



Fachhochschul-Diplomstudiengang
SOFTWARE ENGINEERING
A-4232 Hagenberg, Austria

Realisierung teilautomatisierter Prozesse durch die Kombination von Ablaufsteuerung und unterstützter Kooperation

Diplomarbeit

zur Erlangung des akademischen Grades
Diplom-Ingenieur (Fachhochschule)

Eingereicht von

Robert Franz Schmelzer

Betreuer: Dr. Guido Gryczan, C1-WPS Hamburg
Begutachter: Mag. Dr. Stefan Hinterholzer

Juni 2005

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Hagenberg, am 30. Juni 2005

Robert Schmelzer

Kurzfassung

Durch die stetige Tendenz zum Outsourcing eines Geschäftsprozesses oder Teilen davon ergibt sich die Notwendigkeit die einzelnen Geschäftspartner zu koordinieren. Vor allem bei der Massenabwicklung von Kernprozessen bedeutet das einen erheblichen Aufwand und ist üblicherweise nur mit einer großteils automatisierten Abwicklung effizient realisierbar.

Für die Anbieter von Business Process Management, die sich auf die Koordination von Geschäftsprozessen spezialisieren, ist diese weitgehende Automation von Prozessen der wesentliche Erfolgsfaktor. Dem entgegengesetzt setzen aber sowohl Kunden als auch Auftraggeber eine immer höhere Flexibilität in der Abwicklung einzelner Geschäftsvorfälle voraus. Durch die Weiterentwicklung von Prozessen im Laufe der Zeit ist auch mit einer ständigen Anpassung der Schnittstellen, der Datenhaltung und auch des Prozessablaufes zu rechnen.

In diesem Spannungsfeld sind Systeme zur Geschäftsprozessabwicklung nötig, die typische Prozessabläufe vollautomatisiert abwickeln, aber auch eine flexible Einflussnahme bei besonderen Situationen erlauben.

Diese Arbeit versucht, diese Herausforderung durch die Implementierung eines Prozesssystems zu realisieren, das sowohl der Notwendigkeit der Automatisierung als auch der Herausforderung der Flexibilisierung gerecht wird. Dazu bedient sich das System dem Konzept der Prozessmuster und realisiert dieses durch die Metaphern von Laufzettel und Postkorb. Die Aufgabe der Prozesssteuerung übernimmt eine Workflow Engine.

Begleitend dazu wird das notwendige Investitionsvolumen durch die konsequente Verwendung von Open Source Komponenten minimiert und durch eine Rollentrennung in der Systementwicklung wird die Entwicklung vereinfacht.

In der abschließenden Betrachtung werden auch die Auswirkungen des Prozesssystems auf die Benutzer und im weiteren Sinne auf die Unternehmenskultur betrachtet.

Abstract

As a consequence of the ongoing outsourcing of key processes, the overhead for coordinating business partners is steadily growing. Especially the bulk of key processes needs a highly automated environment to be realized efficiently.

For business process engineering companies specialized in coordinating subcontractors, the grade of automation is the key success factor of their business. Otherwise, end customers and buyers expect high flexibility in processing business cases. Besides the high dynamic involved in executing processes, there is also high dynamic in process definition. In the ongoing development of processes, the interfaces to subcontractors, the database and the process itself will steadily change.

Such problem situations demand software systems that, on the one hand, have the possibility to work fully automatically on typical process situations, and, on the other hand, must provide the possibility to influence the process execution. At any time in the process execution, a redirection or data change should be possible for process managers.

This work will try to solve the problem by introducing a software system which pays special respect to this situation. The concept of process patterns will be implemented with the realization metaphors “routing slip” and “mailbox.” The responsibility of process execution will be passed to an open source workflow engine.

Only open source components will be used for this system. Therefore, the initial investment costs can be kept down. A role-based development model will be introduced, so that programmers possessing different know-how can be assigned to appropriate tasks.

Inhaltsverzeichnis

1	Einleitung	1
1.1	Problemstellung	1
1.2	Lösungsansatz	2
1.3	Struktur der Arbeit	3
2	Grundlagen	4
2.1	Grundlegende Begriffe	4
2.2	Methoden zur Geschäftsprozessmodellierung	5
2.2.1	Ereignisprozessketten mit ARIS	5
2.2.2	PROMET	7
2.2.3	UML	7
2.2.4	Bemerkungen zu den vorgestellten Methoden	8
2.3	Die Methode der exemplarischen Geschäftsprozess Modellierung	9
2.3.1	Überblick	9
2.3.2	Verwendete Notation	10
2.3.3	Kooperationsbild	11
2.3.4	Arbeitsplatzbild	12
2.3.5	Begriffsmodell und Glossar	12
2.3.6	Weitere Modelltypen	13
2.4	Der Werkzeug und Material Ansatz zur Softwarearchitektur	13
2.4.1	Überblick	13
2.4.2	Entwurfsmetaphern	14
2.5	Kooperative Arbeit durch Prozessmuster	15
2.5.1	Überblick	15
2.5.2	Prozessmuster	17
2.5.3	Laufzettel	17
2.5.4	Vorgangsmappen	18
2.5.5	Postkörbe	19
2.5.6	Zusammenspiel	19
2.6	Workflow Management Systeme	19
2.6.1	Einleitung	20
2.6.2	Begriffsdefinitionen	20
2.6.3	Entwicklung	22

2.6.4	WfMS und ablaufgesteuerte kooperative Arbeit	23
2.6.5	WfMS und unterstützte kooperative Arbeit	24
3	Aufgabenstellung	25
3.1	Zielszenario	25
3.2	Anforderungskatalog	26
4	Realisierung	28
4.1	Einführung in das begleitende Beispiel	28
4.1.1	Einführung	28
4.1.2	Aufgabenstellung	28
4.1.3	Zielsystem	31
4.2	Architektur des Gesamtsystems	31
4.2.1	Auswahl der Programmiersprache	31
4.2.2	Auswahl eines Architekturframeworks	32
4.2.3	Auswahl der Komponententechnologie	32
4.2.4	Auswahl weiterer Basiskomponenten	33
4.2.5	Beschreibung des Systems	33
4.3	Auswahl der Workflow Engine	35
4.3.1	Übersicht über Open Source Workflow Engines	35
4.3.2	Entscheidungskriterien	38
4.3.3	Entscheidungsmatrix	39
4.3.4	Entscheidung	39
4.4	Realisierung eines Arbeitsablaufes	39
4.4.1	Zusammenführung von ablaufsteuernder und unterstützender Sichtweise	40
4.4.2	Analyse mittels eGPM	42
4.4.3	Transformation in die Workflow Beschreibung	42
4.4.4	Anreicherung um die Klärungsstelle	45
4.5	Realisierung von Laufzettel und Vorgangsmappe	47
4.5.1	Der Laufzettel	47
4.5.2	Bearbeitung des Laufzettels	48
4.5.3	Die Vorgangsmappe	50
4.5.4	Verwendung von Laufzettel und Vorgangsmappe	50
4.6	Realisierung des Postkorbsystems	51
4.6.1	Design	51

4.6.2	Implementierung	52
4.6.3	Verteilung des Postkorbsystems	53
4.6.4	Verwendung des Postkorbsystems	54
4.7	Realisierung von automatischen Prozessschritten	55
4.7.1	Design	55
4.7.2	Implementierung	55
4.8	Realisierung einer Klärungsstelle	57
4.8.1	Benutzungsmodell	57
4.8.2	Design	59
4.8.3	Weitere Möglichkeiten	60
5	Musterkatalog zur Transformation eGPM – Laufzettel	61
6	Bewertung	64
6.1	Bewertung des Konzeptes	64
6.2	Bewertung der Realisierung	64
6.3	Beurteilung der Performanz	65
6.4	Bewertung des Transformationsvorganges	66
6.5	Beurteilung gegenüber der Anforderungen	67
6.6	Bewertung der Einsetzbarkeit	69
6.7	Persönliches Fazit	69
7	Ausblick und weiterführende Ideen	71
7.1	Laufzeitoptimierung	71
7.2	Rollen- und Benutzermodell	71
7.3	Erkennung des Prozessfortlaufes	71
7.4	Workflow Engine mit Möglichkeiten eines Prozessmusters	71

Abbildungsverzeichnis

Abbildung 1: eEPK Modell.....	6
Abbildung 2: Aufgabenkettendiagramm aus PROMET (Quelle: [Österle, 03]).....	7
Abbildung 3: Aktivitäten Diagramm (Quelle: [Borland, 03]).....	8
Abbildung 4: Gegenstände und Medien in eGPM	10
Abbildung 5: Kooperationen in eGPM	11
Abbildung 6: Begriffsmodell eGPM.....	12
Abbildung 7: Kooperation mit Laufzettel und Postkorb	19
Abbildung 8: Eignung von Prozessstypen für WfMS	23
Abbildung 9: Kooperationsbild ohne Interaktion mit Kunden.....	29
Abbildung 10: Kooperationsbild mit Interaktion mit Kunden.....	30
Abbildung 11: Systemübersicht	34
Abbildung 12: Von der Realität zur Workflow Beschreibung.....	41
Abbildung 13: Vom Arbeitsschritt zum Zustand	43
Abbildung 14: Prozesserzeugung.....	44
Abbildung 15: Klassendiagramm Deployment	46
Abbildung 16: Interfaces für den Laufzettel	48
Abbildung 17: Interface RoutingSlipService	49
Abbildung 18: Laufzettel Service Implementierung	49
Abbildung 19: Klassendiagramm Vorgangsmappen.....	50
Abbildung 20: Schnittstellen Postkorbsystem.....	51
Abbildung 21: Komposition im Postkorbsystem	52
Abbildung 22: Klassendiagramm Postkorbsystem	53
Abbildung 23: Klassendiagramm Verteilung.....	54
Abbildung 24: Sequenzdiagramm Tasks	56
Abbildung 25: Werkzeug für die Klärungsstelle.....	58
Abbildung 26: Klassendiagramm Klärungsstelle.....	59
Abbildung 27: Einfache Zustandsabbildung	62
Abbildung 28: Prozessstart	62
Abbildung 29: Prozessende.....	62
Abbildung 30: Fallunterscheidung	63
Abbildung 31: Batch Schnittstelle	63
Abbildung 32: Performanz Test	66

Tabellenverzeichnis

Tabelle 1: Übersicht über Open-Source Workflow Engines.....	36
Tabelle 2: Entscheidungsmatrix für die Workflow Engine.....	39
Tabelle 3: Möglichkeiten der Klärungsstelle	58
Tabelle 4: Erfüllung der Anforderungen.....	67

Beispielverzeichnis

Beispiel 1: Zustände in jPdl	43
Beispiel 2: Anfangszustand in jPdl	44
Beispiel 3: Konfiguration einer Prozessdefinition	47
Beispiel 4: Verwendung des Vorgangsmappen Service	51
Beispiel 5: Verwendung des Postkorbsystems.....	55
Beispiel 6: Implementierung eines Teilschrittes	56
Beispiel 7: Pseudo Code Performanz Test	65

1 Einleitung

1.1 Problemstellung

Schon seit einigen Jahren ist in der IT die Auslagerung von Geschäftsprozessen (Outsourcing) üblich. Nun sehen sich aber immer mehr Unternehmen mit der Situation konfrontiert, dass die Koordination der ausgelagerten Prozesse immens aufwendig ist. Eine Möglichkeit zur Realisierung automatisierter Koordination ist die Erstellung neuer Software Komponenten. Diese Vorgehensweise birgt wiederum das hohe Risiko der Eigenentwicklung, weshalb auch die Koordination von Prozessen immer öfter ausgelagert wird. Diese Aufgaben übernimmt dann ein „Business Process Management“-Dienstleister. Die beauftragenden Unternehmen konzentrieren sich auf ihre eigenen Kernprozesse und lassen diese vom externen Dienstleister mit den weiteren Prozessen abstimmen und synchronisieren.

Die Koordination von Teilprozessen ist gekennzeichnet durch eine Vielzahl beteiligter Schnittstellen und einer hohen Anzahl möglicher Fehlerquellen. Dabei lassen sich die Fehler entweder auf technische oder auf fachliche Ursachen zurückführen. Technische Fehler können durch den Ausfall von Infrastruktur Komponenten, wie Standleitungen, Servern, Diensten etc. entstehen. Fehler fachlicher Natur entstehen oft durch Falscheingaben, Tippfehlern oder auch durch ungenügende Prüfungen. Zusätzlich besteht bei der Koordination von verteilten Subprozessen die Gefahr, dass Fehler (vor allem fachlicher Natur) durch unklare Abstimmungen oder nicht beachteter Ausnahmesituationen entstehen können.

In der Praxis wird meist der Weg gewählt, dass der „übliche Ablauf“ des Prozesses voll automatisiert abläuft. Treten im Prozessablauf unerwartete Situationen oder auch Fehler auf, sollen diese Prozesse an Prozessmanager oder Sachbearbeiter weitergeleitet werden. Über Benutzeroberflächen kann nun flexibel Einfluss sowohl auf die Steuerung des Prozesses als auch auf die zugrunde liegenden Daten genommen werden.

Durch die Vielzahl angebundener Dienstleister und Unternehmen unterliegen besonders die Schnittstellen einem stetigen Wandel. Aber auch der Prozess selbst soll im Sinne einer hohen Kundenorientiertheit flexibel anpassbar bleiben.

Die Grundanforderung des automatisierten Ablaufes erfüllen Workflow Engines meist sehr zufrieden stellend. Aspekte wie Überwachung, Historisierung und Zuweisung an unterschiedliche Arbeitsplätze erfüllen diese ebenfalls. Allerdings erwarten Workflow Engines üblicherweise, dass der Prozess vollkommen ausspezifiziert wird, was aber nicht nur aufwendig ist, sondern in der Praxis oft auch gar nicht gefordert wird.

Zusätzlich ist die gänzlich generische Verwendung von Workflow Engines in anderen Softwarekomponenten äußerst komplex. Da Workflow Engines meist auf den Konzepten von Zustandsautomaten, Petri-Netzen und Graphen beruhen, setzt der Einsatz solcher Komponenten hohes theoretisches Grundlagenwissen voraus. Aus Kostengründen ist es daher wünschenswert, diese Komponenten mit ihrer hohen Komplexität nur in einem eingegrenzten Bereich des Gesamtsystems einzusetzen.

1.2 Lösungsansatz

Im Zuge dieser Arbeit wird ein Softwaresystem beschrieben, das eine automatische Abwicklung von Prozessen erlaubt, für die nicht alle möglichen Ausnahmesituationen modelliert sein müssen. Die Modellierung des typischen Ablaufes eines Prozesses soll also ausreichen, um diesen auch zu automatisieren. Dennoch muss das System die Flexibilität bieten, auf Sondersituation zu reagieren.

Ein besonderes Augenmerk wird auf die einfache Integrierbarkeit externer Schnittstellen gelegt. Da die Implementierung dieser Schnittstellen meist sehr zeitaufwendig ist, können durch eine effiziente Anbindung Kosten eingespart werden.

Zur Modellierung des typischen Ablaufes wird die exemplarische Geschäftsprozessmodellierung (kurz eGPM) vorgeschlagen. Diese Methode zur Prozessmodellierung stellt eine Alternative zu Vorgehensweisen wie ARIS, Promet oder der Modellierung mit UML dar. Dabei wird die Kooperation der beteiligten Akteure in den Mittelpunkt der Betrachtung gestellt. Die Erfahrung zeigt, dass diese Methode zur Prozessanalyse nur geringe Einarbeitungszeit benötigt und weniger hohe Ansprüche an das Abstraktionsvermögen der beteiligten Personen stellt.

Die so dargestellten exemplarischen Arbeitsabläufe können nicht direkt in Softwaremodelle übertragen werden, da sie weder umfassend noch detailliert genug sind. Die Übertragung der eGPM-Modelle in softwaretechnische Komponenten ist viel mehr ein kreativer Prozess. Als Teil der Arbeit entsteht auch ein Musterkatalog für diese Übertragung der eGPM-Modelle. Inwieweit sich dieser Prozess routinisieren und durch den Musterkatalog unterstützen lässt, ist ein wesentlicher Aspekt dieser Arbeit.

Zur Implementierung der Kooperation und Koordination des Ablaufes werden die Metaphern von Laufzettel und Postkorb verwendet. Der Laufzettel dient einerseits zur Dokumentation des Prozesses und andererseits zur Einflussnahme auf den Ablauf. Für die automatisierten Prozessschritte werden Postkörbe eingerichtet. Automaten holen ihre Aufgaben aus den Postkörben ab. So werden die automatisierten Komponenten vom Prozess entkoppelt. Für den menschlichen Eingriff in die Abläufe stehen so genannte Klärungsstellen zur Verfügung. Die Klärungsstelle ist ein Werkzeug dessen Benutzerschnittstelle dem Anwender die Möglichkeit gibt, den Ablauf beliebig zu verändern und auch wieder dem automatisierten Ablauf zuzuführen.

Die Laufzettel und Postkörbe bieten auch eine einfache und fachlich zu handhabende Schnittstelle. Die einzelnen Prozessschritte werden möglichst entkoppelt vom Gesamtprozess entwickelt und getestet. Damit soll die hohe Komplexität der Workflow Engine gekapselt werden. Entwickler, die Prozessschritte oder automatische Komponenten implementieren, benötigen keine Kenntnisse über Zustandsautomaten, Akteure, Referenzlinien und ähnliche Elemente von Workflow Engines.

Das Prozesssystem wickelt den modellierten (typischen) Ablauf des Prozesses automatisch ab und leitet ansonsten an die Klärungsstelle weiter. Auf diese Weise wird erreicht, dass die häufigen Abläufe automatisiert abgewickelt werden, ohne den hohen Implementierungs- und Modellierungsaufwand für alle Sonderfälle abbilden zu müssen.

Die Funktionalität von Laufzettel und Postkorb wird mithilfe einer Open-Source Workflow Engine bzw. anderen Open-Source Komponenten abgebildet. Diese

Komponenten werden ebenfalls die Aspekte Historisierung, Persistenz, Logging und andere Querschnittsfunktionen übernehmen.

Es wird Java 1.5 zum Einsatz kommen, um auch die neuen Möglichkeiten der Sprache aufzeigen und bewerten zu können. Die Implementierung von Laufzettel und Postkorb erfolgen als wieder verwendbare Komponenten. Der Werkzeug und Material Ansatz [Züllighoven et al, 04] wird als Basis der Entwicklung herangezogen. Es ist nahe liegend, dass deshalb auch JWAM 2.0 [JWAM, 05] als Architekturframework herangezogen wird.

Die Beschreibung der Vorgehensweise erfolgt anhand eines Beispiels aus einem Praxisprojekt der Telekommunikationsbranche.

1.3 Struktur der Arbeit

Nach der kurzen Einführung in die grundsätzliche Fragestellung der Arbeit in Kapitel 1 (Einleitung) werden in Kapitel 2 die theoretischen Grundlagen der Arbeit eingeführt. In diesem Zusammenhang werden auch die exemplarische Geschäftsprozessmodellierung und andere Ansätze der Prozessmodellierung beschrieben.

Im Kapitel 3 (Aufgabenstellung) werden die Ziele und Anforderungen formuliert.

Das Kapitel 4 (Realisierung) beginnt mit der Einführung in das begleitende Beispiel. Nach der Beschreibung der Gesamtarchitektur des Systems erfolgt die Auswahl einer Workflow Engine. Dazu werden die Produkte aus einer Marktübersicht anhand einer Entscheidungsmatrix gegen die Anforderungen gestellt. Anschließend beginnt die ausführliche Beschreibung der einzelnen Aspekte. Auf eine Trennung von Design und Implementierung wird im Hinblick auf die Idee der agilen Vorgehensweisen verzichtet. Es wird davon ausgegangen, dass die Darstellung des Designs bei der unmittelbaren Gegenüberstellung mit der Umsetzung deutlicher ausfällt.

Das Kapitel 5 (Musterkatalog zur Transformation eGPM – Laufzettel) beschreibt den Transformationsprozess von eGPM zur ausführbaren Prozessbeschreibung. Hier wird ebenfalls der Musterkatalog zur Transformation als wesentliches Ergebnis der Arbeit vorgestellt.

Im Kapitel 6 (Bewertung) muss sich die Arbeit rückblickend der kritischen Bewertung stellen. Die Bewertung wird einerseits die möglichst objektive Gegenüberstellung gegen die Anforderungen beinhalten und auch andererseits die persönlichen Eindrücke des Autors darlegen.

Zum Abschluss gibt Kapitel 7 (Ausblick und weiterführende Ideen) einen Überblick, welche Ideen im Rahmen der Arbeit noch entstanden sind, und wo sich Anknüpfungspunkte für weitere Arbeiten ergeben.

2 Grundlagen

Nach der kurzen Einführung in die Aufgabenstellung dieser Arbeit folgen in diesem Kapitel sowohl die Grundlagen der Geschäftsprozessmodellierung als auch die softwaretechnischen Grundlagen.

Als Einstieg in die Grundlagen werden die wichtigsten Methoden und Modelle zur Geschäftsprozessmodellierung vorgestellt. Im Kapitel 2.3 (Die Methode der exemplarischen Geschäftsprozess Modellierung) wird ein alternativer Ansatz zur Prozessmodellierung präsentiert. In Kapitel 2.4 (Der Werkzeug und Material Ansatz zur Softwarearchitektur) wird der WAM Ansatz genauer beschrieben, bevor das Kapitel 2.5 (Kooperative Arbeit durch Prozessmuster) das Konzept der Prozessmuster zur softwaretechnischen Realisierung von Geschäftsprozessen beschreibt, da sich beim Konzept der Prozessmuster auch zahlreiche Elemente aus WAM wieder finden. Den Abschluss der Grundlagen bildet das Kapitel 2.6 (Workflow Management Systeme) in dem die gängige Technik zur Realisierung von Geschäftsprozessen in Software vorgestellt wird.

2.1 Grundlegende Begriffe

Bevor in den folgenden Abschnitten auf die Prozessmodellierung eingegangen wird, beschäftigt sich dieser Abschnitt mit den wichtigsten Grundbegriffen.

Die zwei wesentlichsten Merkmale eines Unternehmens bilden die Aufbau- und die Ablauforganisation. Die Aufbauorganisation beschreibt die hierarchische Gliederung eines Unternehmens in organisatorische Einheiten. Die Ablauforganisation hingegen beschreibt die Arbeitsabläufe im Unternehmen. Zur Strukturierung der Ablauforganisation werden Geschäftsprozesse verwendet. In [Schmelzer, 04] wird der Begriff Geschäftsprozess folgendermaßen beschreiben:

„Geschäftsprozesse sind funktionsübergreifende Verkettungen wertschöpfender Aktivitäten, die von Kunden erwartete Leistungen erzeugen und deren Ergebnisse strategische Bedeutung für das Unternehmen haben. Sie können sich über das Unternehmen hinaus erstrecken und Aktivitäten von Kunden, Lieferanten und Partnern einbinden.“

[Schmelzer, 04]

Diese Definition drückt aus, dass ein Geschäftsprozess über die einzelnen Funktionen der Aufbauorganisation hinausgeht und für die Unternehmensziele von Bedeutung ist. Durch diese Einschränkung wird auch klar, dass nicht alle Arbeitsabläufe in einem Unternehmen auch Geschäftsprozesse sind.

In [Davenport, 92] werden noch weitere Bedingungen an einen Geschäftsprozess gestellt:

Ein Geschäftsprozess ist eine spezielle Anordnung von Aktivitäten, die eine Leistung für bestimmte Kunden oder Märkte erbringen. Er hat einen Beginn und ein Ende, klar definierte In- und Outputwerte und verläuft durch mehrere Bereiche.

[Davenport, 92]

Die klare Abgrenzung von Geschäftsprozessen und die Festlegung von In- und Outputwerten ist vor allem für die Koordination von Prozessen wichtig. In einem Unternehmen stellt sich die Herausforderung eine Vielzahl von Geschäftsprozessen so zu verbinden und zu koordinieren, sodass die Ablauforganisation in ihrer Gesamtheit die Erreichung der Unternehmensziele ermöglicht. Um die Organisation eines Unternehmens nach seinen Geschäftsprozessen zu ermöglichen, müssen diese identifiziert, analysiert, modelliert, organisiert, koordiniert und überwacht werden. Diese Tätigkeiten werden unter dem Begriff „Geschäftsprozessmanagement“ (engl. *Business Process Management*) zusammengefasst. Eine Aktivität des Geschäftsprozessmanagements ist die Geschäftsprozessmodellierung, die im engeren Sinne das Beschreiben und Gestalten von Prozessen umfasst. Im weiteren Sinne jedoch ist diese Aktivität nur in Zusammenhang mit einer Analyse der bestehenden Prozesse sinnvoll.

2.2 Methoden zur Geschäftsprozessmodellierung

Um teilautomatisierte Prozesse in einem Softwaresystem zu realisieren, muss der Prozess zuerst beschrieben werden. Neben den Möglichkeiten der formlosen Beschreibung von Prozessen durch Prosatexte, informelle Skizzen oder auch einfach durch implizites Wissen der beteiligten Personen haben sich auch formale und semiformale Methoden zur Prozessmodellierung etabliert. Diese Methoden werden im Begriff Geschäftsprozessmodellierung (engl. *Business Process Engineering*) zusammengefasst. Da sich die Geschäftsprozessmodellierung auf die Abläufe eines Unternehmens konzentriert, hat sich daraus im weiteren Verlauf die Unternehmensmodellierung entwickelt, die noch weitere Aspekte mit einbezieht. Der Begriff Unternehmensmodellierung umfasst neben der Geschäftsprozessmodellierung als Repräsentation der Ablauforganisation eines Unternehmens auf jeden Fall auch die Erfassung der Aufbauorganisation. In manchen Ansätzen umfasst die Unternehmensmodellierung zusätzlich noch die Datenmodellierung und / oder die IT-Systemmodellierung.

Seit den späten 80er Jahren haben sich mehrere Ansätze zur Unternehmensmodellierung entwickelt. Die Grundlage der meisten Ansätze bildet das Zachmann Rahmenwerk zur Unternehmensmodellierung (Zachmann Enterprise Architecture Framework) [Zachmann, 1987]. Durch die umfangreichen Anforderungen an die Unternehmensmodellierung bieten alle gängigen Methoden, über die Prozessmodellierung hinausgehende Modelle an. Da in dieser Arbeit aber vornehmlich auf den Aspekt der Prozessmodellierung eingegangen wird, wird dieser auch bei der in Folge beschriebenen Methoden fokussiert.

2.2.1 Ereignisprozessketten mit ARIS

Die Architektur integrierter Informationssysteme (kurz ARIS) ist am Institut für Wirtschaftsinformatik an der Universität des Saarlandes unter Leitung von A.-W. Scheer zur Modellierung von Informationssystemen entstanden [Scheer, 02]). Die IDS Scheer AG hat mit dem ARIS Toolset den Ansatz in der Praxis umgesetzt und perfektioniert.

Das ARIS Toolset ist mit über 33.000 Lizenzen das am weitesten verbreitete Werkzeug zur Unternehmensmodellierung und es ist somit auch anzunehmen, dass ARIS (vor allem in Europa) die am weitesten verbreitete Modellierungsmethode ist.

ARIS bietet einen sehr umfassenden Ansatz zur Modellierung. Das ARIS-Haus (als Metapher für den umfassenden Ansatz) beinhaltet die fünf Sichten der Daten-, Funktions- und Organisationssicht, sowie die Steuerungssicht und der Leistungssicht als das verbindende Element. Jede dieser Sichten wird auf den Modellebenen Fachkonzept, DV-Konzept und Implementierung betrachtet.

Die Modelle der Geschäftsprozesse finden sich als integrierende Elemente in der Steuerungssicht. Scheer empfiehlt zur Modellierung der Geschäftsprozesse das Modell der erweiterten Ereignisprozessketten (eEPK). Allerdings ist in den letzten Jahren auch eine Annäherung an die Modelle der UML wahrnehmbar.

Die eEPK kennt als Grundelemente Ereignisse, Funktionen und logische Verknüpfungen. Dabei müssen sich Ereignis und Funktion im Ablauf immer abwechseln. Ein Prozess beginnt und endet immer mit einem Ereignis. Den Namen erweiterte EPK hat das Modell durch das Hinzufügen der Modellelemente Informationsobjekte, Organisationseinheiten, Leistungsobjekte und Prozess-Schnittstellen erhalten. In Abbildung 1 ist die graphische Notation der eEPK dargestellt.

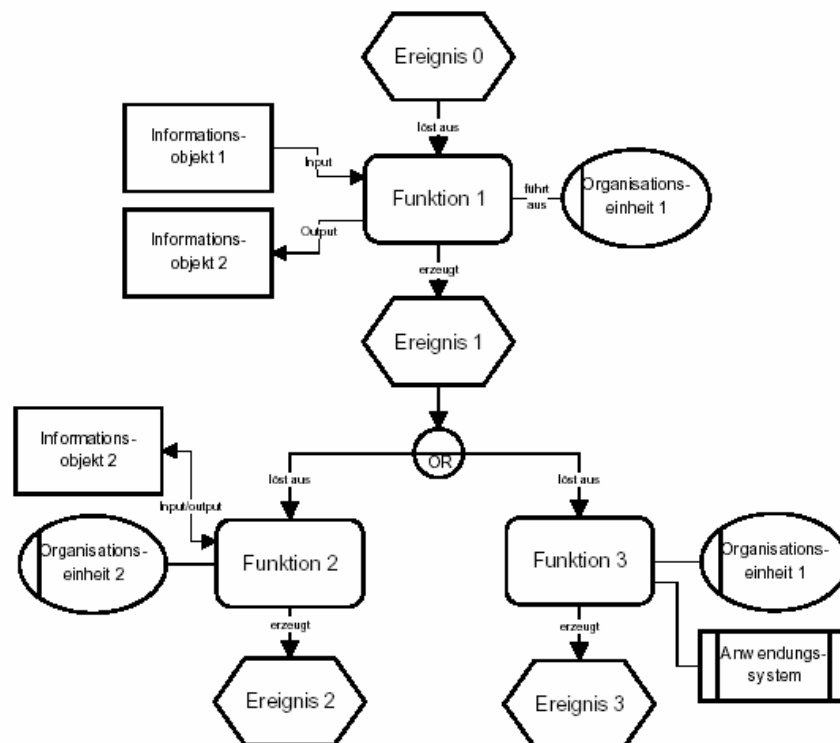


Abbildung 1: eEPK Modell

Die eEPK werden auch in vielen anderen Modellierungswerkzeugen in abgewandelter Form verwendet. So finden sich z.B. in Casewise Corporate Modeler [Casewise, 05], Popkins' Enterprise Architect [Popkins, 05] oder in Adonis von BOC [BOC, 05] die Ideen von eEPK's wieder. Manche Hersteller erweitern die Modelle soweit, um sie als ausführbare Workflows zu verwenden. Andere Hersteller hingegen weichen die formale Bedingung auf, dass Ereignis und Funktion aufeinander folgen müssen.

2.2.2 PROMET

Promet ist eine Abkürzung für PROzess METHode und wurde am Institut für Wirtschaftsinformatik an der Hochschule St. Gallen entwickelt [Österle, 03]. Die Methode wird hauptsächlich bei der prozessorientierten Einführung von ERP-Software angewendet [Wintersteiger, 02]. Zur Darstellung von Abläufen wird in Promet das Aufgabendendiagramm (AKD) verwendet. In Abbildung 2 ist ein Beispiel dargestellt.

Im AKD werden die Aufgaben im Geschäftsprozess durch Rechtecke dargestellt. Die durchgezogenen Verbindungen beschreiben die sequentielle Ausführung von Aufgaben. Gestrichelt verbundene Aufgaben werden gleichzeitig ausgeführt. Die einzelnen Aufgaben werden im Aufgabenverzeichnis genauer beschrieben. Die AKD konzentriert sich auf eine einfache Darstellungsweise und bietet keine Verzweigungen oder zeitliche Parameter an.

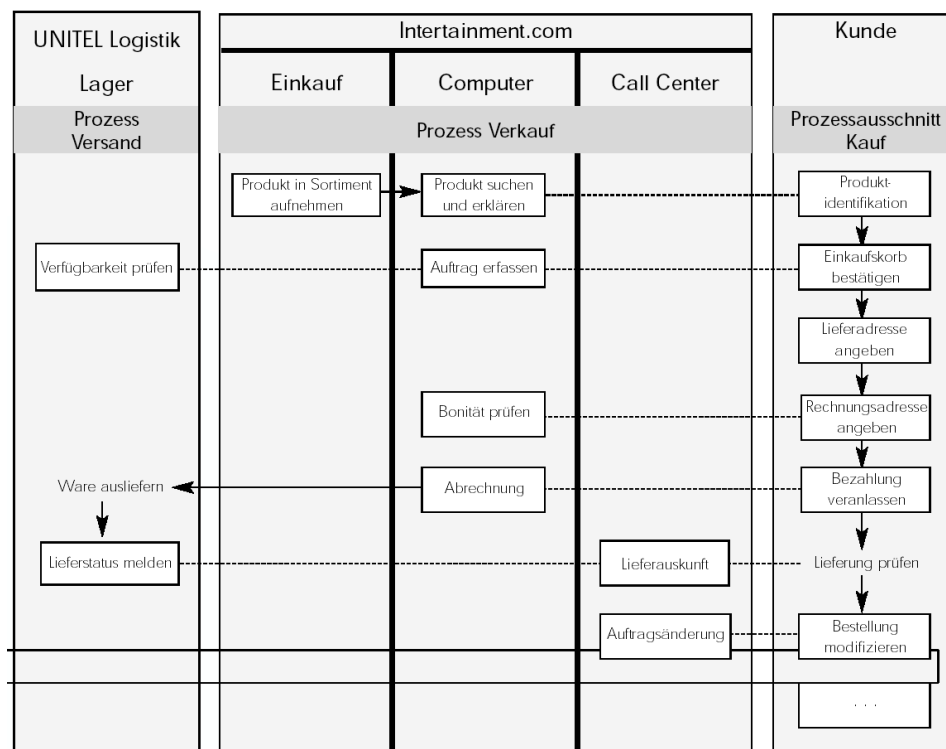


Abbildung 2: Aufgabendendiagramm aus PROMET (Quelle: [Österle, 03])

2.2.3 UML

Die Unified Modelling Language (UML) hat sich aus vielen verbreiteten Modellierungsmethoden der Softwareentwicklung der frühen 90er Jahre entwickelt. Die UML wurde durch ein Konsortium renommierter Softwarehersteller und wichtiger Persönlichkeiten in diesem Bereich standardisiert und wird nun von der Object Management Group (OMG) gewartet [UML, 04].

Die UML ist keine Methode zur Unternehmensmodellierung sondern ein Standard zur formalen Darstellung von (vornehmlich Software-) Modellen. Die UML bietet sechs Arten von Struktur Diagrammen, drei Arten von Ablaufdiagrammen und vier Arten von Interaktionsdiagrammen.

Obwohl in [MID, 03] ein Ansatz zur Einbindung verschiedener UML Modelle in die Unternehmensmodellierung vorgeschlagen wird, findet sich in anderen Quellen wie z.B. [Wintersteiger, 02] die Meinung, dass nur das Aktivitätendiagramm (Activity Diagramm) für die Geschäftsprozessmodellierung gut geeignet ist.

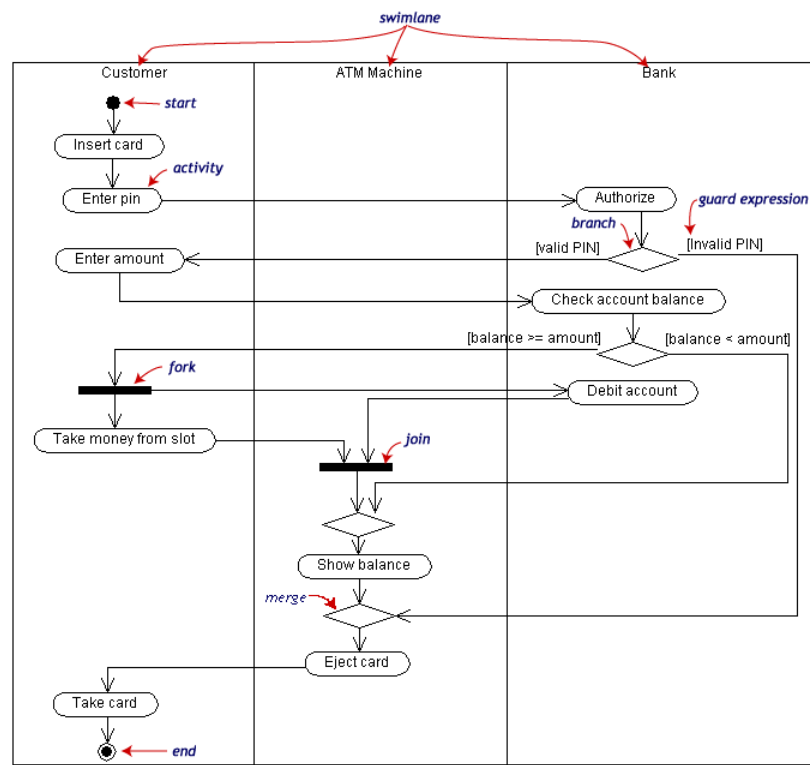


Abbildung 3: Aktivitäten Diagramm (Quelle: [Borland, 03])

Abbildung 3 zeigt ein Aktivitätendiagramm. Ein Diagramm besteht im Wesentlichen aus Aktivitätszuständen, die mittels Zustandsübergängen verbunden sind. Zusätzlich können Kontrollstrukturen, wie Verzweigungen, Prozessteilung (fork) oder Prozessvereinigung (join) eingebracht werden. Die Aktivitäten können durch die „Swimlanes“ an Rollen zugewiesen werden.

Die UML bietet gegenüber anderen Methoden den Vorteil, dass durch die Standardisierung ein sehr großes Angebot an Werkzeugen zur Erstellung der Diagramme verfügbar ist. Darunter finden sich auch zahlreiche frei verfügbare Produkte.

2.2.4 Bemerkungen zu den vorgestellten Methoden

Alle drei vorgestellten Modellierungsmethoden konzentrieren sich auf die Aufgaben und deren zeitliche Abfolge. Zusätzlich bieten auch alle die Möglichkeit, die ausführenden Rollen bzw. Stellen im Modell darzustellen. Wobei hier die eEPK eine sehr unübersichtliche Darstellungsform anbietet. Um die Aufgaben einer Rolle/Stelle aus dem Modell ablesen zu können, müssen alle Organisationseinheiten des Modells durchsucht werden. Bei Promet und UML wird dahingegen nicht dargestellt, mit welchen Daten, Materialien, Werkzeugen und Systemen die Aufgaben zu erfüllen sind.

Ebenfalls allen Methoden gemein ist der Wunsch nach einer möglichst starken Formalisierung. Dabei bietet Promet zwar nicht so viele syntaktische Möglichkeiten, gibt aber dennoch einen sehr strengen Rahmen vor. Trotz dieses Anspruches nach

Formalismus treffen alle Methoden keine Aussage über die Granularität der Aufgaben. Hinter einer Aufgabe kann sich eine komplexe Kooperation zwischen unterschiedlichen Personen verbergen oder auch nur sehr einfacher Arbeitsschritt.

Dennoch führt die Einführung dieses Formalismus zu stark abstrahierten Modellen. Dadurch wird aber die Verständlichkeit und Ausdruckstärke der Modelle verringert. In den Modellen finden sich die typischen Denkmuster aus der Informatik und Mathematik, wie z.B. Zustände, Funktionen, Graphen, etc. Erfahrungsgemäß sind im Anwendungskontext der Fachanwender solche Denkmuster weniger verbreitet. Dadurch begründet sich oft auch die Schwierigkeiten im unmittelbaren Verständnis solcher Modelle und in weiterer Folge die Ablehnung dagegen.

Im folgenden Abschnitt wird daher die Methode der exemplarischen Geschäftsprozess Modellierung vorgestellt. Diese Methode verzichtet bewusst auf diesen Formalismus und wird daher dem konzeptionellen Anspruch von Prozessmodellierung besser gerecht. Erst in einem weiteren Arbeitsschritt werden die Prozessmodelle in ausführbare und formale Beschreibungen überführt.

2.3 Die Methode der exemplarischen Geschäftsprozess Modellierung

2.3.1 Überblick

Neben den bereits beschriebenen Methoden hat sich mit der Methode der exemplarischen Geschäftsprozessmodellierung (kurze eGPM) auch eine alternative Herangehensweisen an die Geschäftsprozessmodellierung entwickelt. Natürlich kann ein Geschäftsprozess mit jeder Methode analysiert, modelliert und dokumentiert werden. Allerdings unterscheiden sich die Methoden dabei hinsichtlich des Schwerpunktes der Betrachtung, ihrer Komplexität und auch aufgrund deren formaler Konzeption.

Die üblichen Ansätze zur Geschäftsprozessmodellierung gehen davon aus, einen Geschäftsprozess abstrahiert vom konkreten Vorfall in einer Beschreibungssprache darzustellen. Als Piktogramme der Darstellungsform werden einfache geometrische Formen wie Rechtecke, Rauten, Kästen usw. verwendet. Die Piktogramme dienen zur Darstellung von Zuständen, Funktionen oder Entscheidungsknoten. In manchen Darstellungsformen werden auch Stellen bzw. Rollen und Datenobjekte gezeigt.

Im Gegensatz dazu verfolgt die exemplarische Geschäftsprozessmodellierung die Idee, eine konkrete Ausprägung eines Geschäftsvorfalles in einer leicht lesbaren und gut interpretierbaren Form darzustellen. Es wird also nicht versucht, den Prozess abstrakt in allen möglichen Abläufen darzustellen. Daher werden in eGPM zur Beschreibung eines Geschäftsprozesses mehrere Bilder verwendet, die jeweils einen Ablauf des Prozesses beschreiben. Soll beispielsweise der Prozess der Antragstellung für eine Kreditkarte modelliert werden, so werden zumindest zwei Bilder entstehen, wobei eines die Genehmigung und ein anderes die Ablehnung des Antrages darstellt. Somit ist die eGPM eine Szenario basierte Modellierungsmethode.

Die Methode eGPM versteht sich nicht nur als graphische Notation, sondern vielmehr als eine Herangehensweise zur Prozessanalyse und –modellierung. Aufgrund der einfachen Verständlichkeit bietet eGPM die Möglichkeit, bereits nach einer kurzen Einführung die

Prozessanalyse gemeinsam mit den Fachanwendern in der eGPM Darstellungsform durchzuführen.

Wie bereits weiter oben erläutert, wird in ein Bild der eGPM immer ein konkreter Ablauf des Prozesses beschrieben. Durch diese Beschreibung in der Ausprägungsebene ergibt sich eine der wesentlichen Besonderheiten der eGPM, da in einer Ausprägungsebene keine Fallunterscheidungen getroffen werden können. So wird in einem Bild der eGPM beispielsweise die Ablehnung des Kreditkartenantrages beschrieben. In dieser Darstellung wird die Genehmigung des Antrages nicht betrachtet. Somit muss auch die Fallunterscheidung gar nicht getroffen werden, da diese ja schon in der Beschreibung des eGPM Bildes enthalten ist. Daher sieht die eGPM auch keine Möglichkeit zur Darstellung von Fallunterscheidungen vor.

Gemeinsam mit den Fachanwendern werden die wichtigsten Prozesse bestimmt und die für den Anwendungszweck bedeutendsten Ausprägungen dargestellt. Damit wird auch schon vorweggenommen, dass die Modelle von eGPM keine lückenlose Beschreibung eines Prozesses darstellen, sondern einen Eindruck der üblichen und wichtigen Abläufe bieten.

Da eGPM im Vergleich zu den vorher vorgestellten Methoden sehr wenig verbreitet ist und in dieser Arbeit weiter verfolgt wird, wird diese Methode in Folge ausführlicher beschrieben als die übrigen Methoden zur GPM.

2.3.2 Verwendete Notation

Für die Visualisierung von eGPM Modellen werden ausdrucksstarke Piktogramme verwendet. Diese werden mit Hilfe von vier unterschiedlichen Pfeilformen verbunden.

In Abbildung 4 (Gegenstände und Medien in eGPM) sind die wichtigsten Piktogramme von eGPM dargestellt. Die zentrale Rolle nehmen die Akteure ein, die später Gegenstände bearbeiten oder auch mit Hilfe von Medien austauschen. Ein Medium dient also immer der Abstimmung zwischen Akteuren oder zur Weitergabe von Gegenständen.

Eine besondere Rolle nimmt hier der Brief ein. Ein Brief ist einerseits ein Medium zur Kommunikation aber auch ein Gegenstand zum Informationsaustausch.

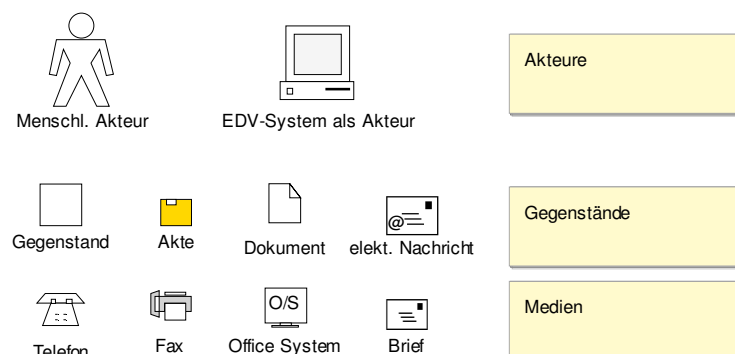


Abbildung 4: Gegenstände und Medien in eGPM

2.3.3 Kooperationsbild

Die Inhalte dieses Abschnittes stammen im Wesentlichen aus [Breitling, 03].

Das Kooperationsbild ist der zentrale Modelltyp von eGPM. Die ersten Arbeiten zu Kooperationsbildern stammen aus [Krabel, 98]. Ein Kooperationsbild stellt die Kooperation von Akteuren mittels Medien und Gegenständen dar und bringt diese in Beziehung zueinander. Die Einzelschritte werden durch eine Nummerierung in eine Reihenfolge Beziehung gebracht.

In Abbildung 5 (Kooperationen in eGPM) sind die Kooperationsformen eines Kooperationsbildes abschließend dargestellt. Der Text beschreibt die Bedeutung des Pfeils. In Kooperationsbildern beschreiben die Texte dann die Art der Kooperation genauer. In Schritt 1 informiert ein Akteur einen anderen mit Hilfe eines Gegenstandes oder auch über ein Medium. Die Frage ob der Gegenstand oder das Medium dargestellt werden, ist abhängig davon, welcher Aspekt betont werden soll. In Schritt 2 wird der Gegenstand an den zweiten Akteur weitergegeben. Diese Weitergabe kann auch in Form einer Kopie erfolgen. Sie drückt aus, dass der nächste Akteur die Verantwortung über den Gegenstand übernimmt. In Schritt 3 wird ein Gegenstand von einem Akteur bearbeitet. In Schritt 4 wird ebenfalls ein Gegenstand bearbeitet, jedoch kann zusätzlich noch ein Medium oder ein Hilfsgegenstand angegeben werden. In Schritt 5 ist der Aufruf eines Geschäftsfalles dargestellt. In eGPM ist ein Geschäftsfall eine Aggregation aus mehreren Kooperationsbildern.

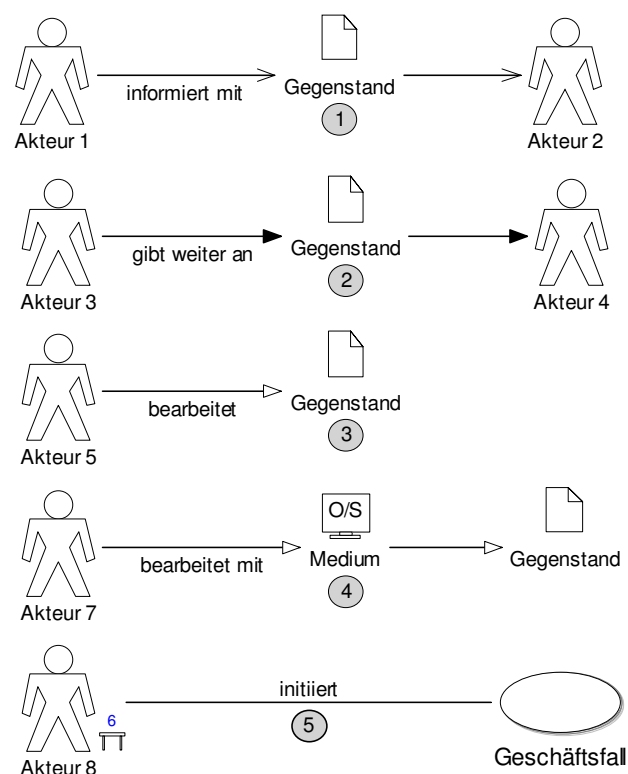


Abbildung 5: Kooperationen in eGPM

In einem vollständigen Kooperationsbild muss noch der Geschäftsvorfall beschrieben werden. Die Beschreibung kann in Form einer Notiz im Kooperationsbild beschrieben werden. Beispiele für Kooperationsbilder finden sich noch später in dieser Arbeit.

Durch die Darstellung des kleinen Tisches mit der Nummer 6 beim Akteur 8, wird gezeigt, dass der Arbeitsschritt 6 durch ein Arbeitsplatzbild dargestellt wird.

2.3.4 Arbeitsplatzbild

Das Arbeitsplatzbild stellt die Arbeitsschritte eines Akteurs auf seinem Arbeitsplatz dar. Es beinhaltet keine Kooperationen mit anderen Akteuren und stellt daher eine Sonderform des Kooperationsbildes dar. Auf Arbeitsplatzbildern ist immer nur ein Akteur dargestellt und somit werden keine Pfeiltypen für „übergeben an“ und „informieren mit“ benötigt.

Die Untergliederung in Kooperationsbilder und Arbeitsplatzbilder bietet den großen Vorteil, dass auch bei der Modellierung getrennt vorgegangen werden kann. Um Kooperationsbilder zu modellieren, müssen alle Beteiligten daran mitarbeiten. Beschreibt man in diesem Kontext auch die Arbeit an einzelnen Arbeitsplätzen, käme es zu unnötigen Verzögerungen in der Modellierung. Die Gestaltung des Arbeitsplatzbildes kann dann mit den beteiligten Mitarbeitern im kleineren Kreis durchgeführt werden.

2.3.5 Begriffsmodell und Glossar

Das Begriffsmodell dient zur Beschreibung der verwendeten Gegenstände. Im Gegensatz zu einem klassischen Glossar bietet das Begriffsmodell nicht nur eine textuelle Beschreibung von Begriffen sondern setzt diese auch in Beziehung zueinander.

Abbildung 6 (Begriffsmodell eGPM) zeigt ein Begriffsmodell. Mit den Symbolen für Werkzeug, Automat, fachl. Service und Behälter finden sich die wichtigsten Entwurfsmetaphern von WAM wieder (siehe dazu auch Kapitel 2.4). Ein Behälter kann andere Gegenstände wie z.B. Dokumente aggregieren. Eine „ist-ein“-Relation kann wie im Beispiel zwischen Dokument und Formular dargestellt werden. Werkzeuge, Automaten und fachliche Services können mit anderen Gegenständen arbeiten. Das ist in der Abbildung mit dem Pfeil zwischen Werkzeug und Formular dargestellt. Durch den grauen Kasten wird eine thematische Aggregation vorgenommen.

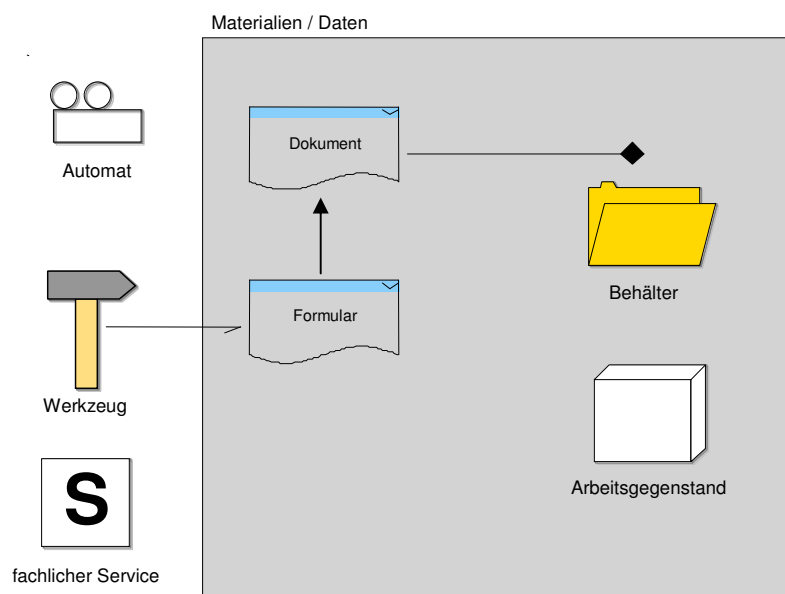


Abbildung 6: Begriffsmodell eGPM

Zu jedem Piktogramm im Begriffsmodell kann ein Text für den Glossareintrag hinterlegt werden. Aus diesen Einträgen kann automatisiert ein Glossar generiert werden.

2.3.6 Weitere Modelltypen

Die eGPM umfasst noch weitere Modelltypen, die aber in diesem Kontext nur kurz beschrieben werden, da sie für diese Arbeit nicht benötigt werden.

IT-Interaktionsbilder und Masken

In IT-Interaktionsbildern wird die schrittweise Interaktion des Benutzers mit einem IT-System dargestellt. Diese Darstellungsform dient einerseits in einer sehr frühen Phase vor dem Prototyping zur ersten Darstellung des Systems aber auch zur Dokumentation von Software und die schrittweise Anleitung zur Abwicklung von Geschäftsvorfällen [Luginsland, 04].

Anwendungsfalldiagramme

Die Anwendungsfalldiagramme in eGPM sind sehr ähnlich zu den Use-Case Diagrammen der UML. In eGPM kommt ihnen zusätzlich die Bedeutung zu, dass sie Kooperationsbilder zu Geschäftsfällen aggregieren.

Arbeitsumgebungsmodelle

Die Arbeitsumgebungsmodelle dienen zur Darstellung der Aufbauorganisation in modellierten Arbeitsumfeldern. Sie werden zur Definition von Rollenmodellen verwendet.

2.4 Der Werkzeug und Material Ansatz zur Softwarearchitektur

Nachdem in den vorhergehenden Kapiteln die Geschäftsprozesse und ihre Modellierung im Vordergrund standen, rückt nun die Entwicklung von Software in den Mittelpunkt. Zur softwaretechnischen Umsetzung dieser Arbeit wird der Werkzeug und Material Ansatz – in Folge kurz WAM-Ansatz (Werkzeug, Automat, Material) verwendet. Der WAM-Ansatz bietet eine integrierte Sichtweise auf die Softwareentwicklung von der fachlichen Modellierung über die Architektur bis hin zur Implementierung. Die Begriffe und Konzepte aus WAM werden dann auch im Kapitel 2.5 (Kooperative Arbeit durch Prozessmuster) verwendet und umgesetzt.

2.4.1 Überblick

Der WAM-Ansatz wird ausführlich in [Züllighoven et al, 98] in deutscher Sprache und in erweiterter Form in [Züllighoven et al, 04] in englischer Sprache beschrieben. Die Inhalte des Kapitels „2.4 Der Werkzeug und Material Ansatz“ stammen sowohl aus [Züllighoven et al, 98] als auch aus [Züllighoven et al, 04].

Der WAM-Ansatz fördert und fordert die Anwendungsorientierung in der Softwareentwicklung. Der Benutzer der Software als selbstverantwortlich handelnder Mensch wird in den Mittelpunkt gestellt. Diese Sichtweise beeinflusst nicht nur die Gestaltung der Funktionalität und der Benutzerschnittstellen eines Softwaresystems sondern auch den Entwicklungs- und Implementierungsprozess, bei dem der Benutzer des Systems eine zentrale Rolle spielt.

Hier sollen aber nur die Auswirkungen des WAM-Ansatzes auf die Architektur eines Softwaresystemes erläutert werden, da diese unmittelbaren Einfluss auf das spätere Design der Komponenten haben wird.

Der WAM-Ansatz verwendet Entwurfsmetaphern zur Veranschaulichung softwaretechnischer Grundkonzepte. Die wichtigsten Entwurfsmetaphern sind:

1. Material
2. Werkzeug
3. Arbeitsumgebung
4. Automat
5. Behälter

Der WAM-Ansatz bietet zur Realisierung der Entwurfsmetaphern die Entwurfsmuster (Design-Patterns) an. Viele dieser Entwurfsmuster werden durch das Framework JWAM [JWAM, 05] realisiert.

In den folgenden Kapiteln werden die grundlegendsten Entwurfsmetaphern näher beschreiben.

2.4.2 Entwurfsmetaphern

Material

Unter dem Begriff „Material“ wird ein Gegenstand verstanden, der Teil des Arbeitsergebnisses wird. Materialien werden mit Werkzeugen und Automaten bearbeitet. In anderen Ansätzen werden dafür die Begriffe „Datenobjekt“ oder „Datenkapsel“ verwendet. Diese Begriffe sind für Materialien nach WAM nicht geeignet, da diese auch eine fachlich motivierte Schnittstelle anbieten. Somit beinhalten Materialien sehr wohl Kenntnis über ihre fachliche Verwendung. Im Zuge der Bearbeitung von Materialien können auch mehrere gleichzeitig verwendet und miteinander kombiniert werden.

In der Modellierung bieten sich die Arbeitsgegenstände des Benutzers bereits als Kandidaten für mögliche Materialien an.

Die Materialien können vom Benutzer nur mittels Werkzeuge sondiert und verändert werden.

Werkzeug

Ein Werkzeug ist ein Gegenstand mit dem ein oder mehrere Materialien bearbeitet oder auch sondiert werden können. Unter dem Begriff „Sondieren“ wird in diesem Zusammenhang das Abfragen und Auslesen des Zustandes bzw. von Werten aus Materialien verstanden. Ein Werkzeug muss die Darstellung eines Materials mit der Bearbeitung kombinieren.

Ein Werkzeug ist zur Bearbeitung bestimmter Materialien geeignet. Im Umkehrschluss muss für die Bearbeitung eines Materials immer auch das passende Werkzeug gewählt werden.

Im späteren Anwendungssystem sind die Werkzeuge jene Komponenten, die die Benutzerschnittstelle zur Verfügung stellen. Durch die Gestaltung der Werkzeuge wird auch die Handhabung des Softwaresystems bestimmt.

Auch hier wird wieder die zentrale Rolle des Benutzers im WAM-Ansatz deutlich. Nicht die Software führt den Benutzer durch das System sondern der Benutzer bearbeitet seine Materialien in flexibler Art und Weise und verwendet dafür als Werkzeuge die Softwarekomponenten.

Arbeitsumgebung

Bisher stehen dem Benutzer Werkzeuge zur Verfügung, mit denen er seine Materialien bearbeiten kann. Es stellt sich nun natürlich die Frage, wie die Werkzeuge und Materialien aufbewahrt, aktiviert und verwendet werden können.

Diese Möglichkeit bietet die Arbeitsumgebung. Im Sinne eines Arbeitsplatzes ist das jener Ort, an dem der Mitarbeiter seine Werkzeuge und Materialien findet. Er kann Werkzeuge benutzen und damit Materialien bearbeiten.

Es kann hierbei auch noch zwischen individuellen und gemeinsamen Arbeitsplätzen unterschieden werden. Individuelle Arbeitsplätze sind geschützt vor anderen Benutzern und repräsentieren die persönliche Arbeitsumgebung. Gemeinsame Arbeitsplätze können als Ort der Interaktion und der gemeinsamen Arbeit verstanden werden.

Der bereits weitgehend etablierte Desktop von Betriebssystemen mit graphischen Benutzeroberflächen ist eine Umsetzung der Entwurfsmetapher „Arbeitsumgebung“.

Automat

Automaten sind Arbeitsmittel, die ihre Arbeit selbsttätig im Hintergrund verrichten. Sie sollen dem Benutzer lästige Routinetätigkeiten abnehmen. Die Arbeit von Automaten wirkt sich üblicherweise wieder auf Materialien aus. Es können neue Materialien entstehen, bestehende verändert werden oder auch gelöscht werden. Durch die Veränderung der Materialien wird dem Benutzer auch die Auswirkung der Automaten deutlich. Oft erzeugt der Automat auch ein Material, das einen Ergebnisbericht seiner Arbeit repräsentiert.

Automaten werden üblicherweise über eine Benutzerschnittstelle konfiguriert und überwacht.

2.5 Kooperative Arbeit durch Prozessmuster

In den vorhergehenden Kapiteln wurden einerseits die Möglichkeiten beschrieben, Prozesse darzustellen und andererseits mit WAM ein Ansatz zur Software Entwicklung vorgestellt. Nun stellt sich die Frage, wie Geschäftsprozesse mit dem WAM Ansatz in Software abgebildet werden können. Daher werden in diesem Kapitel das Konzept des Prozessmusters und die Metaphern von Laufzettel und Postkorb eingeführt.

2.5.1 Überblick

Bei der Abwicklung von nicht trivialen Geschäftsprozessen sind immer mehrere Personen bzw. Akteure beteiligt. Um die notwendigen Aufgaben eines Prozesses zu lösen müssen

die Akteure interagieren. Da diese Interaktion auf die Ziele abgestimmt ist, spricht man auch von Kooperation. [Gryczan, 95] definiert Kooperation folgendermaßen:

Kooperation

Als Kooperation wird das Zusammenwirken (mindestens) zweier Personen bezeichnet, die ihre Handlungen auf Grundlage wechselseitig abgestimmter Ziele und Pläne ausführen.

Der Begriff „Kooperation“ bezieht sich nicht auf spezielle Lebenssituationen oder Aufgabentypen. Daher wird der Begriff zur kooperativen Arbeit spezialisiert, der die Kooperation in Arbeitssituationen zur Erreichung der Unternehmensziele und Aufgaben beschreibt.

In [Gryczan, 95] wird die Unterscheidung zwischen Ablaufsteuerung menschlicher Arbeit und Unterstützung menschlicher Arbeit beschrieben. Dabei werden die Begriffe folgendermaßen definiert:

Ablaufsteuerung

Ablaufsteuerung bedeutet, Operationen zu implementieren, die menschliches Arbeitshandeln regeln und kontrollieren. Die Kontrolle über den Ablauf der Arbeitsschritte des Menschen liegt bei der Maschine. Generell wird bei der Automatisierung das Ziel verfolgt, menschliches Arbeitshandeln durch Maschinen zu ersetzen oder bis auf notwendige Dateneingabe (Input) zu reduzieren. Dazu werden Pläne und Vorschriften verwendet, die im Idealfall durch Algorithmen auf Maschinen implementiert werden.

Unterstützung

„Unterstützung menschlicher Arbeit bedeutet, dass Benutzer von Softwaresystemen als Experten ihres Arbeitsgebietes verstanden werden und dass in ihrem Arbeitshandeln Computer als Arbeitsmittel eingesetzt werden. Ein charakteristisches Merkmal dieser Sichtweise ist, dass die Initiative bei der Computerverwendung vom Benutzer ausgeht und dass die Kontrolle über den Ablauf der Arbeitsschritte beim Benutzer liegt.“

[Gryczan, 95]

Wird ein unterstützendes Softwaresystem eingesetzt, so obliegt es dem Benutzer, sich mit seinen Mitmenschen über die Koordination seiner Arbeit zu verständigen. Daraus entwickelt sich auch der Begriff der situierten Koordination [Gryczan, 95]:

Situierte Koordination ist die wechselseitige Abstimmung über die Reihenfolge und Zuständigkeit von Tätigkeiten bei kooperativer Arbeit.

Im Sinne eines unterstützenden Softwaresystems muss natürlich gerade auch diese Kooperation und Koordination zwischen Benutzern unterstützt werden. Es muss jedoch beachtet werden, dass dabei keine Ablaufsteuerung entsteht, sondern die Entscheidung immer der Benutzer treffen kann. Wird der Benutzer in seinen Entscheidungen eingeschränkt, kann nicht von einem unterstützenden Softwaresystem ausgegangen werden.

Neben den üblichen IT gestützten Kommunikationswegen, wie E-Mail, Instant Message, Foren, Voice-over-IP kann ein Softwaresystem für diese Aufgabe die Prozessmuster anbieten.

2.5.2 Prozessmuster

Es stellt sich nun die Frage, wie die Kooperation von einem Softwaresystem unterstützt werden kann, ohne eine Ablaufsteuerung vorzugeben. In [Gryczan, 95] wird der Ansatz gewählt, dass die kooperative Arbeit selbst zum veränderlichen Gegenstand wird. Gryczan fasst das folgendermaßen zusammen:

„Die zu koordinierende kooperative Arbeit wird selbst wieder zum Gegenstand der Arbeit. Durch dieses Vorgehen wird es möglich, anhand eines Arbeitsgegenstandes kooperative Arbeit durch die Handelnden selbst zu kontrollieren und transparent zu gestalten.“

Mit Hilfe von Prozessmustern wird die kooperative Arbeit veranschaulicht. Da das Prozessmuster nach der Entwurfsmetapher Material aus WAM gestaltet ist, wird auch schon die Veränderbarkeit eines Musters ausgedrückt.

Begriff Prozessmuster

„Ein Prozessmuster ist ein gemeinsames Material zur Vergegenständlichung eines kooperativen Arbeitsprozesses. Durch das Prozessmuster werden Verantwortlichkeiten von Personen oder Rollenträgern und Tätigkeiten in einem kooperativen Arbeitsprozess festgelegt. Ein Prozessmuster besteht aus der Angabe der Abhängigkeiten von und zwischen Tätigkeiten, die bei der kooperativen Arbeit zu erledigen sind und dazu notwendigen Dokumenten.“

[Gryczan, 95]

Ein Prozessmuster ist also ein Material, das in Form einer Schablone den Arbeitsablauf beschreibt. Dabei werden die einzelnen Tätigkeiten nicht ausformuliert, sondern nur beschreibend aufgeführt. Wie die Tätigkeiten zu verrichten sind, ist immer im Verantwortungsbereich des Bearbeiters und wird nicht vom Prozessmuster behandelt.

Das Prozessmuster kann während der Bearbeitung jederzeit verändert oder ergänzt werden. Es können Bearbeitungsschritte eingefügt oder auch übersprungen werden. Erst durch diese Eigenschaften eignen sich Prozessmuster zur Unterstützung kooperativer Arbeit.

Da die Prozessmuster ein unterstützendes Element in der Bearbeitung sind, bietet sich ein Katalog von Prozessmustern an. Das ist eine Sammlung von Prozessmustern, die der Benutzer auswählen und an seine Bedürfnisse anpassen kann. Damit können gängige Kooperationsabläufe erfasst und katalogisiert werden, ohne jedoch eine Einschränkung darzustellen. In [Gryczan, 95] wird dafür der Begriff Prozessmusterrepertoire verwendet.

2.5.3 Laufzettel

Der Begriff Prozessmuster ist durch seine konzeptionelle Bedeutung geprägt, entspricht aber in seiner Aussagekraft nicht den Entwurfsmetaphern von WAM. Darüber hinaus kann aus dem Begriff Prozessmuster noch keine geeignete Darstellungsform abgeleitet werden.

Eine mögliche und oft verwendete Realisierungsform eines Prozessmusters ist der Laufzettel [Züllighoven et al, 04]. Ein Laufzettel ist ein gängiger Begriff und gerade in der Sachbearbeitung ein weit verbreitetes Konzept zur Ablaufsteuerung.

In einer softwaretechnischen Realisierung ist vor allem eine einfache Veränderbarkeit des Laufzettels wichtig. Es müssen neue Empfänger eingeschoben werden können und auch vorgegebene Stellen entfernt oder umsortiert werden können.

Neben der Beeinflussung des Ablaufes erfüllt der Laufzettel eine weitere wichtige Aufgabe. Auf dem Laufzettel sind auch die durchlaufenen Stellen eingetragen und somit wird auch die historische Nachvollziehbarkeit eines Ablaufes unterstützt.

Ein Laufzettel sollte zumindest folgende Daten beinhalten:

- Titel
- Beschreibung
- Erstellungsdatum / -zeit

Eine Zeile eines Laufzettels beschreibt einen Schritt im Ablauf und sollte daher folgende Daten enthalten:

- Empfänger
- Tätigkeit
- Tätigkeitsbeschreibung
- Muss / Kann
- maximale Liegezeit
- Erledigungsdatum / -zeit
- Kommentar zur Bearbeitung

2.5.4 Vorgangsmappen

Im vorigen Abschnitt wurde beschrieben, wie ein Prozessmuster durch einen Laufzettel dargestellt werden kann. Nun müssen in einem Arbeitsablauf auch andere Materialien bearbeitet werden. Dazu müssen die Daten aber auch geeignet abgelegt werden. Eine Möglichkeit dazu bietet die Vorgangsmappe. Dieses Konzept stammt wieder aus der klassischen Bürosachbearbeitung.

In einer Vorgangsmappe werden die Materialien zu einem Vorgang gesammelt. Der Laufzettel zur Vorgangsmappe wird auf den Deckel der Mappe geheftet. In einer softwaretechnischen Realisierung soll der Laufzettel also ohne das Öffnen der Mappe einsehbar sein.

Eine Vorgangsmappe beinhaltet somit alle Materialien zu einem Ablauf. Wenn eine Vorgangsmappe von einem Benutzer zu einem anderen weitergegeben wird, ist somit auch der gesamte Arbeitsablauf übergeben worden.

2.5.5 Postkörbe

Um Arbeitsabläufe auszutauschen ist die Weitergabe der Vorgangsmappen erforderlich. Der Austausch der Vorgangsmappen kann über ein Postkorbsystem funktionieren. Dabei verteilt ein Versandautomat die Vorgangsmappen mit Hilfe der Laufzettel auf die Postkörbe des Anwendungssystems und holt diese auch wieder bei den Postkörben ab.

Daher sollte in einer kooperativen Arbeitsumgebung jeder Postkorb eine Ausgangs- und ein Eingangsfach haben. Die Postkörbe stellen die einzige von außen zugreifbare Ressource eines Arbeitsplatzes dar.

In Abbildung 7 (Kooperation mit Laufzettel und Postkorb) wird die Kooperation über Postkörbe veranschaulicht. Die Abbildung stammt aus [Züllighoven et al, 04].

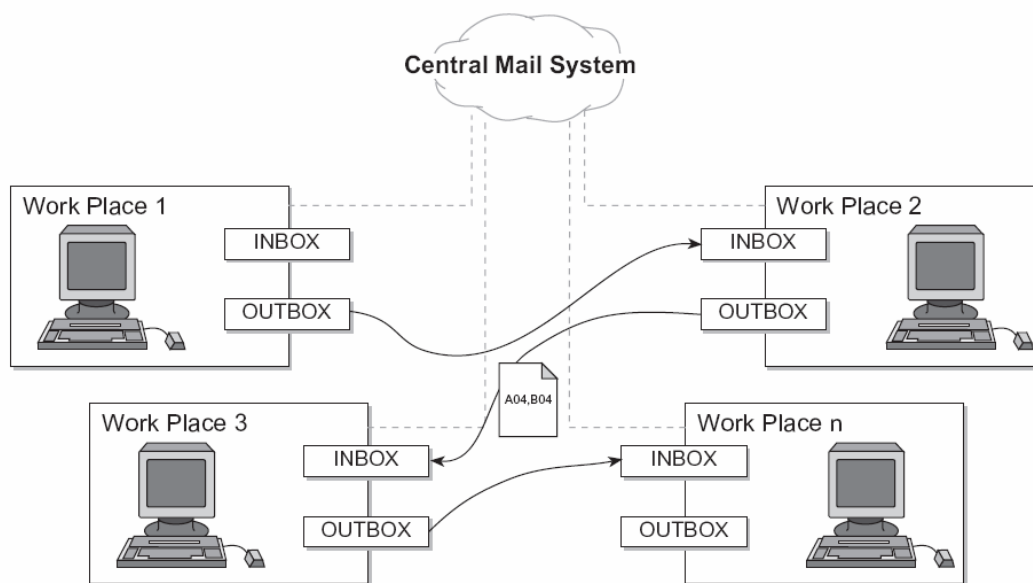


Abbildung 7: Kooperation mit Laufzettel und Postkorb

2.5.6 Zusammenspiel

Der Benutzer kann sich nun aus dem Prozessmusterrepertoire ein Prozessmuster auswählen, dass ihn in seiner kooperativen Arbeitssituation unterstützt. Dargestellt wird dieses Prozessmuster als Laufzettel, der an eine Vorgangsmappe geheftet ist. Der Benutzer kann über seine Werkzeuge den Laufzettel ausfüllen und verändern. Bearbeitete Materialien werden in der Vorgangsmappe abgelegt. Die Vorgangsmappen werden vom Versandautomaten über die Postkörbe verteilt. Zur Bearbeitung werden die Vorgangsmappen aus dem Eingangsfach bezogen und dann bearbeitet. Nach Abschluss der Tätigkeit laut Tätigkeitsbeschreibung am Laufzettel werden die Vorgangsmappen über das Ausgangsfach des Postkorbes an den Versandautomaten übergeben. Der Versandautomat leitet die Vorgangsmappe anhand des Laufzettels an den nächsten Postkorb weiter.

2.6 Workflow Management Systeme

Workflow Management Systeme sind neben der klassischen Individualentwicklung die am weitesten verbreitete Möglichkeit zur Unterstützung von Geschäftsprozessen. Darüber hinaus wurde bereits in der Einleitung zu dieser Arbeit beschrieben, dass zur Realisierung

der Aufgabenstellung eine Open Source Workflow Engine zum Einsatz kommen soll. Vor diesem Hintergrund bietet das folgende Kapitel einen Einstieg in die Entwicklung und die Begriffe von Workflow Management Systemen.

2.6.1 Einleitung

Im Jahr 1984 entstand das Forschungsgebiet über Computer gestützte kooperative Arbeit (engl.: Computer Supported Cooperative Work, CSCW). In [Teufel et al, 95] wird der Begriff folgendermaßen definiert:

„Computer Supported Cooperative Work (CSCW) ist die Bezeichnung des Forschungsgebietes, welches auf interdisziplinärer Basis untersucht, wie Individuen in Arbeitsgruppen oder Teams zusammenarbeiten und wie sie dabei durch Informations- und Kommunikationstechnologie unterstützt werden können.“

Im Rahmen dieser Forschungen sind auch die Workflow Management Systeme (kurz WfMS) entstanden, die eine Realisierung von CSCW darstellen. Die Workflow Management Coalition (WfMC) [WfMC, 05] beschäftigt sich seit 1993 mit der Begriffsbestimmung und der Standardisierung von WfMS.

2.6.2 Begriffsdefinitionen

Um für die weitere Diskussion Klarheit über die Verwendung einzelner Begriffe zu haben, beschäftigt sich dieser Abschnitt mit der Definition der verwendeten Begriffe und vergleicht dabei auch unterschiedliche Quellen.

Workflow

Für den Begriff Workflow gibt es grundsätzlich zwei verbreitete Interpretationsformen. Eine Sichtweise ist vom technischen Standpunkt geprägt, während die zweite vor allem die organisatorische bzw. ablauftechnische Sicht betrachtet.

In [Vogler, 96] findet sich ein Beispiel für diese Sicht:

„Ein Workflow besteht aus mehreren Aktivitäten, die miteinander verbunden sind und vom Aufgabenträger nach festgelegten Regeln ausgeführt werden.“

[Vogler, 96]

Aus technischer Sicht definiert die WfMC den Begriff Workflow folgendermaßen:

“The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.”

[WfMC Glossary, 99]

Während Vogler die Aggregation von Aktivitäten in den Vordergrund stellt, sieht die WfMC vor allem den Aspekt der Automatisierung eines Prozesses im Vordergrund. Im weiteren Verlauf dieser Arbeit wird der Begriff Workflow im Sinne der WfMC verwendet.

Workflow Management

Der Begriff Workflow Management umfasst die Tätigkeiten, die in Zusammenhang mit Workflows notwendig sind. Damit umfasst es Handlungen wie das Organisieren, Planen, Entscheiden, Kontrollieren, Steuern und Führen von Workflows. [Jablonski, 97]. Workflow Management bezeichnet im Gegensatz zu den anderen hier aufgeführten Begriffen nichts Gegenständliches sondern ein Tätigkeitsfeld.

Workflow Management System

Bei der Definition des Begriffes Workflow Management System herrscht weitgehend Einigkeit.

Jablonski bietet eine sehr umfassende Definition für WfMS. Bemerkenswert daran ist vor allem die ausdrückliche Betonung der Kontrollfunktion eines WfMS:

“Workflow management software is a proactive computer system which manages the flow of work among participants, according to a defined procedure consisting of a number of tasks. It coordinates user and system participants, together with the appropriate data resources, which may be accessible directly by the system or off-line, to achieve defined objectives by set deadlines. The coordination involves passing tasks from participant to participant in correct sequence, ensuring that all fulfil their required contributions, taking default actions when necessary.”

[Jablonski, 95]

Die WfMC definiert den Begriff vor allem über die Ausführung des Workflows und die damit verbundene Interpretation der Workflow Beschreibung. Die Wichtigkeit von Interoperabilität drückt die WfMC auch durch die explizite Erwähnung von Drittsystemen aus:

“A system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications.”

[WfMC Glossary, 99]

Workflow Engine

Als Workflow Engine wird von der WfMC die Softwarekomponente bezeichnet, die den Workflow zur Laufzeit ausführt.

“A software service or "engine" that provides the run time execution environment for a process instance.”

[WfMC Glossary, 99]

In der deutschen Übersetzung der WfMC wird der Begriff fälschlicherweise mit dem Workflow Management System gleichgesetzt. Die Unterscheidung ist wichtig und sinnvoll, da ein WfMS mehr Funktionalität und damit höhere Komplexität aufweist als eine Workflow Engine. Es zeigt sich auch, dass sich zahlreiche Workflow Engines fälschlicherweise als WfMS bezeichnen, obwohl sie keine Komponenten zum

Modellieren und Überwachen der Workflows anbieten bzw. auch die direkte Einbindung von Drittsystemen nicht unterstützen [Jablonski, 95].

2.6.3 Entwicklung

In [Schulze, 96] wird die Entwicklung von WfMS in 4 Generationen bis in die Mitte der 90er Jahre beschrieben:

Generation 1: „hard-wired“

Diese Systeme sind hauptsächlich aus einem Katalog von Funktionen und Funktionsbausteinen entstanden, die aufgrund wachsender Anforderungen zu flexibel zusammenstellbaren Vorgängen erweitert wurden. Die Ablauflogik der Vorgänge war in der Anwendung codiert und war nicht als eigenständige Beschreibung verfügbar.

Generation 2: explizites Vorgangsmodell

Aufgrund der steigenden Komplexität der Systeme erster Generation wurde die Definition von Vorgängen in einem Metamodell notwendig. Dadurch wurde es möglich, Vorgänge auf abstrakter Ebene zu beschreiben. Somit konnte auch die Ausführungseinheit von der Beschreibung getrennt werden. Daraus sind auch die Workflow Engines gewachsen, die die Möglichkeit bieten, eine abstrakte Beschreibung auszuführen.

Generation 3: Datenbankeinsatz

Durch die Verwendung von Datenbanken in WfMS konnte eine Datenintegration mit anderen Systemen erreicht werden. Damit wurde der erste Schritt zu Integration von WfMS in andere Anwendungssysteme gemacht.

Generation 4: Integration

Mittlerweile gehen alle Bestrebung hin zu einer nahtlosen Integration von WfMS in die Anwendungs- und Kommunikationssysteme. Daher beschreibt die WfMC auch mit ihrem Referenzmodell zu WfMS eine Menge von Schnittstellen, die der Kommunikation zwischen einzelnen Systemen dienen.

In Anlehnung an [Muehlen, 04] kann eine neue Generation von WfMS definiert werden:

Generation 5: Komponentenorientierung

WfMS werden immer mehr zu Komponenten von größeren Softwaresystemen (z.B. ERP-Systeme oder Applikationsserver). Somit verlieren WfMS als eigenständige Anwendungen immer mehr an Bedeutung. Durch die Realisierung von WfMS als Komponenten größerer Systeme werden diese auch stärker mit unternehmensexternen Applikationen im Supply Chain Management integriert.

2.6.4 WfMS und ablaufgesteuerte kooperative Arbeit

Zur Wiederholung noch einmal die Begriffsdefinition für Workflow nach WfMC:

“The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.”

[WfMC Glossary, 99]

Nach dieser Definition automatisiert ein WfMS einen Geschäftsprozess. Auch nach Vogler erfüllen die Aufgabenträger ihre Aufgaben nach festgelegten Regeln. Jablonski geht in seiner Definition von WfMS von der Überprüfbarkeit der erbrachten Tätigkeit durch das System aus. Dadurch wird die ablaufsteuernde Sichtweise von Workflow Systemen deutlich. Die menschliche Interaktion mit dem System ist streng definierbar und prüfbar.

Diese Aussagen decken sich auch mit der Einschätzung aus [Galler, 97], die in Abbildung 8 (Eignung von Prozesstypen für WfMS) dargestellt ist. Dabei werden Routineprozesse mit Routineaufgaben als bestens geeignet für WfMS dargestellt. Die in Abbildung 8 gezeigte Klassifikation von Prozessen wird in [Picot, 95] vorgestellt. Das deckt sich auch mit der gängigen Meinung. Weiters sieht Galler aber auch bei Regelprozessen, die zwar bestimmbar aber nicht determiniert sind, ebenfalls eine Eignung für WfMS.

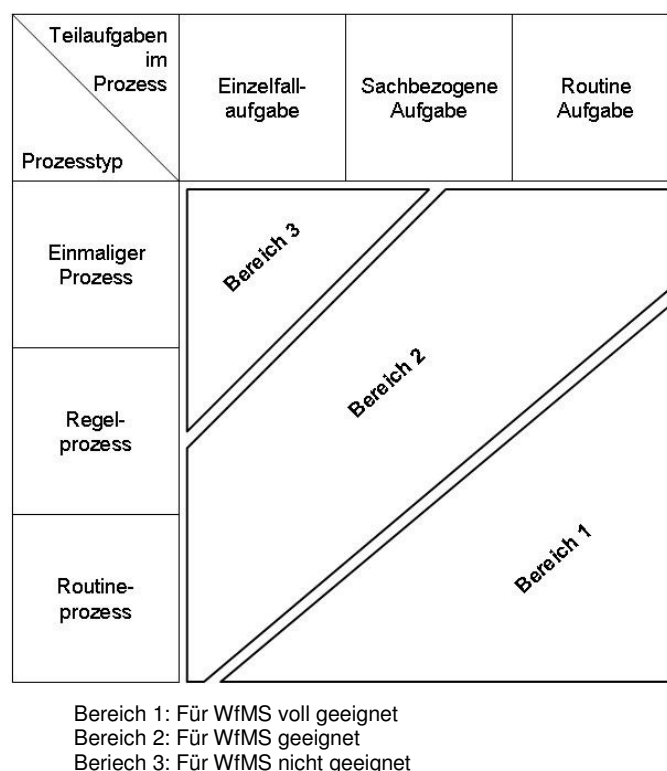


Abbildung 8: Eignung von Prozesstypen für WfMS

Auch in [Schwab, 96] wird von vielen positiven Erfahrungsberichten beim Einsatz von WfMS für stark formalisierbare Vorgänge berichtet.

Die weiter oben angegebenen Quellen beschreiben, dass WfMS für die ablaufgesteuerte kooperative Arbeit gut geeignet sind. Es ist allerdings nicht diskutiert worden, inwieweit eine starke Formalisierung von Vorgängen überhaupt sinnvoll ist. Eine Alternative dazu wird im Rahmen dieser Arbeit beschrieben.

2.6.5 WfMS und unterstützte kooperative Arbeit

Nachdem die Eignung von WfMS zur ablaufsteuernden kooperativen Arbeit gezeigt wurde, stellt sich nun die Frage, wie die Unterstützung kooperativer Arbeit erfolgen kann.

In [Galler, 97] wird davon ausgegangen, dass sich auch Regelprozesse, die durch Mitarbeiter beeinflusst werden, für WfMS eignen (vgl. auch Abbildung 8: Eignung von Prozesstypen für WfMS). Im weiteren Verlauf wird aber auch beschrieben, dass der Übergang zu den Workgroup Computing (auch Groupware)-Systemen für diese Prozesstypen fließend ist.

Über mangelnde Unterstützung von schwach formalisierten Vorgängen durch WfMS berichtet auch [Schwab, 96]. In diesem Zusammenhang wird auf die Problematik verwiesen, dass WfMS Systeme in ganz unterschiedlichen Einsatzgebieten verwendet werden und daher ausgesprochen flexibel sein müssen.

[Gryczan, 95] kommt zum ernüchternden Ergebnis, dass WfMS nur die ablaufsteuernde Sichtweise unterstützen:

„Workflow-Management Systeme sind eine Ausprägung der ablaufsteuernden Sichtweise mit dem Ziel, die Zusammenarbeit nach vordefinierten Regeln zu steuern und zu kontrollieren.“

Es kann somit also festgehalten werden, dass WfMS nur bedingt für die Unterstützung kooperativer Arbeit geeignet sind. Im Rahmen dieser Arbeit soll auch versucht werden, eine Workflow Engine zur Unterstützung kooperativer Arbeit einzusetzen.

3 Aufgabenstellung

Nach den Grundlagen des vorherigen Kapitels wird die Aufgabenstellung in diesem Kapitel losgelöst vom konkreten Beispiel beschrieben. Zusätzlich dazu wird auch ein Anforderungskatalog zusammengestellt, um später auch Bewertungskriterien für die Lösung zur Verfügung zu haben.

3.1 Zielszenario

Im Zielszenario können die in der Arbeit beschriebenen Softwarekomponenten – eingebettet in die beschriebene Vorgehensweise – zur Realisierung von teilautomatisierten Prozessen verwendet werden.

Definition: Teilautomatisierte Prozesse

Unter dem Begriff „teilautomatisierte Prozesse“ werden hier Abläufe verstanden, die vorrangig durch das Zusammenspiel unterschiedlicher Softwaresysteme abgewickelt werden können. Es kann jedoch jederzeit während der Verarbeitung notwendig werden, dass aufgrund von Sondersituationen oder Fehlern ein menschlicher Eingriff notwendig wird. Es sind auch fest definierte Schritte im Ablauf denkbar, die durch einen menschlichen Bearbeiter ausgeführt werden müssen. Es wird aber angenommen, der Großteil des Arbeitsflusses automatisiert passieren soll.

Die Implementierung eines solchen Prozesses soll im Zielszenario mit der Modellierung des Ablaufes mittels eGPM starten. Die Modellierung soll alle automatisch ablaufenden Geschäftsvorfälle exemplarisch darstellen. In einem kreativen – aber durch einen Musterkatalog unterstützten Prozess – werden die eGPM Modelle in Laufzettel überführt, die dann wieder in einem manuellen Transformationsschritt in die Definitionssprache der Workflow Engine übersetzt werden.

Durch die Modellierung aller zu automatisierenden Prozesse müssen im Workflow-Model üblicherweise auch einige, wenn auch einfache, Verzweigungen eingebaut werden. Die Verzweigungsentscheidungen werden auf Basis der Schnittstellen der Workflow Engine implementiert und müssen auch gekapselt testbar sein.

Die notwendigen Schnittstellenimplementierungen erfolgen dann getrennt von der Umsetzung des Arbeitsablaufes als eigenständige Softwarekomponenten, die sich nur auf ein Postkorbsystem und die Laufzettel stützen. Die Schnittstellen werden auch losgelöst vom Gesamtprozess getestet. Es kann zwischen 2 Arten von Schnittstellen unterschieden werden:

a) Einzelaufruf

Die Schnittstelle wird für jeden Geschäftsvorfall einzeln aufgerufen. Das sind üblicherweise Online-Schnittstellen wie z.B. Web-Services.

b) Automaten

Die Schnittstellen sind aktive Komponenten und holen mehrere, gesammelte Geschäftsvorfälle aus dem Postkorbsystem ab. Batch Schnittstellen mit Datei Austausch sind Beispiele für diesen Typ.

Bei der Bearbeitung können die Schnittstellen aufgrund der Laufzettel einen Fehlerzustand rückmelden. Diese Fehlerzustände führen dann zur Übermittlung des Ablaufes an eine Klärungsstelle.

Tritt nun im Produktionsumfeld ein nicht modellierter Vorfall oder Zustand ein, so werden die betroffenen Prozesse an eine Klärungsstelle geleitet. Die Klärungsstelle kann aufgrund der Laufzettel den Ablauf rekonstruieren und ohne Kenntnis der Workflow Engine den Ablauf über das Postkorbsystem an beliebiger Stelle in den automatisierten Ablauf rückführen.

3.2 Anforderungskatalog

Um eine Qualitätssicherung bereits während der Entwicklung gewährleisten zu können, werden hier Anforderungen und Qualitätskriterien sowohl an die zu integrierenden als auch an die zu entwickelnden Softwarekomponenten eingeführt.

Die Reihenfolge hat keine Aussage über deren Priorität. Jede der folgenden Anforderungen wird als wichtig erachtet.

Einfachheit der Modellierung

Die Modellierung des Arbeitsablaufes für die Workflow Engine soll in einer möglichst einfach zu beherrschenden Beschreibungssprache erfolgen. Eine Unterstützung durch eine graphische Benutzeroberfläche ist wünschenswert.

Wartungsfreundlichkeit

Eine Änderung des Arbeitsablaufes soll minimale Auswirkungen auf das Gesamtsystem haben. Im besten Fall sollte die Änderung des modellierten Ablaufes in der Beschreibungssprache genügen. Es muss auch beachtet werden, wie neue Ablaufbeschreibungen in das System eingespielt werden können. Dabei ist die Frage zu beachten, wie die zurzeit laufenden Prozesse weitergeführt werden.

Rollentrennung

Bei der Umsetzung eines Prozesses im Zielsystem ist auf eine Rollentrennung und damit verbunden auch auf unterschiedliches Qualifikationsniveau zu achten. Beispielsweise ist es von Vorteil, wenn die Entwicklung von Schnittstellen ohne Kenntnis der Workflow Engine möglich ist. Andererseits soll auch die Realisierung der Abläufe möglich sein, ohne die verwendeten Schnittstellen zu Drittsystemen kennen zu müssen.

Einfache Schnittstellen

Die Anbindung externer Schnittstellen an das Ablaufsystem und auch die Entwicklung von graphischen Benutzeroberflächen erfolgt über die Schnittstellen des Postkorbsystems sowie der Laufzettel und Vorgangsmappen. Es ist darauf zu achten, dass in diesen Schnittstellen keine Aspekte des Workflow Systems auftauchen und diese auch in ihrer Komplexität einfach handhabbar bleiben.

Kapselung der Workflow Engine

Die Workflow Engine sollte aus Sicht des späteren Anwendungssystems streng gekapselt werden. Bei der Modellierung des Arbeitsablaufes ist das explizit nicht gefordert, da ja genau hier die Stärken der Workflow Engine zum Einsatz kommen sollen.

Kapselung externer Schnittstellen

Die externen Schnittstellen sollen ausschließlich über Interfaces an das Ablaufsystem gebunden werden. Es ist auf eine beiderseits möglichst lose Kopplung zu achten. Vor allem aber muss das Ablaufsystem ohne Kenntnis der externen Schnittstellen arbeiten können.

Testbarkeit

Es ist darauf zu achten, dass die individuell zu entwickelnden Teile des Systems gekapselt und automatisiert testbar sind. Das ist eine wesentliche Voraussetzung für die verteilte Entwicklung und die Rollenteilung in der Entwicklung.

Serviceorientierung

Die Dienstleistungen des Softwaresystems sollen als fachliche Services [Züllighoven et al, 04] bereitgestellt werden.

Komponentenorientierung

Die Einzelteile des Systems sollen als lose gekoppelte Komponenten gestaltet werden, um eine Integration in andere Anwendungen zu erleichtern. Eine Diskussion über die Definition und die Anforderungen an Komponenten wird hier verzichtet. Für weitere Details sei auf [Szyperski, 02] verwiesen.

Historisierung

Für einen Arbeitsablauf sollen alle durchlaufenen Stationen mit eventuellen (Fehler-) Meldungen aufgezeichnet und gespeichert werden. In vielen Systemen wird auch ein Änderungsprotokoll der Datensätze gefordert. Dies muss aber vom Persistenz bzw. Datenservice des Datenmodells geleistet werden. Das Ablaufsystem kann sinnvoller Weise nur die durchlaufenen Stationen protokollieren.

Logging

Die Aktionen des Ablaufsystems sollen auch über einen Logging Mechanismus protokolliert werden. Diese Anforderung grenzt sich von der Historisierung dahingehend ab, dass bei der Historisierung Daten zum Ablauf gespeichert werden, während beim Logging die Aktionen des implementierten Systems protokolliert werden.

Überwachung

Es soll jederzeit möglich sein, die laufenden Prozesse einzusehen und zu überwachen. Im Gegensatz zu einer Klärungsstelle, sollen bei der Überwachung auch die Abläufe, die sich in automatisierten Arbeitsschritten befinden, eingesehen werden können.

4 Realisierung

Nachdem die Grundlagen zu dieser Arbeit und auch die zu lösende Aufgabenstellung ausführlich beschrieben ist, wendet sich das folgende Kapitel der Realisierung zu. Nach der Beschreibung eines begleitenden Beispiels wird im Kapitel 4.2 (Architektur des Gesamtsystems) das Gesamtsystem und die zentralen Komponenten beschrieben. Bevor auf die entwickelten Teile des Systems eingegangen wird, beschäftigt sich das Kapitel 4.3 (Auswahl der Workflow Engine) mit der Evaluierung, Bewertung und Auswahl einer Open Source Workflow Engine. Das Kapitel 4.4 (Realisierung eines Arbeitsablaufes) sollte besonders hervorgehoben werden, da hier die Zusammenführung des Prozessmuster-Konzeptes und der Realisierung mittels Workflow Engine beschrieben wird.

4.1 Einführung in das begleitende Beispiel

4.1.1 Einführung

Die Aufgabenstellung für diese Arbeit stammt aus einem konkreten Projektumfeld heraus. Es ist nahe liegend, dass einzelne Teile dieses Projekts als Beispiele zur Darstellung der Konzepte dienen.

Das Projekt stammt aus dem Bereich der Telekommunikation. Ein namhafter DSL Anbieter in Deutschland will die Aufgabenbereiche Call-Center, First Level Support und Koordination der Subunternehmer auslagern, um sich auf seine Kernkompetenz der Telekommunikationsdienstleistung konzentrieren zu können. Für den Dienstleister stellt sich nun die Aufgabe, diese Prozesskoordination möglichst vollautomatisiert umzusetzen.

4.1.2 Aufgabenstellung

Im Rahmen der Integration sollen insgesamt sechs Dienstleister koordiniert werden. Die meisten davon werden über Batch Schnittstellen mit CSV Dateien angebunden. Bisher bietet nur ein Dienstleister die Möglichkeit über, WebServices zu kommunizieren.

Repräsentativ für alle Prozesse wird der Bestellprozess für ein Produkt dargestellt. Der Kunde bestellt das Produkt bei einem Vertriebsdienstleister. Dabei wird entweder vom Kunden oder von Mitarbeitern des Dienstleisters ein Antrag ausgefüllt. Die Anträge werden über eine Batch Schnittstelle an das Prozesssystem übermittelt. Danach erhält der Kunde eine Auftragseingangsbestätigung. Das Prozesssystem führt eine Prüfung der Antragsdaten durch. Treten dabei Fehler auf, werden diese von einem Sachbearbeiter in Zusammenarbeit mit dem Kunden geklärt. Ist der Fehler behoben kann ein Vertrag im System angelegt werden und der Rechnungsdienstleister wird über den neuen Kunden informiert. Die Übermittlung erfolgt täglich im Batch Verfahren. Der Hardware Dienstleister wird ebenfalls informiert und dieser veranlasst die Auslieferung der notwendigen Geräte. Am Ende des Bestellprozesses erhält der Kunde die Auftragsbestätigung mit der Angabe des Liefertermins für die Hardware.

In Abbildung 9 und Abbildung 10 ist der Bestellprozess in Kooperationsbildern dargestellt. Die Schnittstellen sind zwar ähnlich aber dennoch für die Produkte teilweise unterschiedlich.

In Abbildung 10 (Kooperationsbild mit Interaktion mit Kunden) ist der Bestellprozess für dasselbe Produkt wie in Abbildung 9 dargestellt. Im Unterschied zur vorherigen Abbildung hat der Kunde hier aber Daten nicht korrekt angegeben, sodass ein Mitarbeiter beim Kunden nachfragen muss. Das ist eine typische Aufgabe für die Klärungsstelle. In diesem Fall wurde aber der Fall explizit modelliert, da ja die Klärung aufgrund fehlender Daten einen Standardvorfall für diesen Prozess darstellt.

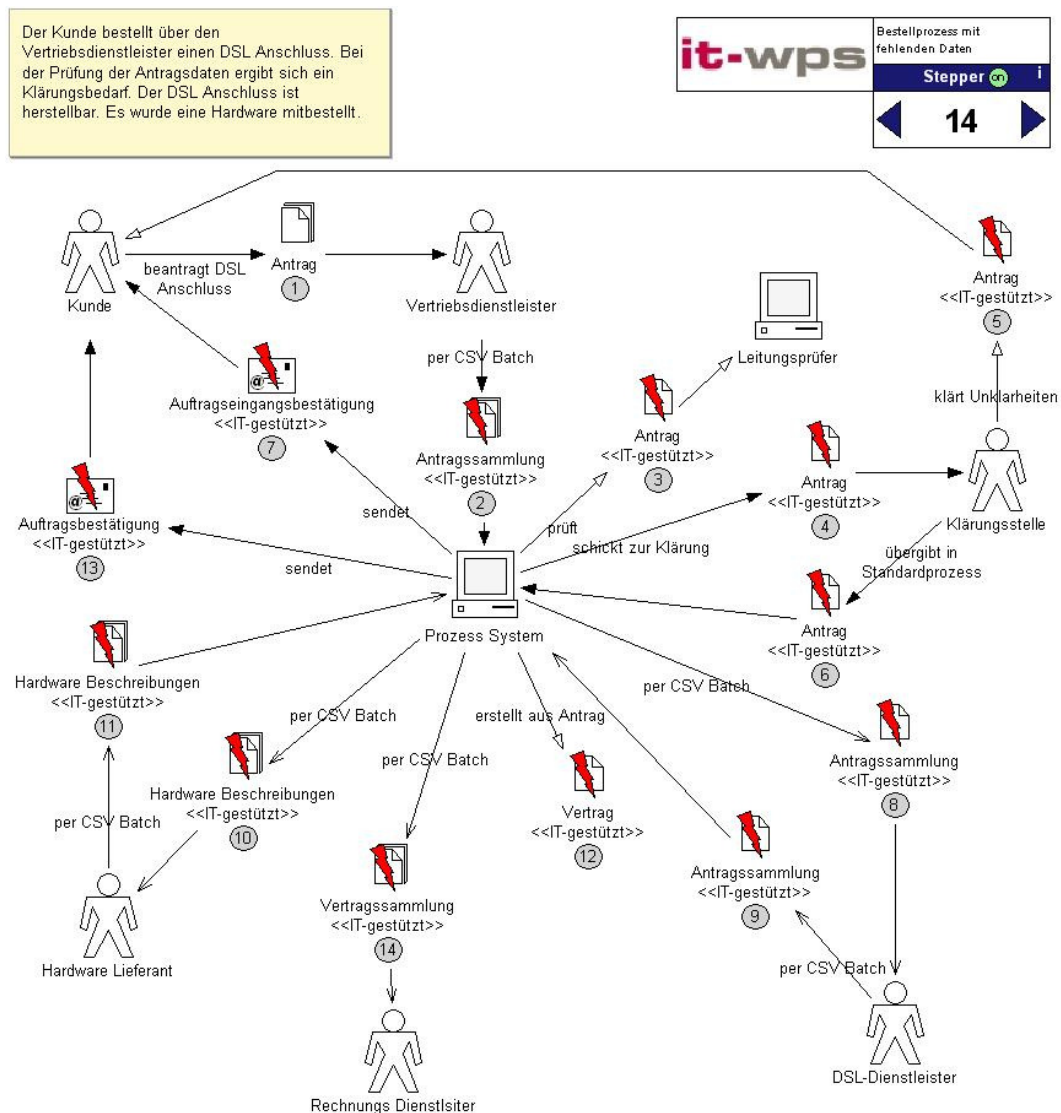


Abbildung 10: Kooperationsbild mit Interaktion mit Kunden

Nun können in jeder Batch Schnittstelle sowohl fachliche als auch technische Fehler auftreten. Ein Kommunikationsfehler zum Dateiserver beim Lesen einer Batch Datei wäre ein möglicher technischer Fehler. Andererseits könnte auch der Hardware Lieferant in Lieferengpässe kommen und das über die Schnittstelle melden. Das würde dann zu einem fachlichen Fehler führen.

Sowohl technische als auch fachliche Fehler sind in den Prozessen nicht modelliert worden, da diese die Modelle unübersichtlich und unverständlich machen würden. Es

würde auch keine weiteren Kenntnisse über den Prozess bringen, da alle dieser Fehler zu einer Einschaltung der Klärungsstelle führen.

4.1.3 Zielsystem

Im Zielszenario steht ein Softwaresystem zur Verfügung, das den typischen Prozessablauf vollautomatisch abwickelt und für alle Sonderfälle einem Prozessbearbeiter die Kontrolle über den Prozess gibt.

Der Ablauf eines Prozesses muss jederzeit für die Prozessbearbeiter nachvollziehbar sein. Ebenfalls muss jederzeit die Statuskontrolle von laufenden Vorgängen möglich sein.

Die Anpassung des Ablaufes an geänderte Anforderungen soll hauptsächlich über die Konfiguration der Workflow Engine erfolgen. Unter Umständen muss auch der Laufzettel und einzelne Entscheidungsknoten angepasst werden. Diese Arbeit wird auf jeden Fall von einem Entwickler mit guten Kenntnissen über das Gesamtsystem erfolgen müssen.

Die Programmierung von Ein- und Ausgabeschnittstellen sollen ohne wesentliche Kenntnis des Gesamtprozesses und des Gesamtsystems erfolgen können. Die Schnittstellen müssen als austauschbare Komponenten realisiert sein. Ein Fachprogrammierer ohne tiefere Kenntnis über Software Architekturen muss in der Lage sein, diese Schnittstellen mit Hilfe eines Beispiels zu implementieren.

Die Benutzeroberflächen können entweder als Rich- oder als Thin-Client Applikationen entwickelt werden. Für die Implementierung der Clients muss der Entwickler keine Kenntnisse über die Workflow-Engine haben.

Der Applikationsserver auf dem die Workflow Applikation betrieben wird, soll aus sicherheitstechnischen Gründen in einer demilitarisierten Zone (DMZ) betrieben werden. Daher müssen auf jeden Fall auch entfernte Aufrufe möglich sein. Die Daten sollen in einem RDBMS (voraussichtlich Oracle) abgelegt werden.

4.2 Architektur des Gesamtsystems

In diesem Kapitel werden motiviert durch den Anforderungskatalog grundlegende Entscheidungen zur Realisierung des Gesamtsystems getroffen und begründet. Einerseits werden die Vorgaben für die Gesamtarchitektur des Systems gegeben und andererseits werden auch notwendige Basistechnologien und Bibliotheken ausgewählt. Dadurch wird ein Rahmen vorgegeben, in den sich die Einzelteile des Systems integrieren und einpassen werden. Es wird im Rahmen der Entwicklung darauf geachtet, möglichst viele Aspekte des Systems durch Open Source Komponenten abzudecken.

4.2.1 Auswahl der Programmiersprache

Da im Unternehmensumfeld, aus dem diese Arbeit entstanden ist, ausschließlich JAVA eingesetzt wird, herrscht diesbezüglich keine Wahlfreiheit. Um das Potential der Neuerungen in JAVA 1.5 auszutesten, wird zur Entwicklung diese Version herangezogen. Die Grundbedingung, JAVA einzusetzen, bringt auch den Vorteil, dass in diesem Umfeld eine Vielzahl an hochwertigen Open Source Produkten existiert.

4.2.2 Auswahl eines Architekturframeworks

Zur Entwicklung einer Software ist es immer notwendig, sich explizit oder auch implizit auf ein Architekturmodell zu einigen. Das Architekturmodell beschreibt die Struktur einer Software, unabhängig vom Anwendungskontext. Dabei ist allerdings zu beachten, dass sich bestimmte Architekturmodelle für manche Anwendungskontexte besser und für andere weniger gut eignen.

Wie bereits im Kapitel 2 (Grundlagen) deutlich wurde, basieren viele Ideen und Konzepte dieser Arbeit auf dem WAM-Ansatz, der mit seinen Entwurfsmetaphern und Entwurfsmustern ein klares Architekturmodell beschreibt. Bei näherer Betrachtung zeigt sich, dass die Ideen des WAM-Ansatzes vor allem aus verteilter und kooperativer Arbeitsplatzsoftware entstanden sind. Aus dieser Überlegung heraus ist es nahe liegend, für die Entwicklung der Software das JWAM 2.0 Framework [JWAM, 05] zu verwenden, das den Entwickler der Implementierung der WAM Entwurfsmetaphern unterstützt.

4.2.3 Auswahl der Komponententechnologie

Der Anforderungskatalog in Kapitel 3.2 fordert bereits – die softwaretechnisch ohnehin sehr sinnvolle – Komponentenorientierung des Systems. In diesem Zusammenhang sind die Forderungen nach einfacher Testbarkeit des Systems und der Serviceorientierung mit zu betrachten, da diese Aspekte von der Komponententechnologie abhängig sind.

Neben diesen Kriterien, die aus dem Anforderungskatalog stammen, ist es für ein Komponentensystem sinnvoll, keine Annahmen über die Laufzeitumgebung zu treffen. Dies fördert die Wiederverwendbarkeit der Komponenten und erleichtert das Testen, da ansonsten die Laufzeitumgebung auch für das Testen zur Verfügung stehen müsste.

Die in JAVA standardisierte Komponententechnologie Enterprise Java Beans (EJB) scheidet aufgrund der geforderten Laufzeitumgebung aus. Im Gegensatz zu diesem schwergewichtigen Komponentenansatz haben sich in den letzten Jahren Ansätze für leichtgewichtige Komponentenmodelle entwickelt [Johnson, 04]. Als führende Vertreter dieses Ansatzes seien hier das Spring Framework (<http://www.springframework.org>), der Pico-Container (<http://www.picocontainer.org>) oder HiveMind (<http://jakarta.apache.org/hivemind>) genannt. Diese Frameworks bieten zwei entscheidende Vorteile zu anderen Ansätzen. Erstens verwalten sie einfach Java Objekte (POJOs) als Komponenten und zweitens kennen sie das Konzept der Dependency Injection. Dabei beziehen die Komponenten ihre Abhängigkeiten nicht aktiv von einem Service Provider sondern erhalten diese automatisch von außen zugewiesen. Damit werden Komponenten sowohl zu ihren Verwendern als auch zu ihren Abhängigkeiten lose gebunden [Fowler, 04].

Das Spring Framework implementiert darüber hinaus auch noch Unterstützung für den Betrieb von Automaten und unterstützt zahlreiche Persistenzmechanismen. Durch die flexiblen Konfigurationsmöglichkeiten wird auch das Problem der Konfiguration von Diensten durch Spring gelöst.

Auch JWAM 2.0 implementiert einen leichtgewichtigen Service Container, der allerdings keine Dependency Injection unterstützt. Aus diesen Überlegungen heraus fällt die Wahl auf das Spring Framework als Basis für die Komponentenorientierung.

4.2.4 Auswahl weiterer Basiskomponenten

Nachdem bereits zwei wichtige Rahmenbedingungen für die Entwicklung abgesteckt wurden, müssen nun noch einige weitere, aber weniger bedeutsame Komponenten ausgewählt werden.

Der Anforderungskatalog beschreibt auch die Notwendigkeit eines ausführlichen Loggings. Mit der commons logging Bibliothek von Apache Jakarta (<http://jakarta.apache.org/commons/logging>) bietet sich die Möglichkeit an, Logs zu schreiben, ohne Kenntnis der Implementierung. Dazu bietet die Bibliothek Logging Mechanismen an, die dann zur Laufzeit an Logging Bibliotheken wie Log4J oder das Java-eigene Logging delegieren.

Für Testzwecke wird das in JAVA weit verbreitete Testwerkzeug JUnit verwendet. Auch wenn es durchaus Alternativen dazu gibt, ist JUnit dennoch als „Best Practice“ anzusehen.

Als Datenbank wird vorerst HSQLDB eingesetzt. Diese Datenbank ist ebenfalls Open Source und in JAVA entwickelt. Sie bietet den großen Vorteil, dass sie In-Process gestartet werden kann und somit für Testzwecke optimal geeignet ist.

4.2.5 Beschreibung des Systems

Da nun die wichtigsten Entscheidungen zur Basistechnologie getroffen wurden, kann nun aus der Aufgabenbeschreibung und den Grundlagen eine sehr grob strukturierte Beschreibung des Gesamtsystems erfolgen.

Die Abbildung 11 (Systemübersicht) stellt das System graphisch dar. Dazu wurde bewusst keine UML Notation gewählt, da die Abbildung keinen formalen Charakter hat. Sie soll ausschließlich dazu dienen, die Zusammenhänge zu erkennen.

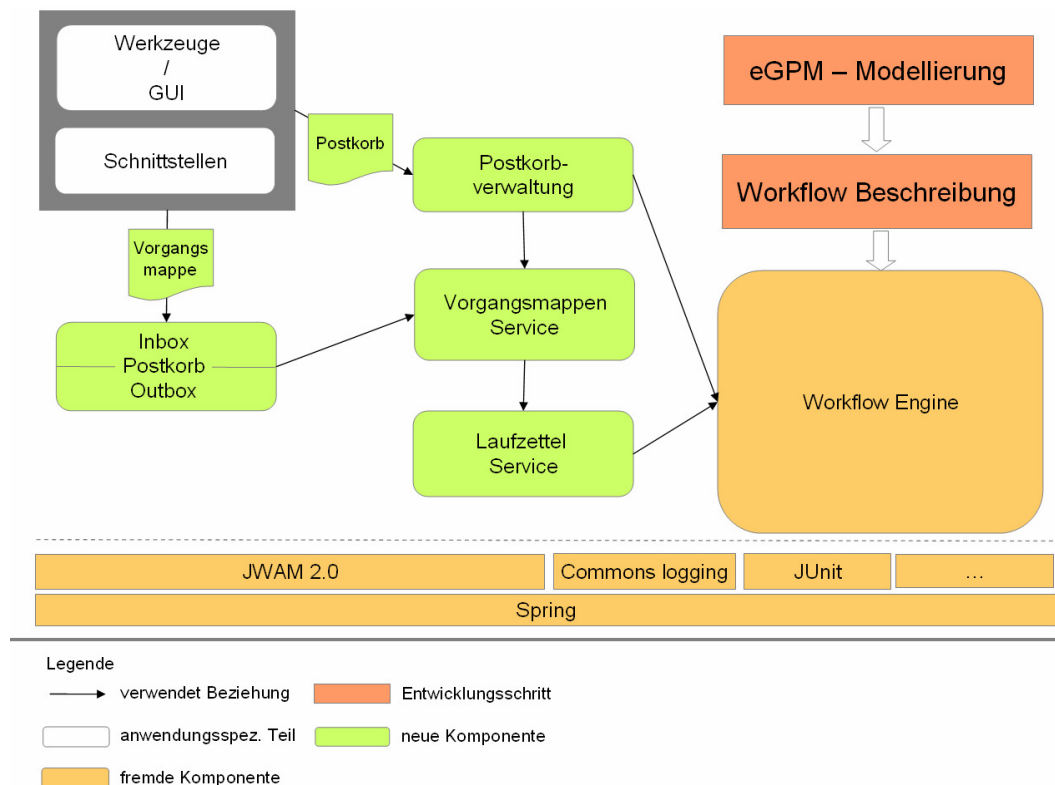


Abbildung 11: Systemübersicht

Die Workflow Engine stellt die zentrale Softwarekomponente dar, die die Prozesse verwaltet und ihre Zustände speichert. Mit der Auswahl der Workflow Engine beschäftigt sich das Kapitel 4.3 (Auswahl der Workflow Engine) ausführlich.

Um die Prozessdefinitionen für die Workflow Engine zu erstellen, werden die Prozesse im ersten Schritt mittels eGPM analysiert und unterstützt vom Musterkatalog in die Workflow Beschreibung übertragen. Dieser Vorgang wird ausführlich im Kapitel 4.4 (Realisierung eines Arbeitsablaufes) beschrieben.

Eine der Anforderung aus dem Anforderungskatalog war die Kapselung der Workflow Engine. Diese Aufgabe wird vom Laufzettel Service und von der Postkorbverwaltung übernommen. Der Laufzettel Service sorgt dafür, dass die Abbildung des Prozesses in der Workflow Engine auf Laufzetteln übertragen wird, da diese leichter verständlich und besser handhabbar sind. Die Postkorbverwaltung bildet die Zuständigkeiten im Prozessablauf auf Postkörbe ab.

Die automatischen Prozessschritte und Werkzeuge in der Prozessbearbeitung können sich von der Postkorbverwaltung die Postkörbe holen. Darin befinden sich die Vorgangsmappen, die vom Vorgangsmappen Service verwaltet werden. In der Vorgangsmappe befindet sich auch der Laufzettel. Nach der Bearbeitung einer Vorgangsmappe wird diese über den Postkorb wieder an den Vorgangsmappen Service übergeben.

Der Vorgangsmappen Service muss die Vorgangsmappen persistent ablegen. Da dies aber nicht unmittelbar mit der Prozessverarbeitung zusammenhängt, wurde im Rahmen dieser Arbeit nur eine Testimplementierung zur Speicherung von Vorgangsmappen angefertigt.

4.3 Auswahl der Workflow Engine

Nach den Grundüberlegungen zum Gesamtsystem muss nun eine Workflow Engine ausgewählt werden. Die Wahl des Produktes wird die Gesamtarchitektur des Systems zwar nicht wesentlich beeinflussen, wirkt sich aber stark in der Implementierung aus. Aus diesem Grund wird bereits vor der Implementierung diese Entscheidung getroffen.

4.3.1 Übersicht über Open Source Workflow Engines

Überblick

Der Markt der Workflow Engines ist in den letzten Jahren stark gewachsen. Neben den zahlreichen kommerziellen Anbietern wie IBM, BEA, Oracle usw. haben sich mittlerweile auch viele Open-Source Projekte durchgesetzt. Deshalb wurde auch die grundsätzliche Entscheidung getroffen, auf ein Open-Source Produkt zu setzen. Wie so oft in der Open-Source Szene ist die Anzahl der Projekte unüberschaubar groß geworden und so kann in der Produktübersicht auch nur auf die etablierten Projekte eingegangen werden.

Als Entscheidungsgrundlage für die Bekanntheit der Projekte werden folgende Faktoren herangezogen:

- Suchtreffer in Google
- Suchtreffer in MSN
- Bekanntheit der Sponsoren / Unterstützer / Betreiber

Weiters werden nur Java-basierte Produkte betrachtet, da im konkreten Projekt- und Unternehmensumfeld nur Java eingesetzt wird. Manche Produkte werden sofort ausgeschlossen, da die Engine grundsätzlich keinen Einfluss auf die zu wählenden Basistechnologien nehmen darf. Die gesamte Arbeit soll in möglichst beliebigem technologischem Umfeld verwendet werden können. Das ist zwar schon ein Vorgriff auf die Anforderungen, an dieser Stelle aber für eine effiziente Produktauswahl notwendig.

Die Tabelle 1 gibt ein Überblick über Open-Source Workflow Engines und die Bewertung deren Bekanntheit. Die Liste stammt von [Manageability, 05] und stellt die umfangreichste gefundene Sammlung dar.

	Google	MSN	Sponsoren
Enhydra Shark http://shark.enhydra.org/	106.000	3.981	ObjectWeb, Together
JBpm http://www.jbpm.org/	18.400	8.017	JBoss Group
OpenSymphony OSWorkflow http://www.opensymphony.com/osworkflow	20.100	1.746	-
OpenWFE http://openwfe.sourceforge.net/	8.040	2.540	
Micro-Workflow http://micro-workflow.com/Downloads.phtml	1.540	653	
Bigbross Bossa http://www.bigbross.com/bossa/	1.170	61	
con:cern http://concern.sourceforge.net/	743	10	
XFlow2 http://www.kgionline.com/xflow2/	425	13	
YAWL http://www.citi.qut.edu.au/yawl	ist für die Verwendung in Tomcat vorgesehen		
Codehaus Workflow http://werkflow.codehaus.org/	kein Release verfügbar		
wfmOpen http://wfmopen.sourceforge.net/	läuft nur in einem EJB-Container		
OFBiz Workflow Engine http://www.ofbiz.org/docs/workflow.html	läuft nur in einem EJB-Container		
ObjectWeb Bonita http://bonita.forge.objectweb.org/	läuft nur auf Basis von JOnAS		
Taverna http://taverna.sourceforge.net/	Workflow Workbench und nicht nur Engine		
JFolder http://www.jfolder.com/	Workflow Workbench und nicht nur Engine		
Open Business Engine http://www.openbusinessengine.org/index.html	kein Release verfügbar		
FreeFluo http://freefluo.sourceforge.net/	nur für WebServices		
ZBuilder	keine Angaben verfügbar		
Twister http://www.smartcomps.org/twister/	benötigt einen Webserver		
Zebra http://zebra.tigris.org/	noch kein Release verfügbar		
ActiveBPEL http://www.activebpel.org/	benötigt einen Servlet Container		
Apache Agila	erst in der Genehmigungsphase		
Antflow http://antflow.onionnetworks.com/	kein Release verfügbar		
MidOffice BPEL Engine http://mobe.objectweb.org/	läuft nur in einem EJB-Container		

Tabelle 1: Übersicht über Open-Source Workflow Engines

Die fünf bekanntesten Produkte werden für die weitere Begutachtung herangezogen. Im nächsten Abschnitt folgt noch eine genauere Beschreibung. Im späteren Verlauf werden die Produkteigenschaften mit den Anforderungen verglichen und so ein Produkt für die Implementierung ausgewählt. Aus aufwandstechnischen Gründen kann die Implementierung nur mit einem Produkt erfolgen. Dabei wird das Risiko eingegangen, dass unter Umständen eine andere Wahl günstiger gewesen wäre.

Produkte

Enhydra Shark

Enhydra ist eine Open Source Community, die unter anderem vom ObjectWeb und von Together unterstützt wird. Eines der Projekte ist die Workflow Engine „Shark“. In der Selbstbeschreibung von Shark wird besonders die Konformität zum Standard der Workflow Management Coalition (WfMC, vgl. [WfMC, 05]) betont. Die WfMC hat mit der extensible Process Definition Language (XPDL, vgl. [WfMC, 02]) einen XML Standard zur Beschreibung von Workflows geschaffen. Dieser Standard wird von Shark implementiert. Ein graphischer Editor für XPDL wird ebenfalls als Projekt von Enhydra entwickelt.

Shark kann in beliebigem technologischem Umfeld eingesetzt werden. Zur Speicherung der Daten wird eine relationale Datenbank verwendet, die über den O/R-Mapper DODS von Enhydra gekapselt wird.

jBpm – Java Business Process Management

Das Projekt jBpm wird mittlerweile unter dem Dach der JBoss Professional Open Source Federation geführt. Die jBpm kann auch außerhalb des JBoss Applications Servers betrieben werden. Zur Prozessdefinition wird die Java Process Definition Language (jPdl) verwendet. Die jPdl ist ebenfalls ein auf XML basiertes Format, das aber nur in der jBpm angewendet wird. Für die jPdl ist auch ein grafischer Editor verfügbar.

Eine Besonderheit der jBpm ist die interne Historisierung von Prozessdefinitionen. Das bedeutet, dass in der jBpm mehrere Versionen der gleichen Prozessdefinition koexistieren und auch parallel ausgeführt werden können.

OpenSymphony OSWorkflow

OpenSymphony ist ähnlich wie Enhydra ein Open Source Dachprojekt zur Entwicklung unterschiedlicher J2EE Komponenten. Im Gegensatz zu Enhydra hat aber OpenSymphony keine namhaften Sponsoren oder Unterstützer auf ihrer Internetpräsenz genannt.

Das Projekt OSWorkflow von OpenSymphony beschreibt sich selbst als eine Basisimplementierung einer Workflow Engine, die im Gegenzug dafür ein hohes Maß an Flexibilität mitbringt. Zur Beschreibung des Ablaufes wird wieder ein eigener XML Dialekt verwendet. Ein rudimentärer GUI Editor ist für den XML Dialekt verfügbar.

OpenWFE

Das Projekt OpenWFE ist in kein Dachprojekt integriert. Es ist auch eines der wenigen Projekte, das neben der Engine selbst auch eine Weboberfläche und weitere Bibliotheken mitbringt. Allerdings ist die Engine auch entkoppelt einsetzbar.

Micro-Workflow

Das Projekt Micro-Workflow stammt von Dragos Manolescu, der für ThoughtWorks arbeitet [Manolescu, 01]. Das Projekt war ursprünglich in SmallTalk entwickelt worden, wurde aber später von Manolescu selbst nach Java portiert. Micro-Workflow ist im

Rahmen seiner Dissertation entstanden. Er verfolgt dabei die Idee von Micro-Kernels um die Komponenten einer Workflow Engines zusammenarbeiten zu lassen.

4.3.2 Entscheidungskriterien

Die Kriterien für die Auswahl der Workflow Engine leiten sich weitestgehend aus den Anforderungen in Kapitel 3.2 Anforderungskatalog ab. Es ist natürlich vorteilhaft, wenn bereits die Workflow Engine möglichst viele der notwendigen Anforderungen implementiert bzw. realisiert.

Zusätzlich zu den Anforderungen an das Zielsystem wird die Entscheidung auch noch von Aspekten der Entwicklung des Gesamtprojektes beeinflusst.

Standard zur Ablaufbeschreibung

Die Beschreibung des Arbeitsablaufes soll in einem gängigen Standard erfolgen.

Einfachheit der Ablaufbeschreibung

Die Beschreibungssprache der Workflow Engine soll einfach erlernbar und gut lesbar sein.

Flexible Einbindung von JAVA Code

Die Einbindung von JAVA Code für Entscheidungsknoten und benutzerdefinierte Aktionen muss einfach möglich sein. Auf eine entkoppelte Testbarkeit der eingebunden Code Teile ist zu achten.

Historisierung

Die Historisierung des Ablaufes könnte bereits von der Engine realisiert werden.

Logging

Das Logging der Aktionen soll flexibel konfigurierbar sein. Die Verwendung einer gängigen Logging API (z.B. Log4j) wäre wünschenswert.

Überwachung

Die Abfrage der laufenden Abläufe muss jederzeit möglich sein. Eine Benutzeroberfläche zur Überwachung ist wünschenswert.

Ausführliche Dokumentation

Die Dokumentation sowohl für die Implementierung des Gesamtsystems als auch für die Beschreibung der Arbeitsabläufe soll ausreichend und verständlich sein.

Einfache Benutzungsschnittstellen

Die Schnittstellen zur Anbindung der Workflow Engine sollen im Sinne eines möglichst geringen Implementierungsaufwandes einfach und überschaubar sein.

Verwendung gängiger Open Source Standards

Es wird davon ausgegangen, dass auch das Gesamtsystem auf Open Source Produkten basiert. Deshalb ist es sinnvoll, dass auch die Workflow Engine für Aspekte wie OR-Mapping, Logging usw. Open Source Produkte verwendet.

4.3.3 Entscheidungsmatrix

Die Entscheidungskriterien des vorhergehenden Abschnitts werden nun gegen die fünf zur Auswahl stehenden Produkte gestellt. Die Bewertung erfolgt aufgrund der Produktbeschreibungen und Dokumentationen der einzelnen Produkte. Zusätzlich wurden jeweils mitgelieferte Code Beispiele und die Schnittstellenbeschreibungen beurteilt.

Die Bewertungsskala reicht von „nicht erfüllt“ mit 0 Punkten bis zu „voll erfüllt“ mit 5 Punkten. Somit ist im Anschluss eine Summierung der Punkte und somit ein Erfüllungsgrad der Anforderungen ablesbar. Die Ergebnisse der Bewertung sind in Tabelle 2: Entscheidungsmatrix für die Workflow Engine dargestellt. Die erste Zahl gibt jeweils die Bewertung an und die zweite Zahl bezieht bereits die Gewichtung mit ein.

	<i>Gew.</i>	Enhydra Shark	jBpm	OSWork-flow	Open WFE	Micro-Workflow
Standard zur Ablaufbeschreibung	2	5 / 10	1 / 2	1 / 2	1 / 2	1 / 2
Einfachheit der Ablaufbeschreibung	3	0 / 0	4 / 12	2 / 6	2 / 6	3 / 9
Flexible Einbindung von JAVA Code	3	1 / 3	5 / 15	4 / 12	2 / 6	4 / 12
Historisierung	1	3 / 3	0 / 0	0 / 0	3 / 3	1 / 1
Logging	1	3 / 3	4 / 4	3 / 3	3 / 3	1 / 1
Überwachung	1	5 / 5	0 / 0	0 / 0	4 / 1	1 / 1
Ausführliche Dokumentation	2	2 / 4	4 / 8	3 / 6	2 / 4	1 / 2
Einfache Benutzungsschnittstellen	3	2 / 6	4 / 12	4 / 12	2 / 6	3 / 9
Verwendung gängiger OS Standards	2	3 / 6	5 / 10	1 / 2	1 / 2	1 / 2
Summe		40	63	43	33	39

Tabelle 2: Entscheidungsmatrix für die Workflow Engine

4.3.4 Entscheidung

Die Tabelle 2 (Entscheidungsmatrix für die Workflow Engine) zeigt die Punkteverteilung für die einzelnen Produkte. Aus dieser Übersicht gehen die jBpm und Enhydra Shark als klare Favoriten hervor.

Die Wahl fällt auf das Produkt jBpm von JBoss. Der Grund dafür liegt vorwiegend in der Einfachheit der jBpm und der flexiblen Einbindung von JAVA Code. Bei der Evaluierung des XPD L Standards, der von Enhydra Shark verwendet wird, hat sich darüber hinaus gezeigt, dass dieser für eine effiziente Modellierung ungeeignet ist. Bereits die Darstellung eines sehr einfachen Prozesses bedarf mehrerer Seiten XML Beschreibung ([WFMC, 02] ab Seite 52).

4.4 Realisierung eines Arbeitsablaufes

Der folgende Abschnitt beschäftigt sich mit einer der zentralsten Fragen dieser Arbeit. Es ist nun zu betrachten, wie ein Arbeitsablauf auf Prozessebene und in der Workflow Engine beschrieben wird. In diesem Zusammenhang ist auch zu klären, wie der

exemplarische Ansatz der eGPM und der formale Ansatz eine Workflow Beschreibung zusammengeführt werden können.

4.4.1 Zusammenführung von ablaufsteuernder und unterstützender Sichtweise

Bereits im Kapitel 2.5 (Kooperative Arbeit durch Prozessmuster) wurden die Begriffe Ablaufsteuerung und Unterstützung kooperativer Arbeit zueinander abgegrenzt. Bei der Realisierung von teilautomatisierten Prozessen ergibt sich die Besonderheit, dass automatisierte Schritte ablaufgesteuert werden sollen, aber im gleichen Prozess der Mensch bei der Klärung von Sonderfällen lediglich unterstützt werden soll. Somit ergibt sich aufgrund der Aufgabenstellung, dass das Softwaresystem sowohl ablaufgesteuerte als auch unterstützende Kooperation realisieren muss.

Die ablaufgesteuerten Teile des Prozesses lassen sich in der Workflow Engine problemlos abbilden (vgl. Kapitel 2.6.4 (WfMS und ablaufgesteuerte kooperative Arbeit)). Die wichtigere Frage ist, wie die unterstützende Sichtweise in der Workflow Engine implementiert werden kann.

Im Kapitel 2.5.2 (Prozessmuster) wurde das Konzept der Prozessmuster als geeignete Mittel zur Unterstützung kooperativer Arbeit vorgestellt. Betrachtet man ein Kooperationsbild aus eGPM, so stellt dieses einen möglichen Ablauf des Prozesses dar und kann somit als ein Prozessmuster angesehen werden. Überträgt man dieses Muster in die Workflow Engine, so ergibt sich eine Ablaufsteuerung dieses einen Musters. Das würde aber der Idee des Prozessmusters widersprechen, da das besondere daran ja gerade die Veränderbarkeit zur Laufzeit darstellt.

Ein Prozess wird üblicherweise durch mehrere Kooperationsbilder beschrieben. Somit existiert zum Prozess eine Menge von Prozessmustern. Diese Menge stellt die Summe der typischen Abläufe des Prozesses dar. Man kann nun die Menge der Kooperationsbilder durch Einfügen von Fallunterscheidungen und Verzweigungen in einer Workflow Beschreibung zusammenfassen. Somit repräsentiert die Workflow Beschreibung nicht ein Prozessmuster, sondern einen Prozessmusterkatalog aller typischen Abläufe für einen Prozess. Die Workflow Engine kann also nun alle typischen Abläufe des Prozesses ablaufsteuern. Damit ist aber die Frage der Flexibilität bei Prozessmustern immer noch nicht geklärt.

Workflow Engines arbeiten üblicherweise (so auch die jBpm) auf gerichteten Graphen und darauf aufbauend auf Petri-Netzen. Das bedeutet, dass ihr Zustandsraum und die Zustandsübergänge streng deterministisch sind. Somit kann eine Workflow Engine die Flexibilität, die für Prozessmuster nötig sind, nicht bieten. Es muss also ein Konzept gefunden werden, wie trotz eines definierten Zustandsraumes und Zustandsübergangsraumes die Flexibilität von beliebigen Zustandsübergängen erreicht werden kann. Diese Fragestellung führt zur Idee der Klärungsstelle.

Die Flexibilität des Prozessmusterkonzeptes ist nur an Stellen der menschlichen Einflussnahme auf den Prozess notwendig. Die automatisierten Teilschritte sollen ablaufgesteuert sein und sollen daher das Prozessmuster gar nicht verändern können.

Da die menschliche Einflussnahme laut Aufgabenstellung nur in Sonderfällen oder unvorhergesehenen Situationen stattfinden soll, wird hier der Begriff „Klärungsstelle“ dafür verwendet.

Klärungsstelle

Eine Klärungsstelle ist ein definierter Zustand im Prozess an dem die Möglichkeit besteht, den Vorgang in jedem beliebigen anderen Zustand zu setzen und weiter zu führen.

Es ist auch denkbar, dass die Klärungsstelle von einem Softwaresystem automatisch bearbeitet wird. Allerdings widerspricht dies der Idee, dass in der Klärungsstelle spontan über die weitere Vorgehensweise entschieden wird.

Daher werden Vorgänge, die der Klärungsstelle zugewiesen sind, üblicherweise über ein Werkzeug vom Benutzer bearbeitet und weitergeleitet.

Wenn es nun die Möglichkeit gibt, von jedem automatisierten Teilschritt aus, an die Klärungsstelle zu leiten und die Klärungsstelle die Möglichkeit hat, den Prozess in jedem verfügbaren Zustand weiterzuführen, hat der Benutzer nun die Flexibilität eines Prozessmusters, da er den Ablauf beliebig beeinflussen kann.

Abbildung 12 (Von der Realität zur Workflow Beschreibung) stellt die beschriebenen Schritte graphisch dar. Durch Analyse werden die typischen Abläufe der Realität graphisch dargestellt. Durch manuelle Transformation werden die Kooperationsbilder in eine Workflow Beschreibung übertragen, die alle typischen Abläufe enthält. Diese Workflow Beschreibung wird schließlich automatisch um die Klärungsstelle angereichert und somit können nun auch alle Sonderfälle und Ausnahmen wieder behandelt werden.

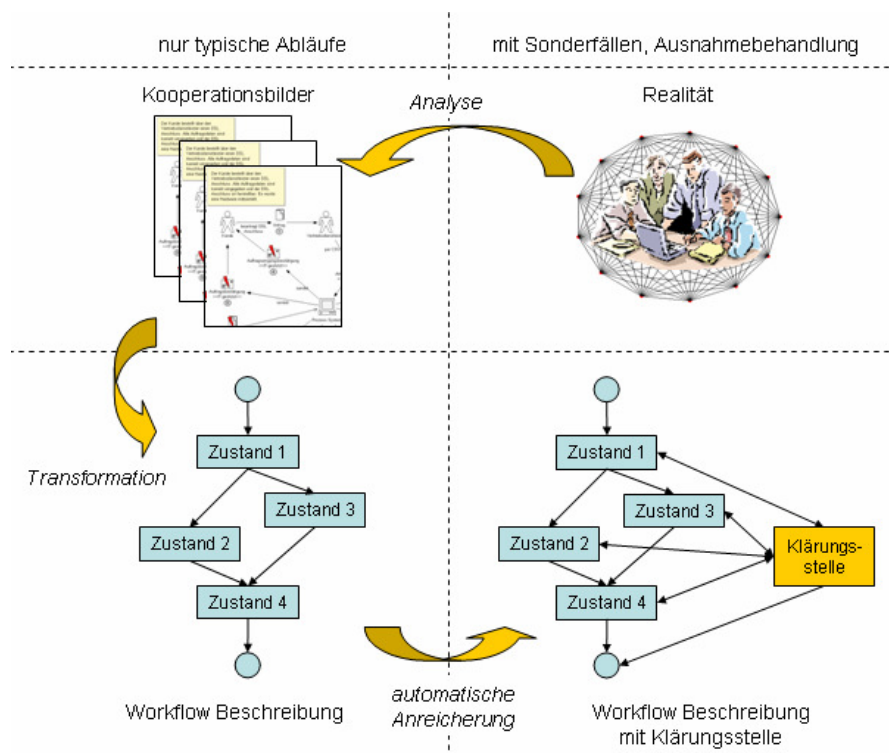


Abbildung 12: Von der Realität zur Workflow Beschreibung

Der Benutzer benötigt zur Entscheidung wie ein Prozess von der Klärungsstelle aus weiter ablaufen soll Informationen über den Prozess. Dazu steht ihm die Vorgangsmappe zur Verfügung. In der Vorgangsmappe befinden sich die Materialien zum Prozess. Diese sind abhängig von der Art des Prozesses und können daher auch nicht auf allgemeiner Ebene beschrieben werden. In der Vorgangsmappe befindet sich zusätzlich noch der Laufzettel. Der Laufzettel beschreibt den bisherigen Ablauf des Prozesses. Daran kann der Benutzer erkennen, woher der Prozess an die Klärungsstelle geleitet wurde.

Die folgenden Abschnitte betrachten die Umsetzung eines Prozesses in der Workflow Engine detaillierter.

4.4.2 Analyse mittels eGPM

Die Grundlagen der eGPM wurden bereits in Kapitel 2.3 (Die Methode der exemplarischen Geschäftsprozess Modellierung) besprochen. Auf die Vorgehensweise bei der Analyse von Geschäftsprozessen geht diese Arbeit nicht ein. In diesem Zusammenhang sei auf [Breitling, 05] verwiesen. Das Ergebnis der Geschäftsprozessanalyse stellen unter anderem die Kooperationsbilder der eGPM dar. Darüber hinaus werden vermutlich auch Begriffsmodelle, Interaktionsbilder und weitere ergänzende Dokumente entstehen, die allerdings für die Beschreibung des Ablaufes nur von zweitrangiger Bedeutung sind.

Bei der Analyse eines Geschäftsprozesses muss dieser im ersten Schritt durch einen definierten Beginn und ein definiertes Ende abgegrenzt werden. Diese Abgrenzung beeinflusst auch die spätere Realisierung in der Workflow Engine. Dabei kann unterschieden werden, ob der Prozess sehr umfassend oder sehr spezialisiert eingeschränkt wird. So kann man den gesamten Lebenszyklus eines Produktes (umfassend) oder auch nur den Bestellprozess des Produktes (spezialisiert) als Prozess betrachten. Eine sehr umfassende Gliederung der Prozesse führt zu sehr komplexen Ablaufbeschreibungen. Es empfiehlt sich, höher spezialisierte Prozesse zu betrachten. So könnte man etwa einen Produktlebenszyklus aus dem Telekommunikationsbeispiel in dessen Bestellung, Verrechnung und Kündigung aufteilen. Damit würde man drei Prozesse trennen und getrennt analysieren.

Ein zweiter wichtiger Aspekt der Modellierung ist der Detaillierungsgrad in den Einzelschritten. So kann beispielsweise die Prüfung einer Bestellung schon als Einzelschritt betrachtet werden, oder die Prüfung wird unterteilt in die Prüfung der Adressdaten und der Produktdaten. Dabei führt ein höherer Detaillierungsgrad zu einer besseren Wiederverwendbarkeit der Einzelschritte, macht aber andererseits die Kooperationsbilder komplexer. Als Richtlinie bei der Modellierung kann man als Vergleichsgröße heranziehen, wie eine entsprechende Arbeitsanweisung aufgegliedert sein würde.

Sind nun Kooperationsbilder entstanden, wie sie auch im Kapitel 4.1 (Einführung in das begleitende Beispiel) dargestellt sind, müssen diese in die Workflow Beschreibung übertragen werden.

4.4.3 Transformation in die Workflow Beschreibung

Die Workflow Engine jBpm verwendet zur Beschreibung des Workflows die jBpm process definition language (jPdl). Die jPdl ist eine XML basierte Beschreibungssprache.

Sie wird ausschließlich in der jBpm verwendet. Unter [jBpmDoc, 05] findet man eine detaillierte Beschreibung der Sprachkonstrukte. Da die jPdl sehr gut lesbar ist, wird hier auf eine weitere Einführung verzichtet.

Die jBpm verwaltet die Workflows als Zustandsautomaten. Jeder Workflow befindet sich immer in einem definierten Zustand (mit Ausnahme der kurzen Zeit des transaktionalen Zustandsübergangs). Vor und nach jedem Zustandsübergang können von der jBpm Aktionen ausgeführt werden. Die jBpm bietet die Möglichkeit an, alle Workflows in einem bestimmten Zustand abzufragen. Per Methodenaufruf wird der jBpm der Zustandsübergang für einen Workflow gemeldet.

Es wird der Ansatz gewählt, dass alle Workflows in einem bestimmten Zustand auf den gleichen Arbeitsschritt warten. Befindet sich also ein Workflow im Zustand „Exportiere zur Rechnungslegung“, so ist als nächster Arbeitsschritt der Export zum Rechnungsdienstleister erforderlich. Ist der Export abgeschlossen, wird ein Zustandsübergang ausgelöst. In Abbildung 13 (Vom Arbeitsschritt zum Zustand) ist ein einfaches Beispiel dargestellt. Die XML Repräsentation für den Zustandsgraphen ist in Beispiel 1 (Zustände in jPdl) dargestellt. Die Zustandsübergänge werden als Standardzustandsübergang bezeichnet, da sie keinen Namen zugewiesen haben. Die jBpm wird somit automatisch diesen Übergang nach Ende des Zustandes wählen. Der nicht benannte Standardzustandsübergang ist nur dann zulässig, wenn es nicht mehr als einen Zustandsübergang gibt.

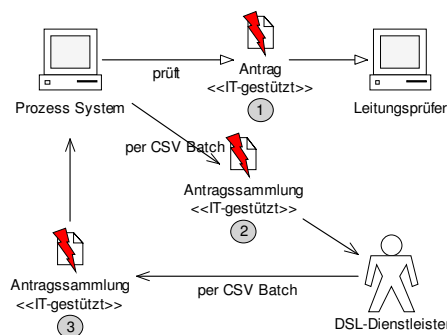


Abbildung 13: Vom Arbeitsschritt zum Zustand

```

...
<process-definition name="Example Order Process">
  <!-- START-STATE -->
  <start-state ... >

  <!-- NODES -->
  <state name="CheckOrder" >
    <transition to="ExportOrderToDSL" />
  </state>

  <state name="ExportOrderToDSL" >
    <transition to="ImportOrderFromDSL" />
  </state>

  <state name="ImportOrderFromDSL" >
    <transition to="CreateContract" />
  </state>
  ...
  <!-- END-STATE -->
  <end-state name="Finish" />
</process-definition>

```

Beispiel 1: Zustände in jPdl

Eine Besonderheit ergibt sich bei der Erstellung von Prozessen. Im Kooperationsbild findet sich üblicherweise ein Arbeitsschritt in dem der Prozess fachlich erzeugt wird. Ein solcher Arbeitsschritt könnte zum Beispiel „Importiere Bestellung“ heißen. In diesem Arbeitsschritt werden die Daten gelesen und der Prozess soll gestartet werden. Die jBpm löst beim Erzeugen eines Prozesses sofort den Übergang in den ersten definierten Zustand aus. Daher wird der Ansatz gewählt, dass für diesen ersten Arbeitsschritt auch ein Zustand definiert wird. Es existieren dann zwar keine Prozesse die in diesem Zustand bleiben, aber vom Verwendungskonzept her entspricht es dennoch dem weiter oben erwähnten Verhalten. Übertragen auf das Beispiel stellt sich die Situation folgendermaßen dar: Fachlich befindet man sich im Arbeitsschritt „Importiere Bestellung“. In diesem Arbeitsschritt wird der Prozess erzeugt und wird in der jBpm in den Zustand „Importiere Bestellung“ überführt. Ist der Arbeitsschritt fachlich abgeschlossen, wird das Zustandsende signalisiert und der Prozess setzt seinen Lebenszyklus fort. Die graphische Darstellung und die jPdl Beschreibung finden sich in Abbildung 14 und Beispiel 2.

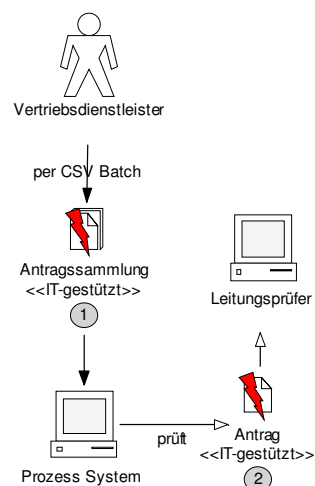


Abbildung 14: Prozesserschöpfung

```

...
<!-- START-STATE -->
<start-state name="Start" >
  <transition to="ImportOrder" />
</start-state>

<!-- NODES -->
<state name="ImportOrder" >
  <transition to="CheckOrder" />
</state>
...

```

Beispiel 2: Anfangszustand in jPdl

Die jBpm bietet auch das Konzept von Fork/Join eines Prozesses an. Durch einen Fork wird der Prozess auf zwei (oder mehr) parallel ausgeführte Zustandspfade geteilt. Beim zugehörigen Join wird gewartet, bis sich alle Zustandspfade im Join-Zustand befinden. Dann wird mit der Ausführung in einem Pfad fortgesetzt. Dieses Konzept wird nicht verwendet, da es sich nicht mit der Metapher von Laufzettel und Vorgangsmappe vereinen lässt. Fachlich gesehen würden damit für einen Prozess mehrere Vorgangsmappen mit kopierten Laufzetteln existieren, die darüber hinaus auch an

unterschiedliche Orte verteilt werden. Aufgrund der unnötigen Komplexität einer solchen Situation wird auf das Fork/Join Verhalten verzichtet.

Bereits weiter oben wurde die Möglichkeit erwähnt, vor und nach Zustandsübergängen Aktionen in die Prozesse zu integrieren. Aus fachlicher Sicht ist das nicht empfehlenswert, da es dem Ansatz widerspricht, die fachliche Logik von der Workflow Engine zu entkoppeln. Um solche Aktionen implementieren zu können, müssen Interfaces der jBpm implementiert werden. Interessante Einsatzfelder können sich für Testzwecke oder spezielle Logging oder Historisierungsmechanismen ergeben. Zur Realisierung von fachlichen Anforderungen sollte diese Möglichkeit nicht herangezogen werden.

Für eine umfangreichere Darstellung der Transformationen vom Kooperationsbild zum Arbeitsablauf sei auf Kapitel 5 (Musterkatalog zur Transformation eGPM – Laufzettel) verwiesen.

4.4.4 Anreicherung um die Klärungsstelle

Die jPdl-Beschreibung des Workflows enthält alle Informationen, die für den typischen Ablauf notwendig sind. Es ist aber noch keine Ausnahme- bzw. Sonderbehandlung über die Klärungsstelle möglich. Daher muss die jPdl-Beschreibung noch um die Klärungsstelle angereichert werden.

Die jBpm lässt nur in der Beschreibung enthaltene Zustandsübergänge zu. Wird versucht, einen nicht vorhandenen Zustandsübergang zu provozieren, so meldet die jBpm einen Fehler. Da die Workflow Engine selbst nicht verändert werden soll, muss die Beschreibung beim Einspielen in die jBpm um die Klärungsstelle angereichert werden.

Die jBpm bietet einen Definitionsservice für das Deployment an. Diesem Service werden Prozess Archive übergeben, die alle für den Workflow notwendigen Beschreibungen und Klassen beinhalten. Die jBpm analysiert die Datei mit einem Archiv Parser und übergibt die XML Beschreibung des Prozesses an einen Definitions-Parser. Dieser erzeugt auch das Objektmodell, das die Workflow-Definition widerspiegelt.

Da die jBpm keine Möglichkeit vorsieht, das Parsen der Definition zu beeinflussen, musste der gesamte Service ersetzt werden. Dazu wurde der Code für das Deployment kopiert und verändert.

Zuerst wurde eine neue Service Fabrik implementiert, die eine spezielle Implementierung mit der automatischen Anreicherung um die Klärungsstelle instanziert und zurückgibt. Die selbst implementierte Factory kann im Konfigurationsfile der jBpm angegeben werden.

Da die Wahl der Namen für die Klärungsstelle, für den Standardzustandsübergang sowie den Zustandsübergang zur Klärungsstelle flexibel sein soll, muss diese Information zur Prozessdefinition hinzugefügt werden. Dazu wurde das JavaBean `ProcessDefinition` eingeführt, das alle notwendigen Informationen für das Deployment eines Prozesses zusammenfasst. Der `Deployer` ist ein JavaBean, das die `ProcessDefinition` Objekte im Definitions-Service der jBpm veröffentlicht. Der `Deployer` wurde eingeführt, um die Prozessdefinitionen über die Spring Konfiguration laden zu können.

Der Definitionsservice und der Archiv Parser wurden so angepasst, dass sie die `ProcessDefinition` an den Definitionsparser weiterleiten. Dieser Parser ergänzt die Informationen um einen Zustand für die Klärungsstelle. Dieser Zustand hat Übergänge zu allen anderen Zuständen. Bei allen Zuständen die der Parser analysiert, wird der Übergang zur Klärungsstelle hinzugefügt. Dabei kann die Situation entstehen, dass vorher nur ein unbenannter Zustandsübergang – der Standardübergang - definiert war. Ist dies der Fall, wird dieser um den Standardübergangsnamen aus der `ProcessDefinition` ergänzt. Die Abbildung 15 (Klassendiagramm Deployment) zeigt die beteiligten Klassen.

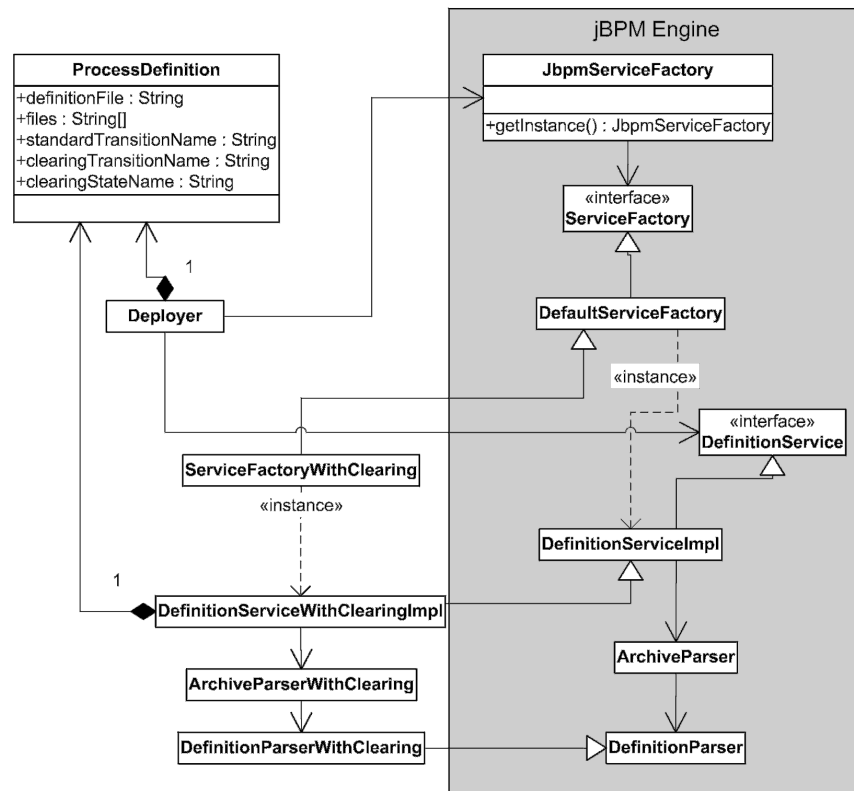


Abbildung 15: Klassendiagramm Deployment

Durch die Einführung des `Deployers` können Prozessdefinitionen über die Spring Konfigurationsdateien geladen werden. Im Beispiel 3 (Konfiguration einer Prozessdefinition) ist eine Beispielkonfiguration dargestellt. Der `Deployer` wird über eine abstrakte Bean Definition mit der Service Fabrik ausgestattet. Die Bean „simpleProcess“ ist eine `ProcessDefinition`, die als Referenz einem konkreten `Deployer` zur Veröffentlichung übergeben wird.

```
<bean id="deployer" class="cc.schmelzer.process.deployment.Deployer"
    abstract="true">
    <property name="factory">
        <ref bean="jbpmServiceFactory"/>
    </property>
</bean>
<bean id="processDefinition"
    class="cc.schmelzer.process.deployment.ProcessDefinition"
    abstract="true">
</bean>
<bean id="simpleProcess" parent="processDefinition">
    <property name="definitionFile">
        <value>cc/schmelzer/example1/simpleOrderProcess.xml</value>
    </property>
    <property name="files">
```



```
        <list>
        </list>
    </property>
    <property name="standardTransitionName">
        <value>standardTransition</value>
    </property>
</bean>

<bean id="simpleProcessDeployer" parent="deployer">
    <property name="definitions">
        <list>
            <ref bean="simpleProcess"/>
        </list>
    </property>
</bean>
```

Beispiel 3: Konfiguration einer Prozessdefinition

4.5 Realisierung von Laufzettel und Vorgangsmappe

Die Workflow Beschreibung – angereichert um die Klärungsstelle - steht der Workflow Engine nun zur Abarbeitung zur Verfügung. Das folgende Kapitel beschäftigt sich mit der Darstellung des Workflows für den Verwender. Im Rahmen der Anforderungen wurde dargestellt, dass bei der Entwicklung von Teilschritten, keine Kenntnis der Workflow Engine bzw. der Workflow Technologie vorhanden sein muss. Diese Anforderung wird durch die Darstellung des Workflows in Form des Laufzettels realisiert.

4.5.1 Der Laufzettel

Bei der Bearbeitung des Prozesses, müssen die Teilschritte die Möglichkeit haben, den Prozess an die Klärungsstelle zu leiten. Die Klärungsstelle, die üblicherweise von einer Person mit Hilfe eines Werkzeuges bearbeitet wird, verändert die Daten des Prozesses und leitet den Prozess dann in einem beliebigen verfügbaren Schritt weiter. Um den Prozess qualifiziert bearbeiten zu können, benötigt ein Bearbeiter die zum Prozess gehörenden Materialien und auch den bisherigen Ablauf des Prozesses. Die Darstellung des geplanten zukünftigen Ablaufes ist für einen Bearbeiter sicherlich auch sinnvoll, da ansonsten ein sehr hohes Maß an Prozess- und auch IT-Wissen vorauszusetzen wäre.

Daraus leiten sich die wichtigsten Anforderungen an den Laufzettel ab:

1. Möglichkeit zur Bestimmung des nächsten Empfängers
2. Darstellung des bisherigen Prozessverlaufes
3. Darstellung des zukünftigen Ablaufes

Die zum Prozess gehörenden Daten werden in der Vorgangsmappe gesammelt und sind nicht Bestandteil des Laufzettels. Die Vorgangsmappe wird im Kapitel 4.5.3 beschrieben.

Im Kapitel 2.5.3 (Laufzettel) wurde bereits die fachliche Bedeutung des Laufzettels beschrieben. Im Rahmen dieser Beschreibung wurden die Möglichkeiten zur Beeinflussung des Prozessmusters hervorgehoben. Da aber bei teilautomatisierten Prozessen und der Verbindung von Ablaufsteuerung und Unterstützung zur Kooperation die Veränderbarkeit des Prozessmusters durch die Klärungsstelle eingeführt wurde, verliert der Laufzettel an dieser Stelle seine Bedeutung zur Beeinflussung des Prozesses. Daher wird der Laufzettel so gestaltet, dass der Ablauf nicht beliebig beeinflusst werden kann, sondern nur der nächste Empfänger beeinflussbar ist.

Vor allem in der Klärungsstelle ist es wichtig, dass dem Bearbeiter gezeigt wird, in welchen Zuständen er fortsetzen kann. Also sollen zusätzlich zur Möglichkeit der Bestimmung des nächsten Empfängers auch alle möglichen nächsten Empfänger abgefragt werden können.

Zur Darstellung des bisherigen Ablaufes wird eine Sammlung von Laufzetteleinträgen verwendet. An diesen Einträgen sind das Eingangsdatum und der Empfänger vermerkt. Um die Nachvollziehbarkeit für den Bearbeiter zu erhöhen, wird auch die Möglichkeit eingeräumt, beim Abschluss des Eintrages einen Textkommentar zu setzen.

Eine besondere Herausforderung stellt die Darstellung des zukünftigen Ablaufes dar. Die Workflow Beschreibung stellt ja alle möglichen Prozessabläufe dar. Deshalb ist es natürlich nicht möglich, den genauen Weg des Prozesses vorherzubestimmen. Daher bieten sich zwei Möglichkeiten an, um den wahrscheinlichsten Ablauf festzustellen. Eine Möglichkeit ist die Betrachtung ähnlicher Prozesse und deren Verlauf und die zweite Möglichkeit ist, den Standardübergängen des Prozesses zu folgen. Die erste Möglichkeit ist fachlich sehr interessant, sprengt aber den Rahmen dieser Arbeit. Deshalb wird hier der einfachere Weg gewählt. Der Laufzettel stellt eine Liste zukünftiger Empfänger zur Verfügung, die aufgrund folgender Kriterien erstellt wird:

1. Folge dem Standardübergang des Prozesses
2. Folge bei mehreren benannten Zustandsübergängen immer dem ersten
3. Wenn der Empfänger bereits in der Liste ist, folge dem Zustandsübergang nicht. (Diese Regel vermeidet Endlosschleifen)
4. Folge niemals dem Zustandsübergang zur Klärungsstelle

Mit diesen einfachen Regeln kann in einer Vielzahl von Fällen eine gute Darstellung des zukünftigen Ablaufes sichergestellt werden. Allerdings wird dadurch nur ein möglicher Ablauf präsentiert.

Die beschriebenen Anforderungen und Lösungsansätze werden von den Interfaces in Abbildung 16 (Interfaces für den Laufzettel) spezifiziert.

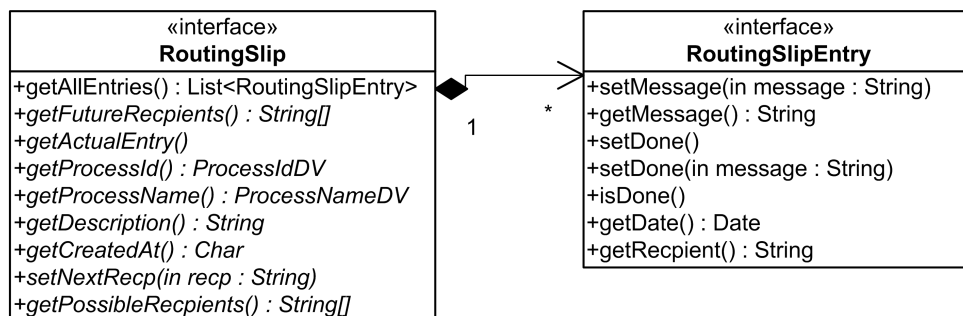


Abbildung 16: Interfaces für den Laufzettel

4.5.2 Bearbeitung des Laufzettels

Im vorherigen Abschnitt wurde auf die Modellierung des Laufzettels eingegangen. Nun wird der Laufzettel Service beschrieben, der die Laufzettel für Prozesse erstellt, abspeichert und wieder lädt. Der Laufzettel Service bietet für genau diese Operation

jeweils eine Methode an. Das Interface ist in Abbildung 17 (Interface RoutingSlipService) dargestellt.



Abbildung 17: Interface RoutingSlipService

Die Methode `void synchronize(RoutingSlip slip)` speichert den Laufzettel in der Engine ab. Wurde der Prozess bereits weitergeführt bzw. von anderer Stelle aus bearbeitet, wirft diese Methode eine Exception.

Für dieses Service Interface wird eine Implementierung für die jBpm zur Verfügung gestellt. Diese Implementierung speichert keine Daten selbst. Alle Daten werden in der jBpm abgelegt und von dort geladen.

Die Historie des Laufzettels – also die Laufzettel-Einträge – werden aus den Workflow Logs der jBpm ausgelesen. Die Textkommentare für die Laufzetteleinträge werden in Form einer Workflow Variable in der jBpm gespeichert. Der zukünftige Verlauf des Workflows wird anhand der Prozessdefinition aus der jBpm rekonstruiert.

Die Zusammenhänge der Implementierungsklassen zeigt Abbildung 18 (Laufzettel Service Implementierung) in den wichtigen Auszügen. Die jBpm Implementierung des Laufzettel Services instantiiert dabei auch die jBpm spezifischen Implementierungen von Laufzettel und Laufzetteleintrag. Dabei enthalten die spezifischen Implementierungen die notwendigen Daten um den Laufzettel mit der jBpm zu synchronisieren.

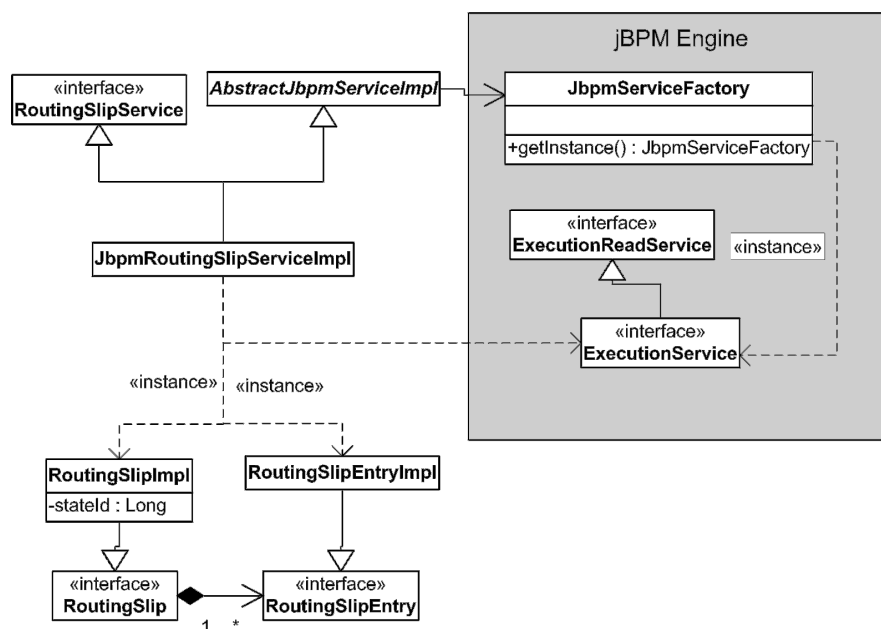


Abbildung 18: Laufzettel Service Implementierung

Um den Prozess in im Laufzettel Service zu identifizieren wird der Fachwert `ProcessIdDV` verwendet. Für nähere Details über das Fachwertmuster sei auf [Bleek et al, 99] verwiesen. Im `ProcessIdDV` ist auch der eindeutige Schlüssel der jBpm für den Workflow enthalten.

4.5.3 Die Vorgangsmappe

Der Laufzettel beschreibt den Prozessablauf selbst, beinhaltet aber keine Materialien zum Prozess. Ein Arbeitsprozess kann aber nur im Kontext mit seinen Materialien bearbeitet werden. Daher müssen die Materialien und die Prozessbeschreibung zusammengeführt werden. Diese Aufgabe wird von der Vorgangsmappe übernommen.

Da die Materialien eines Prozesses sehr stark von dessen Kontext abhängen, kann in diesem Zusammenhang nur eine sehr generische Schnittstelle angeboten werden. Im allgemeinen Fall fasst die Vorgangsmappe einen Laufzettel mit einer Sammlung von Materialien zusammen.

In Abbildung 19 (Klassendiagramm Vorgangsmappen) ist die Schnittstelle für die Vorgangsmappe dargestellt. Die Vorgangsmappe selbst kann überhaupt nicht bearbeitet werden. Die Vorgangsmappe bietet aber die Möglichkeit, Referenzen auf den Laufzettel und den Materialbehälter zu erhalten.

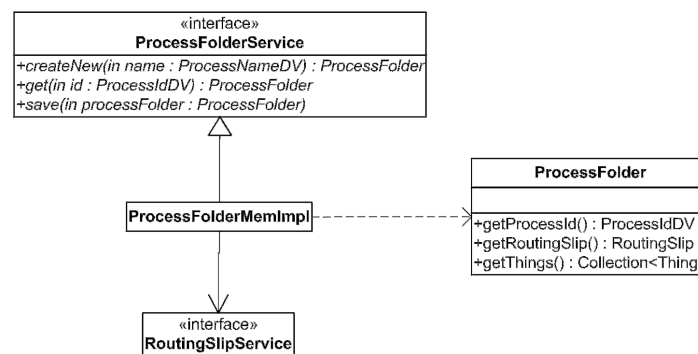


Abbildung 19: Klassendiagramm Vorgangsmappen

Wie schon beim Laufzettel Service bietet auch der Vorgangsmappen Service Methoden zum Erzeugen, Laden und Speichern der Vorgangsmappen. Der Vorgangsmappen Service delegiert zur Handhabung des Laufzettels an die entsprechenden Methoden des Laufzettel Services. Die Materialien der Vorgangsmappe muss der Service selbst persistieren. Im Rahmen der Arbeit wurde eine Variante implementiert, die die Vorgangsmappen im Hauptspeicher hält.

Hervorzuheben ist auch, dass der Lösungsansatz der Vorgangsmappen keine Abhängigkeit zur Workflow Engine aufweist. Die Vorgangsmappen Implementierung benötigt nur einen Laufzettel Service. Dieser kann auf einer beliebigen Workflow Engine oder auch auf anderen Technologien arbeiten.

4.5.4 Verwendung von Laufzettel und Vorgangsmappe

Das Beispiel 4 (Verwendung des Vorgangsmappen Service) zeigt die Verwendung des Vorgangsmappen Service zur Erzeugung einer Vorgangsmappe. Anschließend wird ein Material (in diesem Fall nur eine Zeichenkette) zur Mappe hinzugefügt und der aktuelle Eintrag am Laufzettel als erledigt markiert. Beim Speichern der Vorgangsmappe wird an den Laufzettel Service delegiert, der dafür sorgt, dass in der Workflow Engine der Standard Zustandsübergang ausgelöst wird.

```

ProcessFolder folder = _pFolderService
    .createNew(ProcessNameDV
        .processNameDV("Simple Order Process"));
folder.getThings().add(new StringThing(
    "I have created the process " + folder.getProcessId()));
folder.getRoutingSlip().getActualEntry().setDone();
_pFolderService.save(folder);

```

Beispiel 4: Verwendung des Vorgangsmappen Service

Durch die Kapselung des Laufzettel Services durch den Vorgangsmappen Service muss der Laufzettel Service vom Verwender nicht benutzt werden.

4.6 Realisierung des Postkorbsystems

Es wurde nun bereits beschrieben, wie Laufzettel und Vorgangsmappe gestaltet sind und wie diese verwendet werden können. Es ist noch unklar, wie Teilschritte des Gesamtprozesses zu ihren Vorgangsmappen und somit auch zu Ihren Aufgaben kommen. Zu diesem Zweck wird nun das Postkorbsystem eingeführt.

4.6.1 Design

Die teilautomatisierten Prozessschritte sollen – wie bereits mehrmals ausgeführt – möglichst keine Kenntnis vom Gesamtprozess bzw. dessen Ablauf haben. Daher ist es nahe liegend, diese weitgehend kontextfrei voneinander zu entkoppeln. Eine solche Entkopplung lässt sich über die Postkorbmetapher gut modellieren. Der Prozessschritt selbst kann mit Hilfe einer Postkorbbezeichnung seine Vorgangsmappen abholen und diese im Ausgabefach wieder ablegen. Damit hat der Prozessschritt keinerlei Kenntnis über das Ablaufsystem. Andererseits hat auch das Ablaufsystem keine Kenntnis über den Prozessschritt, da es ja nur die Postkörbe verwalten muss.

In Abbildung 20 (Schnittstellen Postkorbsystem) sind die zentralen Schnittstellen zum Postkorbsystem dargestellt. Der Postkorb Provider stellt eine Menge von Postkörben zur Verfügung. Die Postkörbe werden über einen Namen referenziert. Der Postkorb selbst hat eine Eingangsseite und eine Ausgangsseite. In der Eingangsseite befindet sich ein Behälter mit Materialien. Auf eine Oberklasse für Nachrichten wurde verzichtet, da die Postkorbmetapher auch für allgemeine Materialien passend ist. In das Ausgangsfach können wiederum beliebig viele Materialien als Nachrichten gelegt werden. Das Synchronisieren des Postfaches bewirkt ein Senden der Nachrichten im Ausgangsfach und ein erneutes Abfragen der Eingangsnachrichten.

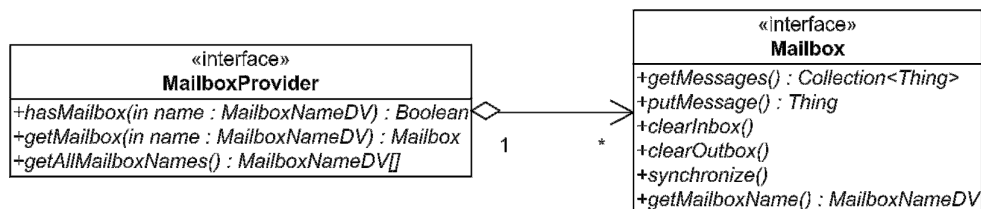


Abbildung 20: Schnittstellen Postkorbsystem

Das Postkorbsystem kann neben dem System zur Vorgangsbearbeitung potenziell auch andere Postkorbsysteme kapseln. Deshalb wird hier eine Flexibilisierung des Systems eingeführt. In Abbildung 21 (Komposition im Postkorbsystem) ist der

`MailboxProviderContainer` dargestellt. Er ist eine Realisierung des Composite-Patterns [Gamma et al, 94] und ermöglicht es, mittels eines einzigen Providers auch verschiedene Kommunikationssysteme zu integrieren, die einen Postkorb anbieten. Analog dazu wurde auch für den Postkorb selbst dieses Pattern eingesetzt, um auch für hier eine Menge an Nachrichtenquellen zuzulassen. Beim Postkorb Container ist allerdings die Besonderheit zu beachten, dass die Verteilung der ausgehenden Nachrichten für das jeweilige Kommunikationsmedium speziell erfolgen muss und daher nicht generisch implementiert werden kann.

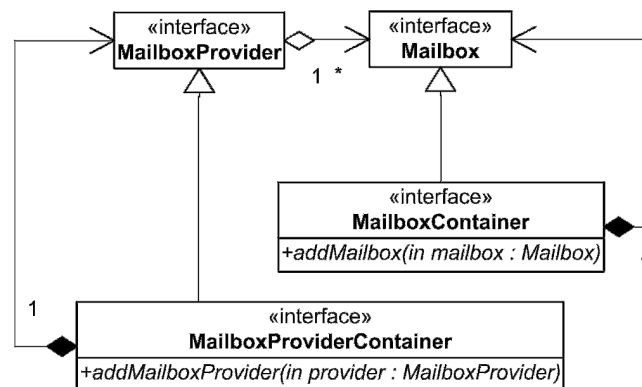


Abbildung 21: Komposition im Postkorbsystem

4.6.2 Implementierung

Im Rahmen der Arbeit ist eine jBpm basierte Implementierung erstellt worden. In der jBpm befinden sich alle Workflows zu jedem Zeitpunkt in einem definierten Zustand. Wie bereits im Kapitel 4.4.3 (Transformation in die Workflow Beschreibung) besprochen, wird jeder Arbeitsschritt in eGPM in einen Zustand der Workflow Engine übertragen. Da die einzelnen Prozessschritte mit Hilfe der Postkörbe ihre Aufgaben erhalten sollen, ist es nahe liegend, dass jeder Zustand in der Workflow Engine auch einen Postkorb repräsentiert. Der Bezeichner des Postkorbs ist dann auch der Name des Zustandes.

Somit muss der Postkorb Provider für die jBpm alle Zustände auslesen und die zugehörigen Postkörbe zur Verfügung stellen. Im Eingangsfach des Postkorbes sollen sich alle Prozesse befinden, die im korrespondierenden Zustand sind. In das Ausgangsfach des Postkorbes werden jene Prozesse (in Form von Vorgangsmappen) gelegt, die den Zustand verlassen sollen.

Für die Befüllung der Postkörbe wurde der Ansatz gewählt, dass die Postkörbe selbst, die jBpm zur Nachrichtenversorgung nutzen. Es muss also auch für die Postkörbe eine jBpm spezifische Implementierung geben. Zur Verwaltung der Vorgangsmappen und der Laufzettel wird zwar ein Vorgangsmappen Service genutzt, die jBpm wird allerdings direkt abgefragt, um alle Prozesse im korrespondierenden Zustand abzufragen.

In der jBpm können eine (beinahe) beliebige Anzahl an Prozessen verwaltet werden. Dabei ist es nahe liegend, dass auch zahlreiche namensgleiche Zustände über die Prozesse hinweg auftauchen. Um diese Zustände in einem Postkorb zusammenzufassen, wird die Komposition von Postkörben, wie sie weiter oben beschrieben ist, verwendet.

Da die `JbpmMailboxImpl` eine Referenz auf den Vorgangsmappen Service als auch auf die `jBpm Service Fabrik` benötigt, wurde die Instanziierung dieser Komponente in eine konfigurierbare Fabrik für die Postkörbe ausgelagert. Diese Fabrik wird von Postkorb Provider verwendet.

Die strukturellen Abhängigkeiten in der `jBpm` basierten Implementierung des Postkorbsystems stellt Abbildung 22 (Klassendiagramm Postkorbsystem) dar.

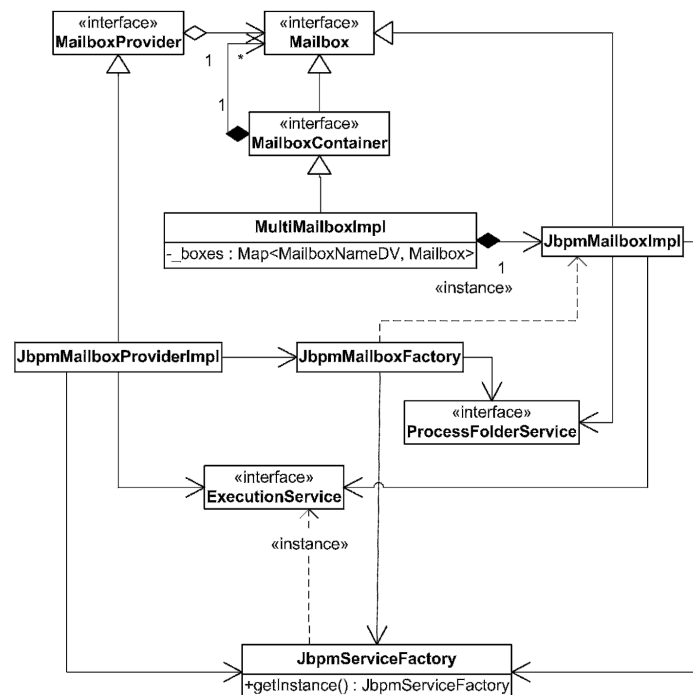


Abbildung 22: Klassendiagramm Postkorbsystem

4.6.3 Verteilung des Postkorbsystems

Wie schon die Metapher vom Postkorb nahe legt, sorgt das Postkorbsystem auch für die Verteilung des Systems über Prozessgrenzen hinweg. Mit dem bisher gezeigten Design des Postkorbsystems ergibt sich allerdings ein Problem. Der Postkorb Provider stellt seine Dienste entfernt zur Verfügung und liefert einen Postkorb aus. Da aber der Postkorb selbst die `jBpm` verwendet, muss auch der Postkorb als verteiltes Objekt realisiert sein. Somit wird aber mit jedem ausgegebenen Postkorb eine neue Verbindung zum Server aufgebaut und somit werden zur Laufzeit immer neue Service Endpunkte angelegt. Das ist zwar technisch möglich, widerspricht aber den modernen Ansätzen zur Service Architektur die zustandslos und mit fest definierten Service Endpunkten arbeiten.

Als zusätzliches Argument für die Anpassung des Designs zur Verteilung kommt hinzu, dass Spring zustandslose Services mit fest definierten Endpunkten ohne Zusatzaufwand über verschiedene Protokolle verteilen kann.

Da aber das Grunddesign des Postkorbssystems sehr komfortabel in der Handhabung ist, wird der Weg gewählt, dass die Services nur für die Verteilung angepasst werden, aber auf der Client Seite wieder auf die gleiche Art und Weise handhabbar sind.

Das Kernstück der Verteilung bildet ein Postkorb Service, der die Aufgaben des Postkorbes erfüllt, bei jeder Methode aber einen Postkorbbezeichner erhält und somit stellvertretend für beliebig viele Postkörbe verwendet werden kann. Die

Serverkomponente zur Verteilung implementiert diesen Service und delegiert die Methoden an die Postkörbe weiter. Um die Postkörbe zu erhalten, benutzt der Server wiederum einen anderen Postkorb Provider.

Der Server ist gleichzeitig auch Postkorb Provider. Er liefert Postkorb Implementierungen aus, die ihre Methoden auf einen Postkorb Service delegieren. Die Server Komponente wird über das Spring Framework ohne Änderungen am Code rein deklarativ verteilt. Die zugehörige Client Komponente erhält zwei Referenzen auf das verteilte Serverobjekt. Die erste Referenz erhält es in der Rolle als Postkorb Provider die zweite Referenz in der Rolle als Postkorb Service.

Die Client Komponente ist ebenfalls ein Postkorb Provider und delegiert an die entfernte Serverkomponente. Diese liefert eine Instanz der delegierenden Postkorbimplementierung aus. Am Client wird der Postkorb mit einer Referenz auf den entfernten Postkorb Service versorgt. Diese Zusammenhänge sind in Abbildung 23 (Klassendiagramm Verteilung) übersichtlich dargestellt.

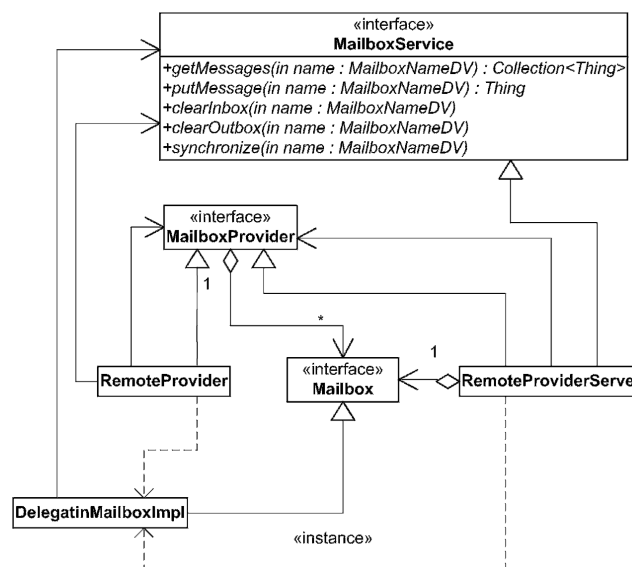


Abbildung 23: Klassendiagramm Verteilung

Durch ein Redesign des ursprünglichen Postkorbsystems hätte diese strukturell recht komplexe Architektur vereinfacht werden können. Allerdings konnte durch dieses Beispiel gezeigt werden, dass auch eine zustandsbasierte Service Architektur mit einer unbestimmten Anzahl an Service Endpunkten durch objektorientierte Methoden auf eine flache, zustandslose Service Architektur abgebildet werden kann. Dies stellt eine sehr interessante Erkenntnis dieser Arbeit dar, obwohl es sich dabei nicht um das Kernthema handelt.

4.6.4 Verwendung des Postkorbsystems

Im Beispiel 5 (Verwendung des Postkorbsystems) ist ein einfaches Beispiel für die Bearbeitung eines Postkorbes gezeigt. Nachdem der Postkorb vom Provider geholt wurde, können die Nachrichten abgefragt werden. Die Nachrichten werden einzeln bearbeitet und anschließend in das Ausgangsfach gelegt, um die Änderungen wieder an das System zurückzumelden. Die tatsächliche Rückführung der Nachrichten im Ausgangsfach passiert aber erst beim Synchronisieren des Postkorbes.


```
_mailbox = _mailboxProvider.getMailbox(MailboxNameDV.mailboxNameDV(mailboxName));
Collection<Thing> messages = _mailbox.getMessages();
for (Thing mess : messages)
{
    ProcessFolder folder = (ProcessFolder) mess;

    // doSomething with folder

    _mailbox.putMessage(folder);
}
_mailbox.synchronize();
```

Beispiel 5: Verwendung des Postkorbsystems

4.7 Realisierung von automatischen Prozessschritten

Im Vorfeld wurde das Gesamtsystem bereits ausführlich beschrieben, sodass nun auch die eigentliche Aufgabe – die Ausführung von automatischen Prozessschritten – in Angriff genommen werden kann. Aufgrund der umfangreichen Infrastruktur, die vom Gesamtsystem geboten wird, kann man sich dabei ganz auf die fachlichen Anforderungen konzentrieren.

4.7.1 Design

In einem teilautomatisierten System werden die Einzelaufgaben von Automaten ausgeführt. Die Automaten überprüfen zyklisch, ob neue Arbeit angefallen ist und führen diese aus. Die Einzelaufgaben in einem teilautomatisierten Prozess können sehr vielfältig sein. In der Praxis beschäftigen sich die meisten Schritte mit Import / Export von Daten, Versenden von E-Mails, Erstellen von Berichten, Prüfung mittels Drittsystem, etc. Aufgrund der Vielfältigkeit der Aufgaben und der dabei verwendeten Bibliotheken sollte im Rahmen der Ablaufsteuerung keine Einschränkung für die Einzelaufgaben erfolgen.

Durch die Verwendung des Postkorb-Systems konnte genau diese Trennung erreicht werden. Jede beliebige Komponente kann auf das Postkorbsystem zugreifen. Es muss von keinen abstrakten Klassen geerbt werden und es müssen auch keine Interfaces implementiert werden. Somit entsteht bei der Implementierung von Prozessschritten eine reine Verwendungsbeziehung zum Postkorbsystem und in weiterer Folge zur Vorgangsmappe und zum Laufzettel.

4.7.2 Implementierung

Im Rahmen der Implementierung hat sich gezeigt, dass die Einzelaufgaben immer eine sehr ähnliche Struktur aufweisen und daher durch eine gemeinsame Abstraktion ihre Implementierung vereinfacht werden kann.

Im Rahmen des Beispielsprojektes wurden alle Einzelaufgaben zyklisch ausgeführt. Für die Planung und Steuerung der Einzelaufgaben wird die Java Klasse `Timer` verwendet. Diese Klasse bietet einen einfachen Mechanismus zur Planung von Aufgaben. Dazu muss die Klasse `TimerTask` beerbt werden. In einer Produktionsumgebung wird vermutlich ein professionelles Werkzeug wie z.B. Quartz [Quartz, 05] zur Steuerung eingesetzt.

Der `AbstractProcessTask` erbt vom `TimerTask` und bietet bereits einen Postkorb Provider und einen Vorgangsmappen Service an und bietet eine Template Method [Gamma et al, 94] an. Darüber hinaus übernimmt diese Klasse auch erste Logging Aufgaben. Da viele der Einzelaufgaben immer den Postkorb abrufen und jede

Vorgangsmappe extra behandeln, wurde auch dafür eine abstrakte Klasse `AbstractMailboxUsingTask` entwickelt. Diese Klasse erhält im Konstruktor den Namen des Postkorbes und ruft dann für jede Vorgangsmappe wiederum eine *Template Method* auf. Nachdem alle Mappen bearbeitet wurden, wird die Mailbox synchronisiert. Die Aufrufhierarchie ist in Abbildung 24 (Sequenzdiagramm Tasks) graphisch dargestellt.

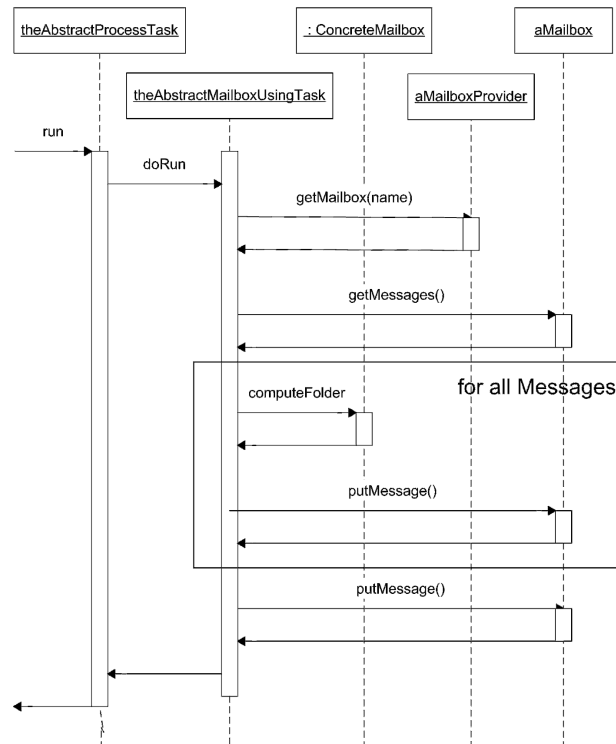


Abbildung 24: Sequenzdiagramm Tasks

Durch die oben beschriebenen Abstraktionen wird die Implementierung von einzelnen Aufgaben sehr einfach. Das Beispiel 6 (Implementierung eines Teilschrittes) zeigt einen Prozessschritt, der abhängig von einer Zufallszahl den Prozess weiterführt oder an die Klärungsstelle mit einer entsprechenden Nachricht leitet.

```

protected ProcessFolder computeFolder(ProcessFolder folder)
    throws ServiceException
{
    if (Math.random() > 0.7)
    {
        LOG.debug("Process " + folder.getProcessId() + ": check okay.");
        folder.getRoutingSlip().getActualEntry().setMessage("The order is okay.");
        folder.getRoutingSlip().getActualEntry().setDone();
    }
    else
    {
        LOG.debug("Process " + folder.getProcessId() + ": check failure.");
        folder.getThings().add(new StringThing(
            "This order is not okay. It needs to be cleared."));
        folder.getRoutingSlip().getActualEntry().
            setMessage("This order is not okay. It needs to be cleared.");
        folder.getRoutingSlip().setNextRecp("Clearing");
        folder.getRoutingSlip().getActualEntry().setDone();
    }
    return folder;
}
  
```

Beispiel 6: Implementierung eines Teilschrittes

4.8 Realisierung einer Klärungsstelle

Die Klärungsstelle ist für Prozesse zuständig, die nicht dem typischen Prozessablauf folgen. Solche Situationen können sehr unterschiedliche Ursachen haben. Es ist vorstellbar, dass der Prozess mit nicht plausiblen Daten versorgt wurde oder dass technische Fehler im System aufgetreten sind, die eine weitere Bearbeitung des Prozesses unmöglich machen.

Solche Situationen sind im Vorfeld nur schwer abschätzbar und somit auch nur schwer modellierbar. Umso wichtiger ist es, dass die Klärungsstelle den Prozess mit einem hohen Maß an Flexibilität beeinflussen kann. Um den Prozess zu bearbeiten, benötigt der Benutzer ein Werkzeug. Wie ein solches Werkzeug gestaltet sein kann, beschreibt dieses Kapitel.

4.8.1 Benutzungsmodell

Wie bereits im Kapitel 4.4.1 (Zusammenführung von ablaufsteuernder und unterstützender Sichtweise) beschrieben, bietet die Klärungsstelle Unterstützung bei der Kooperation an. Es soll also keine Ablaufsteuerung mit streng definierten Rahmenbedingungen erfolgen, sondern ganz im Gegenteil soll dem Benutzer ein Werkzeug zur qualifizierten, selbstständigen Arbeit angeboten werden. Dieser Ansatz deckt sich auch mit dem Gedanken von WAM, dass die Benutzer eines Systems die Werkzeuge für ihre Arbeit benutzen und nicht ihre Arbeit vom Werkzeug vorgegeben bekommen.

Das Werkzeug für die Klärungsstelle muss also alle notwendigen Informationen über den Prozess liefern und die Möglichkeit zur flexiblen Bearbeitung bieten. Die Abbildung 25 (Werkzeug für die Klärungsstelle) zeigt den Prototypen für das Werkzeug der Klärungsstelle. Dieses Werkzeug ist nicht für einen Produktiveinsatz gedacht, sondern es soll nur die wichtigsten Grundelemente eines solchen Werkzeuges darstellen.

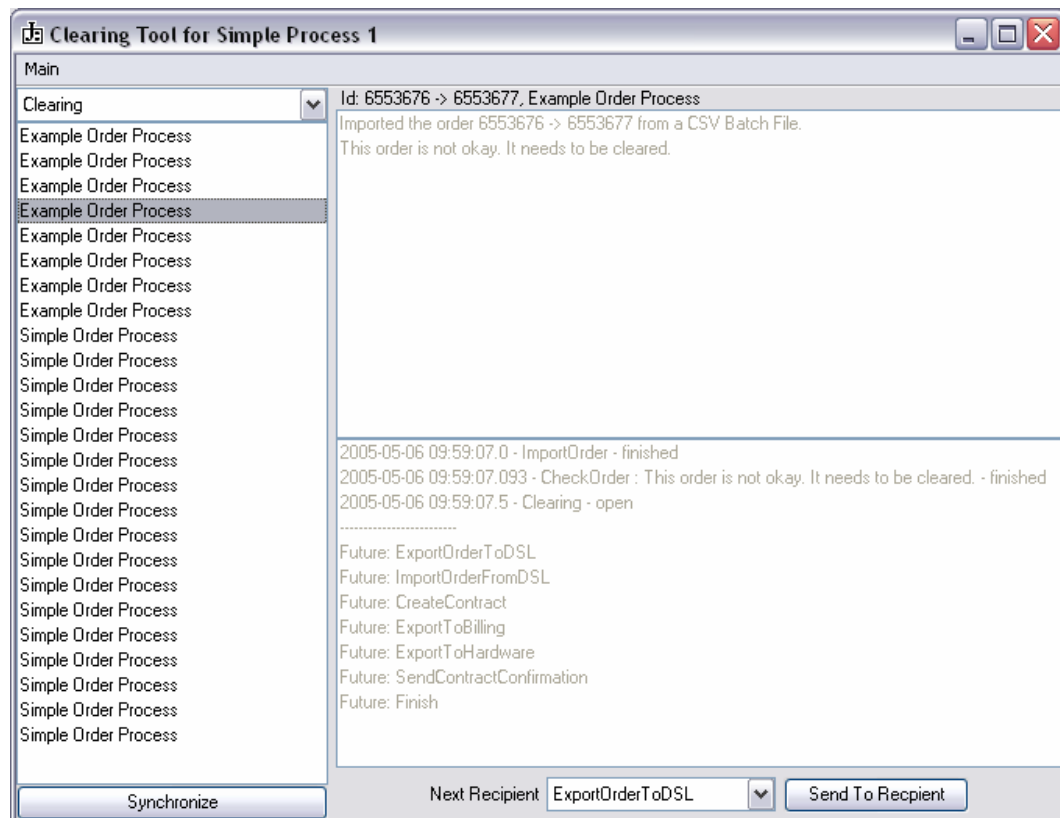


Abbildung 25: Werkzeug für die Klärungsstelle

Da die Daten der Klärungsstelle genau wie die anderen Zustände auch über einen Postkorb angeboten werden, kann das Werkzeug auch die Daten anderer Arbeitsschritte abfragen. Der Postkorb kann über das Auswahlfeld links oben gewählt werden. In der Liste darunter werden alle Vorgangsmappen angezeigt, die sich im gewählten Postkorb befinden. Wählt man eine Vorgangsmappe aus, wird diese im rechten Bereich des Werkzeuges angezeigt. Da im Prototyp die Vorgangsmappen neben dem Laufzettel einfach eine Sammlung an Zeichenketten enthalten, reicht zur Anzeige der Materialien ein Listenfeld aus. Die Materialien sind hier nicht bearbeitbar, da die Bearbeitung der Vorgangsmappe nicht Kernthema dieser Arbeit ist. Das untere Listenfeld zeigt die Daten des Laufzettels. Mit der Auswahlbox „Next Recipient“ wird der Zustand bzw. der Postkorb gewählt, an dem die Vorgangsmappe als nächster geleitet werden soll. In Tabelle 3 sind die Anforderungen an ein Werkzeug für die Klärungsstelle noch einmal zusammengefasst.

	informativ	veränderbar
Wahl des Postkorbes		✓
Vorgangsmappen im Postkorb	✓	
Daten der Vorgangsmappe	✓	
Materialien in der Vorgangsmappe		✓
Daten des Laufzettels	✓	
Laufzetteleintrag und nächster Empfänger		✓

Tabelle 3: Möglichkeiten der Klärungsstelle

4.8.2 Design

Das prototypische Werkzeug wurde nach dem Prinzip der Werkzeugkonstruktion von WAM gebaut. Dabei wurden auch die von JWAM zur Verfügung gestellten Hilfsmittel zur Werkzeugkonstruktion verwendet.

Das Gesamtwerkzeug wurde aus zwei Subwerkzeugen komponiert. Das `ProcessListTool` dient zum Auswählen des Postkorbs und zur Anzeige der Vorgangsmappen. Das `ProcessForwardTool` zeigt die Vorgangsmappe an und leitet diese an den nächsten Empfänger weiter. Diese beiden Werkzeuge werden vom `ClearingTool` zusammengefasst und koordiniert.

In Abbildung 26 (Klassendiagramm Klärungsstelle) ist die Struktur des Werkzeuges dargestellt. Da eine ausführliche Behandlung der Werkzeugkonstruktion nach WAM bzw. mit Hilfe von JWAM den Rahmen sprengen würde, kann hier nur auf [Züllighoven et al, 04] für die theoretischen Grundlagen und auf [JWAM, 05] für praktische Beispiele verwiesen werden. Ein weiteres ausführliches Beispiel zur Werkzeugkonstruktion findet sich unter [EMS, 05].

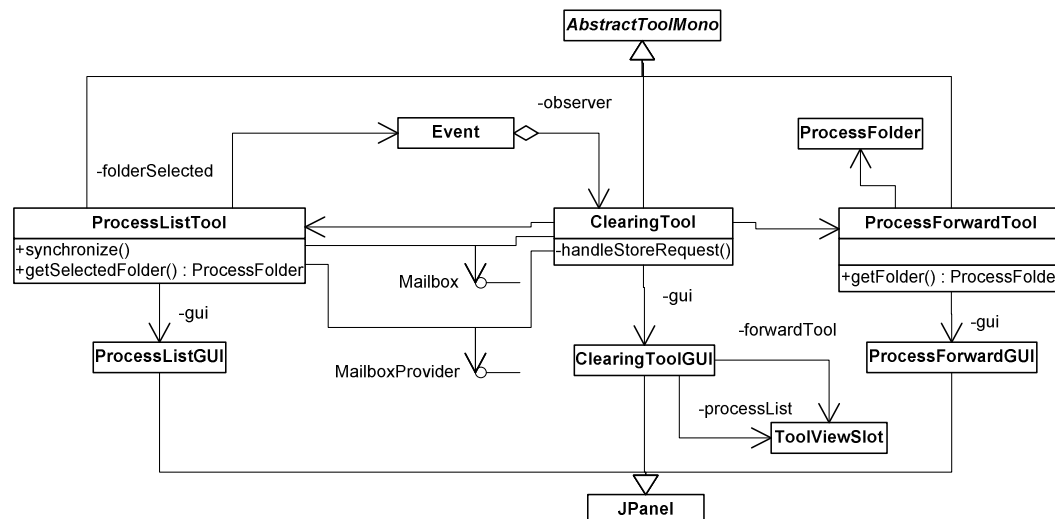


Abbildung 26: Klassendiagramm Klärungsstelle

Besonders zu beachten bei der Gestaltung des Prototypen ist, dass die Abhängigkeit zum Workflow System ausschließlich über die Service Interfaces von Postkorb Provider und Postkorb besteht. Zusätzlich bestehen natürlich Abhängigkeiten zu den Materialklassen und Interfaces von Vorgangsmappe und Laufzettel. Da auch die Architektur zur Verteilung des Postkorbsystems diese Interfaces anbietet, kann das Werkzeug nur durch den Austausch der Interface Implementierung verteilt eingesetzt werden. Durch die Verwendung des Spring Frameworks erfolgt die Erstellung der RMI Kapselung automatisch durch Einsatz von Aspekt orientierten Technologien. Somit bleiben für die Verteilung des Werkzeuges zwei Arbeitsschritte übrig:

1. Konfiguration des Spring Frameworks, mit der Angabe der RMI Objekte, Server und Interfaces
2. Erstellung einer Startklasse, die das Spring Framework und die JWAM Umgebung initialisiert und das Werkzeug startet.

Die Implementierung des Werkzeuges beinhaltet keine wesentlichen Besonderheiten. Daher wird an dieser Stelle auf Details zur Implementierung verzichtet.

4.8.3 Weitere Möglichkeiten

Zur Darstellung der Klärungsstelle wurde aus Aufwandsgründen bewusst nur ein sehr einfacher Prototyp gebaut. Schon bei der Erstellung dieses Prototyps und auch bei den weiteren Analysen sind noch eine Vielzahl von Möglichkeiten zur Verbesserung des Werkzeuges aufgefallen. Die interessantesten Ideen sind hier nochmals zusammengefasst.

Angabe von Bemerkungen für den Laufzetteleintrag

Beim Weiterleiten der Vorgangsmappe an den nächsten Empfänger, sollte auch die Möglichkeit vorgesehen werden, einen Kommentar einzutragen. Diese Funktionalität stellt der Laufzettel auch bereits zur Verfügung.

Erweiterte Suchmöglichkeiten

Der Benutzer sollte die Vorgangsmappen anhand verschiedener Suchkriterien abfragen können. Mögliche Suchkriterien sind die Vorgangsnummern, Stichworte in den Bemerkungen, vorangegangene Empfänger, Alter der Mappen, etc.

Prozess Editoren

In einer Klärungsstelle können unterschiedliche Vorgangsmappen einlaufen. Die Vorgangsmappen beinhalten jeweils andere Materialien und können nicht im gleichen Werkzeug dargestellt werden. Daher sollte jeder Typ von Vorgangsmappen auch eigene Werkzeuge zur Materialbearbeitung anbieten. Die Werkzeuge zur Laufzettel und Postkorbanzeige können für alle Vorgangsmappen gleich sein.

Adapter für Prozesslisten

In der Prozessliste wurde bisher nur der Vorgangsname angezeigt. Liegen viele gleiche Vorgänge im Postkorb ist diese Liste wenig hilfreich. Wenn jeder Vorgang einen Adapter anbietet, der die wichtigsten Daten des Vorganges in einer Zeichenkette zusammenfasst, kann diese Liste viel informativer gestaltet werden.

Herausheben von wiederholten Problemen

Für spezielle Situationen könnten die Vorgänge gesondert hervorgehoben werden. Dazu können die Vorgänge sondierende Methoden anbieten, die eine solche Analyse durchführen. Damit könnte man etwa Vorgänge schnell erkennen, die bereits mehrmals durch die Klärungsstelle gelaufen sind oder man könnte Vorgänge erkennen, bei denen Arbeitsschritte schon mehrmals erfolglos durchgeführt wurden.

PlugIn - Konzepte

Um die jeweils spezifischen Teile der Vorgänge flexibel in das Werkzeug einbinden zu können, empfiehlt sich die Verwendung eines PlugIn Konzeptes. So ist es möglich das Werkzeug bei Einführung von neuen Vorgängen flexibel zu erweitern. Hier könnte zum Beispiel die Eclipse Rich Client Plattform [RCP, 05] als technologische Basis dienen.

5 Musterkatalog zur Transformation eGPM – Laufzettel

Der Transfer der Modellartefakte aus den Kooperationsbildern der eGPM in die zustandsbasierte Beschreibungssprache jPdl der Workflow Engine ist ein kreativer und nicht automatisierbarer Schritt.

In diesem Arbeitsschritt wird ein mit Fachanwendern erstelltes Modell über deren Kooperation mit der Umwelt in eine softwaretechnische Repräsentation durch Zustandsautomaten transferiert. Im Zuge der Transformation können durch die Anforderung von Seiten einer softwaretechnischen Lösung zahlreiche Änderungen notwendig werden. Die Änderungen können unter anderem umfassen:

1. Die schrittweise Verfeinerung von Arbeitsschritten, zu einzelnen softwaretechnisch sinnvollen Arbeitseinheiten
2. Die Einführung von Arbeitsschritten, die im Kooperationsmodell nur implizit auftauchen
3. Das Entfernen von Arbeitsschritten, die das Softwaresystem nicht betreffen
4. Das Einführen von Fallunterscheidungen bei der Zusammenführung von Kooperationsmodellen
5. Das Identifizieren von wieder verwendbaren Arbeitsschritten

Treten im Zuge der Überführung in die Zustandsautomaten Inkonsistenzen in den Kooperationsmodellen auf oder der Prozessablauf ist durch die Vor- und Nachbedingungen der Arbeitsschritte nicht ausführbar, so ist unbedingt eine erneute Modellierung der Prozesse anzuraten. Eine Korrektur von Prozessen bei der Überführung in den Zustandsautomaten stellt eine nicht explizite Änderung der Software dar und verletzt die mit dem Fachanwender vereinbarten Prozesse bei der Prozessmodellierung. Es ist also darauf zu achten, dass der durch das Zustandsmodell beschriebene Workflow auch tatsächlich eine Implementierung des modellierten Prozesses darstellt.

Bei der Modellierung mit eGPM zeigt sich, dass sich viele Strukturen in den Modellen wiederholen, so wie sich auch in der Realität gewisse Ablaufmuster und Kooperationsmuster wiederholen. Im Rahmen des Musterkataloges sind einige dieser Muster aufgeführt und eine mögliche Abbildung in die Workflow Beschreibung dargestellt. Somit soll der Musterkatalog – vor allem beim Einsieg in die Transformation – Unterstützung bieten.

Die nachfolgenden Muster werden einerseits durch die entsprechenden Ausschnitte aus den eGPM Kooperationsbildern und durch Zustandsdiagramme beschrieben. Auf die Darstellung der jPdl konformen XML Notation wird zumeist verzichtet, da der Übergang vom Zustandsdiagramm zur XML Beschreibung trivial und automatisierbar ist.

Einfache Zustandsabbildung

Die Abbildung von Arbeitsschritten des Kooperationsmodells erfolgt – wie bereits angedeutet – durch die Modellierung von Zuständen im Workflow. Dabei wird für jeden Arbeitsschritt der korrespondierende Zustand erstellt. Die Benennung des Zustandes nach dem Arbeitsschritt hat sich bewährt. Die Abbildung 27 (Einfache Zustandsabbildung) zeigt die Übertragung von Arbeitsschritten in Zustände.

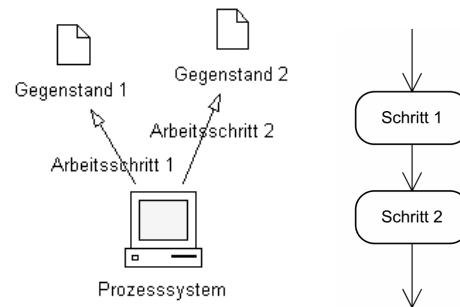


Abbildung 27: Einfache Zustandsabbildung

Prozessstart

Im Kooperationsbild soll sich für die Erzeugung des Prozesses ein fachlicher Arbeitsschritt finden. In diesem Arbeitsschritt muss dann auch die Vorgangsmappe und somit der Laufzettel und der Prozess selbst erzeugt werden. Dazu wird der Erzeugungsschritt auch im Workflow modelliert. Wird nun der Workflow erzeugt, wird er automatisch in diesen Zustand überführt. Dann kann der erzeugende Arbeitsschritt wie bei allen anderen Arbeitsschritten auch, den Abschluss der Aufgabe bekannt geben. Der Workflow wird in den Schritt 1 überführt. Abbildung 28 (Prozessstart) stellt den Sachverhalt graphisch dar.

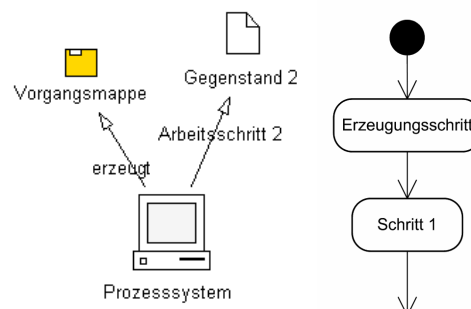


Abbildung 28: Prozessstart

Prozessende

Am Ende eines Prozesses wird der letzte Arbeitsschritt als Zustand im Workflow dargestellt, wie auch in Abbildung 29 (Prozessende) gezeigt. Der Endzustand der jPdI sollte dafür nicht verwendet werden. Dieser Endzustand scheint im Kooperationsmodell nicht auf.

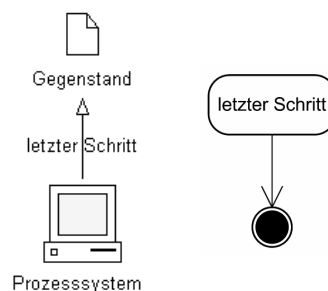


Abbildung 29: Prozessende

Fallunterscheidung

Da die Fallunterscheidungen in den Kooperationsmodellen gar nicht aufscheinen, wird in Abbildung 30 (Fallunterscheidung) kein Kooperationsmodell dargestellt. Zur Realisierung von Fallunterscheidungen in der jPdl stehen zwei Möglichkeiten zur Verfügung. Die erste Variante trifft die Entscheidung in der Workflow Engine und muss als JAVA Klasse, die ein spezielles Interface implementiert, abgebildet werden.

Die zweite Möglichkeit sieht zwei (oder mehr) Zustandsübergänge aus einem Entscheidungsschritt heraus vor. Mit dieser Möglichkeit entscheiden also die Einzelaufgaben über den weiteren Verlauf und können die Entscheidung über den Laufzettel der Workflow Engine mitteilen.

Für fachliche Entscheidungsknoten ist die zweite Vorgehensweise mit Entscheidungsschritt zu bevorzugen, da damit die fachliche Implementierung auch nur über die gewohnten Mechanismen auf das Laufzeit System zugreift.

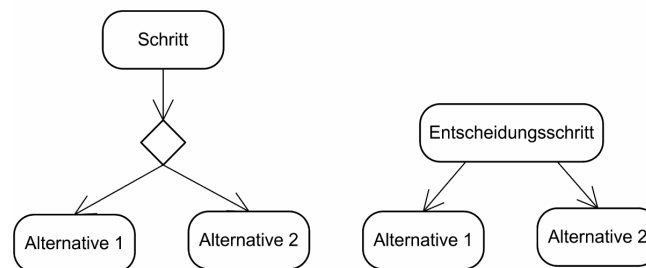


Abbildung 30: Fallunterscheidung

Batch Schnittstelle

Ein sehr typisches Muster für die Koordination von externen Dienstleistern ist die Batch Schnittstelle, die in Abbildung 31 (Batch Schnittstelle) dargestellt ist. In Batch Schnittstellen werden Daten an ein externes System (oder ein externes Unternehmen) exportiert und anschließend daran wird auf eine Antwort des externen Systems gewartet. Dieser Sachverhalt wird im Workflow durch zwei aufeinander folgende Zustände abgebildet. Soll nach dem Export bzw. nach dem Import noch eine Quittierung erfolgen, empfiehlt sich die Einführung von zwei weiteren Zuständen für diese Aufgaben.

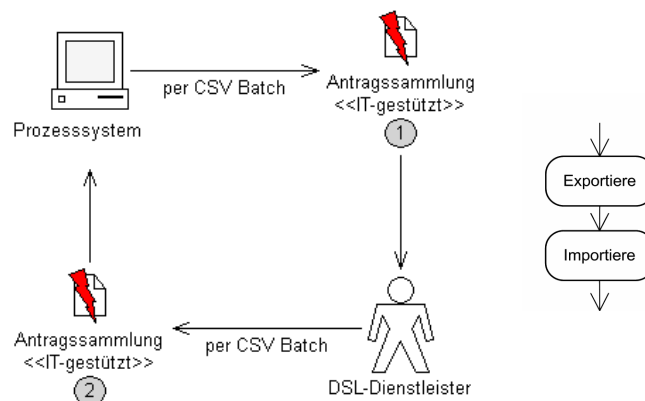


Abbildung 31: Batch Schnittstelle

6 Bewertung

Nach der ausführlichen Vorstellung eines Konzeptes zur Realisierung von teilautomatisierten Prozessen durch eine Kombination aus Ablaufsteuerung und Kooperation beschäftigt sich dieses Kapitel mit der Bewertung des Konzeptes. Besonders dieser Teil der Arbeit spiegelt in vielen Punkten die Meinung des Autors wider.

6.1 Bewertung des Konzeptes

Die Stärke von realen Ablauforganisationen ist die streng geregelte Durchführung von Routinetätigkeiten unter Beibehaltung der Fähigkeit zur flexiblen Reaktion auf unvorhergesehene Entwicklungen und Situationen. Vor allem die Tendenz zur schnellen und flexiblen Reaktion auf Kundenwünsche erfordert ein hohes Maß an Flexibilität einer Ablauforganisation.

Werden nun die Routinetätigkeiten durch ein IT-System automatisiert abgewickelt oder der Benutzer wird in seiner Tätigkeit ablaufgesteuert besteht die Gefahr, die Flexibilität der Ablauforganisation zu verlieren. Daher bietet die Idee, zur Realisierung von Prozessen eine Kombination aus Ablaufsteuerung und computerunterstützter Kooperation zu verwenden, eine gute Möglichkeit die Realität wiederzugeben und die Ablauforganisation zu unterstützen.

In diesem Zusammenhang wird auch deutlich, dass eine solche Denkweise direkte Auswirkungen auf die Unternehmenskultur hat. Ein System, dass die Benutzer in Ihrer Kooperation unterstützt, geht von einem eigenverantwortlich handelnden Benutzer aus. Diese Sichtweise widerspricht der tayloristischen Sichtweise von stark geregeltem und arbeitsteiligem Handeln. Es widerspiegelt vielmehr die Sichtweise eines selbstverantwortlich handelnden Mitarbeiters [Sprenger, 02].

Wird dem Benutzer diese Eigenverantwortung nicht übertragen, verliert vor allem die Klärungsstelle ihre Funktion. Soll die Interaktion der Benutzer bei der Klärung von Sonderfällen ablaufgesteuert werden, so müssen diese Klärungen auch modelliert und die notwendigen Werkzeuge implementiert werden. Im Rahmen des begleitenden Beispiels hat sich etwa die Situation ergeben, dass man den Mitarbeitern des Call Centers diese Fähigkeit nicht zusprechen wollte. Somit war die Bedeutung der Klärungsstelle auf die Behandlung technischer Fehler beschränkt. Dies hat zu massiven Mehraufwand bei der Implementierung der Prozesse geführt, wodurch ein wesentlicher Vorteil des beschriebenen Ansatzes verloren gegangen ist.

6.2 Bewertung der Realisierung

Schon im Vorfeld der Arbeit war klar, dass die Verwendung einer Workflow Engine möglicherweise problematisch sein wird. Im Zuge der Realisierung haben sich tatsächlich auch einige Fragestellungen aufgrund dieser Entscheidung ergeben. Allerdings wäre eine vollständige Eigenentwicklung aus Aufwandssicht dennoch nicht sinnvoll gewesen.

Vor allem das streng definierte Zustandsmodell der Workflow Engine hat bei der Realisierung der Klärungsstelle umfangreiche Zusatzimplementierung erfordert. Ein Zustandsautomat, der zwar den Zustandsraum überwacht, aber dennoch Abweichungen der definierten Zustandsübergänge erlaubt, wäre hier sinnvoll gewesen. Allerdings würde

ein solches Vorgehen der Theorie der Petri-Netze widersprechen und somit die theoretischen Grundlagen einer Workflow Engine aufweichen.

Vor allem eine höhere Flexibilität in der Prozessgestaltung zur Laufzeit (entsprechend dem Konzept der Prozessmuster) wäre wünschenswert. So sollte die Klärungsstelle nicht nur den nächsten Prozessschritt, sondern auch darüber hinaus den Prozess beeinflussen können.

6.3 Beurteilung der Performanz

Aufgrund der Beschreibung des Gesamtsystems ist bereits deutlich geworden, dass das System auf Server Infrastruktur ausgelegt ist. Da im Rahmen der Entwicklung aber keine entsprechende Hardware zur Verfügung stand, konnte die Performanz des Systems nicht in einem realistischen Umfeld getestet werden.

Dass die Laufzeit zur Abwicklung von einzelnen Prozessen akzeptabel ist, hat bereits die Testumgebung gezeigt. Es stellt sich allerdings die Frage, ob das System auch für eine größere Anzahl von Prozessen akzeptabel skaliert werden kann. Aus dieser Motivation heraus wurden Testfälle entwickelt, die die Anzahl der bearbeiteten Prozesse schrittweise erhöhen, um eine Abschätzung des Laufzeitverhaltens in Abhängigkeit der Prozessanzahl zu erhalten.

Es wurden zwei unterschiedliche Performanztests implementiert. Im ersten Testfall werden die Prozesse immer im gesamten Block bearbeitet (blockorientiert) und im zweiten Testfall werden die Prozesse jeweils einzeln erzeugt und abgearbeitet (Einzelprozess). Im Beispiel 7 (Pseudo Code Performanz Test) sind die Performanztests in Pseudo Code beschrieben.

```
testBlockorientiert (int anzahl)
{
    erzeugeProzesse (anzahl);
    überführeInZustand2 (anzahl)
    überführeInZustand3 (anzahl)
}

testEinzelprozess (int anzahl)
{
    for (int i=0; i<anzahl; i++)
    {
        erzeugeProzesse (1);
        überführeInZustand2 (1)
        überführeInZustand3 (1)
    }
}
```

Beispiel 7: Pseudo Code Performanz Test

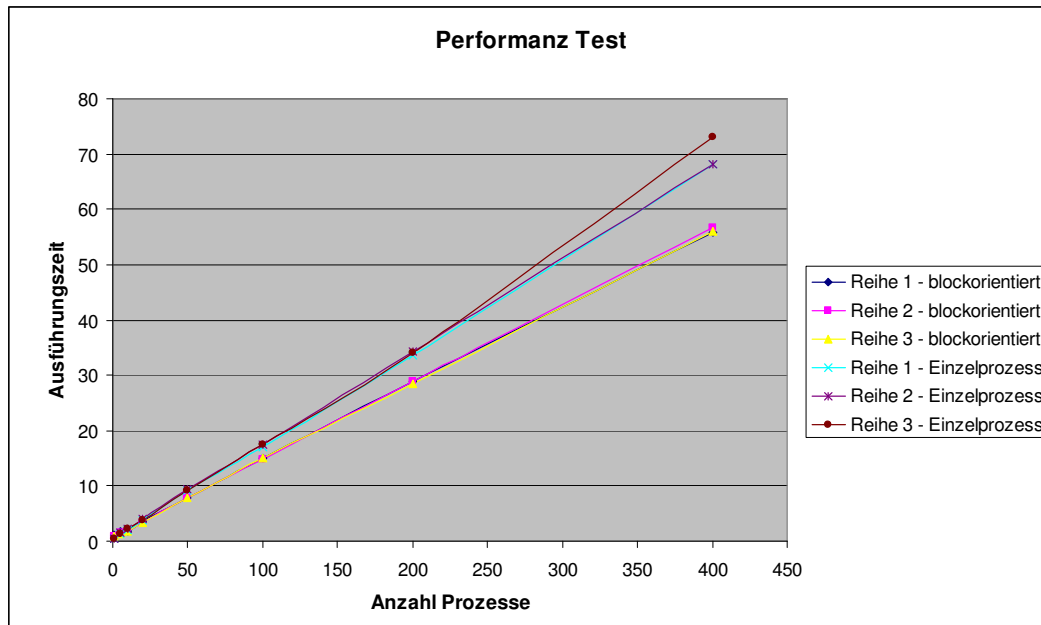


Abbildung 32: Performanz Test

Abbildung 32 (Performanz Test) zeigt die Ergebnisse des Performanztests in drei Testreihen für eine unterschiedliche Anzahl an Prozessen. Die Tests wurden dabei auf einen Intel Celeron System mit 2,6 GHz Prozessor und 496 MB RAM durchgeführt. Als Betriebssystem kam Windows XP Home Edition mit Java 1.5 JRE zum Einsatz. Das System wurde in Eclipse 3.1M6 ausgeführt und nutzte eine InProcess-HSQL Datenbank. Daran wird deutlich, dass die absoluten Ausführungszeiten keinerlei Vergleichsmöglichkeiten zu produktiven Serverumgebungen zulassen. Das Hauptaugenmerk wurde auf die Laufzeitentwicklung in Abhängigkeit der Prozessanzahl gelegt.

Bei beiden Testfällen ist eine annähernd lineare Laufzeitentwicklung über die Prozessanzahl zu beobachten. Die Ausführungszeiten entwickeln sich im Fall der blockorientierten Testmethode etwas günstiger als im Einzelprozess orientierten Fall. Diese Ergebnisse sind sehr zufrieden stellend und lassen eine etwa lineare Skalierbarkeit des Systems erwarten.

6.4 Bewertung des Transformationsvorganges

Der Transformationsvorgang vom Kooperationsbild zur Workflow Beschreibung hat sich im Rahmen der Beispiele, die hier behandelt wurden, äußerst unproblematisch dargestellt. Allerdings ist in diesem Zusammenhang auch zu bedenken, dass im Zuge der Beispiele bereits ausgeprägtes Prozesswissen vorhanden war und auch die Modellierung des Prozesses und die Überführung in die Workflow Engine von derselben Person durchgeführt wurden. Inwieweit eine Transformation in die Workflow Beschreibung rein nach Vorlage der Kooperationsbilder möglich ist, müssten entsprechende Versuche zeigen. Es darf allerdings die praktische Machbarkeit durchaus bezweifelt werden.

Als viel größere Herausforderung hat sich Prozessmodellierung selbst und begleitend die Erstellung der Materialien und Schnittstellen herausgestellt. Der Prozessablauf wurde oft revidiert und damit verbunden auch die benötigten Daten. Diese Änderungen haben

wiederum bewirkt, dass sich Schnittstellendefinitionen verändert haben. Dadurch wird auch wieder die Wichtigkeit der Entkoppelung der verschiedenen Aspekte herausgestrichen. Wenn das Ablaufsystem bereits entwickelt werden kann, ohne die Schnittstellen zu kennen, müssen diese auch erst sehr viel später eingefroren werden.

6.5 Beurteilung gegenüber der Anforderungen

Wie in Kapitel 3.2 (Anforderungskatalog) eingeführt, werden nun die Anforderungen aus dem Katalog nochmals aufgeführt und kurz besprochen. Die Tabelle 4 (Erfüllung der Anforderungen) zeigt eine übersichtliche Bewertung aller Anforderungen aus der Sicht des Autors.

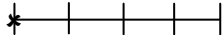
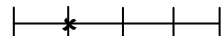
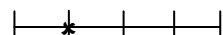
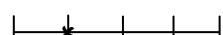
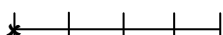
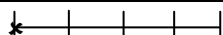
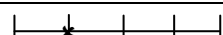
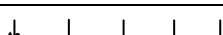
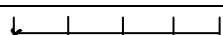
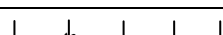
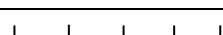
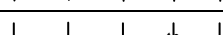
	voll erfüllt	nicht erfüllt
Einfachheit der Modellierung		
Wartungsfreundlichkeit		
Rollentrennung		
Einfache Schnittstellen		
Kapselung der Workflow Engine		
Kapselung externer Schnittstellen		
Testbarkeit		
Serviceorientierung		
Komponentenorientierung		
Historisierung		
Logging		
Überwachung		

Tabelle 4: Erfüllung der Anforderungen

Einfachheit der Modellierung

Die Modellierungssprache jPdl ist sehr einfach aufgebaut und ist mit einem Einarbeitungsaufwand von maximal einem Tag gut beherrschbar. Ein graphischer Editor ist leider erst für Version 3 der jBpm verfügbar. Zum Einsatz der Version 3 müssten aber die entsprechenden Serviceimplementierungen angepasst werden.

Wartungsfreundlichkeit

Änderungen im Ablauf können rein in der Beschreibungssprache erfolgen. Natürlich muss darauf geachtet werden, dass die Einzelaufgaben, die auf den Postkörben aufbauen, auch nach wie vor konsistent zum Ablauf sind.

Wird ein Prozess verändert, so wird diesem eine neue Versionsnummer zugewiesen. Die laufenden Prozesse laufen in der alten Version weiter.

Rollentrennung

Die Rollentrennung kann durch die Benutzung der Postkörbe gut bewerkstelligt werden. Für die Implementierung von Einzelaufgaben ist keine Kenntnis der Workflow Engine notwendig.

Einfache Schnittstellen

Die Schnittstellen zum Ablaufsystem sind unbeeinflusst von der Workflow Engine. Es ist sogar vorstellbar die Backend Technologie vollständig zu ersetzen.

Kapselung der Workflow Engine

siehe „Einfache Schnittstellen“

Kapselung externer Schnittstellen

Das Ablaufsystem ist völlig von externen Schnittstellen getrennt. So können zu Testzwecken beliebige Implementierungen verwendet werden. Dieses Konzept wurde auch während der Entwicklung eingesetzt.

Testbarkeit

Die Schnittstellen können auch unabhängig vom Ablaufsystem getestet werden. Es sind dazu allerdings Mock Implementierungen für das Postkorbsystem bereitzustellen. Weiters muss dafür gesorgt werden, dass die Vorgangsmappen den Tests entsprechend vorbereitet werden.

Serviceorientierung

Das System wurde vollständig service-orientiert entwickelt.

Komponentenorientierung

Bereits in der Beschreibung der Gesamtarchitektur hat sich die Komponentenstruktur widerspiegelt. Diese wurde auch in der Implementierung umgesetzt. Theoretisch wäre jede Komponente durch die Ersetzung der Interface Implementierungen austauschbar. Es stellt sich allerdings die Frage, ob dieser Mechanismus wirklich zum Tragen kommt. Es scheint vor allem unrealistisch, dass die Flexibilität des Postkorbsystems tatsächlich genutzt wird.

Historisierung

Das Prozesssystem speichert jede durchlaufene Station mit einem Protokolleintrag ab. Dabei werden sowohl Datum/Zeit und optional auch ein Kommentar gespeichert.

Logging

Sowohl das implementierte System als auch die jBpm selbst erzeugen eine Vielzahl von Logging Ausgaben. Diese Ausgaben spiegeln aber eindeutig die technischen Aspekte und Abläufe des Systems stärker als fachliche Belange. Für ein fachliches Logging müssen die Einzelaufgaben selbst sorgen.

Überwachung

Über das Postkorbsystem sind jederzeit alle Prozesse abruf- und kontrollierbar. Ein reines Monitoring wurde nicht implementiert.

6.6 Bewertung der Einsetzbarkeit

Die explizite Trennung der Schnittstellen und Einzelaufgaben vom Gesamtsystem hat sich gut bewährt. Vor allem die Komplexitätsminderung in der Entwicklung einzelner Schritte bringt Vorteile.

Eine solche Lösung mit einer expliziten modellierten Prozesslogik ist auf jedem Fall implizit implementierten Prozesssystemen vorzuziehen. Im Rahmen des Projektes konnte das System auch mit einer softwaretechnisch sehr ausgereiften Lösung verglichen werden. Dabei hatte die Vergleichslösung ebenfalls die Metaphern von Laufzettel und Vorgangsmappe verwendet, wobei der Laufzettel die Prozesssteuerung übernommen hat und in JAVA implementiert war. Bei diesem Vergleich hat sich gezeigt, dass beide Systeme die Anforderungen in etwa gleich gut abdecken konnten. Das Vergleichssystem hatte allerdings das Spring Framework nicht verwendet, wodurch sich eine sehr viel stärkere innere Kopplung ergeben hat. Im Rahmen der Integrationstest hatte sich wiederholt die Notwendigkeit ergeben, die Prozesse flexibel zu beeinflussen. Diese Anforderung hätte das hier vorgestellte System abdecken können.

Als problematisch ist die Wartung des Kernsystems einzustufen. Das Kernsystem mit der Abstraktion von der Workflow Engine und der Bereitstellung des Postkorbsystems ist sehr komplex. Die Wartung und Weiterentwicklung eines solchen Systems bedarf fundierter softwaretechnischer Kenntnisse. Durch die massiven Abhängigkeiten zur Workflow Engine kann auch ein Update der Engine meist nicht ohne Nachpflege des Systems erfolgen. Somit bringt der Einsatz eines solchen Systems das Risiko der entstehenden Wartungskosten mit sich. Sehr viel unproblematischer ist der Einsatz, wenn eine solche Technologie als gut geführtes Open Source Projekt zur Verfügung steht.

6.7 Persönliches Fazit

Im Rahmen der Erarbeitung dieser Arbeit haben sich für mich persönlich folgende Erkenntnisse ergeben, die ich in folge auch näher erläutern und argumentieren werde:

1. Der derzeit gängige Ansatz zur Implementierung von Geschäftsprozessen in IT Systemen vernachlässigt die Chancen situierter kooperativer Arbeit.
2. Die Vorstellung, Geschäftsprozesse so zu modellieren, dass diese ohne weiteren Codierungsaufwand ausgeführt werden können, ignoriert die Notwendigkeit zur Informationsabstraktion und –aggregation in Modellen.

Ad 1)

Der Einsatz von ablaufsteuernden Mechanismen in Prozesssystemen macht eine flexible Steuerung von Prozessen und Abläufen unmöglich. Durch Einsatz solcher Systeme können Ablauforganisationen langfristig ihre Fähigkeit zur Selbstregulung und zur Optimierung verlieren. Ein Unternehmen lebt von der eigenen Innovation und der stetigen Verbesserung von innen heraus. Dazu bedarf es aber eigenverantwortlich und selbstständig handelnder Mitarbeiter. Die Mitarbeiter können aber nur dann

eigenverantwortlich handeln, wenn ihnen auch die Verantwortung für ihr Handeln gegeben wird. Dem widerspricht der Ansatz, dass ein Softwaresystem ihr Handeln ablaufsteuert und durch ein Regelwerk überwacht. Nur ein Softwaresystem, das den Benutzern Routinetätigkeiten abnimmt und sie gleichzeitig in Ihrer Flexibilität und Kooperation unterstützt, wird langfristig auch die Eigenverantwortung der Mitarbeiter ermöglichen und fördern.

Ad 2)

Wie bereits viele Softwareprodukte zeigen, ist es natürlich möglich, Prozesse so zu modellieren, dass die Modelle direkt ausgeführt werden können. Dazu muss der Prozess in einem formalen Modell vorliegen. Ein solches Modell kann z.B. eine syntaktisch korrekte eEPK oder auch ein UML Model sein. Wie auch die MDA beschreibt, ist es in diesem Zusammenhang möglich, die Modelle von Implementierungsdetails über die Technologieplattform zu entkoppeln ([MDA Guide, 04] oder [Müller, 05]). Genau dieser Schritt wurde im Rahmen dieser Arbeit durch die Verwendung der Workflow Beschreibung in XML Notation und die Entkoppelung durch das Postkorbsystem ebenfalls beschritten.

Führen wir nun den Gedankengang von entgegen gesetzter Seite aus. Wir modellieren also den Geschäftsprozess in Form eines formalen und syntaktisch korrekten Modells und bringen dieses zur Ausführung. Damit der Prozess wie gewünscht ausgeführt wird, muss das Model die gesamte fachliche Komplexität des Prozesses aufweisen.

Um diese Komplexität modellieren zu können, kann die schrittweise Verfeinerung verwendet werden. In der Softwareentwicklung ist dieses Vorgehen eine übliche Technik, jedoch mit der Rahmenbedingung, das Problem genau zu kennen und sich schrittweise eine Lösung zu erarbeiten. Im Gegensatz dazu ist aber in der Prozessmodellierung das Problem (also der Prozess) im Vorfeld nicht klar.

Es bleibt aber weiterhin der Nachteil bestehen, dass der formale Charakter solcher Beschreibungssprachen Einschränkung in der Ausdrucksvielfalt und in der Verständlichkeit der Modelle mit sich bringt. Somit scheint mir der Einsatz einer leicht verständlichen und einfachen Modellierungsmethode wie der exemplarischen Geschäftsprozessmodellierung von Vorteil, da eine solche Methode den scheinbaren Formalismus und die scheinbar triviale Ausführbarkeit erst gar nicht suggeriert.

Es bleibt also festzuhalten, dass Prozessmodelle sehr wohl ausführbar sein können. Diese Modelle müssen aber in mehreren Abstraktionsstufen aufgebaut und über mehrere Iterationszyklen entwickelt worden sein. Weiters müssen sie die gleiche Komplexität wie auch die implementierende Software (mit Ausnahme der technologischen Aspekte) beinhalten. Es bleibt die Frage offen, ob ein graphisches Modell mit sehr hoher Komplexität eine bessere Darstellungsform ist, als der Quellcode selbst.

Wenn man einen Vergleich mit UML wagt, und die Darstellung eines Algorithmus zwischen UML und Quellcode vergleicht, so erkennt man, dass die graphische Darstellung weniger verständlich ist als der Quellcode selbst. Graphische Darstellungsformen bestehen durch ihre Stärke, Überblick zu schaffen, aber nicht damit, Details darzustellen.

7 Ausblick und weiterführende Ideen

Im Laufe der Diskussionen und Arbeiten zu dieser Arbeit, sind auch bereits einige weiterführende Gedanken aufgekommen. Da eine ausführliche Behandlung dieser Themen nicht möglich war, werden sie an dieser Stelle in aller Kürze beschrieben.

7.1 Laufzeitoptimierung

Die im Zuge der Arbeit implementierten Teile des Systems bieten an vielen Stellen noch Möglichkeiten zur Laufzeitoptimierung. Vor allem durch Verwendung von Caches, um das wiederholte Nachladen der Prozessdefinitionen zu vermeiden, scheint sinnvoll und zweckmäßig. Dieser Schritt war aber für die Erkenntnisse dieser Arbeit nicht notwendig.

7.2 Rollen- und Benutzermodell

In einem Szenario mit mehreren Klärungsstellen und Werkzeugen zur Benutzerinteraktion ist auch ein Benutzer- und Rollenkonzept zur Rechtsteuerung notwendig. Dieser Aspekt wurde im Rahmen der Arbeit überhaupt nicht behandelt.

7.3 Erkennung des Prozessfortlaufes

In der bisherigen Implementierung wird der zukünftige Verlauf des Prozesses durch Anwendung von einfachen Regeln aus der Prozessdefinition erstellt. Eine weitere Möglichkeit besteht darin, den Prozess anhand anderer, bereits abgelaufener, Prozesse zu berechnen. Damit wird der Weg von lernenden Prozesssystemen eingeschlagen. Dieser Gedanke kann noch weiter geführt werden, sodass die Prozessmuster von einer lernenden Prozesskomponente entstehen. Man würde also den Prozessablauf überhaupt nicht mehr vorgeben, sondern durch wiederholte Abwicklung des Prozesses über die Klärungsstelle, das System trainieren und bei ausreichender Genauigkeit automatisch ablaufen lassen.

7.4 Workflow Engine mit Möglichkeiten eines Prozessmusters

Durch die Verwendung der Workflow Engine zur Realisierung der Prozessmuster haben sich einige Probleme ergeben. Daher würde eine angepasste Engine, die auch das Konzept der Prozessmuster besser unterstützt, sicherlich eine Verbesserung bringen. Vorstellbar ist etwa ein Verfahren, das zwar auch auf den Ideen von Zustandsautomaten aufbaut, aber auch nicht definierte Zustandsübergänge erlaubt.

Literaturverzeichnis

- [BEA, 05] BEA Systems: <http://e-docs.bea.com/wli/docs81/index.html> (zuletzt besucht im April 2005)
- [Bleek et al, 99] Bleek, W.-G., Lilienthal, C., Züllighoven, H.: Frameworkbasierte Anwendungsentwicklung (Teil 4): Fachwerte. OBJEKTSpektrum 5/99, September/Oktober 99, S. 75-80 (1999)
- [BOC, 05] BOC Information Systems GmbH, <http://www.boc-eu.com/> (zuletzt besucht im Juni 2005)
- [Miller, 03] Randy Miller, Practical UML™: A Hands-On Introduction for Developers, Borland Developer Network (2003)
- [Borland, 03] Randy Miller, Practical UML™: A Hands-On Introduction for Developers, Borland Developer Network (2003)
- [Breitling, 03] Breitling H.: Integrierte Modellierung von Geschäftsprozessen und Anwendungssoftware, (2004)
- [Breitling, 05] Design Rationale in Exemplary Business Process Modeling, Holger Breitling, Andreas Kornstädt, and Joachim Sauer (erscheint voraussichtlich 2006)
- [Casewise, 05] The Corporate Modeler Suite, <http://www.casewise.com/products/corporate-modeler/index.php> (zuletzt besucht im Juni 2005)
- [Davenport, 92] Thomas H. Davenport, Process Innovation: Reengineering Work Through Information Technology, Harvard Business School Press (1992)
- [EMS, 05] The Equipment Management System (EMS), Petra Becker-Pechau et al, <http://www.it-wps.com/oocb/ems.php> (2005)
- [Floyd, 95] Floyd, C.: Theory and Practice of Software Development - Stages in a Debate. In: TAPSOFT '95: Theory and Practice of Software Development. Hrsg.: Mosses, P.D.; Nielsen, M.; Schwartzbach, M.I. Springer, Berlin u.a. 1995, S. 25–41 (= Lecture Notes in Computer Science 915) (1995)
- [Fowler, 04] Fowler, M.: Inversion of Control Containers and the Dependency Injection pattern, <http://martinfowler.com/articles/injection.html> (2004)
- [Galler, 97] Galler J.: Vom Geschäftsprozessmodell zum Workflow Modell. Gabler (1997)
- [Gamma et al, 94] Gamma E., Helm R., Johnson R., Vlissides J.: Design Patterns – Elements of Reusable Object-Oriented Software, Addison-Wesely (1994)
- [Gryczan, 95] Gryczan G.: Situierete Koordination computergestützter qualifizierter Tätigkeit über Prozeßmuster, Dissertationsschrift (1995)
- [Jablonski, 95] Jablonski, S.: Workflow-Management-Systeme: Motivation, Modellierung, Architektur. Informatik-Spektrum, 18 1, S. 13 – 24 (1995)
- [Jablonski et al. 97] Jablonski S., Böhm M., Schulze W.: Workflow-Management . Entwicklung von Anwendungen und Systemen, dpunkt (1997)

-
- [JBpmDoc, 05] Java Business Process Management Documentation, <http://www.jbpm.org/docs.html>, (zuletzt besucht Juni 2005)
- [Johnson, 04] Johnson R., Hoeller J.: Expert One-on-One J2EE Development without EJB, Wrox (2004)
- [JWAM, 05] C1-WPS JWAM Framework, <http://www.jwam.de>, (zuletzt besucht Juni 2005)
- [Krabbel, 98] Krabbel, A., Wetzel, I.: Analyse bereichsübergreifender Aufgaben im Rahmen der Auswahl eines Krankenhausinformationssystems. In E. Pinter, E. Swart, K.D. Vitt (Hrsg.): Umfassendes Qualitätsmanagement – Beispiele und Erfahrungsberichte für die Einführung im Krankenhaus, S.219-233. (1998)
- [Luginsland, 04] Luginsland R.: Prozesshandbuch für 800 Banken, Präsentation am WAM-Workshop 9./10.12.2004, <http://www.it-wps.de/events/wamworkshop/fohlen/09-Prozesshandbuch-Luginsland.pdf> (2004)
- [Manageability, 05] Perez C.: Open Source Workflow Engines Written in Java, http://www.manageability.org/blog/stuff/workflow_in_java/view, (April 2005)
- [Manolescu, 01] Manolescu D. A.: Micro-Workflow: A Workflow Architecture supporting compositional Object-oriented Software Development, Dissertationsschrift (2001)
- [MDA Guide, 04] Miller J., Mukerji J.: MDA Guide Version 1.0.1, Object Management Group (2004)
- [Muehlen, 04] Michael zur Muehlen, Workflow Based Process Controlling, Logos Verlag Berlin (2004)
- [Müller, 05] Müller Georg, Arbeitstitel: Generierung von Business Objekten aus Metadaten von Objektrelationalen Datenbanken (voraussichtlich 2005)
- [Österle, 03] Hubert Österle, Robert Winter, Business Engineering - Auf dem Weg zum Unternehmen des Informationszeitalters, 2. Auflage, Springer (2003)
- [Picot, 95] Arnold Picot, Peter Rohrbach: Organisatorische Aspekte von Workflow-Management-Systemen, Information Management 1/95 (1995)
- [Popkins, 05] Popkin Enterprise Architecture, http://www.popkin.com/solutions/enterprise_architecture.htm (zuletzt besucht im Juni 2005)
- [Quartz, 05] <http://www.opensymphony.com/quartz/> (zuletzt besucht im Mai 2005)
- [RCP, 05] The Eclipse Rich Client Platform, <http://eclipse.org/rcp/> (zuletzt besucht Juni 2005)
- [Scheer, 02] Scheer A.W.: ARIS - vom Geschäftsprozess zum Anwendungssystem, Springer (2002)
- [Schmelzer, 04] Schmelzer, Sesselmann: *Geschäftsprozessmanagement in der Praxis*. 4. Auflage, Hanser (2004)
- [Schulze, 96] Schulze W., Böhm M.: Klassifikation von Vorgangsverwaltungssystemen; in Geschäftsprozessmodellierung und Workflow-Management, Vossen G., Becker J. (Hrsg.), Kapitel 16, Thomson Publishing (1996)

-
- [Schwab, 96] Schwab K.: Koordinationsmodelle und Softwarearchitekturen als Basis für die Auswahl und Spezialisierung von Workflow-Management-Systemen; in Geschäftsprozessmodellierung und Workflow-Management, Vossen G., Becker J. (Hrsg.), Kapitel 17, Thomson Publishing (1996)
- [Sprenger, 02] Sprenger R. K.: Das Prinzip Selbstverantwortung, Campus Verlag (2002)
- [Szyperski, 02] Szyperski C, Gruntz D., Murer S.: Component software : beyond object-oriented programming, Addison-Wesely (2002)
- [Teufel et al. 95] Teufel S., Sauter C., Mühlherr T., Bauknecht K.: Computerunterstützung für die Gruppenarbeit, Addison-Wesley (1995)
- [UML, 04] Unified Modeling Language: Superstructure, Version 2.0 - Revised Final Adopted Specification, OMG (2004)
- [Vogler, 96] Vogler P., Chancen und Risiken von Workflow-Managment; in Praxis des Workflow-Managements, Vieweg & Sohn Verlagsgesellschaft mbH (1996)
- [WfMC, 05] Workflow Management Coalition, Homepage <http://www.wfmc.org>, (Juni 2005)
- [WfMC Glossary, 99] Workflow Management Coalition: Terminology & Glossary, Document Number WfMC-TC-1011 (1999)
- [WfMC Reference, 95] Hollingsworth, D.: The Workflow Reference Model, Document Number TC00-1003 (1995)
- [WfMC, 02] Workflow Management Coalition: Workflow Process Definition Interface - XML Process Definition Language, Document Number WfMC-TC-1025 (2002)
- [Wintersteiger, 02] Composite Business Software: Mittels Modellsynthese vom Geschäftsprozess zum Anwendungssystem, Dissertationsschrift (2002)
- [Zachmann, 87] A Framework for Information Systems Architecture." John A. Zachman. IBM Systems Journal, vol. 26, no. 3, (1987)
- [Züllighoven et al, 98] Züllighoven, H. et al., Das objektorientierte Konstruktionshandbuch nach dem Werkzeug & Material-Ansatz, d-punkt Verlag (1998)
- [Züllighoven et al, 04] Züllighoven, H., et. al, Object-Oriented Construction Handbook, dpunkt.verlag/Copublication with Morgan-Kaufmann (2004)