine Navigation handelt. Es ist jedoch Voraussetzung dafür, dass man weiter durch den Bestellvorgang navigieren kann. Desweiteren erfolgt die Eingabe oder Änderung von Versandinformationen oder Artikeleigenschaften über PopUpFenster. Wie bereits an früherer Stelle erwähnt wurde, sind diese unabhängig vom restlichen Navigationsfluss, also auch nicht im Diagramm darstellbar. Das größte Problem jedoch, welches sich bei der Modellierung eines WebShops mit Hilfe von Navigationsdiagrammen ergibt, ist die Darstellung von Daten. Es ist nicht möglich die für den Bestellprozess relevanten Daten wie beispielsweise die Lieferadresse darzustellen.

2.3 Konsequenzen für die Weitermodellierung

Die Modellierung der Navigation durch einen WebShop zeigt, dass sie keinesfalls umfassend ist, schon garnicht, wenn es um die Modellierung von Geschäftsprozessen geht. Da es keine Möglichkeit gibt Prozesse in einer Navigation darzustellen, ist der nächste Schritt die Erweiterung des Navigationsmodells in der Hinsicht, dass wichtige Prozesse, die nicht direkt etwas mit der Navigation zu tun haben, dennoch dargestellt werden können.

3 Verfeinerung des Navigationsmodells

Das erweiterte Navigationsmodell(Abbildung 1.2) nach Schmid und Rossi[2] baut auf dem herkömmlichen Navigationsmodell auf. Es existieren auch weiterhin die rechteckigen Knoten, jedoch wird nun zwischen sogenannten Entity Nodes und Activity Nodes unterschieden. Die Entity Nodes repräsentieren weiterhin die verschiedenen Seiten eines WebPortals, also den rein navigatorischen Aspekt. Die Activity Nodes hingegen kennzeichnen die mögliche Interaktivität des Portals mit dem Nutzer sowie das Verhalten bestimmter Prozesse. Dabei ist es möglich verschiedene logisch verknüpfte Aktivitäten in einem sogenannten Activity Nodes Container zusammenzufassen. Eine weitere Neuerung ist die mögliche Beschriftung der Navigationspfeile, welche nun nicht mehr ausschließlich die Navigation verdeutlichen sondern auch kausale oder temporale Abhängigkeiten verdeutlichen können. Ein Beispiel hierfür ist die Beschriftung eines Pfeils mit „cancel/submit“ in Abbildung 1.2, welcher verdeutlicht, dass der Aktivitätencontainer

„Artikel_Änderungs_Node_Container“ erst verlassen wird, nachdem eine entsprechende Eingabe erfolgt ist.

3.1 Modellierung von Otto.de

Da der größte Teil des WebShops Otto.de keine Prozesse oder nutzerseitige Aktivitäten ermöglicht bzw. beinhaltet, bezieht sich das in Abbildung 1.2 modellierte Diagramm lediglich auf einen Teil des Bestellvorgangs, beginnend mit dem Homepagenode bis hin zum Bestellschein_Node. Dabei wurden die Teilaspekte des Paketshop_Suche_Nodes und des Monatsbetragsrechner_Nodes nicht weiter betrachtet. Die beiden Knoten „Persönliche Angaben“ und „Artikeländerungen“ wurden in sogenannte Activity Node Container umgewandelt. Dadurch wird verdeutlicht, dass es sich dabei weniger um einen Navigationsaspekt sondern vielmehr um einen interaktiven Prozess handelt. Die zu den Containern weisenden Pfeile haben die Bezeichnung „start“. Die wegführenden Pfeile sind mit „cancel/submit“ beschriftet. In den Containern befinden sich jeweils mehrere Activity Nodes. Es handelt sich hierbei um Eingaben, die vom Nutzer vorgenommen werden können. Die Activity Nodes sind nicht mit Pfeilen verbunden. Dadurch wird dargestellt, dass die einzelnen Aktivitäten nicht kausal oder temporal verknüpft sind, sondern Ad Hoc ausgeführt werden können. Erst wenn von der Nutzerseite eine bestimmte Aktion, im Fall von Otto.de beispielsweise der Bestätigungsbutton, ausgeführt wird, wird der Activity-Node-Container verlassen und die Ad Hoc Aktivitäten können nicht mehr ausgeführt werden.

3.1.1 Probleme

Obwohl mit dem erweiterten Navigationsmodell eine erste Abstraktion von einer reinen Navigationsansicht möglich ist und erste Modellierungsansätze in Bezug auf Prozesse existieren, kann noch nicht von der Darstellung reiner Geschäftsprozesse gesprochen werden. Es werden noch keine Rollen verdeutlicht, das Diagramm in Abbildung 1.2 zeigt nicht auf, welche Parteien eventuell an der Ineraktion teilnehmen, geschweige denn, welche Prozesse wo ablaufen. Die Kapselung von Aktivitäten zu Prozessen erscheint zwar gelungen, basiert aber darauf, dass das Diagramm dadurch anschaulicher und logischer wird. Eine wirkliche Verdeutlichung von Prozesszugehörigkeiten ist bei diesen Diagrammtyp weiterhin sekundär.

3.2 Konsequenzen für die Weitermodellierung

Die bisher vorgestellten Diagrammtypen waren statischer Natur. Es war noch nicht möglich Geschäftsprozesse zu modellieren. Um dies zu ermöglichen, ist es notwendig sich näher mit den Teilnehmern der Geschäftsprozesse von Otto.de zu beschäftigen. Es ist von Nutzen einen Diagrammtyp zu wählen, welcher aufzeigen kann, wer mit Otto.de interagiert und wo die Schnittstellen liegen, d.h. bei welchen Vorgängen oder Handlungen kommt der Akteur Otto.de mit anderen Akteuren in Berührung bzw. findet ein Austausch von Informationen oder Waren statt.

4 Use Cases

Das Use Case Diagramm ist ebenso wie die Navigationsdiagramme ein statischer Diagrammtyp. Es dient dazu, die Rollen verschiedener Akteure innerhalb eines bestimmten Systems zu verdeutlichen. Dadurch ist es möglich zu erkennen, welche Geschäftsprozesse ablaufen und wer daran beteiligt ist. Akteure werden mit Hilfe eines Strichmännchens symbolisiert (Abbildung 2.1). Das große Rechteck symbolisiert das System in dem die verschiedenen Anwendungsfälle (engl.: use cases) dargestellt werden. Ein Akteur wird mit den Aktionen oder Prozessen an denen er beteiligt ist mit einem einfachen Strich verbunden. Nach UML v1.3 gibt es zwei weitere Verbindungsarten zwischen zwei Use Cases. Zum einen die „Include“ Beziehung. Wenn ein Use Case mit einem anderen mit einem Includepfeil verbunden ist, so impliziert er diesen zweiten Use Case, d.h. wird A ausgeführt, muss auch B ausgeführt werden. Ähnlich verhält es sich bei der zweiten, der „Extends“-Beziehung. Einziger Unterschied hierbei ist, dass das Einbeziehen des zweiten Use Case fakultativer Natur ist, d.h. nach A kann, aber muss B nicht ausgeführt werden.

4.1 Bewertung des Diagrammtyps

Das Use Case Diagramm (Abbildung 2.1) ist ein sehr nützliches Werkzeug, um die Akteure von Prozessen und die Schnittstellen zwischen diesen Akteuren näher zu untersuchen und darzustellen. Ebenso wie das Navigationsdiagramm zeichnet sich das Use Case Diagramm durch eine einfache, intuitive Interpretierbarkeit aus. Einem ungeübten Betrachter fällt es leicht, das Dargestellte zu verstehen. Dieses Diagramm eignet sich jedoch nicht dazu Prozesse zu verdeutlichen. Wenn das Modellierungsziel, wie bei Otto.de die Darstellung von Geschäftsprozessen ist, so dient das Use Case Diagramm lediglich dazu einen Einstieg in die vorhandenen Prozesse zu finden. Es ist quasi eine gute und nützliche Vorbereitung auf eine darauf folgende konkrete Modellierung der einzelnen, vorher grob umrissenen, Prozesse.

4.2 Modellierung von Otto.de

Bei der Modellierung von Otto.de mit Hilfe des Use Case Diagramms kristallisieren sich 5 Hauptakteure heraus. Zum einen der Customer, welcher Ausgangsbasis des Modellierungsvorgangs ist. Durch die UseCases des Customers gibt es eine Interaktion mit dem Otto-Bearbeiter und mit der Bank. Der Otto-Bearbeiter interagiert mit dem Otto-Lager und der Bank. Das Lager wiederum hat eine Verbindung zu dem Versandunternehmen, welches Kontakt zum Kunden hat. Die einzelnen Use Cases verdeutlichen welche Prozesse innerhalb des Systems Otto.de ablaufen. Der Kunde kann Artikel suchen, bestellen, verfolgen, erhalten und bezahlen. In Abhängigkeit davon treten die andern Akteure in Aktion und es werden weitere Use Cases getriggert.

4.2.1 Probleme

Ein großes Problem bei der Modellierung von Use Case Diagrammen ist eine schnell entstehende Unübersichtlichkeit bedingt zum einen durch eine Einbringung von zu vielen Akteuren oder durch eine zu detailierte Use-Case-Aufschlüsselung. Je mehr Akteure in einem Use Case Diagramm auftreten umso mehr Interaktionen gibt es, deswegen ist es sinnvoll, auf weniger wichtige Akteure zu verzichten oder sie nur in einem eventuell dazu geschriebenen Satz zu erwähnen. Das zweite Problem ist die Aufschlüsselung der Use Cases in zu viele Unteraktivitäten. Man kann die meisten Tätigkeiten sehr filigran unterteilen, wodurch die Diagrammstruktur völlig überlastet und dadurch unübersichtlich wird.

4.3 Konsequenzen für die Weitermodellierung

Durch die Modellierung des Use Case Diagramms wurden, wie bereits erwähnt, die einzelnen Prozesse grob herausgearbeitet. Es ist bekannt welche Akteure in die jeweiligen Prozesse integriert sind. Aufgrund dieses Wissens ist es nun möglich gezielt einzelne Geschäftsrozesse zu modellieren oder ihren Zusammenhang darzustellen.

5 Die Business Process Modelling Notation

Die Business Process Modelling Notation [1], kurz BPMN, ist eine speziell für die Darstellung von Geschäftsprozessen entworfene Modellierungssprache. Kennzeichnend für diesen Diagrammtyp ist eine Vielzahl verschiedener Knotentypen. Folgende Grundtypen sind unterscheidbar. Das Event, gekennzeichnet durch einen Kreis kann am Ende, am Anfang oder in der Mitte von Prozessen stattfinden. Eine Activity ist ein abgerundetes Rechteck, welches für einen Subprozess, eine Aktivität oder eine Handlung steht. Ein weiteres Element der BPMN ist das Gateway, welches Message Flows teilen und wieder zusammenführen kann. Man unterscheidet bei der BPMN drei verschiedene Kantentypen. Zum einen den Sequenzfluss, welcher mit Hilfe eines durchgehenden Pfeils mit schwarzer Pfeilspitze dargestellt wird. Der Nachrichtenfluss hingegen ist gekennzeichnet durch einen gestrichelten Pfeil, mit einem leeren Kreis am Anfang und einer leeren Pfeilspitze am Ende. Bei Bedarf können auch Assoziationen mit Hilfe eines gestrichelten Pfeils mit offener Pfeilspitze dargestellt werden. Interessanterweise ist es mit BPMN auch möglich Daten zu modellieren, was bei den anderen Diagrammtypen nicht der Fall war. Diese Möglichkeit wird mit Hilfe eines symbolischen Datenblattes dargestellt. Die einzelnen Akteure eines Geschäftsprozesses werden durch sogenannte Pools repräsentiert, welche wiederum in Lanes unterteilt werden können. Beispielsweise wäre es möglich Vertrieb, Einkauf und Lager eines Unternehmens als drei Lanes eines Pools darzustellen.

5.1 Bewertung des Diagrammtyps

Die BPMN ist eine sehr umfassende Möglichkeit der Darstellung von Geschäftsprozessen. Aufgrund der Vielzahl von Modellierungselementen kann so gut wie jedes Designpattern nachgebaut werden. Doch gerade diese Vielfältigkeit stellt auch ein Problem dar. Der BPMN-Diagrammtyp ist nicht intuitiv interpretierbar. Es ist notwendig diese Modellierungssprache zu lernen und zu üben, bevor man in der Lage ist sinnvolle Diagramme zu erstellen. Auch ist es in den meisten Fällen unumgänglich zur Präsentation den Konstrukteur des Modells interviewen zu können, da einem ungeübten Betrachter das Dargestellte nur mit Erklärungen zugänglich ist. Ein Vorteil der BPMN ist die Unterscheidung zwischen Sequenz- und Nachrichtenflüssen. Dadurch können sowohl kausale als auch assoziative Zusammenhänge dargestellt werden. Die Spezifikation erlaubt jedoch keine Nachrichtenflüsse innerhalb eines Pools, beispielsweise zwischen zwei Lanes, was durchaus als Nachteil empfunden werden kann. Ein bedeutender Vorteil vom BPMN-Diagramm ist das Vorhandensein von Cancel/Rollback- und Exceptionevents, welche häufig wichtige Aspekte von Geschäftsprozessen sind. Somit können bereits ausgeführte Aktionen eines Prozesses rückängig gemacht werden.

Es ist möglich, mit Hilfe der BPMN, Prozesse auf verschiedenen Granularitätsstufen zu betrachten. Dies wird durch die Kapselung von Prozessen erreicht. Man spricht in diesem Zusammenhang von kollabierten Subprozessen. Ein weiterer positiver Aspekt der BPMN-Diagramme ist die Modellierung von Daten. Es ist möglich konkret anzugeben, welche Aktionen auf welche Daten zugreifen, sie erstellen oder ändern. Dies ist häufig wichtig um Geschäftsprozesse besser versehen zu können.

5.2 Arten der Modellierung

Um Geschäftsprozesse mit der Business Process Modeling Notation zu modellieren gibt es zwei grundlegende Ansätze. Zum einen den Top-Down Ansatz. Dabei werden zunächst die Akteure grob mit Hilfe von Pools dargestellt und lediglich die signifikanten Nachrichtenflüsse zwischen diesen Akteuren werden modelliert. In den darauffolgenden Schritten werden die Hauptprozesse der einzelnen Pools und deren Zusammenhänge herausgearbeitet. Mit dieser Vorgehensweise können Schritt für Schritt einzelne Aspekte und Teilprozesse detailierter hervorgehoben werden, ohne dass die Diagramme unübersichtlich werden.

Der zweite Ansatz ist die Bottom-Up-Vorgehensweise. Hierbei sind einzelne Teilprozesse zunächst ausschlaggebend. Diese werden zunächst detailiert modelliert. Anschließend werden diese zu einem Subprozess zusammengefasst welcher wiederum mit anderen Subrozessen einen eigenen Geschäftsprozess bildet. Der Modellierende begibt sich dabei Schritt für Schritt auf ein höheres Abstraktionslevel. Es gibt noch andere mögliche Ansätze um Geschäftsprozesse mit BPMN oder anderen Modellierungssprachen darzustellen, doch diese beiden sind die am häufigsten genutzten.

5.3 Modellierung von Otto.de

Bei der Modellierung von Otto.de wurde der Top-Down-Ansatz gewählt. Ausgangspunkt war eine abstrakte Sicht auf die Pools von Otto.de, dem Lieferservice und dem Finanzinstitut(Abbildung 3.1). Da die Prozesse auf der Seite des Kunden relativ linear verlaufen wurden diese bereits in dem ersten Diagramm festgehalten. Die Besonderheiten des Diagramms liegen in zwei bis dahin nicht näher spezifizierten kollabierten Subprozessen „Select item“ und „Buy item“. Es gibt zwei Gatewaytypen in diesem Diagramm, zum einen den datenbasierten Gateway, welcher in Abbildung 3.1 die Zahlungsmethode als Bedingung für die Sequensflussrichtung abfragt und zum anderen den parallelen oder AND-Gateway, welcher den Sequenzfluss derart splittet, dass die Aktivitäten beider „Flussarme“ ausgeführt werden müssen. Bereits in diesem Stadium werden die einzelnen Nachrichtenflüsse eingezeichnet. Da diese bereits aus dem Use Case Diagrammen bekannt sind, ist dies sehr einfach zu realisieren. Die nächste Detailstufe der Modellierung zeigt den groben Geschäftsprozess der bei Otto.de abläuft. (Abbildung 3.2) Während bei dem Kunden der Prozessfluss noch relativ linear war findet man hier einen Rücksprung. Idealerweise sollten derartige Sprünge vermieden werden, da es sich hierbei um keine saubere Modellierung handelt. Es gibt jedoch Situationen, in denen dies unvermeidbar ist oder wo eine Vermeidung einen erheblichen Mehraufwand an Modellierungsarbeit und eine daraus resultierende Unübersichtlichkeit erzeugt. Eine Besonderheit des dargestellten Prozesses ist das Timerevent an der Aktivität „Get money“. Sollte diese Aktivität nach 2 Wochen nicht abgeschlossen sein, so wird der Sequenzfluss zur Aktivität „Start_money_reminder_process“ geleitet. Damit wird der Fall modelliert, dass ein Kunde die von ihm gekaufte Ware nicht bezahlt.
In diesem Diagramm wird auch erstmals das inclusive oder OR-Gateway genutzt. Hierbei kann der Sequenzfluss entweder einen, mehrere oder alle ausgehenden Verbindungen nutzen. Der OR-Gateway wurde an dieser Stelle eingesetzt, da der Kunde per Kreditkarte den Artikel bezahlen kann („Get money“), obwohl er ihm nicht zusagt und er ihn wieder zurück sendet („Get item back“). Eine genauere Spezifizierung der Pools „Supply Service“ und „Financial Institute“ ist nicht sinnvoll, da die Prozesse dort nicht relevant sind. Deswegen bleiben diese Pools bei allen weiteren Modellierungsschritten Black Boxes. Wie bereits an dieser Stelle zu erkennen ist, verlaufen die Prozesse bei dem Kunden und dem Onlineshop recht ähnlich. Nachdem der Kunde die Seiten des Shops erreicht hat, kann er Artikel auswählen, danach kann er den Kauf bestätigen. Anschließend erhält er die Artikel und kann sie bezahlen oder wieder zurück schicken. Gegebenenfalls bekommt er sein Geld zurück. Bei Otto.de werden die gewählten Artikel in einem sog. Einkaufswagen gesammelt. Anschließend werden die benötigten Verkaufsinformationen angefordert. Danach werden die Artikel verschickt und es wird auf den Geldeingang oder auf die reklamierte Ware gewartet. Sollte es dabei Probleme geben tritt ein Mahnungsprozess in Kraft. Während ein Teil der aufgezählten Teilprozesse trivial und somit nicht näher betrachtenswert sind, lassen sich andere Prozesse weiter aufschlüsseln.

Die kollabierten Subprozesse „Select item“ auf der Kundenseite und „Collect items in cart“ auf der Seite von Otto.de sind in Abbildung 3.3 detailierter modelliert. Hierbei musste auf Kundenseite dargestellt werden, dass der Kunde jederzeit Artikel zum Warenkorb hinzufügen kann, die Eigenschaften dieser Artikel ändern kann, oder

Artikel aus dem Warenkorb entfernen kann. Dies wurde mit Hilfe eines XOR-Gateways realisert. Da die Eigenschaften eines Artikels beliebig oft hintereinander verändert werden können ist diese Aktivität auf Kundenseite mit einem Activity Loop dargestellt. Auf Seiten des Onlineshops beginnt der Subprozess mit einem Message-Startevent, d.h. in diesem konkreten Fall, dass erst wenn der erste Artikel von dem Kunden in den Warenkorb gelegt wird, der Subprozess bei Otto.de gestartet wird. Die bereits erwähnte Veränderung der Artikeleigenschaften ist bei Otto.de mit Hilfe eines Ad Hoc Prozesses modelliert worden, um die Beliebigkeit der Reihenfolge und Zahl der Wiederholungen darzustellen. Da der Customer in diesem Teil des Geschäftsprozesses die direkte Steuerung das Sequenzflusses bei Otto.de übernimmt, wurde dies mit Hilfe eines Event-Based Gateways modelliert. Dabei wird auf eine Nachricht des Kunden gewartet, bevor ein bestimmer Sequenzfluss weiterverfolgt wird.

Da noch weitere, direkt auf den eben beschriebenen Prozess folgende, kollabierte Subprozesse detailiert modelliert werden, werden die Start- und Endevents zweier benachbarter Prozesse mit sogenannten Linkevents gestartet und beendet. Diese sind mit einem Label versehen. Direkt aufeinander folgende Prozesse haben dassselbe Label. In Abbildung 3.4 werden die Prozesse „Buy item“ und „Collect order information“ aufgeschlüsselt. Der Customer startet den Prozess auf der Seite von Otto.de, dargestellt durch ein Message-Startevent. Je nachdem, ob der Kunde bereits eingeloggt ist, dies noch tut oder seine Daten neu eingibt werden verschiedene Sequenzflusspfade genutzt. In jedem Fall wird der sog. „Trust-Status“ des Kunden geprüft, d.h. es wird kontrolliert, ob er noch Zahlungen offen hat oder es andere Probleme mit ihm gab. Ist alles in Ordnung fordert Otto.de noch intern Lieferinformationen an und gibt diese an den Kunden weiter. Anschließend versendet Otto.de die Artikel, wie in Abbildung 3.5 dargestellt wird. Otto.de ordert die Artikel aus den Lagern, sind diese nicht vorrätig werden sie bei den Herstellern nachbestellt. Nach der in den Lieferinformationen angegebenen Zeitperiode wird das Lieferunternehmen damit beauftragt die Artikel an den Kunden zu liefern. Sind zu diesem Zeitpunkt noch nicht alle Artikel vorrätig, wird, nachdem die fehlenden Artikel verfügbar sind, das Unternehmen ein weiteres Mal beauftragt die restlichen Artikel zum Kunden zu bringen. Die beiden Stellen des Sequenzflusses, an denen gewartet werden muss, werden durch Intermediate Timerevents dargestellt. Um sie genauer zu spezifizieren sind sie mit entsprechenden Descriptions, also Beschreibungen, versehen.

Sollte einmal ein Kunde den Zahlungen nicht nachkommen, tritt der „Start money_reminder_process“ in Kraft. Dieser ist in Abbildung 3.6 dargestellt. Mit den bereits vorgestellten Modellierungselementen der BPMN wird gezeigt, dass zunächst eine Mahnung an den Kunden verschickt wird, dies geschieht, sofern er in der Zwischenzeit nicht bezahlt, zweimal im Abstand von jeweils zwei Wochen. Nach weiteren zwei Wochen wird der Vorgang einem Anwalt übergeben.

5.3.1 Probleme

Im Bezug auf die Darstellbarkeit von Sachverhalten ergeben sich bei der BPMN keinerlei Probleme. Mit Hilfe der Spezifikation [1] sind die Modellierungselemente eindeutig nutzbar. Probleme könnten sich eventuell aus dem Umfang der möglichen

Elemente ergeben. Beispielsweise ist es fraglich ob es wirklich notwendig ist, 6 verschiedene Gateways zu definieren. Neben einem Event-Based Gateway würde wahrscheinlich ein Data-Based Gateway reichen, an dessen Ausgangsstellen die entsprechenden Sequenzflussbedingungen vermerkt sind.

Speziell im Fall von Otto.de ergab sich ein Problem daraus, den Prozess auf Kundenseite derart zu gestalten, dass ein eventueller Austausch des Internetkunden gegen einen Telefonkunden möglich ist. Das in Abbildung 3.2 vorliegende Diagramm ermöglicht dies, auch wenn der Telefonkunde nicht explizit modelliert wurde. In diesem Fall müsste ein Otto-Kundenbetreuer oder ein elektronischer Mittler auftreten.

5.4 Zwischenfazit

Die BPMN erweist sich als sehr mächtige Modellierungssprache, welche die Nachteile vorher beschriebener Diagrammtypen ausgleicht. Da es sich um einen sehr komplexen Diagrammtyp handelt ist es notwendig sich intensiv mit der Notation zu beschäftigen. Das Modellierungsbeispiel Otto.de zeigt, dass eine gewisse Übersichtlichkeit gewährleistet ist, auch wenn es sich um recht komplexe Prozesse handelt.

5 YAWL

Die Abkürzung YAWL steht für Yet Another Workflow Language [7]. Ähnlich den in BPMN üblichen Events gibt es in YAWL sogenannte Conditions, welche ebenfalls mit Hilfe eines Kreises dargestellt werden. Es gibt Input Conditions mit dem typischen „PLAY“-Pfeil, wo der Prozess beginnt und Output Conditions mit dem quadratischen „STOP“-Symbol. Das splitten von Markenflüssen, wie sie bei YAWL genannt werden, erfolgt entweder mit Hilfe einer Condition und zweier atomarer Tasks, dargestellt durch ein Viereck, als sog. Deferred Choice, oder durch die direkte Abfrage einer Bedingung. Es ist unter YAWL möglich, Prozesse zu kapseln. In diesem Zusammenhang wird von einer Composite Task gesprochen, ähnlich den kollabierten Subprozessen unter BPMN.

5.1 Bewertung des Diagrammtyps

Es ist möglich mit Hilfe von YAWL sämtliche Workflowpattern zu konstruieren. [6] Aufgrund der Beschränktheit der verwendbaren Elemente ist dies jedoch häufig sehr aufwendig und umfangreich. Die Eventbehandlung unter YAWL ist sehr simpel. Dadurch ist es schwer, unterschiedliche Events auf den ersten Blick zu unterscheiden und die entsprechenden Bedeutungen beizumessen. Ein Vorteil in Hinblick auf die Erlernbarkeit und intuitive Inerpretierbarkeit von YAWL ist jedoch die Einfachheit dieser Sprache.

5.2 Modellierung von Otto.de

Um einen möglichen Vergleich zwischen YAWL und BPMN zu ermöglichen, wurde derselbe Prozess modelliert, welcher in Abbildung 3.1 auf Kundenseite dargestellt wurde. Auch hier (Abbildung 4.1) beginnt der Kunde damit, auf die Homepage von Otto.de zu gelangen. Anschließend kann er Artikel auswählen und diese kaufen. Diese beiden Vorgänge sind in Form von Composite Tasks dargestellt, um zu verdeutlichen, dass eine weitere Granulierung der Vorgänge möglich ist. In diesem Fall wurde jedoch darauf verzichtet, da diese bereits in BPMN (Abbildung 3.3 + 3.4) vorgenommen wurden. Die Verzweigungen der Markenflüsse geschieht über Deferred Choices, AND- und OR-Splits. Zusammengeführt werden sie anschließend über AND-, und OR-Joins. Um die Deferred Choices zu ermöglichen mussten noch weitere Tasks definiert werden.

5.2.1 Probleme

Wie bereits erwähnt, ist YAWL recht benutzerfreundlich und wirft im Bezug auf die Syntax keine Probleme auf. Jedoch wachsen Diagramme, auch mit teilweise recht einfachen Inhalten, schnell an. Dies liegt daran, dass häufig bestimmte Sachverhalte „umschrieben“ werden müssen, statt dass sie direkt dargestellt werden können. Ansonsten ist der Gebrauch von YAWL problemlos, eventuell sollte man eine Erweiterung der Modellierungselemente erwägen um das Aufblähen der Diagramme zu verhindern.

5.3 Vergleich mit BPMN

Ein Vergleich von BPMN und YAWL führt zwangsläufig zu dem Ergebnis, dass YAWL von seiner Struktur und seinem Umfang an Sprachelementen her wesentlich einfacher ist. YAWL ist leichter zu interpretieren und schneller erlernbar. YAWL ermöglicht das Darstellen sämtlicher Workflowpattern, genauso wie BPMN. YAWL bietet jedoch keine Möglichkeit der Unterscheidung einzelner Akteure, wie es unter BPMN mit Hilfe von Pools und Lanes möglich ist. Außerdem können mit YAWL keine Daten modelliert werden. Auch die Darstellung von Ad Hoc Prozessen ist mit YAWL nicht möglich. Ein weiterer gravierender Vorteil von BPMN ist der Nachrichtenfluss, der allerdings auch erst durch die Existenz von mehr als einem Akteur einen Sinn bekommt. Daraus ergibt sich, dass die Modellierung eines einzelnen Prozesses, an dem nur ein Akteur beteiligt ist mit YAWL einfacher und effizienter ist. Sobald jedoch eine Interaktion von mehr als einem Akteur notwendig ist, oder Daten dringend modelliert werden, müssen ist YAWL nicht mehr ausreichend und es sollte BPMN verwendet werden.

6 Schlussfolgerung

Es wurde deutlich, dass erst die Kombination verschiedener Diagrammtypen alle Aspekte des WebShops Otto.de darstellen konnte. Es sollte generell mit statischen

Diagrammtypen begonnen werden, um die Strukturen eines Onlineshops darzustellen. Hierfür eignet sich ein Navigationsdiagramm oder ein erweitertes Navigationsdiagramm, falls einige wichtige Aspekte mit Hilfe des ersteren nicht modellierbar sind. Anschließend sollte man die einzelnen Akteure mit Hilfe eines Use Case Diagramms darstellen. An dieser Stelle weiß man durch das Navigationsdiagramm, wo Prozesse stattfinden und das Use Case Diagramm gibt darüber Auskunft wer daran beteiligt ist. Der letzte Schritt ist die Verwendung eines prozessmodellierenden Diagramms um dieses Wissen mit den eigentlichen Geschäftsprozessen zu verbinden. Es empfiehlt sich hierfür BPMN zu nutzen, da es sich dabei um eine sehr mächtige Modellierungssprache handelt. Interessanterweise ist es mögliche die einzelnen modellierten Prozesse in bestimmte Bereiche des Navigationsdiagramms einzuordnen. Dadurch erhält man Auskunft darüber, an welchen Stellen der Navigation der Kunde Geschäftsprozesse steuert oder darauf Einfluss nimmt.

Quellenverzeichnis

[1] Business Process Modeling Notation Spec. Version 1.0, 2004
[2] H. A. Schmid, G. Rossi, „Modeling and Designing Process in E-Commerce Applications“, IEEE Computer Society, 2004
[3] http://www.bpmn.org/
[4] http://www.otto.de/
[5] http://www.citi.qut.edu.au/yawl/index.jsp
[6] W.M.P. van der Aalst and A.H.M. ter Hofstede, „Workflow Patterns: On the Expressive Power of (Petri-net-based) Workflow Languages ” In K. Jensen, editor, Proceedings of the Fourth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2002), volume 560 of DAIMI, pages 1–20, Aarhus, Denmark, August 2002, University of Aarhus
[7] W.M.P. van der Aalst, L. Aldred, M. Dumas, and A.H.M. ter Hofstede, “Design and Implementation of the YAWL system”,To appear in Proceedings of The 16th International Conference on Advanced Information Systems Engineering (CAiSE 04), Riga, Latvia, June 2004, Springer Verlag

Anhang

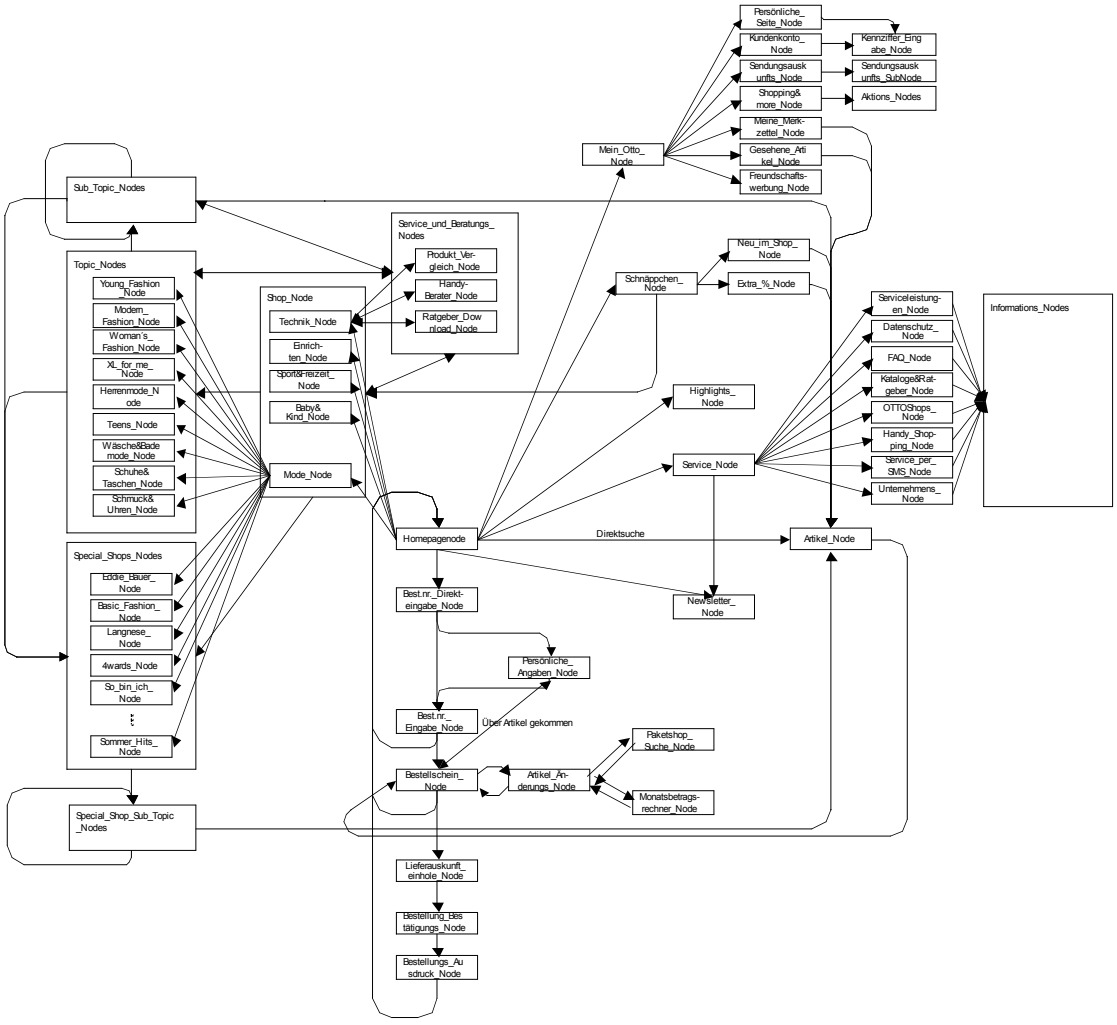


Fig.1.1. Navigationsdiagramm des WebShops Otto.de

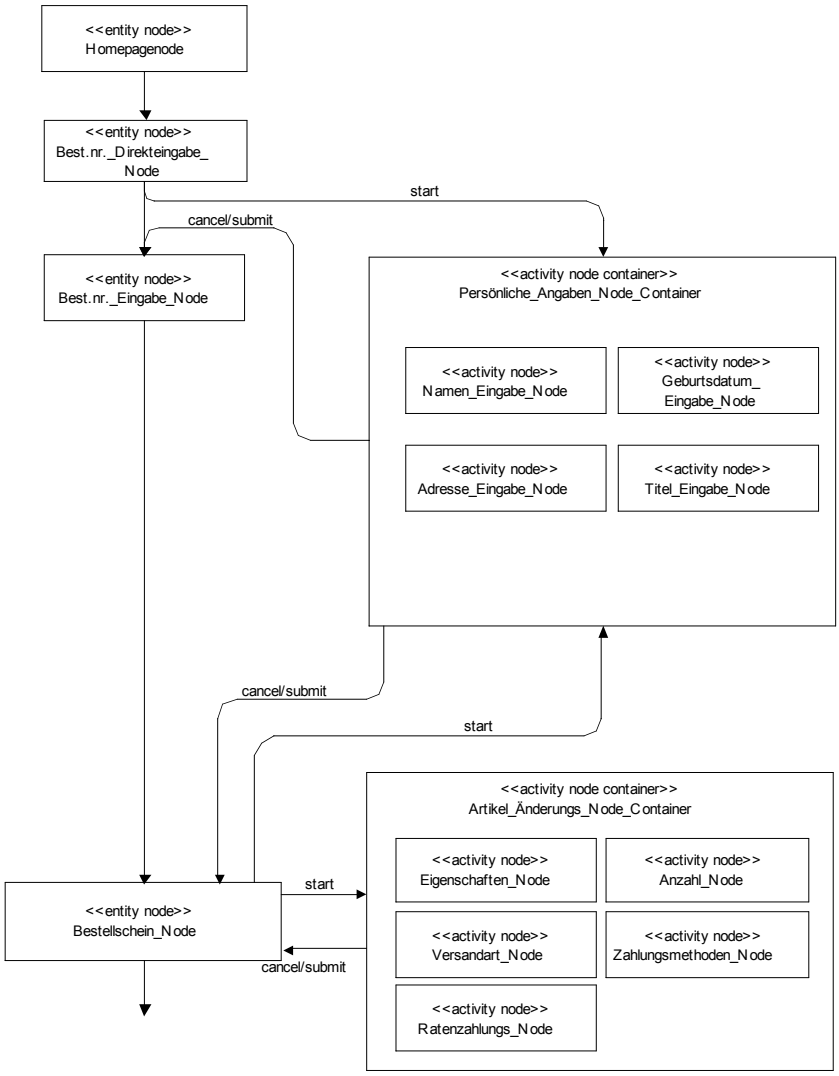


Fig.1.2. Erweitertes Navigationsdiagramm[2] eines Teils des Otto-Bestellsystems

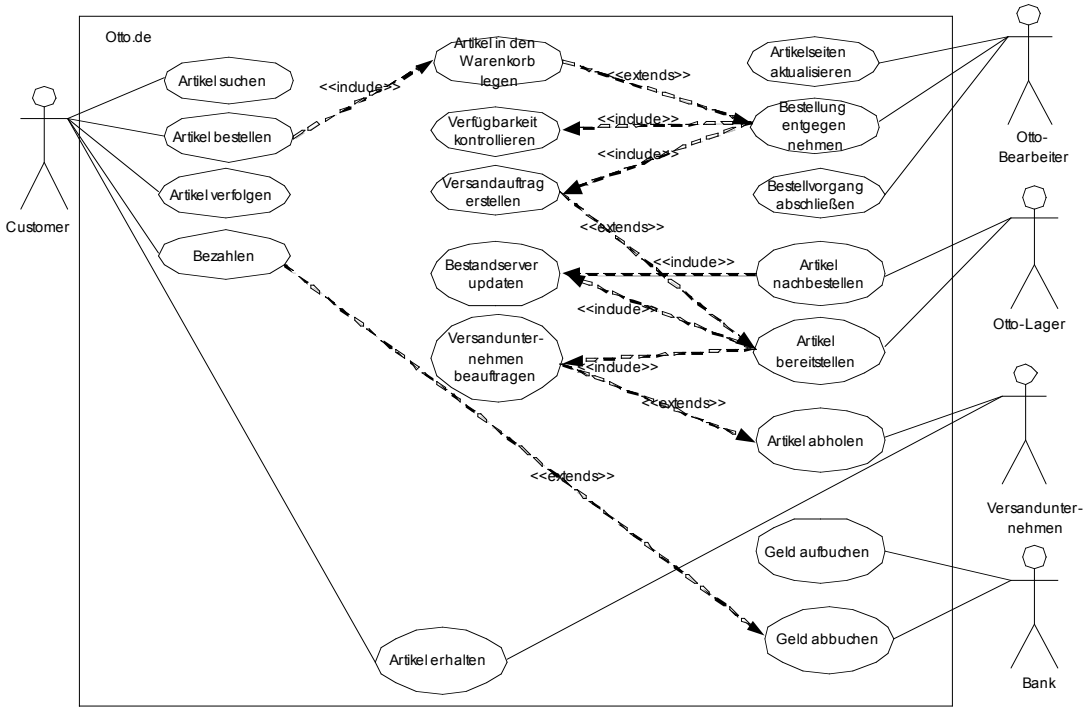


Fig.2.1. Use Case Diagramm des WebShops Otto.de aus der Sicht des Kunden

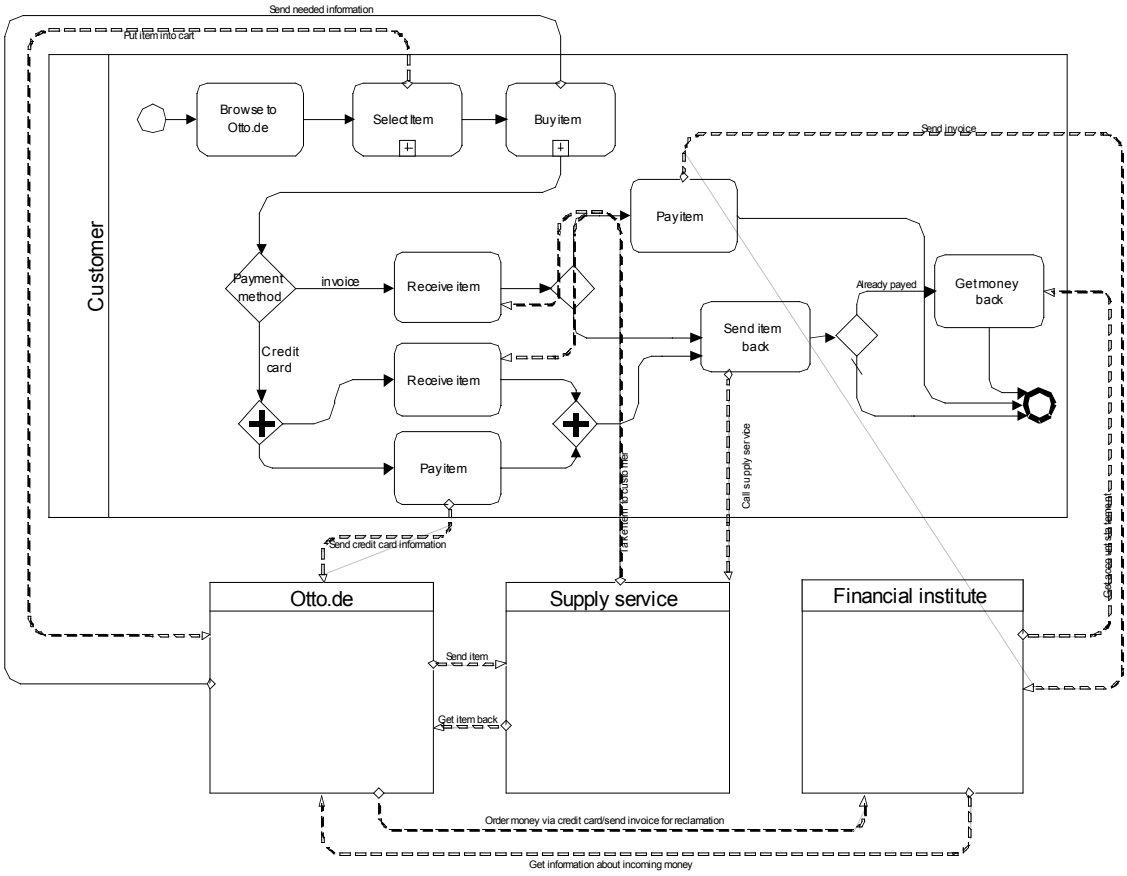


Fig.3.1. BPMN-Diagramm, Kunde als White Box, restliche Akteure als Black Boxes

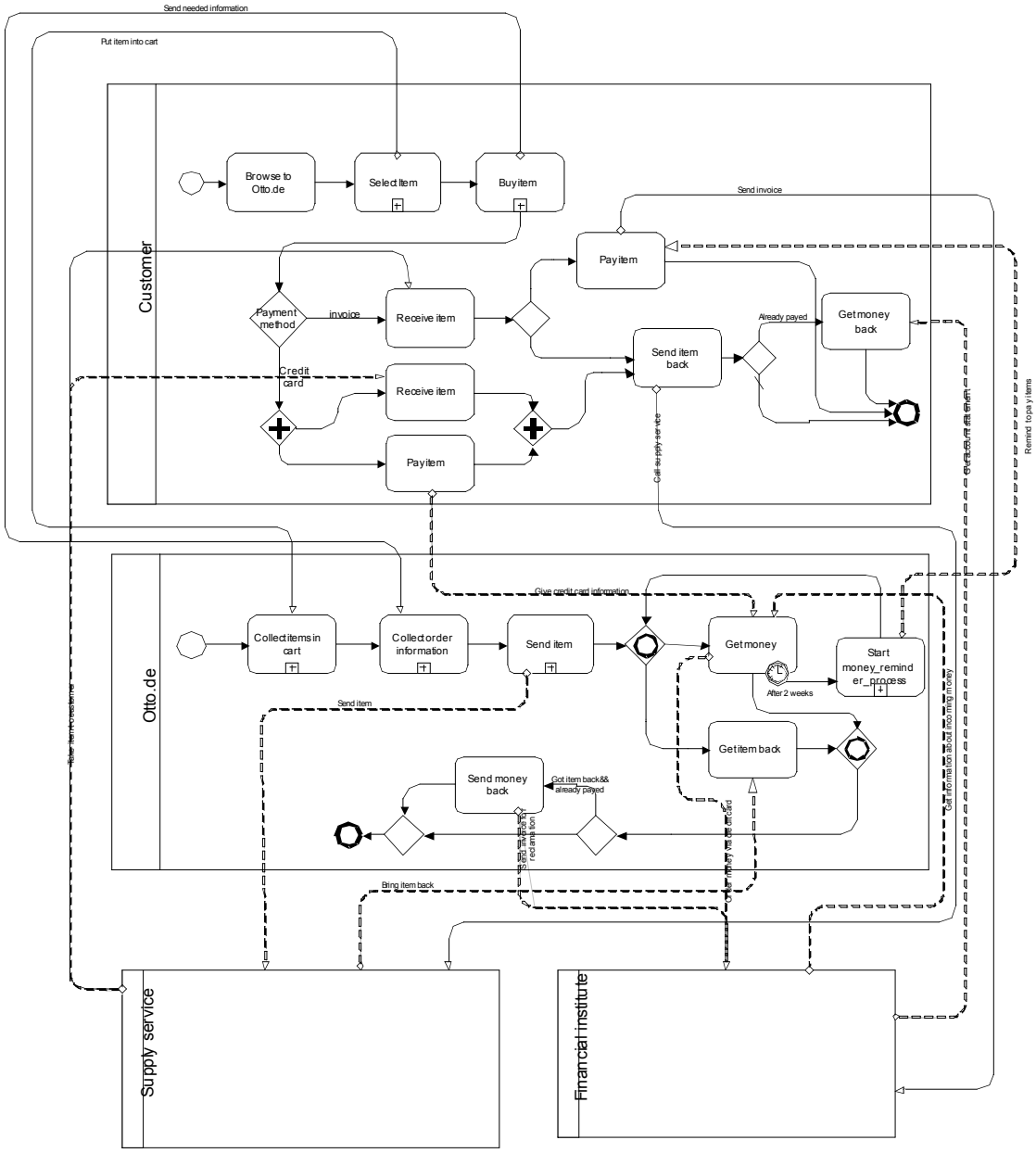


Fig.3.2. BPMN-Diagramm, Kunde und Otto.de als White Boxes

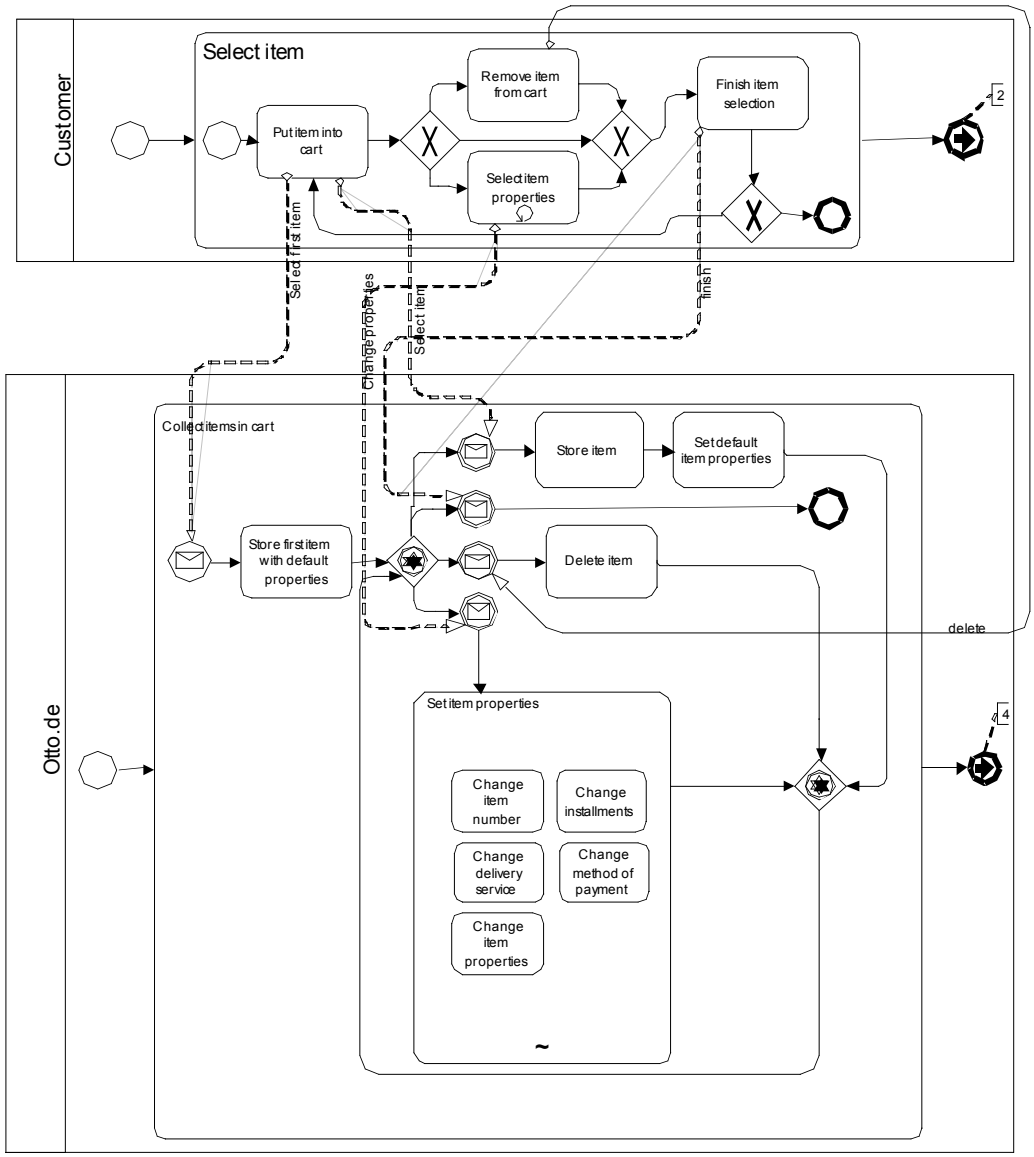


Fig.3.3. BPMN-Diagramm, Subprozess der Artikelauswahl

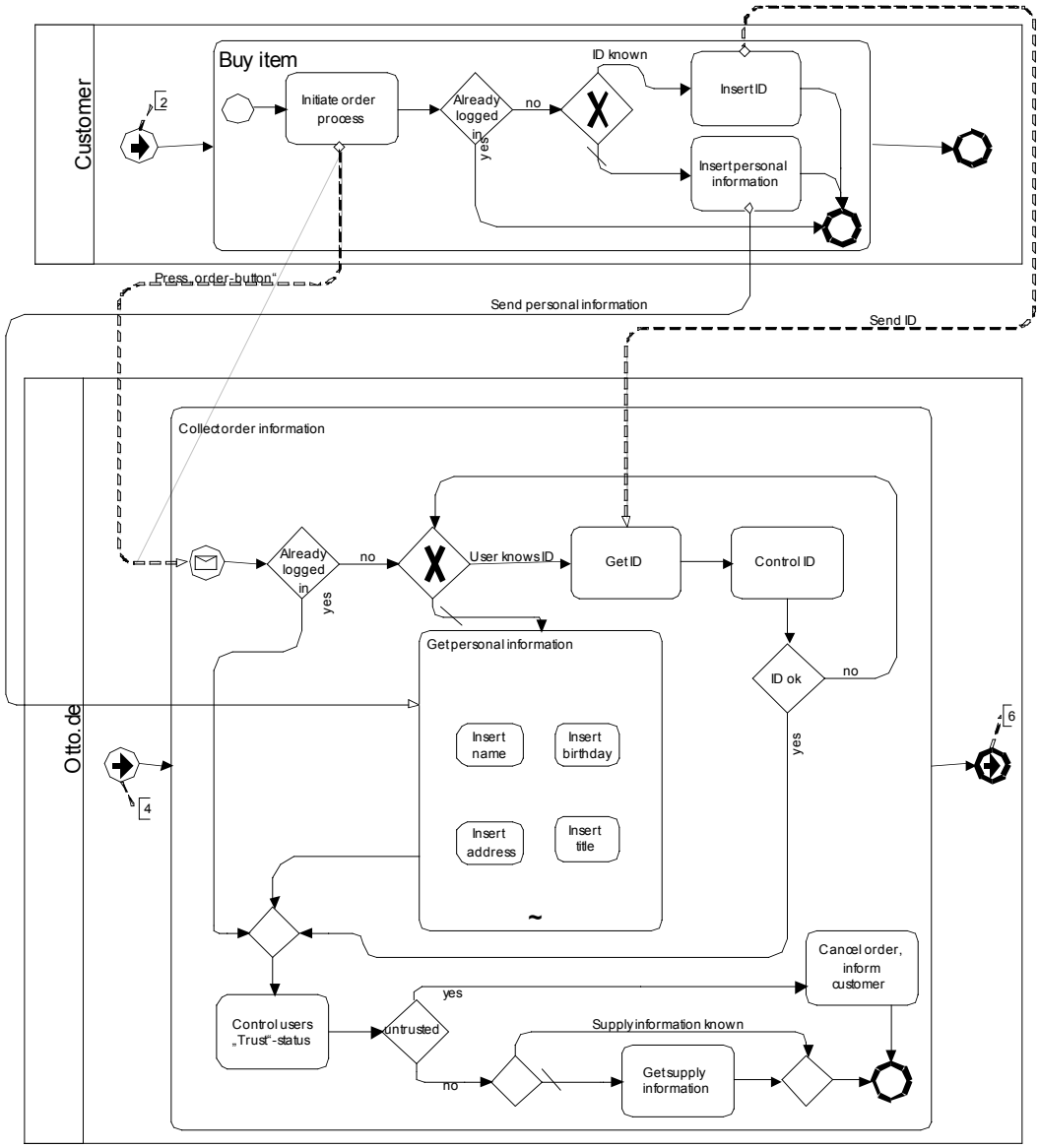


Fig.3.4. BPMN-Diagramm, Subprozess der Bestellinformationseingabe

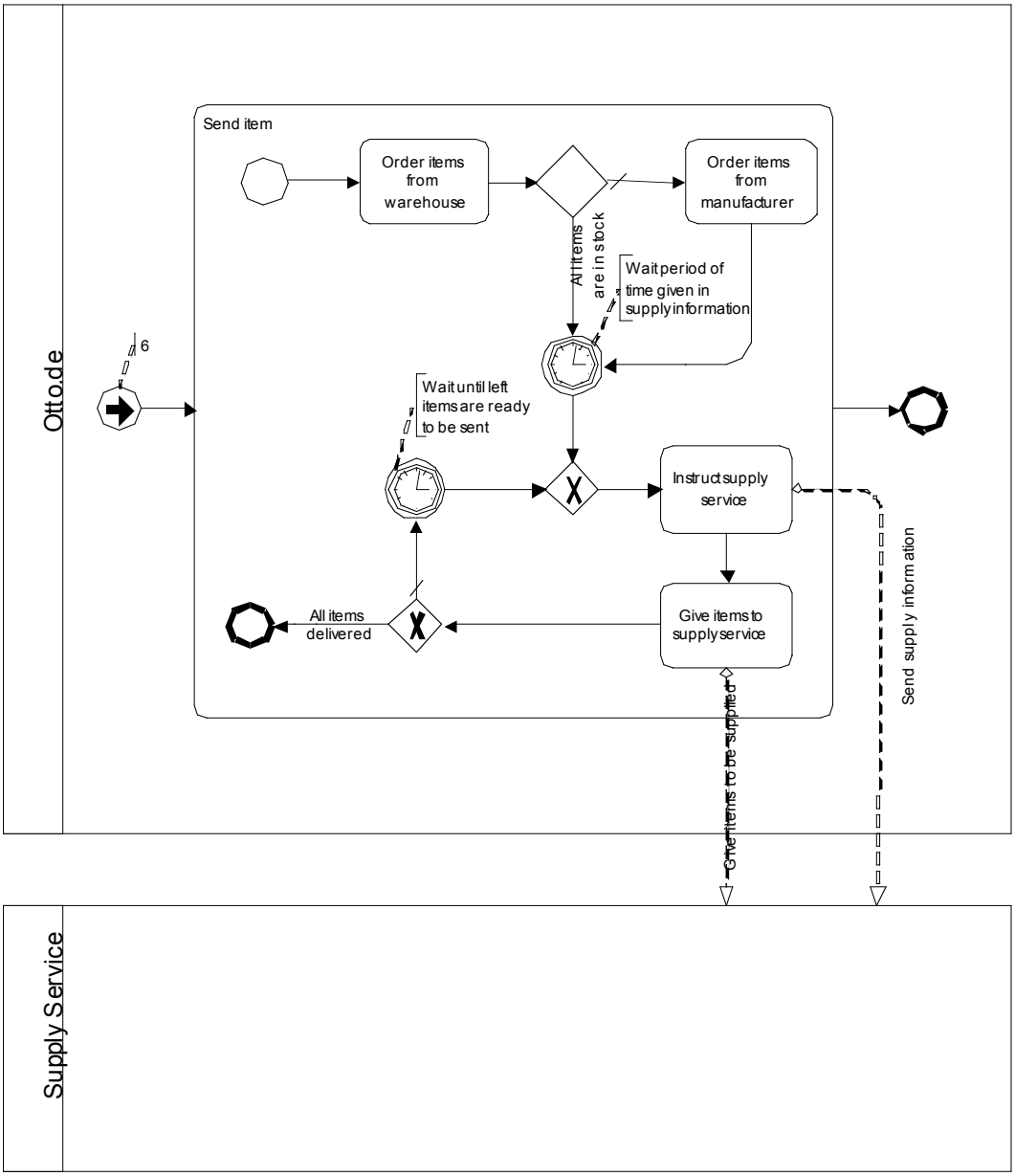


Fig.3.5. BPMN-Diagramm, Subprozess des Artikelversands

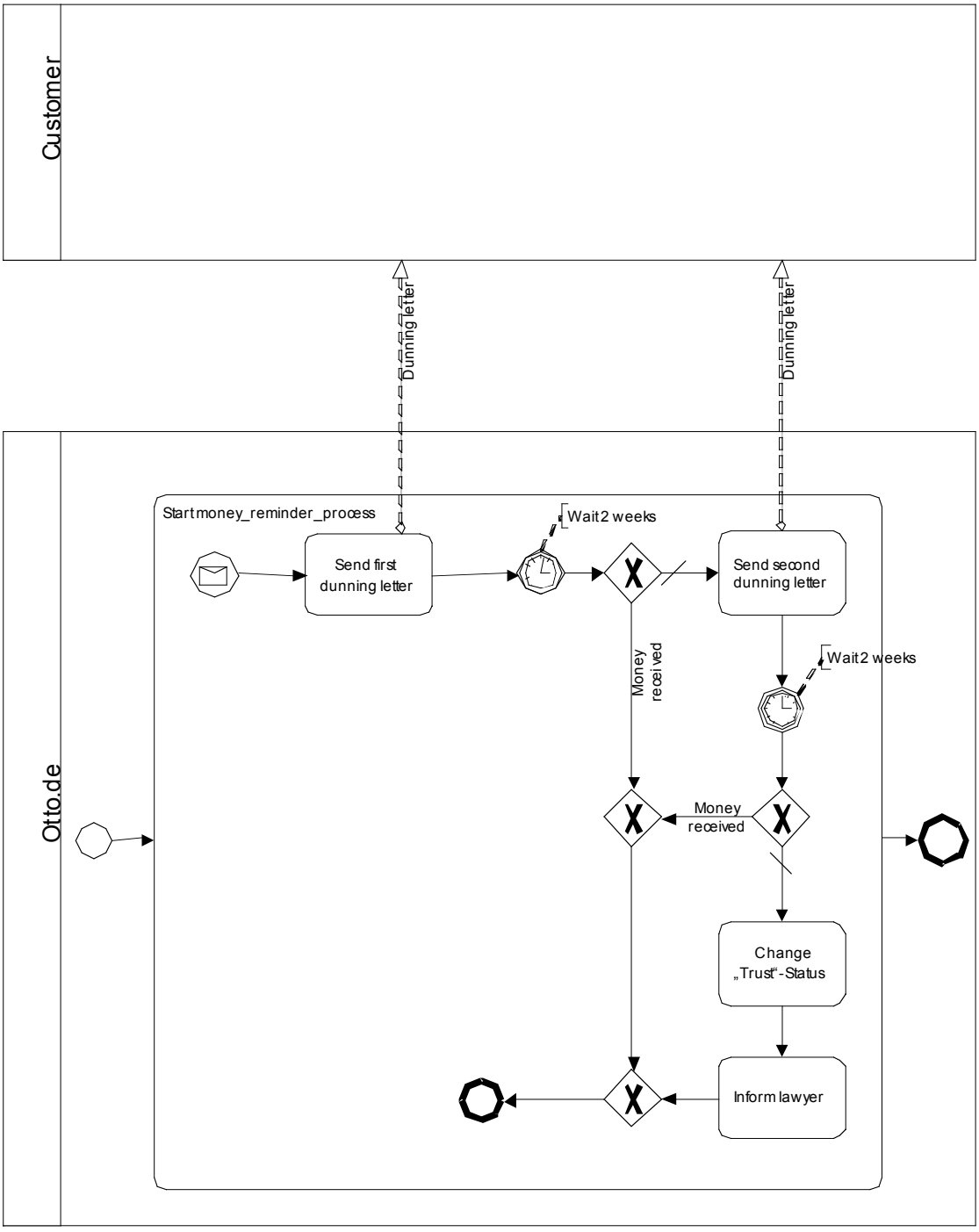


Fig.3.6. BPMN-Diagramm, Subprozess des Kundenmahnens

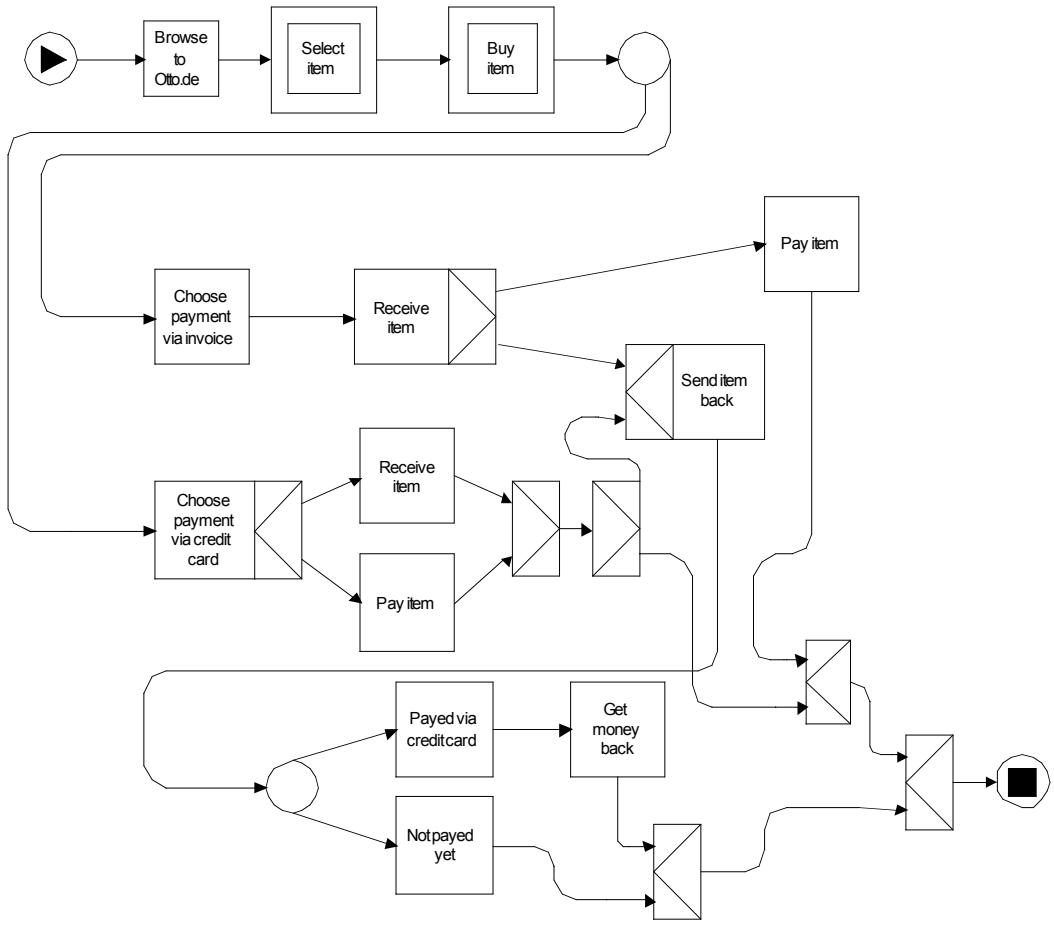


Fig.4.1. YAWL-Diagramm, Artikelkauf aus Kundensicht

Modeling and Designing Processes in E-Commerce

Considering as example DELL

Christian Liesegang

christian.liesegang@hpi.uni-potsdam.de

Zusammenfassung. Der Bereich des E-Commerce stellt mittlerweile einen wichtigen Vertriebskanal für Unternehmen dar. Dennoch wurde besonders auf diesem Gebiet bisher wenig Prozessmodellierung betrieben. Neben der geeigneten Wahl der Mittel ist es besonders die Problematik der Navigation mit Hilfe des Browsers durch den Kunden, die die Prozessmodellierung im E-Commerce von anderen Anwendungsgebieten unterscheidet. Navigation besagt, dass man zu einem beliebigen Zeitpunkt einen Bereich einer Webseite verlassen und einen anderen aufsuchen kann. Dies steht im Widerspruch mit der Definition des Begriffes Prozess, der eine eindeutige Abfolge von Aktivitäten vorsieht. Diese Arbeit beschäftigt sich mit diesem Problem und zeigt an Hand des Onlineshops von DELL auf, wie solche Prozesse erarbeitet und modelliert werden können. Ausgehend von statischen Modellen in Form von ER-Diagrammen werden dynamische Navigationsschemata erstellt. Diese dienen als Orientierungshilfen bei der anschließenden Prozessmodellierung. Für die Modellierung wird BPMN verwendet. Darüber hinaus wird BPMN mit der Prozessbeschreibungssprache YAWL verglichen, um über die Flexibilität und Ausdrucksstärke von BPMN aussagen zu können.

1 Einleitung

In den letzten Jahre wurden immer mehr E-Commerce-Anwendungen entwickelt. Diese Anwendungen stellen mittlerweile einen wichtigen Teil von Unternehmen dar. Dadurch ist man für potentielle Kunden und Geschäftspartner nahezu jederzeit und überall erreichbar. Dabei reichen die Aufgabengebiete dieser Anwendungen von Webportalen über Auktionshäuser bis hin zu Onlineshops.

1.1 Prozesse im Bereich E-Commerce

Betrachtet man die Auswahl der im Internet vorhanden Dienstleister, entdeckt man oft gewisse Ähnlichkeiten. So ist zum Beispiel der Vorgang des Onlineshoppings in der Regel eine Komposition aus den Aktivitäten Produktkataloge durchstöbern, Produkt

auswählen und in einen virtuellen Warenkorb legen, Bestellung abschließen, Zahlungsmodalitäten festlegen und Lieferanschrift hinterlassen. Auffällig ist jedoch, dass gegenüber traditionellen Prozessen in Unternehmen, wie zum Beispiel die telefonische Kundenbetreuung oder die Bestellung mittels eine Bestellkataloges auf dem Postweg, im Bereich des E-Commerce klar definierte und dokumentierte Prozessmodelle nur schwer zu finden sind. Es gilt einige Schwierigkeiten zu überwinden die erklären könnten, warum in diesem Bereich bislang so wenig Prozessmodellierung betrieben wurde. So muss als erstes eine geeignete Modellierungssprache gewählt werden. Neben Petrinetzen, dem UML-Standard oder YAWL gibt es auch Modellierungssprachen wie zum Beispiel BPMN, die für die Beschreibung solcher Prozesse geeignet sind. Besonders dem seit Mai 2004 in der Version 1.0 vorliegende BPMN wird in dieser Arbeit Beachtung geschenkt. Eine kurze Erläuterung der Zielsetzungen von BPMN bietet das Kapitel 1.2. Eine weitere Schwierigkeit beim E-Commerce besteht darin, dass bei der Modellierung die Gefahr besteht, dass Prozessvorgänge und das Navigieren auf einer Webseite miteinander vermengt werden. Es gilt eine strickte Trennung diese beiden Sachverhalte einzuhalten. Ein weiteres Problem besteht in der nicht vorhandenen Transparenz dieser Prozesse. Versucht man als Außenstehender zum Beispiel einen Onlineshop und dessen Prozesse zu modellieren, kann man die Prozesse modellieren, die sich direkt am Browser mitverfolgen lassen. Diese decken aber nur einen gewissen Teil der tatsächlich existierenden Prozesse ab. Viele Prozesse befinden sich im „Inneren“ des Onlineshops, der Shop gleicht einer Blackbox. Von Außen nicht ersichtliche Prozesse sind zum Beispiel die Bestellannahme. Was geschieht intern, wenn eine neue Bestellung aufgegeben wird? Wie wird diese Verarbeitet? Wann und wie spielen Geldinstitute eine Rolle? Unter welchen Bedingungen wird eine Bestellung letzten Endes versandt? Diese Prozesse gehören ebenfalls zur vollständigen Prozessmodellierung dazu und müssen beachtet werden.
Die Schwierigkeit besteht darin, dass diese Prozesse auf Grund der fehlenden Transparenz nicht ersichtlich sind. Diese Prozesse müssen daher durch Überlegung selbst erarbeitet werden. Manchmal verbergen sich Hinweise für die Prozesse direkt auf der Webseite. So ermöglicht ein Blick in die allgemeinen Geschäftsbedingungen von DELL Rückschlüsse darauf, wann eine Bestellung tatsächlich versendet wird. Als anschauliches Beispiel wird im dritten Kapitel an Hand einiger ausgewählter Prozesse der Onlineshop des Computerherstellers DELL modelliert.

1.2 Zielsetzung von BPMN

Dieser Abschnitt liefert einen kurzen Überblick über die Zielsetzungen von BPMN, die bei der Spezifikation eine wichtige Rolle spielten. Eine knappe Einführung in die Notation von BPMN findet man bei [2]. Die komplette Spezifikation von BPMN ist bei [1] verfügbar.

BPMN ist mit dem Ziel entwickelt worden, dass nicht nur die Menschen in der Lage sind die Prozessdiagramme zu verstehen, die diese entworfen haben, sondern auch andere Menschen in einem Unternehmen, die an dem Entwurfsprozess nicht unmittelbar beteiligt waren. Eine zweite wichtige Zielsetzung war es, Diagramme, die mit BPMN entwickelt wurden, in XML basierte Ausführungssprachen für

Prozesse, wie zum Beispiel BPEL4WS, automatisch übersetzen zu können. BPMN versucht die Stärken verschiedener bereits etablierter Notationen in sich zu vereinen. Dies soll eine universelle Einsetzbarkeit von BPMN in vielen Anwendungsgebieten ermöglichen. Diese Flexibilität diente als Motivation, BPMN zur Modellierung der Prozesse zu verwenden.

1.3 Vorgehensweise

Ein guter Ansatzpunkt für die Modellierung der Prozesse bei DELL ist es, zuerst die Webseite aufzusuchen. An Hand der Webseite lassen sich erste statische Modelle entwerfen, die den Ausgangspunkt für die weitere Entwicklung darstellen. Statische Modelle dienen dazu, Abhängigkeiten und Strukturen aufzuzeigen. Besonders im Bereich des E-Commerce bieten Navigationsdiagramme Hinweise darauf, wie Aktivitäten in einem Prozess angeordnet sein könnten. Sie liefern ein Bild des Aufbaus der Webseite. Das anschließende Kapitel zwei wird sich mit dieser Thematik beschäftigen. Ist man sich darüber im Klaren, wie der strukturelle Aufbau beschaffen ist, kann man dieses Wissen verwenden, um erste Prozesse zu modellieren. Dabei ist es hilfreich sich zu erst auf Prozesse zu beschränken, die auf der Webseite unmittelbar ersichtlich sind. BPMN wird im Kapitel drei zur Modellierung der E-Commerce-Prozesse verwendet. Hier wurde mit dem sehr wichtigen Prozess der Veränderung des Warenkorbes begonnen. Dieser Prozess wurde dann in die Prozesse eingereiht, die sich aus der Webseite nicht unmittelbar ableiten lassen. So entstand nach und nach ein vollständiges Bild der Prozesse. Der vierte Abschnitt beschäftigt sich mit einem Vergleich der Modellierungssprachen BPMN sowie YAWL. Dem Vergleich folgt eine Bewertung der beiden Modellierungssprachen und eine Schlussfolgerung über BPMN.

2 Statisches Modell und Navigation

Eine der größten Schwierigkeiten bei der Modellierung von E-Commerce-Anwendungen ist es, die Interaktion des Anwenders mit dem Browser von den Prozessmodellen zu trennen. Im Folgenden soll ein möglicher Ansatz für eine entsprechende Modellierung gegeben werden.

2.1 Statisches Modell

Bei der Prozessmodellierung im Bereich des E-Commerce bietet es sich an, die Funktionalitäten der Anwendung zuerst zu erforschen. Am Beispiel von DELL genügt es, die Webseite mit dem Browser aufzurufen. Hat man die Möglichkeiten, die solch eine Anwendung einen Nutzer bietet, durch Ausprobieren und Durchspielen verschiedener Situationen ergründet, ist man in der Lage, diese Szenarien in einem Use-Case- Diagramm festzuhalten. Solch ein Diagramm wurde für den Onlineshop von DELL aufgestellt. Die Abbildung 1 zeigt, welche unterschiedlichen Geschäftsprozesse beim Onlineshop von DELL vorhanden sind und welche kausalen

Abhängigkeiten untereinander bestehen. Neben den Geschäftsprozessen ist auch ersichtlich, welche Personen daran beteiligt sind. Man kann der Abbildung entnehmen, dass DELL zwischen privaten Kunden und Unternehmen unterscheidet. Dies spiegelt sich vor allem in speziellen Dienstleistungen wieder, die private Kunden nicht beanspruchen können. Zu den wichtigsten Geschäftsprozessen bei DELL zählen unter anderem der eigentliche Bestellvorgang, die Verfolgung von Bestellungen und deren Status, sowie umfangreicher Support. Bei den meisten Tätigkeiten sind Mitarbeiter von DELL involviert. Es ist ersichtlich das DELL besonders Wert darauf legt, dass der Kunde während des Bestellvorganges die Möglichkeit hat, Unterstützung in Form von Beratungsgesprächen mit Mitarbeitern von DELL zu erhalten.

Neben einer Erfassung der vorhandenen Geschäftsprozesse sollte man sich vor Beginn der Prozessmodellierung Gedanken darüber machen, welche Objekte sich hinter diesen Geschäftsprozessen verbergen. Welche Entitäten spielen eine Rolle bei diesen Prozessen? Betrachtet man einmal den Prozess des Einkaufens, so ist denkbar, dass für diesen Prozess ein Objekt „Warenkorb“ benötigt wird. In diesem Warenkorb befinden sich Produkte. Ein Warenkorb gehört zu einem Kunden. Dieser Wiederum besitzt eine Rechnungsanschrift. Es zeigt sich, dass sich hinter allen Prozessen sehr viele Objekten befinden, die teilweise innerhalb eines Prozesses Änderungen unterworfen sind. Dieses Wissen ist bei der späteren Prozessmodellierung dienlich. So muss es im Bezug auf den Warenkorb zum Beispiel eine Aktivität geben, die Produkte in den Warenkorb hineinlegt bzw. wieder entfernt.

Die Abbildung 2 zeigt den konzeptionellen Entwurf der möglichen Objekte im Onlineshop von DELL in Form eines ER-Diagramms in FMC-Notation [3]. Die Abbildung verdeutlicht noch einmal die zwei unterschiedlichen Arten von Kunden, die DELL unterscheidet. Kunden können Bestellungen aufgeben. Zu jeder Bestellung gehört genau eine Zahlungsoption, eine Lieferanschrift, sowie spezielle Modalitäten der Auslieferung. Zusätzlich liegt jeder Bestellung ein Warenkorb zu Grunde. In diesem Warenkorb befinden sich beliebig viele Produkte. Dabei unterscheidet DELL bei den Produkten zwischen Computersystemen und Hardware, Software und Dienstleistungen, die eingekauft werden können. DELL bietet die Möglichkeit Computer dem Wunsch des Kunden entsprechend zu konfigurieren. Jedem System ist eine eindeutige Systemidentifizierungsnummer zugeordnet, die es ermöglicht, mit Hilfe dieser Nummer Hilfestellung, Software und Hardware zu erhalten, die genau auf dieses System zugeschnitten ist.

2.2 Navigation

Eingangs wurde gesagt, dass es schwierig sei Prozessabläufe von E-Commerce-Anwendungen zu modellieren. Zum Teil lässt sich dies durch die jederzeit mögliche browserbedingte Navigation innerhalb eines Prozesses begründen. Nimmt man zum Beispiel die Aktivität des Hinzufügen eines Produktes in den Warenkorb. Durch den Browser ist es einem möglich zum Beispiel durch den „Zurückknopf“ zur Produktauswahl zurückzukehren. Bedeutet dies, dass der Artikel wieder entfernt wurde, kann man damit eine Auswahl rückgängig machen? Die Navigation verhindert eine eindeutige Abfolge von Aktivitäten von Prozessen. Versucht man die Navigation

bei der Prozessmodellierung zu berücksichtigen, kommt es schnell zu unübersichtlichen und vermutlich fehlerhaften Prozessbeschreibungen. Es ist daher ratsam bei der Prozessbeschreibung Navigation und Aktivitäten von einander zu trennen. Die Abbildung 3a zeigt einen Ausschnitt aus dem von DELL. Die verwendete Notation ist an [4] angelehnt und entspricht keinem Standard. Abbildung 3b zeigt das komplette Navigationsschema. Die weißen Knoten, in der Abbildung mit „Node“ gekennzeichnet, beschreiben statische Informationsseiten im Onlineshop. Diese können jederzeit während des Prozesses aufgerufen werden. Die dunklen Knoten beschreiben Aktivitäten im Prozess des Einkaufens bei DELL. Die Kanten bedeuten Übergänge zwischen zwei Knoten. Bei den Aktivitäten deuten diese Kanten auf Zustandsübergänge hin. Übergänge im Bereich der Navigation sind zustandslos.

Da es durch den Browser möglich sein muss Aktivitäten durch Navigation zu verlassen, muss der Zustand des Prozesses erhalten bleiben. Dies wird durch die Kanten „Suspend“ und „Start&Resume“ verdeutlicht. Durch diese Trennung von Aktivitätsknoten und Navigationsknoten können Prozesse definiert werden, die eine eindeutige Abfolge von Aktivitäten besitzen und dennoch durch Navigation unterbrochen werden können, die sich aber nicht auf den Zustand des Prozesses auswirken.

3 Prozessmodellierung von DELL

Durch die implizite Navigation ist es möglich, übersichtliche Prozesse zu modellieren, die zum Beispiel dem schematischen Ablauf eines Bestellvorganges entsprechen. Nur wo Navigation ein wichtiger Teil eines Prozesses ist, wird diese noch als Aktivität modelliert. In den folgenden Beispielen betrifft das lediglich das Aufrufen des Onlineshops mit Hilfe eines Browsers. Im Folgenden soll der Geschäftsprozess einmal exemplarisch modelliert werden. Es wird ein „Bottom-Up-Ansatz“ gewählt. Das Prozessmodell wird dabei Schritt für Schritt erweitert.

3.1 Prozessmodelle

Die Abbildung 4 zeigt in BPMN-Notation die Aktivitäten des Modifizierens eines Warenkorbes und die Bestätigung der Bestellung. Die beiden großen Teilprozesse sind dabei in eine Transaktion eingebettet. Diese soll sicherstellen, dass es jederzeit möglich ist den Bestellvorgang abzubrechen, ohne dass Zustandsinformationen im System zurückbleiben. Dies wird dadurch erreicht, das Kompensationen an die beiden Teilprozesse angefügt wurden. Diese Kompensationen ermöglichen, dass bei Abbruch der Transaktion sämtliche Daten aus dem Warenkorb und die persönlichen Daten gelöscht werden.

Die beiden Prozesse warten auf Ereignisse, die vom Kunden ausgelöst werden. Das Modifizieren des Warenkorbes beinhaltet das Hinzufügen von Produkten, das Entfernen von Produkten, sowie das Laden und Sichern eines Warenkorbes. Beim Hinzufügen muss überprüft werden, ob ein Produkt vorrätig ist. Zu Gunsten der Übersichtlichkeit wurde diese zusammengesetzte Aktivität nicht näher verfeinert. Im Grunde genommen besteht sie aus dem Austausch von Informationen mit dem

Warenlager. Der komplexe Prozess, der aus dem Konfigurieren des gewählten Produktes und der Wahl von zusätzlichen Dienstleistungen besteht, wurde als Adhoc-Prozess modelliert. Dies beschreibt, dass nicht genau festgelegt ist, in welcher Reihenfolge und wie oft diese beiden Aktivitäten ausgeführt werden.

Für Entscheidungen, die von dem Auftreten eines bestimmten Ereignisses abhängen, bietet BPMN ein spezielles Gateway symbolisiert durch eine Raute mit einem Stern. Dieses Symbol erlaubt es tiefgeschachtelte Entscheidungen zu einer einzigen zusammenzufassen. Darüber hinaus besitzt dieses Symbol die Bedeutung, dass der Prozess erst instanziiert wird, wenn ein entsprechendes Ereignis eintrifft. Obwohl BPMN Datenflüsse nicht unmittelbar berücksichtigt, wurde für den Warenkorb ein Datenartefakt verwendet. Die drei Möglichkeiten verwenden das Datenobjekt, um den Warenkorb zu verändern. Der Einfluss der Teilprozesse auf das Datenobjekt wird durch Assoziationen angedeutet. Dies erleichtert das Verständnis für den Betrachter.

Auch der zweite große Prozess der Bestätigung einer Bestellung wird von einem Ereignis des Kunden angestoßen. Dabei beinhaltet der Prozess die Aktivitäten der Erhebung persönlicher Daten, sowie der Zahlungsweise. DELL bietet den Service an innerhalb von 24 Stunden den Kunden telefonisch zu kontaktieren, um noch einmal die Bestellung zu besprechen. Aus Kundensicht fährt der Prozess erst dann fort, wenn innerhalb von 24 Stunden ein Mitarbeiter vom Kundenservice angerufen hat. Es sei angemerkt, das hier Fehlerfälle nicht berücksichtigt werden. Ein Anruf findet dementsprechend in den 24 Stunden definitiv statt. Da nicht unbedingt alle drei Pfade für die Zahlungsmodalitäten in Kombination mit dem Rückruf durchlaufen werden, umschreibt ein komplexes Gateway diese situationsbedingte Entscheidung. Der Pfad des Rückrufes wird allerdings immer beschritten. Dies wird durch eine entsprechende Markierung am Beginn des Pfades symbolisiert. In Abhängigkeit der Zahlungsart wird eine Rechnung verschickt und auf ein Nachricht vom Finanzinstitut gewartet, dass der Rechnungsbetrag für die Bestellung eingetroffen ist. Dies ist eine Besonderheit bei DELL. Die Bestellungen von privaten Kunden werden erst versendet, wenn der Zahlungseingang verzeichnet wurde. Auffällig ist, dass zwischen dem Modifizieren des Warenkorb und der Bestellbestätigung kein direkter Sequenzfluss stattfindet, sondern lediglich eine Assoziation, die ausdrückt, dass die Bestellbestätigung kausal mit dem Warenkorb zusammenhängt. Tatsächlich werden die beiden Prozesse durch den Kunden in ihrer Abfolge gesteuert.

Als nächstes soll betrachtet werden, wie sich die Abwicklung des eigentlichen Bestellvorganges in den kompletten Prozess des Einkaufens bei DELL integrieren lässt. Die eben betrachtete Transaktion ist nur noch ein Teil des kompletten Prozesses. In der Abbildung 5 wird deutlich, wer an den Prozessen beteiligt ist und wo welche Prozesse genau stattfinden. Die Blackbox „Kunde“ beinhaltet nun diverse Aktivitäten, darunter auch die, die für Ereignisse beim Modifizieren des Warenkorbes verantwortlich sind. Die erste Aktivität des Kunden ist die einzige explizit modellierte Navigation. Diese beinhaltet das Aufsuchen der eigentlichen Webseite. Die Komposition „Do shopping“ beinhaltet auch Navigation, allerdings implizit modelliert durch die Aktivität „Forage for product“. Die Aktivitäten des komplexen Prozesses „Do shopping“ sind adhoc modelliert, da nicht genau festgelegt ist, wann und wie oft eine dieser Aktivitäten ausgeführt wird. Mit der Aktivität „Confirm Order“ beginnt die Kette von Aktivitäten, die die Bezahlung der Bestellung

abwickeln. Die Kommunikation findet in Form einfacher Nachrichten statt. Nachrichtenflüsse und entsprechende Ereignisse verdeutlichen dies. Das Finanzinstitut spielt bei der Kommunikation und dem Fortschreiten des Prozesses eine wichtige Rolle; je nachdem, ob mit Kreditkarte oder per Rechnung bezahlt wird, wird die Kontobelastung von DELL oder vom Kunden selbst initiiert. Wieder ist die Wahl der entsprechenden Aktivität, die als nächstes ausgeführt wird vom eintreffenden Ereignis abhängig. Daher wurde hierfür ein Gateway verwendet, dessen Entscheidung durch das Auftreten bestimmter Ereignisse beeinflusst wird. In beiden Fällen erhält DELL bei Erfolg eine Benachrichtigung über den Zahlungseingang. Erst diese Nachricht veranlasst DELL das Warenlager mit der Auslieferung der Bestellung zu beauftragen. Das Warenlager verpackt die Bestellung und verschickt diese an den Kunden. Betrachtet man die Abbildung 5 fällt auf, dass die Veränderung des Warenkorbes und die Bestellbestätigung bei DELL erfolgen und nicht beim Kunden. Auffällig ist auch, dass bei DELL selbst kein online Produktkatalog angeboten wird. Die Abstraktion vom Produktkatalog und damit der Navigation auf der Webseite, sowie die Verlagerung der Bestellvorgänge weg vom Kunden erlauben, dass diese internen Prozesse nicht nur für den Onlinebestellvorgang genutzt werden können.

3.2 Austauschbarkeit von Prozessen

Der Verzicht auf die explizite Modellierung von Navigation ermöglicht es, Teilprozesse wie zum Beispiel die komplette Bestellabwicklung innerhalb von DELL wiederzuverwenden. Ein Unternehmen könnte davon profitieren, wenn nicht für jeden neuen Vertriebskanal neue interne Prozesse benötigt würden. Es ist denkbar, dass es intern keinen Unterschied macht, ob ein Kunde seine Bestellung online selbst aufgibt oder mit Hilfe einen Kundendienstmitarbeiters von DELL. Bei der Modellierung wurde das aus der Softwareentwicklung bekannte Proxy-Pattern verwendet. Dieses Pattern dient dazu, dass Stellvertreter die Manipulation von Datenobjekten tätigen, wenn es keine direkte Möglichkeit der Manipulation gibt. Beim Onlineeinkauf konnte noch direkt durch den Kunden Einfluss auf den Warenkorb und die Bestellung genommen werden. Bei der Telefonbestellung erledigt dies nun der stellvertretende Mitarbeiter von DELL. Die Abbildung 6 soll verdeutlichen, wie unter Einsatz dieses Pattern und durch den Verzicht auf Modellierung von Navigation Teilprozesse, zum Beispiel das Einkaufen des Kunden per Browser durch einen Telefondienst ersetzt werden können.

Die bereits bekannten Pools Finanzinstitut, Warenlager und die Bestellabteilung wurden mit samt den Aktivitäten zu einfachen Pools reduziert. Da die internen Abläufe unverändert bleiben, kann so eine bessere Übersichtlichkeit erreicht werden.

Interessant sind die oberen beiden Pools. Die Hotline und ein Kunde, der mit der Hotline kommuniziert, ersetzen den Kunden, der über den Onlineshop mit DELL direkt kommunizierte.

Damit die internen Prozesse bei DELL möglichst beibehalten werden können, muss die Bestellung über einen Telefonservice die gleichen Ereignisse auslösen, die auch bei der Onlinebestellung ausgelöst wurden. So könnte man sich gut vorstellen, dass der Telefonmitarbeiter ein ähnliches Formular mit den Daten ausfüllt, die ihm am Telefon mitgeteilt werden, wie der Kunde mit der Onlinebestellung. So wird dem

Mitarbeiter am Telefon gesagt, dass der Kunde ein bestimmtes Produkt erwerben möchte; der Mitarbeiter legt das Produkt dann für den Kunden in einen entsprechenden Warenkorb. Analog ist der Ablauf bei der Bestätigung der Bestellung durch den Kunden. Die Entscheidung auf Seiten des Telefondienstes ist durch das eintreffende Ereignis vom Kunden beeinflusst. Daher wurde an dieser Stelle ein Gateway verwendet, das diesen Sachverhalt symbolisiert. Der Mitarbeiter kann die Bestätigung der Bestellung nach Rücksprache mit dem Kunden abschließen. Wichtig für die Wiederverwendbarkeit der internen Prozesse ist hierbei, dass identische Ereignisse vom Telefonmitarbeiter erzeugt werden. In der Abbildung 6 sind die beiden Ereignistypen zwischen der DELL-Hotline und dem DELL-Store mit den Ereignissen der Onlinebestellung identisch. Nachdem der Kunde seine Bestellung über das Telefon getätigt hat, sind die nachfolgenden Aktivitäten, wie zum Beispiel die Bezahlung der Ware oder das Ausliefern der Bestellung die gleichen wie bei dem Onlineverfahren. In den letzten beiden Abbildungen ist zu erkennen, dass der dargestellte Prozess nicht zwangsläufig zu Ende sein muss. Im Prinzip gibt es nach dem Erhalt der Ware noch die Möglichkeit, Teile der Bestellung zu reklamieren oder andere Serviceleistungen in Anspruch zu nehmen. Da bei dieser Arbeit ein Schwerpunkt auf den Aspekt des E-Commerce gelegt wurde, wird auf diese eher traditionellen Serviceleistungen nicht genauer eingegangen. Diese Prozesse der Produktbestellung weisen keine nennenswerten Besonderheiten im Privatkundenbereich auf.

4 BPMN im Vergleich

Nach der Modellierung der wichtigsten Prozesse im Onlineshop von DELL stellt sich die Frage, wie gut sich das dafür verwendete Modellierungswerkzeug BPMN geeignet hat. Wo zeigten sich Unzulänglichkeiten und wie schneidet BPMN im Vergleich mit anderen Modellierungssprachen wie zum Beispiel YAWL ab?

Damit eine vernünftige Ausgangsbasis für einen Vergleich zwischen BPMN und YAWL geschaffen werden kann, wurde der Prozess der Bestellbestätigung aus Abbildung 4 einmal in YAWL modelliert. Eine kurze Einführung in der Notation von YAWL kann man [5] entnehmen. Als erstes fällt in der Abbildung 7 auf, dass YAWL sich auf wenige schlichte Symbole beschränkt. Auffällig ist die Darstellung von Abbruchmöglichkeiten. Der alternative Weg zur Aktivität „Clear personal data“ bietet die Möglichkeit, dass alle Marken aus dem Bereich der gepunkteten Linie entfernt werden. Ohne Marken können diese Aktivitäten nicht mehr schalten, der Prozess wird abgebrochen. Allerdings werden bereits ausgeführte Aktivitäten nicht rückgängig gemacht. BPMN bietet mit Transaktionen und Kompensationen dort eine Alternative, bei der der resultierende Zustand eindeutiger definiert ist.

Nach der Aktivität „Collection of personal data“ ist man zuerst versucht eine „deferred Choice“ zu modellieren, da es aber möglich sein muss, sowohl Ratenzahlung, als auch eine Abbuchung von der Kreditkarte zu wählen, wurde an dieser Stelle das Pattern für „multiple Choice“ verwendet. Dies ermöglicht das jede Kombination von Aktivitäten ausgeführt werden kann. Hier zeigt sich auch deutlich ein Nachteil von YAWL. Es gibt keine Möglichkeit durch ein einfaches Symbol

auszudrücken, dass ein bestimmter Pfad immer ausgeführt werden muss. In diesem Fall der Pfad, der zur Aktivität „Callback customer“ führt. Möchte man aufwendige Konstruktionen vermieden, kann man hervorheben, dass diese Entscheidung auf Grund von Prädikaten gefällt wird. Diese Prädikate müssen so gewählt sein, dass die Aktivität „Callback customer“ immer ausgeführt wird. Die Prozessbeschreibung in YAWL zeigt den sehr abstrakten Einsatz von Ereignissen in Form von T- und E-Aktivitäten. Verwendet wurden hier nur Events um Nachrichteneingänge zu symbolisieren. Vor dem „Send invoice“ findet man einen And-Split, da die Rechnung immer verschickt wird. Anschließend wurde eine „deferred Choice“ verwendet, um auszudrücken, das entweder DELL selbst das Konto des Kunden belastet, oder aber die Eingangsbestätigung der Überweisung des Kunden eintrifft. Da nur einer der beiden Pfade ausgeführt wird, wurde hier ein nichtsynchronisierter Merge verwendet. Anschließend wird die Bezahlung verbucht.

Die Abbildung 7 zeigt, dass es mit YAWL durchaus möglich ist, die Prozesse von DELL abzubilden. Beide Notationen decken darüber hinaus sämtliche Designpattern ab. Die Abbruchaktivität von YAWL und die Transaktion bei BPMN sind beide Vertreter des Chancelation-Pattern. Doch während YAWL dies ausschließlich mit der Verwendung von einfachen Symbolen wie Stellen und Marken ausdrücken kann, verwendet BPMN eine Vielzahl von komplexen Elementen, die in der Regel eine komplizierte Logik oder mehrere elementare Aktivitäten beinhalten.

Diagramme, die mit Hilfe von YAWL erstellt werden, sind daher in der Regel schneller verständlich. Besonders für Menschen, denen die BPMN-Notation nicht so vertraut ist, gestaltet es sich oft schwierig, die teilweise doch sehr kompakten Netze zu verstehen. Erschwert wird das Verständnis auch durch die implizite Modellierung von Sachverhalten. So lässt BPMN einem den Freiraum, Stellen, die den Start und das Ende eines Prozesses symbolisieren, mit in ein Netz aufzunehmen oder nicht. Dadurch wird es besonders in umfangreichen Netzen oft schwierig, den Beginn oder das Ende einzelner Prozesse zu identifizieren. Da BPMN sich nicht nur auf einfache Symbole beschränkt, besteht die Möglichkeit gewisse Abläufe auf verschiedene Art und Weise zu modellieren. Dies trägt wenig zur Lesbarkeit bei, muss doch oft erst über die Modeliierung eines „deferred Choice“ nachgedacht werden, wenn es dafür verschiedene Möglichkeiten gibt.

Einige Vorteile von YAWL sind unter anderem, dass es mittlerweile Werkzeuge gibt, die den Entwurf von Prozessen in YAWL unterstützen sollen. Allerdings sind diese Werkzeuge teilweise noch sehr ungenügend und ihre Bedienbarkeit gestaltet sich als unzulänglich. Ist die Prozessmodellierung mittels YAWL abgeschlossen, biete YAWL die Möglichkeit mit Hilfe spezieller Laufzeitumgebungen, die Ausführung eines Prozesses zu beobachten. Man besitzt die Möglichkeit zu testen, ob die modellierten Prozesse sich wie geplant verhalten.

Die komplexen Elemente in der Notation von BPMN beruhen auf dem Ziel der Abbildbarkeit von BPMN auf BPEL. BPEL ist eine Ausführungbeschreibung von Prozessen im Computer. Ziel von BPMN ist es aus den Prozessentwürfen direkt kompilierbaren Quellcode zu erzeugen. Damit dies gelingen kann, sind komplexere Elemente nötig, da diese sich teilweise leicht auf Sprachkonstrukte von Programmiersprachen abbilden lassen und so eine Übersetzung vereinfacht werden kann. Da für die Übersetzung nach BPEL sehr genau definiert sein muss, nach

welchen Kriterien und Daten an Entscheidungspunkten eine Wahl getroffen wird, benötigt BPMN eine Möglichkeit zusätzliche Informationen in Form von Attributen über das Verhalten in eine Prozessbeschreibung zu integrieren. Diese Attribute sind sehr mächtig, erlauben sie doch eine gute Übersetzbarkeit nach BPEL.
Integriert man Attribute in eine Prozessbeschreibung, wächst die Komplexität diese Beschreibungen so stark an, das diese unübersichtlich und schwer zu verstehen sind. Es ist daher immer drauf zu achten, welche Person eine Prozessbeschreibung in BPMN lesen soll.

5 Schlussfolgerung

BPMN liegt seit Mai 2004 in der Version 1.0 vor. Die komplexen Strukturen und die Möglichkeit, jedem Element Attribute zu zuweisen, erlauben eine gute Abbildung auf Beschreibungssprachen wie BPEL, die eine Erzeugung von ausführbaren Quellcode erlauben. Da Prozessmodelle aber auch dafür gedacht sind Unternehmensstrukturen und Abläufe innerhalb von Unternehmen für andere visuell aufzubereiten, muss bei der Modellierung von BPMN immer die Zielgruppe im Auge behalten werden.

Da es bei dem Onlineshop DELL hauptsächlich um die Aufbereitung von Arbeitsvorgängen im Bereich des E-Commerce ging, wurde auf die Vergabe von Attributen aus didaktischen Gründen verzichtet. Dennoch zeigte sich, dass BPMN auch ohne Attribute mit einigen Problemen behaftet ist.

So gibt es zur Zeit noch keine Werkzeugunterstützung, die speziell für BPMN entwickelt wurde. Im Moment ist man noch auf herkömmliche Entwurfs- und Zeichenwerkzeuge angewiesen. Da BPMN noch wenig erprobt ist und sozusagen im Moment noch Pionierarbeit geleistet wird, fehlt es oft an Referenzen, die Hinweise auf eine korrekte Modellierung, oder zumindest einen Anstoß in die richtige Richtung liefern. Die 300 Seiten umfassende Spezifikation von BPMN erfordert einige Einarbeitungszeit, und selbst danach bedarf es einiger Übung und eines gewissen Entwicklungsprozesses, bis man in der Lage ist, übersichtlich und leicht verständlich Prozesse in BPMN zu modellieren. Durch den großen Sprachumfang ist man dazu verleitet, komplizierte Sachverhalte durch spezielle Symbole darzustellen. Dies ermöglicht zum einen kompaktere Diagramme, wie der Vergleich der Bestellbestätigung mit YAWL gezeigt hat. Zum anderen sind die Diagramme, die den Sprachumfang von BPMN ausnutzen, besonders für Laien oft nur durch Kommunikation mit der Entwurfsperson nachzuvollziehen. Es sollte daher beachtet werden, dass man nicht versuchen sollte den Sprachumfang unnötigerweise voll auszuschöpfen, sondern immer darauf zu achten, welchen Zweck eine Prozessbeschreibung in BPMN dienen soll. Auf Grund der Mächtigkeit von BPMN ist es allerdings für viele Aufgabenbereiche geeignet, auch wenn zur Zeit besonders im Bezug auf die Abbildung nach BPEL viele Punkte der tatsächlichen Umsetzung noch nicht vollständig geklärt sind.

Die Schwierigkeit, Prozesse im Bereich des E-Commerce zu modellieren, lag weniger in der Verwendung von BPMN, als in fehlenden Referenzen. Es gibt einige

Dokumente [4,6], die sich bereits mit dieser Problematik beschäftigen. Diese beschränken sich aber größtenteils auf den Aspekt der Navigation in Verbindung mit Prozessen und wie eine vernünftige Trennung erreicht werden kann. Jedoch fehlte es diesen Arbeiten an einer detaillierten Prozessbeschreibung. Diese Problematik wurde hier aufgegriffen. Bei der Modellierung von DELL wurden in Anlehnung an [4,6] zuerst statische Modelle entwickelt. Anschließend wurde der dynamische Aspekt der Navigation in Verbindung mit der Prozessmodellierung beleuchtet. Die identifizierten Aktivitäten wurden dann mit Hilfe von BPMN modelliert. Am Ende erhält man eine recht vollständige Beschreibung des Onlineshops von DELL mit Prozessmodellen, die sich über das Anwendungsgebiet des E-Commerce hinaus verwenden lassen.

6 Bibliographie

[1] Introduction to BPMN , http://www.bpmn.org, 2004
[2] BPMN Spezifikation 1.0, http://www.bpmn.org, 2004
[3] Fundamental Modeling Concepts, http://fmc.hpi.uni-potsdam.de, 2004
[4] Designing Business Processes in E-commerce Applications, Hans Albrecht Schmid, Gustavo Rossi, Springer Verlag, 2002
[5] YAWL, yet another worklflow language, http://www.citi.qut.edu.au/yawl/index.jsp, 2004
[6] Designing Business Processes in E-Commerce Applications, Hans Albrecht Schmid, Gustavo Rossi, IEEE Computer Society, 2004
[7] Design and implementation of the YAWL system, W.M.P van der Aalst, L. Aldred, M. Dumas, 2003

7 Anhang

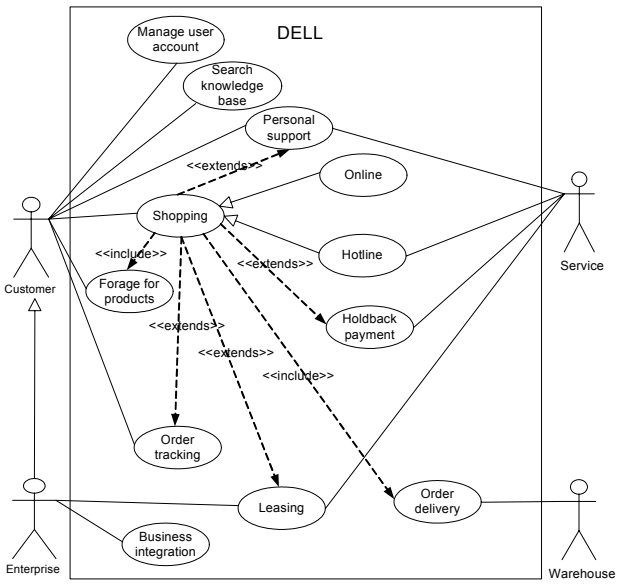


Fig. 1. DELL Use-Case-Diagramm

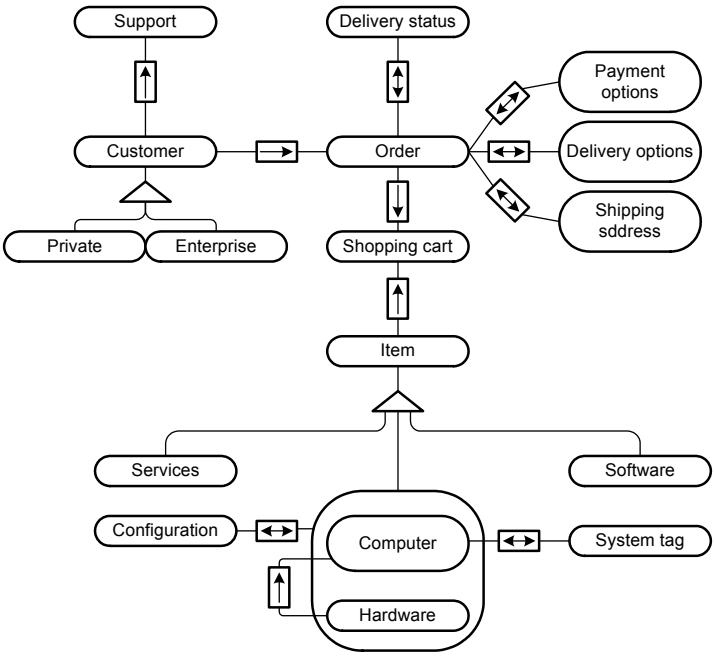


Fig. 2. DELL ER-Diagramm in FMC-Notation

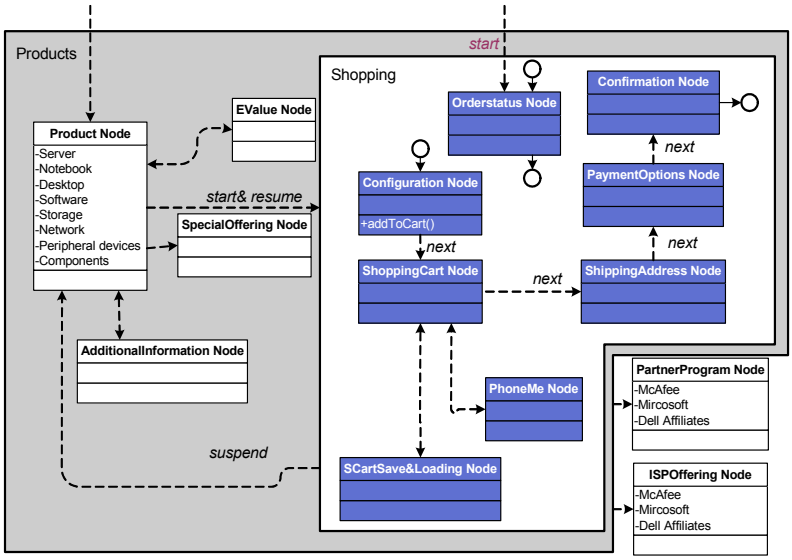


Fig. 3a. Identifikation von Navigations- und Aktivitätsknoten

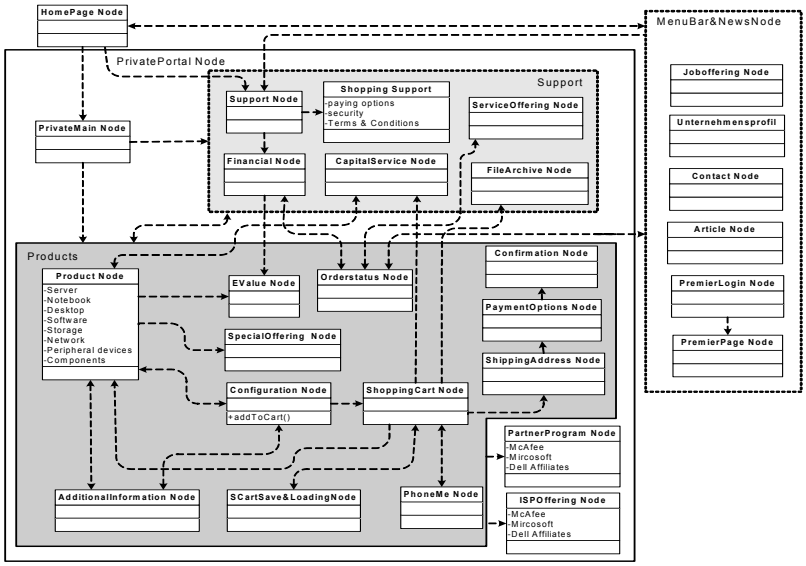


Fig. 3b Navigationsschema von DELL - Privatpersonenportal

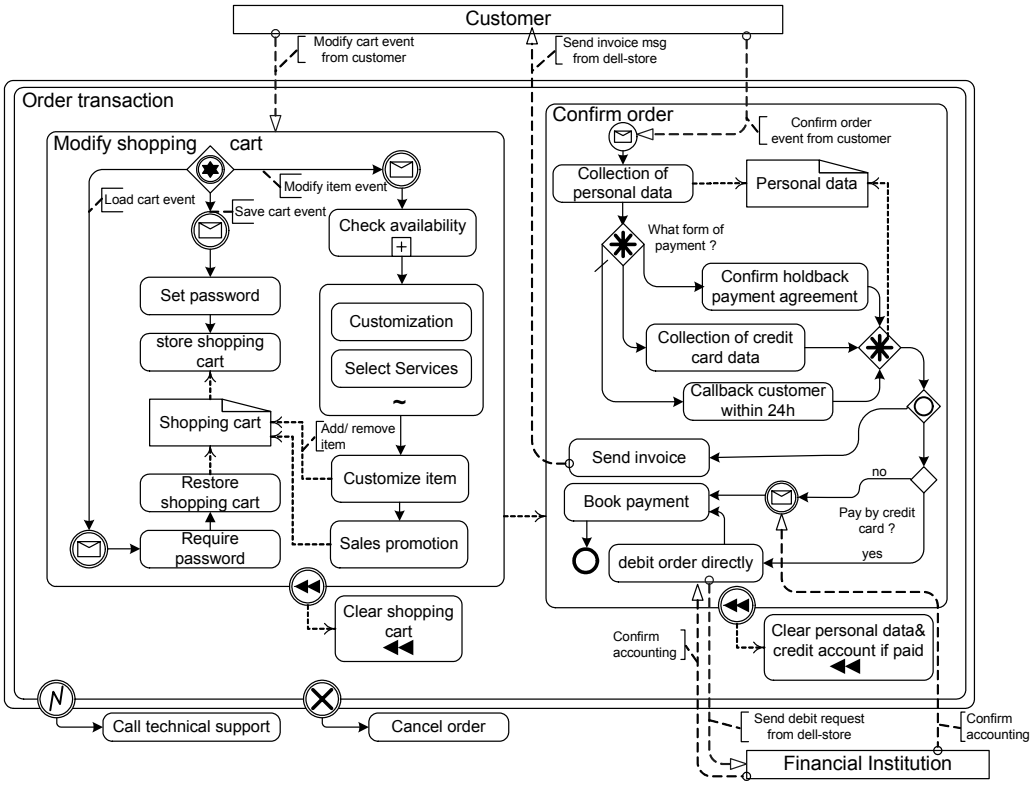


Fig. 4. Prozess: Warenkorb modifizieren und Bestellbestätigung

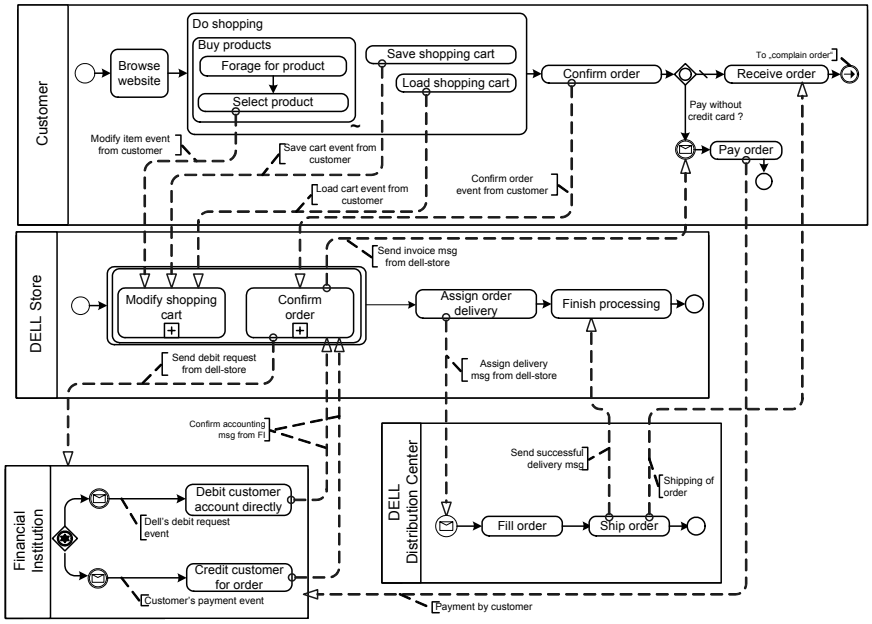


Fig. 5. DELL - Institutionen und ihr Zusammenwirken

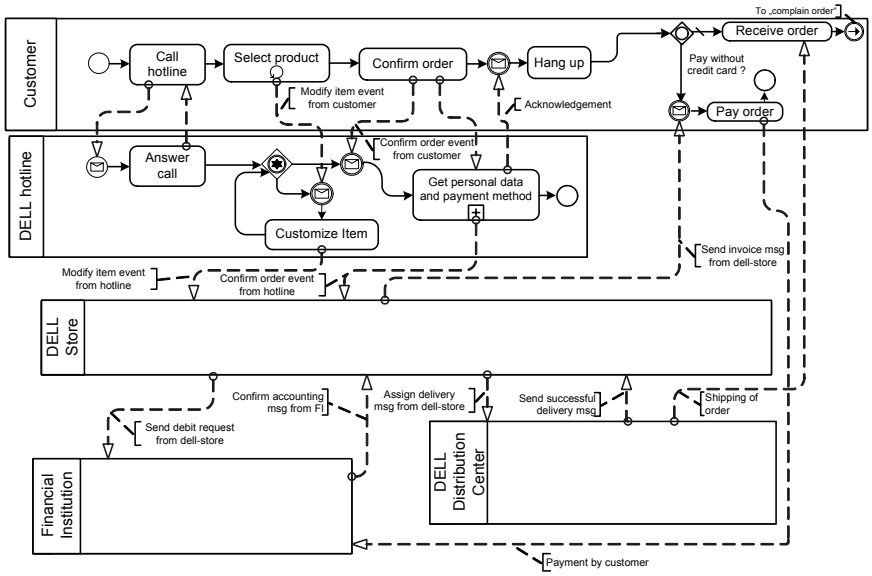


Fig. 6 DELL – Wiederverwendbarkeit von Prozessen

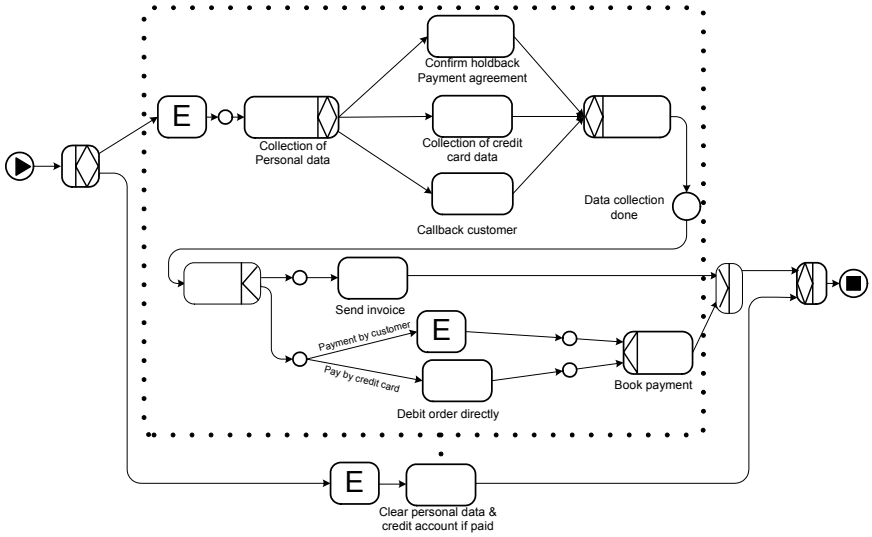


Fig. 7 Prozessmodellierung in YAWL

Ebay

Dominik R. Tornow

Hasso-Plattner-Institut
Seminar Prozessmodellierung
dominik.tornow@hpi.uni-potsdam.de

Abstract. Im Rahmen des Seminars Prozessmodellierung und vor dem Hintergrund des PESOA-Forschungsprojekts diskutiert dieses Paper die Erfahrungen, die bei der Modellierung der Online-Auktionsplattform Ebay gesammelt wurden.

Motivation

Einordnung

Im Rahmen des Seminars Prozessmodellierung wurden Techniken zur Erhebung und Modellierung von Prozessen erarbeitet und diskutiert, um unter anderem Prozesse aus dem Bereich e-Business zu erfassen.

Das Seminar fand vor dem Hintergrund des PESOA-Projekts [1] statt, das 'Methoden und Techniken zur Modellierung von Prozessen in Software-Produktlinien' [2] untersucht und entwickelt; Schwerpunkte sind dabei Techniken des Domain Engineering.

Domain Engineering beschäftigt sich mit Methoden und Techniken, die von Beginn an auf die Entwicklung von Softwarefamilien und nicht auf die Entwicklung einzelner, voneinander unabhängiger Softwareprodukte gerichtet sind: anstatt Systeme individuell zu modellieren und zu entwickeln, wird beim Domain Engineering jeweils eine ganze Softwarefamilie innerhalb einer Domäne betrachtet, die aus Sicht der *Gemeinsamkeiten* und *Variabilitäten* von Softwareprodukten abgegrenzt wird.

Aufgabenstellung

Diese Seminararbeit hat das Ziel, die wesentlichen Prozesse des Online-Auktionshauses Ebay zu erfassen und zu modellieren.

Während ich mich in meinem Vortrag auf die Beschreibung der Prozessmodelle selbst konzentriert habe, werde ich in dieser Ausarbeitung die Erfahrungen, die ich beim Erstellen dieser Modelle gemacht habe, ins Zentrum der Betrachtung stellen.

Ich stelle dabei folgende Frage in den Raum:

> „Wie können die statischen und dynamischen Aspekte eines Software-Systems vor dem Hintergrund der Produktlinienentwicklung *geeignet* modelliert werden?“

Architektur-Framework

Zweifelsohne existieren zahlreiche Möglichkeiten ein Software-System zu beschreiben bzw. zu modellieren – die Vielfalt dieser Möglichkeiten wirft jedoch die Frage auf, welche Kriterien ein Modell erfüllen muss, um in einem gegebenen Kontext anwendbar zu sein.

Zwei Fragestellungen, die im Laufe des Seminars immer wieder diskutiert wurden, haben mich in diesem Zusammenhang besonders beschäftigt:

1. „Welche Betrachtungsebene soll für das Erarbeiten eines Prozessmodells eingenommen werden?“

Beispiel

Gerade im Themenbereich des e-Business kam die Diskussion auf, ob und wie die Interaktion des Browsers mit dem Webserver modelliert werden soll. Dabei waren unter den Seminarteilnehmen durchaus verschiedene Meinungen vertreten.

2. „Welche Aspekte eines Systems müssen modelliert werden, um die Prozesse des Systems hinreichend beschreiben zu können?“

Beispiel

Ziel des Seminars war es, die Prozesse verschiedener Systeme zu beschreiben – dennoch hat sich ein Großteil der Seminarteilnehmer entschlossen, auch den Aufbau der Systeme zu betrachten.

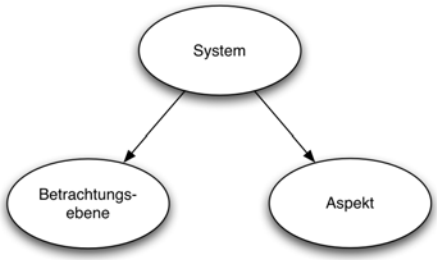


Fig. 0. Kriterien für Systemmodelle

Wie in Abbildung **0** gezeigt, können sich Modelle zu einem System in zwei orthogonal zueinender liegenden Kategorien unterscheiden, die im Folgenden näher vorgestellt werden.

Betrachtungsebene

Jedes Modell kann einer Betrachtungsebenen zugeordnet werden – die Höhe dieser Betrachtungsebene ist dabei ein Maß für die Abstraktion, die angewendet wurde, um dieses Modell zu erstellen.

Zwei Modelle auf unterschiedlichen Betrachtungsebenen, die den selben Sachverhalt widerspiegeln, stehen dabei in einer Implementierungsbeziehung – Andreas Bungert schreibt dazu in [3]:

> Die Sachverhalte einer höheren Betrachtungsebene werden durch die Sachverhalte der jeweils niedrigeren Ebene realisiert bzw. implementiert. Sachverhalte der höheren Ebene werden als *Primäre Sachverhalte* bezeichnet, da sie sich direkt aus der Aufgabe des betrachteten Systems ergeben. Sachverhalte tieferer Ebenen werden als *Sekundäre Sachverhalte* bezeichnet, da sie nur als 'Mittel zum Zweck' auftreten. (**1**)

Modelle der höchsten Betrachtungsebene werden als *Anforderungsvollständige Modelle* bezeichnet – sie dienen der Spezifikation der statischen und dynamischen Aspekte eines Systems, wobei technische Details unberücksichtigt bleiben.

Modelle der tiefsten Betrachtungsebene werden als *Fertigungsvollständige Modelle* bezeichnet – sie dienen als Ausgangspunkt für die Implementierung einer Applikation und spezifizieren alle technischen Details.

Besonders spannend ist die Tatsache, dass zwei Kategorien von Variabilitäten identifiziert werden können:

1. Variabilitäten innerhalb einer Betrachtungsebene.

Variabilitäten im Sinne der Produktlinienentwicklung treten nur innerhalb einer Betrachtungsebene zum Vorschein – sie stellen Variationen der Anforderungen dar und spannen somit eine Applikations-Domäne auf.

2. Variabilitäten beim Übergang zwischen zwei Betrachtungsebenen.

Variabilitäten, die sich beim Übergang zwischen zwei Betrachtungsebenen ergeben, stellen nichts weiter als eine Implementierungsentscheidung dar – eine Entscheidung WIE etwas realisiert wird, jedoch nicht WAS realisiert wird.

Anmerkung • Variabilitäten, die beim Übergang zwischen zwei Betrachtungsebenen auftreten, können mit Hilfe der Model Driven Architecture [4] beherrscht werden.

Als Beispiel dafür, dass Variabilitäten beim Übergang zwischen zwei Betrachtungsebenen keinen Einfluss auf das Modell der höheren Betrachtungsebene haben, soll der Log-in Prozess der Ebay-Plattform dienen.

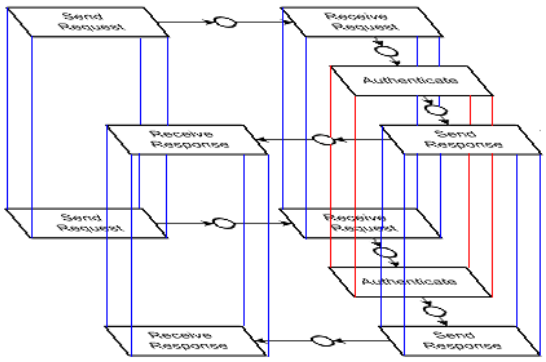


Fig. 1. Log-in Prozess

Abbildung **1** illustriert, wie das anforderungsvollständige Modell über einen Implementierungsschritt in das fertigungsvollständige Modell überführt wird.

Während auf höherer Ebene nur gefordert wird, dass ein Nutzer authentifiziert wird, wird auf der tiefer gelegenen Ebene spezifiziert, wie das geschieht – dazu könnte z.B. ein Datenbanksystem oder ein Verzeichnisdienst herangezogen werden.

Aspekte

Neben der Betrachtungsebene kann jedes Modell einer Sicht, einem sog. Viewpoint zugeordnet werden – die IEEE Computer Society schreibt dazu in [5]:

A viewpoint is a convention for constructing and using a view. [...] A view is a representation of a system from the perspective of a related set of concerns. (**2**)

Ein Viewpoint kann also als ein bestimmter Aspekt eines Systems verstanden werden – die Applikation wird aus einem bestimmten Blickwinkel heraus betrachtet.

Die IEEE Computer Society lässt jedoch völlig offen, welche Sichten zu beschreiben sind – statt dessen ist für jede konkrete Aufgabenstellung individuell festzulegen, welche Aspekte zu betrachten sind.

Wie bereits erwähnt, ist es die Aufgabenstellung dieser Seminararbeit, die Prozesse der Ebay-Plattform zu erfassen. Ist es daher ausreichend, sich auf das Prozessmodell des Online-Auktionshauses zu konzentrieren?

Siegfried Wendt schreibt dazu in [6]:

Ein dynamisches System ist ein konkretes oder zumindest konkret vorstellbares Gebilde, welches ein beobachtbares Verhalten zeigt, wobei dieses Verhalten als Ergebnis des Zusammenwirkens der Systemteile angesehen werden kann. (**3**)

Helmut Balzert konkretisiert diese Aussage in [7]:

Der statische Aufbau eines Systems bedingt das dynamische Verhalten des Systems. (**4**)

Diese kausale Abhängigkeit bewirkt, dass es notwendig ist, die statischen Aspekte einer Applikation zu modellieren, um die dynamischen Aspekten dieser Applikation hinreichend zu beschreiben – dies führt direkt zu dem in Abbildung **2** gezeigten Vorgehensmodell.

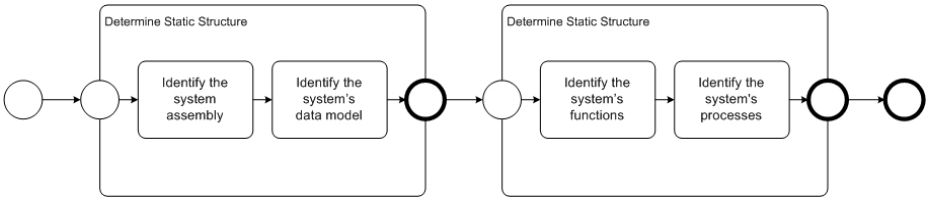


Fig. 2. Vorgehensmodell

Modelle der Ebay-Plattform

In diesem Abschnitt werden die erarbeiteten Systemmodelle der Ebay-Plattform vorgestellt – einleitend und als Zusammenfassung des vorangegangenen Abschnitts werden in Abbildung **3** die verwendeten Modelle zusammengefasst.

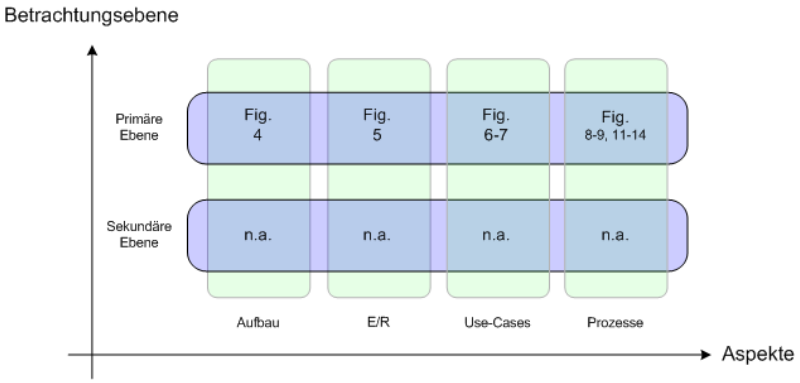


Fig. 3. Übersicht

Aufbau

Abbildung 4 zeigt den konzeptionellen Aufbau der Ebay-Plattform.

Der Aufbau eines Systems legt die Systemkomponenten und deren Schnittstellen fest. Diese Tatsache spiegelt sich überdeutlich im Prozessmodell wider: Systemkomponenten können nur miteinander interagieren, wenn sie über Schnittstellen miteinander verbunden sind. Dementsprechend ist der Nachrichtenfluss zwischen Systemkomponenten im Prozessmodell beschränkt.

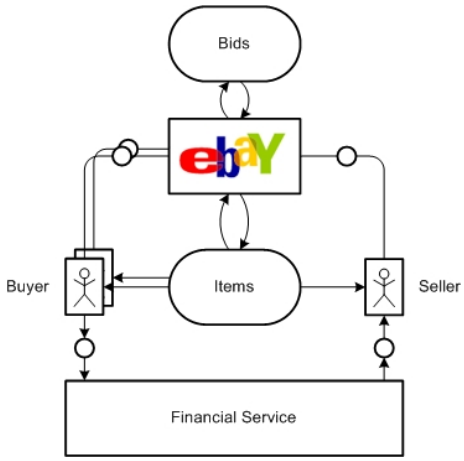


Fig. 4. Aufbau der Ebay-Plattform

Hier können, neben der Ebay-Plattform selbst, die Rollen des Verkäufers, des Käufers und des Finanzdienstleisters identifiziert werden. Käufer und Verkäufer interagieren über die Ebay-Plattform und über den Finanzdienstleister miteinander – so wird sowohl die Versteigerung als auch die Bezahlung eines Artikels abgewickelt.

Datenstrukturen

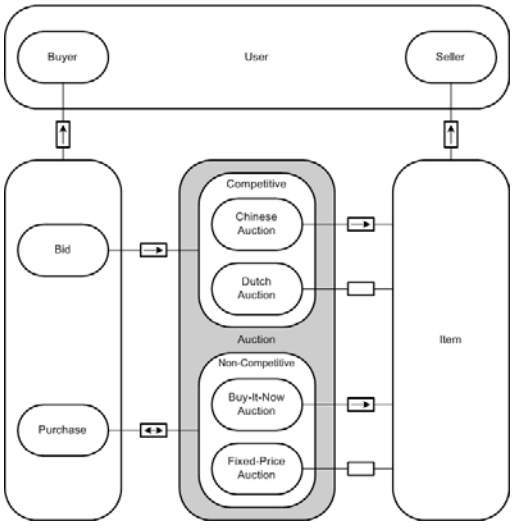


Fig. 5. Entity/Relationship der Ebay-Plattform

Abbildung **5** zeigt die an einer Versteigerung beteiligten Entitäten und deren Beziehungen untereinander. Ein Verkäufer kann einen bzw. mehrere Artikel anbieten, die ein Käufer ersteigern bzw. unmittelbar erwerben kann.

Funktionen

Die Abbildungen **6** und **7** zeigen die Top-Level Use-Case Diagramme der Ebay Plattform.

Ein Use-Case Diagramm zeigt die möglichen Interaktionstypen zwischen Akteuren (Personen und Systeme) und Use-Cases. Ein Use-Case ist dabei in [8] wie folgt definiert:

> The purpose of a use-case is to define the functionality of a system without revealing the internal structure of the system. That is, a single use-case specifies a single service the system provides. The complete set of use-cases describes all different ways to interact with system. (**5**)

Ein Use-Case beschreibt das Systemverhalten aus Benutzer-orientierter Sicht; d.h. es wird aufgezeigt, WAS das System leistet, jedoch nicht WIE. Das bedeutet insbesondere, dass ein Use-Case keine Ablaufbeschreibung darstellt! Statt dessen definiert die Menge aller Use-Cases die Funktionalität, die ein System bereit stellt.

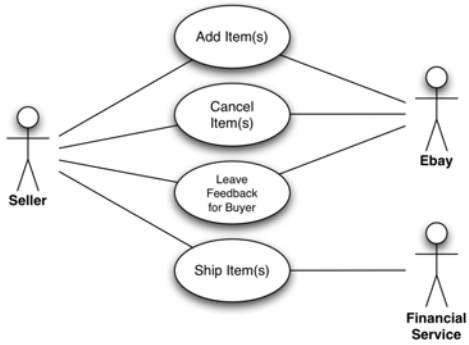


Fig. 6. Dienstrepertoire, das dem Verkäufer zur Verfügung steht

Das Use-Case Diagram aus Abbildung **6** stellt das Dienstrepertoire der Ebay Plattform dar, das dem Verkäufer zur Verfügung gestellt wird. Die zentralen Funktionen sind das Anbieten und Versenden eines Artikels.

Anmerkung • Auf den ersten Blick stellt sich die Frage, warum der Financial Service am Ausliefern der gekauften Ware beteiligt ist. Bei genauerer Betrachtung wird klar,

dass ein Verkäufer in der Regel gekaufte Ware erst dann ausliefert, wenn er eine Bestätigung des erwarteten Zahlungseingangs erhalten hat.

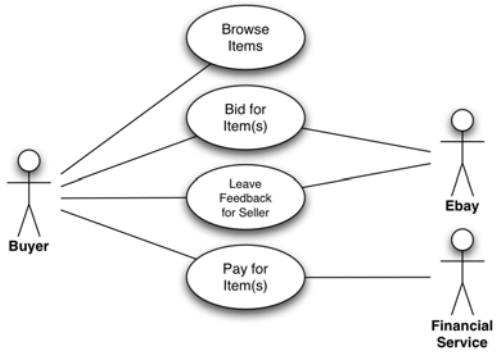


Fig. 7. Dienstrepertoire, das dem Käufer zur Verfügung steht.

Das Use-Case Diagram aus Abbildung **7**. stellt das Dienstrepertoire der Ebay Plattform dar, das dem Käufer zur Verfügung gestellt wird. Die zentralen Funktionen sind das Bieten und Bezahlen für einen Artikel.

Prozesse

In diesem Teilabschnitt werden die Prozesse modelliert, die über das Online-Auktionshaus Ebay abgewickelt werden – dazu wird gemäß der Aufgabenstellung die Business Process Modelling Notation (BPMN) 1.0 [9] verwendet.

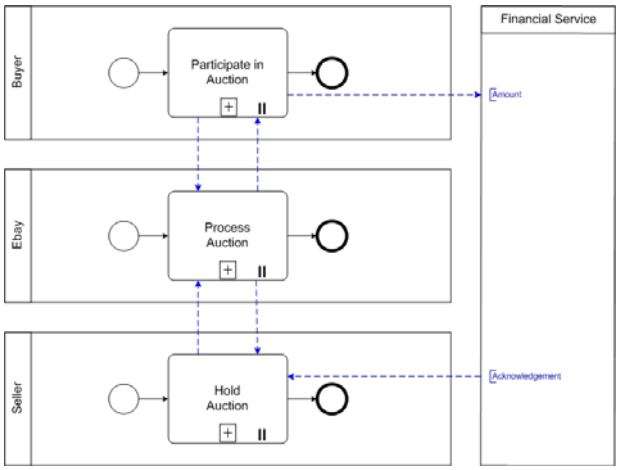


Fig. 8. Top-Level Prozessmodell der Ebay-Plattform

Abbildung **8** zeigt das Top-Level Modell der Ebay-Plattform, das sich, wie bereits geschildert, aus dem Aufbaumodell ableiten lässt:

Wie bereits erwähnt, treten Nachrichtenflüsse nur zwischen Komponenten auf, die im Aufbaumodell über Kanäle miteinander verbunden.

An dieser Stelle zeichnet sich BPMN besonders dadurch aus, dass zwischen Nachrichtenflüssen innerhalb einer Komponente und Nachrichtenflüssen zwischen zwei Komponenten unterschieden wird – die Kompatibilität des Prozessmodells zum Aufbaumodell lässt sich so leicht überprüfen.

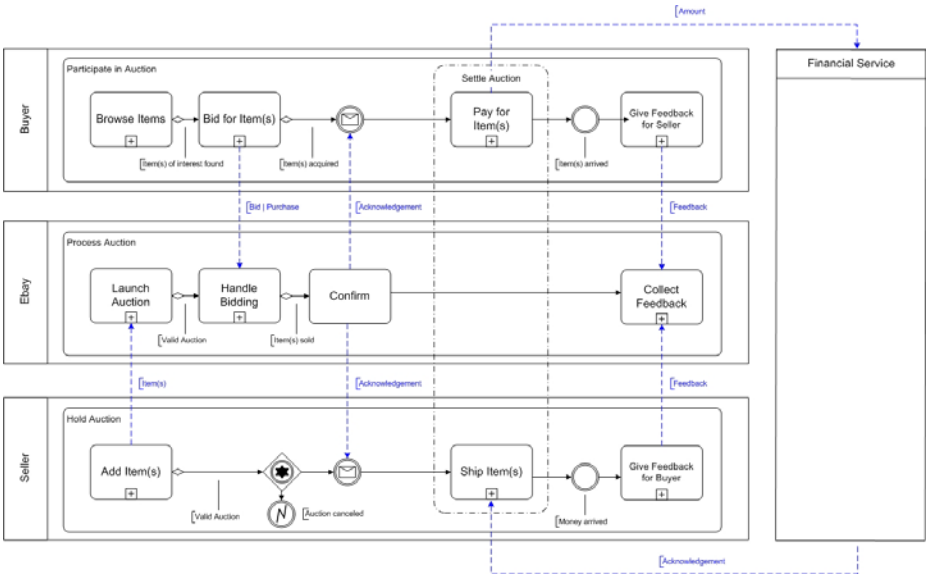


Fig. 9. Verfeinerung der Top-Level Prozessmodell der Ebay-Plattform

Die in Abbildung **8** beschriebenen Top-Level Prozesse können weiter verfeinert werden, was zu dem in Abbildung **9** gezeigten Prozessmodell führt.

Prozessvariabilitäten

In diesem Abschnitt soll der Teilprozess HANDLE BIDDING aus Abbildung **9** näher beleuchtet werden, da dieser ein Beispiel für die Realisierung von Prozessvariabilität darstellt.

In der Literatur werden mehrere Möglichkeiten angegeben, Variabilitäten zu modellieren bzw. zu implementieren.. Um die Varianten des Teilprozess HANDLE BIDDING zu modellieren, wird hier die Technik des *Module Replacement* verwendet:

> Module replacement is the technique of having multiple versions of a particular module and choosing the appropriate one. The interfaces to all versions of the module are compatible and, thus, the modules that depend on the variable module do not need to be modified based on the choice of variants. This choice can be done at build or execution time. (**6**)

In diesem Zitat wird die Kompatibilität der Schnittstellen einzelner Teilprozesse als Vorraussetzung zum Module Replacement genannt.

Der abstrakte Teilprozess kann nun durch jeden Prozess substituiert werden, der diese Schnittstelle erfüllt. Ebay selbst implementiert vier Variationen des Teilprozesses HANDLE BIDDING, die hier in aller Kürze aufgeführt werden – die zugehörigen Abbildungen finden sich in Anhang A, Abbildungen **11** bis **14**.

Chinesische Auktion
Die Chinesische Auktion ist wohl die am weitesten verbreitete Form der Auktion. Im Rahmen der Auktion wird genau ein Gegenstand angeboten, der nach Ablauf der Auktionszeit an den Meistbietenden versteigert wird.

Holländische Auktion
Die Holländische Auktion kann als Verallgemeinerung der Chinesischen Auktion verstanden werden. Im Rahmen der Auktion werden zwei oder mehr Gegenstände gleicher Art und Güte angeboten, die nach Ablauf der Auktionszeit an den bzw. die Meistbietenden versteigert werden.

Fixed-Price Auktion
Die Fixed-Price Auktion ist keine Auktion im klassischen Sinne. Im Rahmen der Auktion wird ein Gegenstand bzw. werden zwei oder mehr Gegenstände gleicher Art und Güte angeboten; jeder Gegenstand wird bei Abgabe eines Angebots verkauft – eine Versteigerung findet nicht statt. Die Auktion endet, wenn der letzte Gegenstand verkauft wurde bzw. die Auktionszeit abgelaufen ist.

Buy-It-Now Auktion
Die Buy-It-Now Auktion besitzt eine Sonderstellung unter den Auktionen, da sie noch zur Laufzeit ihren Typ ändern kann. Im Rahmen der Auktion wird genau ein Gegenstand angeboten; der erste Interessent hat die Möglichkeit, den Gegenstand zu einem fixen Preis zu erwerben oder ein Gebot abzugeben und somit eine Chinesische Auktion zu beginnen. Die Auktion endet, wenn der Gegenstand verkauft wurde bzw. die Auktionszeit abgelaufen ist.

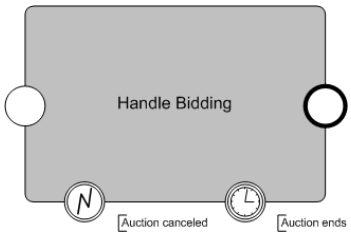


Fig. 10. Schnittstelle von HANDLE BIDDING

BPMN verfügt über geeignete grafische Mittel, die Schnittstelle eines Prozesses zu spezifizieren – dies wird in Abbildung **10** dargestellt. Die Schnittstelle des Teilprozesses HANDLE BIDDING zeichnet sich aus, durch die dargestellten Start- und Endpunkte sowie die vorhanden Austrittspunkte Auction Canceled und Auction Ends.

Dennoch scheinen mir BPMN Modelle an dieser Stelle nicht vollständig zu sein: Die zentrale Frage lautet, wie kann die Kompatibilität von Schnittstellen zweifelsfrei festgestellt werden? Ist es hinreichend, dass ein Prozess über die geforderten Start- und Endpunkte verfügt? An dieser Stelle kommt nur ein klares ’Nein’ als Antwort in Frage.

Über die vorhandenen Start- und Endpunkte hinaus werden noch weitere Bedingungen an eine Schnittstelle gestellt, wie z.B. bzgl. Pre- und Postconditions, Invarianten und Seiteneffekte.

BPMN verfügt nicht über inhärente Mittel, diese Bedingungen zu formulieren – statt dessen muss auf andere Techniken, wie z.B. die Object Constraint Language (OCL) [10] ausgewichen werden. Ob und wie weit dies tatsächlich einen Nachteil darstellt, ist im praktischen Gebrauch noch zu evaluieren.

Zusammenfassung

In dieser Ausarbeitung habe ich meine Gedanken zur Modellierung statischer und dynamischer Aspekte eines Systems im Kontext der Produktlinienentwicklung zusammengefasst.

Besonders hat mich interessiert, Kriterien zu finden und anzuwenden, um *geeignete* Beschreibungen und Modelle einer Applikation – in diesem Fall das Online-Auktionshaus Ebay – zu erstellen.

Anhang A

Chinesische Auktion

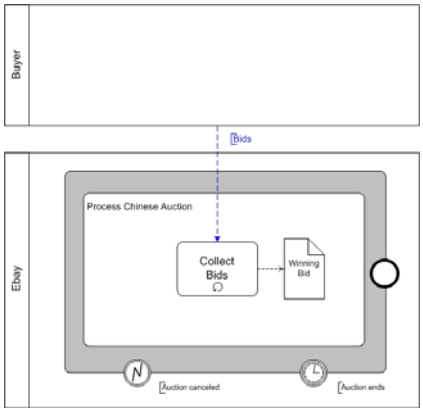


Fig. 11. Chinesische Auktion

Holländische Auktion

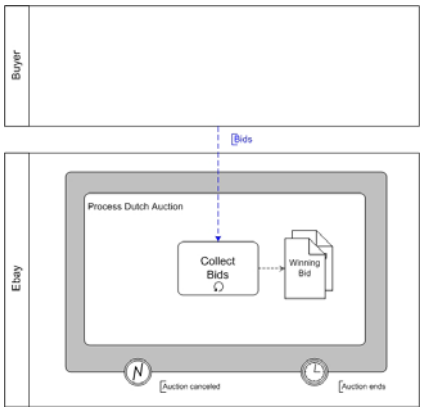


Fig. 12. Holländische Auktion

Fixed-Price Auktion

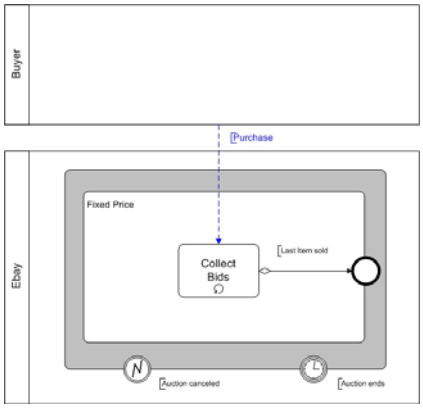


Fig. 13. Fixed Price Auktion

Buy-It-Now Auktion

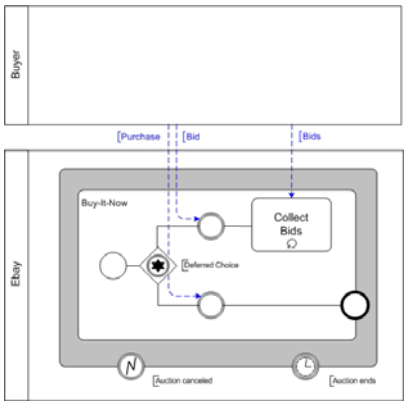


Fig. 14. Buy-It-Now Auktion

Zitate

(1) Andreas Bungert, *Beschreibung programmierter Systeme mittels Hierarchien intuitive verständlicher Modelle*, Shaker Verlag, 1998
(2) IEEE Computer Society, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, IEEE Std 1471-2000

(3) Siegfried Wendt, *Nichtphysikalische Grundlagen der Informationstechnik*, Springer Verlag, 1989

(4) Helmut Balzert, *Lehrbuch der Softwaretechnik*, Spektrum Akademischer Verlag, 2001

(5) Object Management Group, *Unified Modeling Language Specification*, OMG, 2003

(6) Felix Bachmann, Len Bass, *Managing Variability in Software Architectures*, Carnegie Mellon University

Literaturverzeichnis

[1] http://www.pesoa.org

[2] Mathias Weske, *PESOA: Process Family Engineering in Service Oriented Applications*, HPI, 2004

[3] Andreas Bungert, *Beschreibung programmierter Systeme mittels Hierarchien intuitive verständlicher Modelle*, Shaker Verlag, 1998

[4] Object Management Group, *Model Driven Architecture Guide*, OMG, 2003

[5] IEEE Computer Society, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, IEEE Std 1471-2000

[6] Siegfried Wendt, *Nichtphysikalische Grundlagen der Informationstechnik*, Springer Verlag, 1989

[7] Helmut Balzert, *Lehrbuch der Softwaretechnik*, Spektrum Akademischer Verlag, 2001

[8] Object Management Group, *Unified Modeling Language Specification*, OMG, 2003

[9] Business Process Management Initiative, *Business Process Modeling Notation*, BPMI, 2004

[10] Object Management Group, *Object Constraint Language Specification*, OMG, 2003

Modellierung von Hotel-Reservierungs-Systemen

Torsten Hahmann

Universität Potsdam
Hasso-Plattner-Institut für Softwaresystemtechnik
torsten.hahmann@hpi.uni-potsdam.de

Abstract. Allgemeine Geschäftsprozesse in Hotel-Reservierungs-Systemen werden vorgestellt und mit BPMN modelliert. Einzelne Modellierungsaspekte bei der Erstellung der BPMN-Prozessmodelle für ehotel.de offenbaren Stärken und Schwächen der Notation. Ein Ansatz zur Darstellung von Laufzeitvarianten in den Prozessphasen wird entwickelt.

1 Einleitung

Die rasante Entwicklung des Internets hat auch traditionelle Dienstleitungsbereiche erfasst. Kunden buchen Hotelübernachtungen im zunehmenden Maße über Hotel-Reservierungs-Systeme. Diese Systeme ermöglichen eine schnelle und unkomplizierte Durchführung von Reservierungen. Schon heute verlangen Unternehmen eine Integration solcher externen Applikationen in firmeneigene Portale. Egal ob zusätzliche Dienstleistungen für Kunden angeboten werden sollen oder es nur um die unternehmensinterne Vereinfachung von Prozessen gilt: Voraussetzung ist ein grundlegendes Verständnis der Prozesse. Will man aus Enterprise-Systemen (z.B. mySAP.com) auf Anwendungen wie Hotel-Reservierungs-Systems zugreifen, so müssen die Prozesse dieser externen Applikationen zunächst verstanden und dokumentiert werden. Diese Arbeit evaluiert, ob sich die Business Process Modeling Notation (BPMN) dazu eignet und welche Schwierigkeiten bei der Prozessmodellierung auftreten. Ein weiterer Schwerpunkt liegt auf der Betrachtung von Hotel-Reservierungs-Systemen als Produktfamilien. Im Vergleich zu BPM-Systemen[1] müssen zusätzlich Prozessvariabilitäten berücksichtigt werden [2]. Daraus ergibt sich die Notwendigkeit eines allgemeinen Modells pro Produktfamilie, welches möglichst unkompliziert konfigurierbar ist. Am Beispiel von ehotel wird die Modellierung von Varianten mit BPMN untersucht.

Dieser Einleitung folgt ein kurzer Überblick über Hotel-Reservierungs-Systeme, dabei werden Architektur und Funktionalität der Systeme von ehotel und HRS vorgestellt. Kapitel 3 erklärt Hotel-Reservierungs-Prozesse in allgemeiner Form und stellt ein anbieter-unabhängiges Top-Level-Prozessmodell dar. Einzelne Probleme bei der Modellierung mit BPMN werden diskutiert. Kapitel 4 stellt ein umfassendes Prozessmodell für ehotel vor. Die Modellierungsaspekte Laufzeitvarianten, Fehlerbehandlung, transaktionales Verhalten und Nutzernavigation werden ausführli-

[1] Business Process Management - Systeme

cher diskutiert. Letztendlich folgt eine Beurteilung, ob BPMN eine konsistente Modellierung der Reservierungsprozesse erlaubt und ob Varianten in geeigneter Form darstellbar sind.

2 Hotel-Reservierungs-Systeme

Grundsätzlich kann eine Hotel-Reservierung auf zwei verschiedene Arten zustande kommen. Einerseits kann der Kunde direkt die Hotels anrufen, andererseits ist die Beauftragung eines Vermittlers möglich – wenn man ins Reisebüro geht oder eine der vielen Buchungsangebote im Internet nutzt. Beiden Varianten liegt ein komplett unterschiedliches Systemverständnis zugrunde. Wird direkt bei einem Hotel gebucht, so sind genau zwei Akteure notwendig: der Kunde und das Hotel[1]. Wird dagegen ein Vermittler eingeschaltet, so ist der Vermittler Ansprechpartner für den Kunden und das Hotel, erst bei Ankunft tritt der Kunde direkt mit dem Hotel in Verbindung. Der höhere Aufwand durch das Einsetzen eines Vermittlers muss durch Vorteile dieses Konzeptes aufgewogen werden. Das heißt Kunde und Hotel müssen bereit sein, für die höheren Transaktionskosten aufzukommen. Eine solche Bereitschaft entsteht nur, wenn die Dienstleistung einen Mehrwert bietet. Für den Kunden bringt der Vermittler erhebliche Vorteile: er braucht Preise, Ausstattung, Lage und Auslastung der Hotels nicht selbst überprüfen. Der Vermittler sorgt für höhere Transparenz und eine nicht unerhebliche Zeitersparnis. Wenn immer mehr Kunden diesen Vorteil für sich entdecken, profitieren indirekt auch die Hotels: sie erhoffen sich mehr Übernachtungen durch die Bereitstellung ihrer Daten bei einem Vermittler. Oder pessimistischer ausgedrückt: sie hoffen keine Kunden zu verlieren. Diese Einsicht sicherte jahrelang die Existenz der Reisebüros; die in den letzten Jahren aufgekommenen Anbieter von Hotel-Reservierungs-Systemen haben dagegen den Anspruch, noch effizienter und noch kundenfreundlicher als Reisebüros zu arbeiten.

Im diesem Kapitel werden zwei der gängigen Systeme im deutschsprachigen Raum vorgestellt. Sowohl ehotel als auch HRS besitzen jahrelange Erfahrungen beim Aufbau und Betrieb eines Hotel-Reservierungs-Systems und haben ein umfassendes nationales wie internationales Angebot an Hotels. Systemaufbau sowie Funktionalität werden kurz vorgestellt, Unterschiede und Gemeinsamkeiten herausgearbeitet. Anhand dieser beiden Beispiele lässt sich ein Use-Case-Diagramm entwickeln, welches grundsätzliche Anforderungen an Hotel-Reservierungs-Systeme beschreibt. Dies ist die Grundlage für das darauf folgende Kapitel.

2.1 Systemaufbau von ehotel und HRS

Zur Darstellung des groben Systemverständnisses von ehotel und HRS wurde die FMC-Notation[2] gewählt [1]. Bei beiden Systemen (Abbildungen 11 und 12) hat der Kunde die Wahl, sich entweder direkt über einen Webbrowser oder über ein Call

[1] Genauer umfasst der Akteur Hotel alle möglichen Angestellten eines Hotels.
[2] FMC = Fundamental Modeling Concepts

Center mit dem System zu verbinden. Bei beiden Arten der Kommunikation mit dem System gibt es aber Einschränkungen bezüglich der verfügbaren Funktionalität.

ehotel hat zwei verschiedene Möglichkeiten der Verfügbarkeitsprüfung bzw. Buchung: die Anfrage kann mit den eigenen Daten abgeglichen werden und an ein Reservierungssystem von PEGASUS weitergeleitet werden.[1] In Systemen von PEGASUS verwalten die Hotels wiederum ihre Kontingente, ein Abgleich mit einem hausinternen Buchungssystem ist vorstellbar. HRS dagegen spart eine Stufe bei der Anfragebearbeitung aus: Hotels verwalten ihre Kontingente über einen Webbrowser oder Client im HRS-eigenen Reservierungs-System, alle relevanten Hotel- und Verfügbarkeitsdaten werden von HRS selbst gespeichert.

2.2 Use Cases von ehotel und HRS

Die Funktionalität eines Systems lässt sich gut mit Use-Case-Diagrammen nach der UML-Notation[2] darstellen [7]. Im Anhang befinden sich die beiden Use Case Diagramme für ehotel und HRS (Abbildung 13 und 14). Bei beiden gibt es vier identische Anwendungsfälle (Use Cases): (1) Information über Hotels und deren Verfügbarkeit, (2) Reservierung eines Hotels und (3) Stornierung einer Reservierung sowie die (4) Abwicklung. Die Abwicklung ist nicht Teil der von ehotel und HRS bereitgestellten Reservierungs-Systeme. Beide Anbieter treten nur als Vermittler auf, d.h. der Vertrag kommt immer zwischen Kunde und Hotel zustande. Die Abwicklung der Bezahlung obliegt somit diesen beiden Vertragspartnern, die Reservierungs-Systeme können dabei nur unterstützend wirken, indem sie z.B. Kreditkarteninformation erfassen.

Eine Umbuchung wurde als separater Vorgang bei den Use Cases von HRS mit berücksichtigt. Innerhalb des Systems werden Umbuchungen als Komposition aus Stornierung und neuer Buchung betrachtet. In UML 2.0 wird eine solche Einbeziehung anderer Anwendungsfälle durch „<<include>>“ markiert, die Ausführung der so referenzierten Use Cases ist nicht optional. Im Reservierungs-System von ehotel spielt die Umbuchung keine gesonderte Rolle. Wünscht der Kunde eine Änderung, so ist er aufgefordert, selber seine alte Reservierung zu stornieren und eine neue durchzuführen [4].

Welche Akteure bei Ausführung einzelner Anwendungsfälle involviert sind, bietet ein weiteres Unterscheidungsmerkmal zwischen beiden Systemen. Welche Akteure an welchen Aktionen beteiligt sind, ergibt sich schon aus dem jeweiligen Aufbau. Im Vergleich zu HRS ist bei ehotel das System von PEGASUS als zusätzlicher Akteur bei Information, Buchung und Stornierung notwendig. Eine Ausnahme bildet die direkte Stornierung des Kunden beim Hotel. Dieser Fall ist im System von HRS überhaupt nicht vorgesehen.

Beide Systeme akzeptieren Buchungen sowohl online (über ihr Portal) als auch telefonisch oder schriftlich (E-Mail oder Fax). Für Stornierungen bieten beide Anbieter jedoch unterschiedliche Kommunikationswege an. In den Use-Case-Diagrammen

---

[1] PEGASUS Solutions, Inc. bietet Reservierungssysteme (z.B. NetBooker™) an, die von ehotel zur Suche und Reservierung von Hotels verwendet werden. Den Hotels ist es möglich ihre eigenen Kontingente innerhalb des Reservierungssystems von PEGASUS selbst zu verwalten. Weitere Informationen dazu unter www.pegs.com.

[2] UML = Unified Modeling Language

stellt sich das jeweils als eine Vererbungs-Beziehung dar, wie sie auch zur Modellierung der Buchung angewandt wird. Aus einer solchen Darstellung sind Varianten für Buchung bzw. Stornierung problemlos identifizierbar. Denkbar wären auch verschiedene Varianten der Informationsbeschaffung: dies kann online, am Telefon oder auch per Fax bzw. E-Mail erfolgen – entsprechend den Buchungsarten. Da die Informationsphase für die weitere Arbeit jedoch zweitrangig ist, wird hier auf eine Vererbung verzichtet. Die Modellierung der Prozesse in Kapitel 4 wird darauf aufbauen.

2.3 Allgemeines Use-Case-Diagramm

Aus den Beispielen lässt sich ein allgemein gültiges Modell für die Use Cases ableiten (vgl. Abbildung 1). Die Use Cases Information, Buchung und Stornierung werden für jedes vernünftig konzipierte Hotel-Reservierungs-System benötigt. Welche Varianten dabei jeweils angeboten werden, bleibt offen. Weiterhin muss eine Zahlungsabwicklung auf irgendeine Art erfolgen, sie ist aber nicht notwendigerweise Bestandteil des eigentlichen Reservierungs-Systems. Zusätzliche Funktionen zielen darauf ab, dem Kunden einen gewissen Mehrwert zu bieten. Sie sind nicht unabdingbar für die Fähigkeit einen Reservierungsprozess durchzuführen. Daher tauchen solche Funktionen nicht im allgemeinen Use-Case-Diagramm auf.

Unbedingt notwendig für alle Aktivitäten ist der Kunde, des Weiteren sind Hotel bzw. Hotelangestellte bei der Abwicklung der Zahlung involviert. Weitere Akteure hängen von der jeweiligen Umsetzung des Reservierungsprozesses ab und können nicht allgemein dargestellt werden. Beispiele für weitere Akteure sind in den Use-Case-Diagrammen für ehotel und HRS zu finden (vgl. Abbildungen 13 und 14).

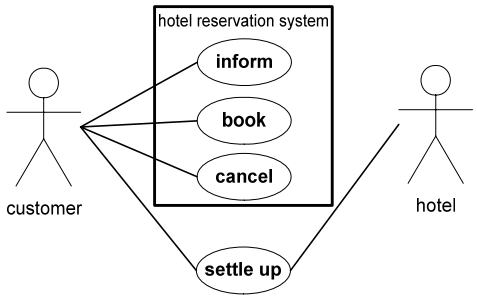


Abb. 1. Allgemeines Use-Case-Diagramm für Hotel-Reservierungs-Systeme

3 Prozesse in Hotel-Reservierungs-Systemen

Das folgende Kapitel stellt den allgemeinen Prozess einer Hotel-Reservierung aus Sicht des Kunden vor. Dabei wird noch kein spezielles Reservierungssystem betrachtet, stattdessen wird von einer intuitiven Vorstellung des Ablaufs ausgegangen. Die Modellierung des Prozesses erfolgt mit BPMN, angelehnt an die Spezifikation 1.0.

Die Buchungsphase wird ansatzweise verfeinert, um allgemeine Probleme bei der hierarchischen Dekomposition zu identifizieren. Im Mittelpunkt der Diskussion steht die Erhaltung einer konsistenten Sicht auf die Prozesse. Anhand zweier Beispiele werden alternative Modelle für identische Sachverhalte in Bezug auf ihre Aussagekraft und Verständlichkeit hin untersucht. Ziel ist eine möglichst selbsterklärende Modellierung der Prozesse.

Für den Kunden besteht auf abstraktester Ebene ein Prozess zur Hotelreservierung aus einer sequentiellen Folge von Aktivitäten. Die vier fundamentalen Use Cases bilden vier äquivalente Phasen zur Beschreibung des Hotel-Reservierungs-Prozesses. Einer optionalen *Informationsphase* folgt die *Buchungsphase*, anschließend kann während der *Stornierungsphase* die Reservierung zurückgenommen werden. Geschieht dies nicht, so schließt die *Abwicklungsphase* den Prozess ab. Die Aktivitäten der Informationsphase können beliebig oft wiederholt werden bis der Prozess mit einer Buchung fortgesetzt wird. Wichtig ist, dass der Prozess während jeder Phase abgebrochen werden kann. Wenn der Kunde nichts Passendes in der Informationsphase findet, so wird er nicht fortsetzen und den Prozess dadurch implizit beenden. Treten Fehler bei der Buchung auf, so kann er sich auch entscheiden, den Prozess abzubrechen. Eine Stornierung bricht den Prozess generell ab und während der Abwicklungsphase wird der Prozess aus Sicht des Kunden entweder durch Bezahlung des Hotels oder durch Nichterscheinen beendet. In der Modellierung des Prozesses müssen weiterhin Stornierungsfristen beachtet werden.

3.1 Modellierung des Top-Level-Prozesses Hotelreservierung mit BPMN

Zur Modellierung der Prozesse wird in diesem und nächsten Kapiteln ausschließlich die Business Process Modeling Notation verwendet. Zum Verständnis der Diskussion ist eine vorherige Einarbeitung in die Notation unerlässlich. Da dies im Rahmen des Seminars zur Genüge getan wurde, verweise ich an dieser Stelle auf die „Introduction to BPMN“ [13] sowie auf das Kapitel 4 des PESOA-Report No. 01/2004 [10] als einführende Literatur.

Das Reservierungs-System ist in Abbildung 15 nur als Blackbox dargestellt. Auf die Prozesse innerhalb des Systems wird nicht näher eingegangen, es dient nur der Darstellung des potentiellen eingehenden sowie ausgehenden Nachrichtenflusses. Die Kommunikation der einzelnen Phasen mit dem Reservierungs-System kann so allgemein modelliert werden. Die folgenden Ausführungen beziehen sich – soweit nicht anders vermerkt – auf das Modell in Abbildung 15.

Informationsphase. Die Informationsphase vor der eigentlichen Buchung ist optional und kann beliebig oft wiederholt werden. In BPMN bietet sich dafür eine Markierung als ad-hoc-Aktivität an, welche ausdrücklich auch eine Nichtausführung ermöglicht. Für den ungeübten Betrachter ist dies aus dem Modell jedoch nicht erkenntlich; eine derartige Modellierung verstößt gegen den Wunsch nach einer selbsterklärenden Notation. Erst in den Attributen kann die Ausführungshäufigkeit genauer spezifiziert werden. Wenn der Kunde schon während der Informationsphase das Interesse verliert, so ist es ihm möglich, schon hier den Prozess zu beenden. Im Modell ist dies durch

einen bedingten Sequenzfluss dargestellt – äquivalent wäre die Verwendung eines XOR-Split, mit einem Sequenzfluss zum Endzustand.

Buchungsphase. Hat der Kunde ein Hotel ausgewählt, so kann er zur Buchung übergehen. Dies bedeutet unter Umständen, dass eine Vielzahl von Prozessinstanzen schon während der Informationsphase beendet werden. Es ist durchaus üblich, dass ein potentieller Kunde erst mehrere Prozesse startet, davon aber im Regelfall nur eine Prozessinstanz bis zur Buchungsaktivität voran schreitet. Im Modell gibt es damit keine Probleme, die anderen Instanzen werden regulär beendet.

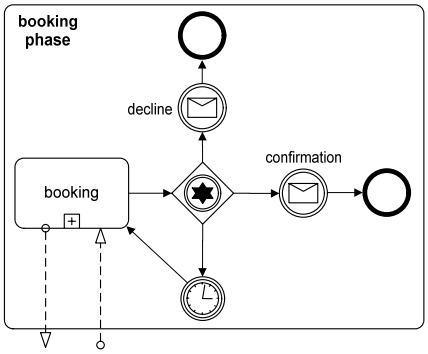


Abb. 2. Variante 1 des verfeinerter Subprozesses Buchungsphase: Wiederholung der Buchungsaktivität bei Time-Out

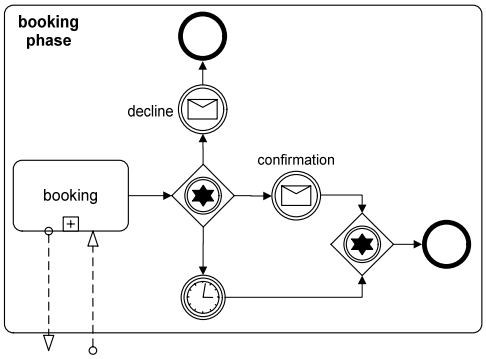


Abb. 3. Variante 2 des verfeinerter Subprozesses Buchungsphase: Forschreiten im Prozess bei Bestätigungsnachricht oder Time-Out; eine anschließende Stornierung sollte sichergestellt werden

In der Verfeinerung der Buchung sind drei wesentliche Stufen erkennbar (vgl. Abb. 16). Zuerst muss eine Übermittlung der buchungsrelevanten Daten (Hotelinformationen sowie Kundendaten) erfolgen, anschließend kann der Kunde den Buchungswunsch absenden. Letztendlich muss er auf eine Rückmeldung des Hotels, Reisebüros oder Hotelbuchungssystems warten. BPMN stellt dafür ein Event-basiertes Gateway zur Verfügung. Eine Prozessinstanz kann erst nach dem Auftreten des erwarteten Events fortgeführt werden.[1] Hier wird auf das Eintreffen einer Nachricht gewartet, diese Nachricht kann entweder eine Buchungsbestätigung oder eine Absage sein, weil das Hotel zum gewünschten Zeitpunkt ausgebucht ist. Problematisch dabei ist, dass die Prozessinstanz ohne eintreffende Nachricht zu keinem regulären Abschluss kommt. Wenn die Kommunikation scheitert, kann eine ordnungsgemäße Termination nicht gewährleistet werden, dazu bedarf es gesonderter Mechanismen. Eine Variante wäre die Einführung eines nach einer definierten Zeit getriggerten Events, welches nach Ablauf einer Frist die kundenseitige Buchungsan-

[1] In BPMN werden Events verwendet, um das Eintreten oder Auslösen bestimmter Ereignisse zu modellieren. Im Folgenden wird keine Unterscheidung zwischen den tatsächlichen Ereignissen und dem Schalten der entsprechenden Stellen gemacht. In Anlehnung an den englischen Begriff der Spezifikation wird in dieser Arbeit beides als Event bezeichnet.

frage wiederholt (vgl. Abbildung 2) oder die Instanz des Prozesses automatisch beendet (vgl. Abbildung 3). Wenn die Nachricht schon unterwegs war (z.B. E-Mail Empfang verzögert), muss das Hotel bzw. das Buchungssystem sicherstellen, dass nur eine von zwei oder mehr inhaltsgleichen Buchungen tatsächlich getätigt wird. Der Wunsch nach einer sauberen Lösung führt uns im Kapitel 4 zur Möglichkeit der Modellierung von transaktionalem Verhalten. Vorerst soll uns der Subprozess mit mehreren Endzuständen als Lösung genügen (vgl. Abbildung 16 im Anhang).

Stornierungsphase. Die Ausführung der folgenden Stornierung ist optional. Der Kunde hat bis zum Ablauf des Timers Zeit, eine Stornierung zu veranlassen. Der Timer ist dabei von den jeweiligen Stornierungsrichtlinien abhängig. Wird die Stornierung innerhalb der gegebenen Frist durchgeführt, so terminiert die Prozessinstanz, ansonsten wird die Abwicklungsphase aktiviert.

Abwicklungsphase. Ist die Frist für eine Stornierung abgelaufen, so wird erwartet, dass der Kunde das Hotel vor Ort bezahlt. Im Modell muss auch berücksichtigt werden, dass der Kunde eventuell nicht erscheint, diese Option realisiert ein Timer, welcher nach Ablauf des Abreisedatums[1] den Prozess auf Kundenseite beendet. Dem Kunden werden eventuelle Gebühren trotzdem berechnet – z.B. per Kreditkartenabbuchung oder Lastschrift. Bezahlt der Kunde ordnungsgemäß, so kommt die Instanz des Prozesses ebenfalls zu einem Ende. Um einen gewissen Grad der Übersichtlichkeit zu erhalten, wird die Möglichkeit eines Inkassoverfahrens im Prozess nicht berücksichtigt. Dies gilt auch für alle späteren Modellierungen.

3.2 Probleme bei der Modellierung des Top-Level-Prozesses

Nachrichtenfluss und Message-Events. Schon die Modellierung dieses überschaubaren Prozesses (Abb. 15) offenbart einige Schwächen von BPMN. Insbesondere Verfeinerungen sind problematisch. Sowohl potentieller ein- als auch ausgehender Nachrichtenfluss kann gegebenenfalls bei der Verfeinerung nicht mehr klar identifiziert werden. Da das Warten oder Senden einer Nachricht auch durch Message-Events dargestellt werden kann, tritt bei ausschließlicher Verwendung der Events in der Verfeinerung ein expliziter Nachrichtenfluss in Form einer gestrichelten Linie nicht mehr auf. Zwar erlaubt die Spezifikation von BPMN eine Antragung von ausgehenden Nachrichtenflüssen an End-Events, sie erfordert es aber nicht ausdrücklich [3]. In Abbildung 4 ist es in der Aktivität möglich, Nachrichten zu senden und zu empfangen – dargestellt durch ein- und ausgehenden Nachrichtenfluss. In der Verfeinerung gibt es diesen Nachrichtenfluss in Form einer gestrichelten Linie nicht mehr. Durch die Verwendung von Message-Events entspricht das Verhalten jedoch der eingeklappten Aktivität. Obwohl dies eine konsistente Verfeinerung ist, erscheint sie auf den ersten Blick nicht als solche und erschwert Konsistenzprüfungen – egal ob automatisch oder manuell durchgeführt. Generelles Problem hierbei ist das Nebenein-

[1] Es muss im Prozess bis zum Abreisedatum gewartet werden, falls der Kunde nur den letzten Tag der Reservierung in Anspruch nimmt. Dann kann er trotz verspäteter Ankunft vor Ort den gesamten Rechnungsbetrag begleichen.

ander von Nachrichtenfluss und Message-Events. Abhilfe könnte die generelle Antragung von Nachrichtenfluss an die Message-Events schaffen, allerdings leidet dadurch die Übersichtlichkeit. Der generelle Verzicht auf eine der beiden Möglichkeiten verringert die Mächtigkeit der Sprache enorm, in späteren Modellen wird klar, dass eine Koexistenz beider Formen sinnvoll ist. Ein direktes Mapping auf BPEL4WS[1] ist jedoch nur bei Events möglich, Nachrichtenflüsse in allgemeiner Form können nicht abgebildet werden. Entscheidend ist der verantwortungsvolle Umgang mit beiden Möglichkeiten. Deshalb kann für komplexere Modelle eine längere Einarbeitung in die Notation von BPMN kaum vermieden werden.

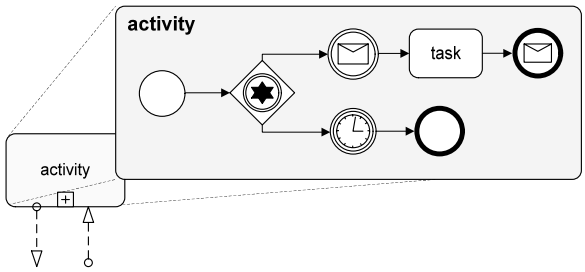


Abb. 4. beispielhafte Verfeinerung einer Aktivität mit potentiellen Nachrichten durch die ausschließliche Verwendung von Message-Events

Bedingte Sequenzflüsse. Ein weiteres Problem bei der hierarchischen Dekomposition stellen bedingte Sequenzflüsse dar. Die Spezifikation von BPMN erlaubt, dass eine Aktivität mehrere ausgehende Sequenzflüsse haben darf – eine Art der Umsetzung des Multiple Choice Pattern [3], [14]. Es wird eine Kennzeichnung als „default“ bzw. „conditional“ empfohlen. Die Abbildung der Kanten auf Endzustände im verfeinerten Ablauf ist in der graphischen Notation von BPMN nicht möglich. Abhilfe schaffen hier Attribute und deren Darstellung als Kommentare wie in Abbildung 5 verdeutlicht. Allerdings beeinträchtigen diese die Übersichtlichkeit der Prozessmodelle. Die Spezifikation von BPMN schlägt deshalb eine graphische Notation vor, in der Start- und Endevents eines expandierten Subprozesses auf dessen Rand gelegt werden, um die Nachvollziehbarkeit des Flusses über Aktivitätsgrenzen hinaus zu gewährleisten (vgl. Abbildung 6) [3]. Trotzdem müssen für eine spätere Abbildung auf BPEL4WS entsprechende Attribute hinzugefügt werden. Diese Art der Darstellung wird in den folgenden Modellen verwendet.

Implizierte Start- und Endevents. Abbildung 5 ist ein Beispiel für die Verwendung impliziter Start- und Endevents. Würde man Start und Ende der Aktivität Buchungsphase jeweils explizit darstellen, so wäre klar ersichtlich, wo der Prozess gestartet bzw. beendet wird. Zusammen mit der schon erwähnten Möglichkeit Start- und Endevents eines Subprozesses auf den Rand zu legen, kann der Leser den Ablauf deutlich einfacher nachvollziehen. Die erhöhte Quantität an Symbolen wirkt sich dagegen negativ aus. Hier ist eine genaue Abwägung der Vor- und Nachteile im Einzelfall erforderlich. Für ein komplexes Modell sollte man sich jedoch auf eine

[1] Business Process Excecution Language for Web Services, vgl. dazu auch [11]

einheitliche Handhabung festlegen. Diese Arbeit stellte Start- und Endevents immer dar.

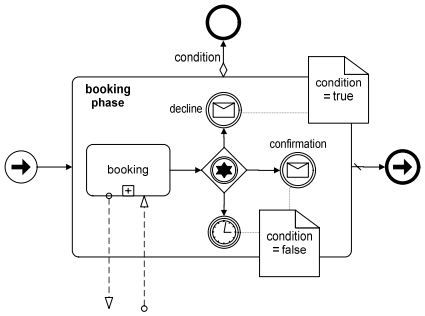


Abb. 5. Subprozess Buchungsphase ohne Start- und Endevents

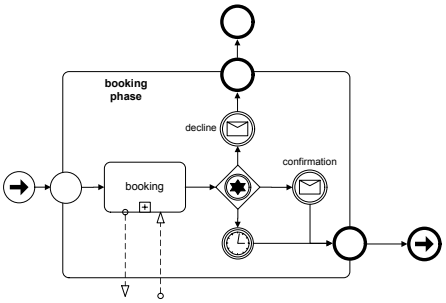


Abb. 6. Subprozess Buchungsphase mit Start- und Endevents auf dem Rand

4 Prozesse ehotel.de

Im letzten Kapitel wurden die Top-Level-Prozesse beim Kunden in allgemeiner Form betrachtet. Am Beispiel der Plattform ehotel.de wird dieser Prozess auf die Akteure Reservierungssystem und Hotel erweitert. Die vier Phasen des allgemeinen Hotel-Reservierungs-Prozesses werden bei allen drei beteiligten Akteuren verfeinert. Einzelne Aspekte der Buchungsphase wie transaktionales Verhalten, die Modellierung von Fehlern sowie von Laufzeitvarianten werden näher beleuchtet.

4.1 Informationsphase

Die Dateneingabe des Kunden zur Information über Hotels und deren Preise zu einem gewünschten Zeitraum sind ausschließlich Navigationsaspekte. Es wird kein Prozess angestoßen, bis der Kunde korrekte und vollständige Daten in die entsprechende Suchmaske eingegeben hat. Wenn schließlich korrekte Daten an die ehotel-Plattform übermittelt werden, kann diese die Anfrage an ein oder mehrere Hotels weiter leiten.[1] Die erhaltenen Daten gibt ehotel wiederum an den Webserver weiter, der die graphische Aufbereitung für den Kunden realisiert. Hier stellt sich die Frage, ob es generell sinnvoll ist, die Informationsphase in den Prozess mit aufzunehmen. Es ist immer möglich, sich über aktuelle freie Kapazitäten in den Hotels zu informieren, diese Suche nach dem passenden Hotel ist aber eher als Navigation aufzufassen. Der Grund für die Entscheidung zugunsten einer Modellierung als Teil des Reservierungsprozesses liegt allein in der Tatsache, dass eine Reservierung online nur durch die Auswahl

[1] In den Diagrammen sind als beteiligte Akteure immer die Hotels dargestellt, das System von PEGASUS leitet Anfragen nur weiter bzw. speichert Informationen über die Hotels. Für die Modellierung ist dieses zwischengeschaltete System irrelevant.

eines Hotels – welches vorher im Informationsprozess ausgesucht wurde – angestoßen werden kann. Die Hoteldaten und Preise sind also essentieller Bestandteil der Buchung. Solche Daten sind jedoch zeitabhängig (z.B. Anwendung von Sonderkonditionen) und somit kann die Informationsphase nicht getrennt vom Rest des Prozesses betrachtet werden.

Wie der Kunde sich informiert, ist allein ihm überlassen. Mögliche Optionen sind die Information über das Onlineportal oder eine telefonische oder schriftliche Anfrage. Ziel ist es, ein geeignetes Hotel auszuwählen, welches (zumindest zum Informationszeitpunkt) für den gewünschten Zeitraum noch freie Zimmer hat. Anhand welcher Daten (Preis, Lage, Hotelkette, etc.) letztendlich die Auswahl getroffen wird, ist für die Prozessmodellierung uninteressant. Der folgende Absatz beschränkt sich auf eine Detailbetrachtung des Informationsprozesses über das Onlineportal. Vergleichbare, aber weit weniger formalisierte Abläufe sind bei telefonischer oder schriftlicher Information denkbar. Varianten der Informationsphase könnten in ähnlicher Weise wie die Varianten der Buchungsphase (4.2) modelliert werden.

Im Diagramm für die Verfeinerung der Aktivität „present hotel information“ wird die Entscheidung über die Art der gewünschten Daten anhand des Kunden-Aufrufs gefällt. Somit wird die Art der Anfrage nicht schon im Event-basierten Gateway getroffen. Dies hat den Vorteil, dass auf der höheren Modellierungsebene genau ein Event für die Informationsabfrage steht. Unabhängig von der gewünschten Information wird die Aktivität „present hotel information“ aufgerufen, welche dann erst z.B. anhand der mitgelieferten Daten entscheidet, ob es sich um eine allgemeine Suchanfrage handelt oder ob Detailinformationen zu einem Hotel gewünscht werden. In diesem Diagramm erscheint die Verwendung von Datenfluss sinnvoll, um den weiteren Ablauf zu modellieren. Der Aufruf von PEGASUS muss vor unerwarteten Fehler geschützt werden. Kommt etwa keine Antwort, so muss ein Timer nach einer festgelegten Zeit eine Exception verursachen, welche wiederum nach wohldefinierten Regeln eine neue Anfrage startet bzw. den Subprozess beendet (indem leere Daten zurückgegeben werden).

4.2 Buchungsphase

Kern des Reservierungsprozesses ist die eigentliche Buchung. Die Modellierung dieses Teilprozesses (vgl. Abb. 7 und Abb. 8) wird deshalb im Folgenden sehr ausführlich diskutiert.

Der Buchungsprozess auf Seite des Kunden ist bei ehotel stark abhängig vom gewählten Kommunikationsmedium. Der Ablauf einer telefonischen Buchung ist nicht formal geregelt, sie erfolgt nach Angabe der persönlichen Daten und Kreditkarteninformationen durch eine mündliche Bestätigung des Kunden. Schriftliche Reservierungen (E-Mail, Fax) bestehen aus Kundensicht aus einer Folge von verschickten und empfangenen Nachrichten (bis alle notwendigen Angaben gemacht wurden), auf Seite von ehotel ist – vergleichbar mit der telefonischen Buchung – nur der endgültige Buchungswunsch relevant. Dagegen ist die Onlinebuchung für den Kunden sehr stark formalisiert (Abb. 20 im Anhang), er wird durch den Prozess navigiert bis alle erforderlichen Daten gesammelt wurden. Für alle drei Fälle sind unterschiedliche Abläufe auf Kundenseite zu modellieren. Wünschenswert wäre es

aber, den Prozess aus Sicht von ehotel unabhängig vom Kommunikationsmedium zu gestalten. Der folgende Abschnitt soll klären, ob dies möglich ist.

Laufzeitvarianten. Aus dem allgemeinen Prozessmodell für Reservierungs-Systeme geht nicht hervor, welche Buchungsarten zur Verfügung stehen. Dies ist von Anbieter zu Anbieter unterschiedlich. Da aber auch einzelne Systemanbieter daran interessiert sind, ihren Prozess flexibel zu gestalten, ist eine Modellierung der Buchungsarten als Varianten wünschenswert. Auf der Seite des Kunden sind im Prozess von ehotel drei verschiedene – sich gegenseitig ausschließende – Buchungsarten möglich (Abb. 17). Ihnen ist erstens gemeinsam, dass während des Buchungsvorgangs aufgetretene Fehler (z.B. keine Antwort oder Hotel mittlerweile ausgebucht, etc.) zum Anfang der Buchungsphase zurückführen. Zweitens kann vom Kunden der Prozess jederzeit terminiert werden, bei der Onlinebuchung etwa durch eine Nichtweiterführung der Dateneingabe oder bei telefonischer Buchung durch unerwartetes Auflegen während des Gespräches. Dies wurde im Prozess als Terminate-Event modelliert. Da im Prozess keine multiplen Instanzen vorkommen, entspricht es hier der Semantik eines normalen Endes. Um aber hervorzuheben, dass es sich nicht um ein normales Beenden des Prozesses handelt, ist der Einsatz anstelle des einfachen Endevents gerechtfertigt.

Innerhalb der verschiedenen Buchungsprozesse müssen drei spezielle Tasks unbedingt ausgeführt werden, das weitere Verhalten kann beliebig variieren. Zu den drei Tasks gehören (in dieser Reihenfolge): (1.) Signalisieren, dass der Buchungsprozess gestartet werden soll, (2.) die Übermittlung der Kreditkarteninformationen und (3.) das Bestätigen des Buchungswunsches. Dies ergibt sich aus der Modellierung auf Seite von ehotel. Anhand des einkommenden Events (meistens eine Nachricht) kann entschieden werden, welcher Subprozess gestartet wird, irgendwann werden die Kreditkartendaten zur Validierung benötigt und „try booking“ muss wiederum vom Kunden angestoßen werden.

Auf Kundenseite ist die Modellierung der Buchungsvarianten durch ein einfaches XOR-Gateway gelöst. Da eine Variante erst zur Laufzeit ausgewählt wird, muss der Prozess alle Optionen offen halten. Sollen weitere Varianten angeboten werden, so kann das XOR-Gateway beliebig erweitert werden. Der Entscheidungspunkt ist somit gleichzeitig Variationspunkt. Neu anzuknüpfende Buchungsarten müssen auf jeden Fall die drei zuvor diskutierten Tasks in der richtigen Reihenfolge implementieren.

Zur Anknüpfung des Nachrichtenflusses wurde eine Gruppierung gewählt - entgegen der BPMN-Spezifikation. Gruppierungen haben eigentlich informalen Charakter, d.h. sie werden später nicht auf BPEL4WS abgebildet. Somit kann der davon ausgehende Nachrichtenfluss bei der Abbildung nicht interpretiert werden. Stellt man das Verhalten dieser Gruppierung allerdings dem einer Aktivität gleich, so wäre die Modellierung legitim. Die Gruppierung dient nur dem Zweck, die verschiedenen potentiellen Nachrichtenflüsse zu entwirren, um zur Laufzeit genau eine der Varianten mit dem ehotel-System kommunizieren zu lassen. Eine Zusammenfassung zu einem Subprozess entspräche nicht dem Ziel der Modellierung. Die leichte Erkennbarkeit von Variationspunkten durch den Einsatz von Gruppierungen ist ein weiterer Vorteil.

In der Modellierung des ehotel-Systems ist die Umsetzung der Buchungsvarianten mit größeren Problemen verbunden. So gibt es zwar eine Aktivität („booking“) in der

die Bearbeitung aller Varianten der Buchung gekapselt ist, zum Start der Aktivität muss jedoch schon entschieden werden, welche Variante gewählt wurde. Zur Modellierung dieser deferred choice ist die Verwendung eines Event-basierten Gateways Pflicht [14]. Problematisch stellt sich die Verknüpfung mit Suchanfragen dar. Da es keinen Zeitpunkt gibt, zu dem entschieden werden kann, ab wann nur noch Buchungsanfragen möglich sind, muss bis zum Start eines Buchungsvorgangs die Option einer Suchanfrage offen gehalten werden. Das Event-basierte Gateway trifft folglich zwei Entscheidungen: (1.) wird der Informationsprozess fortgesetzt oder der Buchungsprozess gestartet und – bei Eintreten des letzteren Ereignisses – (2.) welches Start-Event des Buchungsprozess wird ausgewählt. Das Gateway kann nicht in die Buchungsaktivität verlagert werden, es muss aber mit allen möglichen Startpunkten (variiert je nach Anzahl der Varianten) dieser Buchungsaktivität verknüpft werden. Dies ist bei Hinzufügen neuer Varianten problematisch, da ein alleiniges Erweitern der Buchungsaktivität nicht ausreicht. Vorteilhaft hingegen ist bei dieser Art der Varianten, dass innerhalb der Buchungsaktivität nur ein neues Startevent mit einer zugehörigen Aktivität eingefügt werden kann (zur Bearbeitung von Vorbedingungen, etc.), die Überprüfung der Kreditkartendaten sowie die Buchungstransaktion bleiben unverändert. Da die Auswahl einer Variante immer erst zur Laufzeit erfolgt, müssen auch hier alle Varianten mit einem XOR-Gateway verknüpft werden. Wird eine Variante durch das entsprechende Event gestartet, so werden alle anderen Varianten deaktiviert. Eine Modellierung komplett unabhängig von der Wahl des Kommunikationsmediums ist nicht möglich. Einschränkungen wie bestimmte Geschäftszeiten des Call-Centers bei telefonischer Buchung müssen vor der eigentlichen Bearbeitung berücksichtigt werden. Die drei verfeinerten „preactivities“ sind im Anhang zu finden (Abbildungen 21 bis 23). Auf Kundenseite wurde exemplarisch die Onlinebuchung verfeinert (Abbildung 20).

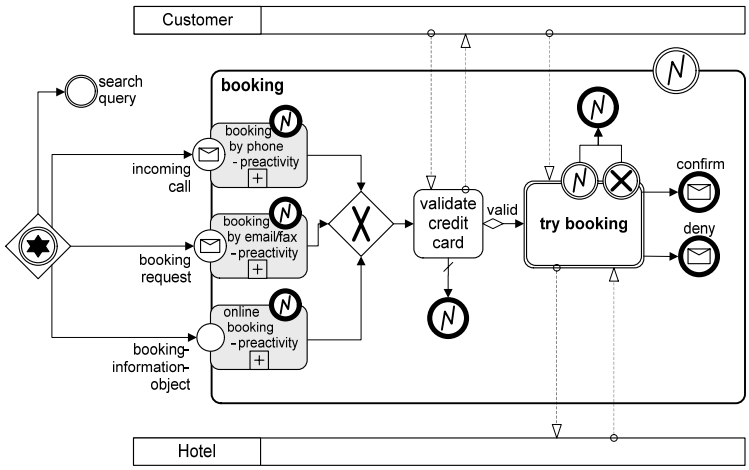


Abb. 7. Buchung über verschiedene Kommunikationsmedien im ehotel-System

Alternativ ist eine Modellierung von Prozessvariabilitäten wie in Abbildung 8 denkbar. Die verschiedenen Events zum Start der Buchungs-Aktivität werden in einem

Multiple-Event zusammengefasst. Erst innerhalb der Buchungs-Aktivität wird anhand der Daten (welche etwa in Form einer Nachricht bei Signalisierung des Events übergeben werden) entschieden, welche „preactivity“ ausgeführt wird. Für die Erweiterung durch neue Buchungsvarianten reicht eine entsprechende Ergänzung des Multiple Start-Events in Verbindung mit neuen Verzweigungen des XOR-Gateways aus. Diese Modellierung hat den Vorteil, dass außerhalb der Buchungs-Aktivität keine Änderungen im Prozessmodell des ehotel-Systems notwendig sind.

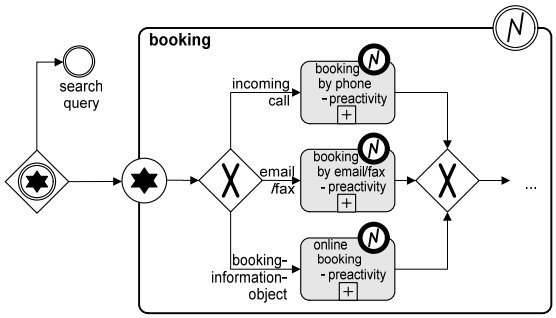


Abb. 8. alternative Modellierung der Buchungs-Aktivität (Ausschnitt)

Fehlerbehandlung. Interessant an dieser Stelle ist das Prinzip des „throw“ und „catch“ bei BPMN. Das End-Event Error erzeugt immer einen Fehler, es ist keine andere Semantik in der Spezifikation vorgesehen. Ein Intermediate Error hat im normalen Sequenzfluss die gleiche Bedeutung, außer dass der Fluss trotzdem weiter geführt wird. Wenn jedoch am Rand eines Subprozesses ein Intermediate Error angetragen ist, so fängt dieses Event eventuelle Fehler ab. In Abbildung 9 werden alle Fehler, die innerhalb des großen Subprozesses auftreten, abgefangen und nach außen weitergegeben. Eine sehr mächtige Modellierung, welche der Semantik objektorientierter Programmiersprachen (z.B. Java) äquivalent ist: ein End-Event Error entspricht einem `throw Exception();` und ein Intermediate-Event Error am Rande einer Aktivität entspricht einem `try { … } catch () { … };` wobei sich die Aktivität im try-Block und der vom Intermediate Error ausgehende Sequenzfluss im catch-Block wieder finden würden. Dieses Prinzip findet Anwendung in der Buchungs-Aktivität: alle innerhalb auftretenden Fehler beenden die Ausführung der Aktivität, auf höherer Ebene führt die Signalisierung des Error Events zum Event-basierten Gateway am Anfang der Buchungsphase zurück (vgl. Abb. 7 und Abb. 17). Ähnliches Verhalten ist während der Stornierungsphase erkennbar, vergleiche dazu Kapitel 4.3. Auf Kundenebene werden alle im Subprozess „cancelation at ehotel“ auftretenden Fehler durch das Intermediate-Event Error abgefangen, in der Verfeinerung der Aktivität kann ein Timeout beim Warten auf eine Rückmeldung solch einen Fehler erzeugen. Die Konsistenz des abstrakteren Diagramms (Abb. 17) und der Verfeinerung (Abb. 24) sind dadurch gewährleistet.

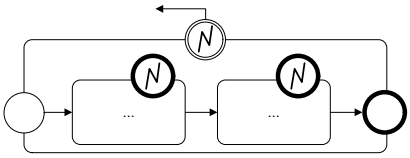


Abb. 9. Erzeugen und Abfangen von Fehlern in BPMN

Transaktionen. Buchungsvorgänge im ehotel-System und beim Hotel müssen transaktionalen Charakter aufweisen, um Inkonsistenzen zu vermeiden. Dafür eignet sich in BPMN die Modellierung als doppelt umrandete Transaktion wie in Abbildung 10 demonstriert. Die Verwendung der Transaktion ist vorteilhaft, wenn bei einer unterbrochenen Kommunikation sichergestellt werden muss, dass die angefragte Buchung wieder gelöscht wird. Wenn unsere Transaktion also vergeblich auf eine Antwort des Hotels wartet, so „canceled" der Timer die gesamte Transaktion. Dabei werden alle Kompensations-Tasks ausgeführt, also die angefragte Buchung wieder storniert. Wenn dem Kunden mitgeteilt wird, dass seine Buchung nicht durchgeführt werden konnte, ist garantiert auch keine Buchung im System registriert.[1] Auf Seite des Hotels stellt die Transaktion „book hotel" sicher, dass sie komplett oder überhaupt nicht ausgeführt wird. Erst wenn die Bestätigung an ehotel versandt wird, ist die Transaktion erfolgreich beendet. Tritt dagegen vorher ein Fehler auf (Hotel ist mittlerweile zum gewünschten Datum ausgebucht), so muss der letzte konsistente Zustand der Daten wiederhergestellt werden.

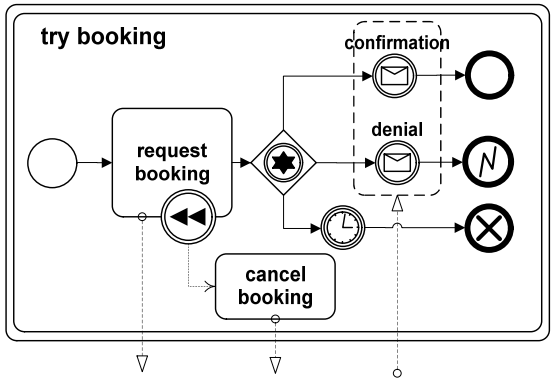


Abb. 10. Buchungs-Transaktion im ehotel-Prozess

4.3 Stornierungsphase

Bei ehotel gibt es zwei grundsätzlich verschiedene Wege zu stornieren: über ehotel oder direkt beim Hotel. Die Variante der direkten Stornierung ist insbesondere für

[1] Scheitert die Kommunikation mit dem System des Hotels, so muss durch die Stornierung später wiederholt werden. Dies kann z.B. durch geschicktes Logging sichergestellt werden.

kurzfristige Stornierungen gedacht. Wird dagegen bei ehotel storniert, so kann dies wiederum über diverse Kommunikationsmedien geschehen. Unabhängig von der Wahl des Kommunikationsmediums ermöglicht die Einfachheit des Stornierungsprozesses eine einheitliche Modellierung. Wesentlicher Grund dafür ist der atomare Charakter der Stornierung. Egal ob der Kunde seine Daten (Buchungsnummer und Stornierungs-TAN) in ein Webformular einträgt, eine E-Mail schickt oder telefonisch eine Stornierung veranlasst, wird bei Ausführung des Stornierungsbefehls im ehotel-System meistens sofort eine Erfolgsmeldung zurückgegeben. Wenn dies nicht der Fall ist – also ein Timeout eintritt – so kann der Kunde den Zustand der Reservierung jederzeit überprüfen – solch eine Statusabfrage ist nicht Teil des Prozesses. Kurz gefasst gibt es genau zwei Zustände, die eine Reservierung nach der Ausführung des Stornierungswunsches haben kann: storniert oder nicht storniert. Stornierungen haben keinen transaktionalen Charakter, eventuelle unerwünschte Datenänderungen können demnach nicht auftreten.

Abstraktion von Navigationsaspekten. Wie zuvor schon erwähnt, sollte die Stornierung unabhängig vom benutzten Kommunikationsmedium modelliert werden. Bei der Betrachtung der Onlinestornierung ist deshalb eine klare Unterscheidung zwischen Prozess und Navigation notwendig – wie generell bei der Modellierung von E-Commerce-Anwendungen [9]. Auf Kundenseite erfolgt erst eine Navigation zur Stornierungsseite, dort kann er – je nachdem ob er ein registrierter Benutzer ist – sich einloggen und eine Buchung zur Stornierung auswählen oder Buchungsnummer und entsprechende Stornierungs-TAN eingeben. Von dieser Navigation kann und muss bei der Prozessmodellierung abstrahiert werden. Die Eingabe der Buchungsnummer entspricht im Prozess nur der Auswahl einer aktiven Prozessinstanz. Innerhalb dieser Prozessinstanz wird mit genau einer Buchung gearbeitet, somit überprüft ehotel nur die dazugehörige Stornierungs-TAN.

Die Ausführung der Online-Stornierungs-Aktivität ist eine reine Navigationsangelegenheit, die an zwei Bedingungen gekoppelt ist. Die erste davon ist prinzipiell zu erfüllen (Voraussetzung), die andere entscheidet über den zu wählenden Pfad (Entscheidungskriterium). Die Voraussetzung besagt, dass der Benutzer auf einer bestimmten Seite des Webauftrittes sein muss, um die Aktion durchführen zu können. Die Stornierung ist nur auf der Stornierungsseite möglich. Diese Bedingung bezieht sich auf einen bestimmten Navigationszustand, der hier in den Prozess integriert ist. Sobald diese Bedingung erfüllt ist, entscheidet die Existenz eines Useraccounts über die Art der Stornierung. Diese Information kann nicht dem Prozessverlauf entnommen werden (der User muss sich nicht prinzipiell einloggen), daher handelt es sich um eine daten-basierte Entscheidung, modellierbar durch ein XOR-Gateway. Abhängig von der Art (registriert oder nicht registriert) des Nutzers ist wiederum nur der weitere Verlauf der Navigation. Will ein nicht registrierter Kunde seine Reservierung stornieren, so muss er Buchungsnummer und Stornierungs-TAN eingeben. Da die Reihenfolge der beiden Subaktivitäten irrelevant ist, empfiehlt sich die Modellierung als „parallel box“ – eine Umsetzung des Workflow-Patterns Synchronisation. [12]. Der eigentliche Prozessaspekt besteht nur aus der atomaren Stornierung.

Variabler Nachrichtenfluss. Die konkurrierenden Varianten der Stornierung erweisen sich bei der Modellierung des Nachrichtenflusses als kritisch. Der Stornierungs-

Task auf Seiten des Hotels kann entweder direkt durch den Kunden oder über ehotel angestoßen werden. Die Rückmeldung kann jedoch nicht eindeutig zugeordnet werden. In der Praxis ist dies unproblematisch, da z.B. der Kunde im Hotel anruft und sofort eine mündliche Stornierungsbestätigung erhält. Gleiches gilt für das Reservierungs-System, welches etwa eine Verbindung zum Hotel (bzw. PEGASUS) aufbaut und auf eine Antwort wartet. Der Betrachter des BPMN-Prozessmodell kann dies nicht zwingend erkennen, hier weist die Notation Defizite auf. Da gerade die Wiederverwendung von Aktivitäten Ziel der Modellierung ist, gestaltet sich die Suche nach alternativen Darstellungen schwierig. Für die Abbildung der gewählten Konstruktion auf BPEL4WS ergeben sich keine Schwierigkeiten, diese ist generell nicht vorgesehen [3].

4.4 Abwicklungsphase

Die Prozesse der Abwicklungsphase benötigen zu ihrer korrekten Ausführung weitere Akteure: Finanzinstitute. Diese könnten als ein Pool modelliert werden, welche Zahlungen der anderen Akteure entgegen nimmt und Mitteilungen über Zahlungseingänge an den jeweiligen Empfänger verschickt. Innerhalb dieses Pools ist für die Modellierung des Hotel-Reservierungs-System also nur ein trivialer Prozess relevant: Buchungseingang und Mitteilung an den Empfänger (vergleiche dazu Abbildung 25).

Sobald eine fristgerechte Stornierung nicht mehr möglich ist, bleibt dem Kunden nur noch die Bezahlung. Egal ob er tatsächlich bezahlt oder nicht erscheint, für ihn beendet sich die Prozessinstanz damit. Das Hotel wartet auf die Bezahlung durch den Kunden oder bis dessen Abreisedatum erreicht ist – modelliert durch ein Event-basiertes Gateway. Die Zahlung kann wiederum durch zwei verschiedene Events signalisiert werden: entweder bezahlt der Kunde bar bzw. per Scheck/Reisescheck (modellierbar durch ein nicht näher spezifiziertes Intermediate Event) oder der Kunde bezahlt mit Karte. Bei Kartenzahlung bekommt das Hotel nicht unmittelbar das Geld, dem Hotel reicht aber eine Mitteilung der Bank (modellierbar durch ein Intermediate Message Event), dass das Geld transferiert wird. Dies geschieht z.B. durch die Anzeige auf dem Display des Kartenlesegeräts. Hier empfiehlt sich die Verwendung des Multiple Event, da die gewählte Zahlungsoption keinerlei Einfluss auf den weiteren Prozessverlauf hat. Eine alternative Modellierung durch zwei verschiedene Events mit gleichem Ziel („order money transfer“) würde eher die Verständlichkeit beeinträchtigen.

Sollte der Kunde nicht erschienen sein, so wird die bei Buchung angegebene Kreditkarte belastet. In beiden Fällen kommt das Hotel zu seinem Geld, anschließend müssen die fälligen Provisionen an ehotel gezahlt werden.[1] Dementsprechend erwartet ehotel diese Zahlung. Da bei der Prozessbeschreibung von ehotel ausdrücklich eine direkte Stornierung des Kunden im Hotel ermöglicht wird, muss das Hotel in diesem Fall ehotel informieren. Fällt eine Stornierungsgebühr an, überweist das Hotel die Provision. Auf Seite von ehotel ist ebenfalls ein bedingter Fluss angebracht. Die

[1] Über die Abwicklung der Provisionszahlungen ist nichts Genaueres bekannt, denkbar wären monatliche Zahlungen, automatische Abbuchungen (Lastschriftverfahren) durch ehotel oder einzelne Überweisungen. Für die Darstellung im Prozess reicht die Modellierung einer allgemeinen Zahlung aus.

Spezifikation von BPMN trifft keinerlei Aussage über bedingten Fluss ausgehend von Events. Die Aussage, dass grundsätzlich jeder Sequenzfluss bedingt sein kann, erlaubt eine solche Modellierung, andererseits wird nur bei Sequenzflüssen ausgehend von Aktivitäten die Notation durch einen „Mini-Diamanten“ eingeführt [3]. Über Events wird an dieser Stelle nichts weiter gesagt. Zu diesem Punkt sollte die Spezifikation noch erweitert werden, da eine Modellierung mit „Mini-Diamanten“ anstelle von XOR-Gateways das Prozessverständnis erleichtern kann.

5 Prozessaspekte im Vergleich: ehotel vs. HRS

Weil HRS auf direkte Stornierung des Kunden beim Hotel verzichtet (vgl. Abbildungen 14 und 26), wird die Modellierung des Nachrichtenflusses für HRS erheblich erleichtert. Es sind keine „verzweigten“ Nachrichtenflüsse mehr notwendig. Die am Ende von Kapitel 4.3 beschriebenen Probleme tauchen nicht mehr auf. Auf Kundenebene fällt ebenfalls eine Stornierungsvariante im Prozessmodell weg. Dies ändert nichts am Prozess selber, da die Anzahl der Varianten ohne weitere Konsequenzen geändert werden kann. Zwischen den Prozessmodellen für die Systeme von ehotel und HRS gibt es signifikante Unterschiede bezüglich zweier weiterer Aspekte, die hier noch erläutert werden sollen.

Synchronisation. Wesentlich einfacher ist der Prozess aus Sicht des Hotels. Im Grunde wird eine Prozessinstanz erst erzeugt, wenn der Kunde nicht mehr stornieren kann. Hier stellt sich die Frage der Synchronisation der Pools. In der Modellierung des ehotel-Prozesses werden automatisch Prozessinstanzen auf Seiten von ehotel und dem Hotel bei der ersten Suchanfrage erzeugt – spätestens aber beim Eintreten eines Events, welches einen Buchungswunsch signalisiert. Die ordnungsgemäße Fortführung des Prozesses nach der Buchung sichern äquivalente Timer-Events in allen Pools. Wird das gebuchte Ankunftsdatum erreicht, schreitet der Prozess in die Abwicklungsphase über, bei Nichterscheinen des Kunden (also wenn das Abreisedatum erreicht ist), terminiert der Prozess auf Kundenseite und gleichzeitig kann der Prozess beim Hotel zur Abwicklung übergehen. Diese Abwicklung endet mit der Provisions-Überweisung an ehotel, wodurch dort der Prozess ebenfalls terminiert. Bei HRS wird dagegen ein extra Task benötigt („notify hotel“), der auf Seiten des Hotels den Prozess erst startet (Abb. 26). Ansonsten gäbe es keine Kommunikation zwischen Hotel und den anderen beiden Pools. Die Benachrichtigung ist somit zur Synchronisation der Prozesse in den unterschiedlichen Pools notwendig.

Umbuchung. Zusätzlich zu den Funktionen von ehotel bietet HRS eine Reservierungsänderung an, wie schon im Use-Case-Diagramm dargestellt (Abbildung 14) [5]. Beobachtungen legen die Vermutung nahe, dass es als eine Verknüpfung der Stornierung mit einer Neubuchung implementiert ist. Die gesamte Aktivität ist teilweise transaktional, was zu unerwünschten Verhalten führt. Änderungen einer Buchung können Anreise- und/oder Abreisedatum, Anzahl der Zimmer oder Gäste betreffen. Wird z.B. ein neues Datum ausgewählt, erfolgt eine problemlose Umbuchung, sofern freie Kapazitäten zum gewünschten neuem Datum zur Verfügung stehen. In der

Buchungsübersicht tauchen dann zwei Buchungen auf: die neue sowie die als storniert gekennzeichnete alte Reservierung. Sind zum neuen Datum nicht genügend freie Zimmer im System vorhanden, so wird gebeten, eine neue Buchung vorzunehmen. Nach Abschluss dieser Buchung zeigt die Übersicht ein eher unerwartetetes Verhalten: beide Buchungen sind eingetragen, die alte aber nicht storniert. Erklärt werden kann dies durch den transaktionalen Charakter der Umbuchung: scheitert die neue Buchung, so wird auch die alte nicht storniert. Da der Benutzer trotzdem in der Navigation weitergeleitet wird, kann er annehmen, dass bei der anschließenden Reservierung automatisch die alte entsprechend storniert wird. Hier bietet ehotel zwar weniger Komfort, ist aber unmissverständlich in der Benutzerführung – eine Notwendigkeit bei der erzwungenen Trennung von Navigation und Prozess.

6 Schlußfolgerungen

Beurteilung von BPMN zur Modellierung. BPMN bietet für die Modellierung komplexer E-Commerce-Anwendungen als eine sehr mächtige Sprache an. Nicht nur die einfache Darstellbarkeit aller Workflow-Patterns, sondern auch darüber hinausgehende Semantiken erlauben eine relativ problemlose Modellierung des beobachteten Verhaltens von ehotel [12], [14]. Die Notation kann den jeweiligen Bedürfnisses angepasst werden; Schwerpunkte können auf einzelne Aspekte in den Modellen gelegt werden. Beispielsweise kann mit Hilfe von Black-Boxes die Einbeziehung weiterer – für einzelne Diagramme aber eher unwichtiger Akteure – sinnvoll dargestellt werden (vgl. u.a. Abbildungen 15 und 19). Positiv hat sich bei der Anwendung die frei wählbare Abstraktionshöhe herausgestellt, vom Top-Level-Prozess (Abb. 15) bis zu Implementierungsdetails lassen sich alle Ebenen der Abstraktion darstellen (Abb. 19). Besonderheiten wie transaktionales Verhalten, komplexe Fehlerbehandlung und Synchronisation lassen sich mit primitiven Mittel darstellen. Modelle werden nicht unnötig verkompliziert. Die flexible Spezifikation erleichtert gerade das Prozessverständnis – der Anwender kann aus unterschiedlichsten Modellierungsvarianten wählen, welche ihm am zweckmäßigsten erscheint. Doch darin liegt gleichzeitig ein gravierender Nachteil der Notation. Es gibt deutlich mehr Notationssymbole als wirklich notwendig – dies erschwert eine schnelle Aneignung der Sprache und kann zu Missverständnissen führen. Hier lässt sich vermuten, dass in naher Zukunft „Best Practices“ zur Modellierung wiederkehrender Muster Abhilfe schaffen.

Modellierung von Varianten. ehotel und andere Hotel-Reservierungs-Systeme bietet dem Kunden verschiedene Möglichkeiten um eine Reservierung vornzunehmen bzw. zu stornieren – diese Varianten müssen im Modell berücksichtigt werden. Bei den hier betrachteten Varianten handelt es sich ausschließlich um Laufzeitvarianten. Bei der Instanzierung des Prozesses kann noch nicht entschieden werden, welche Varianten vom Kunden tatsächlich genutzt werden. Diese Varianten sind durch Semantikänderungen der Gruppierungen in BPMN leicht modellierbar. Eine eigene Notation wie in Abb. 27 verwendet, könnte zu einem besseren Verständnis der Modelle beitragen. In Anlehnung an die Modellierung des Parallel Split Workflow-

Pattern in [14] sollte über eine derartige Erweiterung von BPMN für eine alternative Modellierung des XOR-Splits nachgedacht werden. Es bleibt offen, ob die vorgestellten Konzepte auf die Modellierung von Produktvarianten im Sinne von „Mass Customization“ (vgl. [6]) übertragbar sind. Ansatzpunkt für weitere Arbeiten bildet die Frage der Abhängigkeiten zwischen verschiedenen Variationspunkten.

Mit Hilfe von generischen Modellen ist in BPMN sofort erfassbar, wie der Prozess einer Anwendungsdomäne auszusehen hat, ohne Bezug auf einzelne Implementierungen des Prozesses. Hotel-Reservierungs-Systeme bilden solche Produktfamilien, die sich nur in wenigen Aspekten voneinander unterschieden. Optimierungspotentiale bei der Entwicklung neuer Produktvarianten können durch das Verständnis der allgemeinen Prozesse besser genutzt werden. Beispielsweise könnten Reservierungen per WAP in einem nahe gelegenem Hotel oder Gruppenbuchungen in das vorhandene Modell mit wenig Aufwand integriert werden.

Referenzen

1. Apfelbacher, R., Rozinat, A.: Fundamental Modeling Concepts (FMC) Notations Reference, fmc.hpi.uni-potsdam.de (2004)
2. Bayer, J., Kettemann, S., Muthig, D.: Principles of Software Product Lines and Process Variants. PESOA-Report No. 03/2004, www.pesoa.org/pages/Publications.html (2004)
3. BPMI (Hrsg.): Business Process Modeling Notation (BPMN), Version 1.0, www.bpmn.org (2004)
4. ehotel: http://www.ehotel.de (Juli 2004)
5. HRS: http://www.HRS.de (Juni 2004)
6. Krueger, C.W.: Variation Management for Software Production Lines. In: Chastek, G.J. (Hrsg.): Software Product Lines, Proceedings SPLC 2 San Diego, Springer, Berlin Heidelberg (2002)
7. OMG (Hrsg.): Unified Modeling Language (UML) Superstructure (Final Adopted Specification), Version 2.0, www.omg.org/technology/documents/modeling_spec_catalog.htm (2003)
8. Pegasus Solutions, Inc.: NetBooker, http://www.pegs.com (Juli 2004)
9. Schmid, H.A., Rossi, G.: Modelling and Designing Processes in E-Commerce Applications. IEEE Internet Computing (Januar/Februar 2004) 19-24
10. Schnieders, A., Puhlmann, F., Weske, M.: Process Modeling Techniques. PESOA-Report No. 01/2004, www.pesoa.org/pages/Publications.html (2004)
11. Thatte, S. (Hrsg.): Specification Business Process Execution Language for Web Services, Version 1.1, www.ibm.com/developerworks/webservices/library/ws-bpel/ (2003)
12. van der Aalst, W.: Patterns, tmitwww.tm.tue.nl/research/patterns/patterns.htm (Juli 2004)
13. White, S.A.: Introduction to BPMN, www.bpmn.org (2004)
14. White, S.A.: Process Modeling Notations and Workflow Patterns, www.bpmn.org (2004)

Anhang

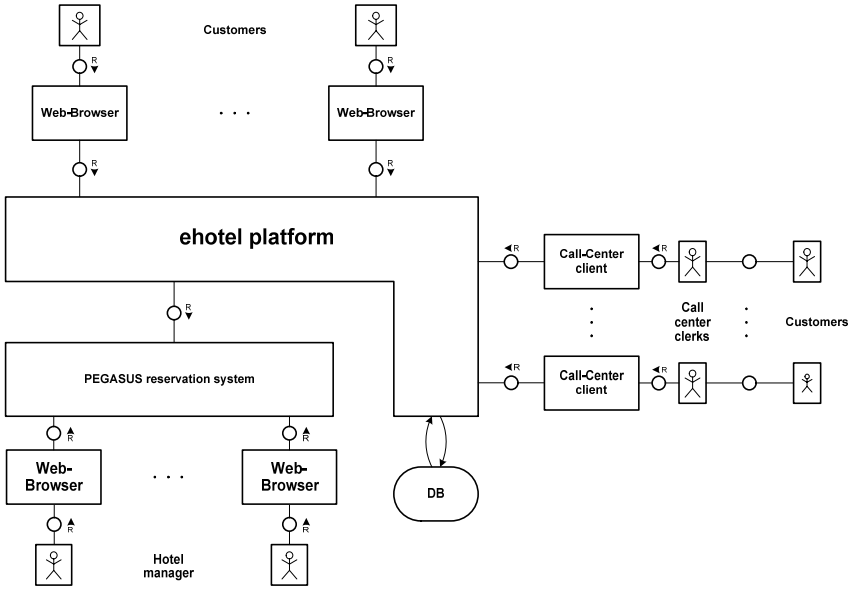


Abbildung 11. Systemaufbau ehotel (FMC)

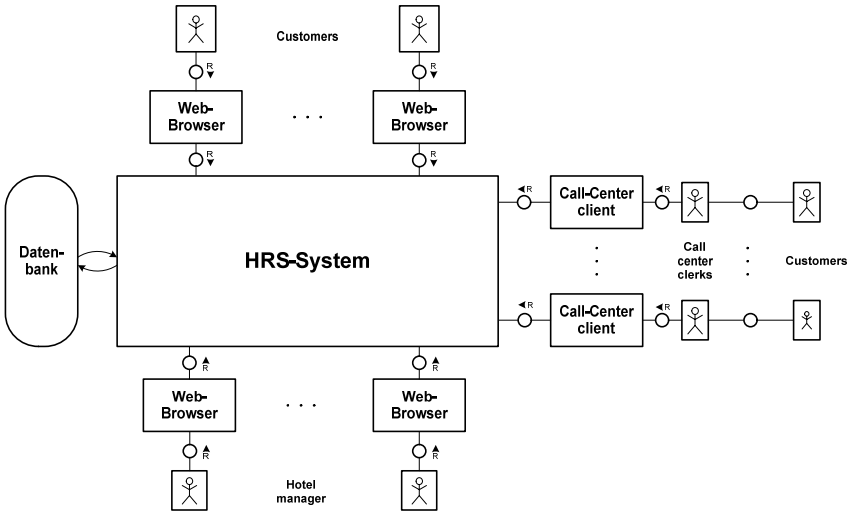


Abbildung 12. Systemaufbau HRS (FMC)

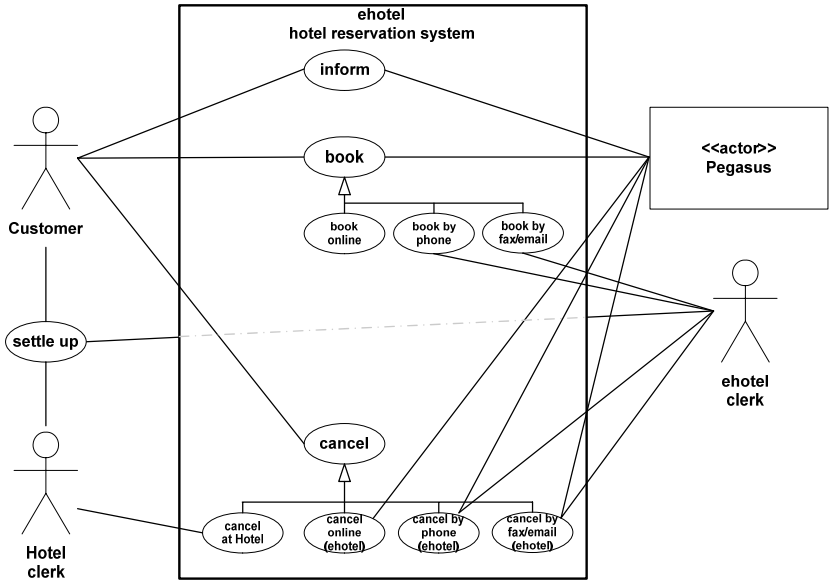


Abbildung 13. Use Cases ehotel (UML 2.0)

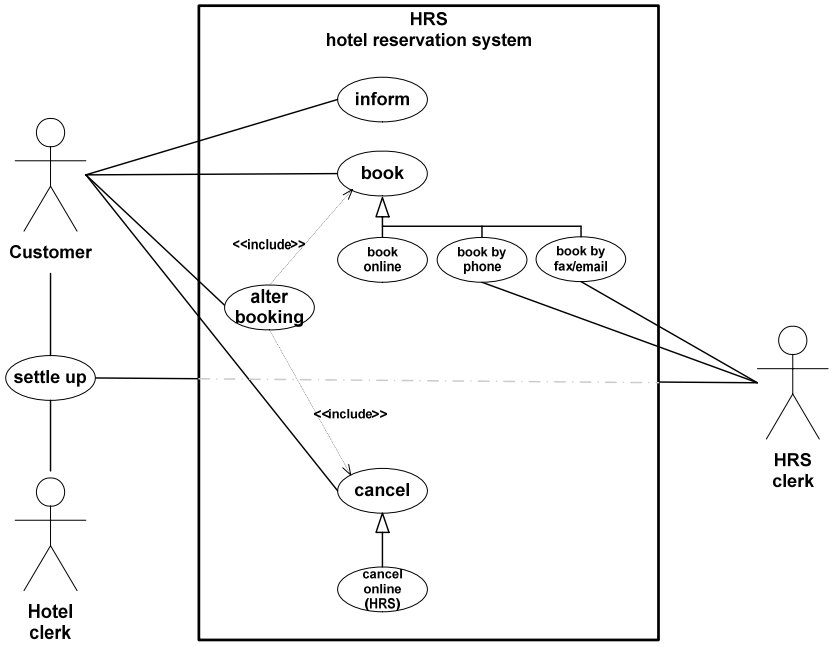


Abbildung 14. Use Cases HRS (UML 2.0)

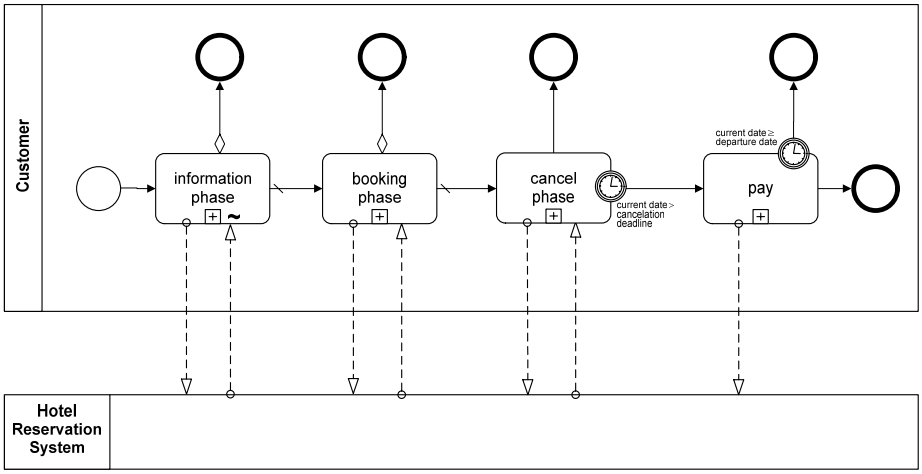


Abb. 15. allgemeiner Top-Level-Prozess Hotelreservierung mit 4 Phasen (BPMN)

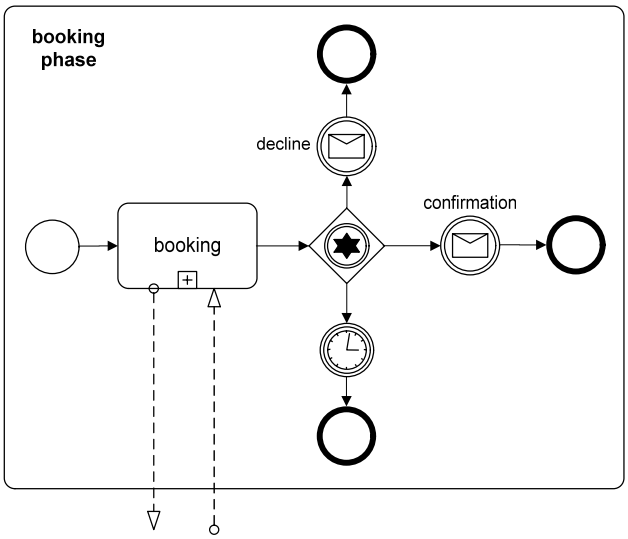


Abb. 16. Ansatzweise Verfeinerung der Buchungsphase des allgemeinen Top-Level-Prozess zur Hotelreservierung (BPMN)

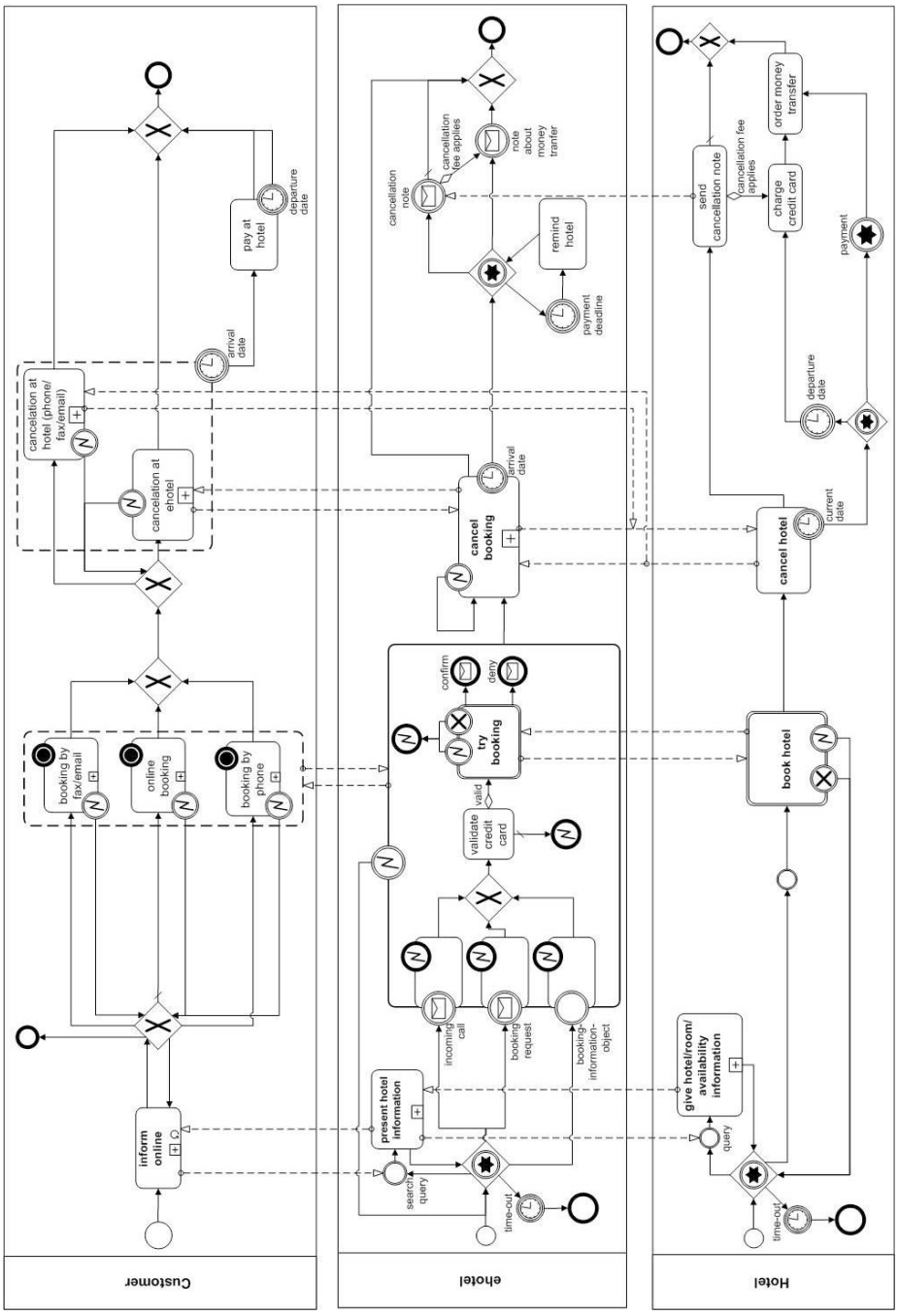


Abb. 17. Prozesse ehotel (BPMN)

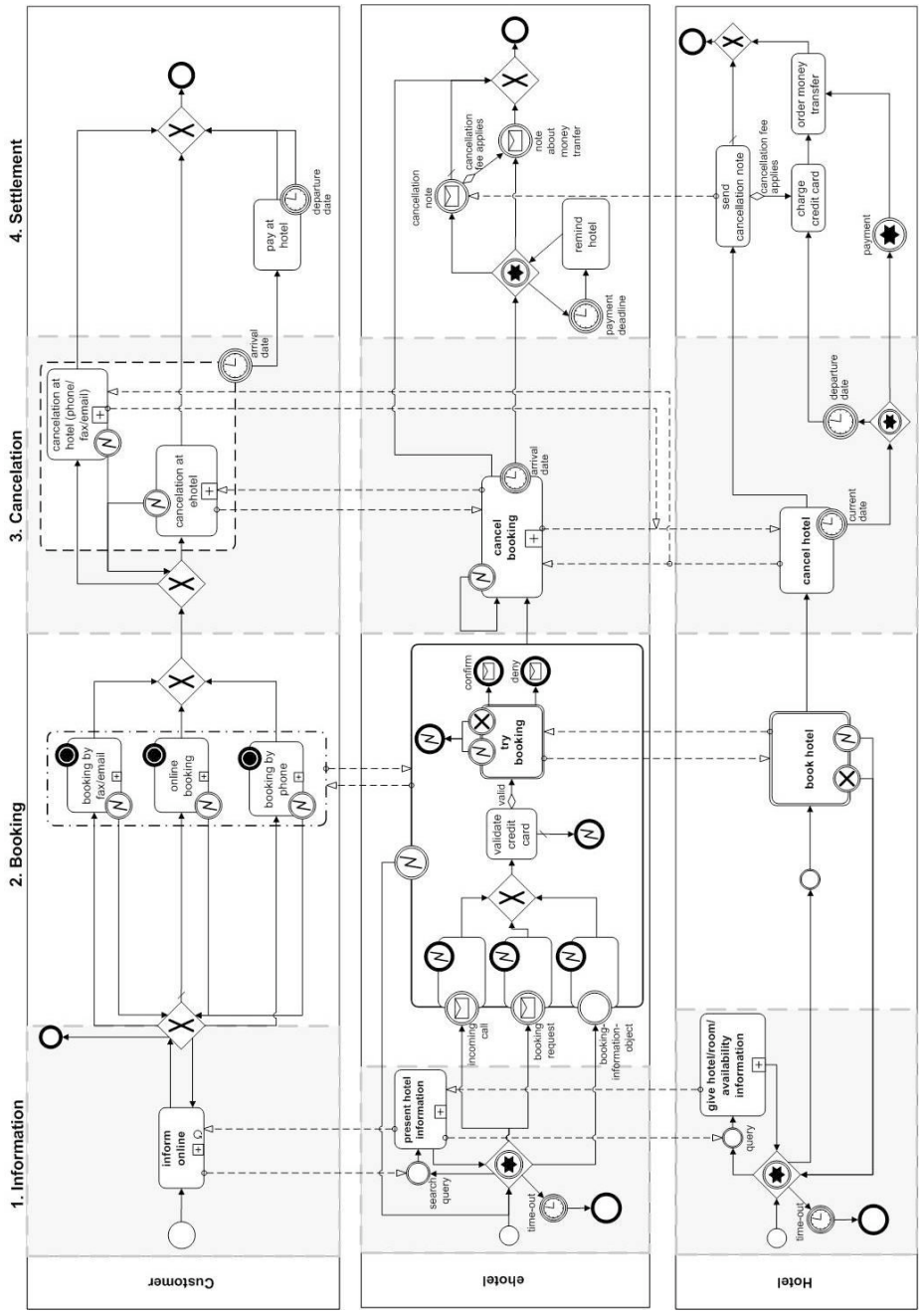


Abb. 18. Phasenzuordnung ehotel (BPMN)

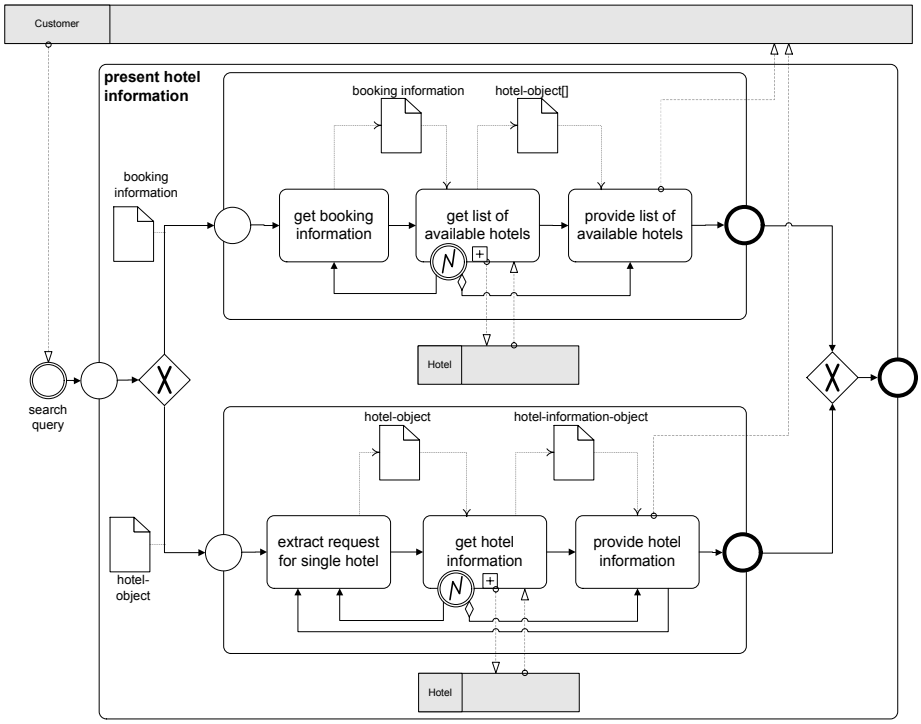


Abb. 19. Verfeinerung Informationsphase: Aktivität „present hotel information“ (BPMN)

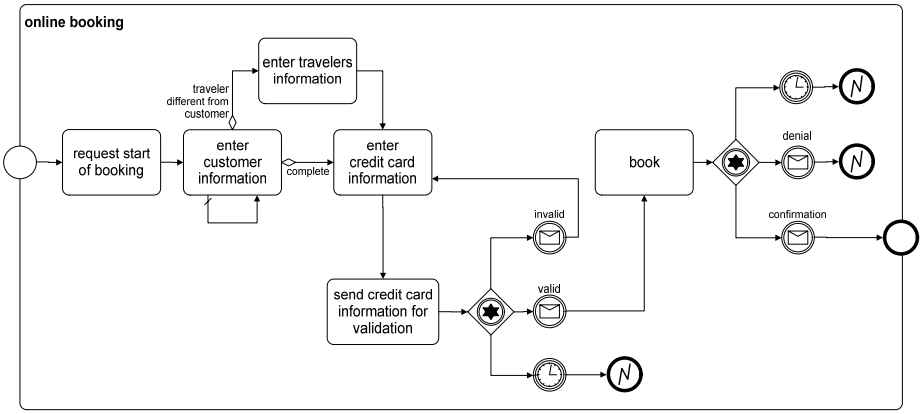


Abb. 20. Verfeinerung Buchungsphase: Variante Internet-Buchung auf Kundenseite (BPMN)

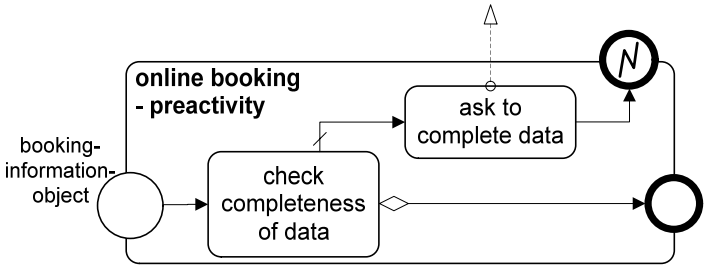


Abb. 21. Verfeinerung Buchungsphase: „Preactivity“ bei Internet-Buchung (BPMN)

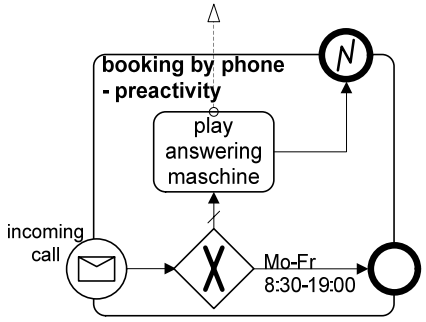


Abb. 22. Verfeinerung Buchungsphase: „Preactivity“ bei telefonischer Buchung (BPMN)

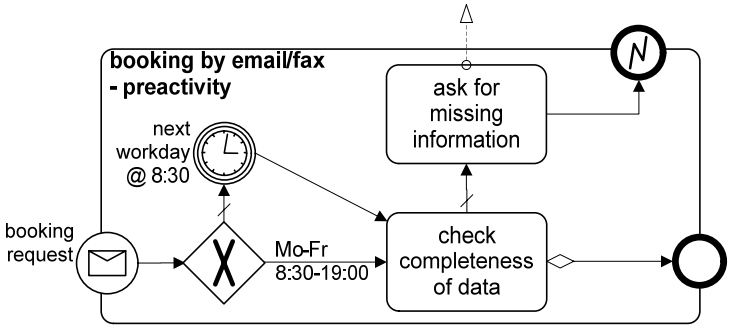


Abb. 23. Verfeinerung Buchungsphase: „Preactivity“ bei schriftlicher Buchung (BPMN)

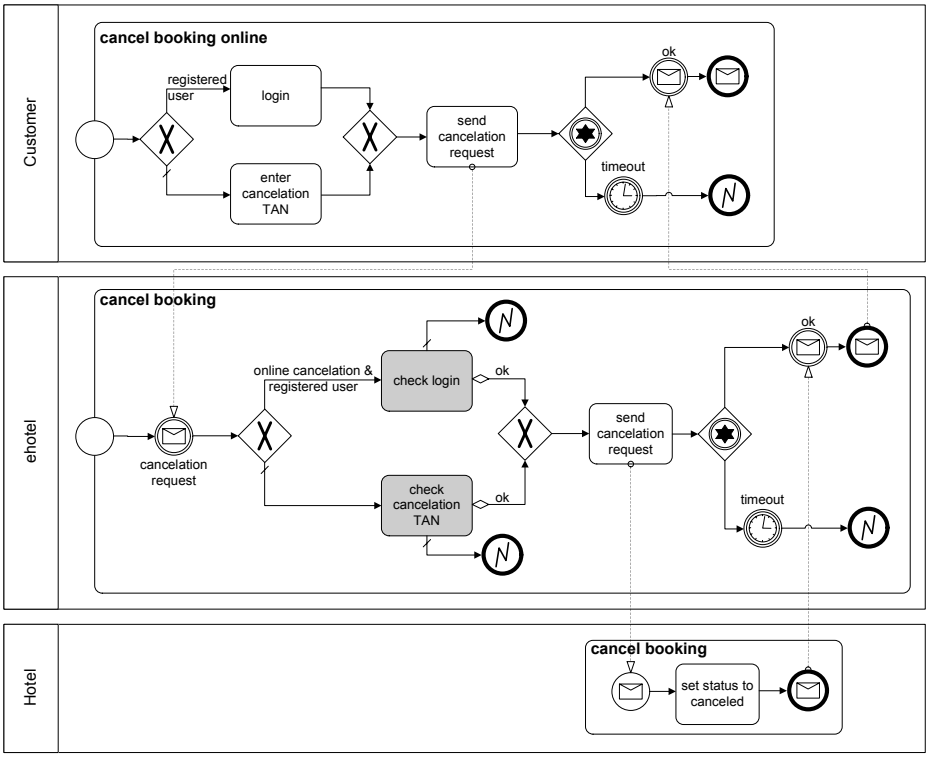


Abb. 24. Verfeinerung Stornierungsphase: Stornierungs-Aktivitäten aller drei Akteure (BPMN)

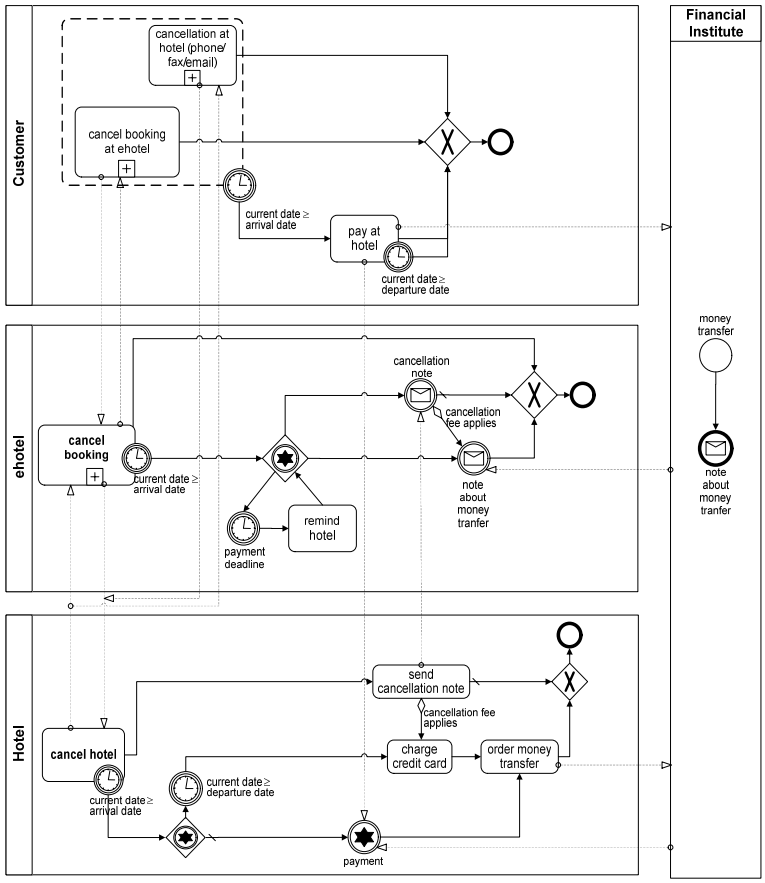


Abb. 25. Verfeinerung Abwicklungsphase: Einbeziehung eines Finanzinstitutes (BPMN)

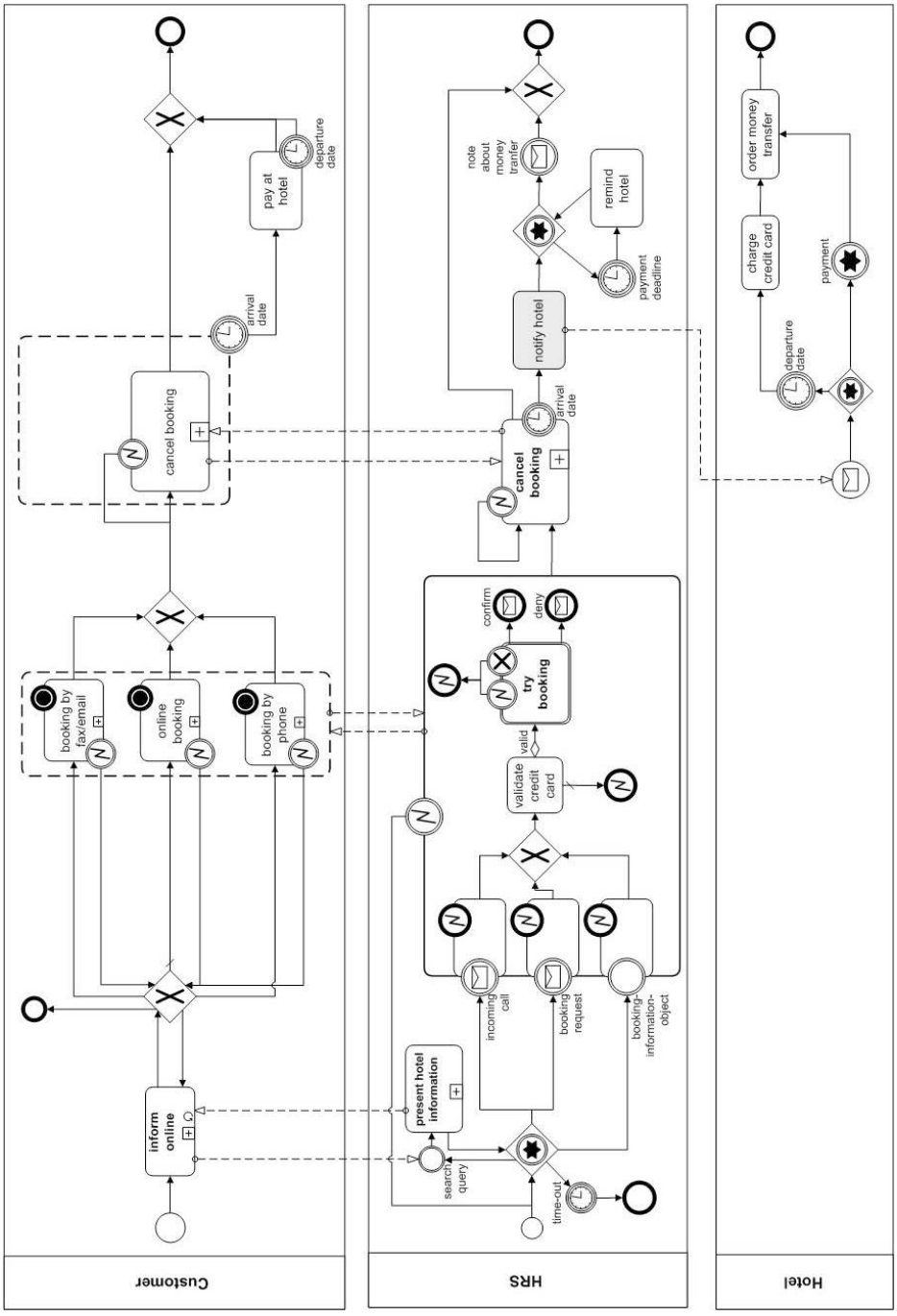


Abb. 26. Prozesse HRS (BPMN)

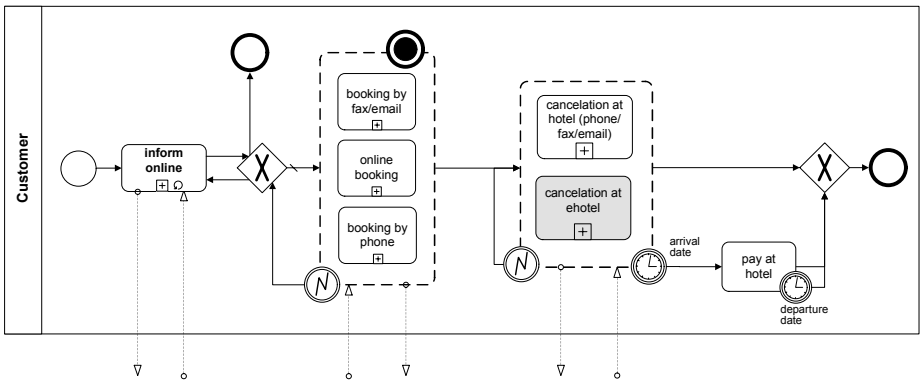


Abb. 27. mögliche Modellierung eines allgemeinen, variablen Reservierungs-Prozesses (Erweiterung zu BPMN)

Die Business Process Modeling Notation im Einsatz

Das Geschäftsprozessmodel von GMX.de und eine Betrachtung der Business Process Modeling Notation (BPMN)

Mario Oschwald

Hasso-Plattner-Institut
mario.oschwald@hpi.uni-potsdam.de

Zusammenfassung. Die Business Process Modeling Notation (BPMN) ist eine neue grafische Notation für die Modellierung von Geschäftsprozessen. In diesem Artikel wird die BPMN benutzt um ein Geschäftsprozessmodel des Webportals von GMX.de aus der Benutzerperspektive zu erstellen. Weiterhin wird die BPMN selbst betrachtet. Einige Probleme der Spezifikation werden erläutert sowie gegebenenfalls Verbesserungsvorschläge gemacht. Zu diesen Problemen gehören insbesondere direkte Widersprüche innerhalb der Spezifikation, unnötige Komplexitäten sowie unklare Aspekte der Syntax und Semantik.

1 Einleitung

Die Business Process Modeling Notation (BPMN)[1] ist eine neue grafische Notation für die Modellierung von Geschäftsprozessen entwickelt von der Business Process Management Initiative (BPMI)[1]. Eines der Ziele der BPMN ist "to provide a notation that is readily understandable by all business users"[1, p. 17], d.h. eine Notation bereitzustellen, welche einfach verständlich für alle Geschäftsleute ist. Die BPMN sollte deshalb einfach erlernbar und benutzbar aber trotzdem genug Ausdrucksstärke besitzen um komplexe Geschäftsprozesse darzustellen.

Inwiefern dieses Ziel erreicht wird ist Bestandteil dieser Ausarbeitung. Aber zunächst wird die BPMN im praktischen Einsatz gezeigt indem beispielhaft einige Geschäftsprozesse der online Plattform GMX.de (Im weiteren einfach nur GMX) mit der BPMN modelliert werden. GMX bietet kostenfreie E-Mail- und Organizer-Dienste an und gehört in diesem Bereich zu den Marktführern. Bei der Modellierung konnte kein "Insider-" Wissen genutzt werden, alle Prozesse welche auf GMX-Seite modelliert wurden basieren deshalb nur auf Annahmen. Nur die Benutzerprozesse konnten verlässlich modelliert werden, das Gesamtmodel ist deshalb nur wirklich aus der Benutzerperspektive zu interpretieren (Oft wurde GMX sogar als Blackbox modelliert).

[1] http://www.bpmi.com

Im Laufe des Modellierungsprozesses wurden einige Probleme der BPMN deutlich. Manchmal war ohne nähere Begründung ein bestimmter Einsatz eines grafischen Elementes verboten, manche grafischen Konfiguration haben keine eindeutige Semantik, sogar direkte Widersprüche innerhalb der Spezifikation konnten entdeckt werden. Weiterhin bietet die BPMN ein gehöriges Maß an syntaktischen Freiheiten (Manchmal kann ein semantisch identischer Prozessteil auf drei verschiedene Arten modelliert werden). Diese Freiheiten stellten sich aber oft als verwirrend und orthogonal zum Ziel der leichten Erlen- und Verständlichkeit dar. Einige Lösungsvorschläge für diese Probleme sind neben deren Darstellung auch Bestandteil dieser Ausarbeitung.

Die Ausarbeitung ist folgendermaßen aufgebaut. Zunächst wird das Objekt der Modellierung, GMX, vorgestellt um den Geschäftsprozessen welche danach modelliert werden einen Kontext zu geben. Danach wird kurz der Modellierungsprozess und erste Erfahrungen mit der BPMN beschrieben, gefolgt von den modellierten Prozessen. Abschließend wird die BPMN selbst zum Thema, wobei gute Erfahrungen aber auch Probleme dokumentiert werden.

2 Was ist GMX.de?

GMX (Global Message eXchange), gegründet FIXME Jahreszahl ist hauptsächlich ein Anbieter kostenloser E-Mail und Organizer-Dienste, damit steht GMX in Konkurrenz mit Anbietern ähnlicher Dienste, wie z.B. Hotmail, WEB.de, Yahoo und neuerdings auch Google.

Ein Kunde kann sich online ein Benutzerkonto erstellen, welchem eine E-Mail Adresse zugeordnet ist, unter der der Kunde erreichbar ist. Zugriff auf dieses E-Mail Konto hat der Kunden über ein Webinterface, aber auch auf klassischem Weg, via POP3 und SMTP. Er kann hierbei alle erdenklichen Aktivitäten in Bezug auf E-Mail durchführen, wie E-Mails lesen, schreiben, löschen, antworten und weiterleiten.

Im Laufe der Zeit wurde das Angebot von GMX immer weiter ausgeweitet. Zu den kostenfreien, werbungfinanzierten Grundiensten kamen noch kostenpflichtige Zusatzdienste wie z.B. erweiterte Organizerfunktionalitäten, größerern Speicherplatz oder erweiterte Zugriffsmöglichkeiten (etwa IMAP). Neuerdings ist GMX sogar in den klassichen ISP-Markt eingetreten und verteibt DSL-Zugänge zum Internet. FIXME Screenshot

Da kein Zugriff auf das kostenpflichtige Angebot von GMX bestand, wird in dieser Ausarbeitung allein auf die klassische Kernkompetenz von GMX eingegangen, d.h. der kostenfreie Basis-E-Mail-Dienst. Dies sollte aber einen genügende Einsicht in die Arbeits von GMX gestatten, da die meisten Zusatzdienste nur aus Erweiterungen das Basis-Angebots bestehen und keine grundlegenden neuen Prozesse beinhalten.

3 Der Modellierungsprozess

Für die Modellierung wurde eine Top-Down herangehensweise gewählt, ausgehend von einem Top-Level Use Case, welcher einfach die Benutzung des web basierten Interfaces beinhaltet (Siehe Abb. 1).

Experimentell wurde nun identifiziert, aus welchen anderen Use Cases dieser Top-Level Use Case zusammengesetzt ist. Die möglichen Use Cases wurden schon früh dahingehend eingeordnet, ob sie sich überhaupt als Prozess modellieren lassen. War dies nicht der Fall (wie z.B. beim einfachen Navigieren druch die Website, welche ohne Zustandspeicherung möglich ist), wurden diese Use Cases nicht näher betrachtet. Als Kriterium, ob ein Use Case Prozesscharakter hat wurden von folgende Eigenschaften benutzt. Ein Use Case mit Prozesscharakter, welcher dann mit der BPMN modelliert werden kann, muss:

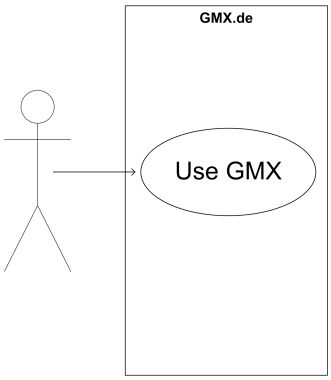


Abb. 1. Der Top Level Use Case

- distinkte Anfangs- und Endpunkte haben
- einen zugehörigen Zustand bzw. Daten besitzen
- eine nicht vernachzulässigende Dauer haben

Ergebnis dieser ersten Unterteilung des Top-Level Use Cases waren vier verschiedene Use Cases (Siehe Abb. 2).

Die Use Cases “Create Account”, “Login” und “Recover Password” sind auf den ersten Blick einleuchtend für eine web basierte Applikation, welche auf Benutzerkonten basiert. Interessant und eigen für GMX ist der Use Case “Handle Emails”, hier liegt die wahre Funktionalität von GMX verborgen. Das Arbeiten mit E-Mails beinhaltet natürlich eine große Menge an möglichen Einzelaktivitäten (z.B. E-Mails lesen, schreiben, löschen, weiterleiten etc.), es würde deshalb natürlich nahe liegen, den wichtigen Use Case “Handle Emails” weiter zu verfeinern um diese Einzelaktivtäten zu beschreiben. Allerdings stellte sich schnell heraus, dass diese Einzelaktivitäten alle von atomarer Natur sind und deshalb keinen Prozesscharakter aufweisen, eine separate Modellierung wäre hier also zwecklos. Letzendlich wurden also folgende Aktivitäten als grundlegend identifiziert und dann auch mit der BPMN modelliert:

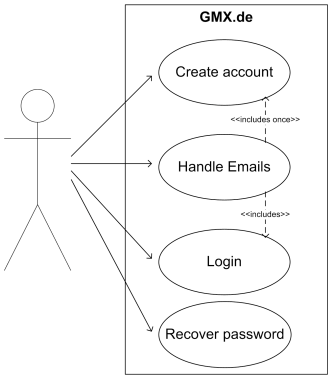


Abb. 2. Die grundlegenden Use Cases

- Arbeiten mit GMX E-Mail

– Erstellen eines Kontos
– Einloggen in das System
– Passwort zurücksetzen
– E-Mail Dienste nutzen

4 Ein Geschäftsprozessmodell von GMX.de

Die Nutzung von GMX ist sehr geradlinig. Nach dem erstellen eines Accounts kann sich der Benutzer einloggen. Sollte dies nicht erfolgreich sein, hat der Nutzer die Möglichkeit sein Passwort auf vier verschiedene Arten zurückzubekommen. Nach dem nun erfolgreichen Login kann der Nutzer nun endlich seine E-Mails bearbeiten. Hierbei ist interessant, dass alle Tätigkeiten (es konnte nur eine Ausahme gefund werden) von GMX-Seite aus atomar sind. Das bedeutet, dass GMX neben der Tatsache, dass der Benutzer eingeloggt ist, keine Zustandsdaten für die einzelnen Aktionen speichern muss. Die ganze Kommunikation folgt einem einfachen Request/Response Muster. Dies ist natürlich vor allem für GMX vorteilhaft, weil Resourcen gespart werden. So können viele Benutzer gleichzeitig einen Server nutzen (Obwohl GMX schon ein gut verteiltes System nutzt).

Für die Modellierung mit der BPMN wurde für alle Prozesse und deren BPDs ein ähnlicher Grundaufbau gewählt: GMX und der Benutzer werden jeweils durch einen eigenen Pool dargestellt, zwischen denen ausschliesslich Messageflows geschehen dürfen.

Da die Prozesse relativ einfach sind, konnte auf einige Modellierungskonzepte der BPMN verzichtet werden. So werden z.B. keine Transaktionen incl Error, Cancelation und Compensation Events. ([1], p.72) eingesetzt (obwohl natürlich auf tieferer Ebene bei GMX durchaus Transaktionen bei der Datenhaltung auftreten werden) auch komplexere Aktivitätstypen, wie Multiple Instance, Loop und Ad Hoc wurden nicht eingesetzt.

In Abschnitt 5 werden einige Vorschläge zu Vereinfachung von BPMN gemacht. Die BPDs für das GMX-Geschäftsprozessmodell wurden mit dieser vereinfachten BPMN erstellt. So wurden z.B. immer Start/Stop Events benutzt, keine Default-Flows, Loop-Marker und Conditional-Flows genutzt und darauf verzichtet alternative Modellierungsmöglichkeiten für XOR-Splits, sowie Flowtrennung bzw. Zusammenführung zu nutzen. Die BPDs enthalten deshalb im Allgemeinen mehr Elemente als zwingend von der BPMN vorgeschrieben.

4.1 Prozessüberblick

In Abbildung 4 wird der grobe Prozessüberblick, der Use Case “Use GMX” als BPD dargestellt. GMX ist als Blackbox modelliert (Die Messageflows gehen nur zum Pool von GMX, keine internen Prozesse werden gezeigt), nur der Benutzerprozess ist zu sehen. Wenn der Benutzer noch keinen Account hat, hat er die Möglichkeit im “Create Account” Sub-Prozess diesen zu erstellen, der SUB-Prozess wird in einem anderen BPD noch dargestellt, deshalb der Plus-Marker.

Wenn der Account erstellt ist kann der Nutzer im “Login” Sub-Prozess sich einloggen, hat dies funktioniert, werden im “Handle E-Mails” Sub-Prozess die E-Mails bearbeitet. Wenn der Benutzer damit fertig ist oder wenn der Login nicht funktioniert hat wird der Benutzer-Prozess beendet.

Dieses BPD ist sehr einfach, und enthält neben Sequenzen nur XOR-Splits und Joins.

4.2 Benutzerkonto erstellen

Beim Erstellen eines Benutzerkontos passiert eigentlich nicht viel Interessantes. GMX fragt sequenziell eine große Menge an persönlichen Informationen ab, nachdem es diese alle erhalten hat, wird der Account erstellt. Da die kostenlosen Dienste rein werbefinanziert sind, möchte GMX natürlich gerne so viel wie möglich über seine Kunden wissen. Ob allerdings, das Abfragen von ca. 200 Einzeldaten auf 10 Formularen nicht doch Overkill ist und eher zu Falschangaben führt, sollte wohl bedacht werden. GMX scheint dies nicht zu glauben und wirbt auf seiner Website für Werbekunden[2]

> “Kaum ein Anbieter ... hat so viele Informationen von seinen Nutzern wie GMX. Denn jedes GMX Mitglied muss bei der Anmeldung einen ausführlichen Registrierungsfragebogen ausfüllen. Durch die gezielte Abfrage ... erhält GMX von jedem Mitglied ein exaktes Nutzerprofil”

Da einfach nur sequenziell etwas abgefragt wird, hatte ich zunächst angefangen dies genauso zu modellieren, als eine lange Sequenz von Einzelabfragen und Antworten (Abbildung 5[3]). Dieser Ansatz erschien aber schnell als sehr unflexibel und unschön, deshalb wurde ein allgemeinerer Ansatz gewählt, der nicht die Einzelabfragen hintereinander schaltet, sondern in einer Schleife die notwendigen Abfragen aus einer Datenbank liest und diese dann an den Benutzer zur Beantwortung schickt (Abbildung 6).

Der Benutzer wählt einen Accounttyp und startet damit über ein Message-Start-Event auf GMX-Seite den Login-Prozess. GMX liest die nächste Anfrage in der “Assemble Data” Aktivität, danach wird diese an den Benutzer geschickt, der in einem Event-Based-Gateway auf die nächste Nachricht von GMX wartet. Der Nutzer beantwortet die Anfrage, diese wird validiert. Wenn die Antworten in Ordnung sind, wird geprüft, ob mehr Anfragen anstehen, dann wird die Schleife noch einmal ausgeführt. Ist die Validierung nicht erfolgreich, wird die gleiche Anfrage noch einmal geschickt. Werden keine weiteren Daten vom Benutzer benötigt, bekommt der Benutzer eine Nachricht, dass der Account erstellt wurde, danach werden auf den jeweilgen Servern, die in einer anderen Swimlane modelliert wurden, der Account erstellt.

---

[2] FIXME URL

[3] Achtung, dieses BPD ist unvollständig und soll nur den ersten, missglückten Ansatz demonstrieren.

Falls der Benutzer für eine bestimmte Dauer keine Antworten mehr schickt, wird über ein Rule-Based-Event, der Login-Sub-Prozess auf GMX-Seite abgebrochen, alle bis dahin gespeicherten Daten gelöscht und der Prozess beendet.

In diesem BPD ist vor allem das Event-Based-Gateway nennenswert, welches erlaubt anhand eines Ereignisses den Prozessfluss zu bestimmen. Je nachdem welche Nachricht von GMX kommt (Datenabfrage oder Erfolgsmedlung) wird auf Benutzerseite ein anderer Pfad eingeschlagen. Weiterhin wurde das Rule-Based-Event dazu benutzt, den Prozess abzubrechen, wenn der Benutzer einfach entscheidet keine Daten mehr zu schicken. Es wurde nicht ein Timer-Event gewählt, da dieser nur einen festen Zeitpunkt oder ein Intervall als Auslöser haben kann ([1], p. 50). Somit müsste nach jeder Benutzereingabe der Timer wieder auf einen neuen Zeitpunkt eingestellt werden, was unpraktisch ist.

4.3 Login

Hat der Benutzer nun (nachdem er sein Innerstes nach Außen gekehrt hat) einen Account kann er sich einloggen. Dieser Prozess, dargestellt in Abbildung 7, ist natürlich aus vielen Applikationen bekannt und auch nicht weiter überraschend in der Modellierung.

Zunächst sendet der Benutzer seine Zugangsdaten und startet damit auf GMX-Seite den Login Prozess. GMX prüft ob die Zugangsdaten korrekt ist, ist dies der Fall wird eine Session gestartet und über ein Link-End-Event an einen anderen Prozess (E-Mail Dienste nutzen, 4.5) übergeben, in dem dann auf GMX-Seite die E-Mail-Bearbeitung stattfindet. Sollten die Zugangsdaten nicht korrekt sein, wird dies dem Nutzer übermittelt. Dieser hat dann drei Möglichkeiten:

1. Aufzugeben, und damit den Login-Prozess beenden
2. Es noch einmal versuchen
3. Sein Passwort zurückzubekommen

Wenn der Kunde es noch einmal versucht, wird auf GMX-Seite ein neuer Login-Prozess gestartet. Dies ist nicht problematisch, da zu diesem Zeitpunkt der vorhegende Login-Prozess schon beendet wurde. Es werden also keine unabgeschlossenen Prozesse übrig bleiben. Sollte der Kunde versuchen sein Passwort zurückzusetzen, passiert dies in einem separaten Sub-Prozess. Dieser kommuniziert auch mit GMX, was über Messageflows dargestellt wurde. Da die Details allerdings erst in einem anderen BPD dargestellt werden, berühren diese Messageflows nur den GMX-Pool.

4.4 Passwort zurücksetzen

Obwohl es hoffentlich nicht oft passiert, ist auch dieser Vorgang ein eindeutiger Geschäftsprozess. Um einen Benutzer sein (vergessenes) Passwort zurücksetzen zu lassen hat sich GMX vier verschiedene Methoden ausgedacht. Bis auf die Methode ein Fax an den Support zu schicken basieren alle drei anderen Methoden

auf dem gleichen Prinzip. Deshalb wurde nur eine davon im nächsten Abschnitt genauer modelliert.

In Abbildung 8 ist das BPD für den Ober-Prozess des Passwort zurücksetzens zu sehen. Nachdem der Benutzer angezeigt hat, dass er sein Passwort zurücksetzen will, macht GMX zwei Dinge parallel:

- Ein Code/Hyperlinkpaar generieren
- Den Benutzer fragen, welche Methode er wählen will

Der Benutzer entscheidet sich nun für eine Methode und gibt dies GMX bekannt. Will er per Fax das Passwort zurücksetzen beendet dies den Prozess, da dies nicht online passiert - dieser Vorgang wurde also nicht modelliert. Entscheidet der Benutzer sich für eine der drei anderen Methoden startet jeweils auf beiden Seiten ein entsprechender Sub-Prozess. In diesen wird auf einem sicheren Weg ein Teil des Code/Hyperlinkpaares dem Benutzer übermittelt (Sicher in dem Sinne, dass es nur den Benutzer selbst erreichen kann), der andere Teil wird direkt angezeigt. Es steht zur Auswahl:

1. Link an alternative E-Mail, Code direkt anzeigen
2. Link an GMX-E-Mail (fals z.B. ein E-Mail Client noch das Passwort gespeichert hat), Code direkt anzeigen
3. Link direkt anzeigen, Code per SMS an Mobil-Telefon schicken

Im Idealfall hat der Benutzer nun den Code und den Hyperlink und kann über das Anklicken des Hyperlinks und Eingabe des Codes sein Passwort zurücksetzen.

Obwohl der ganze Vorgang recht einfach ist, sieht das BPD relativ komplex aus. Dies ist durch die Tatsache verursacht, dass vier verschiedene Pfade auf beiden Seiten gewählt werden müssen, je nachdem wofür sich der Benutzer entscheidet. Man kann sich leicht vorstellen, dass bei einer komplexeren Situation (beispielswiese ein Online-Spiel) diese Modellierungsweise sehr schnell unüberscihtlich wird, wenn die Entscheidungspfade in die Hunderte gehen kann. Hier wäre schön, wenn die Entscheidung nicht über separate Message-Events sondern über ein einziges, welches die Entscheidung enthält transportiert werden könnte. Leider kann aber ein Message-Event nicht von verschiedenen Aktivitäten ausgelöst werden (Nur einen eingehenden Pfeil haben [1], p. 62) - es muss also für jede Entscheidungsmöglichkeit ein unterschiedliches Message-Event ausgelöst werden.

Bis auf dies ist der Prozess recht einfach. Es wird zum ersten Mal ein (kurzer) paralleler Fluss benutzt, ansonsten enthält er nur Muster, die schon in anderen BPDs enthalten waren.

Passwort mit alternativer E-Mail zurücksetzen Hat der Benutzer eine alternative E-Mail Adresse angegeben, kann er mit dieser sein Passwort zurücksetzen (Abbildung 9). Er erhält zunächst von GMX den Code direkt über seinen Browser, an die alternative E-Mail-Adresse wird der Link geschickt. GMX wartet nun, bis der Code über den zugehörigen Link eingegeben wurde, und erlaubt, wenn diese zusammen passen, das Neusetzen des Passwortes. Ist der Code nicht korrekt, kann der Benutzer es noch einmal versuchen.

Interessant ist hier, dass kein eigener Timeout-Mechanismus für die Codeeingabe benutzt wird. Da der Prozess ein Sub-Prozess ist, und sein Super-Prozess (Abbildung 8) schon einen Timeout-Mechanismus hat, wird auch der Sub-Prozess von diesem Timeout abgedeckt.

4.5 E-Mail Dienste nutzen

Alle bis jetzt beschriebenen Prozesse waren eigentlich nur Vorarbeit um den eigentichen Dienst von GMX wirklich zu Nutzen, E-Mail. Wie schon gesagt ist dieser Haupteil von der Modellierung aber eigentlich der einfachste. Wenn der Benutzer eingeloggt ist, hat der die Möglichkeit eine große Zahl an verschiedenen Aktionen auszuführen, welche alle unter dem Titel "E-Mail Dienste nutzen" einzuordnen sind. Allerdings sind diese alle atomarer Natur. Das bedeutet, dass nach Antreffen eines Requests etwas Bestimmtes zu tun, GMX dies sofort durchführen kann (z.B. eine E-Mail queuen, oder eine Adresse aus der Datenbank löschen, ein Verzeichnis anlegen) und dabei entweder Erfolg hat oder eine Fehlermeldung zurückgeben kann. Mögliche Aktionen umfassen:

- Anschrift ändern
- Persönlich Daten ändern
- Bankverbindung angeben
- Rechnungen einsehen
- Passwort ändern
- Erotik-Werbung ein/ausschalten
- E-Mail schreiben E-Card schreiben
- Posteingang ansehen
- E-Mail löschen
- E-Mail weiterleiten
- E-Mail beantworten
- Spamordner ansehen
- Ordnereinstellungen ändern
- Ordner anlegen / löschen / umbenennen
- Kontakt anlegen / löschen
- Visitenkarte ändern
- Verteilerliste erstellen / löschen / umbenennen
- Email an Kontakt / Verteiler schreiben
- Gruppe anlegen / ändern / löschen
- Adress/Telefon/E-Mail Liste anzeigen
- Abwesenheitsnotiz ändern / aktivieren / deaktivieren
- E-Mail Anzeige konfigurieren
- Absendername einstellen
- Adressen anlegen
- Adresse löschen
- Hauptadresse setzen
- Rechnungsadresse setzen
- E-Mail Filterregel einrichten /ändern / löschen

– POP3 Sammeldienst einrichten/ändern/löschen
– SMTP AUTH bzw. SMTP after POP aktivieren
– Signaturen anlegen / ändern / löschen
– Spamschutz aktivieren / konfigurieren / deaktivieren

Es wurde gesagt, daß alle diese Aktionen atomar sind. Zu jeder Regel gibt es ja Ausnahmen, so auch hier. Die beiden obigen Aktionen, welche auch etwas Prozesscharakter haben sind E-Mail schreiben (da hier ein Attachment hochgeladen werden kann!) und der POP3-Sammeldienst (da hier ein Cronjob o.Ä. in regelmäßigen Abständen einen externen POP3-Account abfragt). Im Großen und Ganzen ist die Aussage, das diese Aktionen atomar sind also dennoch wahr.

Da die Aktionen atomar sind, und der Benutzer nur jeweils eine Aktion (mit evtl. zugehörigen Informationen) auswählen kann, ist dieser Prozess von Benutzer-Seite aus auch nichts weiter als eine einfache Schleife in der die Aktionen ausgewählt werden. Diese wird abgebrochen, wenn der Benutzer keine weiteren Anfragen mehr senden möchte. Auch auf GMX-Seite ist, nachdem der Prozess durch ein Link-Start-Event vom Login-Prozess (Abbildung 7) gestartet wurde, eine Schleife zu finden. In dieser wird auf neue Anfragen gewartet, diese bearbeitet und eine Rückmeldung geschickt. Sollte der Benutzer sich manuell ausloggen oder ein Timeout auftreten, wird die Session gelöscht und der Prozess beendet sich.

Interessant ist hier eigentlich nur die Methode, mit der der Timeout modelliert wurde. Am Schleifenanfang auf GMX-Seite steht ein Event-Based-Gateway, welches durch eine neue Anfrage, den Logoutrequest oder den Timeout den Fluss des Prozesses bestimmt. Der Timeout wirkt sich also nicht unterbrechend auf eine etwaige gerade ausgeführte Aktion aus.

5 Die Business Process Modeling Notation (BPMN)

Bei der Arbeit mit der BPMN wurde sehr schnell deutlich, dass diese sehr ausdrucksstark ist. Zu keiner Zeit wahr fraglich, ob ein bestimmter Sachverhalt überhaupt dargestellt werden konnte[4]. Trotzdem war manchmal nicht sofort eindeutig, wie genau ein bestimmter Sachverhalt darzustellen war. So wurde für das Muster, dass in einem Pool eine Entscheidung gefällt wird, diese dann in einen anderen Pool kommuniziert wird, worauf dort der Fluss entsprechend gewählt werden muss, nur der etwas umständliche Weg gefunden, dies über verschiedene Message-Event-Instanzen darzustellen (Siehe Abschnitt 4.4)

Weiterhin war die Syntax der BPMN, entgegen dem Anspruch, dass diese leicht erlenrbar sein soll, nicht immer schnell memorisierbar. Da erstens die syntaktischen Regeln verteilt über die Spezifikation und umgangssprachlich formuliert sind, war oft einiges an Nachschlagen notwendig, in einem Fall (Message-Events dürfen nur einen eingehenden Messageflow haben) mussten sogar einige

[4] Dies mag natürlich auch daran liegen, dass die modellierten Prozesse alle nicht sehr komplex waren.

schon falsch gezeichnete BPDs angepasst werden. Hier kann wohl nur mit Hilfe eines syntaxprüfenden Tools abgeholfen werden.

Als problematisch könnte sich auch erweisen, dass einige nicht-grafische Attribute, welche also nur von einem speziellen BPMN-Tool benutzt werden könnten, doch starke Auswirkungen auf die Semantik eines BPDs haben können. So haben beispielsweise die aus einem Gateway ausgehenden Pfeile eine bestimmte Ordnung, welche grafisch nicht darstellbar ist. Diese Ordnung ist aber für die Festlegung welcher Ausgang gewählt wird mitbestimmend. Auch wird die Reihenfolge (parallel oder sequenziell) der Abarbeitung eines Ad-Hoc Prozesses nur nichtgrafisch bestimmt. Ob eine Aktivität vom Typ Service, Send, Receive etc. ist, kann auch nicht direkt grafisch abgelesen werden, hat aber starke Auswirkungen wie z.B. Messageflows angebracht werden dürfen. In all diesen Fällen kann zwar mit externen Annontation versucht werden, die wichtigen nichtgrafischen Attribute darzustellen - ein effektiver Einsatz wird aber auch wohl nur durch ein spezielles Tool möglich.

5.1 Mögliche Vereinfachungen

Die BPMN erlaubt in einigen Fällen semantisch identische Prozessteile mit verschiedenen grafischen Konfiguration darzustellen. Diese Alternative werden sehr schön in einem Paper[2] von S. White dargestellt. Weiterhin ist die Semantik einiger grafischer Elemente auch mit (Kombinationen) anderer grafischen Elemente darstellbar.

Alternative Darstellungsmöglichkeiten Diese Freiheiten erlaubt die BPMN um in Spezialfällen die BPDs übersichtlicher gestalten zu können, da weniger grafische Elemente für die Darzustellung notwendig sind.

In Abbildung 11 sind zwei semantisch identische Prozesse aufgezeigt, allerdings werden die Modellierungsfreiheiten der BPMN voll ausgenutzt. Folgende Freiheiten bietet die BPMN:

- Es steht frei, ob Start- oder Stop-Events gezeichnet werden, allerdings muss man sich für eine Alternative entscheiden
- XOR-Gateways können mit einem Andreaskreuz oder als leere Raute gezeichnet werden (Da dies wohl der häufigste Gatewaytyp ist)
- Parallel-Splits können alternativ mit einem Parallel-Gateway oder durch mehrere ausgehende Flüsse an einer Aktivität modelliert werden
- Analog können XOR-Joins entweder mit einem XOR-Gateway oder mit mehreren eingehenden Flüssen modelliert werden
- Ein parallel Pfad kann mit einen Parallel-Split, gefolgt von einem Parallel-Join modelliert werden oder mit einer eigenen Aktivität, welche die Pfade separat enthält
- OR-Splits können mit einem OR-Gateway oder mit mehreren ausgehenden Conditionalflows modelliert werden

Auf den ersten Blick erscheinen Freiheiten natürlich als positiv. In diesem Falle bringen diese Freiheiten aber doch einige zusätzliche Probleme mit sich, welche nicht unbedingt von den Vorteile überwogen werden:

- Die BPMN wird schwerer zu Lernen, da auch die Alternativmöglichkeiten gelernt werden müssen
- Kennt jemand nicht eine bestimmte Alternative, wird das Verständins eines BPDs, welche diese nutzt unmöglich
- Wenn verschiedene Alternativen gemischt benutzt werden, verliert das BPD an Konsistenz.

Überflüssige grafische Elemente Eigene Elemente die eigentlich überflüssig, d.h. anders präsentiert werden können, sind der Loop-Marker und Default-Flow-Pfeil. Jede "loopende" Aktivität kann auch mit einer Schleife modelliert werden. Hierbei werden auch die Abbruchbedingung (oder die Durchlaufzahl) der Schleife explizit an der Entscheidung angegeben. Mit dem Loop-Marker müssen diese über Attribute gesetzt werden. Allerdings vereinfacht der Loop-Marker mit den Schleifenattributen natürlich das Mapping auf eines Business Process Execution Language (BPEL). Der Default Pfeil als grafisches Element ist deshalb nicht notwendig, da einerseits ein (nicht-grafisches) Attribut diese Eigenschaft schon beschreibt, andererseits die Default-Bedingung auch immer explizit angeben werden kann. Dies ist z.B. möglich durch einfache Negierung der Disjunktion aller anderen Bedingungen.

Vorschlag zur Vereinfachung Da die BPMN ein "Work in Process" ist, also noch Änderungen vorgenommen werden können, schlage ich für eine nächste Version vor einige Änderungen zur Vereinfachung vorzunehmen (Siehe Abbildung **??**):

- Entfernen des grafischen Elements Conditional-Flow (Dieser kann immer mit Gateways ausgedrückt werden)
- Entfernen des grafischen Elements Default-Flow (Kann immer durch Bedingung/Attribute ausgedrückt werden)
- Entfernen des grafischen Elements Loop-Marker (Kann immer durch bedingten Fluss ausgedrückt werden)
- Verlangen, dass immer explizit Start- und Stop-Events genutzt werden (Damit werden die Anfangs und Endpunkte eines Prozesses sofort deutlich, man muss nicht erst manuell aller Aktivitäten anschauen, und diejenigen identifizieren, welche keinen eingehend bzw. ausgehenden Fluss haben)
- Nur ein einziges Icon für XOR-Gateways zulassen
- Keine Splits- oder Joins ohne Gateways (Der Typ eines Splits bzw. Joins wird sofort aus dem Gatewaytyp ersichtlich)

Diese Änderungen würden dei BPMN auf jedem Fall ihrem Ziel näher bringen leicht erlernbar zu sein. Auch würden BPDs zwar im Mittel etwas mehr Elemente

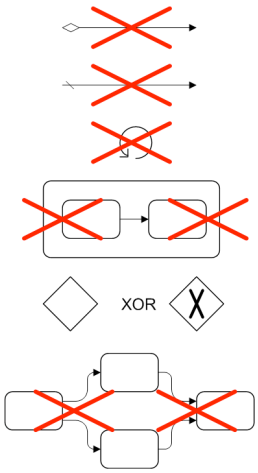


Abb. 3. Mögliche Vereinfachungen der BPMN

enthalten aber dafür deutlicher und schneller interpretierbar werden. Da nichts Neues bei diesen Vorschlägen hinzugefügt wird, sondern nur bestehende Teile der Spezifikation eingeschränkt oder entfernt werden, sollten die Änderungen auch schmerzfrei vornehmbar sein.

5.2 Widersprüche und Fehler in der Spezifikation

6 Fazit

7 Abbildungen

Abb. 4. Top Level Prozessüberblick

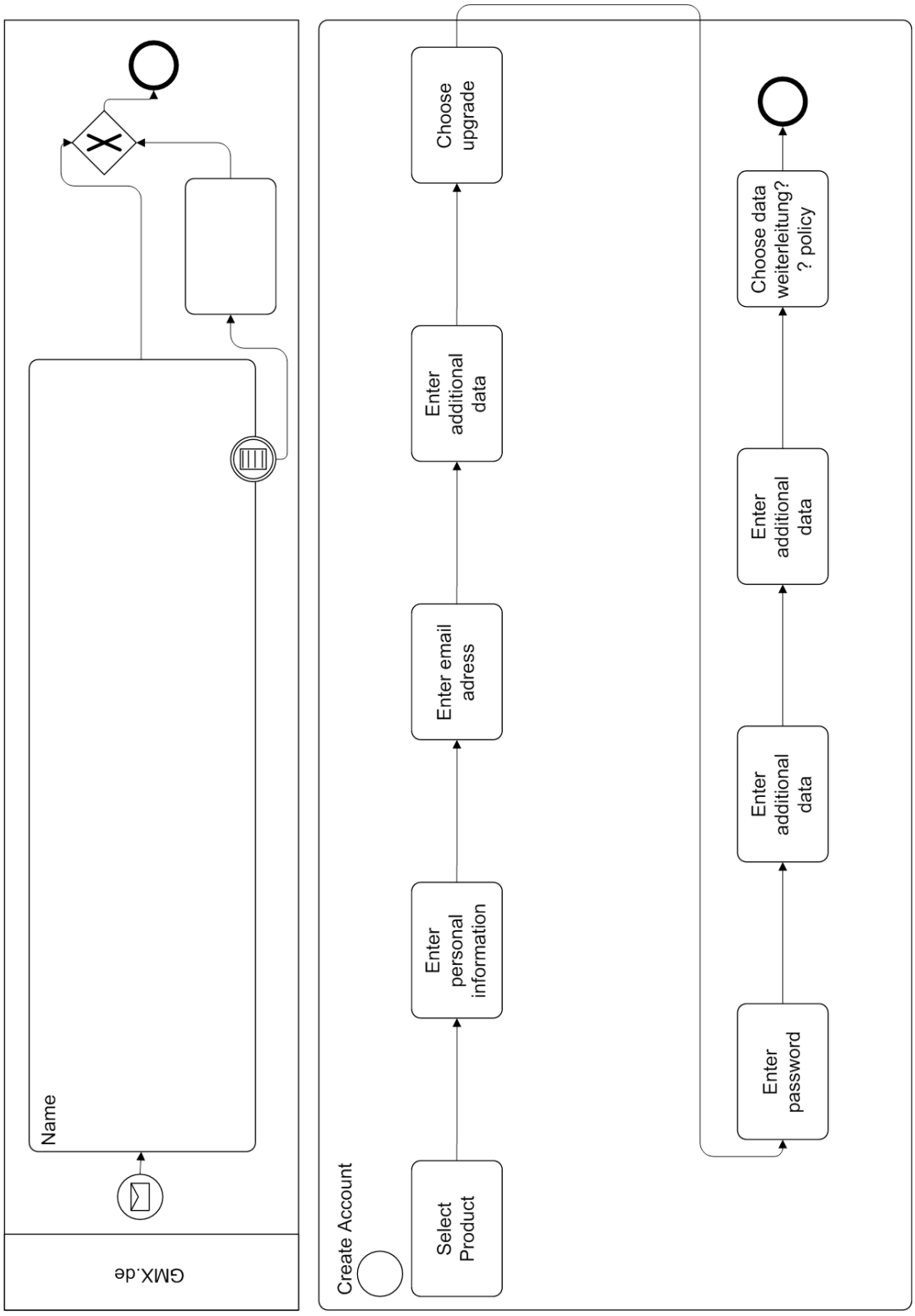


Abb. 5. Benutzerkonto erstellen, erster Versuch

Abb. 6. Benutzerkonto erstellen

GMX.de
Customer
Handle Session
Handle Emails
From the Login Process
Perform Action
Send Result
Remember Timeout
Session Timeout
Stop Session
Choose Action
More Actions
Quit
Send Logout

Abb. 7. Login

Abb. 8. Passwort zurücksetzen

Abb. 9. Passwort mit alternativer E-Mail zurücksetzen

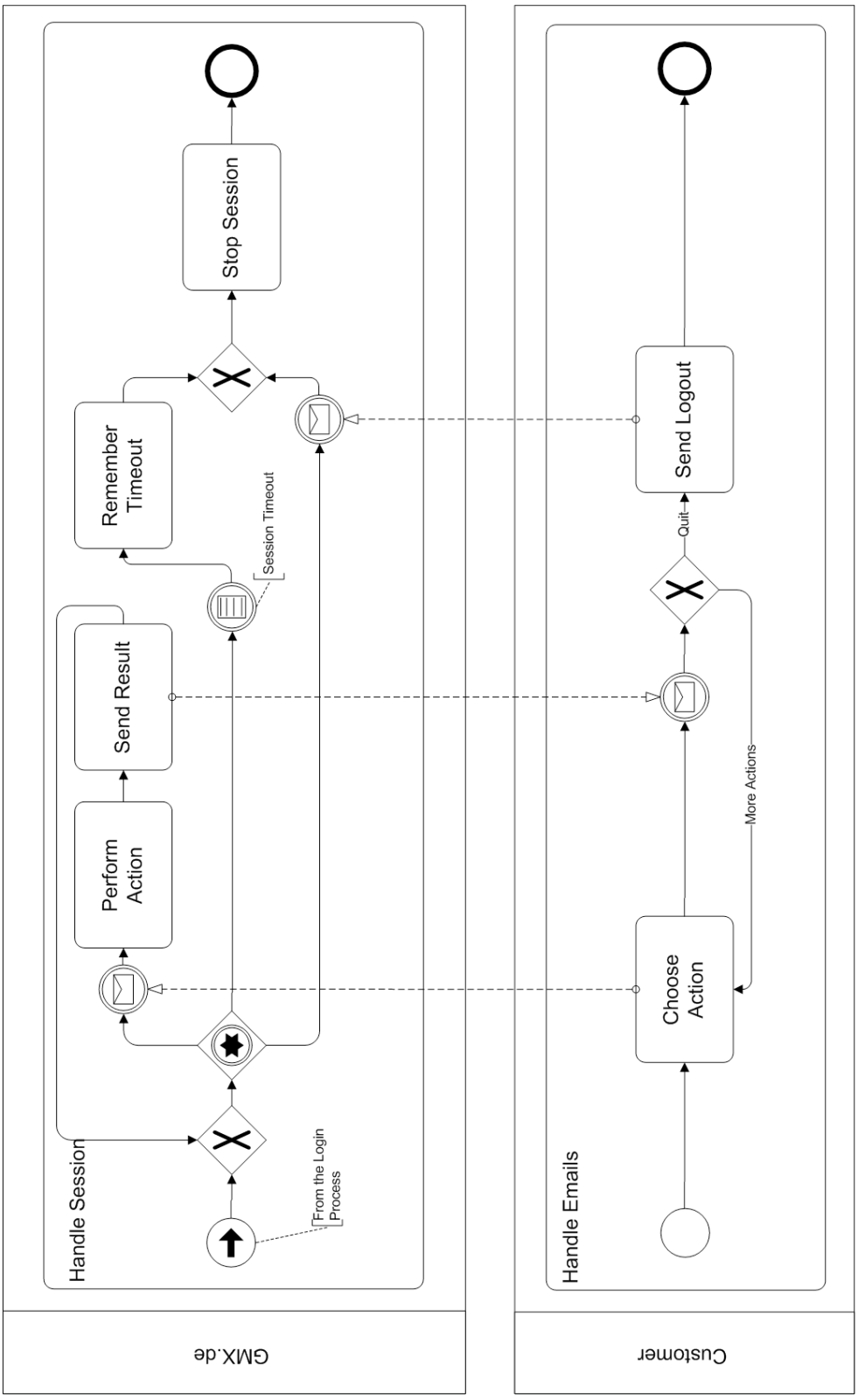


Abb. 10. E-Mail Dienste benutzen

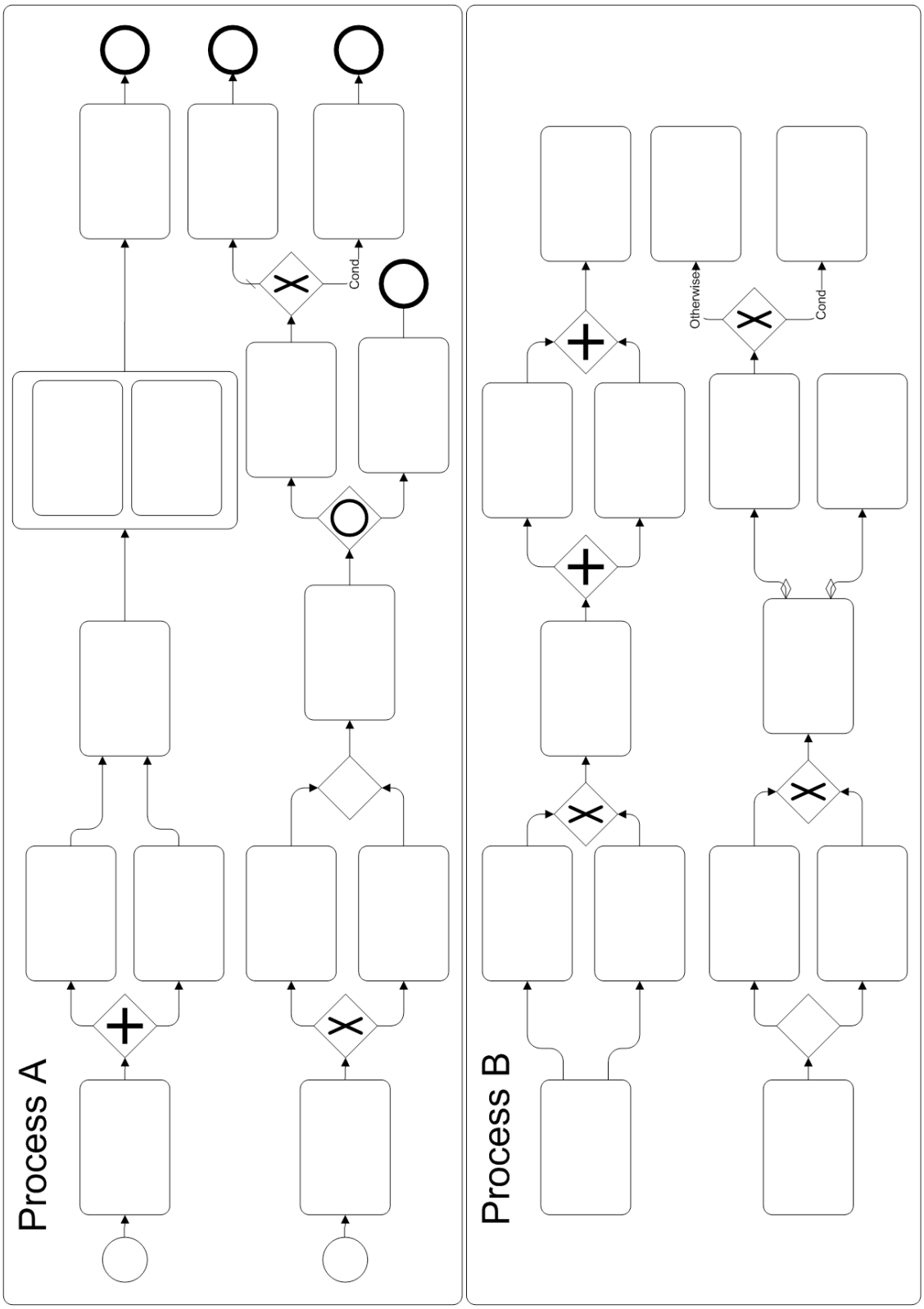


Abb. 11. Zwei semantisch identische Prozesse

Literatur

1. Business Process Management Initiative (BPMI): Business Process Modeling Notation (BPMN). Technical report, Business Process Management Initiative (BPMI) (2004) Only available online.
2. Stephen A. White: Process Modeling Notations and Workflow Patterns. (2004) Only available online.

Modellierung des Auskunfts- und Buchungssystems bahn.de

Florian Brodersen

florian.brodersen@hpi.uni-potsdam.de

Seminar Prozessmodellierung SS 2004
Hasso-Plattner-Institut für Software-Systemtechnik
Universität Potsdam

Zusammenfassung Das Auskunfts- und Buchungssystem der Deutschen Bahn bahn.de ermöglicht den Kunden nach Verbindungen zu suchen, Fahrkarten zu erwerben und Reservierungen durchzuführen. Außerdem ist es möglich, bereits erworbene Fahrkarten und Reservierungen zu stornieren. Diese Ausarbeitung zeigt, wie die Modellierungstechniken UML Use Cases, WebUml und BPMN zur Modellierung eines derartigen Systems genutzt werden können. UML Use Cases und WebUml werden zur Beschreibung der Anwendungsfälle und des Aufbau verwendet. Hauptgegenstand der Betrachtung ist die Modellierung der Geschäftsprozesse in BPMN.

1 Einleitung

Anhand des Auskunfts- und Buchungssystems der Deutschen Bahn bahn.de soll in der Ausarbeitung erarbeitet werden, ob und inwieweit die Notationen “UML Use Cases”, “WebUml” und “BPMN” zur Beschreibung derartiger Systeme geeignet sind.

Hierzu werden sowohl statische als auch dynamische Aspekte der Anwendung mit Hilfe der genannten Notationen beschrieben. Der Schwerpunkt der Modellierung liegt jedoch eindeutig auf der Darstellung der Prozesse. Daher werden die statischen Aspekte auch nur insoweit behandelt, wie sie zum Gesamtverständnis notwendig sind.

Überblick über die Kapitel

In Kapitel 1 werden anhand eines Use Case Diagramms die beteiligten Akteure, deren Rollen sowie ihre jeweiligen Anwendungsfälle ermittelt. Der Aufbau und die Komponenten des Systems werden in Kapitel 2 mit Hilfe der WebUml-Technik dargestellt.

Im Folgenden Kapitel 3 werden die dynamischen Aspekte betrachtet und auf BPMN-Diagramme abgebildet. Das letzte Kapitel fasst die gewonnenen Erkenntnisse zusammen und zeigt weitere mögliche Einsatzgebiete des Auskunfts- und Buchungssystems auf.

2 Anwendungsfälle

Um ein System wie bahn.de beschreiben zu können, ist es hilfreich, sich zunächst einen umfassenden Überblick über die vorhandenen Funktionalitäten zu verschaffen.

Hierzu wird im ersten Schritt analysiert, welche Akteure in dem betrachteten System eine Rolle spielen und auf welche Art und Weise sie das System nutzen. Diese Informationen lassen sich am besten durch einer Analyse der Anwendungsfälle und der Erstellung eines entsprechenden UML 2.0 Use Case Diagramms (OMG03), (CS03), (Lar01), (BHK03) gewinnen[1]. So kann auch festgestellt werden, wie die Umgebung des Systems aussieht und mit welchen anderen Systemen es interagiert.

2.1 Top-Level Use Case

Bei der Analyse lässt sich der Kunde als Rolle und der Zahlungsabwickler als Akteur identifizieren. Der Kunde stellt in diesem Zusammenhang den Hauptnutzer des Systems dar. Der Zahlungsabwickler hingegen repräsentiert eine angeschlossene Anwendung zur Abwicklung von Zahlungen mit Banken und Kreditkartenunternehmen. Darüber hinaus wickelt er auch das Gutschreiben von Beträgen und die Validierung von Kreditkarteninformationen ab.

Aufgrund der Funktionalitäten, die das Auskunfts- und Buchungssystems bahn.de dem Nutzer anbietet, lassen sich folgende fünf Anwendungsfälle ableiten:

- *Verbindung suchen:* Der häufigste Anwendungsfall repräsentiert die verschiedenen Suchmöglichkeiten, die ein Kunde nutzen kann, um eine bestimmte Reiseverbindungen zu ermitteln. Es existieren sowohl eine einfache als auch eine erweiterte Suche, die sich durch den Konfigurationsumfang der Suchparameter unterscheiden.
- *Profil verwalten:* Kunden sind in der Lage ein eigenes Profil anzulegen, dieses zu modifizieren und ggf. auch zu löschen.
- *Festpreisangebote kaufen:* Auf bahn.de werden auch Festpreisangebote wie z.B. bestimmten Ländertickets zum Online-Kauf angeboten.
- *Buchen:* Dieser Anwendungsfall beschreibt den Prozess zum Kaufen einer oder mehrerer Fahrkarte(n) inklusive einer optionalen Reservierung.
- *Stornieren:* Bestimmte bereits gekaufte Fahrkarten können online storniert werden.

Das Endergebnis der Analyse ist in dem Use Case Diagramm in Abbildung 1 dargestellt.

[1] Die Kenntnis von UML wird als bekannt angenommen. Entsprechende Literaturquellen sind im Text angegeben.

2.2 Use Case Template

Einführende Bemerkung: *Im weiteren Verlaufe dieses Abschnittes werden die Begriffe "Use Case" und "Geschäftsprozess" synonym verwendet, um von der Tatsache zu abstrahieren, dass ein Use Case die Repräsentation eines Geschäftsprozesses darstellt.*

Um eine detaillierte Beschreibung von Anwendungsfällen zu erhalten, hat Alistair Cockburn eine Methode namens Use Case Template (Coc98), (Coc00) entwickelt. Dabei werden Use Case Diagramme um eine textuelle Komponente ergänzt. Folgende Elemente[2] werden zusätzlich erfasst:

- *Ziel eines Use Cases:* Das Ziel eines Use Cases beschreibt die globale Zielsetzung bei erfolgreicher Ausführung eines Prozesses.
- *Beteiligte Akteure:* Mit beteiligten Akteure sind diejenigen Akteure gemeint, die an dem Geschäftsprozess beteiligt sind oder ihn auslösen.
- *Beschreibung des Standardfalls:* Eine Beschreibung des Standardfalls[3] ist insofern sinnvoll, als dass bei komplizierten Erweiterungs- und Einschlussbeziehungen[4] zwischen Geschäftsprozessen nicht klar ist, wie das häufigste Anwendungsszenario aussieht.
- *Variationen des Standardfalls:* Variationen des Standardfalls sind alle vom Standardfall abweichenden Anwendungsszenarien. Dieser Abschnitt ist häufig länger und komplexer als der des Standardfalls.
- *Vor- und/oder Nachbedingungen:* Um die Darstellung komplexer Prozesse einfach zu halten, kann mit Hilfe von Vor- und Nachbedingungen festgelegt werden, welcher Zustand vor der Ausführung und nach erfolgreicher Beendigung des Prozesses vorliegen muss.

Diese Use Case Templates werden jeweils in leicht modifizierter Form auch von anderen Autoren (Bal00), (Lar01) zur erweiterten Beschreibung von Use Case Diagrammen genutzt.

2.3 Beispiel

Im Folgenden wird der Anwendungsfall "Buchen" mit Hilfe des vorgestellten Use Case Templates beschrieben. Weitere Beispiele befinden sich im Anhang in Abschnitt A.

Es ist noch anzumerken, dass sich der Abschnitt *Beschreibung* aus den Abschnitten *Beschreibung des Standardfalls* und *Variationen des Standardfalls* zusammensetzt. Alternativen im Kontrollfluss sind durch Kleinbuchstaben gekennzeichnet.

[2] Eine vollständige Auflistung aller Elemente ist der Beschreibung des "Basic Use Case Template" (Coc98) von Cockburn zu entnehmen.

[3] In der Literatur wird hier vom sog. "main success scenario" (Coc98) oder auch vom "happy path" (Lar01) gesprochen.

[4] Damit sind die Abhängigkeiten gemeint, die im Use Case Diagramm durch die Schlüsselworte *«extends»* und *«includes»* gekennzeichnet sind.

Buchen

Ziel	Fahrkarte für gewünschte Verbindung erstellt (ggf. als Online-Ticket) und Zahlungstransaktion ordnungsgemäß verbucht
Vorbedingung	Kunde ist angemeldet und hat sich für eine bestimmte Fahrkarte und bestimmte Zugverbindung entschieden
Nachbedingung	Buchungsbestätigung per Email versendet Online-Ticket erzeugt oder Fahrkarte(n) zum Versand aufbereit
Akteure	Kunde Zahlungsabwickler
Beschreibung	1 Kundendaten prüfen 2a Überprüfung der Kreditkarteninformationen 2b Überprüfung der Bankverbindung 3 Zahlungsvorgang ausführen 4 Reservierung durchführen 5a Online-Ticket erstellen 5b Fahrkarte(n) zum Versand aufbereiten

2.4 Bewertung

Die Analyse hat gezeigt, dass Use Case Diagramme ein gutes Werkzeug darstellen, um eine initiale Sicht über die Geschäftsprozesse zu erhalten. Es kann ohne weiteres mit dem Kontextdiagramm, bekannt aus der strukturierten Analyse (SA) (You89), (Bal00), verglichen werden. Es muss allerdings die Einschränkung gemacht werden, dass bei UML nicht die Dekompositionsregeln existieren, wie es bei SA der Fall ist.

Problematisch bei der Erstellung von Use Case Diagrammen ist, eine adäquate Abstraktionsebene zu finden. Häufig gehen der finalen Version der Diagramme einige Iterationen vorweg, da nicht immer von Anfang an klar ist, welche Akteure und welche Anwendungsfälle tatsächlich für die Betrachtung relevant sind.

Zur detaillierten Spezifikation komplexer Use Case Diagramme eignet sich das vorgestellte Basic Use Case Template sehr gut, da es die grafische Notation um eine Beschreibung von Sequenzen sowie Alternativen erweitert. Vor allem die Herausarbeitung des Standardfalls erleichtert zudem das Verständnis derartiger Diagramme.

3 Modellierung statischer Aspekte

Nachdem im ersten Schritt anhand von Use Cases ein Grundverständnis für das System entwickelt werden konnte, können nun im zweiten Schritt die statischen Aspekte des Auskunfts- und Buchungssystem der Deutschen Bahn bahn.de mit Hilfe der WebUml-Notation, vorgestellt von Bellettini (BMT04), modelliert werden.

WebUml ist ein werkzeug-gestütztes Verfahren zum *Reverse Engineering von Web Applikationen*. Durch Quellcodeanalyse und andere Verfahren wird ein Abbild des Aufbaus und des Verhaltens einer Anwendung generiert. Die statischen Aspekte werden in einem UML Objektdiagramm dargestellt, während die dynamischen Aspekte auf ein UML Zustandsdiagramm abgebildet werden. Letzteres wird in dieser Ausarbeitung nicht berücksichtigt, da die dynamischen in Aspekte Kapitel 4 weiter unten mit Hilfe von BPMN beschrieben werden.

Um den Aufbau einer beliebigen Web-Anwendung modellieren zu können, haben die Entwickler von WebUml ein Metamodell erstellt, aus dem das Objektdiagramm für eine konkrete Anwendung abgeleitet werden kann (Bildung einer Instanz). Im Folgenden wird eine leicht angepasste Version dieses Modells vorgestellt.

3.1 Vorstellung Metamodell

Das Metamodell besteht aus folgenden Klassen:

- WebDocument
- ServerPage
- ClientPage
- Form
- Frame
- Session
- Script
- WindowOpen

Der Kern einer jeden Web Applikation ist das *WebDocument*; dies kann sowohl eine statische Seite (*ClientPage*) als auch eine dynamisch-generierte Seite (*ServerPage*) sein.

Eine *ClientPage* kann Strukturelemente wie Schaltflächen, Eingabefelder, usw. sowie beliebigen client-seitigen Skriptcode enthalten (*Script*). Jede ClientPage ist weiterhin mit einem *Frame* verbunden, der auf weitere Frames oder ein anderes WebDocument verweisen kann. Popups und vergleichbare - mit Hilfe von Skriptcode erzeugte - Fenster werden durch *WindowOpen* repräsentiert.

Eine *ServerPage* kann Daten einer Web-Maske (*Form*) erhalten und dynamisch eine ClientPage erzeugen. Cookies und andere Daten, die über mehrere Anfragen hinweg vorgehalten werden sollen, können in der *Session* gespeichert werden.

Das angepasste WebUml Metamodell ist in Abbildung 2 dargestellt. In den beiden folgenden Abschnitten werden die WebUml Objektdiagramme für den Geschäftsprozess “Suche” (Abbildung 3) und Geschäftsprozess “Buchen” (Abbildung 4) beschrieben.

3.2 Suche

Beim Aufruf *ServerPage_query.exe* hat der Kunde die Wahl zwischen der einfachen Suche (*ClientPage_Suche*) und der erweiterte Suche (*ClientPage_ErweiterteSuche*).

Je nach dem, für welche Variante er sich entschieden hat, wird der Inhalt des entsprechenden Formulars (*Form_Suche* oder *Form_ErweiterteSuche*) an die ServerPage_query.exe geschickt. Diese bereitet die Suchergebnisse dann in der *ClientPage_Suchergebnisse* auf und der Kunde kann auswählen, für welche Verbindung er eine Fahrkarte kaufen möchte.

Seine Auswahl geschieht in dem Formular *Form_Suchergebnisse*, welches wiederum an die ServerPage_query.exe gesendet wird. Falls der Kunde sich zu diesem Zeitpunkt noch nicht gegenüber dem System authentifiziert hat, wird ihm eine Seite *ClientPage_Login* und ein Formular *Form_Login* präsentiert um sich anzumelden.

3.3 Buchen

Nachdem die Anmeldedaten des Kunden an die *ServerPage_hafas.post* gesendet wurden, erzeugt diese eine *ClientPage_Buchungswünsche* und ein entsprechendes Formular *Form_Buchungswünsche*, um die Buchungswünsche des Kunden entgegen zu nehmen. Darunter fallen die Art des Tickets, der Bezahlung und des Ticketversandes.

Außerdem kann angegeben werden, ob eine Reservierung gewünscht wird. Ist dies der Fall, wird von der *ServerPage_buchen.post* eine *ClientPage_Reservierungswünsche* und das entsprechende Formular *Form_Reservierungswünsche* erzeugt.

Hat der Kunde seine Reservierungswünsche geäußert, wird von der *ServerPage_res.post* die *ClientPage_Kasse* und das dazugehörige Formular *Form_Kasse* erzeugt. Dort werden die Zahlungsdetails (Lastschrift oder Kreditkartenabbuchung) und wahlweise die Lieferadresse oder die Daten, die zur Authentifizierung des Online-Tickets notwendig sind, eingegeben. Der Inhalt der Formulars wird abschließend an die *ServerPage_kasse.post* gesendet. Damit ist der Buchungsprozess beendet.

3.4 Bewertung

WebUml erweist sich für die Analyse statischer Aspekte eines Systems als nur eingeschränkt geeignet. Es entsteht eine enorme Datenmenge, die die Diagramme unübersichtlich werden lässt. Dazu kommt noch, dass sich alle generierten Informationen auf derselben Ebene befinden und somit keine automatische Aggregation stattfindet.

Grundsätzlich jedoch ermöglicht diese Notation eine gute Darstellung des Aufbaus und der Komponenten einer Web-Anwendung. Leider stand bei der Evaluierung dieser Technik das im dem Artikel vorgestellte Werkzeug nicht zur Verfügung, weshalb die vorgestellten Diagramme manuell erstellt wurden.

Die generierten Artefakte bieten eine gute Grundlage für die weitere Prozessmodellierung. Darüber hinaus können vorhandene Web-Applikationen nachträglich dokumentiert werden.

4 Modellierung dynamischer Aspekte

Nach der Beschreibung der statischen Aspekte der Anwendung mit Hilfe von WebUML geht es in diesem Abschnitt um die Modellierung der dynamischen Aspekte.

Für die Darstellung der dynamischen Aspekte des Auskunft- und Buchungssystem der Deutsche Bahn wird die von der Business Process Management Initiative (BPMI) definierte Business Process Modeling Notation (BPMN) (Whi04a), (Whi04b) verwendet.

Bei der Modellierung wird der Teil der beim Kunden ablaufenden Geschäftsprozesse durch einen sogenannten abstrakten Prozess “Abstract Process” (Whi04a) repräsentiert und läuft in einem separaten Pool ab.

Diese Entscheidung wurde getroffen, um die Komplexität der Diagramme zu begrenzen und um hervorzuheben, dass die Umsetzung der Anwendungslogik auf Seiten der Deutschen Bahn zu finden ist. Der Kunde interagiert nur mit dem System und steuert es nach seinen Wünschen.

Insgesamt wird die Interaktion zwischen Kunde, Zahlungsabwickler und dem Auskunfts- und Buchungssystem der Deutschen Bahn bahn.de in einem sog. “Collaboration Process” (Whi04a) dargestellt.

Im Folgenden Abschnitt wird das BPMN Business Process Diagramm Geschäftsprozess “Stornieren” (Abbildung 6) beschrieben. Ein weiteres BPMN Business Process Diagramm, welches den Geschäftsprozess “Buchen” (Abbildung 5) beschreibt, befindet sich im Anhang in Abschnitt C.

4.1 Stornieren

Um eine Buchung stornieren zu können, muss der Kunde zunächst eine Buchung aus der Buchungsrückschau auswählen. Seine Auswahl wird vom Auskunfts- und Buchungssystem der Deutschen Bahn bahn.de entgegen genommen und daraufhin überprüft, ob der erste Geltungstag der zu der Buchung gehörenden Fahrkarte bereits abgelaufen ist. Falls dies der Fall ist, wird dem Kunden die Möglichkeit gegeben, einen Erstattungsantrag herunterzuladen und diesen auf dem Postwege der Deutschen Bahn zukommen zu lassen. Der Prozess ist damit beendet.

Ist dies nicht der Fall, wird der Prozess fortgesetzt und ggf. reservierte Plätze wieder freigegeben. Außerdem wird in Abhängigkeit der beim Erwerb der Fahrkarte angegebenen Zahlungsweise (Lastschrift oder Kreditkartenabbuchung) der entsprechende Betrag entweder auf das Bankkonto des Kunden oder seiner Kreditkarte gutgeschrieben. Abschließend wird noch eine Stornierungsbestätigung per Mail an den Kunden versendet.

4.2 Bewertung

Obwohl BPMN zum Zeitpunkt der Bearbeitung erst in Version 1.0 vorliegt und es nur einige wenige Publikationen (Whi04a), (Whi04b), (Whi04c) gibt, macht diese Notation schon einen sehr durchdachten Eindruck. Gerade die Modellierung von Prozessen aus der E-Business-Welt scheint ein gutes Einsatzgebiet für BPMN zu sein.

Um diese Sprache einer breiten Masse an die Hand zu geben, mangelt es noch an entsprechenden Werkzeugen zur einfachen Erstellung von Business Process Diagrammen. Dazu gehört sicherlich auch die automatische Überprüfung der hinreichend komplexen Syntaxregeln.

5 Fazit

Die Modellierung des Auskunfts- und Buchungssystem der Deutschen Bahn bahn.de hat gezeigt, dass sich die verwendeten Techniken insgesamt gut für eine Beschreibung derartiger Systeme eignen. Speziell BPMN hat ein großes Potential andere Sprachen in der Prozessmodellierung zu verdrängen.

Bei der Betrachtung der erstandenen Diagramme fällt auf, dass das betrachtete System eine große Ähnlichkeit mit einem Shopsystem hat. Dabei entspricht die Verbindungssuche der Artikelsuche und der Kauf von Fahrkarten lässt sich auch auf den Kauf eines Artikel abbilden. Lediglich die Reservierung von Plätzen ist hinzugekommen, wobei man eine Reservierung auch als einen Artikel betrachten könnte.

Außerdem ist den Business Process Diagrammen (Abbildungen 5, 6) zu entnehmen, dass sich der gesamte Prozess, der hinter dem Auskunfts- und Buchungssystem steckt, leicht an neue Anforderungen anpassen lässt. Vorstellbar ist beispielsweise eine Anbindung des Systems an Fahrkartenautomaten, Reisebüros oder auch an mobile Anwendungen.

Die Voraussetzungen für eine derartige Anpassung sind schon alleine dadurch gegeben, dass die Schnittstelle zwischen Kunde und bahn.de einfach gehalten ist und das Auskunftssystem tatsächlich physikalisch von dem Buchungssystem getrennt ist.

A Anwendungsfälle

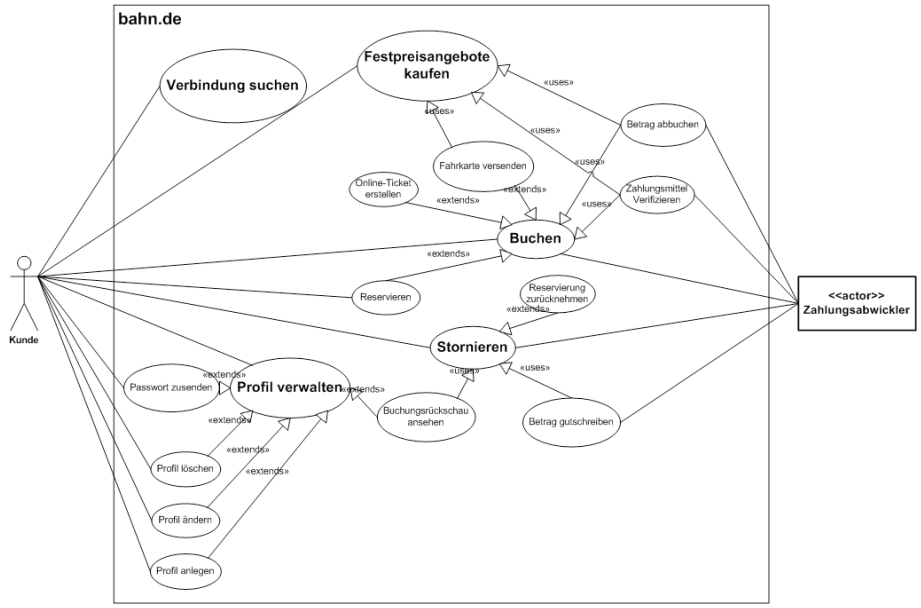


Abbildung 1. Top-Level Use Case Diagramm

Festpreisangebote kaufen

Ziel	Zahlungstransaktion für gewünschtes Festpreisangebot ordnungsgemäß abgeschlossen und Fahrkarte für Versand zum Kunden aufbereitet
Vorbedingung	Kunde ist angemeldet
Nachbedingung	Buchungsbestätigung per Email versendet Fahrkarte zum Versand aufbereit
Akteure	Kunde Zahlungsabwickler
Beschreibung	1 Kundendaten prüfen 2 Überprüfung der Kreditkarteninformationen 3 Zahlungsvorgang ausführen 4 Fahrkarte zum Versand aufbereiten

Stornieren

Ziel	Gewünschte Buchung ist storniert
Vorbedingung	Kunde ist angemeldet
Nachbedingung	Stornierungsbestätigung per Email versendet Betrag dem Kunden gutgeschrieben
Akteure	Kunde Zahlungsabwickler
Beschreibung	1 Buchungsrückschau ansehen und eine Buchung zum Stornieren auswählen 2 Buchungsdaten für gewünschte Buchung abrufen 3 Reservierte(n) Platz/Plätze freigeben 4a Betrag auf Kreditkarte gutschreiben 4b Betrag auf Bankkonto gutschreiben

Profil verwalten

Ziel	Gewünschte Änderungen am Profil durchgeführt
Vorbedingung	Für 1b und 1c: Kunde ist angemeldet
Nachbedingung	Für 1a und 1d: Kunde kann Buchungssystem nutzen Für 1c: Kunde kann Buchungssystem nicht mehr nutzen
Akteure	Kunde
Beschreibung	1a Profil anlegen um Neukunden zu erfassen 1b Profil ändern 1c Profil deaktivieren und Kundendaten zum Löschen markieren 1d Passwort zusenden

B WebUml Diagramme

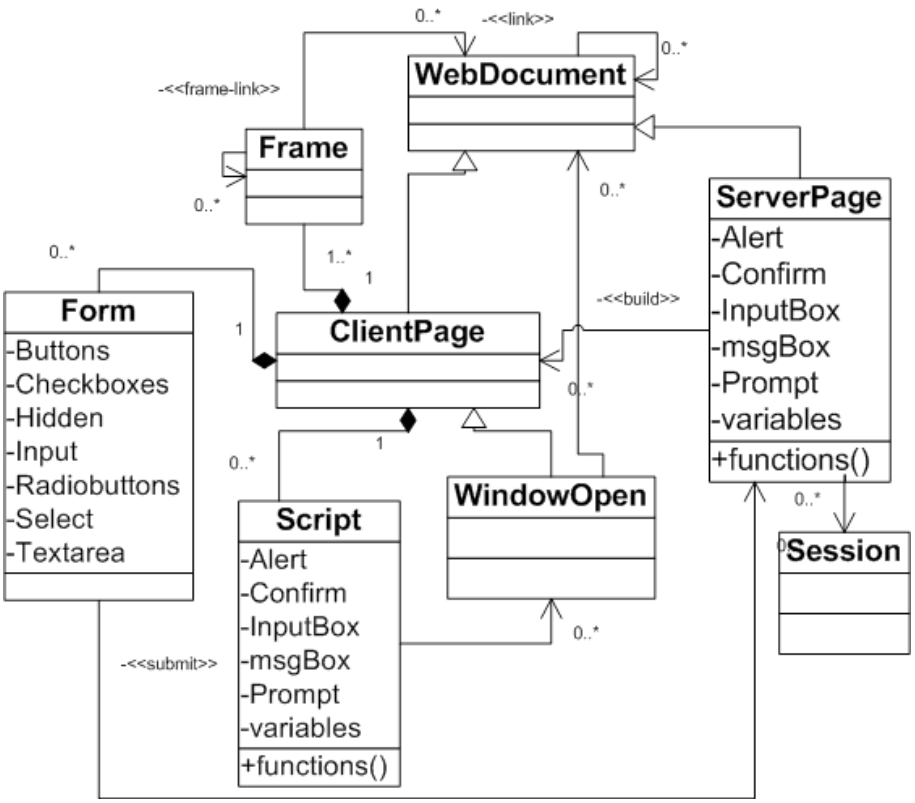


Abbildung 2. WebUml Meta Modell

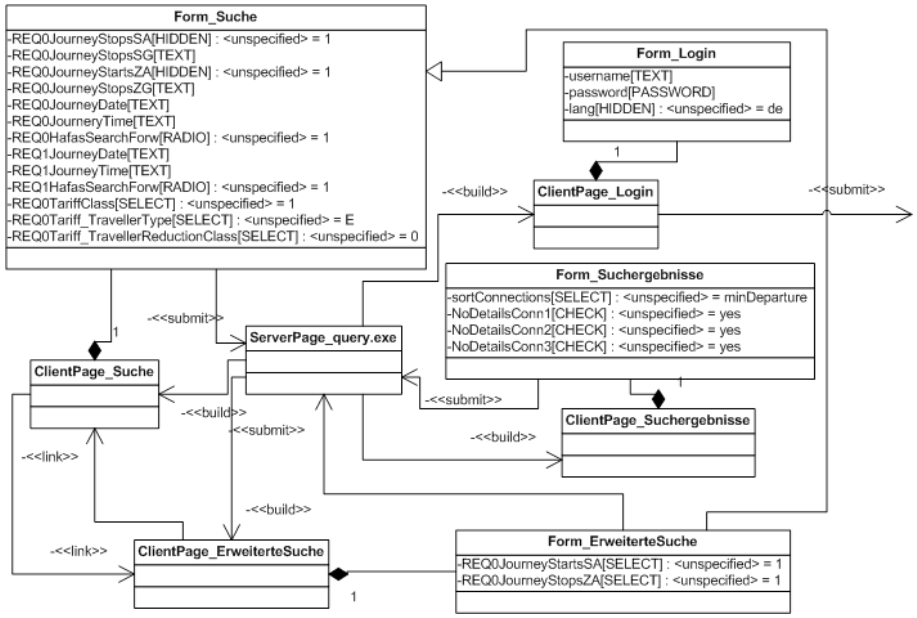


Abbildung 3. Suche

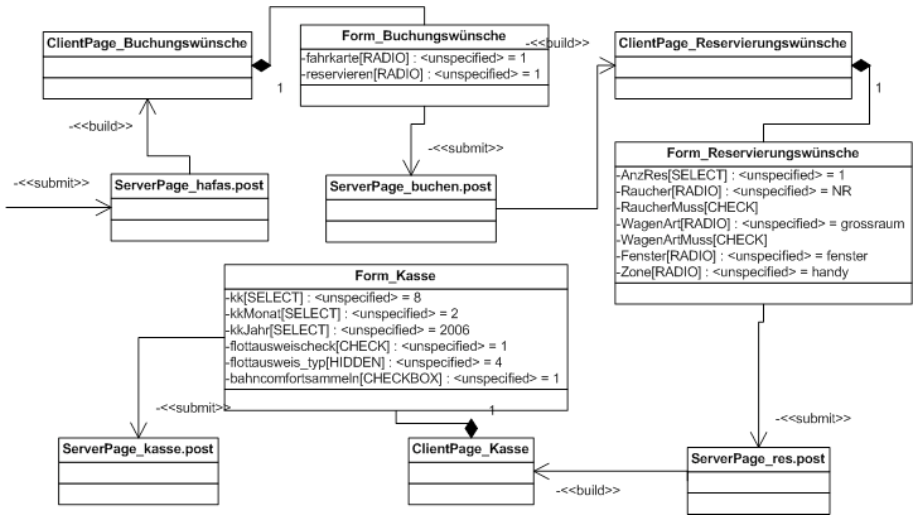


Abbildung 4. Buchen

C Business Process Diagramme

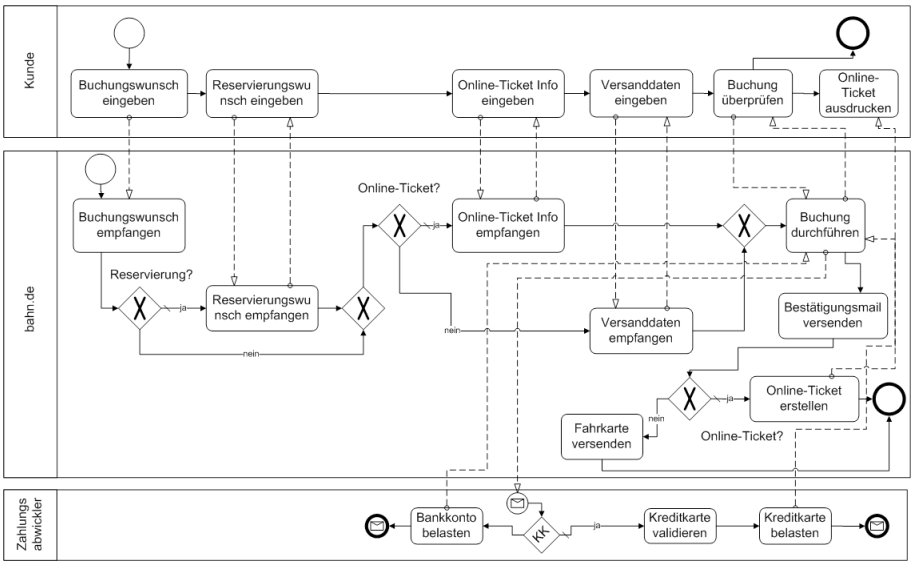


Abbildung 5. Buchen

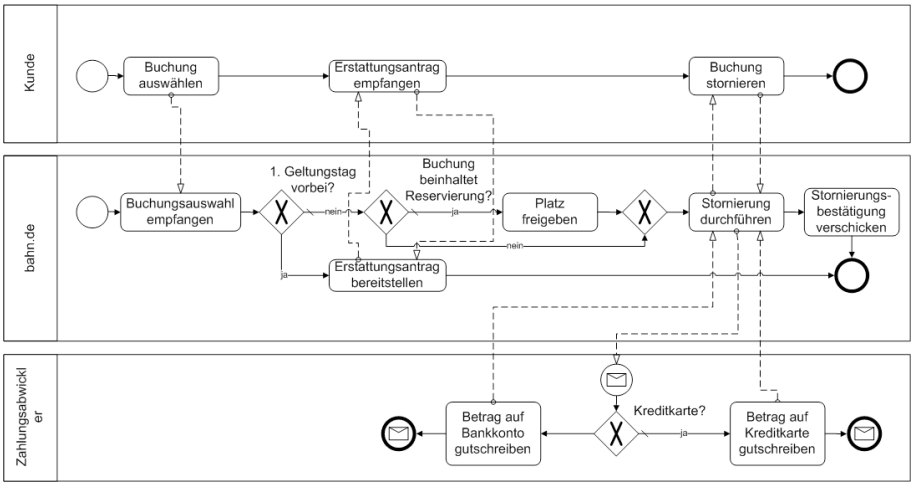


Abbildung 6. Stornieren

Literaturverzeichnis

[Bal00] BALZERT, Helmut: *Lehrbuch der Software-Technik*. 2. Spektrum, 2000

[BHK03] BORN, Max ; HOLZ, Echhardt ; KATH, Olaf: *Softwareentwicklung mit UML 2*. Addison-Wesley, 2003

[BMT04] BELLETTINI, Carlo ; MARCHETTO, Alessandro ; TRENTINI, Andrea: WebUml: Reverse Engineering of Web Applications. In: *Proceedings of the 2004 ACM Symposium of Applied Computing*, 2004

[Coc98] COCKBURN, Alistair. *Basic Use Case Template*. 1998

[Coc00] COCKBURN, Alistair: *Writing Effective Use Cases*. Addison-Wesley Professional, 2000

[CS03] CHONOLES, Michael J. ; SCHARDT, James A.: *UML 2 für Dummies*. Mitp-Verlag, 2003

[Lar01] LARMAN, Craig: *Applying UML and Patterns*. 2. Prentice Hall, 2001

[OMG03] Object Management Group (OMG): *Unified Modeling Language: superstructure. Version 2.0*. 2003

[Whi04a] WHITE, Stephen A. *Business Process Modeling Notation (BPMN)*. 2004

[Whi04b] WHITE, Stephen A. *Introduction to BPMN*. May 2004

[Whi04c] WHITE, Stephen A. *Process Modeling Notations and Workflow Patterns*. January 2004

[You89] YOURDON, Edward: *Modern Structured Analysis*. Prentice Hall, 1989

Der Softwareentwicklungsprozess von SAP Berlin

Gero Decker

Universität Potsdam
Hasso-Plattner-Institut für Softwaresystemtechnik
gero.decker@hpi.uni-potsdam.de

Zusammenfassung. Der Softwareentwicklungsprozesses von SAP in Berlin (Abteilung Netweaver Kernel MaxDB & liveCache) wird unter Verwendung der BPMN (Business Process Modeling Notation) analysiert und präsentiert. Die Nützlichkeit der BPMN für diese Anwendungsdomäne wird gezeigt und ein Vergleich mit UML 2.0 Activity Diagrams wird angestellt.

1 Einführung

Heutzutage werden Softwaresysteme immer größer und ihre Komplexität ist eine Herausforderung für Softwareingenieure. Deshalb ist Forschung im Bereich von Entwicklungsprozessen von Nöten.

Der Autor hat sich konkret mit dem Entwicklungsprozess der Berliner Entwicklungsabteilung von SAP beschäftigt. Da dieser Entwicklungsprozess den Kerngeschäftsprozess der Abteilung darstellt, liegt die Vermutung nahe, dass er sich mit Geschäftsprozessmodellierungstechniken darstellen lässt.

Im Rahmen des Seminars Prozessmodellierung wurde die BPMN (Business Process Modeling Notation) in den Fokus des Interesses gestellt. Aus diesem Grund wird sie in dieser Ausarbeitung verwendet und es wird der Frage nach ihrer Nützlichkeit nachgegangen. Interessant wird sein, welche Aspekte von BPMN sich als am wichtigsten erweisen und wie schnell BPMN-Neulinge sich darin einarbeiten können.

Im folgenden Kapitel wird SAP's Entwicklungsabteilung in Berlin kurz vorgestellt, bevor in Kapitel 3 die Vorgehensweise des Autors beschrieben wird. Kapitel 4 beschäftigt sich mit der Ist-Situation des Entwicklungsprozesses, wohingegen Kapitel 5 einige darauf aufsetzende Verbesserungsvorschläge aufzeigt. Kapitel 6 enthält einen kurzen Vergleich zwischen BPMN und UML 2.0 Activity Diagrams. Kapitel 7 weist auf ein grundsätzliches Problem bei der Modellierung mit multiplen Instanzen hin. Abschließend wird eine Bewertung von BPMN und der speziellen Anwendung auf den Softwareentwicklungsprozess getätigt.

2 Vorstellung von SAP Berlin

SAP ist ein großes, weltweit operierendes Softwareunternehmen für betriebswirtschaftliche Software. Neben der Zentrale in Walldorf (Deutschland) unterhält es eine Vielzahl von Niederlassungen in der ganzen Welt.

Die Entwicklungsabteilung in Berlin hat ein Hauptprodukt: Das Datenbanksystem MaxDB (ehemals SAP DB). Die Wurzeln dieses Systems können datiert werden auf die späten 70er und sind eng verbunden mit einer Ausgründung der TU Berlin. Seit 1997 gehört diese Abteilung zu SAP. MaxDB wird hauptsächlich verwendet für das R/3 System und neuerdings für die Technologieplattform Netweaver.

Im Jahre 2000 wurde MaxDB als Open Source freigegeben. Zwei Jahre später folgte eine strategische Partnerschaft mit dem Datenbanksystemhersteller MySQL, der seitdem als weiterer Absatzkanal fungiert.

2.1 Die Berliner Abteilung und ihre Umwelt

Eine Einordnung der Berliner Entwicklungsabteilung ist in Abb. 1 dargestellt. Als Notation für dieses Diagramm wurde FMC [1] verwendet.

In der Abbildung sind die Akteure (eckige Bildelemente) dargestellt, mit denen die Abteilung hauptsächlich zu tun hat: Der Netweaver-Koordination, das Unternehmen MySQL, sowie SAP-Kunden. Die runden Bildelemente repräsentieren Kanäle und sonstige Aktionsfelder, über die die Akteure in Interaktion stehen.

Organisatorisch ist die Abteilung in den Bereich Netweaver-Kern-Entwicklung eingeordnet, welcher wiederum zum Vorstandsbereich Entwicklung gehört. Die Technologieplattform Netweaver bildet die Basis für alle neueren SAP-Systeme. Um einen reibungslosen Betrieb der MaxDB innerhalb des Netweavers sicherzustellen, bedarf es bei Integration der MaxDB Synchronisation und Kommunikation zwischen der Netweaver-Koordination und der Berliner Abteilung.

Die Kunden, die SAP-Systeme einkaufen, schließen normalerweise auch gleichzeitig Support-Verträge mit SAP ab. Der Support speziell für MaxDB wird dabei in Berlin geleistet. Anfragen von Kunden werden über das sog. Customer Services Network (CSN) an die Abteilung in Berlin weitergeleitet.

Die MaxDB wird auch als Open Source System veröffentlicht. Hierdurch können es Benutzer kostenlos beziehen und einsetzen. Falls in diesem Fall doch Supportleistungen benötigt werden, stellt MySQL diese bereit.

2.2 Aufbau der Abteilung

Um die in Kapitel 4 vorgestellten Prozesse zu verstehen, werden nun die Akteure innerhalb der Abteilung vorgestellt (siehe Abb. 2).

Die Mitarbeiter der Abteilung lassen sich in folgende Teams ordnen: Entwicklungsteams (Kernel, Interfaces, Tools), Infrastructure, Q-Team, Support und Dokumentation. Sämtliche Teamleiter sind darüber hinaus im Management angesiedelt.

Zusätzlich zu diesen organisatorischen Einheiten sind in Abb. 2 der Quellcode, Testfälle und Problem- und Featurebeschreibungen repräsentiert. Der Quelltext wird

regelmäßig als Produkt ausgeliefert, während die Testfälle für interne Qualitätssicherung verwendet werden. Die Problem- und Featurebeschreibungen werden im sog. „Problem-Tracking-System“ verwaltet und dokumentieren die Bearbeitung von Fehlern und die Umsetzung neuer Produktfunktionen.

Die Entwicklungsteams stellen den Quellcode, Integrationstests und Dokumentation her, wohingegen das Q-Team Systemtests definiert und die Gesamtqualität des Produktes überwacht. Das Infrastructure Team stellt die technische Infrastruktur für die Entwicklung bereit (z.B. Computerfarmen und Entwicklungswerkzeuge). Das Support Team bearbeitet Nachrichten von Kunden und versucht das jeweilige Problem zu lösen. Das Dokumentations-Team sammelt und überarbeitet technische Dokumentationen, die für das spätere Benutzerhandbuch verwendet werden. Das Management beschäftigt sich mit sehr grundlegenden technischen Aspekten. Außerdem steuert es den Prozess und kümmert sich um Personalfragen.

3 Vorgehensweise

Angelehnt an [5] wurde nach der Informationserhebung zunächst eine Ist-Modellierung durchgeführt, bevor schließlich Optimierungsvorschläge gesammelt wurden.

3.1 Informationserhebung

Bei SAP existieren unternehmensweite Richtlinien für Entwicklungsprozesse, die im SAPNet, SAP’s Intranet, zu finden sind. Jedoch ist diese Quelle nicht ausreichend gewesen für diese Ausarbeitung: Da die Abteilung in Berlin nicht von SAP gegründet wurde, sondern – wie bereits erwähnt – aufgekauft wurde, hat der alte Prozess in gewissem Umfang diesen Wechsel überlebt. Der Prozess wurde vor drei Jahren durch SAP überarbeitet und optimiert. Zu dieser Prozessoptimierung war jedoch leider keine Dokumentation zu finden und somit blieben Interviews mit Mitarbeitern die einzige Quelle für eine Analyse.

Ursprünglich war es geplant, Informationserhebungstechniken zu verwenden, wie sie in [5] vorgeschlagen werden. In diesem Buch wird jedoch die Annahme gemacht, dass in Geschäftsprozesse eingebundene Benutzer kaum oder gar keine Kenntnisse über die Gesamtstruktur des Geschäftsprozesses haben. Diese Annahme ist nicht erfüllt in unserem Fall: Vor allem die Entwickler sind in fast allen Aktivitäten beteiligt oder sind zumindest betroffen. Regelmäßige Treffen und die Verteilung von Protokollen garantieren einen gewissen Überblick. Aus diesem Grund war es ausreichend, nur eine kleine Zahl von Mitarbeitern zu interviewen.

3.2 Ist-Modellierung

In dieser Phase wurden die gesammelten Informationen in ein Prozessmodell übersetzt. Hierzu wurde BPMN verwendet.

Zunächst wurde der SAP-weite Entwicklungsprozess für den Netweaver in Beziehung gesetzt mit dem lokalen Prozess in Berlin. Ausgehend von diesem „big picture“ fand eine Top-Down-Modellierung statt.

Für eine möglichst zutreffende Prozessmodellierung war ein iteratives Vorgehen von Nöten. Dies bedeutet, dass die Zwischenergebnisse dieser Modellierungsphase zwecks Validierung immer wieder mit den Mitarbeitern diskutiert wurden. Die Informationserhebungsphase und die Modellierungsphase waren also nicht streng voneinander getrennt sondern wechselten sich gegenseitig ab.

Ein interessanter Aspekt dabei war, wie die Mitarbeiter mit BPMN als Beschreibungssprache zurechtkommen: Wie viel verstehen sie ohne weitere Erklärungen und wie lange dauert es zu erklären, was mit den Modellen ausgesagt werden soll? Wie genau schauen sich die Mitarbeiter die Modelle an oder geben sie sich damit zufrieden, die Hauptaspekte zu überblicken? Lassen sie sich auf die BPMN ein, um ihre eigenen Vorschläge zu formulieren?

3.3 Prozessoptimierung

Bereits während der Informationserhebung tauchten zahlreiche Vorschläge für Optimierungen der täglichen Arbeit auf. Meistens handelte es sich dabei um kleinere Ideen, die nicht die Veränderung des Prozesses vorsahen. Z.B. bestand der Wunsch nach dem einen oder anderen Tool, das die täglichen Entwicklertätigkeiten erleichtern sollten. Die grundlegendsten Optimierungsvorschläge sind in Kapitel 5 gesammelt.

4 Der Entwicklungsprozess

Nachdem im zweiten Kapitel die statische Struktur der Abteilung vorgestellt wurde, werden im Folgenden die Abläufe, d.h. die dynamische Struktur, beschrieben. Sämtliche Ablaufdiagramme sind im Anhang zu finden (Abb. 3 bis 8). Zum Verständnis der weiteren Kapitel sind grundlegende Kenntnisse bezüglich BPMN unerlässlich. Als einführende Lektüre eignet sich „Introduction to BPMN“ [9].

4.1 Einordnung des Entwicklungsprozesses

Abb. 3 ist auf dem gleichen Abstraktionsniveau angesiedelt wie schon Abb. 1. Das Diagramm zeigt die Einordnung der Aktivitäten der Berliner Entwicklungsabteilung in den SAP-weiten Netweaver-Entwicklungsprozess. Die in den einzelnen Pools dargestellten Akteure sind bereits aus Abb.1 bekannt. Obwohl die Berliner Abteilung und die Netweaver-Koordination im gleichen Unternehmen beheimatet sind, wurde hier die Darstellung zweier getrennter Pools vorgenommen. Dies ist dadurch zu rechtfertigen, dass die beiden Akteure relativ getrennt voneinander arbeiten.

Der SAP-weite Netweaver-Entwicklungsprozess kann in die dargestellten fünf Aktivitäten aufgeteilt werden. Beim Planning werden Anforderungen aufgestellt und Planungen für den weiteren Verlauf erarbeitet. Die Development-Phase umfasst den Entwurf, Implementierung und Test pro Netweaver-Komponente. Bevor dann Integ-

rationstests auf Netweaver-Ebene (sog. Szenariotests) angefangen werden können, müssen alle Komponenten vorliegen. Anschließend kann die Assembly & Validation Phase beginnen, in der die Komponenten zu den auszuliefernden Systemen endgültig montiert und letzte Systemtests durchgeführt werden. Danach wird das Netweaver-System in die Ramp-Up-Phase überführt: Hierbei handelt sich um die Einführung des Systems bei Pilotkunden. Es wird unter realen Bedingungen („produktiv“) beobachtet und eventuelle Unstimmigkeiten werden noch beseitigt.

Auf der Berliner Seite sind zwei Hauptprozesse zu identifizieren: Entwicklung und Support. Ersterer beschäftigt den Großteil der Mitarbeiter der Abteilung.

Bereits an dieser Stelle trat eine wesentliche Frage bei der Modellierung auf: Mit welchen Ereignissen beginnt und endet der Entwicklungsprozess? Der Autor hat sich dafür entschieden, das stark iterative Vorgehen der Abteilung dadurch abzubilden, dass pro geplantem Release eine Prozessinstanz gestartet wird. Somit werden sämtliche in der Entwicklung anfallende Tasks genau einem Release-Ziel („Release-Target“) zugeordnet, welches durch eine Release-Nummer eindeutig identifiziert wird. Diese Zuordnung ist dabei nicht willkürlich, sondern entscheidet sich danach, in welchem Release ein Task (z.B. Programmierung einer bestimmten Funktion) zum ersten Mal Auswirkungen haben wird.

Zusätzlich zu einem normalen Release-Ziel, welches in regelmäßigen Abständen (also mehr oder weniger zeitgesteuert) samt ungefährem Release-Termin definiert wird, kann ein sog. Hot-Fix-Request auftreten: Falls ein schwerwiegender Fehler des bereits ausgelieferten Produkts festgestellt wurde, wird eine Prozessinstanz initiiert, deren Ziel es ist, den Fehler schnellstmöglich zu beheben und eine entsprechende neue Version auszuliefern.

Neben dem Entwicklungsprozess ist bei SAP in Berlin noch ein weiterer Prozess zu identifizieren: Der Support-Prozess. Dieser soll nicht im Detail betrachtet werden. Er hat Auswirkungen auf den Entwicklungsprozess, dadurch dass Kunden hin und wieder Fehler am System feststellen, welche im Problem-Tracking-System dokumentiert werden und als Input für Entwicklungstätigkeiten dienen. Die Fehlerbeschreibungen wurden als Datenobjekt von BPMN modelliert.

4.2 Der Entwicklungsprozess im Detail

Abb. 4 nimmt eine zentrale Stellung innerhalb dieser Ausarbeitung ein: Es handelt sich dabei um ein Überblicksmodell über den Entwicklungsprozess.

Zu sehen sind zwei Prozessteile, die nebenläufig zueinander ablaufen (dargestellt mit Hilfe von AND-Gateways). Im oberen Teil befindet sich die eigentliche Softwareentwicklung und unten die Erstellung der Dokumentation.

Im Folgenden wird uns die Softwareentwicklung am meisten interessieren. Diese beginnt mit drei aufeinander folgenden Aktivitäten: DEV activities, CONS activities und RAMP activities. Diese drei sind essenziell für den Prozess und deshalb auch farblich hervorgehoben. Sie werden im Folgenden als Subprozesse verfeinert und in den Abschnitten 4.3 bis 4.5 erläutert. Die restlichen Aktivitäten werden im Abschnitt 4.6 angerissen.

Auffallend bei der Betrachtung von Abb. 4 dürfte die exzessive Verwendung von Datenflüssen sein. Diese Darstellungsform wird am besten der Dokumentenzentriertheit des Entwicklungsprozesses gerecht. BPMN erlaubt an dieser Stelle, dass die Datenobjekte, die nur als Input oder nur als Output dienen, nicht im übergeordneten Diagramm erscheinen müssen. Dies ermöglicht didaktische Vorteile: Das „big picture“ wird nicht unnötig überfrachtet mit Datenflüssen, die erst auf unterer Ebene interessant werden.

Außerdem wurde auf dieser Ebene zum ersten Mal das „SAP Translation Department“ als neuer Pool ins Spiel gebracht. Innerhalb von BPMN wird dies streng genommen als syntaktischer Fehler betrachtet, da die Nachrichtenflüsse, die vom Entwicklungsprozess ausgehen, auch schon auf der höheren Ebene hätten auftauchen müssen. Analog zu den eben erwähnten Datenobjekten wurde hier jedoch zugunsten didaktischer Überlegungen auf eine Darstellung des Pools und der Nachrichtenflüsse in der höheren Ebene verzichtet.

4.3 Die DEV-Phase

Diese Phase ist die wohl interessanteste innerhalb des Entwicklungsprozesses. Hier findet die komplette Entwurfs- und Implementierungsarbeit statt und auch große Teile der Qualitätssicherung. Deshalb werden in dieser Phase die meisten Ressourcen der Abteilung eingesetzt. Abb. 5 zeigt eine Übersicht über die DEV Phase.

Ein XOR-Gateway realisiert in diesem Diagramm zwei verschiedene Prozessvarianten: Den Normalfall (unterer Weg) oder die Bearbeitung eines Hot-Fix-Requests (siehe Abschnitt 4.1). Letzteres sieht die schnellstmögliche Behebung des aufgetretenen Fehlers vor, was in der Aktivität „fast core development“ stattfindet.

Der komplexere Fall tritt häufiger auf. Hierbei laufen das „core development“ (siehe Abschnitt 4.3.1), Testentwicklung, Nachtläufe und Automakes nebenläufig ab. Die Testentwicklung ist in Abb. 7 noch einmal verfeinert dargestellt.

Unter dem Automake versteht man das ständige Kompilieren des aktuellen Quelltextes und das Benachrichtigen der verantwortlichen Entwickler bei eventuellen Kompilierproblemen. Die Nachtläufe beinhalten hauptsächlich das Abfeuern der vorhandenen Integrations-, System-, Installations- und Upgradetests auf das System.

Der „core development“-Prozess findet jeweils auf Teamebene ab: Jedem Development-Team (siehe Kapitel 2) ist mindestens ein Arbeitspaket zugewiesen. Dieser Sachverhalt ist durch die Verwendung des Symbols für multiple Instanzen abgebildet: Jedes Arbeitspaket stellt eine Instanz dar. Aus Managementsicht interessant ist hierbei, dass der Entwurf und die Implementierung allein in der Verantwortung der einzelnen Teams liegen, welche relativ unabhängig voneinander agieren.

Alle diese Aktivitäten werden solange ausgeführt, bis das Management ankündigt, „den Stand abzuziehen“ („Request for CONS“). So wie es in Abb. 5 durch die Verwendung von Intermediate Events modelliert ist, werden alle laufenden Aktivitäten einfach abgebrochen und die entstandenen Zwischenergebnisse verworfen. Dies entspricht allerdings nicht der Realität: In Wirklichkeit werden die Aktivitäten nicht abgebrochen, sondern vielmehr an die Prozessinstanz eines späteren Release-Ziels weitergegeben. Die Modellierung einer solchen Weitergabe stellt sich als sehr schwierig dar und wird in Kapitel 7 diskutiert.

Modellierungstechnisch auffällig ist an dieser Stelle außerdem, dass Nachrichtenfluss innerhalb eines Pools stattfindet. Ob dies erlaubt sein soll oder nicht, ist in der aktuellen Spezifikation von BPMN noch ein sog. „Open Issue", also noch ungeklärt. Sinnvoll wäre es aber allemal, denn z.B. in diesem speziellen Fall gäbe es keine intuitive Modellierungsalternative, die das Abbrechen einer Aktivität durch eine andere Aktivität innerhalb eines Pools darstellt.

Die drei Aktivitäten „fast core development", „core development" und „test case development" sind zu einer Gruppe zusammengefasst, um den Datenfluss nicht pro Aktivität modellieren zu müssen und somit ein „aufgeräumteres" Diagramm zu erreichen.

Als Abschluss der DEV-Phase folgt das eigentliche „Stand abziehen" („select changelists for release"). Hier wird geschaut, welche Änderungen („changelists") seit dem letzten Release am Quelltext gemacht worden sind und das Management wählt eine Teilmenge davon für das aktuelle Release aus.

4.3.1 „Core development"

Der Subprozess „core development" gliedert sich, wie Abb. 6 zeigt, in vier Teile: Design, Implementierung, Testentwicklung und „Quantify & Purify".

Der Verlauf der Design-Phase ist nicht eindeutig geregelt (dargestellt durch das Ad-hoc-Symbol), setzt sich aber zumeist aus den Aktivitäten Schnittstellendefinition, Arbeitslistenformulierung, Glossarentwicklung und Testszenariodefinition zusammen.

Nach dem Design folgen nebenläufig die Testentwicklung (siehe Abb. 7) und die Implementierung. Der Implementierungsprozess („Implementation") findet auf Entwicklerebene statt, d.h. jedem Entwickler sind eine oder mehrere Arbeitslisten zugeordnet, welche jeweils entweder die Beseitigung eines Fehlers oder die Umsetzung einer neuen Funktion zum Ziel hat. Auch hierfür wurde wieder das Symbol für multiple Instanzen verwendet.

Der Hauptteil pro Instanz bildet das Implementieren selber („Implement"). Nebenläufig dazu findet meistens eine Testfallentwicklung des/der Entwickler(s) statt. Hier werden spezielle Testfälle erstellt, die die Erfüllung des Aufgabenpaketes überprüfen. Diese Testfälle, sowie andere bereits existierende Testfälle, werden nun dazu benutzt, um die in das Gesamtsystem integrierte neue Funktionalität zu testen (Integrationstest). Dazu wird zunächst auf dem eigenen Rechner getestet und danach auf den anderen Maschinen, die extra für Testzwecke bereitgestellt werden.

Schlagen die Tests fehl, so fällt man wieder in die Implementierung zurück. Werden alle Tests erfolgreich absolviert und hat man alle nötigen Testfälle geschrieben, so wird der neue Quelltext eingecheckt in das Code-Verwaltungssystem. Außerdem werden noch Beschreibungen abgegeben: Im Falle einer Fehlerbeseitigung wird eine sog. „Patch-Info" verfasst und abgegeben, ansonsten eine „Feature-Info" (dargestellt als Datenobjekte).

Nach der erfolgreichen Implementierung kann eine Überprüfung auf Speicherlöcher („purify") und Performanztests („quantify") erfolgen.

4.4 Die CONS-Phase

In der CONS-Phase (CONS steht dabei für consolidation) werden keine neuen Funktionen mehr eingebaut. Hier werden lediglich die vorhandenen Testfälle auf das System abgefeuert, um letzte Fehler aufzuspüren (siehe Abb. 8). Schlägt dabei ein Testfall fehl, so kann dies zwei Gründe haben: Entweder ist der Testfall zu restriktiv oder gar komplett falsch oder es handelt sich wirklich um einen Fehler im Softwaresystem.

Im ersten Fall wird der Testfall lediglich aus der Testbatterie entfernt und ein neuer Testlauf wird gestartet. Im zweiten Fall wird versucht, den Fehler zu beheben. Wird dabei festgestellt, dass es zu lange dauern würde den Fehler zu beheben, gilt das Release als gescheitert. Dies wird dadurch modelliert, dass ein Error End Event verwendet wird. Auf dem übergeordneten Diagramm (Abb. 5) wird das Abfangen dieses Errors explizit modelliert.

4.5 Die RAMP-Phase

Die Ramp-Phase ist die kürzeste der drei Phasen. Der ungewöhnliche Name dieser Phase lässt sich durch das Bild einer Verladerampe klären: Bevor die Ware auf den LKW verladen wird, finden noch letzte Überprüfungen statt, damit auch die gewünschte Qualität erreicht wird.

Die Phase besteht im Groben aus einer Sequenz dreier Aktivitäten: Nach einem Kompiliervorgang („Make“) werden noch einmal alle Testfälle durchlaufen. Schließlich findet das Packaging statt, wobei Installationspakete erstellt werden für die verschiedenen Rechnerplattformen.

Tritt bei einer der drei Aktivitäten ein Fehler auf, wird der Prozess abgebrochen und ein „Hotfix“ veranlasst, d.h. eine neue Entwicklungsprozessinstanz wird gestartet mit dem Ziel, diesen Fehler zu beheben. Das Auftreten dieses Fehlers wird wieder mit Hilfe eines Error Events modelliert.

4.6 Nachgelagerte Aktivitäten

Nachdem die drei Hauptphasen DEV, CONS und RAMP erfolgreich durchlaufen worden sind, kann – wie Abb. 4 zeigt – das eigentliche Software-Release stattfinden: Die erzeugten binaries („builds“) und der Quellcode („source“) werden auf der Homepage veröffentlicht und Benachrichtigungen werden über entsprechende Mailing-Lists verbreitet.

Danach stellt das Support-Team die Beschreibungen aller neu eingebauten Features und der beseitigten Fehler in einem Dokument zusammen. Dieses sog. „Hand over“ wird später dafür verwendet, bei Kundenanfragen in Abhängigkeit der gerade beim Kunden eingesetzten Version schneller Problemlösungen (z.B. Upgrade auf eine bestimmte Version) zu finden.

Nun ist die Entwicklungsphase fast abgeschlossen. Falls dafür Ressourcen frei sind, werden an dieser Stelle Benchmarks (Performanzvergleiche) für das fertig gestellte Produkt durchgeführt. Diese vergleichen das Produkt mit Vorgängerversionen oder auch mit Konkurrenzprodukten.

5 Verbesserungsvorschläge

Im Folgenden ist eine Auswahl der den Entwicklungsprozess betreffenden Verbesserungsvorschläge aufgeführt. Diese werden lediglich verbal und relativ oberflächlich beschrieben, da eine genaue Definition eines optimierten Prozesses (z.B. mittels expliziter Prozessmodelle) eine sehr umfangreiche Aufgabe darstellt, die den Rahmen dieser Ausarbeitung sprengt.

5.1 Verwendung von Konstruktionsplänen

Wie bereits in Abb. 2 zu sehen ist, sind die zentralen Dokumente innerhalb der Abteilung Quelltext, Fehler- / Featurebeschreibungen und Testfälle. Dies zeigt, dass die Entwicklung eher code-getrieben ist. Doch gerade in einem so komplexen System wie der MaxDB verlieren die Entwickler oft den Überblick und viel Zeit wird für das Verstehen von vorhandenem Quelltext aufgewendet. Durch eine durchgehende Verwendung von geeigneten Modellierungstechniken und Konstruktionsplänen wäre es wahrscheinlich möglich, die Produktivität jedes einzelnen zu steigern, indem vor allem Einarbeitungszeiten verkürzt und Missverständnisse vermieden werden. Eine kürzere Einarbeitungszeit ist dabei auch ein interessanter Punkt für die Anwerbung von Open-Source-Entwicklern.

Anbieten würde sich z.B. FMC [1]. FMC ist an dieser Stelle sehr geeignet, da es eine konzeptionelle Sicht („Systemsicht“) bietet und sich nicht auf eine bloße Beschreibung der Quelltextstruktur beschränkt. FMC ist nicht beschränkt auf objektorientierte Software, wodurch auch die zahlreichen in Pascal und C programmierten Module modelliert werden können.

Zu diesem Thema ist jedoch zu sagen, dass bereits einmal die Verwendung von entsprechender technischer Dokumentation (in gewissem Umfang) bestand. Da die Entwickler allerdings die Modellierungsarbeit als unzumutbare Mehrarbeit zum Programmieren angesehen haben, wurde diese Tradition nach 1999 größtenteils wieder aufgegeben.

5.2 Verwaltung von Spezifikationen

Wie bereits im vorigen Abschnitt erwähnt ist die Entwicklung sehr code-getrieben. Dies äußert sich auch darin, dass vielmals Testfälle als formale Spezifikation genutzt werden. Eine separate Spezifikation wird – wenn überhaupt – in Form eines Word-Dokuments angefertigt, folgt dabei aber keiner Standardisierung oder Formalisierung. In den meisten Fällen besteht eine Spezifikation nur als abstrakte Vorstellung der jeweiligen Entwickler. Dies führt dazu, dass hin und wieder Unsicherheiten entstehen, wie bestimmte Module sich verhalten sollen. In diesem Fall ist ein persönliches Gespräch unumgänglich.

Zu überlegen ist an dieser Stelle allerdings, ob eine Ansammlung formaler Dokumente schneller zu einer Klärung der Unsicherheiten führt.

5.3 Teststrategie

Bei SAP Berlin wird sehr viel Wert auf Qualitätssicherung gelegt. Zu bemängeln ist jedoch, dass nur Integrations- und Systemtests durchgeführt werden. Modultests gibt es nur in wenigen Ausnahmefällen, da hierfür nötige „Dummies“ nicht entwickelt werden. Diese dienen dazu, das Verhalten derjenigen Module, die vom zu testenden Modul verwendet werden, zu simulieren. Dadurch dass nur selten Modultests durchgeführt werden, ist die Fehlersuche relativ langwierig. Die Erstellung der Integrationstests wird unübersichtlich, da die Prüfungen der Modultests nachgeholt werden müssen.

Außerdem ist anzumerken, dass Testabdeckungsmaße nicht zum Einsatz kommen. Testabdeckungsmaße geben eine Aussage darüber, z.B. wie viel Prozent des Quelltextes (Anweisungsüberdeckung) von Testfällen abgedeckt werden. Die Einführung solcher Softwaremaße u.a. dazu führen, toten Code zu identifizieren. Nach Aussagen von Mitarbeitern werden schätzungsweise zwanzig Prozent des Quelltextes nicht mehr verwendet.

Weitere Informationen zu Qualitätssicherung und -management sind in [2] und [6] zu finden.

5.4 Reifegradüberlegungen

Bei genauem Betrachten der Diagramme im Anhang wird man erkennen, dass dort Requirements-Engineering-Prozesse nicht auftauchen. Wären sie genau definiert, könnte eine stärkere Kundenorientierung erreicht werden.

Weiterhin könnte man sich um die Anwendung des an der Carnegie Mellon University (Pittsburgh, PA, USA) entwickelte „Capability Maturity Model“ (CMM) Gedanken machen. Es handelt sich hierbei um ein Assessment-Verfahren, bei dem der „Reifegrad“ des vorherrschenden Prozesses anhand mehrerer Kriterien analysiert wird. Man geht beim CMM davon aus, dass man von der gemessenen Prozessqualität Rückschlüsse darauf machen kann, ob die Produktentwicklung zuverlässig (d.h. termin-, kosten- und qualitätsgerecht) ist oder nicht.

Im CMM sind fünf Stufen definiert, in die der aktuelle Reifegrad des Prozesses eingeordnet wird: initial, repeatable, defined, managed, optimized. Pro Stufe sind Aktivitäten festgelegt, die im Prozess auftauchen müssen.

Dadurch dass im untersuchten Prozess Konfigurations-, Projekt- und Integriertes Softwaremanagement auftauchen, sowie organisationsweite Datenkonsistenz und Reviewaktivitäten durchgeführt werden, befindet sich der aktuelle Prozess auf Stufe 3 (defined). Um auf die Stufe 4 (managed) vorzustoßen, ist zusätzlich zu dem bereits vorhandenen Softwarequalitätsmanagement eine umfassendere Quantifizierung des Prozesses von Nöten. Hierfür müssen z.B. die Dauer des Gesamtprozesses bzw. einzelner Aktivitäten, der Aufwand an Ressourcen für die Durchführung eines bestimmten Prozesses oder einer Aktivität, die Anzahl von Ereignissen, die während der Ausführung eintreten, die durchschnittliche Fehlerzahl pro Quelltextzeile etc. gemessen werden.

6 Vergleich mit UML 2.0 Activity Diagrams

Um BPMN mit Activity Diagrams zu vergleichen, wurde ein Teilprozess mit beiden Notationen modelliert: Abb. 4 (BPMN) und Abb. 9 (Activity Diagram). Besonderes Augenmerk lag dabei auf der Modellierung von Datenfluss in Activity Diagrams. Hier bieten sich die sog. DataNodes an.

Bei Datenfluss von einer Aktivität auf die unmittelbar folgende (z.B. die Übergabe des Quelltextes und der Testfälle von DEV zu CONS) ist die Modellierung mit Activity Diagrams sehr ähnlich und die Semantik gleich im Vergleich mit BPMN.

Schwierig wird es, sobald Daten- und Kontrollfluss entkoppelt voneinander sind: z.B. von DEV zu „revise documents“. Hier geht die Semantik der beiden Notationen auseinander. Während bei BPMN vom Modellierer über Attribute definiert werden kann, dass das Datenobjekt schon vor Beendigung der Aktivität erzeugt wird, ist dies mit Activity Diagrams nur schwer darzustellen. So wie es in Abb. 9 dargestellt ist, würden die „document drafts“ erst erzeugt werden bei Beendigung der DEV-Phase und die „revise documents“-Aktivität dürfte erst starten, sobald die Daten erzeugt worden sind.

BPMN lässt uns also an dieser Stelle mehr Freiheiten als UML, welche aufgrund stärkerer Formalisierung eine intuitive Modellierung nicht zulässt.

Unpraktisch bei Activity Diagrams ist außerdem, dass Dateninputs und -outputs eines Prozesses (z.B. „feature-list“ und „Hand over“) im übergeordneten Diagramm unbedingt auftauchen müssen. Eine Auswahl von dargestellten Sachverhalten nach didaktischem Wert wird hierdurch in Activity Diagrams leider verhindert.

7 Weitergabe von Aktivitäten an andere Prozessinstanzen

In Abschnitt 4.3 wurde bereits erwähnt, dass es problematisch ist, die Weitergabe von Subprozessinstanzen an andere Prozessinstanzen zu modellieren. Dies soll noch einmal näher erläutert werden anhand der Abb. 10 bis 12.

Sei P ein Prozess, der in multiplen Instanzen ausgeführt werden kann. P enthalte einen Subprozess S, der wiederum in multiplen Instanzen ausgeführt wird. Die Ausführung der dazugehörigen Subprozessinstanzen (S-Instanzen) erzeugt Information I, welche in der Aktivität V weiterverarbeitet wird. Dies ist in Abb. 10 dargestellt.

Nehmen wir an, dass P in einer Instanz P1 ausgeführt wird und der Start mehrerer S-Instanzen erfolgt: z.B. S1, S2 und S3. Wie es in Abb. 10 modelliert ist, müssen alle drei Instanzen (S1, S2, S3) beendet sein, bevor V ausgeführt werden kann.

Nehmen wir nun den Fall an, dass die Ausführung der S-Instanzen abgebrochen werden kann durch einen Timer Event (siehe Abb. 11). Wäre in diesem Fall eine S-Instanz noch nicht beendet, würde die bisher gemachte Arbeit innerhalb dieser S-Instanz verloren gehen.

Aus diesem Grund wollen wir die Weitergabe der S-Instanz an eine andere P-Instanz modellieren. Hierzu muss die Ausführung von P-Instanzen und S-Instanzen entkoppelt werden: S-Instanzen dürfen im Gegensatz zur Darstellung in Abb. 11 nicht mehr abgebrochen sondern lediglich weitergereicht werden.

Eine zentrale Frage, die sich bei der Modellierung dieses Sachverhalts stellt, ist: Wie kann Information zwischen verschiedenen Prozessinstanzen weitergereicht werden? An dieser Stelle kann auf die Datenobjekte von BPMN zurückgegriffen werden. Wir können also modellieren, dass Informationen, die von einer S-Instanz produziert werden, über mehrere P-Instanzen hinweg verwendet werden können. Ein Vorschlag für das resultierende Modell ist in Abb. 12 zu sehen. Die Tatsache, dass S-Instanzen angestoßen werden, ohne auf ihre Beendigung warten zu müssen, ist durch die Verwendung von Nachrichtenfluss realisiert. Vernachlässigt wurde hierbei die Einführung von Pools und Lanes. Das neue Datenobjekt D enthält sowohl die resultierenden Daten aus jeder S-Instanz als auch Metainformationen zu den Instanzen (d.h. zu welcher P-Instanz es ursprünglich gehört). In A3 wird geprüft, welche zu der aktuellen P-Instanz gehörenden S-Instanzen bereits beendet worden sind und welche sonstigen S-Instanzen „eingesammelt“ werden könnten.

Unschön an Abb. 12 ist vor allem die gestiegene Unübersichtlichkeit. Nicht mehr sofort ersichtlich ist, dass S eigentlich ein wesentlicher Bestandteil von P ist.

8 Schlussfolgerungen

Die Modellierung von Daten- und Kontrollfluss wird von BPMN gut und vor allem intuitiv unterstützt. Dies hat sich für die analysierten Prozesse als sehr hilfreich erwiesen, da es sich um eine stark dokumenten-zentrierte Entwicklung handelt.

BPMN bietet allerdings sehr viel mehr, als im vorliegenden Fall benötigt wurde. So wurde nur eine kleine Teilmenge von Bildelementen verwendet. Transaktionen und Kompensationen, OR- und komplexe Gateways, sowie zahlreiche Ereignistypen tauchen in diesen Diagrammen nicht auf.

Außerdem ist eine spätere Abbildung der Modelle auf ausführbaren BPEL-Code nicht gewünscht, da eine weitere Automatisierung des Entwicklungsprozesses nicht erforderlich ist. Ziel bei der Modellierung war es deswegen zu jedem Zeitpunkt, ein möglichst für den Menschen verständliches Modell zu entwickeln. Didaktische Überlegungen hatten deshalb immer Vorrang vor Formalismus.

Spannend war zu beobachten, wie die Mitarbeiter auf die Modelle reagieren: Viele wunderten sich zunächst über das eine oder andere Bildelement, da man Symbole wie einen Brief, einen Stern oder einen Blitz nicht in einer spezifizierten Syntax vermuten würde. Kontroll- und Datenfluss wurden jeweils auf Anhieb unterschieden dank der intuitiven Darstellung von Datenobjekten (Blatt mit Eselsohr). Nach wenigen Minuten Erklärung wurden auch die restlichen Bildelemente verstanden. Erklärungsbedarf bestand bei der Hierarchie der verschiedenen Pläne, der Unterscheidung von XOR- und AND-Gateways, sowie bei der Verwendung des „multiple instances“-Symbols. Besonders interessant war festzustellen, dass eigentlich nicht dargestellte Sachverhalte hin und wieder impliziert wurden: So wurde an einer Stelle, wo sich Nachrichtenfluss nach links bewegte, vermutet, dass es sich hierbei immer um ein verspätetes Eintreffen der Nachricht handeln müsste.

Das Interesse an den Diagrammen war sehr unterschiedlich. Manche begnügten sich damit, die Hauptaussagen der Pläne zu verstehen, während andere sich auch

intensiv mit der Syntax und mit Layout-Fragen beschäftigten. Manch einer verschmähte die Modelle völlig aufgrund einer generellen Skepsis gegenüber Plänen.

Zu BPMN kann insgesamt ein positives Fazit gezogen werden. Ein Verbesserungsvorschlag ist lediglich eine größere Modellierungsfreiheit in Bezug auf Nachrichtenfluss innerhalb eines Pools. Pools und Lanes sollten frei nach didaktischen Überlegungen verwendet werden können und sich nicht an Unternehmensgrenzen o.ä. orientieren müssen.

Besonders gut lässt sich BPMN mit FMC-Aufbauplänen integrieren. Die Akteure in den Aufbauplänen tauchen als Pools und Lanes in BPMN auf und die Aktionsfelder in Form von Datenobjekten.

Zur Modellierung des Softwareentwicklungsprozesses ist zu resümieren, dass es einen kleinen Unterschied im Gegensatz zu sonstigen Geschäftsprozessen zu geben scheint: Bei der Softwareentwicklung handelt es sich um einen kreativen Prozess, bei dem auch an geeigneter Stelle vom Standardvorgehen abgewichen wird. Deshalb kann nur ein Idealprozess modelliert werden, an den sich jeder Mitarbeiter möglichst halten sollte, aber nicht immer muss. Würde man alle Varianten, die im Laufe der Zeit auftreten, modellieren wollen, würden die Diagramme sehr viel umfangreicher ausfallen und der Blick aufs Wesentliche ginge an dieser Stelle verloren. Die Diagramme haben sich insgesamt als sinnvolle Diskussionsgrundlage erwiesen.

8.1 Danksagungen

An dieser Stelle möchte sich der Autor noch einmal recht herzlich bei den beteiligten Mitarbeitern von SAP für ihre Unterstützung bedanken.

Besonderer Dank geht an Burkhard Diesing, Alexander Schröder, Marco Paskamp und Jürgen Aurisch. Dank gebührt auch Jürgen Primsch, der erlaubt hat, die Idee einer Prozessanalyse in die Tat umzusetzen und die daraus entstandenen Ergebnisse zu publizieren.

Quellennachweise

1. Apfelbacher, R., Rozinat, A.: Fundamental Modeling Concepts (FMC) Notations Reference, http://fmc.hpi.uni-potsdam.de (2004)
2. Balzert, H.: Lehrbuch der Software-Technik, Band 2. Spektrum Verlag, Heidelberg, Berlin (2000)
3. BPMI (Hrsg.): Business Process Modelling Notation (BPMN), Version 1.0 (2004)
4. Interviews mit Mitarbeitern von SAP Netweaver Kernel MaxDB & liveCache (2004)
5. Leymann, F., Roller, D.: Production Workflow. Prentice Hall, Upper Saddle River (2000)
6. Liggesmeyer, P.: Software-Qualität. Spektrum Verlag, Heidelberg, Berlin (2002)
7. OMG (Hrsg.): Unified Modeling Language (UML) Superstructure (Final Adopted Specification), Version 2.0 (2003)
8. Paulk, M. C., Weber, C. V., Garcia, S. M., Chrissis, M. B., Bush, M.: Key Practices of the Capability Maturity Model, Version 1.1, Carnegie Mellon University (1993)
9. White, S. A.: Introduction to BPMN (2004)

Anhang

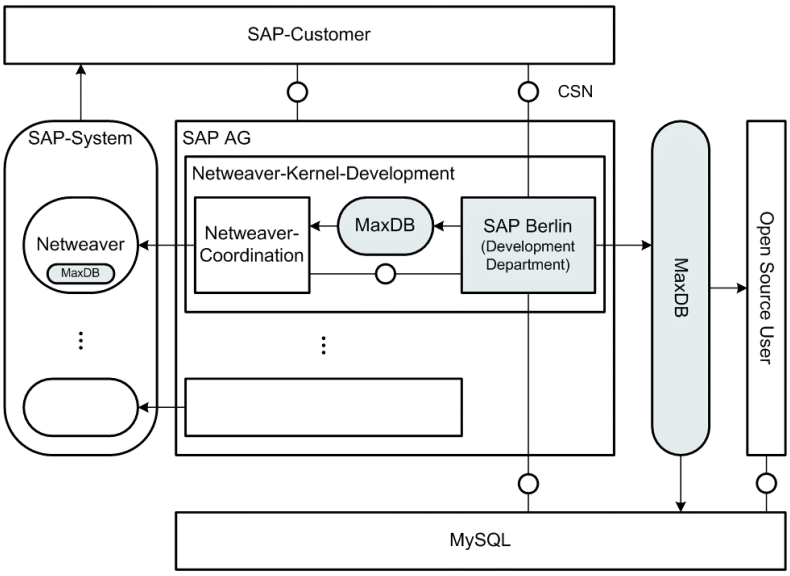


Abb. 1 Die Berliner Entwicklungsabteilung und ihre Umwelt

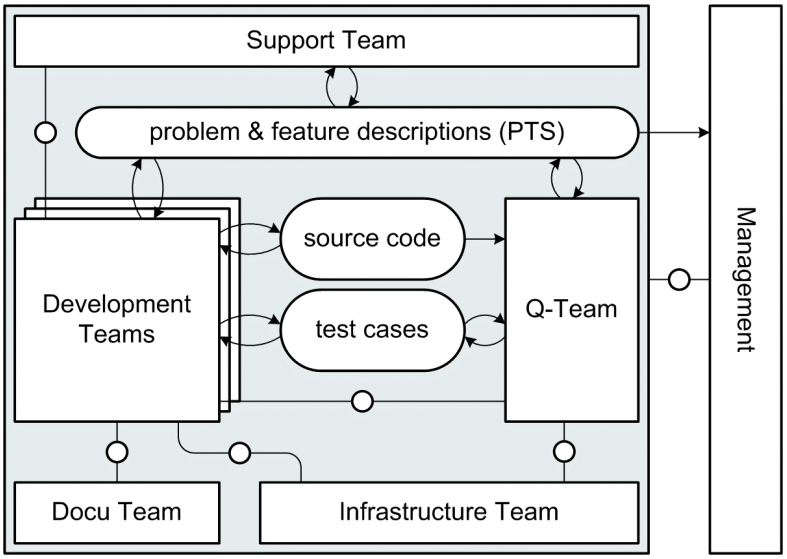


Abb. 2 Aufbau der Entwicklungsabteilung

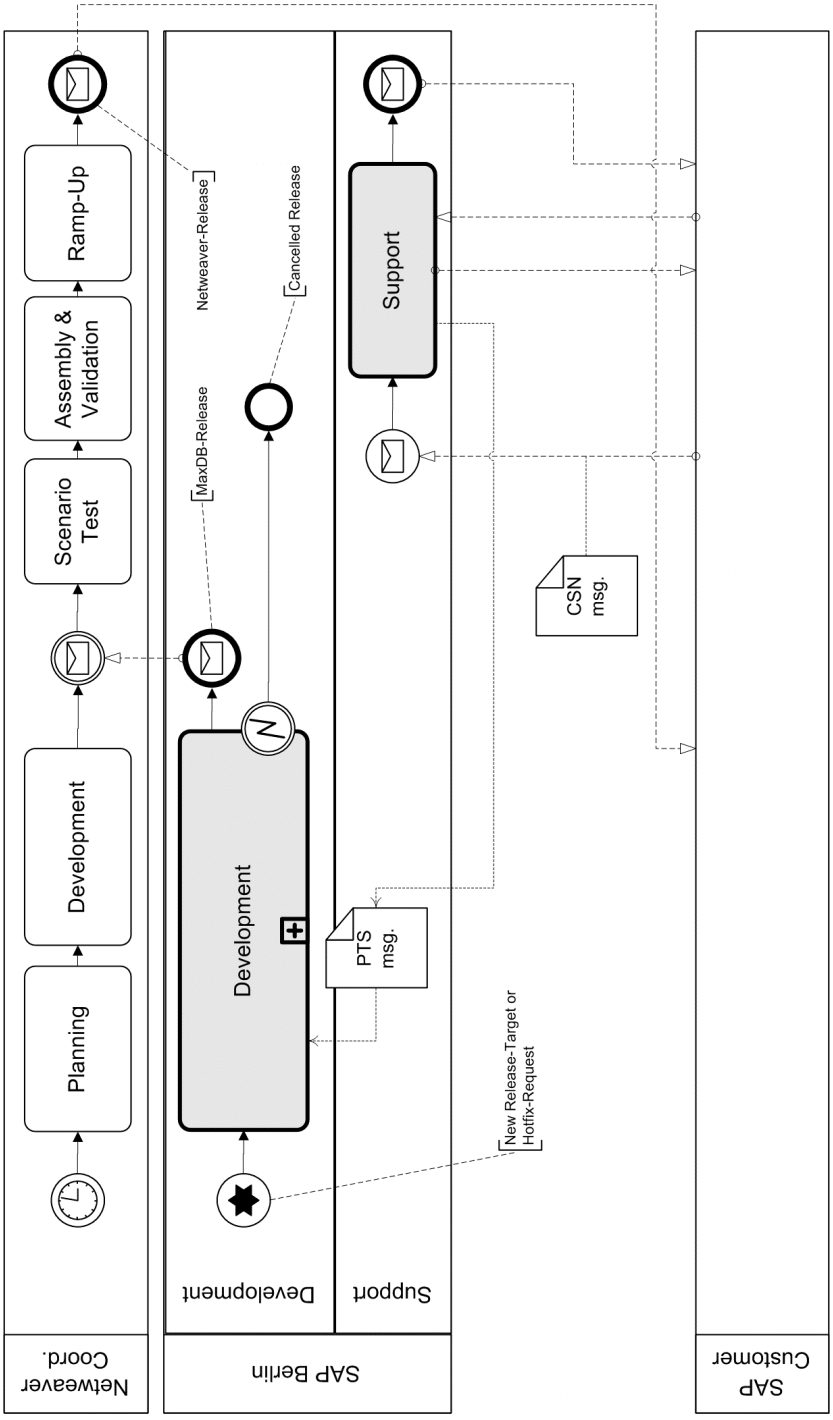


Abb. 3 Einordnung des Entwicklungsprozesses

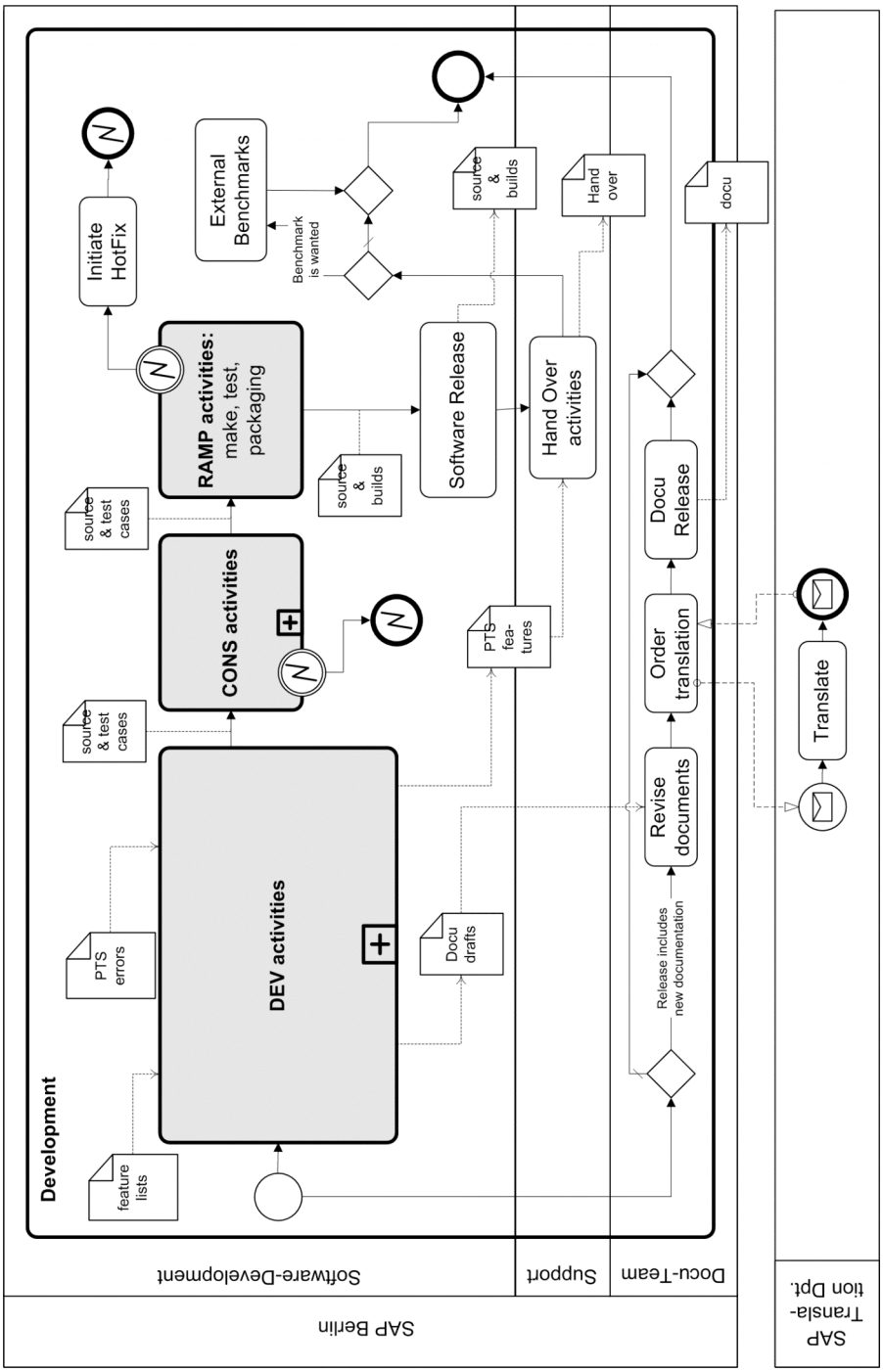


Abb. 4 Entwicklungsprozess

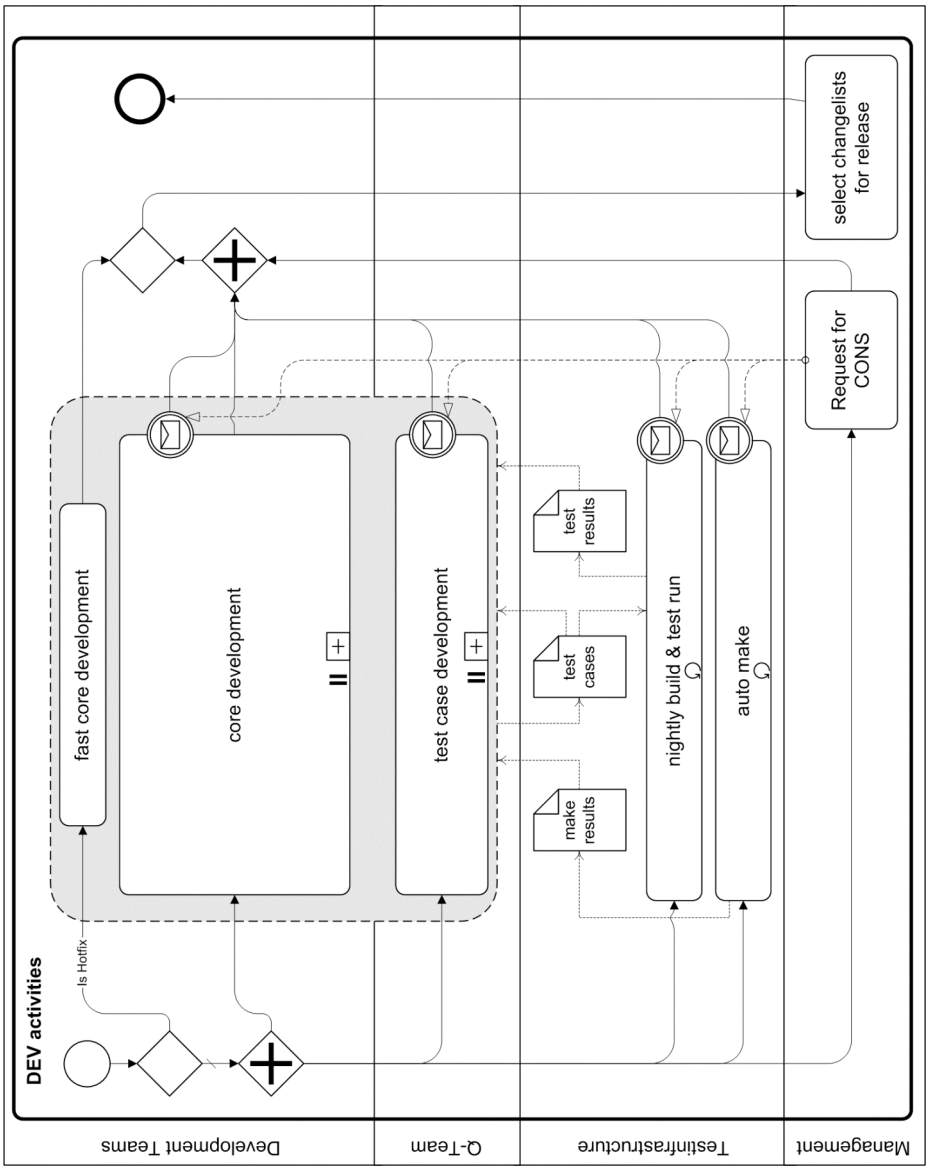


Abb. 5 DEV activities

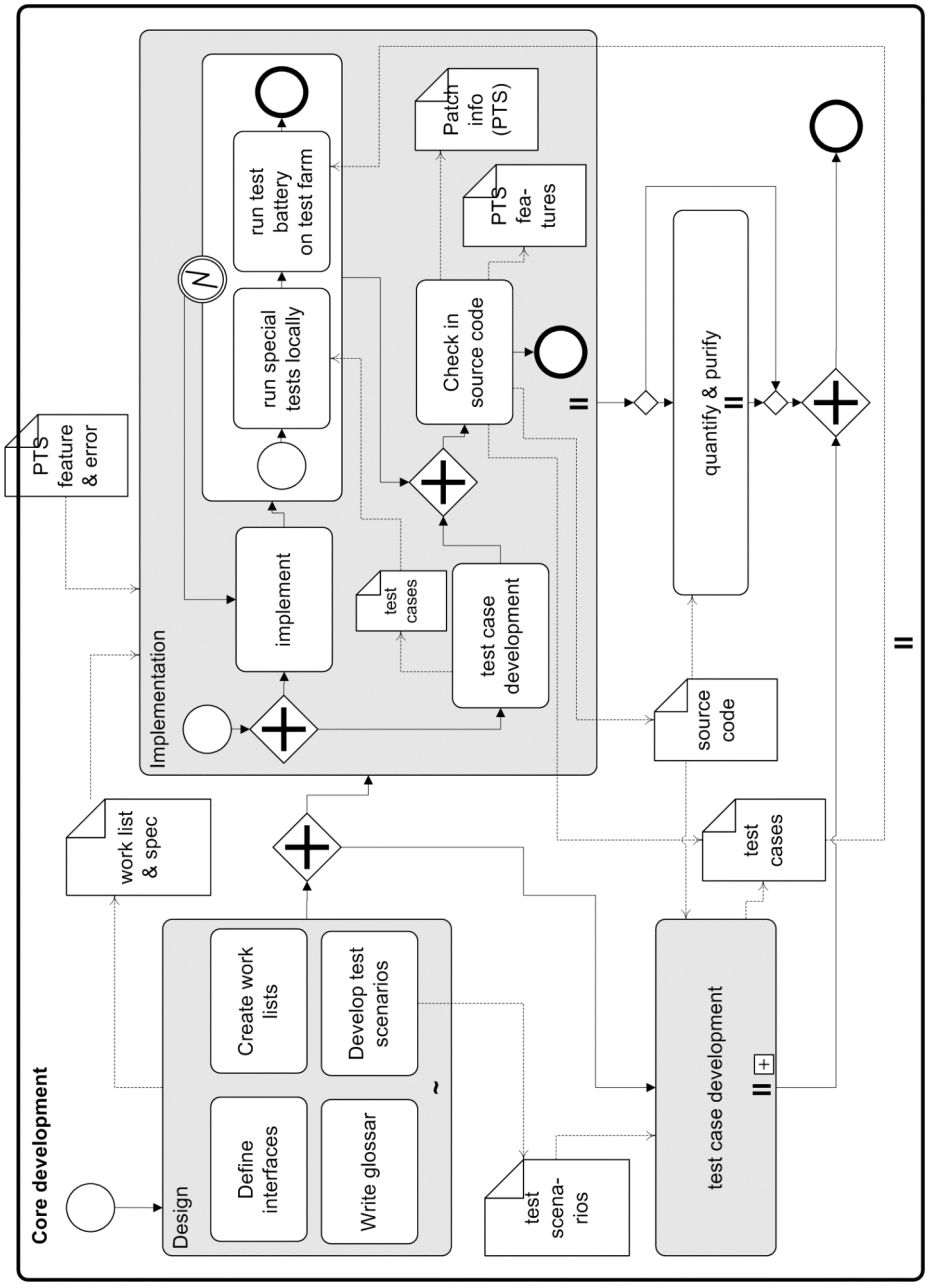


Abb. 6 DEV – Core development

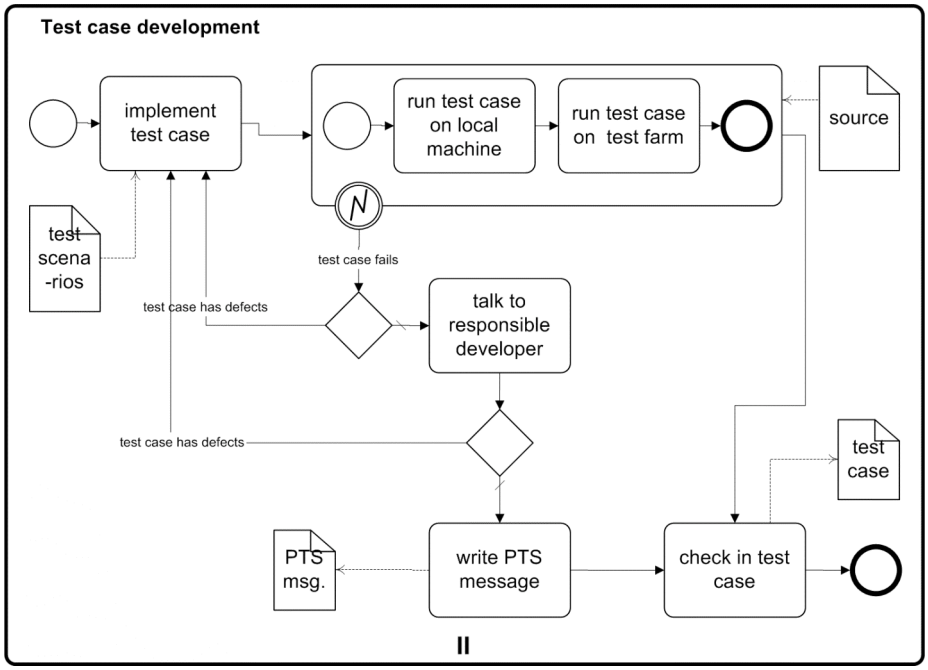


Abb. 7 DEV – Testentwicklung

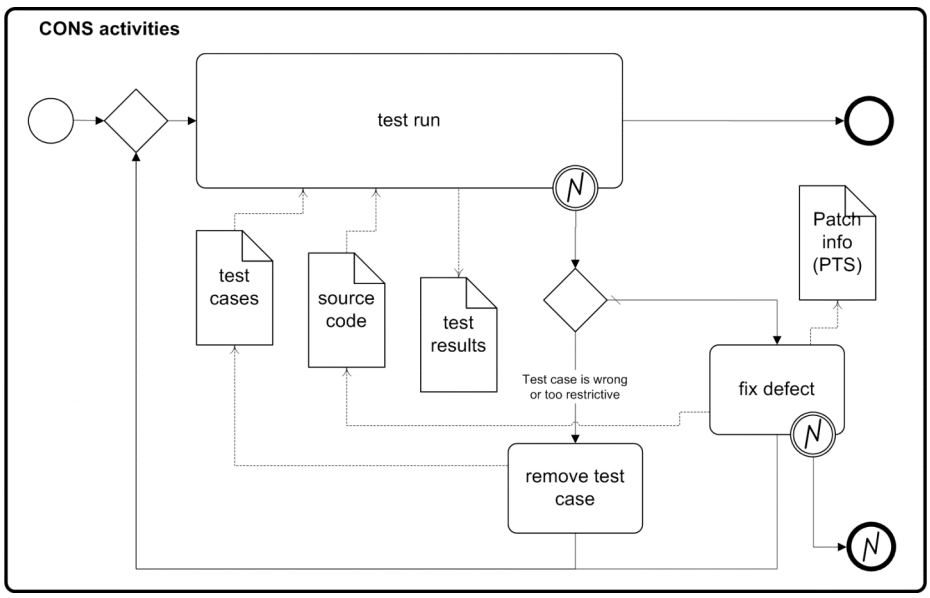


Abb. 8 CONS activities

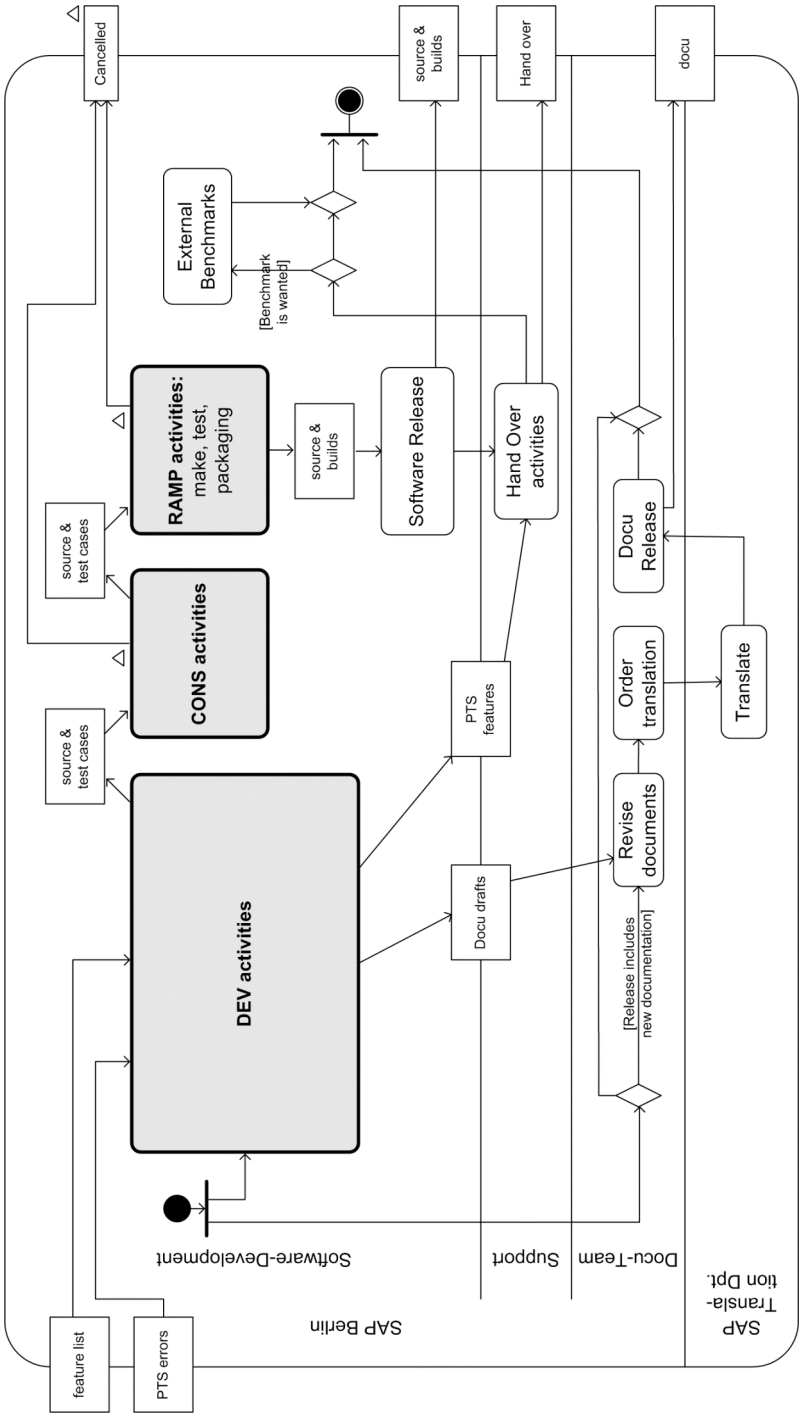


Abb. 9 Entwicklungsprozess als UML 2.0 Activity Diagram

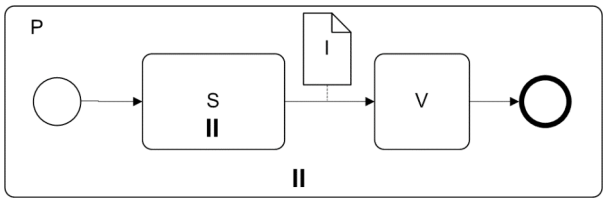


Abb. 10 Multiple Instanzen – Einfaches Beispiel

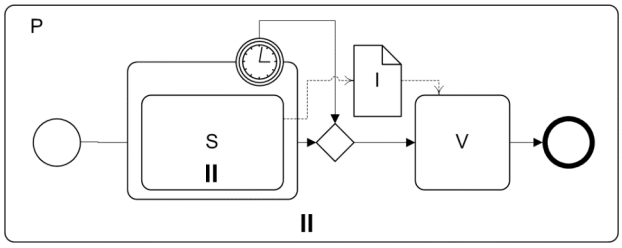


Abb. 11 Multiple Instanzen mit Abbruch von Subprozessinstanzen

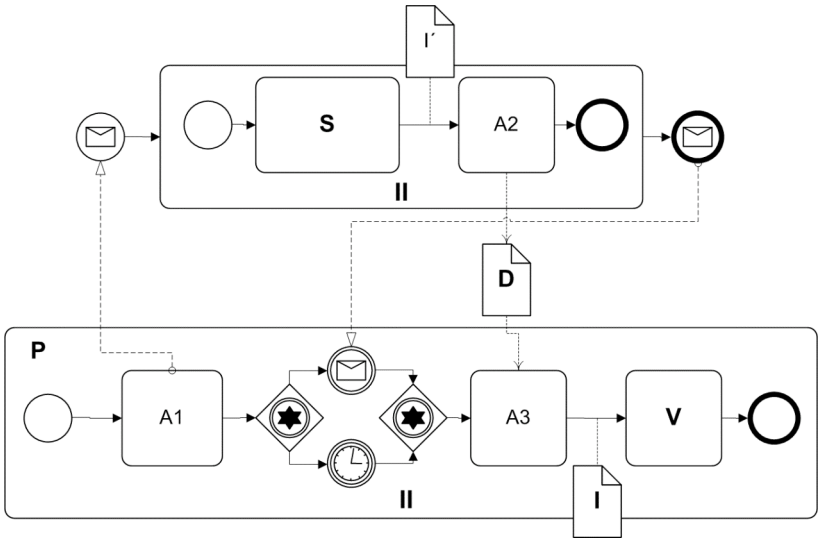


Abb. 12 Multiple Instanzen mit Weitergabe von Subprozessinstanzen