Freie wissenschaftliche Arbeit zur Erlangung des akademischen Grades
Master of Science in Wirtschaftsinformatik

Ein Framework für Production Case Management

Masterthesis

im Fachbereich Wirtschaftswissenschaften II
im Studiengang Wirtschaftsinformatik
der Hochschule für Technik und Wirtschaft Berlin

vorgelegt von: Jan Oehlert
Rungiusstrasse 29
12347 Berlin
Matrikel-Nr.: 521782

Erstbetreuer: Prof. Dr. Margret Stanierowski
Zweitbetreuer: Dr. Frank Puhlmann

Abgabetermin: 21.02.2013

Zusammenfassung

In den letzten Jahren hat sich Case Management auf dem Gebiet der Erfassung und Ausführung hochkomplexer und unstrukturierter Arbeit viel Aufmerksamkeit verschafft. Dabei beschäftigten sich die meisten Untersuchungen mit den Prinzipien und Funktionsweisen von Adaptive Case Management (ACM), welches traditionell auf Prozessflüsse verzichtet und den Verlauf eines Cases zur Laufzeit durch den Case-Worker bestimmen lässt. Production Case Management (PCM) hingegen fokussiert sich auf vordefinierte Prozessmodelle, typischer Weise sog. „Happy Path Modelle“. Bislang erfuhr PCM jedoch eher theoretische Betrachtungen, ohne dass diese mit konkreten Umsetzungsmethodiken versehen wurden. Diese Arbeit stellt ein Framework zur Realisierung von Prozessen mittels Production Case Management vor.

Die Grundidee dieses Frameworks besteht darin, mehrere *Happy-Path-Modelle* für einen Prozess zu erstellen und diese miteinander zu *verschmelzen*. Angereichert um Prozessdaten und Zustände wird das Vorantreiben eines Cases sowohl daten- als auch prozesstechnisch realisiert. Die Arbeit stellt dazu das *PCM-Metamodell* vor, welches alle fachlichen Bestandteile, die zur Ausführung eines Prozesses benötigt werden, zusammenführt und deren Zusammenhänge darstellt. Zu diesem Modell wurde eine Ausführungssemantik entwickelt, anhand derer erläutert wird, wie Cases mit Hilfe der erfassten Prozessbestandteile erzeugt, durchgeführt und abgeschlossen werden. Außerdem werden grundlegende Validierungsalgorithmen gezeigt, die die erfassten Prozessbestandteile auf Widersprüche prüfen um Inkonsistenzen zu verhindern.

Basierend auf dem entwickelten PCM-Metamodell und der zugehörigen Ausführungssemantik zeigt ein implementierter Prototyp die grundlegenden Funktionalitäten des PCM-Frameworks auf. Auch die Validierungsalgorithmen wurden zu großen Teilen umgesetzt und sorgen in dem Prototyp dafür, dass grobe Fehler bei der Erstellung von Prozessdefinitionen noch vor der Ausführung von Cases erkannt werden. Der Prototyp liegt in Form eines Plugins für die inubit Suite vor, unterstützt durch generische und individuell zu erstellende Workflows.

Abstract

Over the last few years, Case Management has gained a lot of attention in the field of supporting highly-complex business processes and unstructured work. The majority of scientific papers on this topic however, focus on Adaptive Case Management (ACM). Despite slight distinctions between different ACM definitions, most publications agree that ACM provides a data-driven approach, enabling a case worker to tie together the sequence of activities at runtime. Production Case Management (PCM), in contrast, builds on pre-defined processes, typically representing so-called *happy paths*. Yet is the research on PCM still limited to theoretical investigations, lacking a precise methodology for execution. This master thesis introduces a framework to model, validate and execute processes via Production Case Management.

The core idea of this framework is to merge together a set of *happy path* models. Through extending these models by data artifacts and case states, cases are enabled to be both process- and data-driven. For this purpose, the PCM meta model is presented, uniting all functional process elements and illustrating their correlation. The meta model is featured by execution semantics that define how cases are instantiated, proceeded and finally terminated. Furthermore, the presented framework contains a set of validation algorithms, aiming to ensure that a defined process is free from conflicts that may appear from contradictory process models or state definitions.

Based on the PCM meta model and the according execution semantics, an implemented prototype demonstrates the core functionalities of the developed framework. Also most validation algorithms were built into the prototype, avoiding basic mistakes during the process definition. The implementation appears in the form of a plug-in for the BPMS *inubit-Suite*, supported by both generic and individual structured workflows.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Abkürzungsverzeichnis

ACM	Adaptive Case Management
BOD	Business Object Diagram
BPD	Business Process Diagram
BPM	Business Process Management
BPMN	Business Process Model and Notation
BPMS	Business Process Management System
CM	Case Management
CMMN	Case Management Modeling Notation
JVM	Java Virtual Machine
PCM	Production Case Management
PCMS	Production Case Management System
REST	Representational State Transfer
SDK	Standard Development Kit
UML	Unified Modeling Language
WFMS	Workflow Management System
XML	Extensible Markup Language

1 Einleitung

In den vergangenen Jahrzehnten hat sich das Business Process Management (BPM, dt. Geschäftsprozessmanagement) zur Erfassung, Steuerung und Verbesserung von betrieblichen Abläufen als etablierte Disziplin der Wirtschaftsinformatik durchgesetzt. Insbesondere dort, wo Geschäftsprozesse wiederholt gleich und nach einem festen Schema ablaufen, lässt sich mit Hilfe von Geschäftsprozessmanagement ein Grad an Standardisierung und Automatisierung erreichen, der zu erheblichen Wertsteigerungen im Unternehmen führen kann. Zunehmend soll das Geschäftsprozessmanagement aber auch auf Prozesse ausgedehnt werden, die sich aufgrund ihrer Komplexität und anderer Faktoren weniger leicht standardisieren lassen. Bei derartigen Prozessen mögen zwar die einzelnen Elemente, wie z.B. die durchzuführenden Aktivitäten und einflussnehmende Geschäftsdaten, vorhersagbar sein, doch lässt sich oft nur schwer sagen, in welcher Reihenfolge Aktivitäten durchgeführt und Daten erfasst oder erzeugt werden. Dies findet man beispielsweise bei einem Produktentwicklungsprozess, der bei jedem zu entwickelnden Produkt anders abläuft, oder bei der Behandlung von Kundenreklamationen, wo einzelne Aktivitäten zwar vordefiniert werden können, das konkrete Vorgehen jedoch von vorliegenden Daten oder den Rechten des bearbeitenden Mitarbeiters abhängt. An dieser Stelle soll das Konzept des *Case Managements* greifen. Dabei wird jede Prozessinstanz, also z.B. die konkrete Bearbeitung einer Reklamation, als eigener Fall (engl. Case) betrachtet, der jedes Mal andersartig ablaufen kann.

1.1 Motivation und Zielsetzung

Die Ausweitung von Geschäftsprozessmanagement auf unstrukturierte Prozesse ist zu einem stark diskutierten Thema geworden. Obwohl es einem einheitlichen Verständnis fehlt, hat sich Case Management als die Methode der Wahl für diese Aufgabe herauskristallisiert. Generell sind dabei die zwei Entwicklungen von Adaptive Case Management (ACM) und Production Case Management (PCM) zu beobachten. Während ersteres bereits sehr ausführlich diskutiert und untersucht wurde, existiert für PCM kaum mehr als eine ideologische Betrachtung. Ziel dieser Masterthesis soll es daher sein, ein Konzept für eine Umgebung zu entwerfen, in der Geschäftsprozesse mit Hilfe von PCM realisiert werden können. Die Grundidee besteht darin, einen Prozess mit mehreren leicht zu erstellenden *Happy-Path-Modellen* zu unterstützen. Deren Inhalte werden miteinander verschmolzen und

helfen dem *Case Worker* die richtigen Entscheidungen zu treffen. Das Konzept soll aufzeigen, welche fachlichen Inhalte für eine PCM-Unterstützung erfasst werden müssen, wie diese miteinander in Verbindung stehen und wie man sie zur Ausführung bringen kann. Anhand einer anschließend zu entwickelnden, prototypischen Implementierung soll dieses Konzept auf seine Funktionalität hin untersucht werden.

1.2 Vorgehensweise

Zum Beginn der Arbeit sollen die notwendigen Grundlagen zur Bearbeitung des Themas vorgestellt und erläutert werden. Dies umfasst in erster Linie, in den entsprechend benannten Kapiteln, die Technologien *Geschäftsprozessmanagement* und *Case Management*. Dazu beschäftigt sich Kapitel 2 mit dem Geschäftsprozessmanagement und geht dabei auch auf dessen, gemessen am Grad der technischen Unterstützung, unterschiedlichen Ausprägungen *Geschäftsprozessmodellierung* und *Ausführung von Geschäftsprozessen* ein. Generell wird das Geschäftsprozessmanagement als Ausgangsbasis behandelt und die klassische Vorgehensweise bei heutigen BPM-Projekten erläutert. Das Kapitel endet mit einer Kritik am Geschäftsprozessmanagement, welche die Notwendigkeit einer neuen Technologie aufzeigen soll.

Als diese neue Technologie wird anschließend das Case Management vorgestellt, wie es bereits in verschiedenen wissenschaftlichen Arbeiten und Büchern beschrieben wurde. Kapitel 3 gibt eine detaillierte Einführung in das Case Management und auch hier soll ein Hauptaugenmerk auf der Ausarbeitung verschiedener Ansätze zu dieser Technologie liegen. Betrachtet werden sollen dazu die Ansätze *Adaptive Case Management* und *Production Case Management*.

Neben der Erläuterung einzelner Grundbegriffe und der Vermittlung eines allgemeinen Verständnisses für die Technologien Geschäftsprozessmanagement und Case Management, sollen diese durch die Kapitel 2 und 3 vor allem auch gegeneinander abgegrenzt und ihre Unterschiede hervorgehoben werden.

Kapitel 4 versteht sich als erster Abschnitt des Hauptteils dieser Arbeit. Es beschäftigt sich mit der Frage danach, welche fachlichen Prozessbestandteile aufgenommen und definiert werden müssen, um einen Prozess mit Hilfe von PCM umsetzen, d.h. ausführbar machen, zu können. Zu jedem dieser Prozessbestand-

teile soll auch untersucht werden, ob und wie deren Aufnahme bereits durch vorhandene Technologien wie Notationen und Diagrammarten unterstützt wird.

Ein konkretes Konzept zur Umsetzung eines Prozesses mit Hilfe von PCM stellt anschließend Kapitel 5 vor. Zunächst sollen dazu die in Kapitel 4 identifizierten Prozessbestandteile, deren Aufnahme für eine Umsetzung benötigt wird, in einem übergreifenden Modell zusammengeführt werden. Zusätzlich zu diesem Modell soll auch eine darauf anzuwendende Ausführungssemantik entwickelt und in diesem Kapitel vorgestellt werden. Das Modell und deren Ausführungssemantik bilden die Grundlage für eine prototypische Entwicklung.

Kapitel 6 dokumentiert den praktischen Teil dieser Arbeit. Dieser soll darin bestehen, das im vorherigen Kapitel vorgestellte Modell und dessen Ausführungssemantik in einen Prototyp umzusetzen. Zu dieser Umsetzung wird zu Beginn des Kapitels ein Geschäftsszenario vorgestellt, anhand dessen die prototypische Umsetzung beispielhaft vorgeführt werden soll. Das Kapitel nennt die zur Realisierung verwendeten Technologien, dokumentiert die realisierten Funktionalitäten und liefert ggf. Handbücher zu deren Verwendung.

Abschließend findet im Kapitel 7 neben einer Zusammenfassung vor allem eine kritische Betrachtung der Arbeit statt. Gesetzte Ziele sollen mit dem letztendlichen Ergebnis verglichen und eventuelle Abweichungen ergründet werden. Die Arbeit endet mit einem Ausblick wie es auf diesem Gebiet weitergehen könnte.

2 Geschäftsprozessmanagement

Das folgende Kapitel stellt das Geschäftsprozessmanagement, im Folgenden auch Business Process Management (BPM) genannt, als Grundlage für die Gesamtthematik vor. Es erklärt die Absichten und die Motivation von BPM, sowohl aus betriebswirtschaftlicher als auch aus technischer Sicht. Zur fachlichen Betrachtung wird näher auf die Geschäftsprozessmodellierung eingegangen, während ein kurzer Anriss zur Ausführung von Geschäftsprozessen die technische Seite behandelt. Abschließen soll das Kapitel mit dem Aufzeigen von Grenzen von BPM und einer damit einhergehenden kritischen Betrachtung, die die Notwendigkeit weiterführender Technologien aufzeigen soll.

2.1 Motivation von Geschäftsprozessmanagement

Schon seit langer Zeit hat man in Unternehmen erkannt, dass die Schaffung von Prozessorientierung, d.h. das Ausrichten des eigenen Unternehmens an seinen Prozessen, ein Grundstein ist, um in einer Geschäftswelt, die sich zunehmend globalisiert und damit wettbewerbsintensiver präsentiert, zu überstehen. Ziel dieser Prozessorientierung ist es, die betrieblichen Abläufe im eigenen Unternehmen nicht länger als unkoordinierte Gegebenheiten zu betrachten, sondern vielmehr diese zu dokumentieren, zu strukturieren, zu verbessern, zu überwachen und damit transparenter zu gestalten. Neben betriebswirtschaftlichen Interessensgruppen, denen es bei der Verbesserung von betrieblichen Abläufen u.a. darum geht, die Kundenzufriedenheit zu erhöhen, Kosten zu reduzieren oder neue Produkte und Dienstleistungen einzuführen, herrscht auch in der *IT-Gemeinde* ein großes Interesse an Business Process Management.[1] Zum Einen hat sich die Lieferung und Bereitstellung von IT-Systemen zur Unterstützung von BPM als eigenes Geschäftsfeld entwickelt, zum Anderen stellt die Integration bestehender Systeme eine wichtige Disziplin bei der Umsetzung von Geschäftsprozessen dar. Mit einer derart stark ausgeprägten Schnittstelle zwischen Betriebswirtschaft und IT kann Business Process Management als ein Paradebeispiel für die Wirtschaftsinformatik verstanden werden.

[1] Vgl. **Weske 2007**, S3

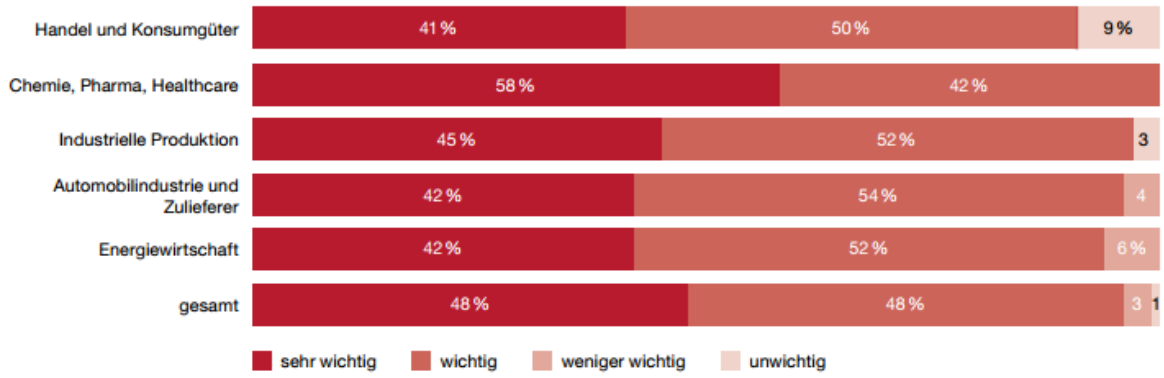


Abbildung 1: Rolle des Geschäftsprozessmanagements in Unternehmen

[Quelle: **PWC2011**, S. 16]

Für wie wichtig die Rolle von Business Process Management in Unternehmen gehalten wird, erfragte das Wirtschaftsprüfungsunternehmen PricewaterhouseCoopers in der Studie „Zukunftsaufgabe Geschäftsprozessmanagement“[2]. Abbildung 1 zeigt, dass mehr als 95% der befragten Unternehmen diese Rolle als „sehr wichtig“ oder „wichtig“ einstufen. Mit Hilfe von Geschäftsprozessmanagement sparen Unternehmen Ressourcen, steigern ihre Performance und erhöhen die Kundenzufriedenheit.[3] Es benötigt Verständnis, Transparenz und Kontrolle über Geschäftsprozesse, um diese in ihrer Rolle als strategische Vermögensgegenstände verwalten zu können.[4]

2.2 Prozess

Ein Prozess stellt im Allgemeinen den Ab- bzw. Verlauf oder auch den Fortschritt einer Sache dar. Obwohl der Terminus vor allem im juristischen Umfeld Anwendung findet, spielt er auch in der Betriebswirtschaft eine erhebliche Rolle. Dort auch als Geschäftsprozess bezeichnet, beschreibt der Prozess einen Satz an Aktivitäten, welche koordiniert in einem organisatorischen und technischen Umfeld durchgeführt werden. Die Aktivitäten dienen der Erfüllung eines Geschäftsziels.

[2] Vgl. **PWC 2011**
[3] Vgl. **Walter 2009**, S.9
[4] Vgl. **Swenson 2010**, S.30

Jeder Geschäftsprozess wird von einem einzelnen Unternehmen durchgeführt, kann aber mit Geschäftsprozessen anderer Unternehmen interagieren.[5]

Geschäftsprozesse stellen außerdem einen wichtigen Aspekt des Qualitätsmanagements dar. Immer häufiger müssen Unternehmen in einer Geschäftswelt mit hohen Qualitätsanforderungen ihr eigenes Qualitätsmanagement unter Beweis stellen. Zertifiziert wird dieses z.B. über die ISO Standards 9001:2000 bzw. 9001:2008. Die Standards rücken die im Unternehmen ablaufenden Geschäftsprozesse in den Fokus und machen diese somit zum „Kern des Qualitätsmanagements“[6].

2.3 Management

In der Geschäftswelt beschreibt der Begriff *Management* zwei verschiedene Ressourcen eines Unternehmens. Zum Einen meint es die Institution *Management*, d.h. die organisatorischen Einheiten, die mit leitenden Aufgaben betraut sind und damit die Interessen des Arbeitgebers gegenüber der Arbeitnehmerschaft vertreten. Zum Anderen ist *Management* eine Funktion im Unternehmen, ausgeführt von Führungskräften zur Erfüllung derer Aufgaben.[7] Das heißt Management kann als Prozess umschrieben werden, bei dem Menschen versuchen, durch effiziente Nutzung von Ressourcen bestimmte Ziele zu erreichen. Management wird häufig in die vier Funktionen Planung, Kontrolle, Personalführung und Organisation unterteilt.

Eine Unternehmensressource, auf die Managementfunktionen immer häufiger angewendet werden, sind die im Unternehmen ablaufenden Geschäftsprozesse. Dieses *Geschäftsprozessmanagement* umfasst Konzepte, Methoden und Techniken, um das Design, die Administration, die Konfiguration, die Umsetzung sowie die Analyse von Geschäftsprozessen zu unterstützen. Grundlage von Geschäftsprozessmanagement ist die reine Darstellung von Geschäftsprozessen mit deren Aktivitäten und den zwischen diesen Aktivitäten bestehenden Ausführungsbedingungen. Sobald Geschäftsprozesse definiert werden, können sie analysiert, verbessert und umgesetzt werden.[8]

[5] Vgl. **Weske 2007**, S. 5
[6] Vgl. **Walter 2009**, S. 8
[7] Vgl. **Gabler**
[8] Vgl. **Weske 2007**, S.5

2.4 Stufen von Geschäftsprozessmanagement

In Abhängigkeit des technischen Anteils lässt sich BPM grob in die beiden Stufen *fachliches* und *technisches* BPM unterteilen. Beim fachlichen BPM geht es vorrangig darum, Geschäftsprozesse aufzunehmen und zu dokumentieren. Mit der dazu notwendigen Geschäftsprozessmodellierung beschäftigt sich das folgende Unterkapitel. Daran anschließend wird das zur Ausführung von Geschäftsprozessen benötigte technische BPM näher erläutert. Dazu wird näher auf sog. Workflows und Workflow-Systeme eingegangen.

2.4.1 Geschäftsprozessmodellierung

Zur Dokumentation von Geschäftsprozessen und ihren Abläufen reicht es im einfachsten Falle aus, diese mittels Textbeschreibung oder in tabellarischer Form festzuhalten. Sobald Prozesse jedoch komplexer, d.h. mit Verzweigungsregeln, Ereignissen oder den zugehörigen Daten und Organisationseinheiten dargestellt werden sollen, wird eine genau definierte Sprache benötigt, die dafür sorgt, dass gleichartige Prozesselemente auch immer einheitlich dargestellt werden. Da diese Sprachen eher graphischer Natur sind, bezeichnet man sie als Notationen. Diese Notationen sorgen auch dafür, dass Prozessmodelle, die sich an die Regeln einer Notation halten, von jedem verstanden werden können, der die Notation beherrscht.[9] Zu diesem Zweck bedienen sich die Notationen graphischer Elemente, die den Ablauf eines Prozesses möglichst realitätsnah darstellen. Kernelemente der meisten Notationen sind dabei:

- Aktivitäten oder Funktionen, die die aktive Ausführung einer Aufgabe durch Menschen, Maschinen oder IT-Systeme darstellen,
- Ereignisse, die das Eintreten bestimmter Zustände repräsentieren,
- Verzweigungen, die den Ablauf des Prozesses anhand von bestimmten Regeln teilen oder wieder zusammenführen.

Um die genaue Abfolge der Prozesselemente darzustellen, werden diese durch sogenannte Kanten (Pfeile) miteinander verbunden.

[9] Vgl. **Allweyer 2009**, S. 8

Je nach Notation und Detaillierungsstufe kommen weitere Elemente zur Modellierung von Daten, Organisationseinheiten und Verfeinerungen der oben beschriebenen Kernelemente zum Einsatz. Einen Einblick in die komplette Bandbreite der sich als Industriestandard etablierten Notation *BPMN 2.0*, inkl. Veranschaulichungen, findet sich in **Allweyer 2009**.

2.4.2 Ausführung von Geschäftsprozessen

Sinn und Zweck der Erhebung von Geschäftsprozessen ist häufig deren anschließende, ggf. automatische Ausführung sowie deren Monitoring. Sogenannte Workflow Management Systeme (WFMS) transferieren fachliche Prozessmodelle in ausführbare Workflows. Dabei entspricht ein Workflow nicht immer exakt einem Geschäftsprozess. Ein Workflow kann einen, mehrere oder auch nur einen Teil eines Prozesses technisch unterstützen. Wird ein Workflow beispielsweise so generisch gehalten, dass seine Funktionalität wiederverwendbar ist, so unterstützt er die Ausführung mehrerer Prozesse. Ebenso ist es denkbar, dass ein Workflow nur einen einzigen Prozess technisch umsetzt und dies möglicherweise auch nur in Teilen. Die Workflow Management Coalition unterscheidet vier Typen von Workflows:[10]

[10] Vgl. **Müller 2005**, S. 8

Workflow	Funktion
Production Workflow	Unterstützt fest strukturierte und vordefinierbare Vorgänge, die zumeist zeitkritisch und von strategischer Bedeutung sind.
Administrative Workflow	Unterstützt strukturierte Routineabläufe, die nicht strategisch, selten zeitkritisch und von geringem Geldwert sind.
Collaborative Workflow	Unterstützt das gemeinsame Erarbeiten eines Ergebnisses; dieser Begriff wird auch als Synonym für Groupware verwendet.
Ad-hoc Workflow	Unterstützt einmalige oder stark variierende Prozesse, die wenig strukturiert und nicht vorhersehbar sind.

Tabelle 1: Workflow-Typen gem. Workflow Management Coalition

[Quelle: **Müller 2005,** S. 8]

Eine andere Art der Unterteilung von Workflows als die oben gezeigte, ist die Betrachtung, ob und inwieweit die Arbeit von Menschen an dem umgesetzten Prozess beteiligt ist. Während manche Vorgänge, wie zum Beispiel das Laden, Aufbereiten und Weiterschreiben von Datensätzen, komplett automatisiert ablaufen können, benötigt eine Vielzahl von Prozessen die Ergebnisse menschlicher Arbeit als Eingabe. Das bedeutet, dass die durch Menschen unterstützten Workflows an gegebener Stelle unterbrochen werden müssen, um auf die benötigten Eingaben zu *warten*. Diese Workflows werden auch als *Human Workflows* bezeichnet.

Der Ablauf von Workflows wird durch Ereignisse gesteuert. Definierte Startereignisse lösen einen Workflow bzw. eine Workflowinstanz aus und starten damit die Abarbeitung der in diesem Workflow umgesetzten Funktionalitäten. Die Reihenfolge der Abarbeitung folgt einem Sequenzfluss, der meist starke Ähnlichkeit mit dem Prozessfluss im zugehörigen fachlichen Prozess aufweist. Abbildung 2 zeigt einen fachlichen Prozess (oben) mit dem zugehörigen technischen Workflow (unten). Da Workflows aufgrund des Sequenzflusses immer nach dem gleichen oder zumin-

dest einem ähnlichen Schema ablaufen, werden sie auch als *strukturierte* Workflows bezeichnet.

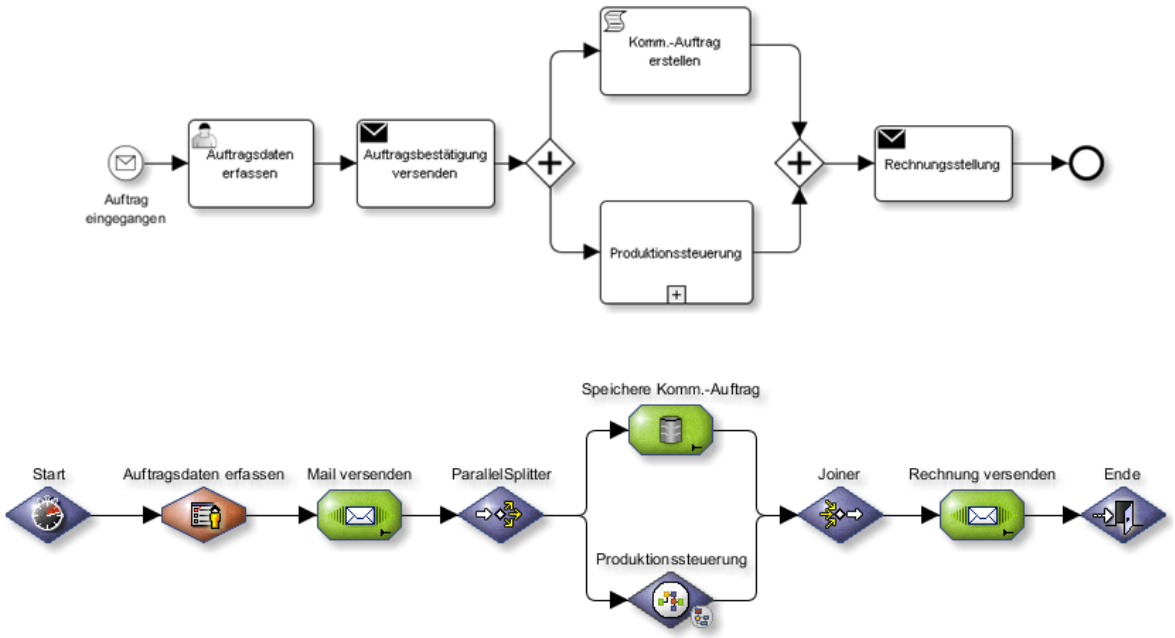


Abbildung 2: Fachlicher Prozess (oben) und technischer Workflow (unten)

Welche Funktionalitäten ein Workflow umsetzen kann hängt vom verwendeten Workflow Management System (WFMS) bzw. Business Process Management System (BPMS) ab. Zu den Standards gehören beispielsweise

- das Lesen und Schreiben von Dateien,
- verschiedene Datenbankoperationen,
- der Empfang und Versand von E-Mails,
- das Aufrufen und/oder die Bereitstellung von WebServices,
- Datenempfang und –versand über unterschiedliche Protokolle (z.B. HTTP).

Grundlage dieser Masterarbeit ist die Beobachtung, dass sich nicht alle Geschäftsprozesse dafür eignen, mit klassischem BPM, d.h. für die technische Ausführung mit strukturierten Workflows, umgesetzt zu werden. Dazu werden im Laufe der Arbeit Beispiele und Eigenschaften genannt, wann sich ein Prozess für alternative Vorgehensweisen eignet. An dieser Stelle soll jedoch erwähnt werden, unter welchen Bedingungen sich ein Prozess speziell dafür eignet, mit eben jenem

klassischen Herangehen umgesetzt zu werden. Folgende Eigenschaften lassen annehmen, dass ein Prozess mit strukturierten Workflows realisierbar ist:[11]

- Der Prozess kann leicht in gut definierbare Schritte unterteilt werden, die in jeder Prozessinstanz auftauchen.
- Für jeden Prozessschritt lassen sich Ein- und Ausgaben klar definieren.
- Es lässt sich eindeutig zuordnen, welche Ausgaben des einen Prozessschrittes als Eingaben für andere Prozessschritte dienen.
- Die Ausführung eines Prozessschrittes benötigt nichts weiter als die zuvor formalisierten Eingaben.
- Zwei Prozessschritte dürfen nicht parallel ausgeführt werden, wenn die Ergebnisse des einen als Eingaben des anderen Schrittes dienen. Ansonsten ist Parallelität, genauso wie Schleifen, möglich und erwünscht.
- Jeder Schritt wird von genau einer bestimmten Gruppe oder Person ausgeführt.

Wie auch bei anderen Software Engineering Projekten oder allgemein im Projektmanagement hat sich auch für das Workflow-Management ein gewisser Lebenszyklus etabliert. Abbildung 3 zeigt die einzelnen Phasen dieses Zyklus und unterteilt diesen nochmal in drei Teilzyklen. Der erste dieser Teilzyklen beschäftigt sich mit der Modellierung, sowie mit der Analyse von Geschäftsprozessen und schließt mit Hilfe der Restrukturierung auch die Anpassung dieser Prozesse an die Geschäftsstrategieentwicklung mit ein. Im zweiten Teilzyklus werden Workflows aus den zuvor erstellten Geschäftsprozessmodellen abgeleitet und modelliert. Die Simulation, Analyse und schließlich auch die Optimierung der Workflows gehört ebenfalls zu diesem Teilzyklus.

[11] Vgl. **Bider et al. 2012**, Kap. 2

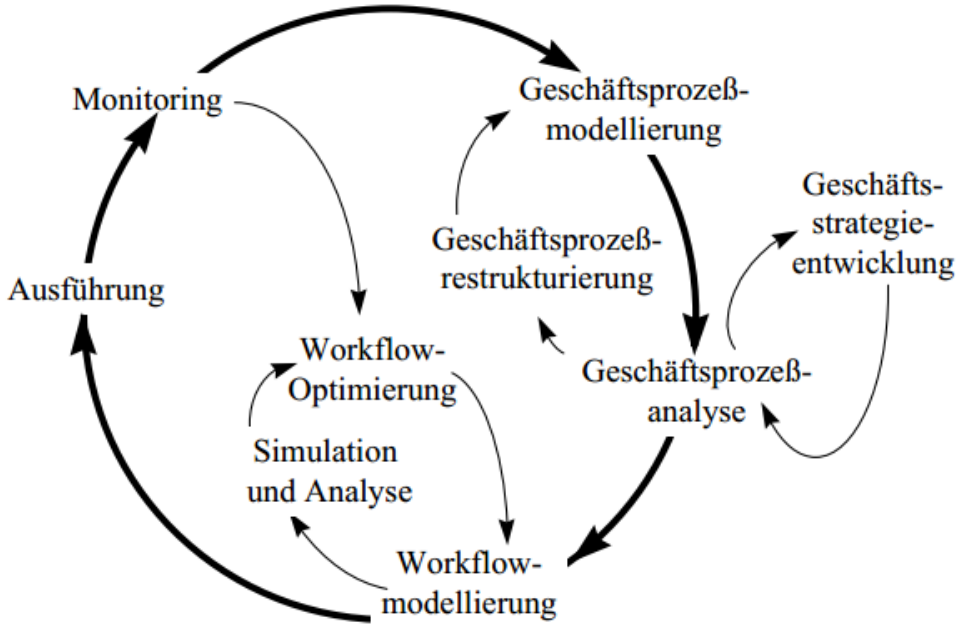


Abbildung 3: Workflow Life-Cycle-Modell

[Quelle: **Gadatsch 2008**, S. 75]

Zuletzt werden die Workflows im dritten Teilzyklus ausgeführt und die entstehenden Workflowinstanzen einem Monitoring, d.h. einer Überwachung hinsichtlich verschiedener Kriterien, unterzogen. Aus diesem Monitoring lassen sich ggf. neue Optimierungen ableiten, welche dann einen erneuten Anstoß für den zweiten Teilzyklus darstellen.[12]

Workflow-Systeme dominieren heute die Welt der automatischen Prozessausführung. Grund dafür ist, dass strukturierte Workflows bei der technischen Umsetzung den Standard innerhalb von klassischem Geschäftsprozessmanagement darstellen und dass das klassische Geschäftsprozessmanagement als Werkzeug angesehen wird, mit dessen Hilfe Unternehmensressourcen optimaler genutzt werden können. Das Geschäftsprozessmanagement setzt dazu auf Standardisierung, Spezialisierung und Automatisierung.[13]

Es existieren bereits verschiedene Veröffentlichungen zu Vorgehensweisen und Methodiken bei der Anwendung von Geschäftsprozessmanagement. So gibt **Freund et al. 2010** einen Überblick über die Verwendung von strategischen, operativen und technischen Prozessmodellen. In **Slama et al. 2011** wird das sog.

[12] Vgl. **Gadatsch 2008**
[13] Vgl. **Bider et al. 2012**, Kap. 1

IBPM-Framework vorgestellt, welches mit seinen Methoden *Prozessorientierte Analyse und Design* sowie *Serviceorientierte Analyse und Design* zehn Säulen definiert, die ein BPM-Projekt in seine wichtigsten inhaltlichen Bereiche unterteilt.[14] Eine Art *Kochrezept* zur Entwicklung konsistenter *End-to-End* BPMN-Modelle liefert **Silver 2011**.

2.5 Kritik

Fast immer, wenn heutzutage von Geschäftsprozessmanagement die Rede ist, ist damit der erläuterte Ansatz der Prozessmodellierung (oft mit Hilfe der BPMN) und der anschließenden Ausführung dieser Prozesse durch strukturierte Workflows gemeint, wie es auch in vielen Geschäftsprozessmanagement-Systemen (BPMS) realisiert wird. Im weiteren Verlauf der Arbeit wird dieses Vorgehen auch als *klassisches BPM-Vorgehen* referenziert.

Der Grund dafür, dass sich dieses Vorgehen zum Standard in der heutigen BPM-Welt entwickelt hat, liegt u.a. an der Tatsache, wie praktikabel sich vor allem routinemäßige Prozesse darüber standardisieren, spezialisieren und automatisieren lassen. Doch es mehrt sich auch die Kritik an diesem Ansatz. Es wird hinterfragt, ob dieses Vorgehen tatsächlich für alle, oder zumindest viele Arten von Geschäftsprozessen, anwendbar ist. Insbesondere im Geschäftsumfeld moderner Unternehmen mit stark wissensgetriebener Arbeit ist es denkbar, dass sich nur rund 20% – 40% aller Prozesse zur Aufnahme durch hochdetaillierte Notationen wie der BPMN und zur Ausführung über strukturierte Workflows eignen. Wissensgetriebene Arbeit, siehe auch Kapitel 3.1- Knowledge Work und Knowledge Worker, wird häufig von menschlichen Entscheidungen und äußeren Einflüssen getrieben, wodurch sich die Frage stellt, inwieweit sich ein solcher Prozess, der eigentlich gar keinem Prozessfluss folgt, überhaupt modellieren lässt.[15] Obwohl sich daraus der Bedarf nach alternativen BPM-Ansätzen ergibt, bleibt für diese Alternativen, wie z.B. für Case Management, aufgrund der Dominanz des klassischen BPM-Ansatzes nur wenig Platz.[16]

Ein weiterer, wichtiger Kritikpunkt ist der Mangel an Flexibilität, den klassisches Geschäftsprozessmanagement mit sich bringt. Der Weg von der Prozessmodellierung, die zumeist von Prozessspezialisten vorgenommen wird, bis hin zur techni-

[14] Vgl. **Slama et al. 2011**, S. 75
[15] Vgl. **Swenson 2010**, S. 29-31
[16] Vgl. **Bider et al. 2012**, Kap. 1

schen Realisierung der zugehörigen strukturierten Workflows, ist extrem zeit- und kostenintensiv. Es wäre unrealistisch anzunehmen, dass ein über Workflows auszuführender Prozess mit einem Male definiert und technisch umgesetzt werden kann, ohne dabei mehrere Verbesserungsschleifen zu durchlaufen. Veränderungen an einem einmal entwickelten Prozess bedeuten ebenfalls zeit- und kostenintensive Anpassungen über das verwendete BPMS. In der erwartet hoch dynamischen Geschäftswelt der Zukunft wird es aber möglicherweise an der Zeit mangeln, einen Prozess ausführbar zu machen, bevor sich die Anforderungen an den Prozess grundlegend geändert haben.[17] Daraus resultierte eine fortwährende Schleife bei der Prozessentwicklung, ohne dass ein Prozess jemals wirklich auf dem aktuellen Stand wäre. Ein Mangel an Flexibilität führt zu einem Mangel an Nutzbarkeit der Prozesse.[18] Es ist aber genau diese Flexibilität, die Unternehmen einer schnelllebigen Umwelt benötigen, um effizient auf neue Technologien, Marktanforderungen und Gesetze reagieren zu können.[19] Weitere Diskussionen um das Thema *Flexibilität* von Prozessmodellen und Workflows finden sich in **Bernstein et al. 1998** und **Siebert et al. 1998**.

Als Grund für den Mangel an Flexibilität gelten oft die strengen Prozessdefinitionen mit den eindeutig festgelegten Ausführungspfaden, die ein Prozess durchlaufen kann. Bereits bei der Modellierung von Prozessen stellt sich die Frage nach der optimalen Detaillierung der Modelle. Ein möglichst geringer Detaillierungsgrad führt eher zu einer idealisierten Version des tatsächlich gelebten Prozesses. Auf der anderen Seite führt ein sehr hoher Grad an Detaillierung mit allen denkbaren Ausnahmepfaden schnell zu Prozessmodellen, die viel zu groß werden, als dass man sie effizient verwenden, umsetzen und warten könnte.[20] Welche Probleme schon der Versuch der Erstellung eines so umfangreichen Prozessmodells mit sich bringt, wird ausführlich beschrieben in **Strong et al. 1995**. Besonders für wissensgetriebene Arbeit ist das zweidimensionale Designformat der zumeist verwendeten Modellierungsnotationen wie BPMN mit ihren strengen Ausführungspfaden ungeeignet.[21] Die einzige Möglichkeit eine Prozessinstanz voranzutreiben besteht dabei in dem sog. Routing, welches, i.d.R. durch Pfeile dargestellt, von einem Element (z.B. einer Aktivität oder einem Ereignis) zum nächsten führt. Daraus ergeben sich verschiedene Problematiken:[22]

[17] Vgl. **Bider et al. 2012**, Kap. 3
[18] Vgl. **van der Aalst et al. 2005**, S. 3
[19] Vgl. **Casati et al.**, Kap. 1
[20] Vgl. **van der Aalst et al. 2005**, S. 2
[21] Vgl. **Swenson 2012a**, Kap. 1
[22] Vgl. **van der Aalst et al. 2005**, S. 3

- Zerlegung von Arbeit in Workflow-Aktivitäten: Was für ein WFMS eine atomare Aktivität darstellt, ist oft *in Wahrheit* viel fein-granularer.
- Fehlende Trennung von Ausführung und Autorisierung der Arbeit.
- Die Steuerung von Workflows rückt in den Vordergrund und der eigentliche Inhalt damit in den Hintergrund.
- Fokussierung auf das, was getan werden sollte und nicht auf das, was getan werden könnte. Daraus resultieren unflexible Workflows.

Zusammenfassend fokussiert sich die Kritik am klassischen BPM-Ansatz aus zweidimensionaler Prozessmodellierung und Umsetzung durch strukturierte Workflows auf fehlende Flexibilität, resultierend aus zu restriktiven Ablaufpfaden. Dies nimmt Unternehmen die Fähigkeit zur Anpassung, um auf externe Veränderung, z.B. mit Innovation, zu reagieren.[23] Es ist jedoch genau diese Agilität, die eine der wichtigsten Eigenschaften von Unternehmen der nächsten Generation darstellt, um in einer hoch dynamischen Geschäftswelt zu bestehen.[24]

[23] Vgl. **Swenson 2010**, S. 94
[24] Vgl. **Bider et al. 2012**, Kap. 1

3 Case Management

Case Management, oder auch *Case Handling*, soll dort ansetzen, wo sich die im Kapitel zuvor beschriebenen Grenzen von traditionellem Geschäftsprozessmanagement mit seinen strukturierten Workflow-Systemen auftun. Dazu werden im folgenden Kapitel das Prinzip und die Grundlagen des Case Managements vorgestellt. Als Erstes wird dabei auf einen speziellen Arbeitstypus, die sog. Knowledge Work, die den gesamten Case Management Ansatz überhaupt erst notwendig macht, eingegangen. Der sich daraus ergebende Grundgedanke von Case Management wird im zweiten Unterpunkt dieses Kapitels aufgezeigt. Anschließend folgt eine Einführung in das Metamodell und die Funktionsweise von Case Management, wie es bereits in wissenschaftlichen Arbeiten vorgestellt wurde. Danach wird die Unterteilung von Case Management in *Adaptive Case Management* und *Production Case Management* erläutert.

3.1 Knowledge Work und Knowledge Worker

Die in den Unternehmen ablaufenden Prozesse lassen sich, je nach Betrachtung, in verschiedene Kategorien unterteilen. Betrachtet man den Wiederholungsgrad von Prozessen, lässt sich die geleistete Arbeit in Routinearbeit (hoher Grad an Wiederholung) und wissensgetriebene Arbeit (geringer Grad an Wiederholung), auch Knowledge Work, separieren. Routinearbeit bedeutet zwar nicht, dass die entsprechenden Prozesse bei jeder Ausführung geradezu mechanisch und bis ins kleinste Detail gleich ablaufen, zumindest aber, dass der Grad an Gleichheit unter den Instanzen so hoch ist, dass es sich lohnt ein konkretes und detailliertes Ablaufmuster für solche Prozesse zu identifizieren. Diese Prozesse können im Allgemeinen mit traditionellen Mechanismen wie Workflow-Systemen automatisiert werden.[25] Knowledge Work hingegen, auch bezeichnet als *Case Work*, wird definiert als Arbeit, die keinem gleichbleibenden Ausführungspfad folgt, sondern deren Ablauf stark von der gegebenen Situation abhängt und sich mit dieser verändert. Es kommt im Rahmen von Knowledge Work also vor, dass diese erst voran- und dann wieder einen oder mehrere Schritte rückwärtsgeht, bevor sie ihr angestrebtes Ziel erreicht.[26] Ein weiteres Merkmal von Knowledge Work ist deren Ausführung durch Experten, sog. *Knowledge Worker*. Diese entscheiden anhand ihrer spezifischen Erfahrungen darüber, wie ein jeweiliger Case, also eine konkret vor-

[25] Vgl. **Swenson 2010**, S. 11
[26] Vgl. **Gartner 2012**, S. 3

liegende Knowledge Work Instanz, Schritt für Schritt zu seinem Ziel geführt werden soll. Dies geschieht also in einer Art und Weise, wie es durch einen starr vorgegebenen Prozessfluss nicht möglich wäre. Zudem ist Knowledge Work häufig kollaborativ, also in Bearbeitung von mehreren Mitarbeitern oder Mitarbeitergruppen. Durch diese Merkmale wird jede im Umfeld von Knowledge Work durchgeführte Arbeit zu einem Einzelfall, der ein komplexes Zusammenspiel aus fachlichen Informationen, Menschen, dem Unternehmen und mitunter auch behördlichen Regulierungen darstellt.[27] Statt um einen Prozess im klassischen Sinne, handelt es sich bei Knowledge Work also eher um ein ständiges Einstellen auf sich ändernde Situationen.[28] Daher verwundert es auch nicht, dass als Beispiel für Knowledge Work (und später Case Management) immer wieder die Behandlung von Patienten in einem Krankenhaus herangezogen wird. Trotz gleicher Krankheitsbilder kann die angewandte Therapie bei zwei Patienten einen völlig unterschiedlichen Verlauf nehmen. Als Experte entscheidet der behandelnde Arzt, in Abhängigkeit von begleitenden Faktoren wie Alter und Geschlecht des Patienten oder dem Krankheitsverlauf, über durchzuführende Schritte, wie Medikation oder weitere Diagnosen. Ein vorher festgelegter Verlauf der gesamten Behandlung ist hier nicht anwendbar. Zudem demonstriert das Zusammenspiel zwischen verschiedenen Stationen, welche ein Patient möglicherweise in einem Krankenhaus durchläuft, wie der Diagnostik, der Pflege oder der Chirurgie, den kollaborativen Faktor von Knowledge Work. Weitere häufig genannte Beispiele für Knowledge Work sind das Abwickeln von Versicherungsanträgen und die Bearbeitung von Kundenreklamationen.

3.2 Grundgedanke

Im vorherigen Kapitel wurde näher auf den sog. Knowledge Worker und die von ihm/ihr geleistete Knowledge Work, d.h. wissens- bzw. erfahrungsbasierte Arbeit eingegangen. Geschäftsprozesse, die sich als Knowledge Work definieren lassen, lassen sich nur schwer mit klassischen BPM-Methoden technisch realisieren. Dies liegt vorwiegend an dem mangelnden Grad an Flexibilität, den Workflow-Systeme, wie sie im traditionellen BPM-Ansatz verwendet werden, liefern. Aus diesem Grund hat sich für derartige Geschäftsprozesse die Unterstützung durch Case Management entwickelt.

[27] Vgl. **Gartner 2012**, S. 3
[28] Vgl. **Swenson 2010**, S. 7

Das Prinzip des Case Managements im Umfeld von Unternehmensprozessen besteht schon seit mehreren Jahren und wurde in verschiedenen wissenschaftlichen Arbeiten thematisiert. So liefert **van der Aalst et al. 2001** ein Konzept inkl. Metamodell zur Steuerung von *Cases* durch Datenobjekte anstatt durch Kontrollflüsse und stellt die zugehörige Implementierung *FLOWer* vor. Dieses Konzept wird in **van der Aalst et al. 2005** verfeinert und später in diesem Kapitel erläutert. Eine Gegenüberstellung von Workflow- und Case Handling-Systemen findet sich in **Reijers et al. 2003**. Die Anwendung von Case Handling in einem konkreten Projekt und die dort gemachten Erfahrungen zeigt **van der Aalst et al. 2003**.

Case Management versteht sich als eine Disziplin des Geschäftsprozessmanagements, in der es darum geht, Prozesse abzubilden und auszuführen, die einen höheren Grad an Flexibilität aufweisen als weitgehend standardisierbare Prozesse. Dabei handelt es sich vorwiegend um Prozesse, bei denen viele Alternativpfade möglich sind oder bei denen es nicht vordergründig auf die Reihenfolge der auszuführenden Aktivitäten ankommt. Derartige Anforderungen können von klassischen Workflowlösungen oft nur schlecht oder gar nicht umgesetzt werden. Während beim klassischen Geschäftsprozessmanagement der sog. Kontrollfluss („auf Aktivität A folgt Aktivität B, dann Aktivität C oder D“) die Steuerung einer Prozessinstanz übernimmt, stellt Case Management die Prozessdaten in den Mittelpunkt einer Prozessinstanz und entscheidet anhand des Vorhandenseins oder des Nicht-Vorhandenseins von Daten, ob eine Aktivität ausführbar, gesperrt, wiederholbar oder überspringbar ist. Der Verzicht auf vorgegebene Sequenzflüsse wirft die Frage auf, inwieweit sich BPMN zur Erfassung solcher Prozesse eignet. Diese Fragestellung wird ausführlich diskutiert in **Swenson 2012a**. Tatsächlich ist das einzige Element der BPMN, welches ein unstrukturiertes Vorgehen während eines Prozesses unterstützt der *Ad-hoc Teilprozess*[29]. Abbildung 4 zeigt einen Ad-Hoc Teilprozess mit mehreren Aktivitäten, welche über keinen Prozessfluss untereinander verfügen und somit in beliebiger Reihenfolge durchgeführt werden können. Gesteuert wird der Prozess einzig über die Ein- und Ausgabedaten der einzelnen Aktivitäten, die hier als Datenobjekte abgebildet sind.

[29] Vgl. **OMG 2011**, S. 181

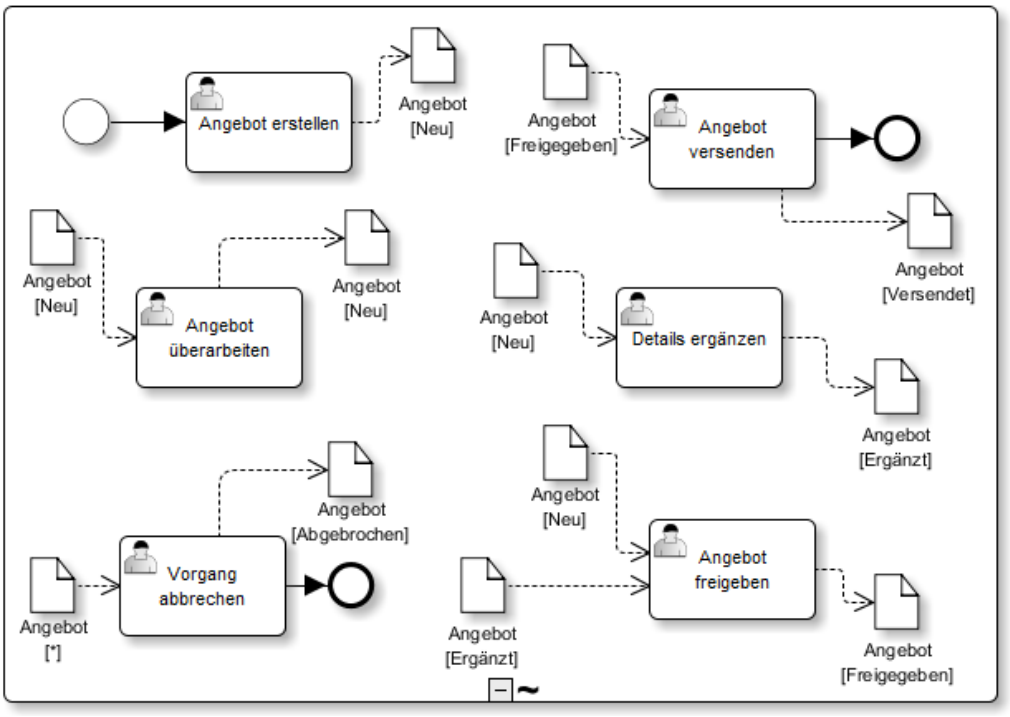


Abbildung 4: Ad-Hoc Teilprozess

Case Management bedient sich zur Prozesssteuerung verschiedener Zustände, die von Prozessinstanzen, Aktivitäten und Datenobjekten eingenommen werden können. Für die meisten Prozesse lässt sich ein zentrales Datenobjekt identifizieren, dessen verschiedene Zustände Auskunft darüber geben, an welcher Stelle sich ein laufender Case befindet. Wie ein zentrales Datenobjekt von einem Zustand in einen anderen gelangt, hängt vorwiegend von dessen eigenen Attributen ab. Den Auslöser zum Verändern eines Zustandes bilden dabei unter anderem zeitliche Ereignisse (z.B. bei einer Eskalation) oder Benutzeraktionen (z.B. das Klicken eines Buttons). Für die Umsetzung Case Management-gestützter Prozesse bedarf es also neben Prozess- und Datenmodellierung auch der fachlichen Definition von Zuständen und deren Übergängen untereinander.

Es sind jedoch nicht immer zwangsläufig hochkomplexe Prozesse, die den Einsatz von Case Management erfordern. Abbildung 5 zeigt beispielsweise, wie mit Hilfe von Case Management eine einfache fachliche Anforderung erfüllt wird, nach der es möglich sein soll, ein bearbeitetes Angebot im zweiten Durchlauf direkt, also ohne weitere Prüfung, freizugeben (z.B. nach der Korrektur einfacher Rechtschreibfehler).

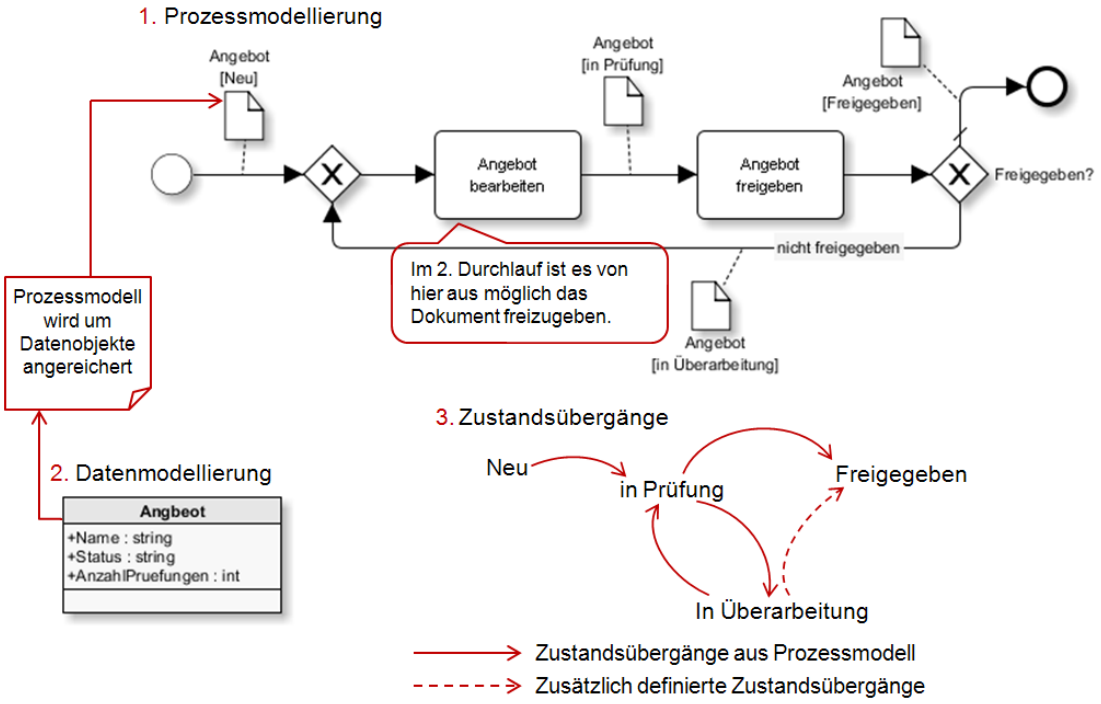


Abbildung 5: Ein Case Management Beispiel

Aus dem Prozessmodell allein ist diese Vorgehensweise nicht ersichtlich; erst das zugehörige Zustandsübergangsdiagramm zeigt diese Möglichkeit auf (gestrichelter Pfeil vom Status „in Überarbeitung“ zu Status „Freigegeben“). Hier ist es weiterhin der Prozess, der zum überwiegenden Teil den Bearbeitungsablauf steuert. Die Steuerung durch Daten ist hier nur ein Ausnahmefall.

Denkbar sind auch gezielte Mischformen von Prozess- und Datensteuerung. So stellt sich häufig die Frage danach, wie mit einer von extern verursachter Veränderung von Daten an einer dafür unvorhergesehenen, d.h. nicht modellierten, Stelle umgegangen werden soll. Die Änderung von Daten macht es ggf. nötig, bestimmte Aktivitäten zu wiederholen oder andere Prozessdaten erneut zu berechnen. Die Berücksichtigung einer solchen Datenänderung mit Hilfe eines jederzeit möglichen Zwischenereignisses würde bereits das fachliche Prozessmodell der klassischen Vorgehensweise deutlich komplexer und unübersichtlicher machen.

3.3 Metamodell

Nach der Vorstellung der Motivation von Case Management und dem Umfeld, in dem sich dessen Anwendung anbietet, geht es in dem folgenden Kapitel um einen

konkreten Ansatz, wie dieses Prinzip in ein IT-System umgewandelt werden kann. Dazu soll das Case Management Metamodell nach **van der Aalst et al. 2005** vorgestellt werden.

In dem veröffentlichten Modell stehen *Case Definitionen* (im Modell *case definitions*, s. Abbildung 6) im Mittelpunkt. Diese *Case Definitionen* sind entweder komplex oder atomar. Komplexe Case Definitionen (im Modell *complex case definitions*) bestehen aus einer oder mehreren Case Definitionen während atomare Case Definitionen nicht weiter verschachtelt sind und daher als *Aktivitäten* bezeichnet werden.

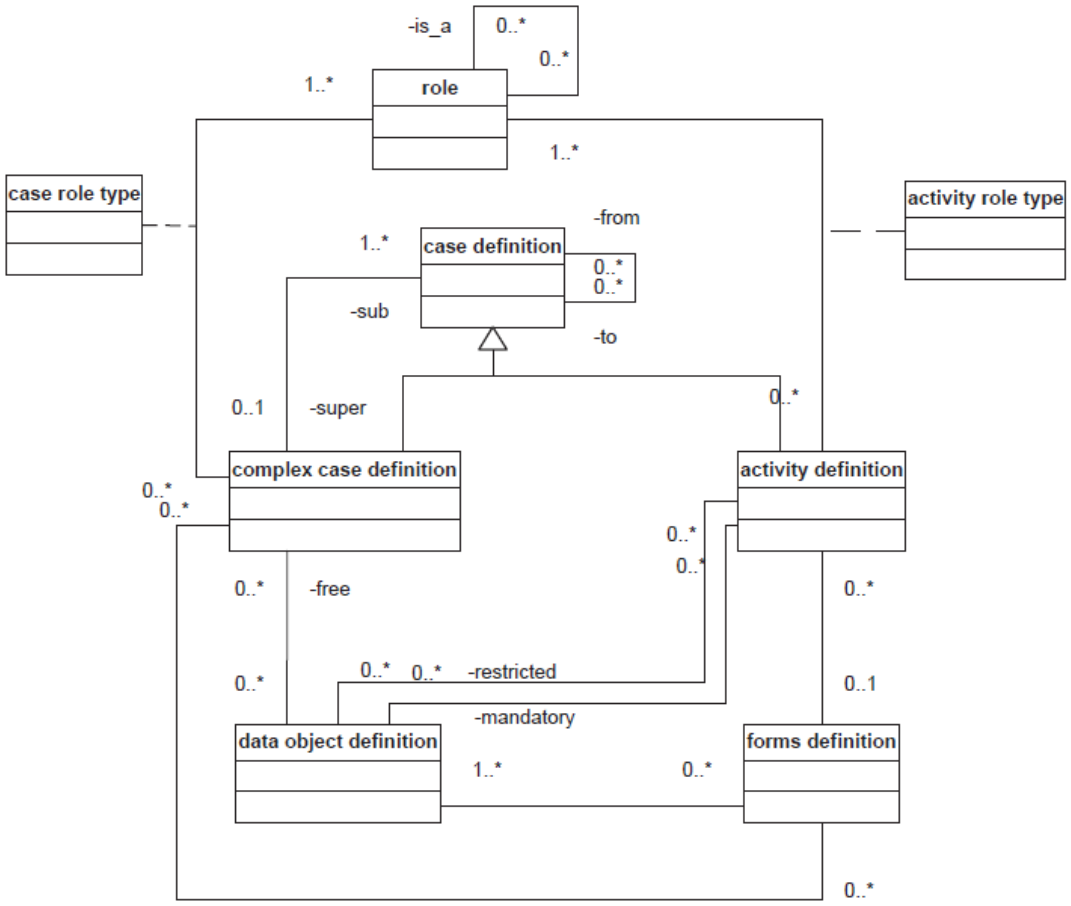


Abbildung 6 : Metamodell Case Handling

[Quelle: **van der Aalst et al. 2005,** S.9]

Aktivitäten stehen im engen Zusammenhang mit Daten, im Modell als *data object definition* bezeichnet. Ein mit einer Aktivität verknüpftes Datum ist entweder verpflichtend (*mandatory*) oder restriktiv (*restricted*) zu dieser Aktivität. Verpflichtende Daten müssen zwingend gesetzt sein, um die entsprechende Aktivität abzuschlie-

ßen, können aber auch schon in früheren Aktivitäten, wann immer der Knowledge Worker es für angebracht hält, bearbeitet werden. Restriktive Daten hingegen können nur während der mit ihnen verknüpften Aktivität gesetzt oder verändert werden. Ein weiteres Element in dem vorgestellten Metamodell sind Formulare. Sie machen dem Benutzer die Datenobjekte zugänglich und ermöglichen damit die Veränderung der Werte dieser Daten. Aber auch Aktivitäten können mit Formen verknüpft sein. Diese Formen zeigen dann *mindestens* die mit den Aktivitäten verknüpften Daten an. Denkbar ist aber auch, dass Pflichtfelder nachgelagerter Aktivitäten (Aktivitäten stehen untereinander in einer von-zu-Beziehung) gleich mit angezeigt werden, um dem Knowledge Worker ein möglichst flexibles Arbeitsumfeld zu ermöglichen.

Das Metamodell beinhaltet außerdem Rollen in zwei verschiedenen Ausführungen. Case Rollen sind mit komplexen Case Definitionen verknüpft und repräsentieren Stellen in einer Organisation, die für diesen Case verantwortlich sind. Dies könnten bspw. *Manager* oder *Vorgesetzte* sein. Rollen hingegen, die mit Aktivitäten verknüpft sind, sind klassischerweise die verschiedenen Rechte wie *ausführen*, *überspringen* oder *wiederholen*, die auf atomare Aktivitäten angewendet werden können.

3.4 Funktionsweise

Das im vorhergehenden Kapitel vorgestellte Case Management Metamodell ist mit einem Funktionsprinzip ausgestattet, welches den Ablauf eines Cases, also die Bearbeitung eines Geschäftsvorfalls, beschreibt. Die entsprechende Funktionsweise soll in diesem Kapitel kurz erläutert werden.

Das Auftreten eines bestimmten Geschäftsereignisses instanziiert einen Case und aktiviert damit alle zu dem Case gehörenden Aktivitäten, die zu diesem Zeitpunkt den Zustand *bereit* erhalten. Durch die Auswahl einer Aktivität zur Bearbeitung durch einen Knowledge Worker erhält diese den Zustand *in Bearbeitung*. Die Gesamtheit aller möglichen Zustände von Aktivitäten zeigt die Abbildung 7.

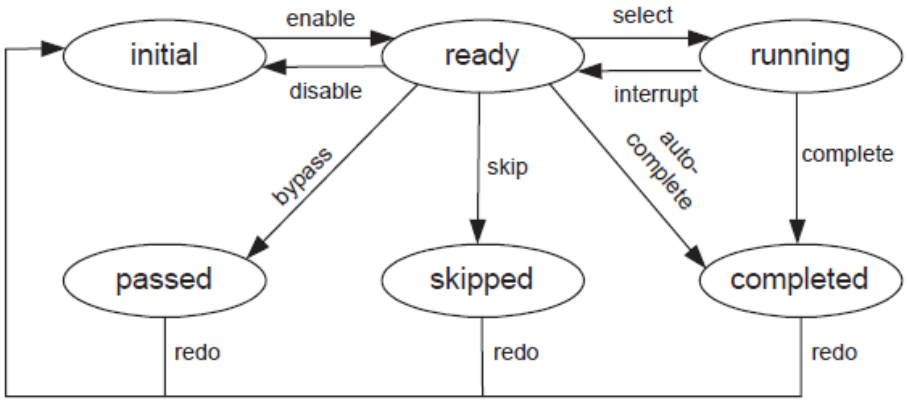


Abbildung 7: Zustände von Aktivitäten

[Quelle: **van der Aalst et al. 2005**, S. 16]

Nach der Bearbeitung einer Aktivität ist diese entweder abgeschlossen oder wurde abgebrochen. Im Falle eines Abbruchs bleiben jedoch alle gespeicherten Daten erhalten und die Aktivität wechselt wieder in den Zustand *bereit*. Sind hingegen alle Daten zu einer Aktivität vorhanden, wechselt sie automatisch in den Zustand *abgeschlossen*.

Eines der wichtigsten Ziele von Case Management ist es, dem Knowledge Worker eine größtmögliche Flexibilität bei seiner Arbeit zu gewährleisten. Entscheidet der Knowledge Worker, dass eine Aktivität mal nicht zur Bearbeitung des vorliegenden Cases benötigt wird, kann er diese ganz einfach auslassen. Sie erhält den Zustand *ausgelassen*. Wenn sogar ein ganzer Pfad eines Cases nicht verfolgt werden soll, dann werden die Aktivitäten auf diesem Pfad als *umgangen* (im Modell *passed*) definiert und erhalten den entsprechenden Zustand. Genauso wie abgeschlossene, können auch ausgelassene oder umgangene Aktivitäten mit Hilfe der Rolle *wiederholen* erneut in den Zustand *bereit* versetzt und damit zur Bearbeitung möglich gemacht werden.

Analog zu Aktivitäten erhalten auch die an einem Case beteiligten Datenobjekte verschiedene Zustände und durchlaufen damit einen gewissen Lebenszyklus. Folgende drei Zustände können Datenobjekte einnehmen:

Zustand	Beschreibung
Undefiniert	Initialer Zustand, den Datenobjekte nach ihrer Erstellung erhalten.
Definiert	Zustand, sobald für das Datenobjekt ein Wert gesetzt wurde.
Unbestätigt	Zustand, wenn eine Aktivität, für die das Datenobjekt ein Pflichtfeld ist, wiederholt wird. Der Wert des Datenobjektes bleibt erhalten, muss aber erneut bestätigt werden.

Tabelle 2 : Zustände von Datenobjekten

Case Management verfolgt den Ansatz mehr daten-, und weniger prozessgetrieben zu sein. Möglich wird das dadurch, dass die Datenobjekte, mit Ausnahme solcher mit restriktiven Beziehungen zu Aktivitäten, jederzeit verändert werden können. Durch die Änderung von Datenobjekten bzw. von deren Werten ändern sich auch die Zustände von Aktivitäten. Dies ist bspw. der Fall, wenn durch das Setzen eines Wertes für ein Datenobjekt eine Aktivität automatisch abgeschlossen wird, da es sich um das letzte Datenobjekt ohne Wert handelte. Außerdem können Datenobjekte als Vorbedingung für Aktivitäten fungieren. Dadurch kann das Setzen eines Wertes für ein Datenobjekt, wenn es die einzige oder letzte zu setzende Vorbedingung für eine Aktivität ist, auch dazu führen, dass diese Aktivität in den Zustand *bereit* versetzt wird. Die Zustände von Datenobjekten haben demnach direkten Einfluss auf die Zustände der Aktivitäten. Der Zustand des gesamten Cases wiederum ist sowohl von den Zuständen der Datenobjekte als auch von den Zuständen der Aktivitäten abhängig. Folglich sind es vorrangig die Daten, die einen Case steuern und ihn durch seine verschiedenen Zustände führen. Damit wird eine Case Management Kernessenz, nämlich die Datengetriebenheit, realisiert.

Eine weitere Anforderung an ein Case Management System ist die Trennung von Arbeitsausführung und der Autorisierung. Während sich diese beiden Aspekte bei traditionellen Workflow-Mechanismen überschneiden, sind sie bei Verwendung des beschriebenen Metamodells und der zuletzt erläuterten Funktionsweise strikt voneinander getrennt. Die Bearbeitung von Cases geschieht hier durch das Auswählen eines geeigneten Cases, z.B. nach Ausführung einer Suchfunktion, und anschließender Erledigung von Aktivitäten des gewählten Cases, die sich im Status *bereit* befinden. Die Autorisierung, d.h. bspw. welcher Benutzer welche Cases sehen und welche Aktivitäten bearbeiten darf, wird einzig über das vorgestellte Rollensystem realisiert.

3.5 Adaptive Case Management

Case Management lässt sich in die zwei Kategorien *Adaptive Case Management* und *Production Case Management* unterteilen.[30] Dabei meint Ersteres eine deutlich offenere Form mit noch mehr Freiheiten für den Knowledge Worker und entspricht dem im Kapitel zuvor erläuterten Funktionsprinzip. Wird heutzutage lediglich von *Case Management* gesprochen, so ist damit i.d.R. ACM gemeint. In **Gartner 2012** wird ACM auch als *besonders komplexe Case-Arbeit*[31] beschrieben, die während ihrer Ausführung ständig angepasst wird. Im Gegensatz zum später erläuterten PCM, verzichtet ACM fast vollständig auf die Steuerung anhand eines vorgegebenen Prozessflusses. Stattdessen konzentriert es sich darauf, dem Knowledge Worker alle verfügbaren Ressourcen an die Hand zu geben, damit dieser sie zur richtigen Zeit einsetzen kann.[32] Dazu bedient sich ACM sog. Templates, die für die Abarbeitung von Cases herangezogen werden. Ein Template vereint alles, was für solch eine Abarbeitung benötigt wird oder benötigt werden könnte. Dazu gehören durchzuführende (Standard-)Aufgaben, die Prozessdaten und Platzhalter für Dokumente, deren Erstellung oder Verarbeitung während der Abarbeitung erwartet wird.[33] Templates stellen jedoch nur eine Art Richtlinie zur Durchführung eines Cases dar und sind jederzeit durch den Knowledge Worker anpassbar. Wird beispielsweise eine konkrete Aufgabe immer wieder im Umfeld eines Prozesses benötigt, so sollte sie Teil des entsprechenden Templates werden. Auf diese Weise werden das Wissen und die Erfahrung der Knowledge Worker gespeichert und ist für Vertreter, Nachfolger oder weitere Kollegen verfügbar.

Die Tatsache, dass zwischen den Aktivitäten innerhalb eines Templates keine oder nur lose Verbindungen bestehen, und somit so gut wie kein vorgegebener Prozessfluss vorliegt, macht ACM aber noch nicht *adaptiv*, d.h. anpassungsfähig. Die jederzeit mögliche Veränderung von Templates durch die Knowledge Worker ist ein erster Schritt in diese Richtung, doch der Grundstein zur Erreichung dieser Anpassungsfähigkeit liegt bereits in der Prozess- bzw. Case-Definition. Statt wie beim strukturierten BPM-Vorgehen, oder im Rahmen von PCM, bereits im Vorfeld ein Prozessmodell für alle aufkommenden Prozessinstanzen zu modellieren, wird im ACM das Vorgehen für jeden einzelnen Case erst dann festgelegt, wenn dieser Case tatsächlich vorliegt. Der Knowledge Worker selbst stellt anhand seines eigenen Ermessens die benötigten Aufgaben, Daten und Dokumente, bzw. deren

[30] Vgl. **Swenson 2012b**
[31] Vgl. **Gartner 2012**, S. 9
[32] Vgl. **Swenson 2010**, S. 29
[33] Vgl. **Swenson 2010**, S.51

Platzhalter, für die Bearbeitung des Cases zusammen und bedient sich dabei aus einer Bibliothek solcher Artefakte. Daraus entsteht sozusagen ein Case-spezifisches Prozess- oder Vorgehensmodell, welches u.U. nur ein einziges Mal zum Einsatz kommt, dafür aber exakt auf die Anforderungen dieses Cases zugeschnitten ist. Kommt ein Modell mehrere Male erfolgreich zum Einsatz, sollte daraus ein Template werden.[34] Die Modellierung geschieht hier also direkt vor, bzw. sogar simultan mit der Ausführung der Arbeit und liegt in der Hand des Knowledge Workers, also demjenigen, der vermutlich am besten weiß, was zu tun ist. Damit realisiert ACM den Umschwung von der Erwartung (festgelegtes Prozessmodell für alle Instanzen) hin zur Anpassung (eigenes Vorgehen für jede Instanz), also „von Antizipation zu Adaption“[35]. Dies gilt als eine der Hauptanforderungen, um in einer modernen, dynamischen Geschäftswelt bestehen zu können.

Um es dem Knowledge Worker zu ermöglichen für jeden vorliegenden Case schnell ein passendes Prozessmodell aus Aktivitäten, Daten und Dokumenten zusammenzustellen, ohne dass dabei jedes Mal immens viel Aufwand betrieben werden muss, bedarf es einer entsprechenden Notation. Es stellt sich die Frage, ob die BPMN mit ihrer zweidimensionalen Natur und ihrem umfangreichen Repertoire an Elementen für diese Aufgabe geeignet ist. Die Anforderungen an eine Notation zur Abbildung von Cases im Rahmen von ACM könnten wie folgt aussehen:[36]

- Möglichkeit einen Prozess schnell und unkompliziert durch den Case Manager modellieren zu können (wichtiger als einen Prozess bis ins kleinste Detail spezifizieren zu können).
- Prozessmodellierung darf Fähigkeiten des normalen Nutzers nicht übersteigen.
- Keine versteckten Annahmen.
- Sehr einfache Sprache, leicht zu verstehen. Änderungen ziehen keine Inkonsistenzen nach sich.
- „Weniger ist mehr“.

Als ein Hauptziel von Geschäftsprozessmanagement wird immer wieder die Realisierung von Verständnis, Transparenz und Kontrolle von Geschäftsprozessen ge-

[34] Vgl. **Swenson 2012a**, Kap. 4
[35] Vgl. **Swenson 2010**, S. 41
[36] Vgl. **Swenson 2012a**, Kap. 4

nannt und damit sollten dies auch Ziele von (Adaptive) Case Management sein. Bei traditionellen BPM-Ansätzen werden diese Ziele vorwiegend mit Hilfe des für alle Instanzen gültigen Prozessmodelles realisiert. Es liegt auf der Hand, dass dieses Modell für ein Verständnis des Prozesses sorgt. Transparenz entsteht dadurch, dass sich genau sagen lässt, an welcher Stelle im Prozessmodell sich die aktuell laufenden Instanzen befinden. Kontrolle wird durch Messungen ermöglicht, wie viel Zeit die Instanzen benötigt haben, um das Modell, oder auch Teilabschnitte zu durchlaufen. Es stellt sich jedoch die Frage, wie dieses Ziel im Rahmen von ACM, wo es kein generalisiertes Prozessmodell für sämtliche Instanzen gibt, erreicht werden kann. Zu jedem Prozess, also auch zu jedem Case, sollte es ein exakt definiertes Prozessziel geben, welches Auskunft darüber gibt, was getan werden muss, um das gewünschte Ergebnis des Prozesses zu erreichen.[37] Dieses Prozessziel sorgt zum Einen für das Verständnis für einen Prozess und ermöglicht zum Anderen die Kontrolle des Prozesses, indem nachverfolgt werden kann, ob und wie es nach Beendigung des Prozesses eingehalten werden konnte. Die außerdem gewünschte Transparenz entsteht durch das Monitoring laufender Cases, genauer gesagt durch das Monitoring von deren aktuellen und bereits durchlaufenden Zuständen.

3.6 Production Case Management

Production Case Management (PCM) stellt in seiner Grundidee einen Übergang von klassischen BPM-Ansätzen zum Adaptive Case Management dar. Zwar verfolgt PCM die grundlegenden Case Management Ideen, wie das Vorantreiben eines Prozesses durch dessen Daten und deren Veränderungen, sowie die konsequente Einbeziehung der Entscheidungen und Erfahrungen des Knowledge Workers, doch existieren auch markante Unterschiede zum zuvor erläuterten ACM. Nach **Swenson 2012b** bestehen diese Unterschiede vorwiegend in den folgenden Punkten:

- Die Entwicklung eines PCM-Systems folgt dem klassischen Software-Lebenszyklus aus Umsetzung, Testen und abschließend dem Ausliefern an den End-Nutzer.

- Bei der Entwicklung kommen Formalisierungen wie Prozessmodellierung und Programmierung zum Einsatz.

[37] Vgl. **Swenson 2010**, S.36

- Nach der Auslieferung der Anwendung besteht keine Möglichkeit zu deren Anpassung.

Während beim ACM erst zur Laufzeit entschieden wird, wie ein bestimmter Case verlaufen soll, folgt dieser im Rahmen von PCM einem oder mehreren zuvor definierten Prozessmodellen. Um den Bezug zum Case Management jedoch nicht zu verlieren, entscheidet weiterhin der Knowledge Worker darüber, welche Bestandteile eines zugrunde liegenden Prozessmodells er für einen bestimmten Case tatsächlich ausführt und welche er möglicherweise überspringt. Dadurch behält PCM die Fähigkeit auf die Unvorhersehbarkeiten einzelner Cases zu reagieren. Die verwendeten Prozessmodelle dienen dem Knowledge Worker daher mehr als Unterstützung in seiner Entscheidungsfindung. Er erhält, basierend auf der aktuellen Datenlage und dem Zustand des Cases, verschiedene Optionen, wie mit dem Case vorangegangen werden kann. Zwar hat er wenig Kontrolle über die zentral von Experten definierten Arbeitsabläufe, aber dennoch ist er maßgeblich an dem Ausgang eines Cases beteiligt. Verantwortlich für die unterschiedlichen Wege, die ein Case nehmen kann, ist er aber nicht.[38]

Was den Umgang mit Unvorhersehbarkeiten betrifft, verhält sich PCM also ähnlich wie ACM; anders sieht es da bei der Fähigkeit zur Anpassung aus.[39] Spezialisten, wie z.B. Prozessverantwortliche oder externe Berater, nehmen einen Prozess mit seinen möglichen Abläufen auf und definieren seine Anforderungen. Mit aufgenommen werden in der Definitionsphase eines Prozesses auch die unterschiedlichen Zustände, die ein Case durchlaufen kann. Diese Zustände beschreiben allerdings eher den aktuellen Status eines Cases, als dass sie Auskunft darüber geben, was als nächstes zu tun ist.[40] Die Anwendung, also der umgesetzte Prozess, wird anschließend gemäß diesen Anforderungen entwickelt und als fertiges Produkt an den Endnutzer, den Knowledge Worker, ausgeliefert.[41] Erweiterungen um vollständig neue Prozessabläufe oder auch nur das Hinzufügen einer weiteren möglichen Aktivität zieht ein aufwendiges und zeitintensives Refactoring der Anwendung nach sich.

In seiner Prozessorientierung platziert sich PCM damit zwischen dem annähernd prozesslosen ACM und stark prozessorientierten Konzepten wie *Human Process*

[38] Vgl. **Swenson 2012b**, Kap. *PCM In A Nutshell*
[39] Vgl. **Sem et al. 2012**, Kap. 5.3
[40] Vgl. **Swenson 2012b**, Kap. *PCM In A Nutshell*
[41] Vgl. **Swenson 2012a**, Kap. 3

Management (HPM) und *Process Driven Server Integration* (PDSI).[42] Mehr Informationen zu den zuletzt genannten Konzepten beschreibt **Swenson 2012c**.

„Zum Einsatz kommt PCM am besten dann, wenn die Anzahl der Knowledge Worker mit der gleichen Aufgabenstellung groß, das Aufgabengebiet ziemlich gut bekannt, aber der Prozess nicht vollständig vorhersagbar ist“[43] (*Übersetzung durch den Autor der Masterarbeit aus dem Englischen*). Die Anzahl der Knowledge Worker mit gleicher Aufgabenstellung sollte deswegen möglichst groß sein, damit sich die Kosten für die Entwicklung einer PCM-Anwendung durch die Verteilung auf dessen Anwender relativieren.

Als ein geeignetes Beispiel für die Anwendung von PCM führt **Swenson 2012b** den Reparaturdienst eines Fernsehanbieters an. Der Servicemitarbeiter, hier der Knowledge Worker, muss sich im Störungsfall zunächst ein Bild von der Lage vor Ort machen, um dann aus einer Reihe von Lösungsmöglichkeiten zu entscheiden, welche in diesem Fall die passendste ist. Für einen vordefinierten, unveränderlichen Prozess wie im *Human Process Management* ist diese Tätigkeit nicht geeignet, da sie dazu nicht vorhersehbar genug ist. Der Servicemitarbeiter versucht anhand seiner gewählten Methode das Problem zu beheben und sammelt dabei weitere Informationen über den vorliegenden Fall. Die Situation entwickelt sich vor Ort und der Servicemitarbeiter muss eventuell mehrere mögliche Lösungsansätze versuchen. Er ist jedoch nicht in der Position sich komplett neue Lösungsansätze vor Ort auszudenken und unterliegt damit der Restriktion vorgegebener Lösungsansätze, von denen bekannt ist, dass ihre Durchführung dem Betrieb des durchaus komplexen TV- und/oder Telefonsystems nicht noch weiter schadet.

PCM ist eine der neusten Entwicklungen auf dem Gebiet der Prozessautomatisierung und des Case Managements. Grundideen zu dessen Funktionsweise wurden in diesem Kapitel erläutert. Konkrete Modelle und Semantiken zur Ausführung von PCM sucht man in der Literatur jedoch noch vergeblich. Ab dem folgenden Kapitel 4 unternimmt diese Masterthesis den Versuch ein passendes Konzept zur Durchführbarkeit von PCM zu erstellen.

[42] Vgl. **Swenson 2012b**, Kap. *PCM In A Nutshell*

[43] Vgl. **Swenson 2012b**, (eigene Übersetzung). Originalwortlaut: *PCM is used when the number of knowledge workers doing the same job is large, and the domain relatively well known, but the process is not entirely predictable*.

4 Fachliche Aufnahme zur PCM-Unterstützung

Nachdem die bisherigen Kapitel dieser Masterarbeit ein grundlegendes Verständnis für die Technologie des Case Managements und dessen Notwendigkeit schaffen sollten, nähert sich der folgende Abschnitt der tatsächlichen Anwendung von Case Management zur Umsetzung von Geschäftsprozessen. Sollen Geschäftsprozesse mit Hilfe von PCM umgesetzt werden, so müssen diese Prozesse zunächst fachlich aufgenommen werden. Dieses Kapitel beschäftigt sich mit den Inhalten dieser fachlichen Aufnahme und erläutert, welche Informationen ein Production Case Management System (PCMS) benötigt, um diese Prozesse ausführbar zu machen. Dazu werden im Folgenden *Prozessmodelle*, *Prozessdaten* und *Prozesszustände* betrachtet und untersucht, wie diese Inhalte sich mit bereits bewährten Methoden, Notationen oder Diagrammtypen standardisiert erfassen lassen.

Die unterschiedlichen Konzepte von ACM und PCM haben auch Auswirkungen auf deren jeweilige fachliche Prozessaufnahme, was eine Abgrenzung an dieser Stelle nötig macht. Im weiteren Verlauf dieses Kapitels und dieser Arbeit wird daher ausschließlich auf die Anforderungen von PCM eingegangen. Für die fortan verwendete Terminologie bedeutet dies, dass das Wort *Prozess* nicht vollständig durch *Case* ersetzt wird. Prozesse im abstrakten Sinne, d.h. Prozessdefinitionen, werden weiterhin als *Prozess* bezeichnet, während konkrete Instanzen von Prozessen als *Cases* bezeichnet werden.

4.1 Prozessmodelle

Im Gegensatz zu ACM legt das Konzept des PCMs der Realisierung von Geschäftsprozessen strukturierte Prozessmodelle zugrunde. Diese Modelle stehen am Anfang der Entwicklung einer PCM-Anwendung und werden von Spezialisten und Prozessverantwortlichen erstellt. Wie bereits in Kapitel 2.4.1 beschrieben, werden zur Darstellung von Prozessabläufen im Wesentlichen die folgenden drei Elemente benötigt:

- *Aktivitäten*, zur Repräsentation durchzuführender Handlungen von Mensch oder System,
- *Ereignisse*, die das Eintreten von Zuständen abbilden, sowie

- *Gateways*, die den Fluss eines Prozesses steuern, indem sie ihn teilen oder wieder zusammenführen.

Als weit verbreiteter Standard in der Prozessmodellierung bietet die BPMN diese Elementtypen, neben weiteren Typen wie z.B. Artefakten, zur Erstellung von Prozessmodellen an.

Was zunächst nach klassischem Business Process Management klingt, unterscheidet sich dahingehend von diesem Ansatz, als dass die Prozessmodelle im Rahmen von PCM keine unumstößlichen Vorgaben darstellen. Viel mehr dienen sie dazu, den Nutzer einer PCM-Anwendung bei der Bearbeitung eines Cases zu unterstützen und ihm die Wahl der optimalen Vorgehensweise zu erleichtern. Auf die Definition dieser Prozessmodelle hat das erhebliche Auswirkungen. So besteht, im Gegensatz zum klassischen BPM, keine Notwendigkeit auf Vollständigkeit eines Prozessmodells. Darin nicht abgebildete Aktivitäten und Ablaufpfade führen nicht zwangsläufig zu fehlender Funktionalität. Das Ausblenden besonders seltener oder unwahrscheinlicher Prozessverläufe hält die Modelle übersichtlich und macht aus ihnen sogenannte „80%-Modelle“, die ganz gezielt keinen Anspruch darauf erheben einen Prozess vollständig abzubilden, jedoch die Entwicklung eines PCMS und dessen Nutzer umfassend unterstützen. Zusätzlich zu den seltenen oder unwahrscheinlichen Prozessverläufen können aber auch jene Prozessverläufe ausgeblendet werden, die gemäß einer Anforderung jederzeit möglich sein sollen und deren explizite Aufführung im Prozessmodell daher nicht immer nötig ist. Existiert beispielsweise im Rahmen der Umsetzung einer Kundenreklamationsbearbeitung die Anforderung, eine Reklamation jederzeit abschließen zu können (z.B. wenn der Kunde selbst das Problem als behoben meldet), würde es die Komplexität des zugrunde liegenden Prozessmodells deutlich erhöhen, wenn diese Möglichkeit tatsächlich an jeder beliebigen Stelle modelliert werden würde. Es empfiehlt sich, ein jederzeit mögliches Vorgehen lediglich an einer Stelle, ggf. in einem eigenen Prozessmodell, zu modellieren und es dann als globale Funktionalität zu definieren. Dies reduziert die Komplexität von Prozessmodellen, ohne dabei den umgesetzten Prozess in seiner Funktionalität zu beschränken.

Eine weitere Möglichkeit, unterschiedliche Lösungswege für einen komplexen Geschäftsprozess darzustellen, ist die Erstellung mehrerer Modelle für diesen Prozess. Auf diese Weise entstehen leicht verständliche und umsetzbare *Happy-Path-Modelle* zur Abbildung differenzierter Vorgehensweisen, bspw. in Abhängigkeit einer unterschiedlichen Datenlage. Mehrere Modelle zu einem Prozess sollten

untereinander valide sein, d.h. sie sollten z.B. über gleiche Eingangsparameter verfügen und zum gleichen Ziel führen. Dazu bietet sich die Definition eines übergeordneten, möglichst generischen Prozessmodells an, in dem sich alle daraus abgeleiteten Happy-Path-Modelle wiederfinden und anhand dessen eine Validierung dieser Modelle möglich ist. Die folgenden Abbildungen zeigen beispielhaft die Aufteilung eines Prozesses in mehrere Ablaufdiagramme.

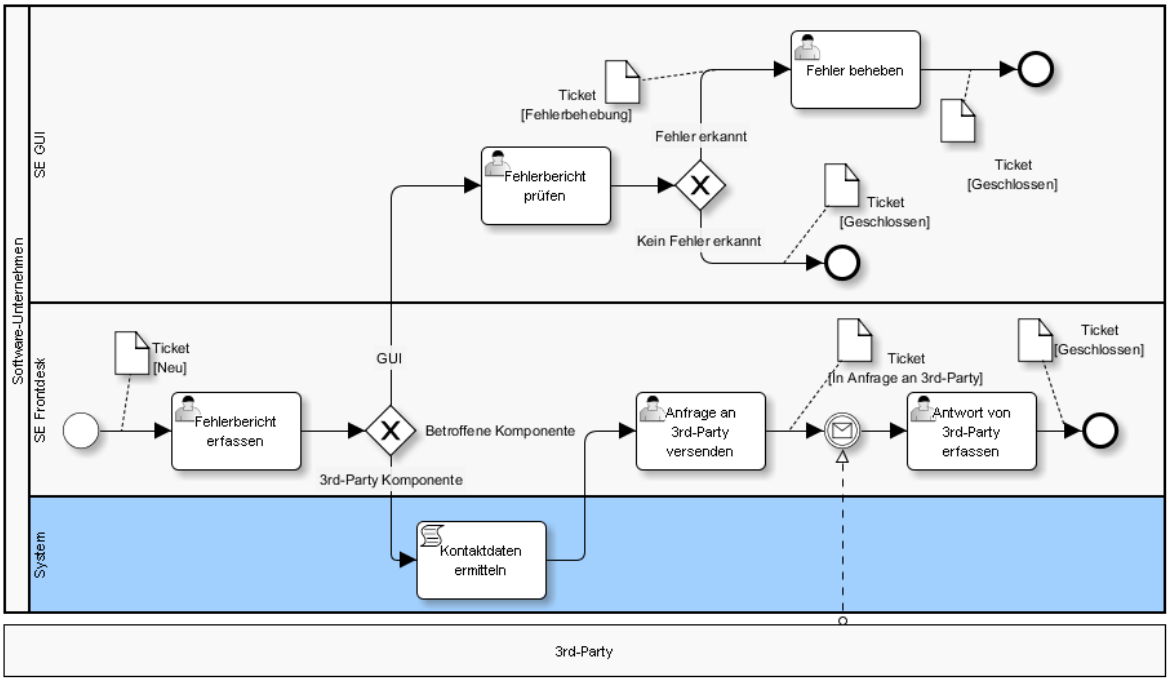


Abbildung 8: Prozessmodell "Ticketbearbeitung" 80%

Abbildung 8 zeigt einen stark vereinfachten Prozessablauf für die Bearbeitung eines Tickets in der Softwareentwicklung. Ein neu erstelltes Ticket, welches einen Fehler in einer Anwendung meldet, wird zunächst auf die betroffene Komponente untersucht. Bezieht sich der Fehler auf die graphische Benutzeroberfläche (GUI) der Anwendung, wird das Ticket in die entsprechende Entwicklungsabteilung weitergegeben. Dort wird entschieden, ob es sich bei dem berichteten Verhalten tatsächlich um einen Fehler oder doch um gewünschtes Verhalten handelt. Im Falle eines Fehlers, wird dieser behoben. Betrifft der gemeldete Fehler eine Komponente, die von einem anderen Anbieter (3rd-Party) bereitgestellt wird, z.B. ein Fehler in einem verwendeten Framework, wird dieser Hersteller kontaktiert und dessen Antwort später dokumentiert. Dieses Prozessmodell hat den Anspruch den aufgezeigten Prozess so darzustellen, wie er üblicherweise, also z.B. „in 80% der Fälle“, abläuft und lässt daher verschiedene Pfade aus. Zu den ausgelassenen Pfaden gehört beispielsweise der Fall, dass weder die GUI noch eine 3rd-Party Kompo-

nente von dem beschriebenen Fehler betroffen sind. Des Weiteren wird in dem Modell nicht dargestellt, wie eine ausbleibende Antwort der 3rd-Party behandelt wird. Zuletzt befindet sich das Ticket am Ende des Prozesses immer im Zustand *Geschlossen*, obwohl keine entsprechende Aktivität *Ticket schließen* in dem Modell vorkommt. Diese Abläufe werden in anderen, ebenfalls auf den Prozess *Ticketbearbeitung* bezogenen, Prozessmodellen dargestellt.

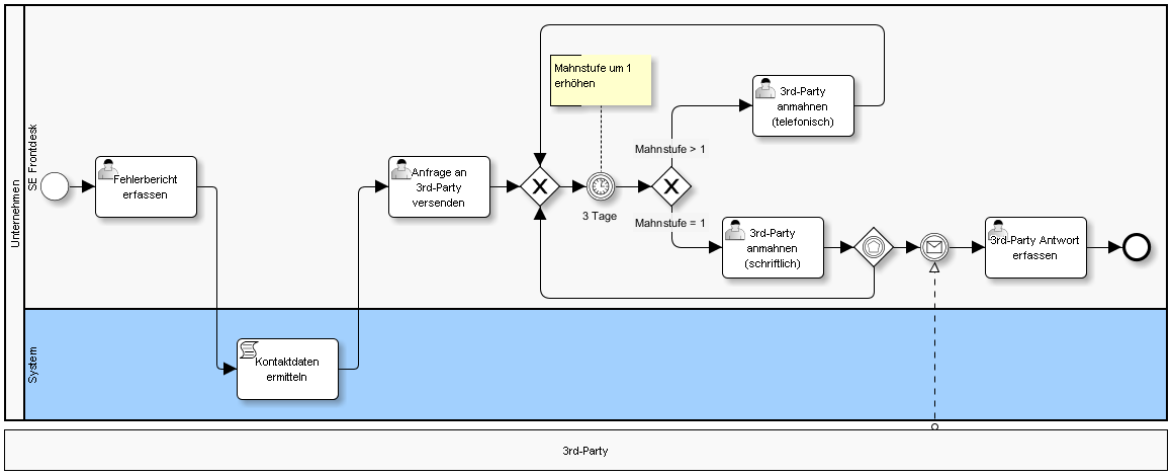


Abbildung 9: Behandlung ausbleibender Antwort durch 3rd-Party

So zeigt Abbildung 9 das Vorgehen bei ausbleibender Antwort der 3rd-Party, wobei immer nach 3 Tagen ohne Antwort die Mahnstufe um den Wert *1* erhöht wird. Beim ersten Mal wird die 3rd-Party noch schriftlich angemahnt. Jedes weitere Mal erfolgt eine telefonische Ermahnung.

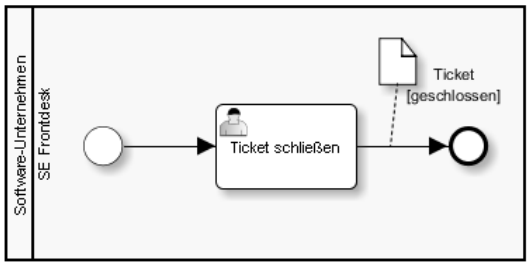


Abbildung 10: Jederzeit mögliche Aktivität im Prozess Ticketbearbeitung

Zum Abschluss des Beispiels, wie PCM mit mehreren Prozessmodellen für einen Prozess umgeht, zeigt Abbildung 10 die elementare Aktivität *Ticket schließen*, welche ein Ticket jederzeit, d.h. aus beliebigem Zustand, in den Zustand *Geschlossen* überführt. Da die Aktivität ausschließlich in diesem Prozessmodell vor-

kommt und hier keine Vorbedingungen in Form weiterer Aktivitäten vorfindet, ist sie in einer auf diesen Modellen beruhenden Umsetzung jederzeit durchführbar. PCM erlaubt die Umsetzung des Prozesses *Ticketbearbeitung* bezogen auf alle gezeigten Prozessmodelle. Es werden verschiedenste Anforderungen umgesetzt und Ablaufpfade ermöglicht, ohne dabei ein einziges Prozessmodell bis zur Unleserlichkeit auszudehnen.

4.2 Prozessdaten

Auch wenn im Rahmen von PCM dem Prozessfluss ein deutlich größerer Anteil bei der Ablaufsteuerung zukommt als dies bei ACM der Fall ist, so spielen auch hier die zu einem Prozess und dessen Cases gehörigen Daten eine sehr wichtige Rolle. Als *Casedaten* werden im Folgenden die *beweglichen* Daten verstanden, die mit jedem neu ausgelösten Case erzeugt und im Zuge dessen Abarbeitung verändert oder hinzugefügt werden. Solche Daten, die sich eher selten ändern und nicht für jede Instanz separat existieren, wie z.B. der Prozessverantwortliche, werden als *Prozessdaten* bezeichnet. Letztere können auch als *Prozessstammdaten* charakterisiert werden.

Sobald ein Ereignis einen neuen Case auslöst, werden oft schon die ersten Daten mitgeliefert, die u.U. auch sofortigen Einfluss auf den Verlauf des Cases nehmen. Im weiteren Verlauf des Cases werden die Daten durch die Ausführung von Aktivitäten oder äußere Ereignisse verändert oder neu hinzugefügt. Wie auch bei ACM übernimmt so die sich ändernde Datenlage, d.h. die Gesamtheit aller zu einem Case gehörenden Daten, eine Steuerungsfunktion, da in Abhängigkeit dieser Datenlage Aktivitäten gesperrt oder freigegeben werden können.

Die Modellierung von Daten ist, besonders in der Software-Entwicklung, nichts Neues. Dabei hat sich speziell die UML (Unified Modelling Language) als funktionales Werkzeug bewiesen, um Gegebenheiten der *echten Welt* in maschinenlesbaren Modellen abzubilden. Die Gegenstände dieser *echten Welt*, sowie deren Eigenschaften werden in der UML durch *Klassen* und deren *Attribute* repräsentiert. Kanten und Pfeile setzen dann die zuvor unabhängigen Klassen auf verschiedene Art und Weise in Beziehungen. Auf diese Weise lassen sich *Kompositionen, Aggregationen,* sowie *Generalisierungen* darstellen.

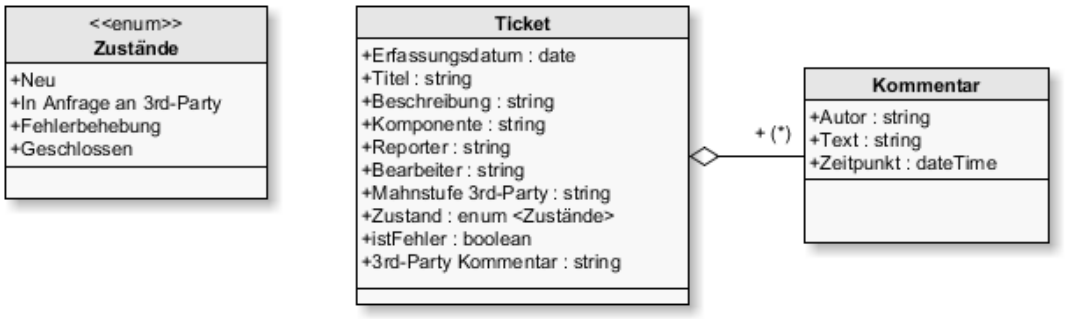


Abbildung 11: UML-Datenmodell zum Beispielprozess Ticketbearbeitung

Abbildung 11 zeigt das Datenmodell, welches im vorgestellten Beispielprozess *Ticketbearbeitung* verwendet wird. Die zentral angeordnete Klasse *Ticket* enthält dabei die Casedaten und stellt das *zentrale Datenobjekt*, d.h. die Klasse, die den Großteil der im Prozessverlauf benötigten Daten hält, dieses Prozesses dar. Obwohl ein zu einem Prozess gehörendes Datenmodell i.d.R. über mehrere Klassen verfügt, handelt es sich jedoch stets bei einer dieser Klassen um das *zentrale Datenobjekt*. Das Beispiel-Datenmodell enthält zusätzlich die Klasse *Kommentar*, die über eine Komposition mit dem zentralen Datenobjekt *Ticket* verbunden ist. Dies bedeutet, dass es zu einem Ticket beliebig viele Kommentare mit den aufgeführten Attributen geben kann.

Die Modellierung der Daten, die im Umfeld eines Prozesses existieren, ist allerdings nur ein Teil der für PCM benötigten Datenmodellierung. Es muss zusätzlich definiert werden, wann, d.h. an welcher Stelle in einem durchgeführten Case, Daten geändert werden. Zwar sollen Daten grundsätzlich jederzeit veränderbar sein, doch kann es Stellen im Prozess geben, an denen bestimmte Daten gesetzt werden müssen, um den geplanten Prozessverlauf fortzusetzen. Die dabei entstehende Verschmelzung von Prozess- und Datenmodellierung erleichtert außerdem die Identifikation von benötigten Attributen. Hängt der Prozessfluss von einem bestimmten Datenwert ab, so muss das zugehörige Datum definitiv im Datenmodell vorhanden sein. Im Beispielprozess *Ticketbearbeitung* verdeutlicht dies ein Blick auf die Behandlung einer ausbleibenden Antwort durch die 3rd-Party, dargestellt in Abbildung 9. Lässt eine Antwort länger als drei Tage auf sich warten, so wird durch einen Skript-Task die *Mahnstufe* um den Wert *1* erhöht. Anhand der Höhe der Mahnstufe wird anschließend entschieden, ob eine schriftliche oder telefonische Mahnung erfolgt. Um den Wert der Mahnstufe zu verändern und abzufragen, ist es Voraussetzung, dass dieses Attribut im zentralen Datenobjekt existiert. Dies führt vor Augen, wie bei der Prozessmodellierung benötigte Teile der Datenmodel-

lierung sichtbar werden und die beiden Modellierungsvorgänge dadurch zusammenwirken.

4.3 Prozesszustände

Von seiner Instanziierung bis hin zu seinem Abschluss durchläuft ein Case verschiedene Zustände. Dies sind typischerweise Zustände wie *Neu*, *Offen*, *in Bearbeitung* oder *Geschlossen*. Letztendlich sollten hier bei der Umsetzung jedoch keine Grenzen gesetzt sein.

Definiert werden die Zustände, die ein Case während seiner Abarbeitung einnehmen kann, ebenfalls während der fachlichen Aufnahme. Dazu bietet es sich an, ähnlich wie bei der Datenmodellierung, festzulegen, an welcher Stelle im Prozess ein neuer Zustand eintritt. Dabei können sowohl die Durchführung von Aktivitäten, als auch das Eintreten von Ereignissen eine Zustandsänderung auslösen. In dem Beispielprozess *Ticketbearbeitung* befindet sich der aktuelle Zustand eines Cases direkt am zentralen Datenobjekt in den Prozessmodellen. Wie in Abbildung 12 gezeigt, ist dabei das zentrale Datenobjekt jeweils mit dem Sequenzfluss assoziiert, welcher den neuen Zustand herbeiführt. In dem gezeigten Beispiel erhält der Case nach der Durchführung der Aktivität *Anfrage an 3rd-Party versenden* den neuen Zustand *In Anfrage an 3rd-Party*.

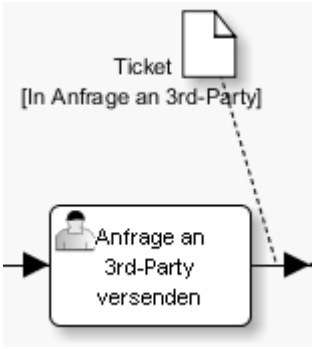


Abbildung 12: Zustandswechsel im Prozess Ticketbearbeitung

Wenn alle möglichen Zustände eines Prozesses identifiziert wurden, bilden sie sozusagen den Lebenszyklus des Prozesses bzw. der durchzuführenden Cases. Dieser Lebenszyklus kann dann anhand eines Zustandsübergangsdiagrammes dargestellt und später implementiert werden. Bei der Erstellung eines solchen Diagrammes existieren zwei Herangehensweisen, um Zustandsübergänge zu er-

kennen. Die Erste ist die Orientierung am Prozessfluss. Wurde ein Prozessmodell um Datenobjekte angereichert, um anzuzeigen, wo und auf welche Weise diese Objekte durch den Prozess verändert werden und welchen Zustand sie dadurch einnehmen, beschreibt der Prozessfluss selbst welche Zustandsübergänge möglich sind. Durch die Verfolgung der Prozessflüsse des Beispielprozesses *Ticketbearbeitung* ergeben sich die in Tabelle 3 aufgeführten Zustandsübergänge.

Alter Zustand	Neuer Zustand	Auslöser	Bedingung
---	Neu	Startereignis	
Neu	Fehlerbehebung	Fehlerbericht prüfen	Fehler erkannt
Fehlerbehebung	Geschlossen	Fehler beheben	
Neu	Geschlossen	Fehlerbericht prüfen	Kein Fehler erkannt
Neu	In Anfrage an 3rd-Party	Anfrage an 3rd-Party versenden	
In Anfrage an 3rd-Party	Geschlossen	Antwort von 3rd-Party erfassen	
In Anfrage an 3rd-Party	Telefonische Anmahnung erforderlich	Zeitereignis *3 Tage*	Mahnstufe > 1
In Anfrage an 3rd-Party	Schriftliche Anmahnung erforderlich	Zeitereignis *3 Tage*	Mahnstufe = 1
Telefonische Anmahnung erforderlich	In Anfrage an 3rd-Party	3rd-Party anmahnen (telefonisch)	
Schriftliche Anmahnung erforderlich	In Anfrage an 3rd-Party	3rd-Party anmahnen (schriftlich)	
In 3rd-Party Anmahnung	Geschlossen	Antwort von 3rd-Party erfassen	

Tabelle 3: Prozessbedingte Zustandsübergänge der Ticketbearbeitung

Wie in Kapitel 3.5 erläutert, stellen Prozessmodelle im PCM-Kontext mehr eine Richtlinie als vorbestimmte Abläufe dar. Dies hat zur Folge, dass grundsätzlich jede Aktivität zu jedem Zeitpunkt durchgeführt werden kann und außerdem jedes Ereignis jederzeit eintreten kann. Daher ist es auch möglich, dass es bei der Abarbeitung eines Cases zu Zustandsübergängen kommt, die aus keinem der zum Prozess modellierten Prozessmodelle hervorgehen. Diese Zustandsübergänge müssen von Prozessverantwortlichen /–experten definiert werden, damit sie in der technischen Realisierung zur Anwendung kommen. Im Folgenden werden diese Zustandsübergänge als *prozessunabhängige Zustandsübergänge* bezeichnet. Definieren lassen sich diese Zustandsübergänge, genau wie prozessabhängige Übergänge, durch folgende Eigenschaften:

1. Ein alter Zustand,
2. ein neuer Zustand,
3. einen Auslöser (Ereignis oder Aktivität),
4. Bedingungen, d.h. Datenwerte mit bestimmten Werten (optional).

Für die Definition von zusätzlichen Zustandsübergängen besteht weder eine spezifische Modellierungsnotation, noch sieht die BPMN Elemente zu diesem Zwecke vor. Zur Festlegung solcher Zustandsübergänge innerhalb einer PCM-Umgebung bieten sich daher bspw. Formulare an, mit denen die o.g. Eigenschaften von zusätzlichen Zustandsübergängen erfasst werden können.

Im Beispielprozess *Ticketbearbeitung* existiert die Anforderung, ein Ticket jederzeit schließen zu können. Daraus ergibt sich, wenn auch nicht direkt aus den verwendeten Prozessmodellen, dass jeder definierte Zustand in den neuen Zustand *Geschlossen* wechseln kann. Für das Beispiel ergeben sich daraus die in Tabelle 4 aufgeführten, zusätzlichen Zustandsübergänge:

Alter Zustand	Neuer Zustand	Auslöser	Bedingung
Telefonische Anmahnung erforderlich	Geschlossen	Ticket schließen	
Schriftliche Anmahnung erforderlich	Geschlossen	Ticket schließen	

Tabelle 4: Prozessunabhängige Zustandsübergänge im Prozess Ticketbearbeitung

Die bisher verwendete Darstellung von Zustandsübergängen in Tabellenform ist nur eine Möglichkeit der Veranschaulichung. In der Softwareentwicklung wird dazu häufig das Zustandsdiagramm der UML verwendet, welches aus Zuständen und Transitionen, d.h. Zustandsübergängen, besteht.[44] Die Zustände werden dabei als Textknoten dargestellt, während die Transitionen diese Knoten in Form von Pfeilen miteinander verbinden. Zumeist werden die Transitionen außerdem mit dem Ereignis, durch welches sie ausgelöst werden, beschriftet. Als Ereignis wird dabei u.a. auch das Abschließen einer Aktivität verstanden. Abbildung 13 zeigt ein Zustandsdiagramm für den Beispielprozess *Ticketbearbeitung*, wobei jene Transitionen, die sich direkt aus den Prozessflüssen ergeben, als durchgehende Linien dargestellt sind, während prozessunabhängige Zustandsübergänge durch gestrichelte Linien repräsentiert werden.

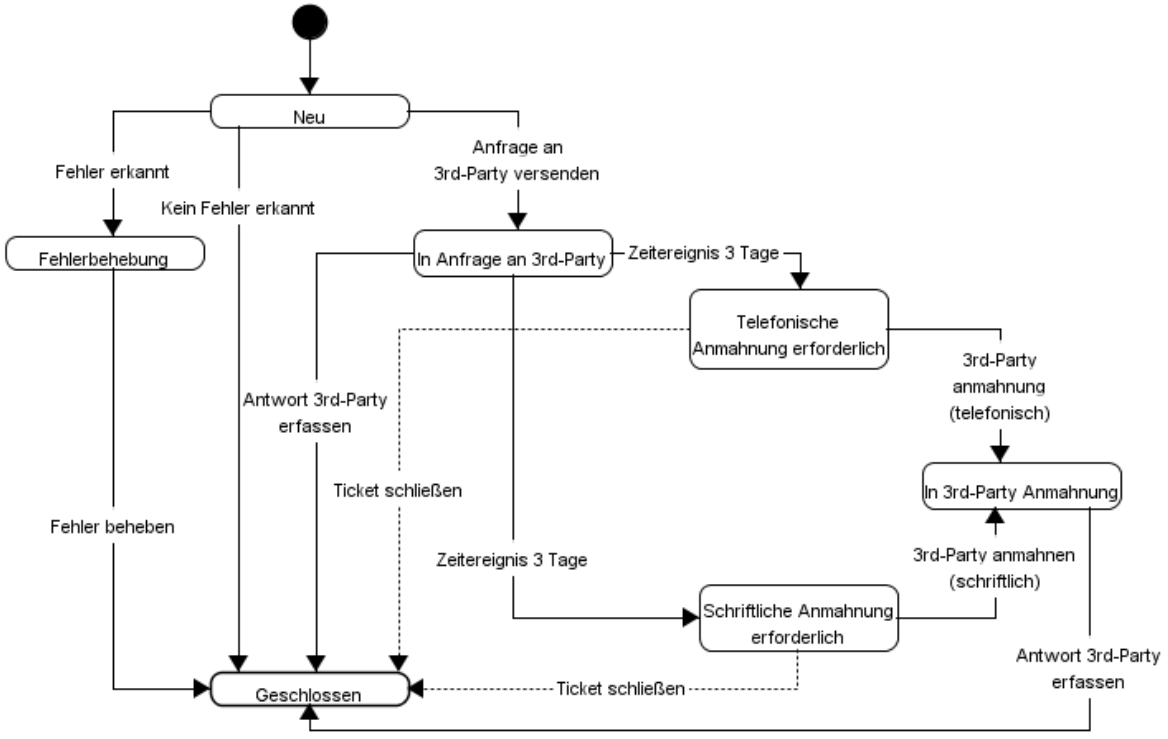


Abbildung 13: Zustandsdiagramm für den Prozess Ticketbearbeitung

Der finale Zustand *Geschlossen* ist als solcher durch einen dicken Rand gekennzeichnet.

[44] Vgl. **Kecher 2011**, S. 293ff

5 Konzeption einer PCM-Umgebung

Im vorherigen Kapitel wurden benötigte Inhalte einer fachlichen Prozessaufnahme im Rahmen einer Prozessumsetzung über PCM vorgestellt und erläutert. Bei der Untersuchung, welche Informationen auf welche Art und Weise erfasst werden können, wurde festgestellt, dass dies in der Gesamtheit auf ganz unterschiedlichen Wegen realisiert wird. Vor allem haben sich mit einem Geschäftsprozessmodell, einem Datenmodell und einem Zustandsübergangsdiagramm drei Elemente herauskristallisiert, die gemeinsam einen Großteil der benötigten Informationen *dezentral* erfassen. Um die gesammelten, fachlichen Inhalte validieren, oder daraus (Teil-)Lösungen generieren zu können, obwohl sie aus unterschiedlichen Quellen stammen, stellt dieses Kapitel ein Modell vor, das alle fachlichen Inhalte eines über PCM umzusetzenden Prozesses vereint. Auf Grundlage dieses *PCM-Metamodells* soll es möglich sein, aufgenommene Prozesse und deren fachliche Inhalte in einer Umgebung zusammenzutragen und dann beliebige Operationen auf diesem Informationsbestand auszuführen. Wie genau diese Ausführung auf dem Modell funktionieren kann, beschreibt im darauf folgenden Unterkapitel die Ausführungssemantik. Den letzten Teil des Konzeptes stellen Validierungsalgorithmen dar, die die erfassten Bestandteile eines Prozesses auf fachliche Fehler untersuchen. Vor allem die Verwendung mehrerer Prozessmodelle und das Definieren von zusätzlichen Zustandsübergängen können potentiell für Widersprüche innerhalb einer Prozessdefinition sorgen. Die Validierung soll noch während der Definitionsphase darauf aufmerksam machen.

5.1 PCM-Metamodell

Im Folgenden wird ein Modell vorgestellt, welches die Bestandteile von Prozessen, die mit Hilfe von PCM umgesetzt werden sollen, miteinander in Verbindung setzt. Es liegt in Form eines UML Klassendiagrammes vor und wird auf Abbildung 14 gezeigt.

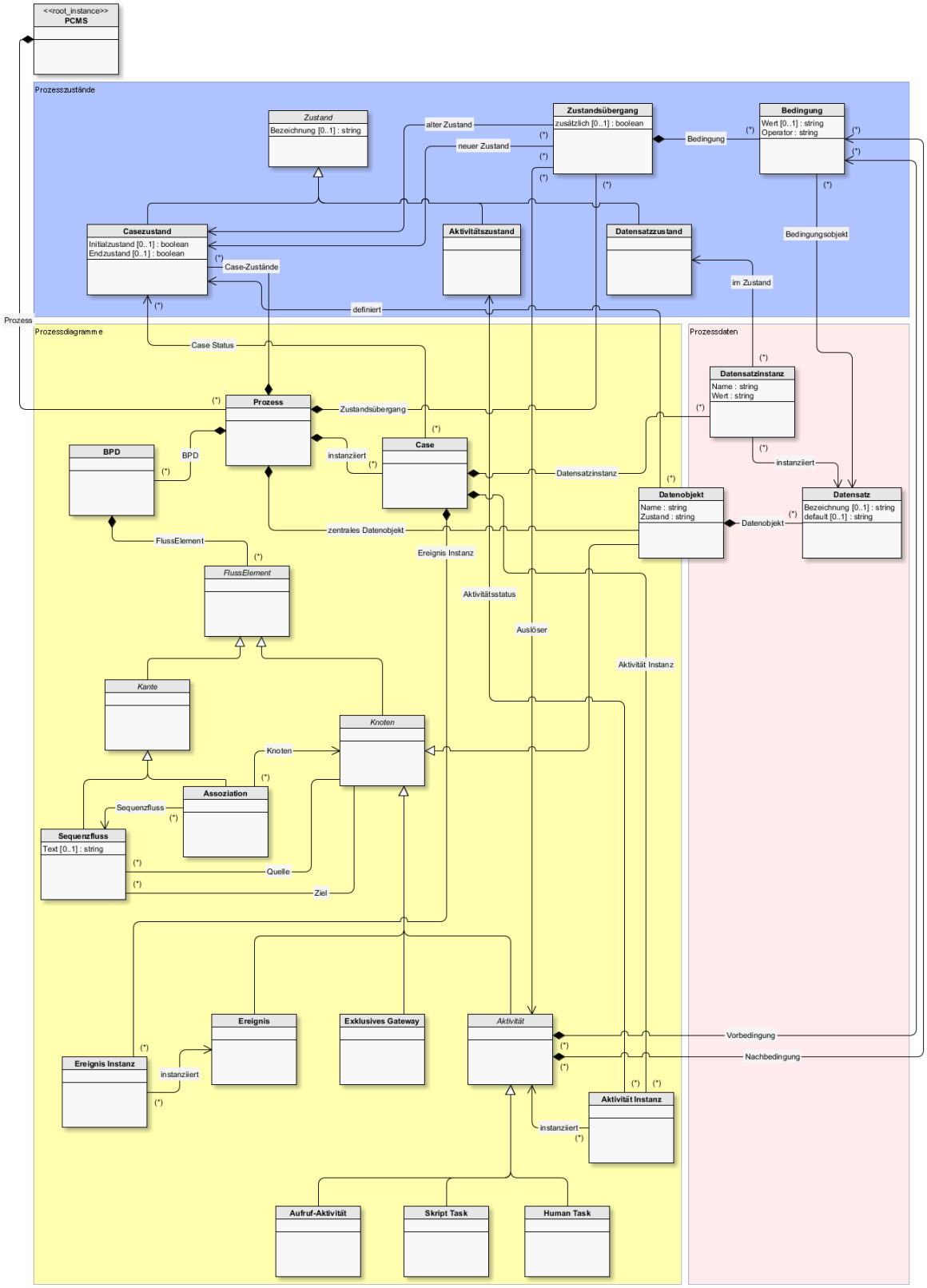


Abbildung 14: PCM-Metamodell

Entsprechend der in Kapitel 4 beschriebenen Inhalte, die zur Umsetzung mit PCM aufgenommen werden müssen, teilt sich das Modell in die drei Bereiche *Prozesszustände*, *Prozessdiagramme* und *Prozessdaten*. Als Einstiegspunkt verwendet das Modell die Klasse *PCMS*, unterhalb derer alle weiteren Inhalte aufgehängt sind. Die Klasse repräsentiert ein *Production Case Management System*, welches Prozesse mit Hilfe von PCM umsetzt.

Die Persistenz und Präsentation von Prozessen und deren Bestandteilen sind die Hauptaufgaben dieses Modells. Daher steht die Klasse *Prozess* als Einzige direkt mit der Root-Klasse *PCMS* in Assoziation. Ein *PCMS* kann dabei mehrere Prozesse halten, während ein *Prozess* immer genau einem *PCMS* zugeordnet ist. An dieser Stelle sei nochmal an die bereits zuvor festgelegte Terminologie erinnert, nach der ein *Prozess* eine abstrakte Bedeutung im Sinne einer Prozessdefinition einnimmt, während ein *Case* eine Instanz eines Prozesses darstellt. Diese Beziehung zwischen *Prozess* und *Case* findet sich auch in dem vorgestellten Modell wieder, in welchem die Klasse *Case* über eine Komposition mit der Klasse *Prozess* verbunden ist. Ein *Prozess* besteht also unter anderem aus null bis beliebig vielen *Cases*.

Zunächst soll aber näher auf die eigentlichen Modellierungselemente für die Prozessdiagramme eingegangen werden. Abbildung 15 zeigt die dazu im Modell verwendeten Klassen, beginnend mit der Klasse *BPD*, die als Komposition an der Klasse *Prozess* hängt.

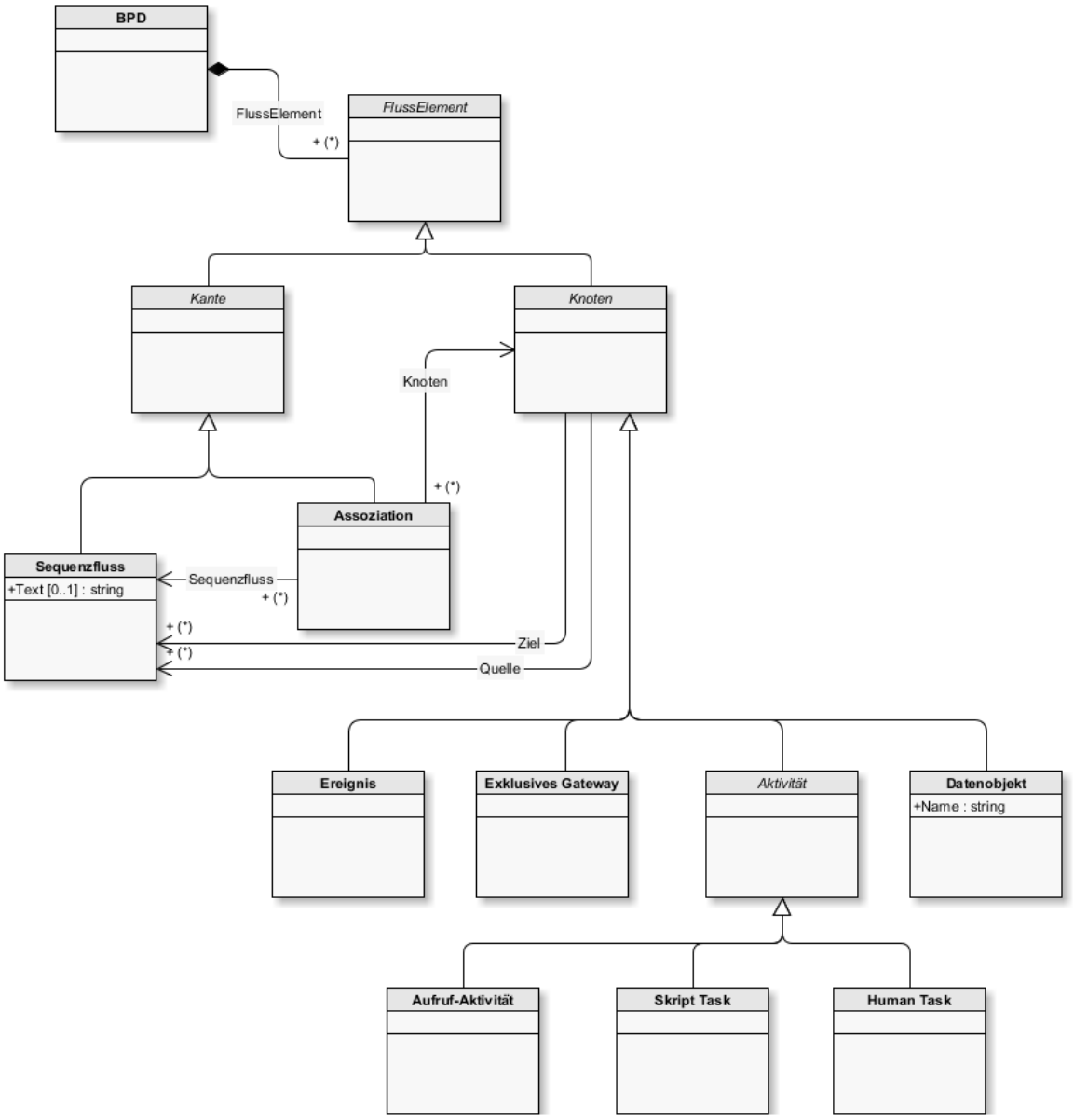


Abbildung 15: PCM-Metamodell, Bereich Prozessdiagramme

Die zwischen den Klassen *BPD* und *Prozess* gewählte Kardinalität erfüllt die Anforderung, dass ein mit PCM umgesetzter Prozess auf mehreren Prozessmodellen beruhen kann. Dies ermöglicht z.B. die Nutzung mehrerer Happy-Path-Modelle und/oder die Verwendung eines generischen Modells, welches *sonstige* Ablaufmöglichkeiten abdeckt. Unterhalb der Klasse *BPD* folgt das vorgestellte Modell, in Ausschnitten, der BPMN 2.0 Spezifikation[45]. Demnach besteht ein Geschäftsprozessmodell aus *Flusselementen*, von welchen jedes entweder eine *Kante* oder einen *Knoten* darstellt. Sämtliche dieser drei Klassen sind abstrakt und können

[45] Vgl. **OMG 2011**, S.87ff

daher nicht instanziiert werden. Von der Klasse *Kante* erben die Klassen *Sequenzfluss* und *Assoziation*. Während ein *Sequenzfluss* immer zwei *Knoten* miteinander verbindet, ermöglicht die *Assoziation* eine Verbindung von einem *Knoten* mit einem *Sequenzfluss*. Dies wird für die Kopplung von Datenobjekten an Sequenzflüssen benötigt, wie es beispielsweise in Abbildung 12 gezeigt wurde. Von der Klasse *Knoten* hingegen erben die Klassen *Datenobjekt*, *Ereignis*, *Gateway* und *Aktivität*. Welche Funktionalität diese Elemente in einem Prozessmodell einnehmen, wurde in Kapitel 2.4.1 kurz erläutert. Ähnlich wie das Verhältnis von einem *Prozess* zu einem *Case*, verhält es sich auch bei Ereignissen und Aktivitäten zu deren Instanzen. Während ein *Ereignis* oder eine *Aktivität* jeweils formelle Definitionen darstellen, repräsentieren die Klassen *Ereignis Instanz* und *Aktivität Instanz* die tatsächlich für einen konkreten Case vorliegenden Ausprägungen. Verdeutlicht wird dies durch die im Modell vorhandenen Assoziationen zwischen der Klasse *Case* und den Klassen *Ereignis Instanz* bzw. *Aktivität Instanz*. Die Klasse *Aktivität* ist ebenfalls abstrakt und vererbt an die Klassen *Skript Task*, *Human Task*, sowie *Aufruf-Aktivität*. Anhand der Differenzierung zwischen *Human Task* und *Skript Task* kann das PCMS später solche Aktivitäten unterscheiden, die automatisch durchgeführt werden (*Skript Tasks*) und solche, die Eingabe durch Mitarbeiter (*Human Tasks*) erfordern. Aufruf-Aktivitäten repräsentieren Aktivitäten, die bereits an anderer Stelle im gleichen oder einem anderen Prozessmodell verwendet wurden.

Als letzte Klasse erbt auch *Datenobjekt* von dem Flusselement *Knoten* und ist damit Teil eines Prozessmodells. Die Platzierung dieser Klasse in dem *PCM-Metamodell* ist jedoch nicht zufällig so gewählt, dass sie die Bereiche *Prozessdiagramme* und *Prozessdaten* überschneidet. Repräsentiert diese Klasse einerseits die Datenartefakte in einem Prozessmodell, so steht sie gleichzeitig für eine in einem Datendiagramm (Business Object Diagram, BOD) verwendete Klasse, wie z.B. *Ticket* aus dem Beispielprozess *Ticketbearbeitung*. Dass in dem PCM-Metamodell die Datenartefakte eines Prozessmodells und die Klassen eines Datendiagramms durch nur ein Klasse repräsentiert werden, stellt die Anforderung an spätere Prozessdesigner, dass sich in einem Prozessmodell verwendete Datenartefakte immer im zum Prozess gehörigen Datenmodell in Form einer Klasse wiederfinden müssen.

An dieser Stelle befindet sich die Erläuterung des PCM-Metamodells in dem Abschnitt Prozessdaten (rechts), wie es in Abbildung 16 gezeigt ist. Ein Datenobjekt beschreibt einen in der Realität vorliegenden Gegenstand oder eine Zusammensetzung einzelner aber dennoch zusammen gehöriger Informationen. Diese ein-

zelnen Informationen finden sich in dem PCM-Metamodell in der Klasse *Datensatz* wieder, von denen beliebig viele zu einem *Datenobjekt* zusammengefügt werden können.

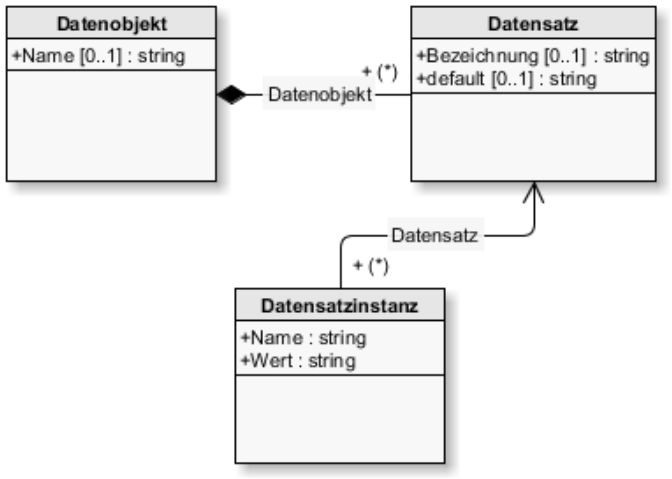


Abbildung 16: PCM-Metamodell, Bereich Prozessdaten

Ein *Datensatz* ist damit das, was in einem Datenmodell als Attribut bezeichnet wird. Auch an dieser Stelle muss, wie schon zuvor bei *Prozess* und *Case* bzw. bei *Aktivität* und *Aktivität Instanz*, in Definition- und Instanzebene unterschieden werden. Dabei ist der *Datensatz* die Klasse, die eine Information der Realität definiert, während die tatsächlich eintretenden Geschäftsdaten durch die Klasse *Datensatzinstanz* repräsentiert werden. Eine *Datensatzinstanz* gehört demnach immer genau zu einem *Datensatz*, durch welchen sie definiert wird. Zu einem *Datensatz* können beliebig viele *Datensatzinstanzen* gehören. Die Klasse *Datensatz* enthält ein optionales Attribut *default*, welches, wenn es denn mit einem Wert belegt wurde, bei der Erzeugung einer *Datensatzinstanz* seinen Wert in das *Wert*-Attribut der neuen *Datensatzinstanz* überträgt. Als Teil der Instanzebene hängt die Klasse *Datensatzinstanz* als Komposition am *Case*. Der *Datensatz* hingegen hängt, als Teil eines *Datenobjekts,* über die Assoziation *zentrales Datenobjekt* an der Klasse *Prozess*.

Als letzter Teil des PCM-Metamodells folgt nun die Vorstellung des Bereiches *Prozesszustände* (oben), separiert dargestellt in Abbildung 17. Die meisten der hier vorkommenden Klassen stehen in Beziehung zu bereits zuvor erläuterten Klassen der anderen zwei Bereiche *Prozessdiagramme* und *Prozessdaten*. Zentrales Element im Bereich *Prozesszustände* ist die abstrakte Klasse *Zustand*.

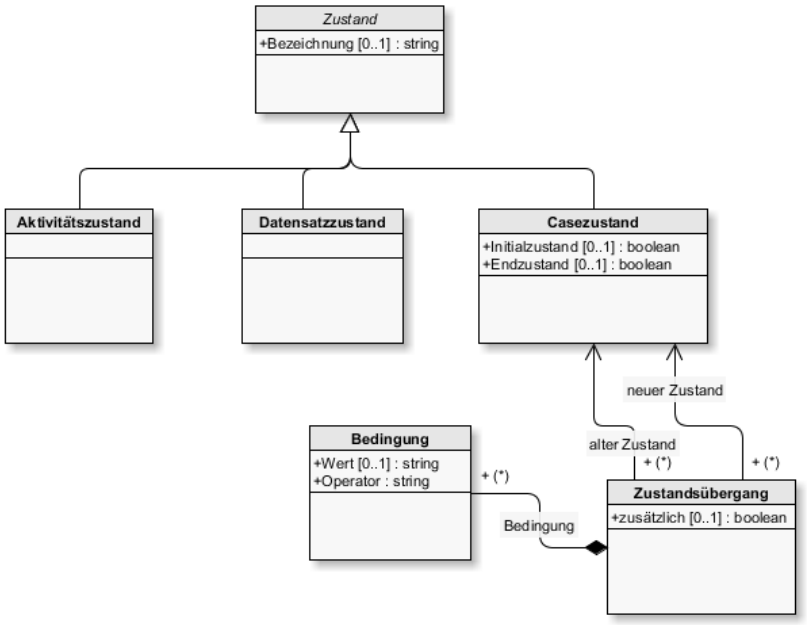


Abbildung 17: PCM-Metamodell, Zustände

Konkrete Ausprägungen eines Zustands definieren die Klassen *Casezustand*, *Aktivitätszustand* und *Datensatzzustand*. Entsprechend ihrer Namensgebung sind dies Zustände, die die Instanzen der jeweiligen, zugehörigen Klassen einnehmen können. Das bedeutet, eine Datensatzinstanz befindet sich immer in einem Datensatzzustand, eine Aktivitätsinstanz in einem Aktivitätszustand und ein Case in einem Casezustand. In dem PCM-Metamodell ist dies durch einfache Assoziationen dargestellt. Die Definition, welche Casezustände existieren, ist abhängig vom umgesetzten Prozess. Daher ist die Klasse *Casezustand* als Kompositionen mit der Klasse *Prozess* verbunden. Welche Casezustände für einen Prozess existieren wird über die in den zugehörigen Prozessmodellen definiert. Die darin enthaltenen Datenobjekte tragen die möglichen Zustände in ihrem Attribut *Zustand*, dargestellt durch die Assoziation *definiert* zwischen den Klassen *Casezustand* und *Datenobjekt*. Für einen Casezustand kann gemäß des PCM-Metamodells festgelegt werden, ob es sich bei ihm um einen Initialzustand oder einen Endzustand dieses Cases handelt. Dies wird über die gleichlautenden booleschen Attribute der Klasse *Casezustand* definiert. Natürlich können beide Attribute den Wert *falsch* einnehmen, wodurch ein Casezustand definiert wird, der weder Initial- noch Endzustand ist. Theoretisch erlaubt es das Modell auch beide Attribute mit dem Wert *wahr* zu versehen. Ob das fachlich betrachtet sinnvoll sein kann, ist für das Modell nicht von Bedeutung.

Dort wo verschiedene Zustände existieren, gibt es auch Zustandsübergänge. Diese werden in dem PCM-Metamodell von der Klasse *Zustandsübergang* repräsentiert, welche als Komposition an der Klasse *Prozess* hängt. Allerdings beschränkt sich diese Klasse darauf, die verschiedenen Übergänge von Casezuständen abzubilden. Dazu existieren für einen Zustandsübergang immer genau zwei Assoziationen zu vorhandenen Casezuständen. Während die Assoziation *alter Zustand* den *Casezustand* verknüpft, der vor dem Zustandsübergang von dem betroffenen Case eingenommen wurde, definiert die Assoziation *neuer Zustand*, welchen Zustand der Case nach dem Zustandsübergang einnimmt. Zusätzlich wird für jeden *Zustandsübergang* ein Auslöser festgelegt, welcher bei seinem Eintreten den Zustandsübergang einleitet. Das Modell sieht als Auslöser ausschließlich Aktivitäten, d.h. die erfolgreiche Durchführung jener Aktivitätsinstanz, die die als Auslöser festgelegte Aktivität in einem konkreten Case vertritt, vor. Mehr dazu, wie genau Zustandsübergänge, auch für Aktivitäten und Datensätze, von einem PCMS erkannt und umgesetzt werden, beschreibt im Anschluss an dieses Kapitel die Ausführungssemantik des PCM-Metamodells. Zustandsübergänge ergeben sich vor allem aus den zu einem Prozess erstellten Prozessmodellen. Einer der einschlägigsten Mehrwerte, die durch PCM jedoch erzielt werden sollen, ist die Möglichkeit zusätzliche Zustandsübergänge von Cases definieren zu können. Ein dazu erstellter, *zusätzlicher* Zustandsübergang kann über das entsprechende Attribut *zusätzlich* der Klasse *Zustandsübergang* gekennzeichnet werden.

Weiterhin erlaubt das PCM-Metamodell die Verknüpfung von *Zustandsübergängen* mit *Bedingungen*. Diese Bedingungen ermöglichen die Festlegung, dass ein Zustandsübergang trotz Eintreten seines Auslösers nur dann durchgeführt wird, wenn eine bestimmte Datensatzinstanz einen festgelegten Wert einnimmt. Ein *Zustandsübergang* kann beliebig viele *Bedingungen* haben, was durch die zwischen diesen Klassen bestehende Komposition *Bedingung* festgelegt ist. Die Klasse *Bedingung* ist mit genau einem *Datensatz* assoziiert, dessen Wert darüber entscheidet, ob die Bedingung erfüllt ist oder nicht. Dazu hält eine Bedingung das Attribut *Wert*, dessen Inhalt mit dem aktuellen Wert des zu prüfenden Datensatzes abgeglichen wird. Auf welche Art und Weise dieser Abgleich stattfindet, entscheidet das Attribut *Operator*. Mögliche Inhalte für dieses Attribut sind beispielsweise *gleich*, *größer*, *kleiner*, *existiert* oder *existiert nicht*.

Bedingungen müssen allerdings nicht zwangsläufig Teil eines Zustandsübergangs sein. Das PCM-Metamodell sieht es ebenfalls vor, dass Aktivitäten mit beliebig vielen Bedingungen versehen werden können. Dort fungieren sie entweder als Vor- oder Nachbedingungen, in Abhängigkeit davon über welcher der beiden im

Modell möglichen Assoziationen sie mit einer Aktivität verknüpft sind. Welche Auswirkungen Vor- und Nachbedingungen von Aktivitäten auf die Durchführung eines Cases haben, wird in der folgenden Ausführungssemantik näher beschrieben.

5.2 Ausführungssemantik

Dieses Kapitel beschreibt die Ausführungssemantik des zuvor vorgestellten PCM-Metamodells. Diese Ausführungssemantik soll aufzeigen, wie die Elemente des PCM-Metamodells dazu verwendet werden können, um Prozesse in Form von Cases ausführbar zu machen. Als Erstes wird dazu erläutert, durch welche Umstände ein Case instanziiert und wodurch er wieder terminiert, d.h. abgeschlossen, wird. Das Voranschreiten eines Cases zwischen seiner Instanziierung und seiner Terminierung ist Thema des darauf folgenden Unterkapitels. Dort wird beschrieben, wie sich der aktuelle Zustand eines Cases, sowie die Zustände der zum Case gehörenden Aktivitätsinstanzen, auf weitere Ablaufmöglichkeiten des Cases auswirken.

5.2.1 Case-Instanziierung und -Terminierung

Ein Case wird gestartet, wenn eines der modellierten Startereignisse eintritt. Gehören zu einer Prozessdefinition mehrere BPDs, so reicht es, wenn nur eines der Startereignisse eintritt. Das Eintreten eines weiteren Startereignisses aus einem anderen BPD löst einen weiteren Case aus.

Ein Case gilt als beendet, wenn sich sämtliche Aktivitätsinstanzen dieses Cases in dem Zustand *durchgeführt* oder *ausgelassen* befinden. In diesem Moment kann davon ausgegangen werden, dass keine weiteren Tätigkeiten innerhalb dieses Cases mehr durchgeführt werden müssen. Unterstützt wird dieses Vorgehen durch die *finalen* Case-Zustände. Die Rolle von Case-Zuständen und wodurch genau sich ein *finaler* Case-Zustand auszeichnet, wird im folgenden Kapitel näher erläutert.

5.2.2 Case-Zustände und deren Übergänge

Der Fortschritt eines Cases wird durch seinen aktuell eingenommenen Zustand, den Case-Zustand (vgl. PCM-Metamodell, Abbildung 14, Klasse *Casezustand*), widergespiegelt. Neben der deskriptiven Funktion, die ein Case-Zustand damit übernimmt, hat er aber auch Einfluss auf die Ausführung eines Cases. In diesem Kapitel wird zunächst erläutert, welchen Zyklus ein Case-Zustand während der Abarbeitung eines Cases durchläuft und wie dabei Zustandsübergänge realisiert werden. Als Übergang zur Rolle der Aktivitäten in der Ausführungssemantik im darauffolgenden Kapitel, wird anschließend ebenfalls beschrieben, wie genau sich der Case-Zustand auf die Ausführung eines Cases auswirkt.

Im Laufe seiner Abarbeitung durchläuft ein Case in der Regel mehrere Zustände. Zu Beginn wird ihm dabei immer ein Initialzustand zugewiesen, welcher während der Definitionsphase des Prozesses festgelegt wird. Dies ist wichtig, da der aktuelle Zustand eines Cases mit darüber entscheidet, welche Aktivitätsinstanzen als nächstes durchführbar sind. Würde einem Case kein Initialzustand zugewiesen werden, wären dadurch auch keine Aktivitätsinstanzen durchführbar. Nach Zuweisung eines Initialzustandes bestimmen die Zustandsübergänge des definierten Prozesses darüber, wann und wodurch sich der Zustand eines Cases ändert. Wie Zustandsübergänge realisiert und ausgelöst werden, erklärt der anschließende Absatz. Schließlich erreicht der Case einen finalen Zustand, welcher den Abschluss des Cases zur Folge hat. Ein Case-Zustand wird dann als *final* betrachtet, wenn die beiden folgenden Sachverhalte auf ihn zutreffen:

1. Der Case-Zustand bzw. das Datenobjekt, welches den Case-Zustand trägt, ist assoziiert mit einem Sequenzfluss, der zu einem Endereignis führt.

2. In keinem der Prozessmodelle, auf denen der Case basiert, existieren Aktivitäten nach Übergang in diesen Case-Zustand.

Bei der fachlichen Aufnahme eines Prozesses, erläutert in Kapitel 4 dieser Arbeit, wurde beschrieben, dass die Zustände, die ein Case einnehmen kann in den Prozessmodellen mit Hilfe von Datenartefakten modelliert werden. Diese Datenartefakte werden dann mit den ausgehenden Sequenzflüssen jener Aktivitäten oder Ereignisse assoziiert, die den dann neuen Case-Zustand auslösen. Kapitel 4 zeigt am Beispielprozess *Ticketbearbeitung*, wie sich Zustandsübergänge aus Prozessmodellen ergeben. Bei möglichen Zustandsübergängen eines Cases muss es sich allerdings nicht zwangsläufig um solche handeln, die sich direkt aus den Prozessmodellen ergeben. Es können auch zusätzliche Zustandsübergänge frei fest-

gelegt werden. Bei der Definition dieser zusätzlichen Zustandsübergänge ist es wichtig, einen Auslöser, also eine Aktivität oder ein Zwischenereignis für diesen festzulegen. Bei der Gesamtheit aller zu einem Prozess gehörenden Zustandsübergänge, egal ob sich diese aus den Prozessmodellen ergeben oder zusätzlich definiert wurden, kommt es darauf an, dass sich keine einzige Kombination aus zwei Zustandsübergängen widerspricht. Dies ist immer dann der Fall, wenn zwei Zustandsübergänge

1. den gleichen Ausgangszustand haben,
2. durch den gleichen Auslöser angestoßen werden, aber
3. zu unterschiedlichen Zielzuständen führen.

Unter diesen Umständen wäre es dem PCM-System nicht möglich eindeutig zu ermitteln, in welchen neuen Zustand der Case übergehen soll. Aufgelöst werden kann so eine Situation einzig noch mit Hilfe von Bedingungen an den Zustandsübergängen. Dazu müssen die Zustandsübergänge an Bedingungen geknüpft sein, die nicht zur gleichen Zeit eintreten können. Tabelle 5 zeigt einige Beispiele von Zustandsübergängen, die die o.g. Kriterien erfüllen, jedoch nicht zwangsläufig widersprüchlich sind.

Ausgangs-zustand	Auslöser	Zielzustand	Bedingungen	W
Situation 1				
In Bearbeitung	Fehlerbericht prüfen	Fehlerbehebung		X
In Bearbeitung	Fehlerbericht prüfen	Geschlossen		
Situation 2				
In Bearbeitung	Fehlerbericht prüfen	Fehlerbehebung	istFehler==wahr	X
In Bearbeitung	Fehlerbericht prüfen	Geschlossen		
Situation 3				
In Bearbeitung	Fehlerbericht prüfen	Fehlerbehebung	istFehler==wahr	
In Bearbeitung	Fehlerbericht prüfen	Geschlossen	istFehler==falsch	

Tabelle 5: Widersprüchliche Zustandsübergänge

Situation 1 zeigt zwei Zustandsübergänge, die die drei o.g. Kriterien erfüllen, ohne dabei über jegliche Bedingungen zu verfügen. Das *X* in Spalte *W* zeigt daher an, dass es sich in dieser Situation um widersprüchliche Zustandsübergänge handelt. Dies ist auch in Situation 2 der Fall, da hier zwar der erste Zustandsübergang über eine Bedingung verfügt, der zweite allerdings nicht. Ist die Bedingung des ersten Zustandsübergangs erfüllt, kommt es zum Konflikt zwischen beiden Zustandsübergängen, da dann beide ausgelöst werden müssten. Nur in Situation 3 sieht man zwei Zustandsübergänge, die zwar den gleichen Ausgangszustand voraussetzen, über den gleichen Auslöser betätigt werden, dabei zu unterschiedlichen Zielzuständen führen und sich trotzdem nicht widersprechen. Dies liegt daran, dass die an ihnen definierten Bedingungen nicht zur gleichen Zeit eintreten können. Eine Umgebung zur Definition von Prozessen für die Ausführung via PCM sollte in der Lage sein, solche sich widersprechenden Zustandsübergänge zu erkennen und den bzw. die Anwender darüber zu informieren.

Teil der Ausführungssemantik ist es Zustandsübergänge während des Fortschritts eines Cases zu erkennen und ggf. durchzuführen. Dies wird realisiert, indem im-

mer dann, wenn ein Sequenzfluss beschritten, d.h. eine Aktivität durchgeführt wird oder ein Zwischenereignis eintritt, geprüft wird, ob dieser Sequenzfluss den Auslöser für einen Zustandsübergang darstellt. Ist dies der Fall, so muss als nächstes festgestellt werden, ob eventuell vorhandene Bedingungen für diesen Zustandsübergang erfüllt sind. Wie genau Bedingungen funktionieren wird später während der Zustandsprüfung von Aktivitätsinstanzen näher erläutert. Sind keine Bedingungen für diesen Zustandsübergang definiert, bzw. sämtliche Bedingungen erfüllt, erfolgt der Übergang des Cases in den neuen Zustand.

Die Realisierung von Zustandsübergängen ist jedoch nicht der einzige Beitrag von Case-Zuständen zur Ausführungssemantik. Wie bereits eingangs erwähnt, identifiziert der aktuelle Zustand eines Cases, an welcher Stelle der zugrunde liegenden Prozessmodelle er sich befindet. Dies stellt die Grundlage für die im nächsten Kapitel erläuterte Zustandsprüfung von Aktivitätsinstanzen dar. Dabei wird der aktuelle Case-Zustand in einem Prozessmodell gesucht und nur die darauffolgenden Aktivitäten als potentiell ausführbar betrachtet. Das bedeutet, dass Aktivitäten, denen im gesamten Prozessmodell kein Datenobjekt mit definiertem Case-Zustand vorausgeht für die Ausführungssemantik nicht erreichbar sind und es sich folglich um eine nicht sinnvolle Modellierung handelt. Dies wird auch im Kapitel 5.3 nochmals aufgegriffen. Kommt der aktuelle Case-Zustand nicht im untersuchten Prozessmodell vor, folgt der Case zu diesem Zeitpunkt nicht dem untersuchten Prozessmodell und daher kann keine der in diesem Modell vorkommenden Aktivitäten durchgeführt werden. Es ist aber nicht ausgeschlossen, dass der Case zu einem späteren Zeitpunkt einen Zustand einnimmt, der die Aktivitäten dieses Prozessmodells zur Durchführung bereitstellt.

5.2.3 Case-Fortschritt durch Aktivitäten

Zwar übernehmen Case-Zustände, wie im Kapitel zuvor beschrieben, neben ihrer deskriptiven Rolle auch einen wichtigen Teil der Ausführungssemantik, doch sind es vor allem die Aktivitäten, bzw. die sich aus ihnen ergebenen Aktivitätsinstanzen, die einen Case vorantreiben. Durch ihre Abarbeitung, egal ob sie erfolgreich durchgeführt oder ausgelassen werden, schreitet der Case voran. Entscheidendes Element einer PCM-Umgebung ist das situationsgerechte und modellierungskonforme Anbieten von durchführbaren Aktivitätsinstanzen. Dieses Kapitel beschäftigt sich daher mit der Frage, unter welchen Umständen die Aktivitätsinstanzen eines Cases zur Durchführung angeboten werden und welchen Lebenszyklus sie folglich durchleben.

Aktivitäten sind im Rahmen dieser Arbeit ausschließlich *atomare* Prozessschritte, die entweder manuell durch einen Menschen (*Human Task*) oder automatisch durch ein System (*Skript Task*) durchgeführt werden. Von der Instanziierung eines Cases bis hin zu seiner Terminierung, durchlaufen alle zum Case gehörenden Aktivitätsinstanzen den in Abbildung 18 gezeigten und im Folgenden erläuterten Lebenszyklus.

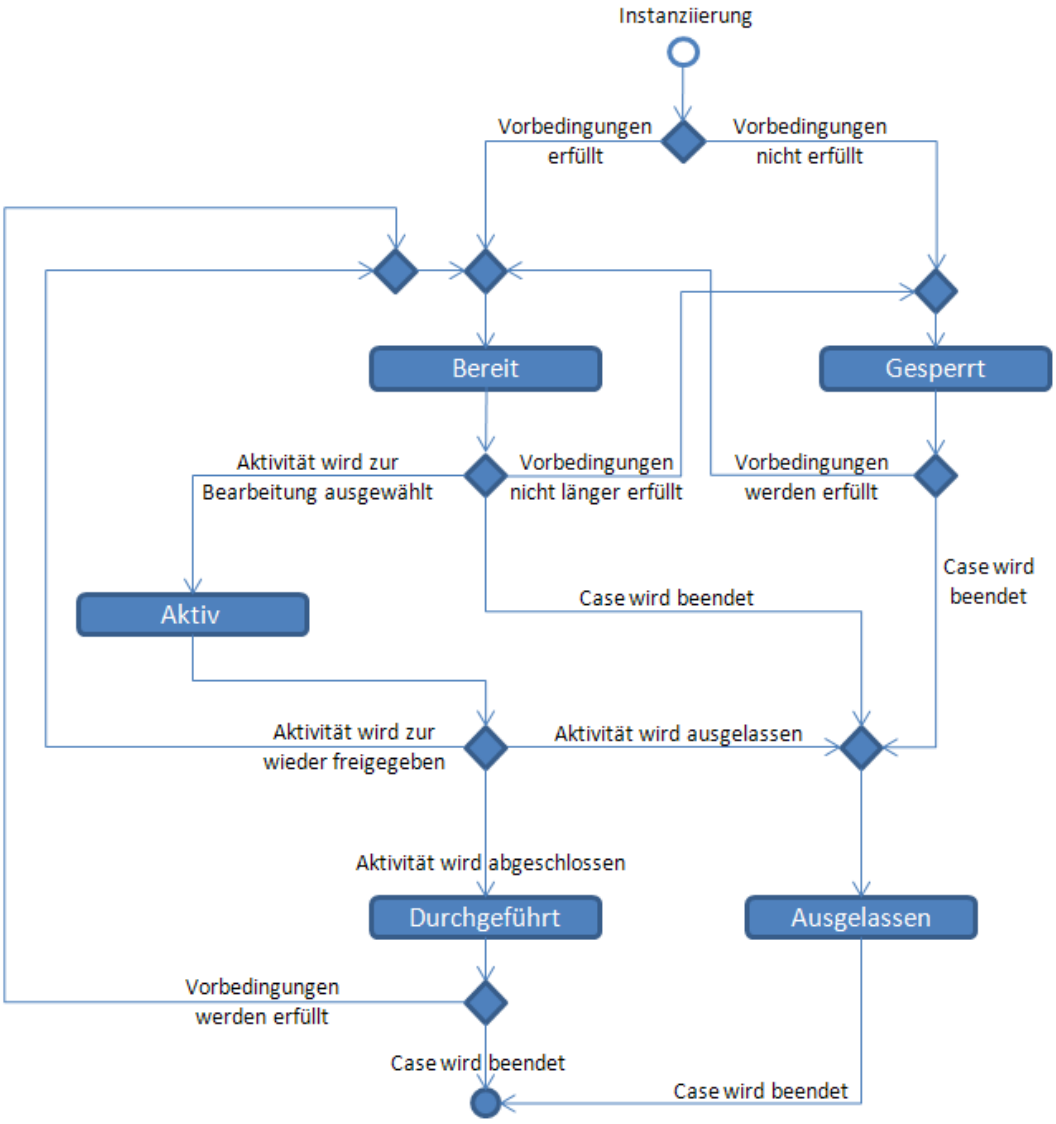


Abbildung 18: Lebenszyklus von Aktivitäten

Sobald ein Case zu einem Prozess instanziiert wird, werden auch sämtliche Aktivitäten, die zu dem definierten Prozess gehören, für den neuen Case instanziiert und in einen Initialzustand versetzt. Während im Rahmen von ACM sämtliche Aktivitäten eines Cases zu jedem Zeitpunkt, d.h. auch direkt zu Beginn des Cases, durchführbar sein sollen, orientiert sich PCM an dieser Stelle näher an der Ausfüh-

rungssemantik von BPMN 2.0. In dieser werden Aktivitäten erst durch das Auftreten einer entsprechenden Anzahl an Token in den Zustand „Bereit zur Ausführung“ versetzt.[46] Zuvor sind sie für die Ausführung gesperrt. Auch innerhalb eines PCM-Systems wird bei der Instanziierung eines Cases und damit auch seiner Aktivitätsinstanzen darüber entschieden, ob diese Aktivitätsinstanzen dem Case-Worker zur Durchführung angeboten werden oder ob sie zunächst für die Durchführung gesperrt werden. Für diese Entscheidung sind neben den zugrunde liegenden Prozessmodellen auch andere Faktoren von Bedeutung. Wie genau diese Entscheidung funktioniert erläutert in folgenden Abschnitt die *Zustandsprüfung von Aktivitätsinstanzen*. Mit der Frage, zu welchen Zeitpunkten in einem Case solch eine Zustandsprüfung durchgeführt werden muss, beschäftigt sich dann das darauf folgende Kapitel *Auslöser der Zustandsprüfung*.

5.2.3.1 Zustandsprüfung von Aktivitätsinstanzen

Die Zustandsprüfung von Aktivitätsinstanzen weist anhand der aktuellen *Case-Situation* jeder zum Case gehörenden Aktivitätsinstanz den passenden Zustand zu, und trifft damit die Entscheidung, welche Aktivitätsinstanzen dem Case Worker als nächstes zur Durchführung angeboten werden. Damit setzt sie den in Abbildung 18 gezeigten Lebenszyklus von Aktivitätsinstanzen in die Tat um und stellt somit ein zentrales Element eines PCM-Systems dar. Die zuvor erwähnte *Case-Situation* setzt sich aus folgenden drei Faktoren zusammen:

1. der Case-Zustand, den der Case zum Zeitpunkt der Zustandsprüfung inne hat,

2. die aktuellen Zustände der einzelnen Aktivitätsinstanzen, sowie

3. die Datenlage des Cases, d.h. die Werte, die die Datensätze des zentralen Datenobjektes zum Zeitpunkt der Zustandsprüfung inne haben.

Abbildung 19 skizziert, wie diese drei Einflussfaktoren als Eingabewerte auf die Zustandsprüfung von Aktivitätsinstanzen wirken und wie diese Zustandsprüfung die neuen Aktivitätszustände als Ausgabewerte zur Folge hat.

[46] Vgl. **OMG 2011**, S. 429

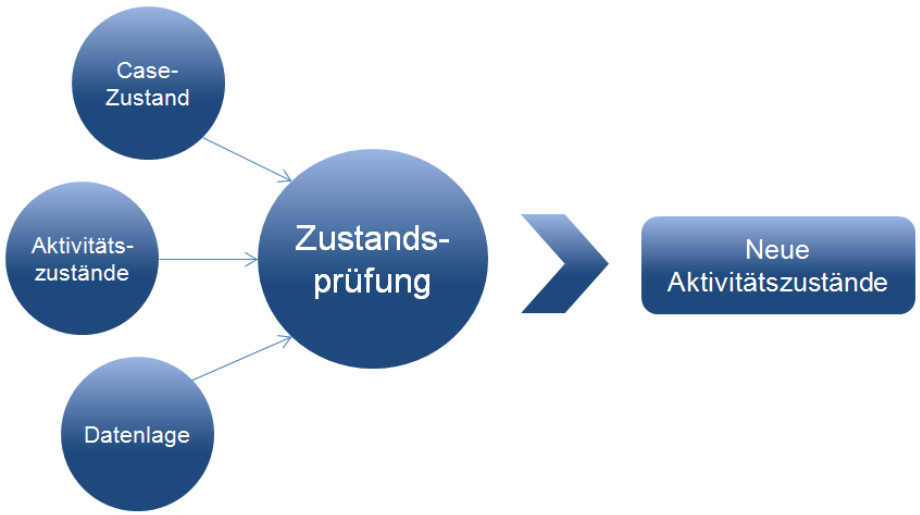


Abbildung 19: Zustandsprüfung von Aktivitätsinstanzen

Als erster Einflussfaktor auf die Berechnung der neuen Zustände von Aktivitätsinstanzen wurde der aktuelle Case-Zustand aufgeführt. Wie bereits in Kapitel 5.2.2 angedeutet, hilft dieser Case-Zustand dabei, genau die Stellen in den zugrunde liegenden Prozessmodellen aufzufinden, an denen sich der Case aktuell befindet. Damit steht das Auffinden von Datenobjekten, die den aktuellen Case-Zustand tragen, an erster Stelle der Zustandsprüfung von Aktivitätsinstanzen. Diese Datenobjekte sind stets mit einem Sequenzfluss assoziiert, wodurch der Sequenzfluss eines zum Case-Zustand passenden Datenobjektes als die Stelle, an der sich der Case befindet, identifiziert ist. Ausschließlich Aktivitäten, die sich *hinter* diesem Sequenzfluss befinden, können im weiteren Verlauf der Zustandsprüfung als potentiell durchführbar betrachtet werden. Sämtliche Aktivitäten *vor* diesem Sequenzfluss können in diesem Case-Zustand nicht zur Durchführung angeboten werden. An dieser Stelle wird deutlich, warum es wichtig ist, dass ein Case stets über einen Case-Zustand verfügt, d.h. auch direkt nach seiner Instanziierung. Würde er über keinen Case-Zustand verfügen, wären auch keine Aktivitätsinstanzen jemals durchführbar.

Nachdem nun die Rolle des aktuellen Case-Zustandes innerhalb der Zustandsprüfung von Aktivitätsinstanzen dargestellt wurde, soll nun erläutert werden, wie sich auch die aktuellen Zustände der Aktivitätsinstanzen auswirken. Durch das Auffinden eines zum Case-Zustand passenden Datenobjektes und dessen Sequenzfluss lässt sich auch der Zielknoten dieses Sequenzflusses bestimmen. Handelt es sich dabei um eine Aktivität, so besteht eine große Wahrscheinlichkeit, dass die zu dieser Aktivität gehörende Aktivitätsinstanz als nächstes zur Durchführung ange-

boten wird. Handelt es sich bei dem Zielknoten allerdings um ein Gateway, so werden die ausgehenden Sequenzflüsse dieses Gateways ebenfalls beschritten, so lange bis entweder eine Aktivität gefunden wird oder das Prozessmodell in einem Endereignis endet. Als nächstes werden die auf diese Weise gefundenen Aktivitäten bzw. deren Instanzen auf ihren aktuellen Zustand hin untersucht. Die Aktivitätsinstanzen werden nur dann in den Zustand *bereit* versetzt, und damit dem Case-Worker zur Durchführung angeboten, wenn sie nicht bereits den Zustand *ausgelassen* inne haben. Der Grund dieser Vorgehensweise besteht darin, dass das Auslassen von Aktivitätsinstanzen (vgl. Kap. 5.2.3.2) keinen Zustandsübergang zur Folge hat (vgl. Kap. 5.2.2). Daraus resultiert, dass eine Aktivitätsinstanz, die aufgrund des passenden Case-Zustands in den Zustand *bereit* versetzt und anschließend vom Case-Worker ausgelassen wurde, im nächsten Schritt sofort wieder zur Durchführung angeboten würde, da der Case-Zustand nach wie vor unverändert ist. Das heißt, dass wenn mit Hilfe des Case-Zustandes die passende Stelle in einem Prozessmodell gefunden wurde, sich die darauffolgende Aktivitätsinstanz allerdings im Zustand *ausgelassen* befindet, werden erneut Nachfolger dieser Aktivitätsinstanz ermittelt und ebenfalls auf ihren Zustand hin untersucht. Befindet sich eine auf diese Weise ermittelte Aktivitätsinstanz nicht im Zustand *ausgelassen,* wird sie in den Zustand *bereit* versetzt. Ihre Nachfolger hingegen erhalten den Zustand *gesperrt*. Dies gilt auch für Aktivitätsinstanzen, die sich bereits im Zustand *durchgeführt* befinden. Dadurch ermöglicht ein PCM-System den Rücksprung innerhalb der dem Case zugrunde liegenden Prozessmodelle. Dies veranschaulicht das Kapitel 6.2 mit Hilfe von Best Practices zur Erzeugung von Schleifen.

Listing 1 zeigt mit Hilfe von Pseudocode, wie die Zustandsprüfung bis zu dieser Stelle funktioniert. Der Einfluss des aktuellen Case-Zustandes findet sich in Zeile 2 wieder, in welcher Datenobjekte mit eben jenem aktuellen Case-Zustand, ermittelt werden. Anschließend werden die Nachfolger dieser Datenobjekte, das heißt die Zielknoten der zu den Datenobjekten gehörenden Sequenzflüsse, einzeln durch die Methode *prüfeNachfolger* untersucht werden. Handelt es sich bei einem dieser Nachfolger nicht um eine Aktivität, sondern bspw. um ein Gateway, werden wiederum dessen Nachfolger ermittelt (s. Zeile 20) und ein rekursiver Aufruf der Methode für diese Nachfolger gestartet. Handelt es sich jedoch um eine Aktivität und steht die Variable *aktivieren* auf *wahr* (Zeile 12), wird der Zustand dieser Aktivität bzw. ihrer Instanz ermittelt und ausgewertet. Steht dieser Zustand nicht auf *ausgelassen,* wird die Aktivitätsinstanz in den Zustand *bereit* versetzt (s. Zeile 15). Außerdem wird die Variable *aktivieren* auf *falsch* gesetzt, damit bei den nun folgen-

den, rekursiven Aufrufen kein Nachfolger dieser Aktivitätsinstanz mehr den Zustand *bereit* erhalten kann.

```
1:      aktivieren = wahr;
2:      List<Datenobjekt> datenobjekte =
        FindeDatenobjekteMitZustand(aktuellerCaseZustand);
3:
4:      Für jedes datenobjekt in datenobjekte
5:        nachfolger = FindeNachfolgerVon(datenobjekt);
6:        pruefeNachfolger(nachfolger);
7:      Ende Für;
8:
9:      Methode pruefeNachfolger(nachfolger):
10:     Für jeden nachfolger
11:
12:       WENN nachfolger ist eine Aktivität UND aktivieren
          ist wahr DANN
13:           zustand = ErmittleZustandVon(nachfolger);
14:           WENN zustand ist NICHT ausgelassen
15:                SetzeZustandAufBereit(nachfolger);
16:                aktivieren = falsch;
17:           Ende WENN;
18:       Ende WENN;
19:
20:       weitere_nachfolger = FindeNachfolgerVon(nachfolger);
21:       pruefeNachfolger(weitere_nachfolger);
22:     Ende Für;
```

Listing 1: Pseudocode zur Ermittlung von Aktivitätszuständen

Die angesprochenen, rekursiven Aufrufe durchlaufen das entsprechende Prozessmodell solange, bis das Ende jedes Pfades, d.h. ein Endereignis, erreicht wurde. Bei einem derartigen Durchlaufen eines Prozessmodells sind möglicherweise im Modell enthaltene Schleifen-Konstruktionen besonders zu beachten. Durch Schleifen können Aktivitäten zu ihren eigenen Nachfolgern werden und würden in dem oben gezeigten Pseudocode eine Endlos-Schleife verursachen. Eine produktive Implementierung muss hier spezielle Sicherheitsmechanismen

einbauen. Abbildung 20 zeigt anhand der roten Sequenzflüsse die Schleife im Beispielprozess *Ticketbearbeitung*.

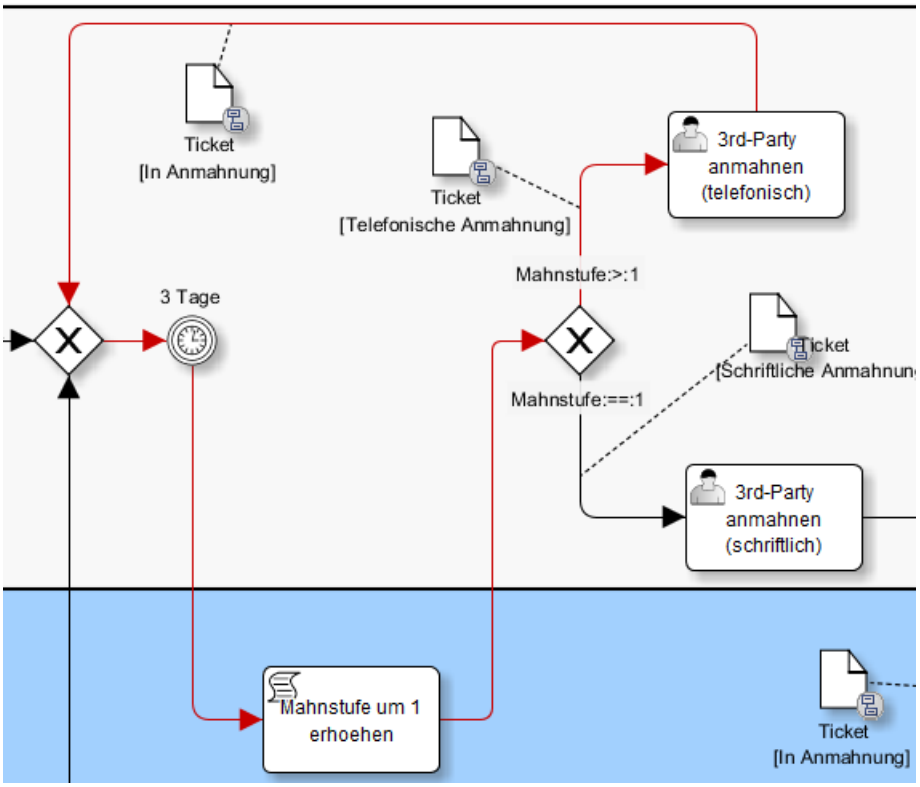


Abbildung 20: Schleife im Prozessmodell

Die bisherige Ermittlung durchführbarer Aktivitätsinstanzen auf Basis des Case-Zustandes und der Zustände von Aktivitätsinstanzen macht besonders deutlich, wie sich PCM, mit seiner Einbeziehung von Prozessmodellen, von ACM unterscheidet.

Im Folgenden wird der dritte und letzte Einflussfaktor der Zustandsprüfung von Aktivitätsinstanzen, die Datenlage des Cases, behandelt. Wie bereits eingangs erwähnt, setzt sich die Datenlage des Cases aus der Gesamtheit der Werte zusammen, welche die einzelnen Datensätze des zentralen Datenobjektes zum Zeitpunkt der Zustandsprüfung einnehmen. Diese Werte nehmen ihren Einfluss über Bedingungen, die während der Definitionsphase eines Prozesses an Aktivitäten oder Sequenzflüssen modelliert wurden. Eine Prüfung von Aktivitätsinstanzen auf vorliegende Bedingungen ist allerdings nur dann notwendig, wenn sich die untersuchte Aktivitätsinstanz nach der Auswertung des Case-Zustandes und der derzeitigen Zustände der Aktivitätsinstanzen noch immer im Zustand *bereit* befindet. Wurde sie bereits zuvor, z.B. aufgrund des Case-Zustandes, gesperrt, dann ist eine Prüfung möglicher Bedingungen nicht notwendig.

Das vorgestellte PCM-Modell erlaubt die Definition von Vor- und Nachbedingungen an Aktivitäten, wobei für die in diesem Kapitel erläuterte Zustandsprüfung für Aktivitätsinstanzen lediglich Vorbedingungen von Bedeutung sind. Auf Nachbedingungen wird im nächsten Kapitel im Rahmen des Case-Fortschritts durch Aktivitäten eingegangen. Vorbedingungen setzen einen Datensatz über einen Operator mit einem bestimmten Wert in Assoziation. In dem in Kapitel 4 vorgestellten Beispielprozess *Ticketbearbeitung* könnte die Aktivität *Antwort von 3rd-Party erfassen* bspw. die fachliche Vorbedingung *Antwort von 3rd-Party muss vorliegen* erhalten. Technisch ausgedrückt würde das heißen, dass der Datensatz *3rd-Party Kommentar* aus dem Datenmodell nicht leer sein darf. Den technischen Ausdruck zeigt Abbildung 21, wobei *set* den Operator für *gesetzt* darstellt und ausdrückt, dass der zur Bedingung gehörende Datensatz, in diesem Beispiel also *3rd-Party-Kommentar*, nicht leer sein darf. Tabelle 6 zeigt weitere mögliche Operatoren zur Erstellung von Bedingungen, welche im Anhang A ausführlich und anhand von Beispielen erläutert werden.

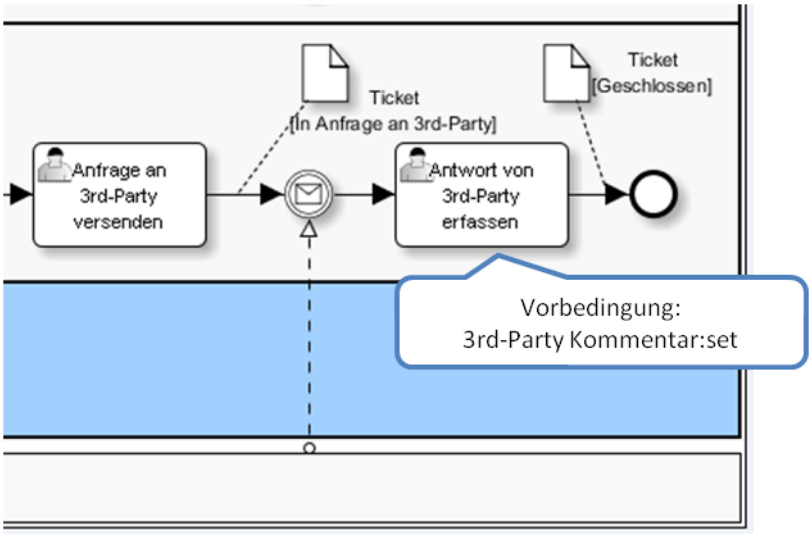


Abbildung 21: Vorbedingung für eine Aktivität

Verfügt eine Aktivitätsinstanz zum Zeitpunkt der Zustandsprüfung über eine oder mehrere Vorbedingungen, die nicht erfüllt sind, erhält sie den Zustand *gesperrt* und ist damit nicht durch den Bearbeiter ausführbar. Wird eine Aktivität in mehreren dem Prozess zugrunde liegenden Prozessmodellen verwendet, sollte sie immer über die gleichen Vorbedingungen verfügen. Andernfalls würde es sich fachlich gesehen nicht um die gleiche Aktivität handeln. Realisiert werden kann dies in BPMN-Modellen am besten mit Hilfe von *Aufruf-Aktivitäten*, die es ermöglichen,

Aktivitäten, die bereits an anderer Stelle modelliert wurden, wiederzuverwenden. Kapitel 6.2.4 liefert dazu ein entsprechendes Best-Practice.

Operator	==	!=	<	>	set	!set
Bezeichnung	gleich	ungleich	kleiner	größer	gesetzt	nicht gesetzt

Tabelle 6: Mögliche Operatoren für Bedingungen

Bedingungen können auch direkt an Sequenzflüssen definiert werden, wodurch sie, wenn solch ein Sequenzfluss eine Aktivität als Zielknoten hat, ebenfalls zur Vorbedingung dieser Aktivität werden. Dabei funktionieren diese an Sequenzflüsse geschriebenen Bedingungen auf die gleiche Art und Weise, wie die zuvor erläuterten Vorbedingungen, d.h. auch hier wird ein Datensatz aus dem Datenmodell mit einem angegebenen Operator auf einen bestimmten Wert geprüft. Ist die sich daraus ergebene Bedingung nicht erfüllt, erhält die Aktivität, die das Ziel des untersuchten Sequenzflusses ist, den Zustand *gesperrt*. Im Beispielprozess *Ticketbearbeitung* befindet sich solch eine Bedingung u.a. vor der Aktivität *Fehlerbericht prüfen*, welche nur dann in den Zustand *bereit* versetzt werden kann, wenn der Datensatz *Komponente* den Wert *GUI* eingenommen hat. Abbildung 22 veranschaulicht die Funktionsweise am genannten Beispiel.

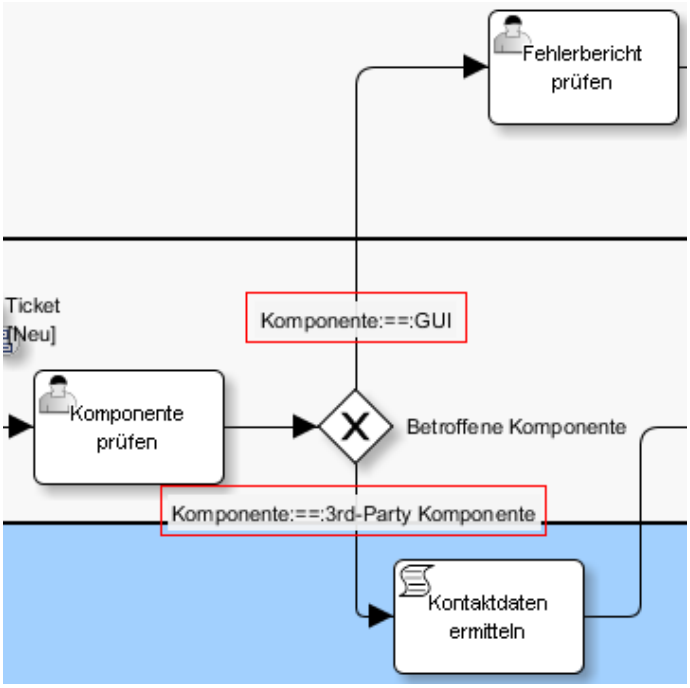


Abbildung 22: Bedingungen an einem Sequenzfluss

Für eine Aktivität macht es funktional betrachtet keinen Unterschied, ob sie mit einer Vorbedingung versehen wurde, oder ob ihr eingehender Sequenzfluss über eine Bedingung verfügt. Dass trotzdem beide Möglichkeiten existieren, begründet sich mit unterschiedlichen Modellierungsaspekten. Bedingungen an Sequenzflüssen kommen demnach meist nach einem datenbasierten Gateway zum Einsatz, um die, basierend auf dem geprüften Datensatz, möglichen Ablaufpfade zu kennzeichnen. In der Regel besitzen sämtliche ausgehende Sequenzflüsse aus solch einem Gateway Bedingungen, die den gleichen Datensatz abfragen und die verschiedenen Eintrittsmöglichkeiten behandeln. Um einen Stillstand des Cases zu vermeiden, sollte ein ausgehender Sequenzfluss eines exklusiven Gateways als *Standard-Sequenzfluss*[47] markiert werden, der dann durchlaufen wird, wenn keine der anderen Bedingungen zutrifft. Vorbedingungen direkt an Aktivitäten zu modellieren empfiehlt sich dann, wenn diese eher in der Natur der Aktivität selber liegt als das Ergebnis einer vorhergehenden Entscheidung zu sein. BPMN sieht bislang keine Methode und kein spezielles Element dafür vor, Vorbedingungen zu modellieren. Allerdings können Aktivitäten gemäß der BPMN 2.0 Spezifikation mit beliebig vielen Eigenschaften (*Properties*[48]) belegt werden. Diese Eigenschaften stellen eine Möglichkeit dar, Vorbedingungen an Aktivitäten zu hinterlegen.

Die Zustandsprüfung von Aktivitätsinstanzen wird zum ersten Mal beim Instanziieren eines Cases durchgeführt. Allerdings muss diese Prüfung an verschiedenen Stellen während des Case-Verlaufes immer wieder durchgeführt werden. Nachdem sämtliche Aktivitätsinstanzen mit einem Initialzustand belegt wurden, wird der Case schließlich durch den Case-Worker und äußere Ereignisse vorangetrieben. Damit die Aktivitätsinstanzen eines Cases den bereits zuvor in Abbildung 18 gezeigten Lebenszyklus durchlaufen können, muss die Zustandsprüfung von Aktivitäten regelmäßig wiederholt werden. Ein zeitliches Intervall für die Zustandsprüfung ist dafür allerdings nicht geeignet, da

1. ein zu kurz gewähltes Zeitintervall, unter Berücksichtigung der möglichen Komplexität von Prozessen und deren Cases, unnötig viel Last auf das verarbeitende System bringt und da

2. ein zu lang gewähltes Zeitintervall das zeitgerechte Abarbeiten eines Cases behindert, indem bspw. Aktivitäten, die sich *eigentlich* im Zustand *bereit* befinden müssten, noch bis zur nächsten Zustandsprüfung im Zustand *ge-*

[47] Vgl. **OMG 2011**, S. 98
[48] Vgl. **OMG 2011**, S. 210

sperrt verbleiben und dem Bearbeiter dadurch nicht zur Abarbeitung angeboten werden.

Folglich muss die Zustandsprüfung von Aktivitäten durch bestimmte Ereignisse ausgelöst werden, nämlich immer dann wenn sich die *Case-Situation* verändert hat. Durch welche Ereignisse sich eine Case-Situation verändert und eine erneute Zustandsprüfung nötig wird, behandelt das nächste Kapitel.

5.2.3.2 Auslöser der Zustandsprüfung

Dieses Kapitel beschäftigt sich mit der Frage danach, zu welchen Zeitpunkten eine Zustandsprüfung für Aktivitätsinstanzen in einem Case erfolgen muss, d.h. welche Ereignisse die *Case-Situation* so verändern, dass erneut geprüft werden muss, welche Aktivitätsinstanzen nun dem Case-Worker zur Ausführung angeboten werden können. Dazu wird u.a. auf die unterschiedlichen Optionen des Case-Workers bei der Abarbeitung von Aktivitätsinstanzen und deren erwartete Auswirkungen auf den Case und dessen weitere Aktivitäten eingegangen. Dies beinhaltet auch die Auswirkungen, die Datenänderungen bei der Abarbeitung von Aktivitäten verursachen. Außerdem soll der Einfluss von externen Datenänderungen behandelt werden.

Es wurde bereits erläutert, aus welchen Gründen eine Aktivitätsinstanz eines Cases sich im Zustand *gesperrt* befinden kann. Einer dieser Gründe war eine nicht erfüllte Vorbedingung dieser Aktivitätsinstanz, sprich, dass ein Datensatz nicht den *richtigen* Wert besitzt um die Aktivitätsinstanz in den Zustand *bereit* zu versetzen. Dieser Wert kann sich jedoch im Laufe der Bearbeitungszeit des Cases ändern. Mit jeder Datenänderung ist es also potentiell möglich, dass sich der Zustand einer oder mehrerer Aktivitätsinstanzen ändert. Denkbar sind dabei folgende zwei Varianten:

1. Eine Aktivitätsinstanz befindet sich im Zustand *gesperrt*. Durch eine Datenänderung wird auch die letzte Vorbedingung dieser Aktivitätsinstanz erfüllt, wodurch sie (wenn sie nicht aufgrund einer Bedingung an einem Sequenzfluss oder aufgrund des Prozessflusses gesperrt ist), in den Zustand *bereit* versetzt wird.

2. Eine Aktivitätsinstanz befindet sich im Zustand *bereit*. Durch eine Datenänderung wird eine zuvor erfüllte Vorbedingung dieser Aktivitätsinstanz nicht länger erfüllt. Sie erhält dadurch den Zustand *gesperrt*. Befindet sich die

Aktivitätsinstanz zum Zeitpunkt der Datenänderung bereits im Zustand *aktiv* oder wurde sie sogar bereits abgearbeitet, behält sie ihren aktuellen Zustand.

⇨ Der erste Auslöser für die Zustandsprüfung von Aktivitätsinstanzen ist also die Veränderung eines oder mehrerer Datensätze.

Aktivitäten im Zustand *bereit* können durch den Bearbeiter des Cases zur Bearbeitung ausgewählt werden und erhalten dadurch den Zustand *aktiv*, welcher aufzeigt, dass sich die Aktivität in Bearbeitung befindet. Ein Human Task behält den Zustand *aktiv*, bis

- der Case-Worker ihn als abgeschlossen kennzeichnet (z.B. durch Button-Klick) und gleichzeitig sämtliche Nachbedingungen erfüllt sind (neuer Zustand: *durchgeführt*), oder
- der Case-Worker ihn auslässt (neuer Zustand: *ausgelassen*), oder
- der Case-Worker ihn unverrichteter Dinge schließt und ihn damit wieder freigibt (neuer Zustand: *bereit*).

Skript Tasks werden durch ihre Aktivierung direkt ausgeführt und gehen daher anschließend unmittelbar in den Zustand *durchgeführt* über.

⇨ Der zweite Auslöser für eine erneute Zustandsprüfung ist demnach das Auswählen einer Aktivitätsinstanz zur Bearbeitung.

Durch das erfolgreiche Abschließen einer Aktivitätsinstanz kann, wie in Kapitel 5.2.2 beschrieben, der Case in einen neuen Case-Zustand versetzt werden. Da der Case-Zustand einen erheblichen Einfluss darauf hat, welche Aktivitätsinstanzen als nächstes zur Durchführung bereitstehen, ist an dieser Stelle eine erneute Zustandsprüfung der Aktivitätsinstanzen erforderlich. Allerdings können Aktivitätsinstanzen nur dann erfolgreich abgeschlossen werden, wenn eventuell definierte Nachbedingungen der Aktivität in diesem Case erfüllt sind. Nachbedingungen sorgen also dafür, dass Aktivitätsinstanzen nicht vorzeitig, d.h. ohne Erfüllung ihrer *muss*-Kriterien, abgeschlossen werden. Wenn sämtliche Nachbedingungen einer

Aktivitätsinstanz erfüllt sind, findet ein Zustandswechsel der Aktivitätsinstanz in den Zustand *durchgeführt* statt. Andernfalls verbleibt die Aktivitätsinstanz im Zustand *aktiv*.

⇨ Das erfolgreiche Abschließen einer Aktivitätsinstanz ist damit der dritte Auslöser einer erneuten Zustandsprüfung der Aktivitätsinstanzen.

Ein Grundsatz von Case Management ist das Vorantreiben eines Cases durch die Entscheidungen des Bearbeiters, welche Aufgaben zu erledigen sind. Dabei entscheidet der Bearbeiter auch welche Aktivitäten *nicht* durchgeführt werden, obwohl diese in einem oder mehreren der Prozessdiagramme modelliert wurden. Um diesen Grundsatz zu erfüllen, können Aktivitäten ausgelassen werden. Damit signalisiert der Bearbeiter, dass die Durchführung dieser Aktivität nicht zum Abschließen dieses konkreten Cases notwendig ist. Durch das Auslassen einer Aktivitätsinstanz erhalten deren Nachfolger die Möglichkeit den Zustand *bereit* einzunehmen, was eine erneute Zustandsprüfung aller Aktivitätsinstanzen des Cases erforderlich macht. Allerdings ist das Auslassen von Aktivitätsinstanzen nur unter bestimmten Voraussetzungen möglich. Das Auslassen einer Aktivitätsinstanz zieht keinen Wechsel des Case-Zustandes nach sich. Würde es sich bei einer ausgelassenen Aktivitätsinstanz um die derzeit einzige im Zustand *bereit* handeln und würde ihr im Prozessmodell keine weitere Aktivität folgen, z.B. weil sie einzig in einem separaten Prozessmodell modelliert wurde, so würde anschließend keine weitere Aktivitätsinstanz in den Zustand *bereit* versetzt werden können. Der Case würde sich in einem *Deadlock* befinden. Um sicher zu stellen, dass durch das Auslassen einer Aktivitätsinstanz der Case immer weiterlaufen kann, darf eine Aktivitätsinstanz nur dann ausgelassen werden, wenn sich mindestens einer ihrer Nachfolger ebenfalls bereits im Zustand *bereit* befindet. Dieser Effekt lässt sich durch das Nutzen mehrerer Prozessmodelle erzielen. Wie Prozessmodelle modelliert werden müssen, damit bestimmte Aktivitäten ausgelassen werden können, zeigt ein entsprechendes Beispiel im Kapitel 6.2.

⇨ Damit wurde das Auslassen als ein weiterer Auslöser einer Zustandsprüfung identifiziert.

Zusammenfassend lässt sich sagen, dass eine Zustandsprüfung der Aktivitäten eines Cases immer nach folgenden Ereignissen durchgeführt werden muss:

1. Die Veränderung eines oder mehrerer Datensätze
2. Die Auswahl einer Aktivitätsinstanz zur Bearbeitung
3. Das erfolgreiche Abschließen einer Aktivitätsinstanz
4. Das Auslassen einer Aktivitätsinstanz

In einer produktiven Umsetzung sollte das Eintreten dieser Ereignisse unbedingt behandelt werden, um eine Zustandsprüfung auszulösen.

5.3 Validierung von Prozessdefinitionen

In der zuvor erläuterten Ausführungssemantik wurde bereits an verschiedenen Stellen auf Sachverhalte hingewiesen, die eine sinnvolle Ausführung von Cases verhindern. Dieses Kapitel soll anhand des ebenfalls vorgestellten PCM-Metamodells aufzeigen, wie widersprüchliche Modellierungen und Definitionen erkannt und somit verhindert werden können. Diese Validierung soll dafür Sorge tragen, dass das Metamodell und die Ausführungssemantik ihre vorgesehenen Zwecke erfüllen können.

Zu Beginn einer solchen Validierung sollten bestimmte Prämissen, die im Laufe der Ausführungssemantik vorausgesetzt wurden, überprüft werden. Dazu gehört zum Beispiel, dass Datenobjekte nur mit Sequenzflüssen assoziiert werden dürfen. Gemäß des PCM-Metamodells erbt die Klasse *Datenobjekt* von der Klasse *Knoten*, welche sowohl mit Assoziationen als auch mit Sequenzflüssen in Beziehung stehen kann. Um die eben genannte Prämisse zu erfüllen, dürfen Datenobjekte jedoch nur mit Assoziationen verknüpft sein. Wie aus Abbildung 23 hervorgeht, dürfen diese Assoziationen nur mit Sequenzflüssen verbunden sein. Ist in einem der Prozessmodelle ein Datenobjekt jedoch via Assoziation mit einem anderen Knoten, zum Beispiel einer Aktivität, verbunden, widerspricht dies dem Metamodell und der Anwender sollte über diesen Modellierungsfehler in Kenntnis gesetzt werden.

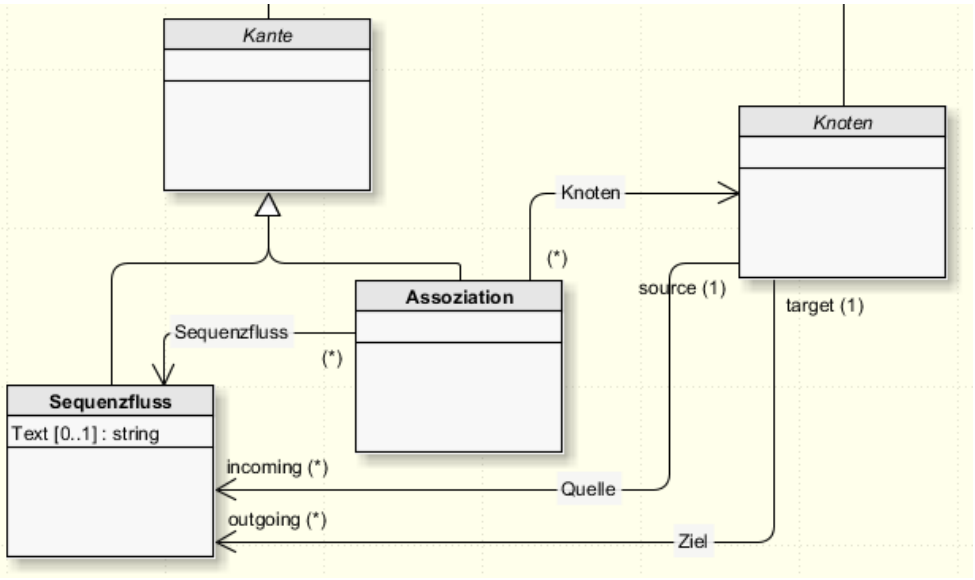


Abbildung 23: Auszug PCM-Metamodell, Assoziationen

Des Weiteren wurden in der Ausführungssemantik nur bestimmte Elemente der BPMN berücksichtigt. Zur Gewährleistung der gewünschten Funktionalität sollten Prozessmodelle daraufhin validiert werden, dass sie ausschließlich diese Elemente enthalten. Gemäß PCM-Metamodell werden nur die Elemente *Ereignis*, *Aktivität* und *Exklusives Gateway* unterstützt. Andere BPMN-Elemente wie z.B. parallele Gateways sollten per Validierung den Anwender auf eine nicht konforme Modellierung gemäß der vorgestellten Ausführungssemantik hinweisen.

Die Klasse *Datenobjekt* stellt in dem PCM-Metamodell den Übergang zwischen Prozess- und Datenmodellierung dar. Aus der verwendeten Kardinalität zwischen *Prozess* und *Datenobjekt* geht eindeutig hervor, dass es innerhalb eines Prozesses nur ein einziges Datenobjekt gibt. Dies kann und soll in den Prozessmodellen zwar mehrfach wiederverwendet werden, doch bezieht es sich immer auf die gleiche Klasse des zugehörigen Datenmodells. Das bedeutet, dass nach der fertigen Definition eines Prozesses und der Erfassung seiner Bestandteile die Anzahl der Ausprägungen der Klasse *Datenobjekt* genau *1* betragen muss.

Das PCM-Konzept erlaubt an verschiedenen Stellen die Definition von Bedingungen. Diese können an Sequenzflüssen, als Vor- oder Nachbedingungen an Aktivitäten oder in zusätzlichen Zustandsübergängen auftauchen. Gemäß des PCM-Metamodells muss jede Bedingung auf genau einen Datensatz verweisen, welcher wiederum ein Element des zentralen Datenobjektes des Prozesses sein muss. Teil einer Validierung muss es daher sein zu überprüfen, dass jeder in einer Bedingung genannte Datensatz auch tatsächlich im zentralen Datenobjekt existiert.

Wird eine Bedingung bspw. in Textform an einen Sequenzfluss geschrieben, würde bereits ein einfacher Schreibfehler bei der Nennung des Datensatzes dazu führen, dass diese Bedingung zur Laufzeit nicht sinnvoll ausgewertet werden kann. Folgendes Listing 2 zeigt einen Pseudocode zur Durchführung dieser Validierung.

```
1: List<Bedingung> alleBedingungen = ladeAlleBedingungen();
2: List<Datensatz> alleDatensätze = ladeAlleDatensätze();
3: FÜR JEDE bedingung AUS alleBedingungen
4:       Datensatz ds = ladeDatensatzAus(bedingung);
5:       WENN ds IST NICHT ELEMENT AUS alleDatensätze
6:         meldeFehler();
7: ENDE FÜR
```

Listing 2: Pseudocode Validierung von Bedingungen

Wichtig für eine erfolgreiche Abwicklung von Cases ist auch das Vorkommen von mindestens einem finalen Case-Zustand. Diese wurden definiert als solche, denen in keinem der modellierten Prozessmodelle eine Aktivität folgt. Aus dem PCM-Metamodell geht hervor, dass ein Case-Zustand über ein Datenobjekt definiert wird. Ein Datenobjekt wiederum wird über eine Assoziation mit einem Sequenzfluss verbunden und dieser Sequenzfluss hat einen Zielknoten. Dieser Zielknoten stellt den ersten Nachfolger eines Datenobjektes dar und hat, wenn es sich nicht um ein Endereignis handelt, weitere ausgehende Sequenzflüsse mit den nächsten Nachfolgern als Zielknoten. Abbildung 24 zeigt, als Teilabbildung des PCM-Metamodells, diese Zusammenhänge.

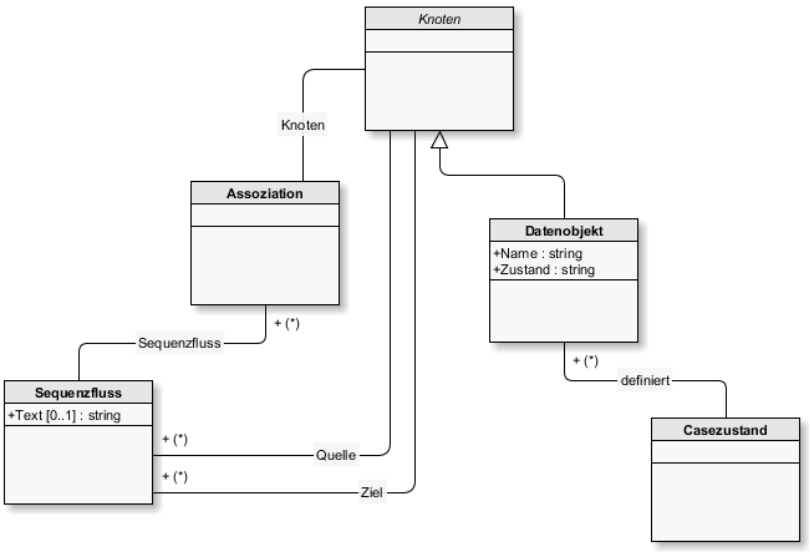


Abbildung 24: PCM-Metamodell, Nachfolger

Somit lassen sich für alle modellierten Datenobjekte die nachfolgenden Knoten bestimmen. Um festzustellen, ob es sich bei einem Case-Zustand um einen finalen Zustand handelt, muss in jedem Prozessmodell nach diesem Case-Zustand gesucht und festgestellt werden, ob sich unter dessen nachfolgenden Knoten eine Aktivität befindet. Sollte dies in keinem der Prozessmodelle der Fall sein, handelt es sich um einen finalen Case-Zustand und das in der Klasse *Case-Zustand* vorhandene Attribut *Endzustand* kann auf *wahr* gesetzt werden. Die Validierung überprüft dann nach der Prozessdefinition, ob die Anzahl der Case-Zustände, bei denen dieses Attribut auf *wahr* steht, mindestens *1* beträgt.

Ähnlich wie das zuvor beschriebene Auffinden von Nachfolgern, funktioniert auch die Identifikation von Vorgängern eines bestimmten Knotens. Jeder Knoten, bei dem es sich nicht um ein Startereignis handelt, verfügt über einen oder mehrere eingehende Sequenzflüsse. Die Quell-Knoten dieser Sequenzflüsse stellen die ersten Vorgänger des untersuchten Knotens dar und verfügen ggf. selber ebenfalls über Vorgänger. Die Sequenzflüsse, die beim Auffinden von Vorgängern beschritten werden, können außerdem daraufhin untersucht werden, ob sie über Assoziationen mit Datenobjekten verbunden sind. Die dabei gefundenen Datenobjekte und die an ihnen definierten Case-Zustände sind dann ebenfalls Vorgänger des ursprünglich untersuchten Knotens. Auf diese Weise lässt sich eine weitere Validierung auf sinnvolle Prozessdefinitionen vornehmen. Sämtliche modellierte Aktivitäten sollten daraufhin untersucht werden, ob sich unter ihren Vorgängern mindestens ein Datenobjekt mit definiertem Case-Zustand befindet. Wie in Kapitel 5.2.3.1 beschrieben, dienen Case-Zustände dazu, in einem Prozessmodell die nächstmöglichen Aktivitäten aufzufinden. Würde eine Aktivität keinen Case-Zustand als Vorgänger haben, wäre sie somit zur Laufzeit nicht *erreichbar* und würde folglich niemals zur Durchführung angeboten werden.

Damit eine konsistente Ausführung von definierten Prozessen gewährleistet werden kann, ist es essentiell zu überprüfen, dass sich die Zustandsübergänge des Prozesses nicht widersprechen. Das Widersprechen von zwei Zustandsübergängen hätte zur Folge, dass wenn der entsprechende Auslöser eintritt, das System nicht ableiten könnte, in welchen neuen Zustand der Case versetzt werden soll. Die verschiedenen Zustandsübergänge ergeben sich entweder aus den Prozessmodellen oder wurden zusätzlich definiert. Laut PCM-Metamodell besteht jeder Zustandsübergang aus einem alten Zustand, einem neuen Zustand, einem Auslöser und optional aus einer oder mehreren Bedingungen. Zur Validierung auf sich widersprechende Zustandsübergänge werden diese einzeln miteinander verglichen. Listing 3 zeigt dieses Vorgehen.

```
1: List<Zustandsuebergang> alleZustandsuebergaenge =
        ladeAlleZustandsübergaenge();
2: FÜR JEDEN zustandsuebergang AUS alleZustandsuebergaenge
3:      WENN zustandsuebergangInKonflikt(zustandsuebergang)
4:        meldeFehler();
5: ENDE FÜR
```

Listing 3: Validierung auf widersprüchliche Zustandsübergänge

Die Methode *zustandsuebergangInKonflikt(Zustandsuebergang)* in Zeile 3 übernimmt dabei die Prüfung, ob sich der untersuchte Zustandsübergang in Konflikt zu einem der anderen Zustandsübergänge befindet und meldet *wahr* oder *falsch* zurück. Wird ein Konflikt entdeckt und *wahr* zurückgemeldet, wird dem Anwender ein entsprechender Fehler gemeldet. Wie genau die Prüfung auf einen Konflikt funktioniert zeigt Listing 4.

```
1: METHODE zustandsuebergangInKonflikt(Zustandsuebergang zu)
2:  List<Zustandsuebergang> alleZustandsuebergaenge =
        ladeAlleZustandsübergaenge();
3:  FÜR JEDEN tmp AUS alleZustandsuebergaenge
4:      WENN zu.alterZustand == tmp.alterZustand
5:       WENN zu.ausloeser == tmp.ausloeser
6:        WENN zu.neuerZustand != tmp.neuerZustand
7:         WENN zu.bedingungen == tmp.bedingungen
8:          RETURN wahr
9:  ENDE FÜR
10: RETURN falsch
11:ENDE METHODE
```

Listing 4: Prüfung auf Zustandsübergang in Konflikt

Das Listing zeigt, wie der zu überprüfende Zustandsübergang *zu* mit jedem weiteren Zustandsübergang *tmp* verglichen wird. Die Methode gibt den Wert *wahr* zurück, und signalisiert damit einen gefundenen Konflikt, wenn die verglichenen Zustandsübergänge über den gleichen Ausgangszustand, den gleichen Auslöser, aber über einen unterschiedlichen Zielzustand verfügen. Außerdem müssen eventuelle Bedingungen ebenfalls gleich sein, um einen Konflikt zu erzeugen. Wie genau sich Bedingungen auf mögliche Konflikte von Zustandsübergängen auswirken, wurde in Kapitel 5.2.2 erläutert.

6 Prototypische Umsetzung und Best Practices

Dieses Kapitel stellt die prototypische Implementierung eines PCM-Frameworks, wie es im Rahmen dieser Masterarbeit konzipiert wurde, vor. Zunächst wird dazu die Architektur des Prototyps mit seinen unterschiedlichen Komponenten, sowie deren Zusammenspiel erläutert. Anschließend werden einige Best-Practices zur PCM-Modellierung vorgestellt. Diese bedienen sich dem Szenario eines Angebotsprozesses und dienen im späteren Verlauf des Kapitels der Veranschaulichung der Funktionalitäten des Prototyps. Diese Funktionalitäten sind Gegenstand des dann folgenden Kapitels, in welchem erläutert wird, wie ein Prozess definiert werden kann und wie er anschließend validiert und zur Ausführung gebracht wird. Das Kapitel endet mit der Dokumentation der Schnittstelle, über welche auch abseits der zum Prozesspaket gehörenden technischen Workflows die Daten eines Prozesses und dessen Cases ausgelesen und verändert werden können.

6.1 Architektur

Der entwickelte Prototyp besteht im Wesentlichen aus einem mit Java entwickelten Plugin für die inubit Suite, sowie einem Paket an technischen Workflows, die dieses Plugin verwenden.

Das erstellte Plugin erweitert die inubit Suite um ein neues Modul, welches in technischen Workflows verwendet werden kann und auf welches im Folgenden als *PCM Utility* verwiesen wird. Für die Entwicklung eigener Plugins stellt die inubit Suite ein SDK bereit und erläutert die nötige Vorgehensweise in ihrer Administrationsanleitung. Abbildung 25 aus dieser Anleitung skizziert, wie Eingabedaten- und Variablenströme in ein Modul hineingegeben, dort verarbeitet und am Ende als Ausgabe- und Variablenströme zurückgegeben werden.

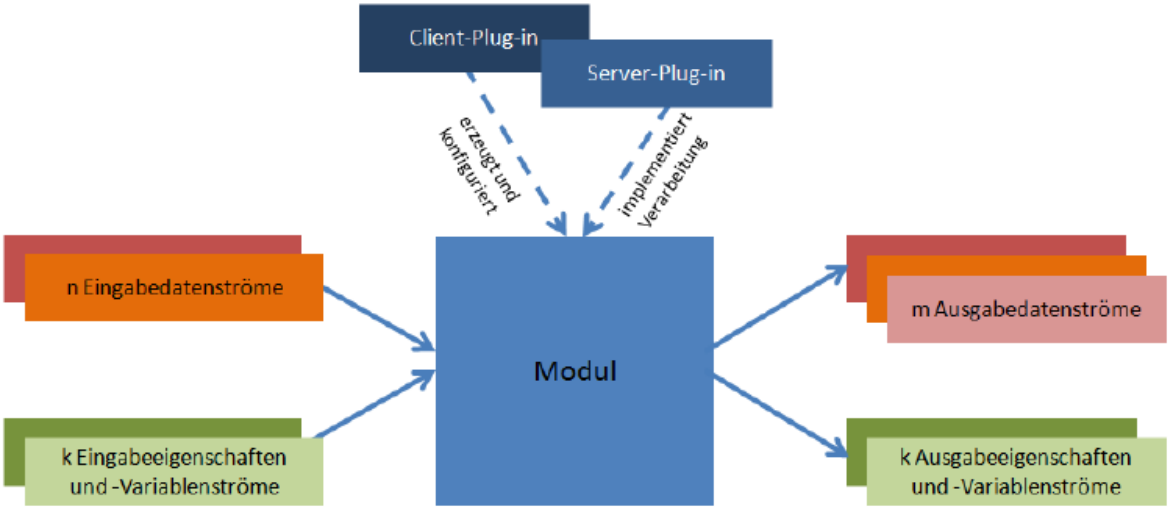


Abbildung 25: Plugin-Funktionsweise

[Quelle: **inubit Suite 6.1 Administration Guide**]

Das PCM Utility verfügt sowohl über eine Client-Plugin-Klasse, die eine graphische Oberfläche zur Einstellung der einzelnen Prozessparameter bereitstellt, als auch über eine Server-Plugin-Klasse, in welcher die implementierten Teile der Ausführungssemantik verarbeitet werden. Abbildung 26 zeigt die drei Packages, in denen sich der Quellcode des Plugins befindet.

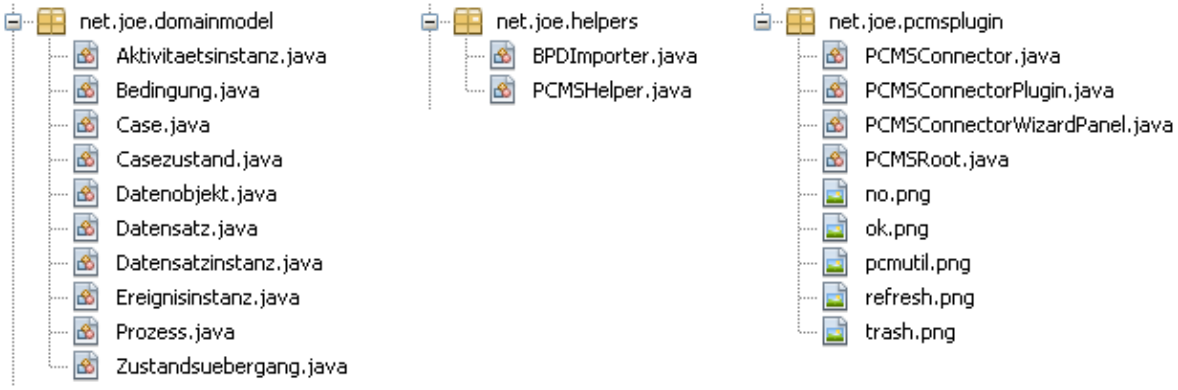


Abbildung 26: Die drei Packages des Java-Plugins

In dem Package *net.joe.domainmodel* befinden sich die Klassen zur Umsetzung des PCM-Metamodells, mit Ausnahme der Klassen, die BPMN-Modelle und deren Elemente repräsentieren. Dazu verwendet das Plugin eine inubit-interne Bibliothek, welche entsprechende Klassen anbietet. Das Package *net.joe.helpers* enthält Hilfsklassen, die verschiedenste Methoden anbieten, deren Funktionalität

fachlich betrachtet nicht in die Klassen des PCM-Metamodells gehören. Eine dieser Hilfsfunktionalitäten besteht bspw. darin, herauszufinden, ob ein identifizierter Zustandsübergang bereits in einem Prozess vorhanden ist. Das letzte Package *net.joe.pcmsplugin* stellt die eigentliche Plugin-Funktionalität bereit. Die darin enthaltenen Klassen nehmen Datenströme aus technischen Workflows entgegen und sorgen dafür, dass sie entsprechend der Anfrage verarbeitet werden. Die Klasse *PCMSConnectorWizardPanel.java* erstellt die Benutzeroberfläche, über die der Anwender später Prozessparameter festlegen kann. Auch die Grafiken, die zur Gestaltung der Benutzeroberfläche benötigt werden befinden sich in diesem Package. Entwickelt wurden sämtliche Klassen mit Java in der Version 1.6. Insgesamt besteht das Java-Plugin aus 4.549 Zeilen Quellcode.

Grundlage zur Durchführung der implementierten Ausführungssemantik ist das Auslesen der zu einem Prozess gehörenden und mit der inubit Suite modellierten Prozess- und Datenmodelle. Zuständig dafür ist die oben gezeigte Klasse *BPDImporter.java*, welche dazu das *External Interface* der inubit Suite verwendet. Mit Hilfe dieser externen Schnittstelle können alle Modelle eines Benutzers passwortgeschützt via REST-Aufruf in XML-Form abgefragt werden. Diese XML-Repräsentation der Modelle wandelt die genannte Importer-Klasse dann in Java-Repräsentationen um.

Neben dem *PCM Utility* stellt das mitgelieferte Prozesspaket mit verschiedenen technischen Workflows den zweiten Hauptbestandteil des hier dokumentierten Prototyps dar. Abbildung 27 zeigt die fünf standardmäßig im Paket enthaltenen Workflows, deren genaue Funktionsweisen im Kapitel 6.3.3 erläutert werden.

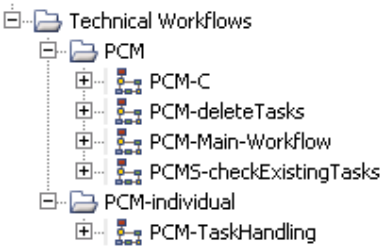


Abbildung 27: Im Prozesspaket enthaltene technische Workflows

Während der Workflow *PCM-TaskHandler* für eine Nutzung des Prozesspaketes durch den Anwender angepasst werden muss, realisieren die restlichen Workflows ihre Funktionalität ohne nochmals verändert werden zu müssen.

Zuletzt verwendet der Prototyp das über die inubit Suite mitgelieferte *Enterprise Portal*. Dabei handelt es sich um ein Liferay[49]-Portal, in dem individuell festgelegt werden kann, welche Portlets ein Benutzer verwenden kann. Dieser Prototyp bedient sich dem Portlet der *Taskliste*, in der Benutzer-Aufgaben (Tasks) erscheinen, die die Aktivitätsinstanzen repräsentieren und hier zur Bearbeitung ausgewählt werden können.

6.2 Best-Practices und Szenario

Im Laufe der in Kapitel 5 beschriebenen Ausführungssemantik wurden bereits verschiedene Restriktionen für die Modellierung in einer PCM-Umgebung genannt, jedoch nicht weiter erläutert. Dieses Kapitel soll anhand von Beispielen aufzeigen, wie bestimmte fachliche Gegebenheiten am besten modelliert werden, damit das erläuterte PCM-Metamodell und dessen Ausführungssemantik das gewünschte Ergebnis erzielen. Die Beispiele bedienen sich zur Veranschaulichung eines einfachen Angebotsprozesses, der mit der *inubit Suite 6.1*[50] modelliert wurde. Gleichzeitig dienen die hier entwickelten Modelle als Szenario für die anschließend dokumentiere prototypische Implementierung.

6.2.1 Aller Anfang ist einfach

Zu Beginn der fachlichen Modellierung eines Prozesses empfiehlt es sich, ein möglichst generisches Prozessmodell zu entwerfen, welches den einfachsten Weg durch einen Case aufzeigt und dabei mit dem später als *Initialzustand* zu definierenden Case-Zustand beginnt und in einem der möglichen *Endzustände* endet. Was Initial- und Endzustände kennzeichnet und welche Funktion sie bei der Ausführung der Cases übernehmen, wurde in Kapitel 5.2.2 erläutert. Ein solches Basis-Prozessmodell des Angebotsprozesses zeigt Abbildung 28.

[49] Vgl. **http://www.liferay.com/de**
[50] Vgl. **http://www.inubit.com/inubit-suite.html**

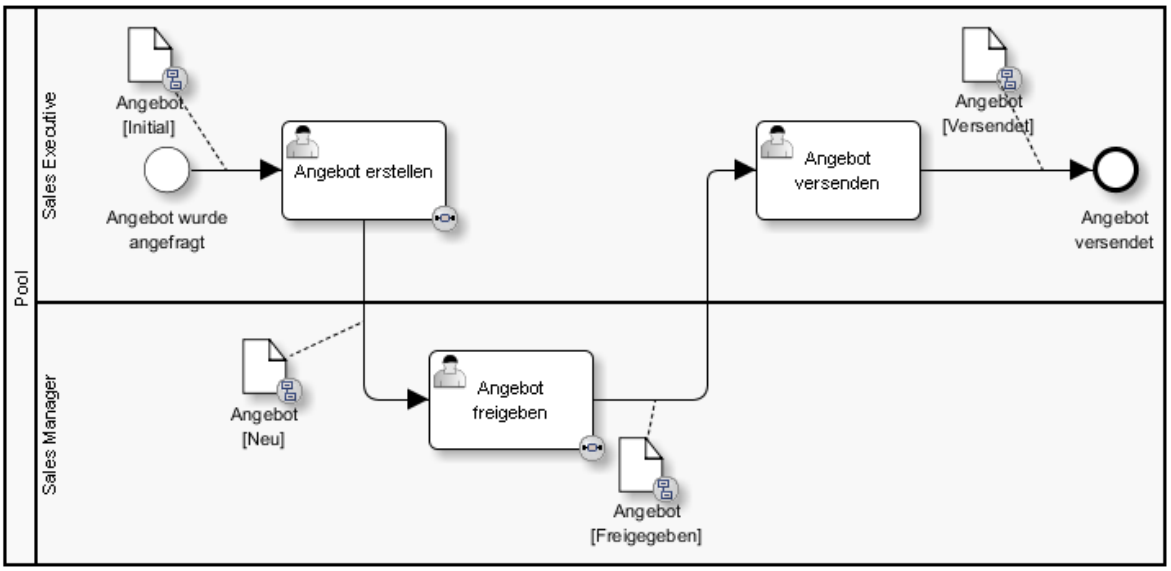


Abbildung 28: Einfachste Prozessdarstellung des Angebotsprozesses

Das Modell verwendet keine Alternativpfade und beinhaltet, zusätzlich zu Initial- und Endzustand, die gängigsten Case-Zustände. Damit ist es kaum fehleranfällig und gleichzeitig in der Lage auf die häufigsten Case-Zustände zu reagieren. Es beschreibt einen *Happy Path* und schafft damit auch Verständlichkeit für prozessfremde Personen.

6.2.2 Zusätzliche Pfade durch erweitertes Prozessmodell

Um zusätzliche Schritte in den Prozess einzubauen, die zunächst nicht im ersten Happy-Path-Modell aufgenommen wurden, empfiehlt es sich, Teile des ersten Modells zu kopieren und mögliche Zwischenschritte an passender Stelle einzufügen. In Abbildung 29 wird dazu der vordere Teil des ersten Happy-Path-Modells um die Aktivität *Technische Details einfügen*, sowie den Case-Zustand *PS-ergänzt* erweitert. Nach der Aktivität *Angebot freigeben* endet dieses zusätzliche Prozessmodell, da fortan der Prozess so weiterläuft wie im ersten Happy-Path-Modell.

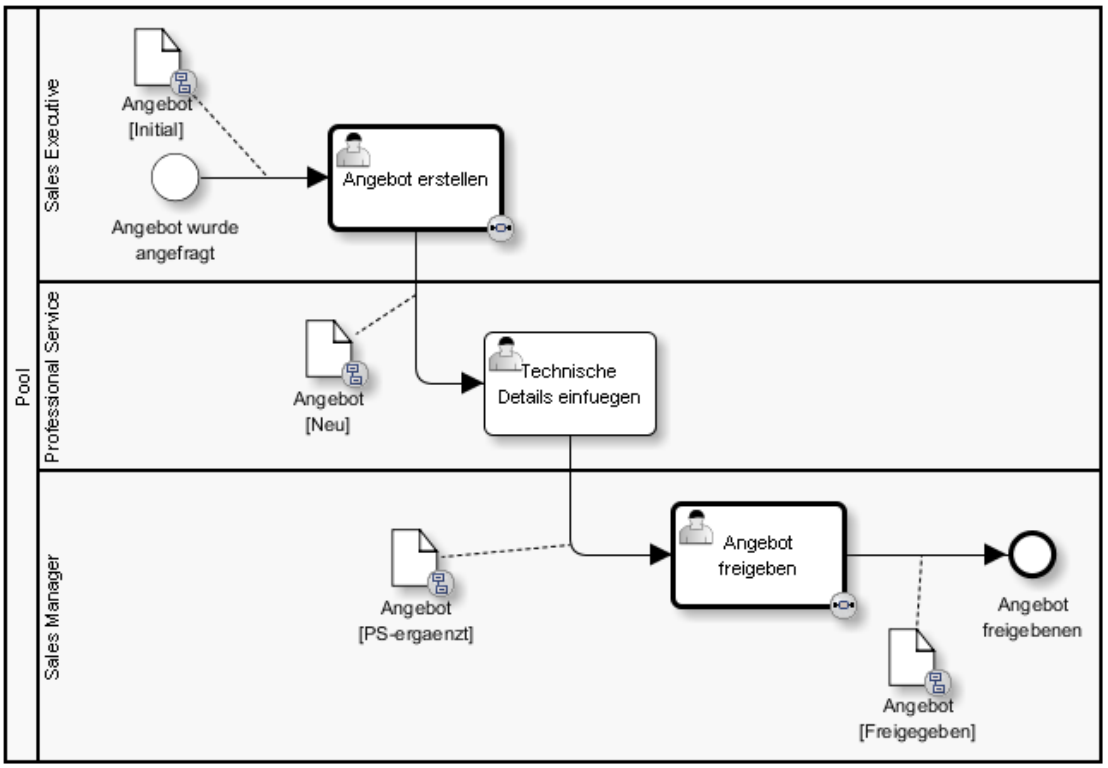


Abbildung 29: Erweitertes Prozessmodell

Die Kombination der beiden Prozessmodelle aus Abbildung 28 und Abbildung 29 ermöglicht sowohl das sofortige Freigeben des Angebots nach dessen Erstellung, als auch das vorherige Einfügen technischer Details durch die entsprechende Fachabteilung, hier *Professional Service*. Zur Ausführungszeit hat der Case Worker also die Wahl, welche der Möglichkeiten in dem konkreten Fall zum Einsatz kommen soll. Wie die Wiederverwendung von Aktivitäten anderer Prozessmodelle gehandhabt wird, beschreibt das Kapitel 6.2.4.

6.2.3 Unerreichbare Aktivitäten verhindern

In der Ausführungssemantik für ein PCMS wurde beschrieben, dass am Anfang der Prüfung danach, welche Aktivitäten dem Case Worker als nächstes angeboten werden, der aktuelle Case-Zustand darüber entscheidet, ab welcher Stelle in einem Prozessmodell die möglichen Aktivitäten geprüft werden. Es sind also nur jene Aktivitäten potentiell durchführbar, die sich *hinter* dem aktuellen Case-Zustand befinden. Für den in obiger Abbildung 28 gezeigten Angebotsprozess bedeutet dies konkret, dass sämtliche Aktivitäten potentiell durchführbar sind, wenn sich der Case im Zustand *Initial* befindet, da sich alle modellierten Aktivitä-

ten zwischen dem Sequenzfluss mit besagtem Case-Zustand und dem Endereignis befinden. Hat der Case allerdings den Zustand *Neu*, kann nur noch die Aktivität *Angebot freigeben* durchgeführt werden. Die Aktivität *Angebot erstellen* befindet sich im Prozessfluss *vor* dem Case-Zustand *Neu* und kommt daher für eine Durchführung nicht in Frage. Aus diesem Prinzip resultiert, dass Aktivitäten, denen kein einziger Case-Zustand vorausgeht, niemals durchführbar sind.

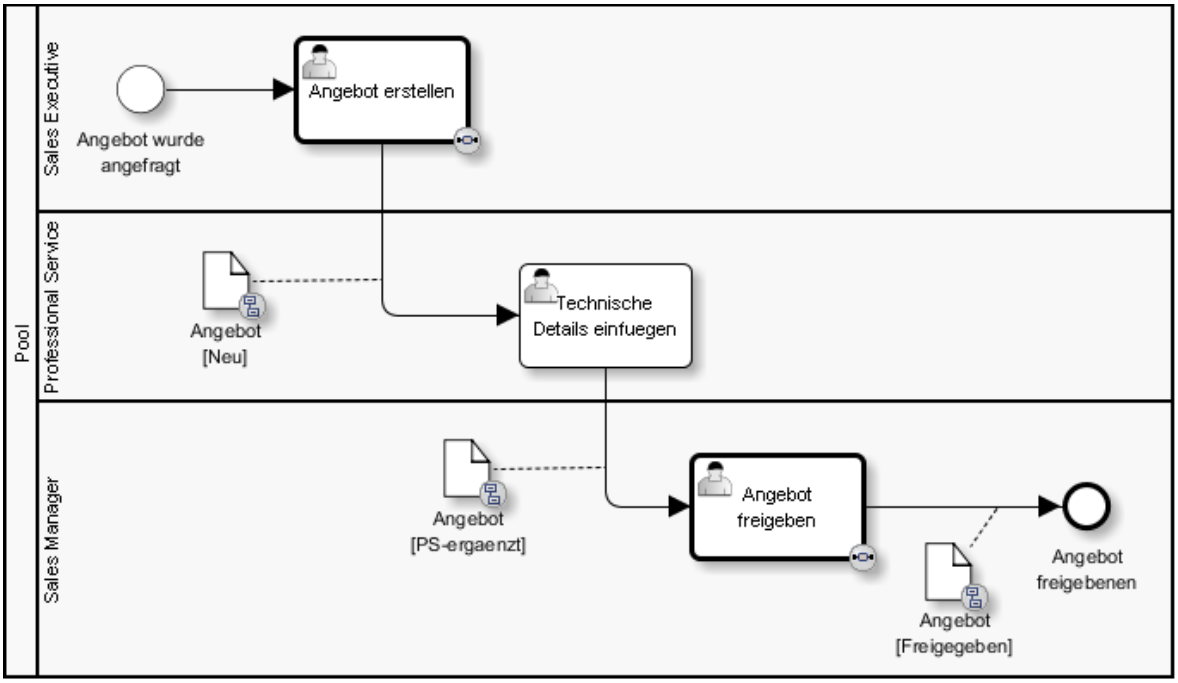


Abbildung 30: Unerreichbare Aktivität *Angebot erstellen*

Abbildung 30 zeigt solch ein Negativ-Beispiel. Unabhängig davon, welchen Zustand ein Case in seinem Verlauf einnimmt, wäre die Aktivität *Angebot erstellen* in diesem Modell nicht durchführbar. Es handelt sich um eine *unerreichbare Aktivität*.

Um unerreichbare Aktivitäten zu verhindern, muss demnach sichergestellt werden, dass jeder Aktivität mindestens ein Datenartefakt mit definiertem Case-Zustand vorausgeht. Am einfachsten wird dies realisiert, indem die ausgehenden Sequenzflüsse von Startereignissen mit solchen Datenartefakten assoziiert werden. Da Prozessmodelle oft nur über ein einziges Startereignis verfügen, lässt sich diese Maßnahme zur Verhinderung nicht erreichbarer Aktivitäten leicht umsetzen.

6.2.4 Aktivitäten wiederverwenden

Durch die Verwendung mehrerer Prozessmodelle kann es vorkommen, und wird auch ausdrücklich unterstützt, dass eine Aktivität an mehr als nur einer Stelle modelliert wird. Um dem PCMS deutlich zu machen, dass es sich dennoch um ein und dieselbe Aktivität handelt, eignet sich die Verwendung des BPMN-Elements *Aufruf-Aktivität* (engl. *Call Activity*[51]). Dazu wird die Aktivität bei ihrem ersten Auftreten als *globale Aufgabe* (engl. *Global Task*[52]) definiert und damit wiederverwendbar gemacht. Taucht die Aktivität an anderer Stelle, im gleichen oder einem anderen Modell, erneut auf, wird sie als Aufruf-Aktivität modelliert und mit der zuvor als globale Aufgabe definierten Aktivität verknüpft. Das Prozessmodell in Abbildung 29 und Abbildung 30 (s. Kapitel 6.2.2 und Kapitel 6.2.3) bediente sich bereits solcher Aufruf-Aktivitäten, welche anhand des dickeren Rahmens zu erkennen sind.

Durch die Verwendung von Aufruf-Aktivitäten für wiederkehrende Aufgaben wird sichergestellt, dass das PCMS immer nur eine Instanz für diese Aufgabe erzeugt. Außerdem werden Eigenschaften, wie z.B. definierte Vor- und Nachbedingungen in Aufruf-Aktivitäten übernommen und müssen nicht erneut erstellt werden.

6.2.5 Alternativpfade in einem Modell

Nicht jeder Alternativpfad muss durch ein eigenes Prozessmodell dargestellt werden. Diese Best Practice veranschaulicht, wie sich in einer PCM Umgebung ein auf Daten basierender, alternativer Prozessverlauf realisieren lässt. Dazu unterstützt das vorgestellte PCM-Metamodell das in Kapitel 2.4.1 vorgestellte *exklusive Gateway*. Dieses wird an der Stelle im Prozessmodell platziert, an welcher der Prozessfluss geteilt werden soll und mit dem Namen des Datensatzes, anhand dessen der Fortlauf des Prozessflusses entschieden wird, beschriftet. Diese Beschriftung ist für die Ausführung zwar nicht von entscheidender Bedeutung, erhöht aber die Nachvollziehbarkeit des entwickelten Modells. Um sicher zu stellen, dass auch wirklich eine Entscheidung anhand des zu untersuchenden Datensatzes getroffen werden kann, sollte die letzte Aktivität vor dem Gateway mit einer Nachbedingung versehen sein, welche das Setzen eines Wertes für diesen Datensatz zur erzwingt. Die Teilung des Prozessflusses geschieht anschließend durch die Mo-

[51] Vgl. **OMG 2011**, S. 183
[52] Vgl. **OMG 2011**, S. 187

dellierung mehrerer ausgehender Sequenzflüsse aus dem Gateway, welche zu weiteren Aktivitäten führen. Diese Sequenzflüsse werden dann mit Bedingungen versehen, die sich auf den untersuchten Datensatz beziehen und dabei unterschiedliche Werte des Datensatzes behandeln. Abbildung 31 zeigt die Teilung des Angebotsprozesses anhand der Frage, wer ein Angebot freigeben darf.

.

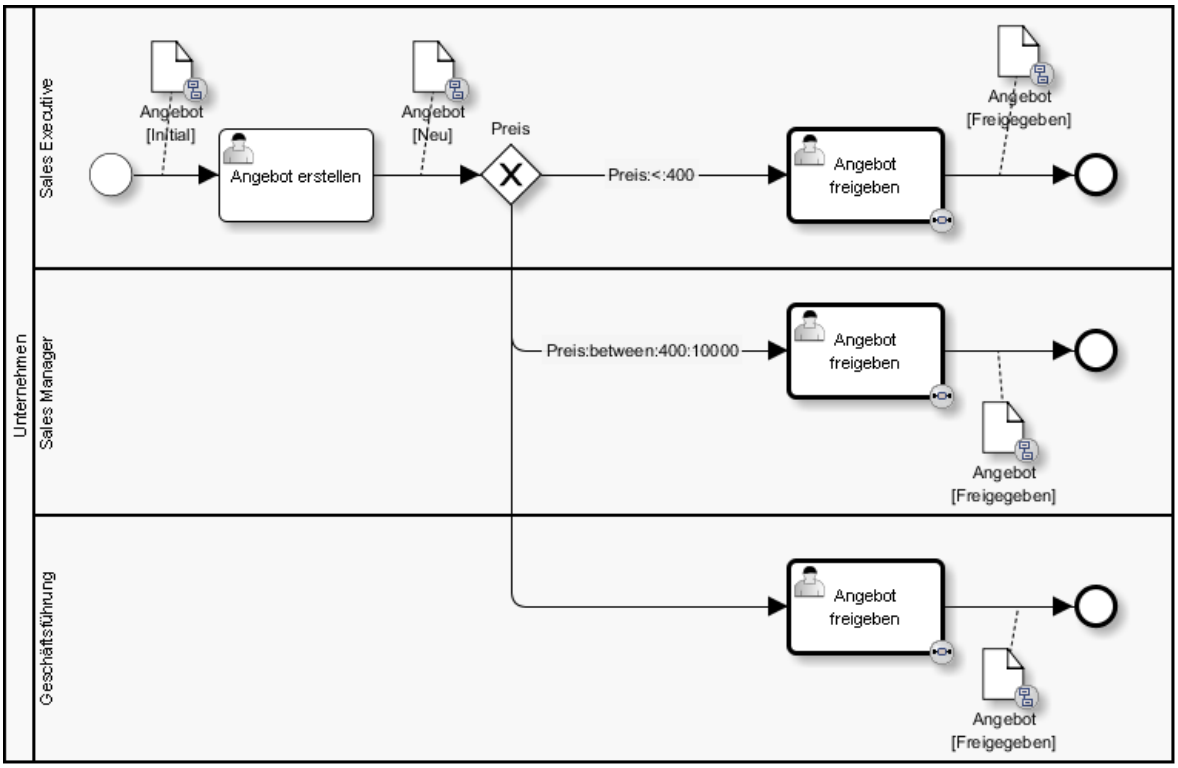


Abbildung 31: Prozessmodell Angebot freigeben

Der *Preis* ist ein Attribut des zentralen Datenobjektes *Angebot* und wird an dem Gateway abgefragt. Hat das Attribut nach der Aktivität *Angebot erstellen* einen Wert kleiner als 400, folgt das Modell anschließend dem oben verlaufenden Pfad und wird die Aktivität *Angebot freigeben* dem Sales Executive anbieten. Befindet sich der Wert zwischen 400 und 10.000, erhält der Sales Manager die Aufgabe das Angebot freizugeben. Der nach unten verlaufende Pfad stellt ein weiteres wichtiges Konstrukt für die Teilung eines Prozessflusses dar. Dadurch, dass dieser Sequenzfluss keine Bedingung trägt, fungiert er sozusagen als *Sonstiges*-Pfad und wird dann beschritten, wenn keine der anderen Bedingungen zutrifft. Auf diese Weise wird ein Stillstand des Cases verhindert, falls mal keine der modellierten Bedingungen zutrifft. Es darf jedoch nur ein einziger ausgehender Sequenzfluss ohne Bedingung modelliert werden. Andernfalls könnten nach dem Gateway meh-

rere Pfade beschritten werden, was der Semantik eines exklusiven Gateways widerspricht. Aus dem gleichen Grund ist darauf zu achten, dass wenn mehrere ausgehende Sequenzflüsse aus einem Gateway mit Bedingungen belegt sind, sich diese widersprechen, d.h. tatsächlich andere Werte behandeln. In Abbildung 32 wurde die Bedingung für den Sales Manager auf *kleiner 10.000* geändert.

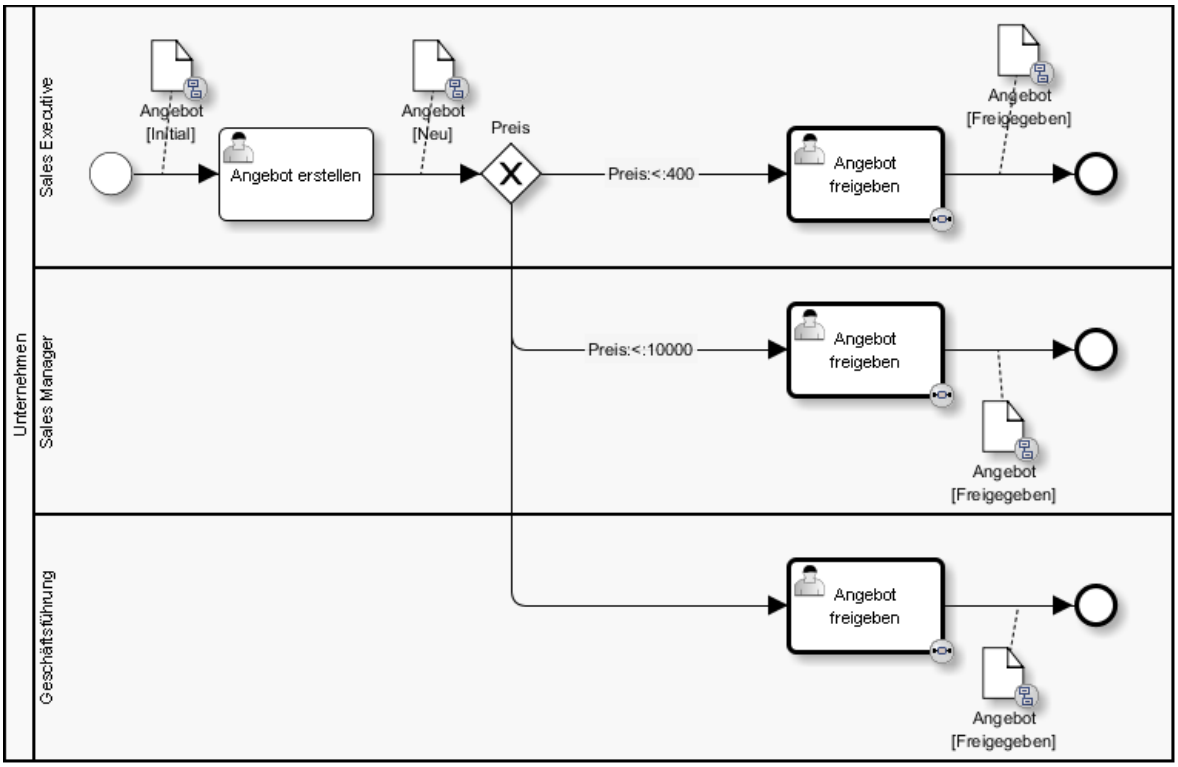


Abbildung 32: Bedingungen führen zu parallelem Ablauf

Würde das Attribut *Preis* in diesem Fall beispielsweise den Wert *350* annehmen, so wären sowohl die Bedingungen für den obersten (Sales Executive) als auch den mittleren (Sales Manager) Prozessfluss erfüllt und es würde zu einem parallelen Ausführungspfad kommen.

6.2.6 Aktivitäten jederzeit durchführbar machen

Immer wieder kann es vorkommen, dass ein Prozess die Anforderung aufstellt, bestimmte Aktivitäten in mehr als nur einer Situation, d.h. in mehr als nur einem Case-Zustand, zur Durchführung anzubieten. Tatsächlich wurde dies sogar bereits durch die bisher gezeigten Prozessmodelle des Angebotsprozesses realisiert. Die Aktivität *Angebot freigeben* ist schon jetzt immer dann möglich, wenn sich ein Ca-

se entweder im Zustand *Neu* oder *PS-ergänzt* befindet. Dies entstand dadurch, dass die Aktivität in zwei unterschiedlichen Prozessmodellen vorkam. Wie eine Aktivität auch durch einmaliges Auftreten in einem einzigen Prozessmodell trotzdem in mehreren Case-Zuständen ausführbar ist, zeigt Abbildung 33.

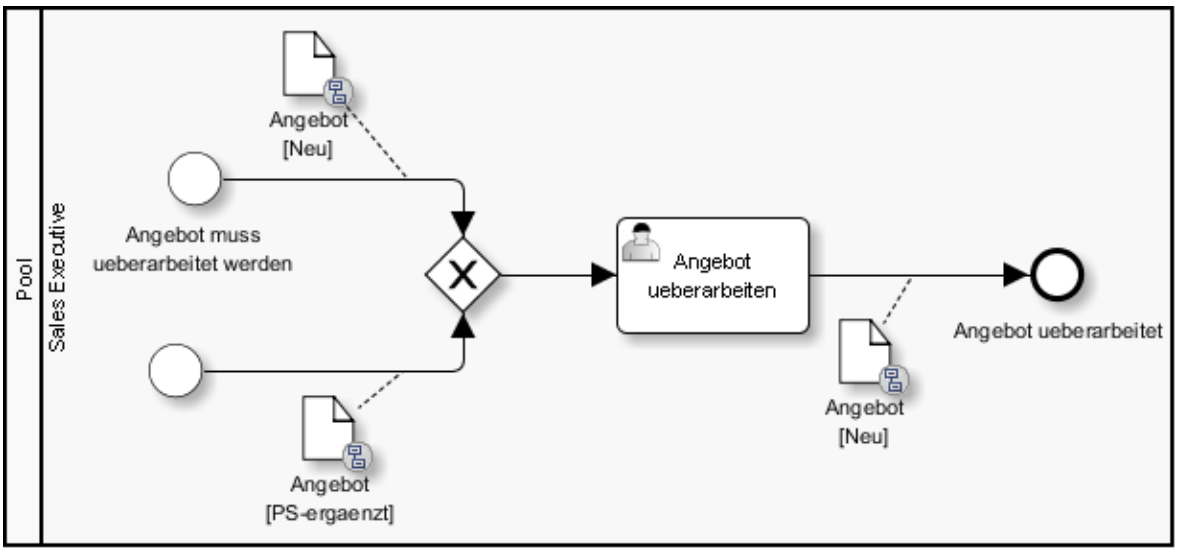


Abbildung 33: Mehrere Case-Zustände für eine Aktivität

Die Aktivität *Angebot überarbeiten* soll möglich sein, sobald das Angebot angelegt wurde, sich der Case also mindestens im Zustand *Neu* befindet, und solange es noch nicht freigegeben wurde. Folglich muss die Aktivität durchführbar sein, wenn sich der Case entweder im Zustand *Neu* oder *PS-ergänzt* befindet. Das oben gezeigte Prozessmodell realisiert dies durch zwei Startereignisse, deren ausgehende Sequenzflüsse mit den entsprechenden Case-Zuständen assoziiert sind. Dadurch wird die Aktivität *Angebot überarbeiten* zum Nachfolger beider Zustände und deshalb gemäß der Ausführungssemantik auch in beiden Fällen durchführbar.

Lautet eine Anforderung gar, dass eine bestimmte Aktivität *immer* durchführbar sein soll, lässt sich dies entweder wie zuvor beschrieben über eine Vielzahl zusammengeführter Sequenzflüsse (einen pro möglichem Case-Zustand) realisieren, oder es wird für diese spezielle Anforderung eine spezifische Lösung implementiert. Der Angebotsprozess soll, solange er keinen finalen Case-Zustand erreicht hat, jederzeit abgebrochen werden können. Das in Abbildung 34 gezeigte Prozessmodell realisiert diese Anforderung, indem es der Aktivität *Case abbrechen* den Case-Zustand * (Sternchen) als Einstiegspunkt vorsetzt. In diesem Fall fungiert * als Platzhalter für sämtliche Case-Zustände, die der Case einnehmen könnte.

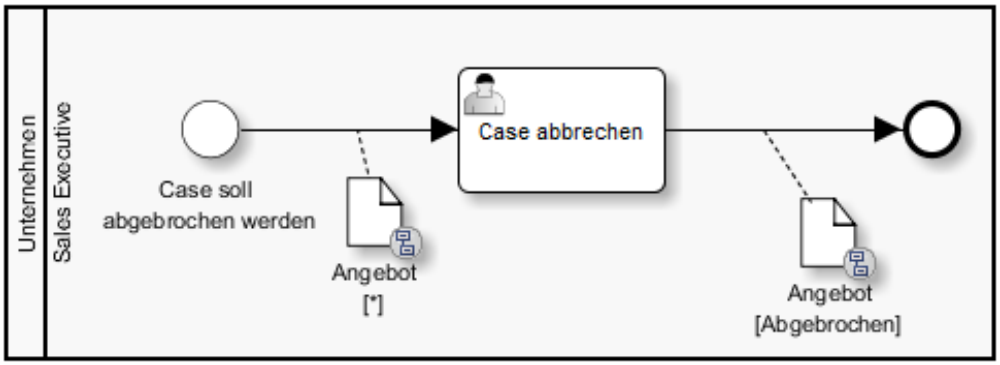


Abbildung 34: Aktivitäten jederzeit durchführbar machen

Es gilt zu beachten, dass * als Case-Zustand ausschließlich *vor* Aktivitäten, und damit als Einstiegspunkt in ein Prozessmodell, verwendet werden darf. Die Verwendung dieses Zustandes *nach* einer Aktivität hätte eine unklare Aussage darüber zur Folge, in welchen konkreten Zustand der Case nach dieser Aktivität übergehen soll.

6.2.7 Überspringbare Aktivitäten modellieren

Die vorgestellte PCM-Ausführungssemantik lässt es zu, dass der Case Worker darüber entscheiden kann, Aktivitätsinstanzen unter bestimmten Voraussetzungen auszulassen, d.h. zu überspringen. Es wurde erläutert, dass eine Aktivität immer dann ausgelassen werden kann, wenn mindestens ein Nachfolger dieser Aktivität ebenfalls zur Durchführung bereit ist. Auch dieser Umstand wird bereits durch die gezeigten Modelle des Angebotsprozesses ermöglicht. Nach Durchführung der Aktivität *Angebot erstellen* befindet sich ein Case im Zustand *Neu*. Aus dem Prozessmodell in Abbildung 28 ergibt sich daher, dass nun die Aktivität *Angebot freigeben* bereit ist. Aus dem Prozessmodell in Abbildung 29 wiederum geht hervor, dass die Aktivität *Technische Details einfügen*, welche in diesem Modell ein Vorgänger von *Angebot freigeben* ist, ebenfalls zur Durchführung bereitsteht. Damit erfüllt sich für diese Aktivität die o.g. Bedingung, um ausgelassen werden zu können.

6.2.8 Schleifen über Prozessfluss modellieren

Es gilt als selbstverständliche Anforderung, dass sich ein Geschäftsprozess auch mal an gegebener Stelle wiederholt und damit eine Schleife durchlaufen kann. Diese Best-Practice zeigt die Realisierung einer Schleife im Angebotsprozess mit Hilfe von Prozessmodellierung. Dazu wird das bereits erweiterte Modell aus Abbildung 29 um zwei Gateways ergänzt und die Schleife durch den entsprechenden Einsatz von Sequenzflüssen und Bedingungen vervollständigt. Abbildung 35 zeigt, wie der Case nach der Aktivität *Angebot freigeben* entweder in den Zustand *freigegeben* übergehen kann oder eine Schleife dreht und zurückspringt zum Zustand *Neu*.

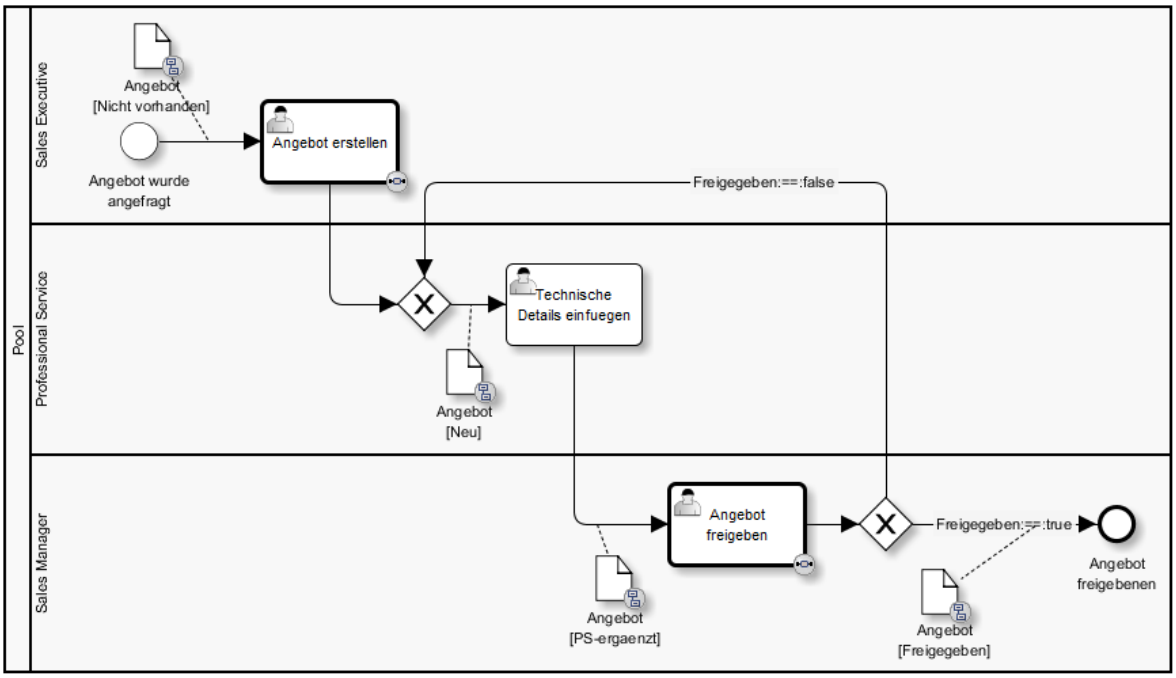


Abbildung 35: Angebotsprozess mit modellierter Schleife

Zur Realisierung der Schleife wird nach der Aktivität *Angebot freigeben* am dann folgenden Gateway der Wert des Datensatzes *Freigegeben* abgefragt und in Abhängigkeit dessen entweder eine Schleife gedreht oder der Case in den Zustand *Freigegeben* versetzt. Zu beachten sind die Bedingungen an den ausgehenden Sequenzflüssen des Gateways. Würde einer der Sequenzflüsse, oder gar beide, über keine Bedingung verfügen, dann würde es sich hier um sich widersprechende Zustandsübergänge handeln. Da der Datensatz *Freigegeben* von booleschem Datentyp ist, ist es auch nicht erforderlich, das Setzen dieses Datensatzes per Nachbedingung an der Aktivität *Angebot freigeben* zu erzwingen. Aufgrund seines Datentyps kommen für diesen Datensatz nur die Werte *wahr* oder *falsch* in Frage.

Ist kein Wert gesetzt, wird dies als *falsch* interpretiert. Ein *Sonstiges*-Pfad ist hier also ebenfalls nicht erforderlich.

6.2.9 Schleifen über zusätzliche Zustandsübergänge modellieren

Während die vorhergehende Best-Practice die Realisierung einer Schleife mit Hilfe von Prozessmodellierung aufgezeigt hat, soll die im Folgenden vorgestellte Vorgehensweise skizzieren, wie sich die gleiche fachliche Schleife auch mit Hilfe von zusätzlichen Zustandsübergängen realisieren lässt. Dazu wird für den Prozess der in Tabelle 7 gezeigte zusätzliche Zustandsübergang definiert.

Alter Zustand	PS-ergaenzt
Auslöser	Angebot freigeben
Neuer Zustand	Neu
Bedingung	Freigegeben == falsch

Tabelle 7: Zusätzlicher Zustandsübergang für den Angebotsprozess

Dieser gezeigte Zustandsübergang übernimmt die gleiche Funktion wie der im Best-Practice zuvor modellierte Schleifen-Prozessfluss. Allerdings muss auch für diese Realisierung das Prozessmodell geringfügig angepasst werden.

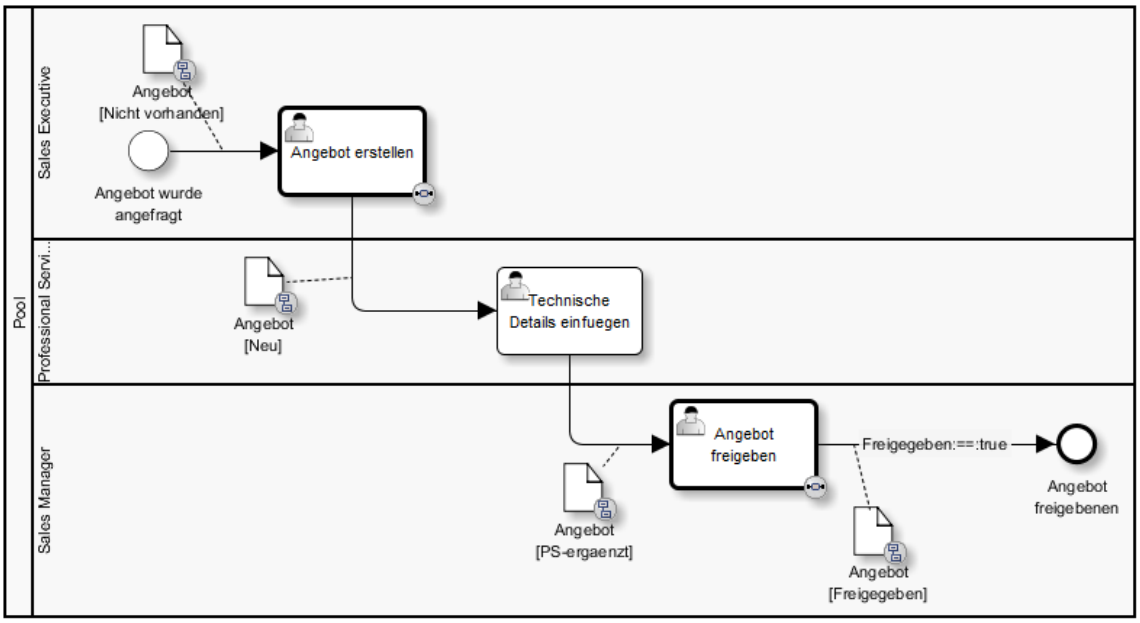


Abbildung 36: Erweitertes Prozessmodell mit Bedingung am Sequenzfluss

Damit der in Tabelle 7 gezeigte zusätzliche Zustandsübergang keinen Widerspruch zu den modellierten Zustandsübergängen verursacht, muss dort der ausgehende Sequenzfluss der Aktivität *Angebot freigeben* mit der Bedingung *Freigeben == wahr* versehen werden, wie in Abbildung 36 gezeigt.

6.2.10 Skript-Tasks verwenden

Abbildung 37 zeigt die Erweiterung des Ausgangsmodells um einen Skript-Task. Durch die Vergabe des Stereotyps *Skript* wird festgelegt, dass es sich hierbei um eine Aufgabe handelt, die automatisch durchgeführt wird, sobald sie durch die Zustandsprüfung von Aktivitätsinstanzen in den Zustand *bereit* versetzt wird.

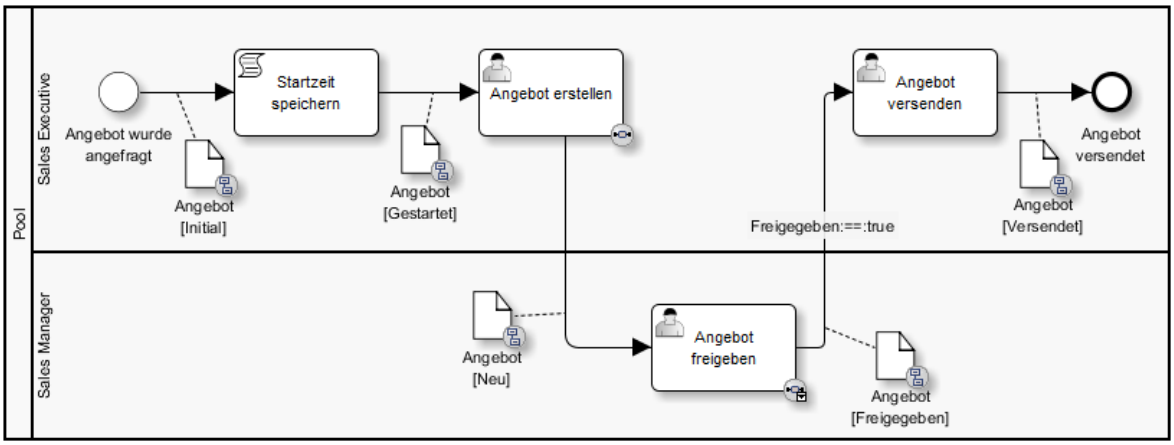


Abbildung 37: Prozessmodell mit Skript-Task

Um die automatische Durchführung einer Aufgabe zu ermöglichen, muss an der modellierten Aktivität hinterlegt werden, wo sich die implementierte Logik befindet und wie diese angestoßen wird. Wie genau die Anbindung solch einer Logik im Einzelnen funktioniert, ist abhängig vom konkret entwickelten System. Das hier gezeigte Beispiel des Angebotsprozesses wurde mit der inubit Suite erstellt, weswegen der modellierte Skript-Task auf einen technischen Workflow verlinkt, welcher zu entsprechendem Zeitpunkt angestoßen wird und dabei den Datensatz *Startzeit* des zentralen Datenobjektes auf die aktuelle Server-Zeit setzt.

6.3 Funktionalität

Nach den bisherigen Abschnitten dieses Kapitels mit eher vorbereitendem Charakter, geht es im Folgenden um die tatsächlichen Funktionalitäten des Prototyps. Diese werden in die drei Kategorien *Prozessdefinition*, *Validierung* und *Ausführung* unterteilt und in jeweils einem dafür vorgesehenen Unterkapitel erläutert.

6.3.1 Prozessdefinition

Die Prozessdefinition stellt die Phase dar, in der ein Prozess zu seiner Ausführung vorbereitet wird. Für die Realisierung einer Unterstützung durch PCM sind mehrere Tätigkeiten nötig. Zunächst erstellt der Anwender, oder ein Team von Anwendern, die Prozessmodelle, auf denen der Prozess und damit später die konkreten Cases beruhen sollen. Dazu gehört auch die Einbindung von Datenobjekten in die

Prozessmodelle, welche zum Einen die unterschiedlichen Case-Zustände definieren und zum Anderen auf das zentrale Datenobjekt dieses Prozesses verweisen, welches als *Business Object Diagram* erstellt wird. Diese Tätigkeiten führt der Anwender im Rahmen dieses Prototyps mit den *Bordmitteln* der inubit Suite durch. Für den weiteren Verlauf der Prototypdokumentation werden die Modelle des bereits aus den Best-Practices bekannten Angebotsprozesses genutzt. Diese befinden sich in vollständiger Ausführung im Anhang B. Nachdem die Modelle erstellt und in der Process Engine der inubit Suite veröffentlicht wurden, kommt zum ersten Mal das implementierte Plugin *PCM Utility* zum Einsatz. In dem mitgelieferten Workflow *PCM-Utility-C* befindet sich, wie in Abbildung 38 gezeigt, als einziges Modul eine Instanz des PCM Utilitys. Die Separierung dieses Moduls in einen eigenen Workflow ermöglicht später dessen Aufruf aus verschiedenen Stellen des *Hauptworkflows* oder auch aus individuellen Workflows.

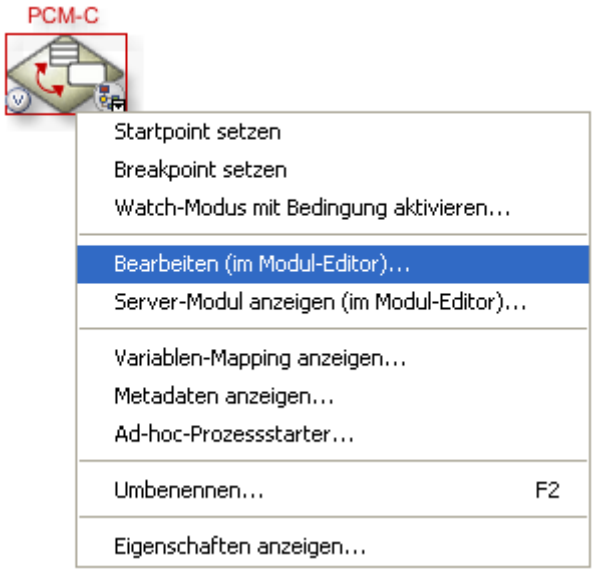


Abbildung 38: PCM Utility Modul mit Kontextmenü

Ein Rechtsklick auf das Modul und die Auswahl von *Bearbeiten (im Modul-Editor)…* aus dem Kontextmenü öffnet das Modul zur Bearbeitung. Nach der Oberfläche für allgemeine Moduleigenschaften erscheint schließlich die zu diesem Zeitpunkt noch leere Eingabemaske für einen PCM-Prozess. Zunächst wird hier ein Name für den umzusetzenden Prozess vergeben. In diesem Fall *Angebotsprozess*. Als Nächstes bietet der Dialog an, die Prozessmodelle, auf denen der Prozess beruhen soll, per Häkchen auszuwählen. Zur Auswahl stehen sämtliche Prozessmodelle, die sich in Verzeichnissen des aktuellen Benutzers bzw. seinen Obergruppen befinden. Die Festlegung des zum Prozess gehörenden Datenmo-

dells ist nicht erforderlich, da dies über Verlinkungen in den Prozessmodellen aufgelöst wird. Für den Angebotsprozess werden die folgenden Prozessmodelle ausgewählt:

- PCM-BPD-001-Einfaches Angebot erstellen
- PCM-BPD-102-Komplexes Angebot erstellen
- PCM-BPD-201-Angebot ueberarbeiten
- PCM-BPD-206-Case abbrechen

In der nächsten Sektion des Dialogfeldes lassen sich der Initialzustand von Cases, sowie die zusätzlichen Zustandsübergänge festlegen. Da die zugrunde liegenden Modelle gerade erst ausgewählt wurden, ist ein Klick auf den Button *Refresh* nötig, um die Liste der möglichen Initialzustände zu erhalten. Ist dies erfolgt, wird für diesen Prozess der Case-Zustand *Initial* als solcher gewählt. Die letzte Einstellungsoption bietet die Möglichkeit zur Erstellung zusätzlicher Zustandsübergänge. Über den *Add*-Button lassen sich diese einzeln hinzufügen und dann anschließend gemäß des PCM-Modells *Ausgangszustand*, *Zielzustand*, *Auslöser* und *Bedingung* festlegen. Für den Fall, dass ein zusätzlicher Zustandsübergang nicht länger benötigt wird, lässt er sich über den *Delete*-Button wieder entfernen. Für den Angebotsprozess wird der in Abbildung 39 gezeigte, zusätzliche Zustandsübergang definiert. Dieser ermöglicht eine Schleife nach der Aktivität *Angebot freigeben* hin zum Case-Zustand *Neu*, wenn der Datensatz *Freigegeben* den Wert *falsch* einnimmt. Abbildung 39 zeigt den gesamten Dialog nach dem Vornehmen sämtlicher, beschriebener Einstellungen.

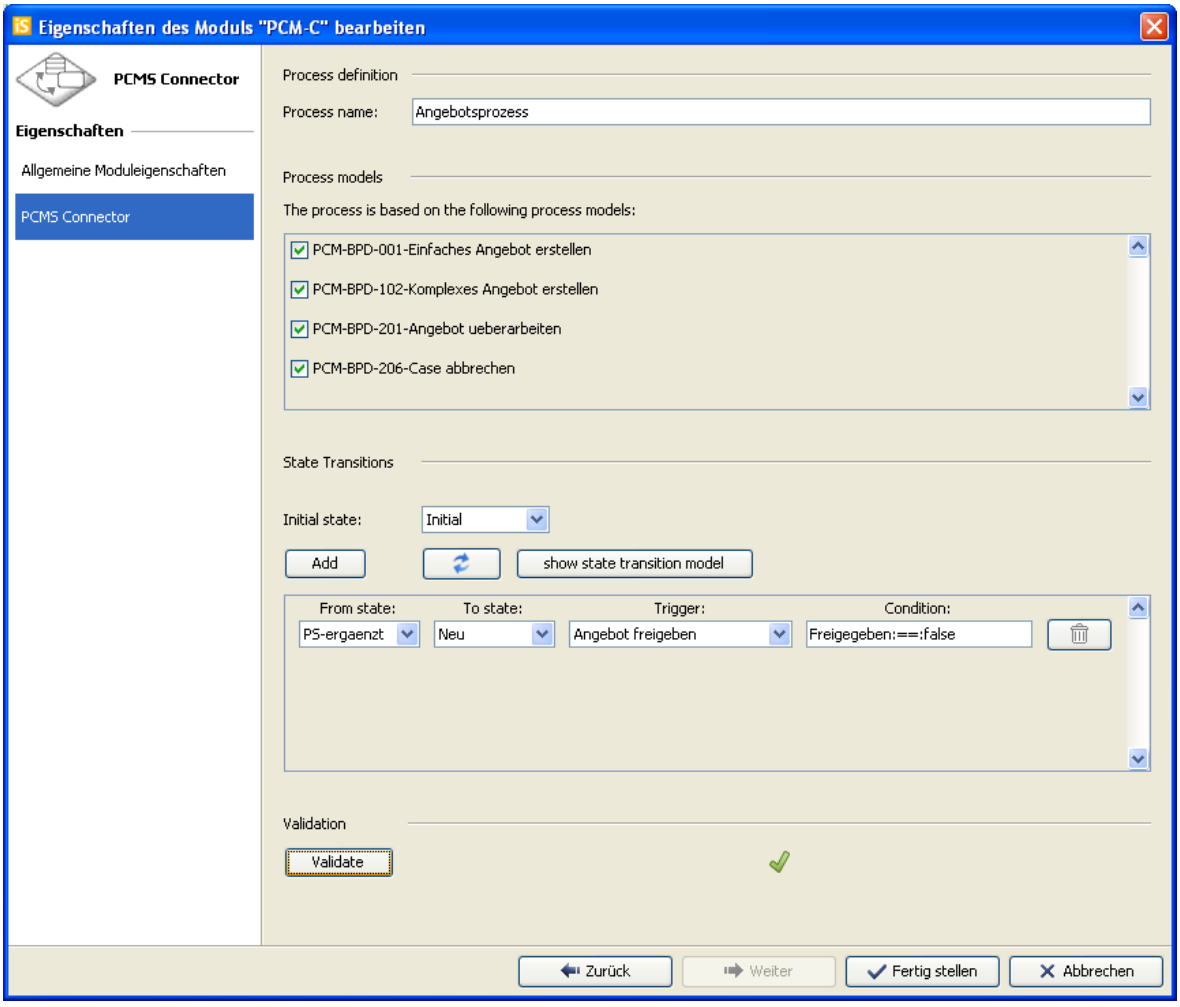


Abbildung 39: Ausgefüllter Dialog des PCM Utilitys

Der rechts neben dem *Refresh*-Button angeordnete Button *show state transition model* öffnet ein Informationsfenster mit dem dynamisch generierten Zustandsübergangsdiagramm, wie es sich aus den gewählten Prozessmodellen und den zusätzlichen Zustandsübergängen definiert. In dem Modell werden alle Zustandsübergänge sichtbar, wobei durchgezogene Linien die prozessbasierten Übergänge repräsentieren und gestrichelte Linien für zusätzliche definierte Zustandsübergänge stehen. Des Weiteren, wie in Abbildung 40 zu sehen, werden finale Zustände durch dickere Rahmen dargestellt.

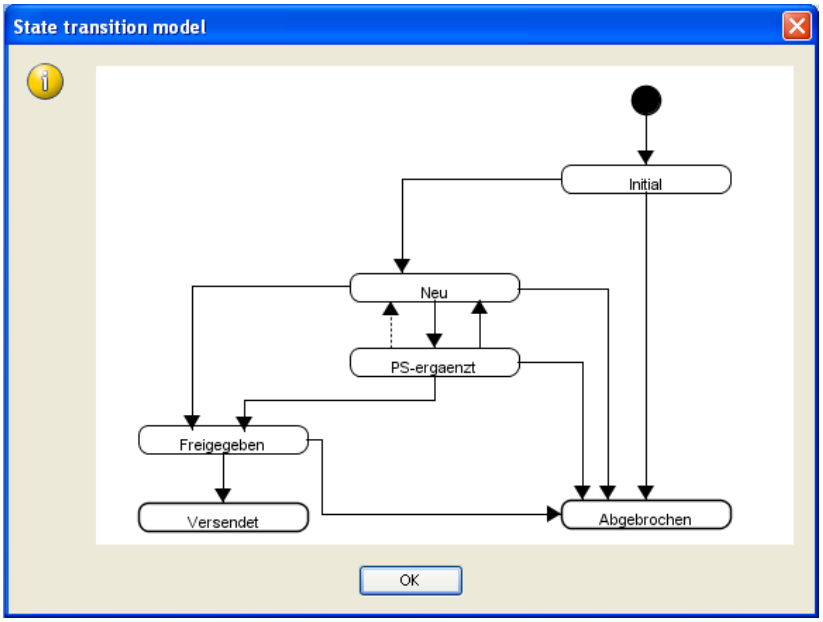


Abbildung 40: Dynamisch erstelltes Zustandsübergangsdiagramm

Die Funktionalität des zuletzt angebotenen *Validate*-Buttons erläutert das nun folgende Kapitel.

6.3.2 Validierung

Bei der Realisierung von PCM kommen viele Informationen aus unterschiedlichen Quellen zusammen. Diese Quellen sind die verschiedenen Prozessmodelle, das verlinkte Datenmodell, sowie die zusätzlich im Modul-Editor definierten Zustandsübergänge. Damit all diese Informationen sinnvoll verarbeitet werden können, muss untersucht werden, ob sämtliche Angaben zum Prozess auch Sinn ergeben. Dazu wurden im Kapitel 5.3 bereits verschiedene Sachverhalte beleuchtet, die eine konsistente Ausführung gefährden könnten und daher eine Validierung notwendig machen.

Der hier vorgestellte Prototyp validiert verschiedene dieser Gegebenheiten und gibt ggf. entsprechende Fehlermeldungen zurück. Im Folgenden werden die einzelnen, untersuchten Gegebenheiten erläutert.

Als erstes untersucht die Validierungsfunktion, welche über den in Abbildung 39 gezeigten *Validate*-Button ausgelöst wird, die verwendeten Datenobjekte. Eine Prämisse dieses Prototyps lautet, dass alle in Prozessmodellen verwendeten Datenobjekte auf dasselbe Geschäftsdatenobjekt aus demselben Geschäftsdaten-

modell verlinken müssen. Beim Einlesen der Prozessmodelle und deren Datenobjekten speichert der Prototyp die ID des verlinkten Geschäftsdatenobjektes und den Namen des Geschäftsdatenmodells als Eigenschaft eines jeden Datenobjektes. Zum Zeitpunkt der Validierung werden dann für jedes Datenobjekt diese ID bzw. dieser Modellname abgefragt. Wird dabei kein Modellname gefunden, erhält der Anwender eine Meldung darüber, dass ein Datenobjekt zu keinem Datenmodell verlinkt wurde. Wie in Abbildung 41 zu sehen, wird das Prozessmodell, in dem das betroffene Datenobjekt gefunden wurde, bei der Fehlermeldung mit angezeigt und rot markiert.

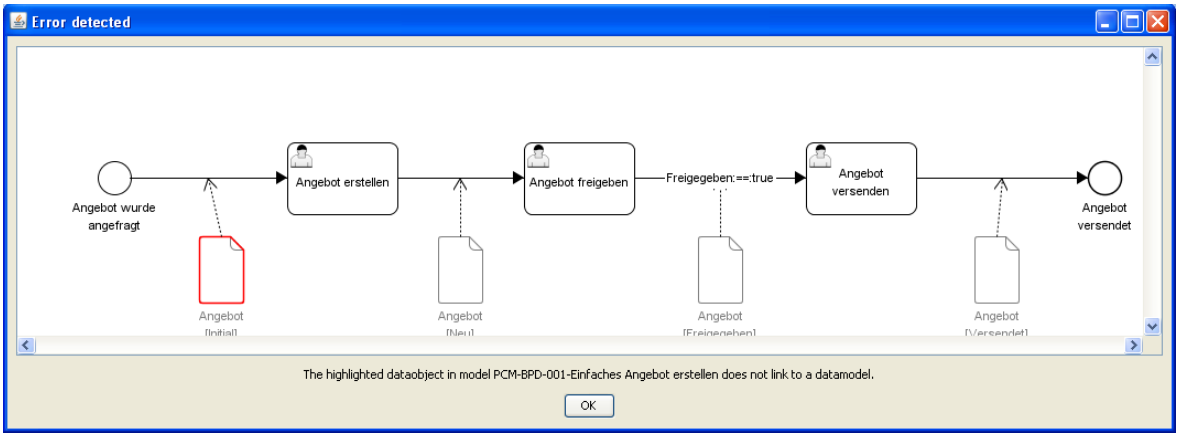


Abbildung 41: Nicht verlinktes Datenobjekt

Eine ähnliche Fehlermeldung erhält der Anwender auch, wenn zwar zu einem Datenmodell, nicht aber auf eine konkrete Klasse, also ein Geschäftsdatenobjekt, verlinkt wird. Ebenso wird eine Fehlernachricht ausgegeben, wenn zwar jedes Datenobjekt in jedem Prozessmodell auf ein Geschäftsdatenobjekt eines Geschäftsdatenmodells verlinkt, sich die ID des untersuchten Geschäftsdatenobjektes allerdings von der des letzten Geschäftsdatenobjektes unterscheidet. In diesem Fall verweisen nicht alle Datenobjekte aus den Prozessmodellen auf das gleiche Geschäftsdatenobjekt.

Als nächstes beschäftigt sich die Validierung mit den Aktivitäten der zu dem Prozess gehörenden Prozessmodelle. Es soll sichergestellt werden, dass sich keine modellierte Aktivität noch *vor* dem ersten Datenobjekt in einem Prozessmodell befindet. Wie in Kapitel 5.2.2 beschrieben, würde es sich dabei um eine nicht erreichbare Aktivität handeln, da sie, unabhängig davon welchen Zustand ein Case einnimmt, nie als potentiell ausführbar in Betracht gezogen werden würde. Die Validierung prüft also für jede modellierte Aktivität, ob sich zwischen ihr und dem Startereignis, bzw. einem der Startereignisse, des Prozessmodells ein Datenob-

jekt befindet. Ist dies nicht der Fall, erhält der Benutzer die in Abbildung 42 gezeigte Fehlermeldung. Die betroffene Aktivität ist rot markiert.

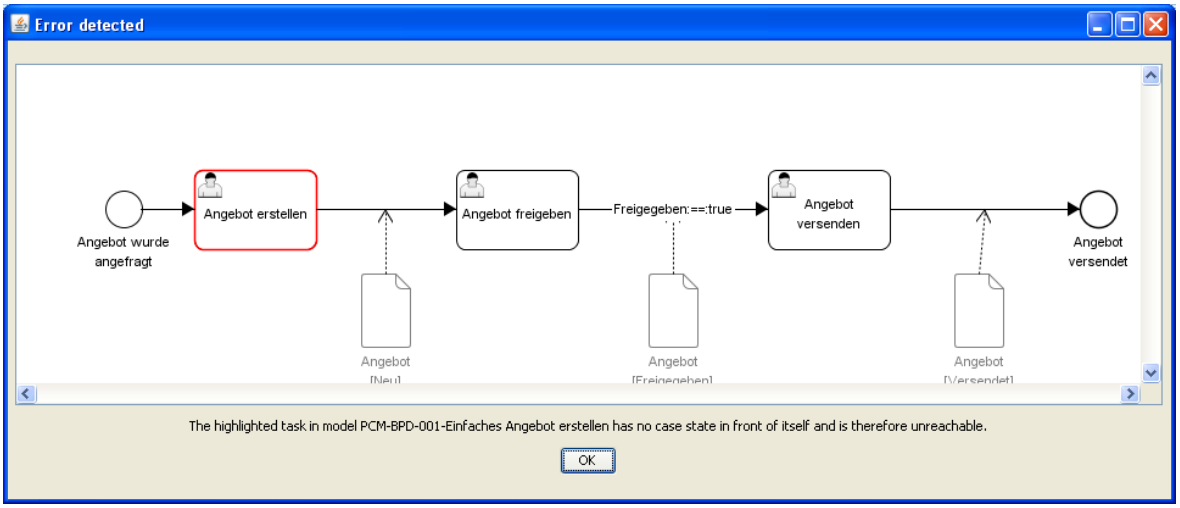


Abbildung 42: Fehlermeldung für eine unerreichbare Aktivität

Des Weiteren soll die Validierung Fehler im Kontext mit Case-Zuständen verhindern. So wird geprüft, ob Datenobjekte, welche * als Case-Zustand tragen, ausschließlich *vor* Aktivitäten in einem Prozessmodell vorkommen. Dazu ermittelt die Validierung sämtliche Knoten zwischen solch einem Datenobjekt und dem Startereignis des Modells. Befindet sich darunter eine Aktivität, wird der Anwender mit einer entsprechenden Fehlermeldung versorgt, welche das betroffene Datenobjekt in dem Modell rot markiert. Außerdem werden bereits im Rahmen der Validierung die Endzustände des Prozesses ermittelt. Die Prozessmodelle müssen so gestaltet sein, dass sich aus ihnen mindestens ein finaler Case-Zustand ableiten lässt. Ist dies nicht der Fall, d.h. ist die Anzahl der ermittelten, finalen Case-Zustände gleich null, erhält der Anwender dazu die entsprechende Fehlermeldung. Das Aufzeigen eines Prozessmodells ist an dieser Stelle nicht notwendig, daher beschränkt sich diese Fehlermeldung auf einen einfachen Error-Dialog, welcher in Abbildung 43 zu sehen ist.

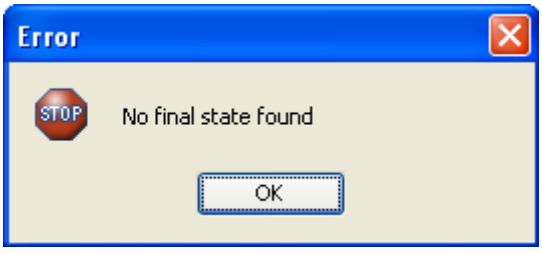


Abbildung 43: Fehlermeldung über fehlenden finalen Zustand

Eine sehr komplexe, aber ebenso wichtige Validierung ist die Überprüfung darauf, dass sich keine der definierten Zustandsübergänge widersprechen.

Der Prototyp erkennt selbstständig die sich aus den Prozessmodellen ableitenden Zustandsübergänge, indem er für jedes Datenobjekt anhand von Sequenzflüssen *Vorgänger-Datenobjekte* sucht. Für jeden gefunden Vorgänger erstellt er einen Zustandsübergang von diesem Vorgänger-Zustand hin zu dem Zustand, der sich am untersuchten Datenobjekt befindet. Die dabei zuerst gefundene Aktivität stellt den Auslöser für diesen Zustandsübergang dar. Eventuell gefundene Bedingungen an Sequenzflüssen werden ebenfalls am Zustandsübergang gespeichert. Die zusätzlich definierten Zustandsübergänge liest der Prototyp direkt aus der Benutzeroberfläche, in der sie erstellt wurden. Während der Validierung werden sämtliche Zustandsübergänge, unabhängig davon, ob sie prozessflussbasiert oder zusätzlich definiert sind, auf Widersprüche hin untersucht. Dabei folgt der Prototyp dem in Kapitel 5.3 vorgestellten Validierungsprinzip für Zustandsübergänge. Stößt die Validierung auf solch einen Zustandsübergang, welcher einem anderen widerspricht, wird der Anwender darüber mit Hilfe der in Abbildung 44 zu sehenden Fehlermeldung unterrichtet.

Abbildung 44: Fehlermeldung für widersprüchliche Zustandsübergänge

Erzeugt wurde diese Fehlermeldung dadurch, dass die Bedingung des zuvor in Abbildung 39 gesehenen, zusätzlichen Zustandsübergangs auf *Freigegeben:==:wahr* geändert wurde. Damit widerspricht er dem prozessbedingten Zustandsübergang, welcher den Case durch den Auslöser *Angebot freigeben* vom Zustand *PS-ergaenzt* in den Zustand *Freigegeben* versetzt, wenn die Bedingung *Freigegeben == wahr* erfüllt ist.

Laufen sämtliche Validierungen ohne das Auffinden eines Fehlers durch, erhält der Anwender auch darüber die passende, in Abbildung 45 gezeigte, Meldung. Der Prozess ist nun definiert und validiert, was seine Ausführung, erläutert im nächsten Kapitel, ermöglicht.

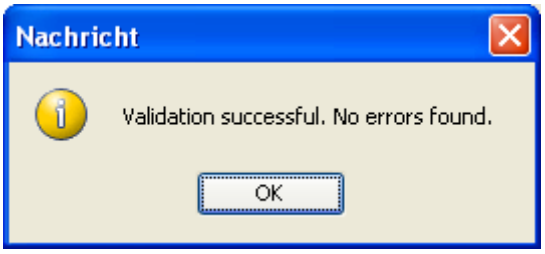


Abbildung 45: Erfolgreiche Validierung

Erst wenn die Validierung einmal erfolgreich durchgelaufen ist, lässt sich der Dialog des PCM Utilitys über den Button *Fertig stellen* beenden und sämtliche Einstellungen werden übernommen. Nun ist der definierte und validierte Prozess bereit für die Ausführung.

6.3.3 Ausführung

Die Ausführung von Cases für definierte und validierte Prozesse wird durch die zum Prozesspaket gehörenden technischen Workflows realisiert. Einen großen Anteil daran übernimmt der Workflow *PCM-Main-Workflow*, welcher in Anhang C vollständig abgebildet ist und im Folgenden als *Haupt-Workflow* bezeichnet wird.

Zur Erstellung von Cases verfügt der Haupt-Workflow über einen Prozessstarter in Form des Moduls *PCM-Start*, welches als Ad-hoc-Prozessstarter konfiguriert ist. Das bedeutet, dass sich über die Taskliste des Enterprise Portals der inubit Suite der Workflow an dieser Stelle starten lässt. Abbildung 46 zeigt das Start-Modul, mit einer Tilde (~) in der linken, oberen Ecke als Kennzeichnung für Ad-hoc-Prozessstarter, und den Button auf der Taskliste des Enterprise Portals, um diesen Prozessstarter auszulösen.

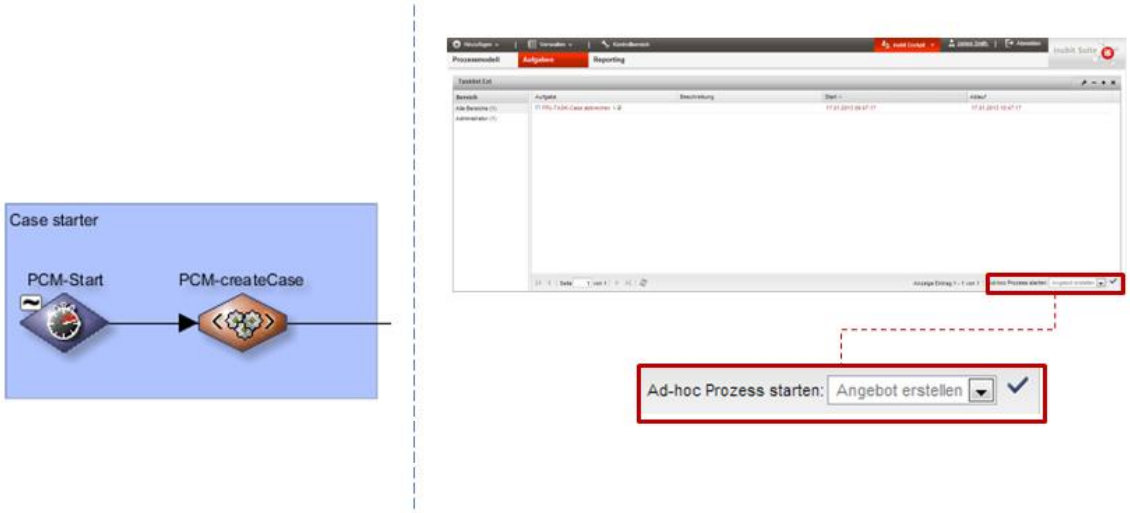


Abbildung 46: Case Starter Modul (links) und auslösender Button (rechts)

Das ebenfalls in Abbildung 46 zu sehende Modul *PCM-createCase* erzeugt die passende XML-Nachricht, die im PCM-Utility die Instanziierung eines Cases auslöst. Diese Nachricht fließt anschließend durch das Modul *PCM-join*, welches die Nachricht unverändert in den Workflow-Connector *PCM-WFC-PCM-Util,* und damit in das PCM Utility, weitergibt. Dort wird ein neuer Case erzeugt, mit einer ID versehen und Instanzen für sämtliche Aktivitäten und Datensätze des Prozesses angehängt. Das PCM Utility meldet daraufhin die erfolgreiche Erstellung des Cases. Anschließend entscheidet das Modul *PCM-demux* wie mit dem Case weiter verfahren wird. Dieses Modul, welches in Abbildung 47 zu sehen ist, verfügt über vier ausgehende Sequenzflüsse, die unter verschiedenen Bedingungen beschritten werden. Der in dieser Abbildung gezeigte Teil des Haupt-Workflows wird auch als *Schleife* bezeichnet, da der Workflow immer wieder hier ankommt, solange ein Case nicht abgeschlossen wurde.

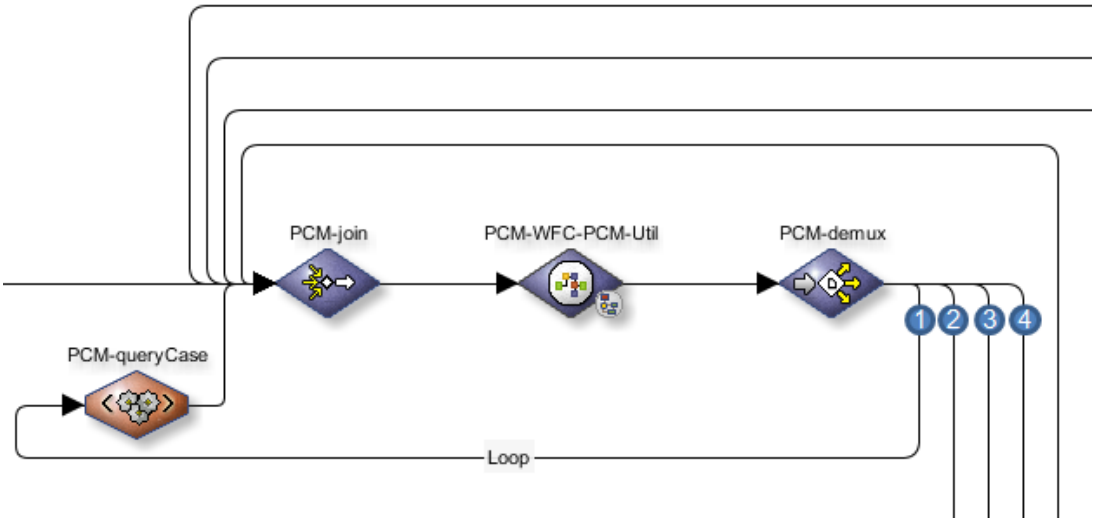


Abbildung 47: Schleife des Haupt-Workflows

Der erste dieser Sequenzflüsse, in Abbildung 47 mit der Ziffer *1* belegt, wird immer dann begangen, wenn

- ein neuer Case erzeugt wurde, oder
- die Werte von Datensatzinstanzen eines Cases geändert wurden, oder
- der Zustand einer Aktivitätsinstanz geändert wurde.

In allen drei Fällen wurde eine erneute Zustandsprüfung von Aktivitätsinstanzen (vgl. Kapitel 5.2.3.1) durchgeführt, weshalb der Case nun erneut abgefragt werden muss. Die dazu nötige Eingangsnachricht erzeugt das Modul *PCM-queryCase* und das PCM Utility antwortet daraufhin mit allen Informationen zu dem abgefragten Case. Wie so eine Antwort aussehen kann, zeigt Abbildung 48. Zu den gelieferten Informationen gehören neben der Case-ID und dem Case-Zustand auch die Datensatzinstanzen mit ihren aktuellen Werten, sowie sämtliche Aktivitätsinstanzen mit ihren Zuständen und ggf. einer Information darüber, ob es sich bei ihnen um Instanzen von Skript-Tasks handelt.

Abbildung 48: Antwort des PCM Utilitys nach Abfragen eines Cases

Nach einer solchen Abfrage eines Cases wird einer der drei weiteren ausgehenden Sequenzflüsse des Moduls *PCM-join* beschritten. Sollte eine der Aktivitätsinstanzen des abgefragten Cases sich im Zustand *bereit* (hier: *ready*) befinden und gleichzeitig vom Typ *Skript* sein, so würde der Sequenzfluss mit der Nummer *4* (s. Abbildung 47) weiter verfolgt werden. Dieser führt zu dem Teil des Haupt-Workflows, welcher mit dem Titel *Script Task Handling* beschriftet ist. Dort ruft das Modul *PCM-generic-wfc* mit Hilfe des konfigurierten Variablen-Mappings den entsprechenden Workflow, welcher im Prozessmodell am Skript-Task hinterlegt wurde, auf. Nach Abarbeitung dieses Workflows, wird die Aktivitätsinstanz des Skript-Tasks in den Zustand *durchgeführt* versetzt. Dies geschieht über die XML-Nachricht, die das Modul *PCM-setScriptTaskToDone* erstellt und zum PCM Utility weiterleitet.

Existiert nach der Abfrage eines Cases keine Aktivitätsinstanz eines Skript-Tasks im Zustand *bereit*, wird der ausgehende Sequenzfluss mit der Nummer *3* beschritten. Dieser führt zum Rahmen mit dem Titel *Human Task Handling*, wo der Datenstrom zunächst per Workflow-Connector zum Workflow *PCM-TaskHandling* weitergeleitet wird, welcher für alle Aktivitätsinstanzen im Zustand *bereit* einen Task erstellt. Abbildung 49 (Hintergrund) zeigt die Taskliste mit den zwei Tasks *Angebot erstellen* und *Case abbrechen*, welche direkt nach der Instanziierung zur Durchführung bereit sind. Hier kann sie der Case-Worker zur Bearbeitung auswählen,

wodurch sich jeweils ein Formular öffnet (s. Abbildung 49, Vordergrund), über welches ein Task bearbeitet, abgeschlossen oder ggf. ausgelassen werden kann.

Abbildung 49: Taskliste (Hintergrund) und Taskformular (Vordergrund)

Der Workflow *PCM-TaskHandling* muss vom Anwender individuell angefertigt werden und wird zu einem späteren Zeitpunkt dieses Kapitels näher erläutert. Dort wird ebenfalls darauf eingegangen, wie Tasks erzeugt und deren Formulare gestaltet werden.

Der Haupt-Workflow läuft weiter, sobald ein Task über die Taskliste bearbeitet, d.h. abgeschlossen oder ausgelassen, wurde. Für den Fall, dass ein Task über den Button *Fertig* abgeschlossen wurde, werden zunächst alle Formulardaten des Tasks, also die Werte der Datensatzinstanzen dieses Cases gespeichert. Dazu erstellt das Modul *PCM-XSLT-changeData* den entsprechenden Datenstrom, welcher dann per Workflow-Connector *PCM-WFC-PCM-Util* in das PCM Utility gegeben wird, wo die Änderung der Daten vorgenommen wird. Wie in Abbildung 50 zu sehen, erfolgt anschließend ein weiterer Aufruf eines anderen Workflows. Der Workflow *PCM-deleteTasks* löscht anhand der Case-ID alle zum Case gehörenden Tasks aus der Taskliste, da diese auf Grund der Datenänderung möglicherweise inzwischen veraltete Daten beinhalten. Zuletzt wird die Aktivitätsinstanz des abgeschlossenen Tasks mit Hilfe des Moduls *PCM-XSLT-setActivityStateToDone* in den Zustand *durchgeführt* versetzt.

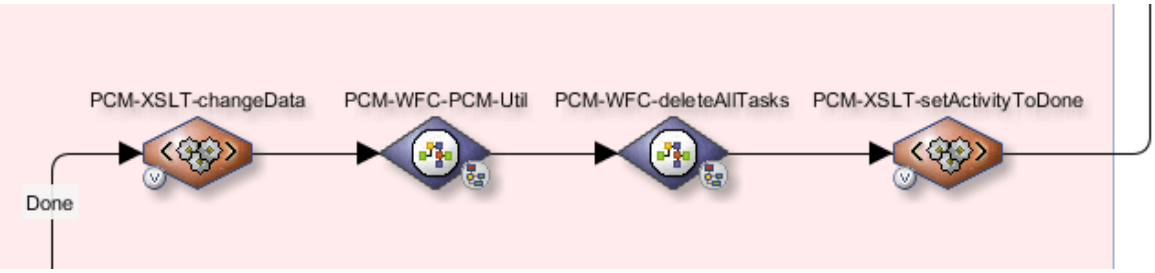


Abbildung 50: Behandlung eines abgeschlossenen Tasks

Wurde ein Task über den Button *Auslassen* ausgelassen, wird lediglich der Zustand der zugehörigen Aktivitätsinstanz in den Zustand *ausgelassen* versetzt. In diesem Fall werden keine Datenänderungen wirksam, wodurch auch kein Löschen der anderen, zum gleichen Case gehörenden, Tasks notwendig ist. Unabhängig davon, ob ein Task durchgeführt oder ausgelassen wurde, führt der Workflow anschließend wieder zurück zur o.g. Schleife, wodurch eine erneute Zustandsprüfung der Aktivitätsinstanzen ausgelöst wird und die bisher beschriebene Prozedur von vorne beginnt.

Der letzte ausgehende Sequenzfluss des Moduls *PCM-demux* der Schleife dieses Workflows, in Abbildung 47 mit der Nummer *2* gekennzeichnet, stellt den Alternativpfad an dieser Stelle dar. Er wird dann begangen, wenn keine Bedingungen der anderen Sequenzflüsse zutreffen. Dies ist bspw. dann der Fall, wenn zu einem Case keine Aktivitätsinstanzen im Zustand *bereit* existieren.

Damit ist die Funktionalität des Haupt-Workflows beschrieben. Bei der Erläuterung des *Human Task Handlings* wurde darauf verwiesen, dass der Workflow zur Erstellung von Tasks individuell vom Anwender erstellt werden muss. Anhand des dazu entwickelten Workflows für den Angebotsprozess soll beschrieben werden, wie solch eine Implementierung aussehen kann. In Anhang D befindet sich eine komplette Abbildung dieses Workflows *PCM-TaskHandling*.

Der Workflow beginnt mit einem generischen Teil, in welchem der Datenstrom an einen weiteren, mitgelieferten Workflow übergeben wird. Der Workflow *PCM-checkExistingTasks* prüft, ob für Aktivitätsinstanzen, die sich im Zustand *bereit* befinden bereits Tasks existieren, da in diesem Fall für sie kein neuer Task erzeugt werden müsste. Am Ende erstellt dieser Hilfs-Workflow eine XML-Nachricht, die sämtliche Aktivitätsinstanzen auflistet und jene Knoten, zu denen ein Task erzeugt werden muss, mit einem zusätzlichen Kind-Element *erzeugeTask* kennzeichnet. Abbildung 51 zeigt beispielhaft den Eingangs- und Ausgangsdatenstrom

dieses Workflows. Der Ausgangsdatenstrom wird dann wieder in den individuellen Workflow *PCM-TaskHandling* zurückgegeben.

Abbildung 51: Ein- und Ausgangsdatenstrom von *PCM-checkExistingTasks*

Die beiden Aktivitätsinstanzen *Angebot erstellen* und *Case abbrechen* befinden sich im Zustand *bereit* (in der Abbildung *ready*). Da der Workflow zu keiner der beiden Aktivitätsinstanzen einen bereits bestehenden Task gefunden hat, wurden beide um das Kind-Element *erzeugeTask* (rechts in der Abbildung) ergänzt. Dies dient dem nun weiterlaufenden Workflow *PCM-TaskHandling*. In diesem existiert für jede Aktivität vom Typ *Human Task*, die in einem der Prozessmodelle vorkommt, ein eigener Task Generator. Vor jedem dieser Task-Generatoren existiert außerdem ein Modul vom Typ *Assign*, wie in Abbildung 52 beispielhaft für die Aktivität *Angebot erstellen* gezeigt.

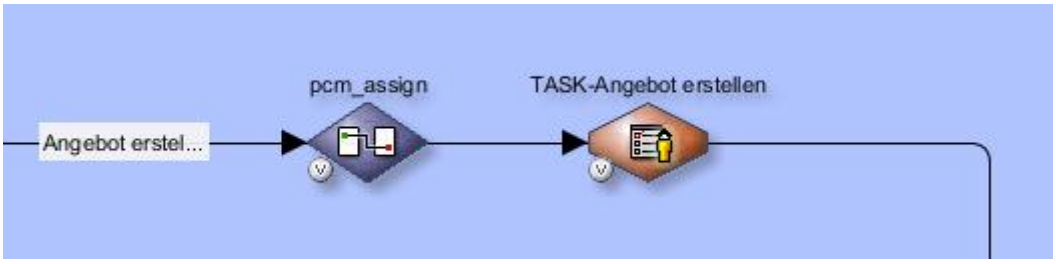


Abbildung 52: Task Generator mit führendem Assigner

Die Funktion des Assign-Moduls besteht darin, die Workflow-Variable *par.Aufgabe* mit dem Namen der aktuellen Aktivität zu belegen. Diese Variable wird u.a. dafür verwendet, dem Task seinen passenden Namen zu geben. Tasknamen bestehen in diesem Prototyp immer aus folgender Zeichenkette: *<Aktivitätsname> (<Case-ID>)*. Aber auch bei der Gestaltung des Tasks durch das Task-Generator-Modul kann auf die Variable p*ar.Aufgabe* zugegriffen werden, um bspw. eine passende Überschrift zu wählen. Der Task-Generator selbst stellt dann den Task in Form eines Formulars bereit. Die Gestaltung dieses Formulars obliegt ganz dem Anwender und kann für jede Aktivität individuell, über die verschiedenen Task-Generatoren, festgelegt werden. Dass nur für die Aktivitäten, zu denen sich eine Aktivitätsinstanz im Zustand *bereit* befindet, ein Task angelegt wird, garantieren die Sequenzflüsse, die zu den einzelnen Assign-Modulen führen. So wird der in Abbildung 52 gezeigte Sequenzfluss nur dann beschritten, wenn die an ihm definierte und in Abbildung 53 zu sehende Bedingung erfüllt ist.

Abbildung 53: Bedingung am Sequenzfluss im Task-Handling Workflow

In Worten ausgedrückt besagt diese Bedingung:

Wenn in dem Datenstrom ein Knoten erzeugeTask *existiert, der unterhalb eines Knotens* Aktivitaetsinstanz *liegt, welcher wiederum über ein Kind-Element mit dem Namen* Name *und dem Wert* Angebot erstellen*, verfügt. Außerdem muss der Knoten* Aktivitaetsinstanz *ein Kind-Element des Knotens* case *sein, welcher wiederum ein Kind-Element des Knotens* pcmsresponse *ist*.

Angewandt auf den Datenstrom der linken Seite aus Abbildung 51 trifft diese Bedingung zu. Der Sequenzfluss würde beschritten und damit ein Task für diese Aufgabe erstellt werden. Jeder einzelne Sequenzfluss, der in diesem Workflow zu einem der Assign-Module und darüber in einen Task-Generator führt, verfügt über die o.g. Bedingung mit der Anpassung des Namens der Aktivität. Sollte für keine Aktivität ein Task erzeugt werden müssen, bspw. weil bereits alle Aktivitätsinstanzen im Zustand *bereit* über einen Task verfügen, wird der Sequenzfluss mit dem Titel *otherwise* beschritten, wodurch der Workflow ohne Taskerzeugung zu Ende läuft.

6.4 Schnittstelle

Zum Abschluss der Dokumentation der prototypischen Implementierung soll in diesem Kapitel noch die Möglichkeit zum *externen Zugriff* auf Prozess- und Case-Daten erläutert werden. Externer Zugriff meint an dieser Stelle das Auslesen oder Verändern von Prozess- und Case-Daten über individuell entwickelte Workflows, z.B. zur Realisierung von Skript-Tasks. Dazu werden im Folgenden die existierenden Methoden beschrieben und jeweils ihre Eingangs- wie Ausgangsnachricht in XML-Form präsentiert. Um die Methoden anzusteuern, muss ihre Eingangsnachricht in einem Workflow erstellt und anschließend per Workflow-Connector an das PCM-Utility übergeben werden.

6.4.1 Prozess abfragen

Die Abfrage eines Prozesses wird über die in Listing 5 gezeigte XML-Nachricht ausgelöst. Da ein PCM-Utility immer nur genau *einen* Prozess definiert, ist keine weitere Angabe darüber erforderlich, welcher Prozess abgefragt werden soll.

```
1: <?xml version="1.0" encoding="UTF-8"?>
2: <pcms>
3:     <method>queryProcess</method>
4: </pcms>
```

Listing 5: XML-Nachricht zum Abfragen eines Prozesses

Die Antwort des PCM-Utilitys enthält folgende Informationen:

- Der Name des Prozesses
- Die verknüpften BPDs mit den in ihnen vorkommenden Aktivitäten
- Die Datensätze des verknüpften Datenmodells
- Sämtliche Case-Zustände
- Sämtliche Zustandsübergänge
- Die zum Prozess instanziierten Cases

6.4.2 Case abfragen

Listing 6 zeigt die XML-Nachricht zum Abfragen der Daten eines Cases. Der Knoten *caseID* muss eine gültige Case-ID enthalten. Andernfalls besteht die Antwort des PCM-Utilitys in der Fehlermeldung, dass kein Case mit übergebener Case-ID vorhanden ist.

```
1: <?xml version="1.0" encoding="UTF-8"?>
2: <pcms>
3:      <method>queryCase</method>
4:      <caseID/>
5: </pcms>
```

Listing 6: XML-Nachricht zum Abfragen eines Cases

Die Antwort des PCM-Utilitys enthält folgende Informationen:

- Die Case-ID.
- Den aktuellen Case-Zustand.
- Sämtliche Datensatzinstanzen mit ihren aktuellen Werten.
- Sämtliche Aktivitätsinstanzen mit ihren aktuellen Zuständen.

6.4.3 Case erzeugen

Die XML-Nachricht in Listing 7 löst bei Übergabe an das PCM-Utility die Instanziierung eines neuen Cases aus.

```
1: <?xml version="1.0" encoding="UTF-8"?>
2: <pcms>
3:      <method>createCase</method>
4: </pcms>
```

Listing 7: XML-Nachricht zur Instanziierung eines Cases

Im Falle einer erfolgreichen Instanziierung gibt das PCM-Utility die ID des neu erstellten Cases zurück.

6.4.4 Werte von Datensätzen ändern

Die Schnittstelle ermöglicht das Ändern der Werte von Datensatzinstanzen für einen Case. Dazu muss der Knoten *caseID* aus Listing 8 (Zeile 4) eine gültige Case-ID enthalten. Der folgende Knoten *datasets* kann eine beliebige Anzahl von *dataset*-Knoten enthalten. Jeder *dataset*-Knoten enthält einen Knoten *datasetName*, welcher den Namen des zu ändernden Datensatzes angibt. Wenn zu dem Prozess kein Datensatz mit dem hier angegebenen Namen gehört, gibt das PCM-Utility eine entsprechende Fehlermeldung zurück. Der Knoten *newValue* liefert den neuen Wert, welchen der zuvor genannte Datensatz einnehmen soll.

```
1: <?xml version="1.0" encoding="UTF-8"?>
2: <pcms>
3:      <method>changeData</method>
4:      <caseID/>
5:      <datasets>
6:        <dataset>
7:             <datasetName/>
8:             <newValue/>
9:        </dataset>
10:     </datasets>
11: </pcms>
```

Listing 8: XML-Nachricht zum Verändern von Datensätzen

Nach der Verarbeitung einer solchen XML-Nachricht gibt das PCM-Utility für jeden übergebenen Datensatz eine Meldung darüber zurück, ob die Datenänderung erfolgreich abgeschlossen werden konnte.

6.4.5 Aktivitätszustand ändern

Das Abarbeiten von Tasks ist nicht der einzige Weg um den Zustand von Aktivitätsinstanzen zu ändern. Über die Methode *setActivityState* und die in Listing 9 gezeigte XML-Nachricht, kann dies auch aus individuellen Workflows heraus geschehen. Dazu muss der Knoten *caseID* eine gültige Case-ID enthalten. Während der Knoten *activity* die zu ändernde Aktivitätsinstanz identifiziert, gibt der Knoten *newState* den neuen Zustand an, den die Aktivitätsinstanz einnehmen soll. Dieser Zustand muss einer des in Kapitel 5.2 vorgestellten Lebenszyklus von Aktivitätsin-

stanzen sein. Andernfalls gibt das PCM Utility eine entsprechende Fehlermeldung zurück. Wird eine Aktivitätsinstanz über diese Schnittstelle in den Zustand *durchgeführt* versetzt, führt das PCM-Utility automatisch eine erneute Zustandsprüfung sämtlicher Aktivitätsinstanzen durch.

```
1: <?xml version="1.0" encoding="UTF-8"?>
2: <pcms>
3:      <method>setActivitiyState</method>
4:      <caseID/>
5:      <activity/>
6:      <newState/>
7: </pcms>
```

Listing 9: XML-Nachricht zum Ändern des Zustands einer Aktivitätsinstanz

Beim Setzen eines neuen Zustandes für eine Aktivitätsinstanz übernimmt das PCM-Utility *keine* fachliche Prüfung, ob die Aktivitätsinstanz in diesen Zustand übergehen kann. Es liegt in der Verantwortung des Anwenders sicherzustellen, dass nur sinnvolle Zustandsänderungen vorgenommen werden und somit Inkonsistenzen innerhalb eines Cases verhindert werden.

6.4.6 Prozess zurücksetzen

Die letzte angebotene Funktionalität der hier beschriebenen Schnittstelle ist die des Zurücksetzens eines Prozesses. Ausgelöst durch die in Listing 10 gezeigte XML-Nachricht werden dabei sämtliche eingelesenen und an dem Prozess gespeicherten Daten gelöscht und beim nächsten Aufruf des PCM-Utilitys erneut eingelesen. Auch alle zu diesem Zeitpunkt bereits instanziierten Cases des Prozesses sind anschließend nicht mehr vorhanden. Die am PCM-Utility vorgenommenen Konfigurationen bleiben allerdings erhalten.

```
1: <?xml version="1.0" encoding="UTF-8"?>
2: <pcms>
3:      <method>resetProcess</method>
4: </pcms>
```

Listing 10: XML-Nachricht zum Zurücksetzen eines Prozesses

Das Zurücksetzen eines Prozesses kann bspw. dann sinnvoll sein, wenn vor einer produktiven Nutzung bereits zu Testzwecken Cases erstellt und *durchgespielt* wurden. Bei einer erfolgreichen Zurücksetzung meldet das PCM-Utility dies über eine XML-Nachricht mit dem Knoten *<resetProcess>ok</resetProcess>*.

7 Zusammenfassung und Ausblick

Zum Abschluss dieser Masterthesis soll dieses Kapitel die geleistete Arbeit noch einmal reflektieren und mögliche Ansätze zur Weiterführung des Themas aufzeigen. Dazu folgen zunächst eine Zusammenfassung der Thematik und ein Überblick über das angewandte Vorgehen. Dabei soll vor allem herausgearbeitet werden, welche zu Beginn dieser Masterarbeit gesetzten Ziele auf welche Art und Weise erreicht worden sind. Im Falle eines nicht, bzw. nicht vollständig, erfüllten Ziels soll darauf eingegangen werden, wie sich das Delta zwischen Ziel und Erfüllung gestaltet und worauf es sich begründet. Am Ende steht der Ausblick mit den zentralen Fragen danach, wie das entwickelte Konzept in Zukunft unterstützt werden kann und inwieweit sich der entwickelte Prototyp für eine produktive Realisierung von Geschäftsprozessen eignet bzw. wie dieser Prototyp für eine Produktivlösung erweitert werden müsste.

7.1 Zusammenfassung und kritische Betrachtung

Im Rahmen dieser Arbeit sollte untersucht werden, wie sich Geschäftsprozesse mit Hilfe von Production Case Management unterstützen und technisch umsetzen lassen. Bei der Erläuterung dazu notwendiger Grundlagen wurde der Begriff *Case Management* anhand von unterschiedlichen Quellen in Adaptive Case Management und Production Case Management unterteilt und letzteres als Möglichkeit zur Verschmelzung von klassischem Prozessmanagement und Case Management erkannt. Hauptgrund dafür ist die Verwendung von Prozessmodellen, deren Anreicherung um Prozessdaten und die daraus resultierende Steuerung von Cases sowohl durch Prozessflüsse als auch durch Prozessdaten. Beide Aspekte führen einen Case durch seine möglichen Zustände. Anschließend wurde untersucht, welche fachlichen Inhalte demnach aufgenommen werden müssen, um PCM zu realisieren und welche Methoden oder Notationen dafür bereits existieren. Für die Aufnahme von Prozessflüssen wurde die BPMN als geeignetes und weit verbreitetes Mittel erkannt, während UML-Klassendiagramme die nötige Funktionalität bereitstellen, um Geschäftsdaten zu erfassen. Das BPMN-Artefakt *Datenobjekt* schafft die Verbindung zwischen Prozess- und Datenmodell und bietet außerdem die Möglichkeit Zustände einzunehmen, wodurch das Bedürfnis an modellierbaren Case-Zuständen erfüllt werden kann. Es wurde erkannt, dass neben den Case-Zuständen, die sich durch die Modellierung von Prozessmodellen ergeben, auch solche von Nutzen sind, die sich zusätzlich definieren lassen. Zwar konnte hierfür

keine vorhandene Notation gefunden werden, dafür wurden diese *zusätzlichen Zustandsübergänge* bei der Konzeption einer PCM-Umgebung genau definiert und die Anforderungen an sie deutlich gemacht. Auch für Vor- und Nachbedingungen von Aktivitäten wurde keine standardisierte Modellierungsmethode gefunden.

Nach der Zusammenstellung aller benötigten, fachlichen Inhalte für eine PCM-Unterstützung, wurden diese in einem gemeinsamen Modell, im Verlauf der Arbeit als *PCM-Metamodell* bezeichnet, vereint. Mit Hilfe dieses Modells konnten alle Verbindungen zwischen den fachlichen Inhalten dargestellt werden. Auch jene, die in unterschiedlichen Modelltypen erfasst oder gar unabhängig von vorhandenen Notationen erfasst werden. Das PCM-Metamodell wurde in seinen Einzelheiten erläutert und diente anschließend als Grundlage zur Erstellung einer Ausführungssemantik für PCM-Prozesse. Diese Ausführungssemantik beschreibt sehr detailliert, wie in einer PCM-Umgebung Cases instanziiert, durchgeführt und wieder terminiert werden. Dabei wurden Case-Zustände, die Datenlage eines Cases sowie das Abarbeiten von Aktivitäten als die Elemente beschrieben, die einen Case zwischen Instanziierung und Terminierung vorantreiben. Als Hauptaufgabe eines PCM-Systems wurde das Bereitstellen *durchführbarer* Aktivitätsinstanzen identifiziert. Dazu wurde die sog. Zustandsprüfung von Aktivitätsinstanzen vorgestellt, welche diese Aufgabe, in Abhängigkeit der *Case-Situation*, die sich durch die drei zuvor genannten Elemente definiert, übernimmt. Während der Erstellung der Ausführungssemantik hat sich gezeigt, dass das Zugrundelegen mehrerer Prozessmodelle für einen Prozess gewisse Schwierigkeiten mit sich bringt. So können bspw. sich widersprechende Prozessflüsse oder Zustandsübergänge entstehen. Auch die Verwendung der gleichen Aktivität in mehreren Modellen könnte, z.B. bei unterschiedlicher Definition von Vor- oder Nachbedingungen, eine nicht eindeutige Prozessdefinition zur Folge haben. Um grundlegende, fehlerhafte Prozessmodellierung zu verhindern, ist auch die Vorstellung einiger Modellierungs-Best-Practices Teil des PCM-Konzepts. Diese Best-Practices sollen einem möglichen Anwender einen einfachen Einstieg in die PCM-Modellierung ermöglichen und auf mögliche Fehlerquellen hinweisen.

Neben der Erstellung eines Konzeptes für eine PCM-Umgebung war es ein weiteres Ziel dieser Masterarbeit das angefertigte Konzept mit Hilfe eines technischen Prototyps zu erproben. Solch ein Prototyp wurde erstellt und dokumentiert. Er ist in das inubit-Umfeld aus inubit Process Engine, inubit Workbench und dem Enterprise Portal eingebunden. In seiner Funktionalität setzt er weite Teile der Ausführungssemantik in die Tat um und bedient sich dabei den Möglichkeiten der inubit Suite zum Erstellen von Prozess- und Datenmodellen. Die tatsächliche

Ausführung übernimmt ein speziell für diesen Zweck implementiertes Modul und eine Reihe von generischen, technischen Workflows. Das Modul stellt dabei auch die Möglichkeit bereit, zusätzliche Zustandsübergänge zu definieren. Der Prototyp zeigt, dass die Realisierung des erstellten Konzeptes möglich ist, zeigt aber auch die Notwendigkeit einer zuverlässigen Validierung einer erstellten Prozessdefinition.

7.2 Ausblick

Abschließend soll in diesem Kapitel ein Ausblick darauf gerichtet werden, wie es mit der Case Management Thematik, vor allem in seiner prozessbezogenen Variante PCM, weitergehen kann. Besonderes Augenmerk soll dabei zunächst auf Modellierungsaspekte gelegt werden, die im Verlauf der Masterarbeit als bislang nicht standardisiert erkannt wurden. Anschließend wird untersucht, in welcher Form der entwickelte Prototyp weitere Verwendung finden kann.

Die möglicherweise interessanteste Neuerung könnte auf dem Gebiet der Prozessmodellierung entstehen. Derzeit wird von der OMG eine eigene Modellierungsnotation für die Zwecke von Case Management, die Case Management Modelling Notation (CMMN), entwickelt. Potentielle Inhalte dieser neuen Notation skizziert Stephen White in *Keynote 3*[53]. Dabei könnten Vor- und Nachbedingungen von Aktivitäten, die im Rahmen dieser Arbeit als Eigenschaften an betroffenen Aktivitäten definiert wurden, zukünftig eigene Modellierungselemente erhalten. Abbildung 54 zeigt, wie diese Elemente aussehen könnten.

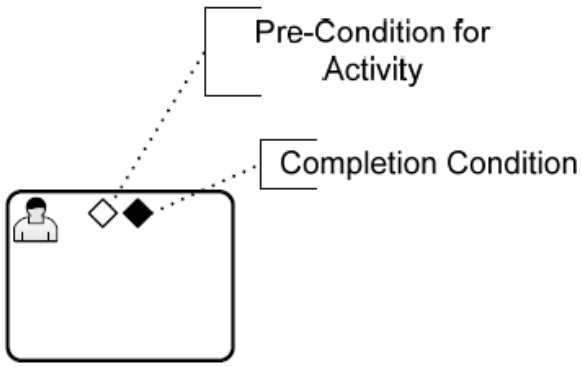


Abbildung 54: Mögliche Modellierung von Vor- und Nachbedingungen

[Quelle: **White 2012**]

[53] Vgl. **White 2012**, ab min. 33:30

Zur Definition von Case-Zuständen wurden in dieser Arbeit Datenobjekte mit ihrem Attribut *Zustand* herangezogen. In einer möglichen CMMN könnte dies durch den Knoten *Meilenstein* übernommen werden, wie in Abbildung 55 zu sehen ist.

Abbildung 55: Mögliche Modellierung von Meilensteinen

[Quelle: **White 2012**]

Auch das in dieser Quelle gezeigte Element *Content Event*, siehe Abbildung 56, könnte für das vorgestellte PCM-Konzept von Nutzen sein, indem es deutlich macht, an welcher Stelle die Datenlage eines Cases verändert wird.

Abbildung 56: Mögliche Modellierung von Content Events

[Quelle: **White 2012**]

Mit der Verwendung solcher Elemente könnten vorgestellte Modellierungen und entsprechende Teile der Ausführungssemantik vereinfacht werden. Überhaupt stellt das gesamte in dieser Arbeit vorgestellte Konzept zu einem PCM-Framework eine erste Idee auf diesem Gebiet dar. Weiterführende Untersuchung zu diesem Thema in technischen, algorithmischen und präsentationsrelevanten Fragestellungen ist jedoch wünschenswert.

Eine weitere ausstehende Erkenntnis ist die nach der Akzeptanz des entwickelten Frameworks in der Praxis. Die Entwicklung eines Tools zu dessen Umsetzung ist noch offen und der implementierte Prototyp stellt vielmehr einen Ansatz zum Aufzeigen grundlegender Funktionalitäten des Frameworks als eine Produktivlösung dar. Im Folgenden sollen daher verschiedene Aspekte genannt werden, um die der vorgestellte Prototyp für eine produktive Verwendung aus funktionaler und qualitätstechnischer Sicht erweitert werden müsste.

Funktional betrachtet sind der Umsetzung des Prototyps in eine produktreife Implementierung kaum Grenzen gesetzt. Die Einbeziehung von Zwischenereignissen ist zwar im PCM-Metamodell und auch der Ausführungssemantik vorhanden, jedoch nicht in dem Prototyp umgesetzt worden. Solche Zwischenereignisse, die Zustandsübergänge auslösen oder deren Eintreten eine Voraussetzung für nachfolgende Aktivitäten sein könnten, würden die Funktionalität deutlich steigern und sicherlich eine größere Anzahl an Anwenderwünschen erfüllen. Dies gilt auch für Einbeziehung verschiedener Rollen eines Geschäftsprozesses. Welcher Bearbeiter oder welche Abteilung in einem Unternehmen bestimmte Aktivitäten durchführt, wird in der BPMN klassischerweise über sog. *Bahnen*[54] (engl. *lanes*) ausgedrückt. Der Prototyp speichert diese Information, in welcher Bahn sich eine Aktivität befindet jedoch bisher nicht. Mit dieser zusätzlichen Information könnten die Task-Generatoren die Zuweisung der Aktivitätsinstanzen an verschiedene Anwender übernehmen. Überhaupt darf die Frage gestellt werden, ob die Umsetzung der Prozesse mit Hilfe von Tasks und der Taskliste realisiert werden muss. Die Datenströme, die das PCM-Utility erzeugt, könnten bspw. auch dazu genutzt werden, individuelle HTML-Seiten zu generieren oder ein Ticket-System zu integrieren. Für die weitere Realisierung mit Tasks ist auch die Entwicklung eines generischen Task-Generators denkbar, welcher den individuellen Teil der Umsetzung stark vereinfachen könnte. Des Weiteren verfügt der bisherige Prototyp bisher über keinerlei Möglichkeit zum Auswerten abgeschlossener oder noch laufender Cases. Das Speichern verschiedener Zeitpunkte, z.B. wann ein Case gestartet und abgeschlossen wurde, würde bereits erste Funktionalitäten zum Monitoring erlauben. Würde sogar jeder einzelne Schritt des Cases persistiert werden, ließen sich auch Auswertungen darüber erstellen, welche Prozesspfade die Cases zu welchen Prozentsätzen durchlaufen. Bisher werden die Prozess- und Casedaten ausschließlich in der JVM der inubit Suite gehalten und liegen damit nach einem Neustart des Servers nicht mehr vor. Für eine produktive Implementierung würde ein Konzept für eine konsistente Persistenz nötig sein. Möglicherweise reichen die Datenströme, die beim Abarbeiten eines Cases in den technischen Workflows erzeugt und im Dateisystem gespeichert werden aus, um einen Case neu zu laden. Dies ist jedoch bislang nicht realisiert und müsste umfassend getestet werden. Zuletzt ist es die Validierung, die auf ihre Vollständigkeit hin untersucht und ggf. erweitert werden müsste. Mögliche Anforderungen dazu könnten durch das Sammeln von Erfahrungen bei der Umsetzung verschiedener Szenarien identifiziert werden.

[54] Vgl. **OMG 2011**, S. 305

Aus qualitätstechnischer Sicht fehlt es dem Prototyp bisher vor allem an Tests, mit denen seine Funktionalität auch nach Änderungen am Quellcode oder innerhalb der Workflows schnell und einfach sichergestellt werden kann. Für den beteiligten Java-Quellcode wäre dies in Form von JUnit-Tests möglich. Außerdem könnten technische Workflows die Funktionalität des PCM-Utilitys anhand von erwarteten und tatsächlichen Ausgabe-Datenströmen bei fixen Eingabe-Datenströmen testen.

Auch wenn das entwickelte Framework aus Metamodell, Ausführungssemantik, Validierung und prototypischer Implementierung keinen Anspruch auf Vollständigkeit erhebt, so dient es doch als Einstieg in das komplexe Thema des Production Case Managements und gibt einen Überblick über die Herausforderungen und Möglichkeiten auf diesem Gebiet.

Literaturverzeichnis

Allweyer 2009 ALLWEYER, THOMAS: *BPMN 2.0 Business Process Model and Notation*. 2. Auflage. Kaiserslautern: Books on Demand GmbH, 2009. ISBN: 978-3-8391-2134-4

Bernstein et al. 1998 BERNSTEIN, A.; DELLAROCAS, C.; KLEIN, M.: *Towards Adaptive Workflow Systems: CSCW-98 Workshop Report* in *ACM SIGGROUP Bulletin*, Ausgabe 20, 1999.

Bider et al. 2012 BIDER, Ilia; JOHANNESSON, Paul; PERJONS, Erik: *Do workflow-based systems satisfy the demands of the agile enterprise of the future?* in *ACM 2012 1st International Workshop on Adaptive Case Management and other non-workflow approaches to BPM*, S. 89-94

Casati et al. CASATI, F.; CERI, S.; PERNICI, B.; Pozzi, G.: *Workflow Evolution* in *Conceptual Modelling – ER '96*, Seiten 438 – 455. Auflage 1996: Springer Verlag, 2008. ISBN: 978-3540617846

Freund et al. 2010 FREUND, J.; RÜCKER, B.; HENNINGER, T.: *Praxishandbuch BPMN*. 1. Auflage. München: Carl Hanser Verlag, 2010: ISBN: 978-3-446-41768-7

Gabler GABLER Wirtschaftslexikon: Management, URL: http://wirtschaftslexikon.gabler.de/Definition/management.html - Überprüfungsdatum: 09.10.2012

Gadatsch 2008 GADATSCH, Andreas: *Grundkurs Geschäftsprozess-Management*. 5. Auflage. Wiesbaden: GWV Fachverlage GmbH, 2008. ISBN: 978-3-8348-0363-4

Gartner 2012 GARTNER Inc.: *The Case for Case Management Solutions*. Gartner Inc., 2012.

Kecher 2011 KECHER, Christoph: UML 2 – Das umfassende Handbuch: 4. Auflage. Bonn: Galileo Press, 2011. ISBN: 978-3-8362-1752-1

Müller 2005 MÜLLER, Joachim; *Workflow-based Integration.* 1. Auflage. Berlin: Springer-Verlag, 2005. ISBN: 3-540-204-39-3

OMG 2011 OBJECT MANAGEMENT GROUP: *Business Process Model and Notation*: Version 2.0. URL: http://www.omg.org/spec/BPMN/2.0/PDF Überprüfungsdatum: 03.01.2013

PWC 2011 PRICEWATERHOUSECOOPER: *Zukunftsthema Geschäftsprozessmanagement*. URL: http://www.pwc.de/de_DE/de/prozessoptimierung/assets/PwC-GPM-Studie.pdf - Überprüfungsdatum: 17.09.2012

Reijers et al. 2003 REIJERS, H.; RITGER, J.; VAN DER AALST, W.: The Case Handling Case, 2003. URL: http://wwwis.win.tue.nl/~wvdaalst/publications/p212.pdf - Überprüfungsdatum: 06.02.2013

Sem et al. 2012 SEM, Helle Frisak; CARLSEN, Steinar; COLL, Gunnar John: *On two approaches to ACM.* Computas AS, Lysaker, Norway, 2012

Siebert et al. 1998 SIEBERT, R.; WESKE, M.: *Flexibilität und Kooperation in Workflow-Management-Systemen* in *Groupware und organisatorische Innovation. Tagungsband der D-CSCW '98*, Stuttgart 1998.

Silver 2011 SILVER, B.: *BPMN Method and Style*. 2. Auflage. Aptos: Cody-Cassidy-Press, 2011. ISBN: 978-0-9823681-1-4

Slama et al. 2011 SLAMA, D.; NELIUS, R.: *Enterprise BPM*. 1. Auflage. Heidelberg: dpunkt.verlag GmbH, 2011. ISBN: 978-3-89864-687-1

Strong et al. 1995 Strong, D.M.; Miller, S.M.: *Exceptions and Exception Handling in Computerized Information Processes* in *ACM Transactions on Information Systems*. Ausgabe 13, 1995. ISSN: 1046-8188.

Swenson 2010 SWENSON, K.: *Mastering The Unpredictable*. 1. Auflage. Meghan-Kiffer Press, 2010. ISBN: 978-0-929652-12-2

Swenson 2012a SWENSON, K.: *Position: BPMN is incompatible with ACM* in *ACM 2012 1st International Workshop on Adaptive Case Management and other non-workflow approaches to BPM*, S. 85-88..

Swenson 2012b SWENSON, K.: *Production Case Management vs. Adaptive Case Management*. URL: http://social-biz.org/2012/05/14/production-case-management-vs-adaptive-case-management/ - Überprüfungsdatum: 29.10.2012

Swenson 2012c SWENSON, K.: *Seven Categories to Replace BPM*. URL: http://social-biz.org/2012/04/25/not-to-praise-bpm-but-to-bury-it/ - Überprüfungsdatum: 06.01.2013

van der Aalst et al. 2001 VAN DER AALST, W.; BERENS, P.: *Beyond Workflow Management: Product-Driven Case Handling* in *International ACM SIGGROUP Conference on Supporting Group Work,* 2001.

van der Aalst et al. 2003 van der Aalst, W.; Stoffele, M.; Wamelink, J.: *Case Handling in Construction* in *Automation in Construction*, Ausgabe 12, 2003.

van der Aalst et al. 2005 VAN DER AALST, W.; WESKE, M.; GRÜNBAUER, D.: *Case Handling: A New Paradigm for Business Process Support*. Eindhoven, Potsdam, Apeldoorn, 2005.

Walter 2009 WALTER, Johann: *Geschäftsprozessmanagement umsetzen*. 1. Auflage. München: Carl Hanser Verlag, 2009. ISBN: 978-3-446-41801-1

Weske 2007 WESKE, Mathias: *Business Process Management*. 1. Auflage. Heidelberg: Springer Verlag, 2007. ISBN: 978-3-540-73521-2

White 2012 WHITE, Stephen: *Keynote 3: BPMN and its future*. URL: http://uttv.ee/naita?id=12923&keel=eng – Überprüfungsdatum: 04.02.2013

Markenerklärung

Die in diesem Werk wiedergegebenen Gebrauchsnamen, Handelsnamen, Warenzeichen usw. können auch ohne besondere Kennzeichnung geschützte Marken sein und als solche den gesetzlichen Bestimmungen unterliegen.

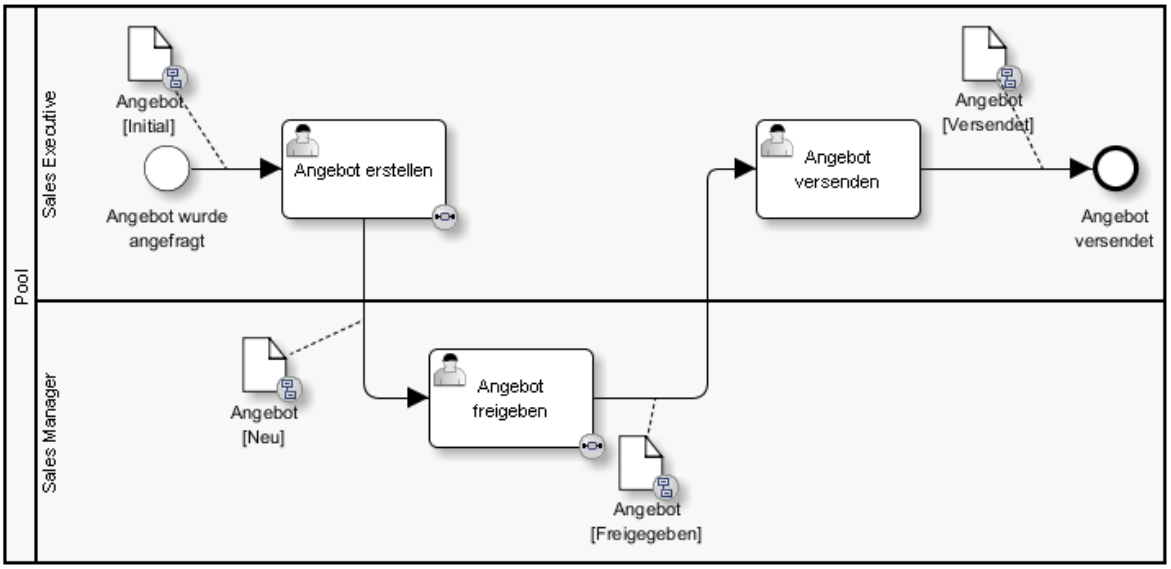
Anhang

A Operatoren für Bedingungen (mit Beispielen)

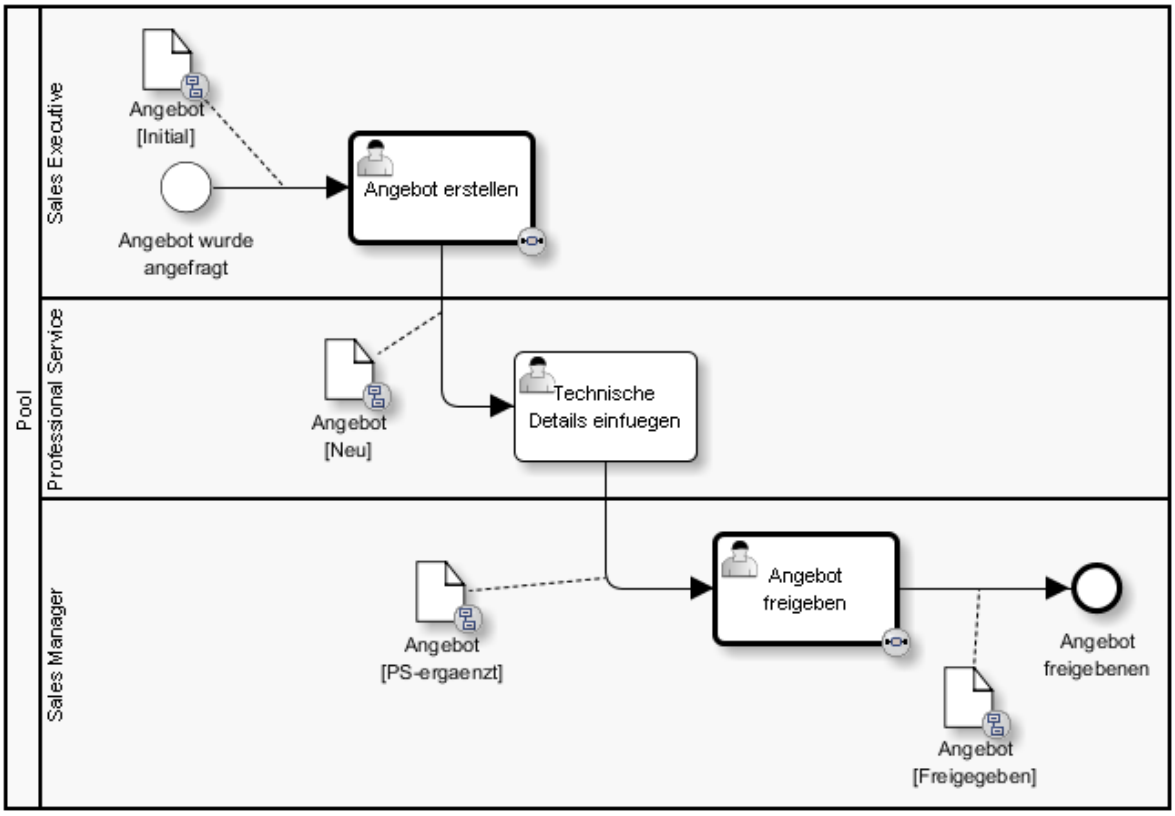
Operator	Bezeichnung	Format	Beispiel (bezogen auf Format)
==	ist gleich	Komponente == „GUI“	Komponente: „3rd-Party-Komponente“ ➔ Bedingung nicht erfüllt Komponente: „GUI“ ➔ Bedingung erfüllt
!=	ungleich	Komponente != „GUI“	Komponente: „3rd-Party-Komponente“ ➔ Bedingung erfüllt Komponente: „GUI“ ➔ Bedingung nicht erfüllt
<	kleiner als	Mahnstufe < 1	Mahnstufe: 0 ➔ Bedingung erfüllt Mahnstufe: 1 ➔ Bedingung nicht erfüllt
>	größer als	Mahnstufe > 3	Mahnstufe: 4 ➔ Bedingung erfüllt Mahnstufe: 3 ➔ Bedingung nicht erfüllt
set	gesetzt	Komponente set	Komponente: „3rd-Party-Komponente“ ➔ Bedingung erfüllt Komponente: „ “ ➔ Bedingung nicht erfüllt
!set	nicht gesetzt	Komponente !set	Komponente: „GUI“ ➔ Bedingung nicht erfüllt Komponente: „“ ➔ Bedingung erfüllt

B Fachliche Modelle des Angebotsprozesses

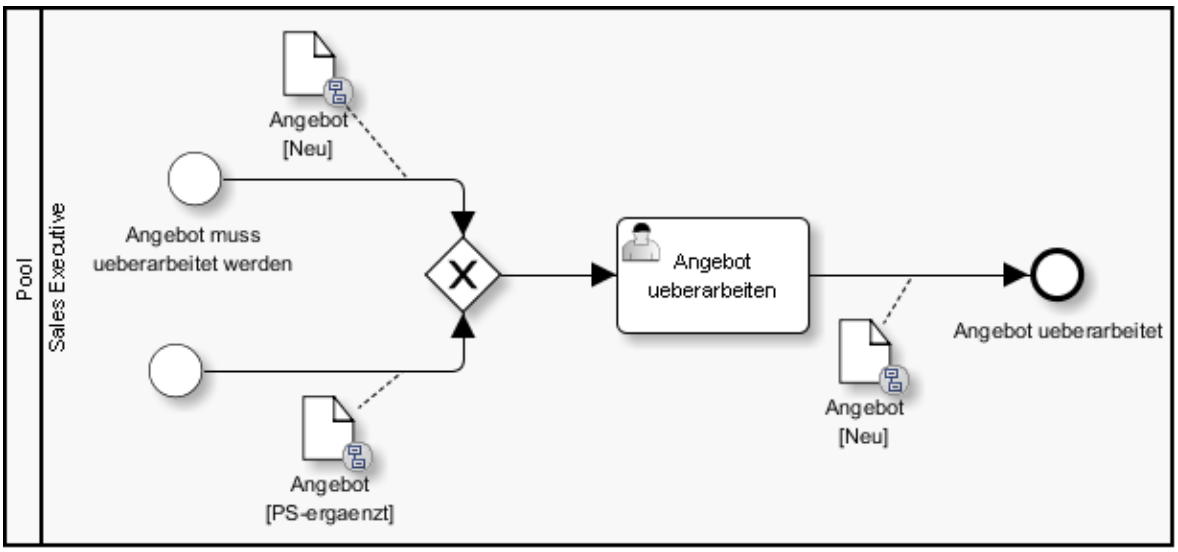
BPD, Einfaches Angebot erstellen



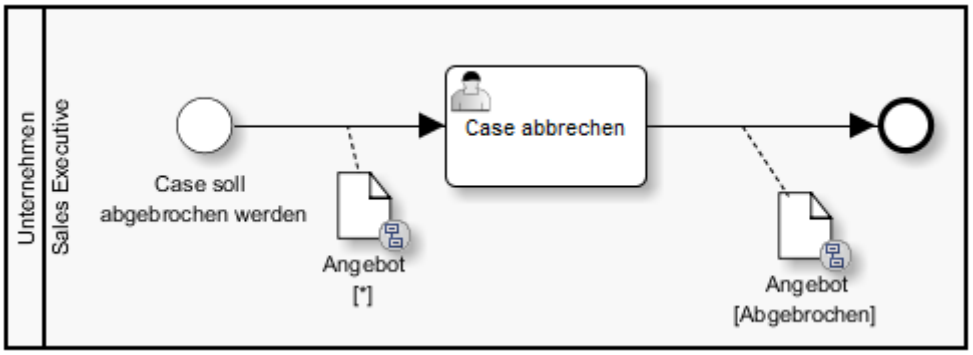
BPD, Komplexes Angebot erstellen



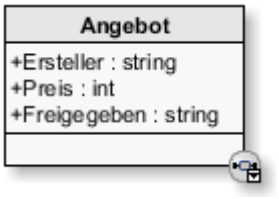
BPD, Angebot überarbeiten



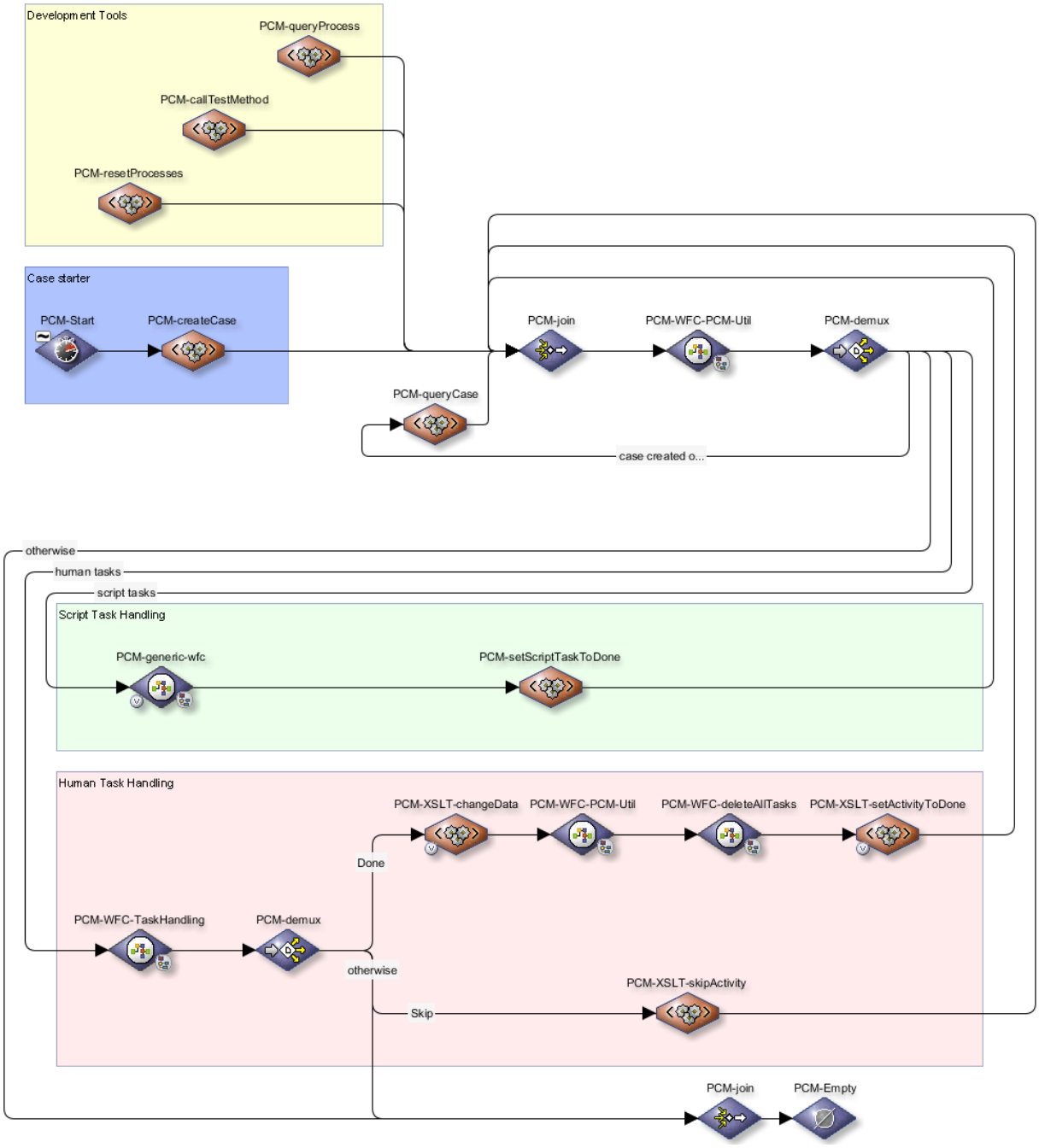
BPD, Case abbrechen



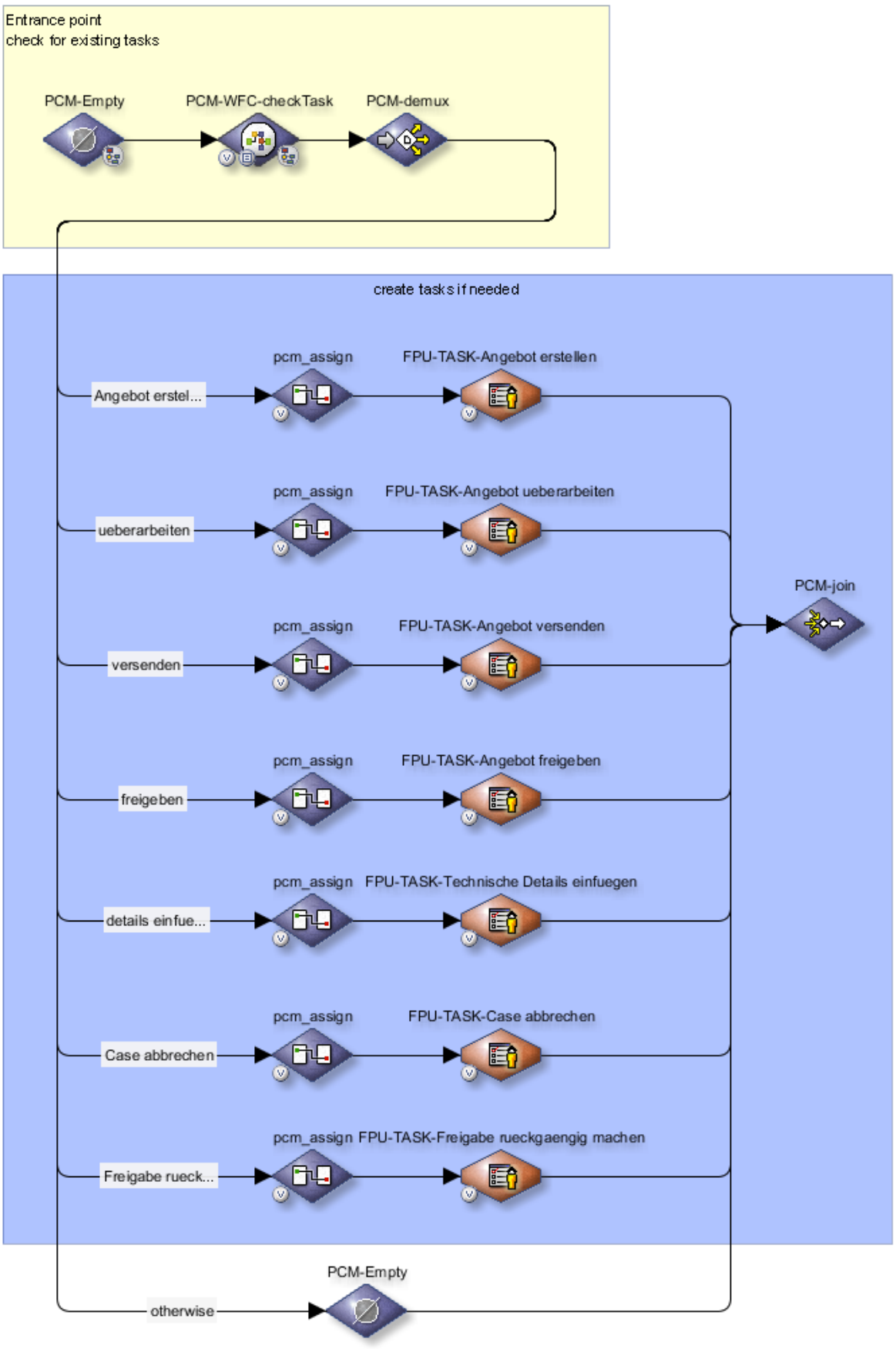
BOD, Angebot



C PCM Haupt-Workflow



D Workflow *PCM-TaskHandling*



E CD-Inhalte

Die beigefügte CD-ROM enthält die folgenden Inhalte:

- Diese Masterarbeit als PDF-Datei
- Die Installationsdateien für den entwickelten Prototyp. Dazu gehört:
    - Eine Installationsanleitung
    - Der Quellcode des PCM-Utilities inkl. javadoc-Dokumentation
    - Die verwendeten Diagramme und technische Worklfows
    - Weitere Bibliotheken
- Die folgenden, als PDF-Datei verfügbaren Quellen
    - **Bider et al. 2012**
    - **Casati et al.**
    - **OMG 2011**
    - **Reijers et al. 2003**
    - **Strong et al. 1995**
    - **Swenson 2012a**
    - **van der Aalst et al. 2001**
    - **van der Aalst et al. 2003**
    - **van der Aalst et al. 2005**
    - **White 2012**

Eidesstattliche Versicherung

Ich versichere hiermit, dass ich die vorliegende Masterthesis selbstständig und ohne fremde Hilfe angefertigt und keine andere als die angegebene Literatur benutzt habe. Alle von anderen Autoren wörtlich übernommenen Stellen wie auch die sich an die Gedankengänge anderer Autoren eng anlehnenden Ausführungen meiner Arbeit sind besonders gekennzeichnet. Diese Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Berlin, den 25. März 2013

Jan Oehlert