Diplomarbeit

Unterstützung kooperativer Arbeitsprozesse: Vorgangsmappen im JWAM-Framework

Holger Breitling
Talstraße 34
20359 Hamburg

August 2000

Erstbetreuung: Dr. Ingrid Wetzel
Zweitbetreuung: Prof. Dr. Horst Oberquelle

Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Straße 30
22527 Hamburg

Diplomarbeit

Unterstützung kooperativer Arbeitsprozesse: Vorgangsmappen im JWAM-Framework

Holger Breitling
Talstraße 34
20359 Hamburg

Erklärung:

Ich versichere hiermit, diese Arbeit selbständig und unter ausschließlicher Zuhilfename der in der Arbeit aufgeführten Hilfsmittel erstellt zu haben.

Hamburg, den 1. August 2000

Holger Breitling

Talstraße 34
20359 Hamburg

Betreuung:

Dr. Ingrid Wetzel (Erstbetreuerin)
Prof. Dr. Horst Oberquelle (Zweitbetreuer)

Dr. Ingrid Wetzel

Arbeitsbereich Softwaretechnik
Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Str. 30
22527 Hamburg

Prof. Dr. Horst Oberquelle

Arbeitsbereich Angewandte und Sozialorientierte Informatik
Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Str. 30
22527 Hamburg

Inhaltsverzeichnis

1 Einleitung

Diese Arbeit verknüpft zwei Themengebiete: Die Unterstützung kooperativer Arbeit durch Computer und die Konstruktion von Komponenten für ein objektorientiertes Framework.

Motivation

Der Arbeitsplatzcomputer ist im Büro schon seit Beginn der Achtziger Jahre verbreitet und wird zur Erledigung einzelner Aufgaben wie Textverarbeitung, Tabellenkalkulation und Vergleichbarem verwendet. Mit der wachsenden Vernetzung der Rechner entstand technisch die Möglichkeit, neben der Arbeit, die ein jeder einzeln zu erledigen hat, auch die Zusammenarbeit zwischen verschiedenen Menschen mit Hilfe des Computers effektiv zu unterstützen.

Diese technische Möglichkeit lieferte allerdings noch keine Vorstellung davon, wie eine solche Unterstützung aussehen soll. Die passenden Formen der Unterstützung mussten und müssen auch weiterhin gesucht werden. In der Informatik beschäftigt sich das Forschungsgebiet *CSCW (Computer-Supported Cooperative Work)* mit diesen Fragen; auch kommerzielle Anbieter haben in diesem Thema einen Markt erkannt und bieten eine ganze Reihe von Systemen mit verschiedenen Ansätzen an, die Unterstützung und/oder Organisation von kooperativer Arbeit bereitstellen sollen.

Die verschiedenen Systeme, die etwa als *Groupware* oder als *Workflow Management Systeme* eingeordnet werden, müssen dabei an die spezifische Anwendungssituation angepasst werden, was entsprechende Gestaltungsmöglichkeiten voraussetzt. Zudem besteht die Notwendigkeit, vielfältige andere im Büroumfeld verwendete Anwendungssysteme einzubinden und anzuschließen. Die nötigen Anpassungen und Anschlüsse können meist nur über dürftige Skriptsprachen oder proprietäre Programmiersprachen erfolgen, was wirklich große gestaltende Eingriffe aus softwaretechnischer Sicht sehr erschwert. Ein weiteres Problem ist, dass die Systeme häufig keine guten *Team Player* sind und beim Zusammenspiel mit anderen Anwendungen stets davon ausgehen, dass sie selbst jeweils die anderen Anwendungen ansteuern. Mithin ist ein Mangel an Gestaltungsmöglichkeiten und Integrationsfähigkeiten solcher Systeme festzustellen.

Objektorientierte Frameworks werden mit dem Ziel konstruiert, gute Entwürfe, Konzepte und Implementationen für eine flexible Wiederverwendung bereitzustellen. Frameworks liefern einen Rahmen für die Erstellung von Anwendungen und stellen passende Software-Bausteine bereit, die das angemessene Ausfüllen dieses Rahmens erleichtern. Die im Framework vorhandenen Konzepte und Implementationen werden dabei für die gewünschte Anwendung spezialisiert und konfiguriert. Da objektorientierte Frameworks auf modernen, objektorientierten (teilweise sogar standardisierten) Programmiersprachen wie Java, Smalltalk, Eiffel oder C++ aufsetzen, können diese bruchlos genutzt werden.

Die dieser Arbeit zugrundeliegende Idee ist es deshalb, Kooperationsunterstützung in einem Framework zur Verfügung zu stellen, genauer: als *Framework-Komponenten*[1] des Java-Frameworks JWAM. Diese Idee war auch die Ausgangsbasis des CoJAC-Projekts (CoJAC ist ein Akronym für "**Co**operative **J**WAM **A**rchitecture **C**omponents"), aus dem drei Arbeiten hervorgegangen sind. Dies ist eine von ihnen.

Mit der Bereitstellung von Kooperationsunterstützung in einem Framework soll die Entwicklung maßgeschneiderter Groupware erleichtert werden. Die Gestaltungsmöglichkeiten, die ein

[1] Der Begriff *Framework-Komponente* wird in Abschnitt 2.1.6.2 ab Seite 25 erläutert.

Entwickler bei der Verwendung eines Frameworks und einer Programmiersprache wie Java hat, sind viel größer als die Gestaltungsmöglichkeiten bei Anpassung einer "fertigen" Groupware. Gleichzeitig ist die Programmiersprache Java mit ihren zahlreichen standardisierten Schnittstellen zu externen Systemen[2] sehr gut geeignet, andere Anwendungen und Systeme anzusprechen und zu integrieren. Mit der Wiederverwendung bewährter Konzepte und Implementationen aus dem Framework verbindet sich die Hoffnung, dass der zusätzliche Programmieraufwand (im Vergleich zu Verwendung und Anpassung einer eingekauften Groupware) in einem vertretbaren Rahmen bleibt.

Das Framework JWAM basiert auf dem WAM-Ansatz, der am Arbeitsbereich Softwaretechnik des Fachbereichs Informatik der Universität Hamburg entwickelt wurde und der die Anwendungsorientierung in seinen Mittelpunkt stellt. Die Konzepte des WAM-Ansatzes zur Kooperationsunterstützung zielen auf den kompetenten Anwender, den eigenverantwortlichen Experten, und stellen eine gute konzeptionelle Basis für Groupware-Lösungen dar.

Entsprechend dem WAM-Ansatz enthält JWAM Konzepte und Implementationen für die Konstruktion von Softwarewerkzeugen und –materialien, einen Desktop und weitere Teile, von denen eine auf JWAM basierende Groupware-Anwendung (neben den CoJAC-Bestandteilen) noch Nutzen haben könnte

Ziele der Arbeit

Guido Gryczan hat in seiner Dissertation „Prozessmuster zur Unterstützung kooperativer Tätigkeit" ([Gry96]) ein Konzept zur Unterstützung von kooperativen Arbeitsprozessen vorgestellt, bei denen die Arbeitsbeiträge der einzelnen Mitarbeiter von den Ergebnissen der Arbeitsbeiträge anderer abhängen.

Das Konzept basiert auf der Idee, die „Blaupause" eines Arbeitsprozesses als veränderbaren (softwaretechnischen) Gegenstand gemeinsam mit einer (elektrifizierten) Mappe, in der sich die passenden Arbeitsgegenstände befinden, von Bearbeiter zu Bearbeiter zu schicken. Dabei wechselt mit dem Besitz des Prozessmusters auch die Verantwortlichkeit. Auf diese Weise steuern die Bearbeiter den Verlauf des Arbeitsprozesses selbst.

Andere im WAM-Ansatz bereits "angedachte" Themen sind die "Verwendung der Raummetapher zur Entwicklung kooperationsunterstützender Softwaresysteme für Organisationen" (so der Titel einer im WAM-Umfeld entstandenen Arbeit von Stefan Roock und Henning Wolf [RW98]) und die Gestaltung eines elektronischen Postversandsystems, das Arbeitsmaterialien von Arbeitsplatz zu Arbeitsplatz befördern kann. Diese Themen sind Gegenstand der beiden anderen CoJAC-Arbeiten von Mirko Freund und Jörn Koch.

Ich greife in dieser Arbeit das Konzept des Prozessmusters in Gestalt der von Züllighoven in [Zül98] vorgeschlagenen Vorgangsmappen und Laufzettel auf und untersuche es im Hinblick auf Realisierungsmöglichkeiten in JWAM. Die Herausforderung besteht hierbei zum einen darin, das Konzept in konkrete Werkzeuge und Materialien (die Grundbausteine von WAM-Systemen) umzusetzen. Andererseits fehlt dem Konzept noch ein Mechanismus, um im Umlauf befindliche Prozessmuster zu verfolgen oder über eine Suchfunktion aufzufinden, und diesen Mangel gilt es zu beheben.

Ziel dieser Arbeit ist es also, JWAM um eine Framework-Komponente zur Kooperationsunterstützung zu erweitern, die Vorgangsmappen und Laufzettel sowie einen Mechanismus zur Vorgangsverfolgung bereitstellt.

---

[2] Hier sind etwa zu nennen: *Java Native Interface* für Zugriff auf C-APIs, *JDBC* für den Anschluss relationaler Datenbanken, mehrere verfügbare *Object Request Broker* für den Anschluss von CORBA-Objekten u.v.m.

Der Entwurf der Komponente soll dabei, ausgehend vom Konzept des Prozessmusters, schrittweise entwickelt werden, wobei die einzelnen Entwurfsentscheidungen anhand der dargelegten Argumente nachvollziehbar sein sollen. Neben der Betrachtung wissenschaftlicher Arbeiten sollen auch Erkenntnisse anhand existierender Anwendungen aus den Bereichen *Computer-Supported Cooperative Work* bzw. *Workflow-Management-Systeme* bei der Argumentation beachtet werden.

Die Komponente soll so konzipiert und konstruiert werden, dass dem Entwickler und dem Benutzer für darauf aufbauende Anwendungen eine gemeinsame Sichtweise, ein gemeinsames Benutzungsmodell ermöglicht wird.

Die Komponente soll mit den Framework-Komponenten der anderen CoJAC-Arbeiten, Postversandsystem und Raumsystem, gemeinsam verwendet werden können und möglichst gut mit ihnen zusammenarbeiten.

Übersicht über die Arbeit

Der WAM-Ansatz und das JWAM-Framework sind die Gegenstände des zweiten Kapitels. Hier behandle ich mit dem zentralen Leitbild des Arbeitsplatzes für eigenverantwortliche Tätigkeit und den zugehörigen Entwurfsmetaphern und -mustern die Grundzüge des WAM-Ansatzes. Auch zum Thema JWAM stelle ich mit der Schichtenarchitektur des Frameworks und den wichtigsten Bestandteilen wie Werkzeug- und Materialkonstruktion oder Fachwerten Grundlegendes in den Vordergrund. Am Ende des Kapitels erläutere ich die Begriffe *Software-Komponente* und *Framework-Komponente* im JWAM-Kontext. Ziel des Kapitels ist es, ein Grundverständnis von WAM/JWAM zu vermitteln, das es erleichtert, den Rest der Arbeit flüssig zu lesen und gut zu verstehen.

Im anschließenden Kapitel mit dem Namen **Kooperationsunterstützung für JWAM** stelle ich die CoJAC-Arbeiten kurz vor und ordne sie anhand verschiedener Kriterien für kooperationsunterstützende Software ein. Den Hintergrund dieser Einordung bildet eine knappe Übersicht zu den Forschungsgebieten *Computer-Supported Cooperative Work* und *Workflow-Management-Systeme* sowie die Darstellung der Begriffe, die im WAM-Ansatz im Zusammenhang mit der Unterstützung kooperativer Arbeit verwendet werden.

Das Kapitel **Kooperationsunterstützung durch Prozessmuster** beginnt mit einer Zusammenfassung der theoretischen Fundierung der Prozessmuster, die Gryczan in [Gry96] formuliert hat. Im Anschluss begründe ich, warum Vorgangsmappe und Laufzettel als Umsetzung des Prozessmuster-Konzepts anzusehen sind verdeutliche ihre Verwendung anhand eines Beispiels aus dem Bankenbereich. Den Abschluss des Kapitels bildet eine stark an den physischen Vorbildern orientierte Modellierung der Verwendung von Vorgangsmappe und Laufzettel durch ein Use-Case-Diagramm. Die Gegenstände selbst werden mittels CRC-Karten modelliert. Dies schließt das Kapitel mit einer Grundlage für den weiteren Entwurf ab.

Dieser Entwurf wird im Verlauf des Kapitels **Entwurf und Implementation von Vorgangsmappe und Laufzettel** entwickelt. An dessen Beginn steht die vergleichende Untersuchung einiger den Vorgangsmappen ähnlicher Ansätze. Aus meiner Sicht besonders gelungene Aspekte dieser Ansätze nehme ich als Anregungen auf. Diese Anregungen und die Diskussion einiger Details mündet schließlich in eine Reihe von Entwurfsentscheidungen. Bevor ich das Kapitel mit einer relativ technischen Beschreibung der konstruierten Werkzeuge und Materialien abschließe, stelle ich die getroffenen Entwurfsentscheidungen in einem stimmigen Benutzungsmodell dar.

Das Kapitel **Verfolgung von Vorgängen** spricht ein Thema an, das in Gryczans Ausführungen zu Prozessmustern keine intensive Behandlung erfahren hat: Die Verfolgung der Vorgänge, deren

Vorgangsmappen im System “unterwegs” sind. Zu Beginn des Kapitels wird die Notwendigkeit einer solchen Verfolgung anhand beispielhafter Informationswünsche belegt. Im Anschluss untersuche ich die Vorgangsverfolgungsmechanismen bei ähnlichen Ansätzen wie dem der Vorgangsmappen sowie bei Workflow-Management-Systemen.

Schlussfolgerungen aus diesen Betrachtungen ziehe ich im Kapitel **Entwurf und Implementation eines Vorgangsmonitors**, indem ich anhand der im vorigen Kapitel dargestellten Informationsbedürfnisse aufzeige, wie und wann diese Informationen gesammelt und bereitgestellt werden können. Auf Basis dieser Überlegungen entwickliche ich den Entwurf des Vorgangsmonitors und erläutere einige Implementationsdetails.

Mit dem letzten Kapitel, **Zusammenfassung und Ausblick,** möchte auf die vorigen Kapitel bilanzierend zurückblicken und die offenen Fragen formulieren.

Konventionen

Die formalen Abbildungen und Diagramme in dieser Arbeit entsprechen entweder dem durch die UML definierten Standard (vgl. [BRJ99]), oder aber sie sind im Text ausführlich erläutert oder ohnehin intuitiv verständlich.

Die Sprache, in der Klassennamen und Kommentare des JWAM-Frameworks verfasst werden, ist Englisch. Ich habe die Namen konstruierter Klassen nicht wieder künstlich eingedeutscht, was zum Teil zu Abbildungen führt, in denen Deutsch und Englisch vermischt sind. Ich bitte in diesem Punkt um Nachsicht.

Danksagung

An dieser Stelle möchte ich all den Menschen danken, die mir beim Erstellen dieser Arbeit eine Hilfe waren. Dr. Ingrid Wetzel danke ich für die Erstbetreuung der Arbeit und die überabzählbar vielen Hinweise, Anregungen und Diskussionen zu allen ihren Teilbereichen. Bei Prof. Dr. Horst Oberquelle bedanke ich mich für die Zweitbetreuung.

Großen Dank schulde ich auch der CoJAC-Gruppe: meinen Mitstreitern Mirko Freund und Jörn Koch, sowie Henning Wolf, der uns ein gutes Stück des Weges begleitet hat. Ihnen danke ich für viele intensive Diskussionen zu Entwurf und Implementation. Carsten Doehring und Mareile Wilkens danke ich für wertvolle Hinweise zu Workflow-Management-Systemen beziehungsweise zu kooperativen Arbeitsprozessen.

Auf meiner Dankesliste nicht vergessen möchte ich Dr. Guido Gryczan, dessen Dissertation eine der Wurzeln dieser Arbeit darstellt und der mich darüber hinaus mit Einzelheiten des Systems der RWG versorgt hat, sowie Prof. Dr. Heinz Züllighoven, der dem CoJAC-Team mit Rat zur Seite gestanden hat.

Schließlich gilt mein besonderer Dank der JWAM-Architekturgruppe, deren Mitglieder stets bereit waren, mit mir über Konstruktionsfragen zu diskutieren und ihr Wissen mit mir zu teilen, und Katharina Adam, die mit großem Einsatz Korrektur gelesen hat.

2 Der WAM-Ansatz und das JWAM-Framework

Der Entwurf und die Implementation einer Framework-Komponente, die das auf dem WAM-Ansatz basierende Java-Framework JWAM erweitert, sind Gegenstand dieser Arbeit. Deshalb setzt sie über weite Strecken ein Verständnis sowohl der Grundzüge des WAM-Ansatzes wie des JWAM-Frameworks voraus.

Um dieses Verständnis zu vermitteln, enthält dieses Kapitel deshalb eine komprimierte, zielorientierte Darstellung der beiden Themen.

Im letzten Abschnitt des Kapitels werden die Begriffe Software-Komponente und Framework-Komponente von mir erklärt und in Beziehung gesetzt, um so zu verdeutlichen, was mit der für diese Arbeit formulierten Zielstellung der "Konstruktion einer Framework-Komponente" gemeint ist.

Der WAM-Ansatz

Der WAM-Ansatz ist ein Ansatz für die objektorientierte Softwareentwicklung, der am Arbeitsbereich Softwaretechnik des Fachbereiches Informatik der Universität Hamburg vertreten, gelehrt und weiterentwickelt wird. Die Anfänge des Ansatzes reichen bis in die späten Achtziger Jahre zurück, grundlegende Arbeiten zum WAM-Ansatz sind etwa [BZ90] und [BBS+95]. Eine umfangreiche Darstellung des WAM-Ansatzes, die Arbeiten und Ergebnisse aus den vergangenen Jahren stimmig bündelt, ist "Das objektorientierte Konstruktionshandbuch nach dem Werkzeug & Material-Ansatz" ([Zül98]). Die folgende Kurzdarstellung des WAM-Ansatzes stützt sich auf dieses Werk ab und enthält zahlreiche Zitate.

Der WAM-Ansatz ist

"

- eine Sicht der objektorientierten Anwendungsentwicklung,
- eine Sammlung bewährter Konstruktions-, Analyse- und Dokumentationstechniken,
- eine Beschreibung zusammenpassender Konzepte
- eine Auswertung unterschiedlicher und umfangreicher Projekterfahrungen
- eine Anleitung, um eine konkrete Konstruktionstechnik und eine dazu passende Vorgehensweise zu entwickeln " ([Zül98], S. 10)

Der WAM-Ansatz ist als "Meta-Methode" zu verstehen, er soll die Erarbeitung einer maßgeschneiderten Softwareentwicklungs-Methode für ein konkretes Projekt und eine konkrete Entwicklungsorganisation unterstützen.

Der Name des Ansatzes steht dabei für Werkzeug-Automat-Material (vormals für Werkzeug-Aspekt-Material). Diese Begriffe werden im Verlauf dieses Abschnitts noch erläutert.

Als Bausteine für den Softwareentwicklung beschreibt der WAM-Ansatz eine Reihe von Elementen: Leitbilder, Entwurfsmetaphern, Konzeptions- und Entwurfsmuster sowie Dokumenttypen für den Entwicklungsprozess.

Die Aussagen des WAM-Ansatzes zum Themenkomplex "Entwicklungsprozess/Vorgehen in Softwareprojekten" werden im Rahmen dieser Arbeit jedoch nicht im Einzelnen dargestellt.

2.1.1 Anwendungsorientierung

Der WAM-Ansatz stellt die *Anwendungsorientierung* in der Softwareentwicklung in den Mittelpunkt: Die Funktionalität von *anwendungsorientierter Software* richtet sich an den Aufgaben aus dem Anwendungsbereich aus, wobei die festgelegten Schritte und Abläufe sich je nach Anwendungssituation problemlos an die tatsächlichen Erfordernisse anpassen lassen und die Handhabung benutzergerecht gestaltet ist.

Ein bewährtes Prinzip für die Entwicklung anwendungsorientierter Software besteht aus Sicht des WAM-Ansatzes darin, Gegenstände und Konzepte des Anwendungsbereiches als Grundlage des softwaretechnischen Modells zu nehmen. Das Ziel ist die Herstellung einer Korrespondenz zwischen den anwendungsfachlichen Begriffen und Gegenständen und der Softwarearchitektur.

Diese *Strukturähnlichkeit* erleichtert auf der einen Seite den Anwendern den Umgang mit einem Softwaresystem, denn sie finden die ihnen bekannten Begriffe und Gegenstände (in anderer Form, aber in den gewohnten Zusammenhängen) wieder. Auf der anderen Seite setzen die Software-Entwickler die Softwarekonstrukte und Anwendungskonzepte besser in Beziehung und können so beispielsweise bei fachlichen oder softwaretechnsichen Änderungen die welchselseitigen Abhängigkeiten schneller und deutlicher erkennen.

Die möglichst bruchlose Entsprechung dieser verschiedenen Ebenen soll auch Benutzern und Entwicklern ein gemeinsames Grundverständnis der Software ermöglichen, ein gemeinsam geteiltes Benutzungsmodell:

Definition 1: Benutzungsmodell[3]
Ein Benutzungsmodell ist ein fachlich orientiertes Modell darüber, wie Anwendungssoftware bei der Erledigung der anstehenden Aufgaben im jeweiligen Einsatzkontext benutzt werden kann. Das Benutzungsmodell umfasst eine Vorstellung von der Handhabung und Präsentation der Software, aber auch von den fachlichen Gegenständen, Konzepten und Abläufen, die von der Software unterstützt werden.
Es ist sinnvoll, ein Benutzungsmodell auf der Grundlage eines Leitbilds mit Entwurfsmetaphern zu realisieren

Das Benutzungsmodell ist die für den Anwender maßgebliche Abstraktion der Software; auf seiner Basis plant der Anwender seine Handlungen und antizipiert die Reaktionen der Software. Wenn es dem Anwender gelingt, sich ein der Software angemessenes Modell zu machen, erlaubt ihm dies eine zuverlässige und zielführende Bedienung der Anwendung. Eine falsche Vorstellung von der Funktionsweise einer Software führt hingegen zu ungewollten Resultaten und Verunsicherung.

Es ist die Aufgabe des Softwareentwicklers, bei Entwurf und Konstruktion ein geeignetes Benutzungsmodell “mitzudenken”, das sich die Anwender möglichst einfach zu eigen machen können. Bei dieser schwierigen Aufgabe sollen dem Entwickler Leitbilder und Entwurfsmetaphern helfen.

[3] aus [Zül98], Seite 71

2.1.2 Leitbilder

Der WAM-Ansatz schlägt die Verwendung von Leitbildern und Entwurfsmetaphern im Softwareentwicklungsprozess vor und stellt bestimmte konkrete Leitbilder und Entwurfsmetaphern und ihre Verwendung in den Mittelpunkt seiner Betrachtung. Ich möchte zunächst den Begriff des Leitbilds erläutern.

Ein Leitbild ist die Formulierung einer grundsätzlichen Sichtweise, die eine klare Orientierung bietet. Es repräsentiert immer auch eine Wertvorstellung, bezieht Position. Die folgende Definition konkretisiert den Begriff für die Softwareentwicklung.

Definition 2: Leitbild in der Softwareentwicklung[4]
Ein Leitbild in der Softwareentwicklung gibt im Entwicklungsprozess und für den Einsatz einen gemeinsamen Orientierungsrahmen für die beteiligten Gruppen.
Es unterstützt den Entwurf, die Verwendung und die Bewertung von Software und basiert auf Wertvorstellungen und Zielsetzungen.
Ein Leitbild kann konstruktiv oder analytisch verwendet werden.

Der zuletzt genannte Aspekt betont, dass Leitbilder einerseits Orientierung bei der Konstruktion von Softwaresystemen bieten, andererseits aber auch dazu dienen können, bestehende Anwendungssoftware vor diesem Hintergrund zu bewerten, ihr eine Ausrichtung an einem bestimmten Leitbild zu bescheinigen oder zu bestreiten.

Die folgende Tabelle vergleicht verschiedene existierende Leitbilder:

Leitbild	Gestaltungsziel	Rolle der Anwender	Rolle der Entwickler
Objektwelten	Objektorientierung unmittelbar auf den Entwurf übertragen	Impulsgeber, der die Objekte aktiviert	Schöpfer von Miniwelten
Direkte Manipulation von Arbeitsgegenständen	bekannte Arbeitsgegenstände selbstverständlich manipulieren	Akteur, Bearbeiter	Konstrukteur von Artefakten
Fabrik	menschliche Arbeit automatisieren und kontrollieren	Maschinenbediener, Störfaktor	Maschinenbauer, Maschineneinrichter
Arbeitsplatz für eigenverantwortliche Expertentätigkeit	qualifizierte Arbeit durch geeigneten Arbeitsplatz unterstützen	eigenverantwortlicher Experte, der Fachsprache spricht	Werkzeugbauer, Arbeitsplatzgestalter

Abbildung 1: Leitbilder in der Softwareentwicklung (aus [Zül98], Seite 74)

Die in der Tabelle aufgeführten Leitbilder sind teils explizit formuliert worden, teils sind sie implizit in Softwaresystemen vorzufinden. Die Übersicht betont die Gestaltungsziele der Leitbilder und die Rollenverteilung zwischen Anwender und Entwickler, die diese vorsehen.

Eine kurze Zusammenfassung der in der Tabelle aufgeführten Leitbilder:

[4] aus [Zül98], Seite 73

Leitbild Objektwelten

Hier wird die Objektorientierung selbst zum Leitbild. Die Objekte selber sind “Akteure” oder “Agenten”, die in einer virtuellen Welt miteinander kommunizieren. Sie sind künstliche, intelligente “Artefakte”, die die Benutzer zu Randfiguren, zu “Impulsgebern” machen, während die Softwareentwickler als Schöpfer von virtuellen Welten in Erscheinung treten.

Leitbild Direkte Manipulation von Arbeitsgegenständen

Das Leitbild der direkten Manipulation von Arbeitsgegenständen ist mit der Entwicklung grafischer Benutzeroberflächen und interaktiver Systeme entstanden. Es hat als Ziel, dass die Benutzer die einzelnen Arbeitsgegenstände, die häufig als direktes grafisches Abbild physischer Gegenstände dargestellt werden, scheinbar wie mit den eigenen Händern (angezeigt durch ein grafisches Symbol und bewegt durch eine Computer-Maus) bearbeiten und zueinander in Beziehung setzen können. “Drag and Drop”, eine Form der Interaktion, bei der der Benutzer Gegenstände mit Hilfe des Mauszeigers “aufheben” und auf andere Gegenstände “fallen lassen” kann, um bestimmte Resultate zu erreichen, folgt diesem Leitbild.[5] Die Anwender treten nach diesem Leitbild als Akteure mit einer relativ großen Handlungsfreiheit auf, während die Entwickler für die Konstruktion der verschiedenen direkt manipulierbaren Arbeitsgegenstände zuständig sind.

Leitbild Fabrik

Das Leitbild der Fabrik verfolgt den Ansatz, die Tätigkeiten des Anwenders in einem durch das Softwaresystem gesteuerten (starren) Arbeitsprozess zu organisieren. Dabei geht die Steuerung von der Maschine aus, sie gibt den Takt vor, fordert den Anwender zu Eingaben auf. Die Rolle des Anwenders ist die eines Maschinenbedieners, der sogar gelegentlich nur als Störfaktor betrachtet wird, weil er nicht automatisiert ist und Fehler macht. Die Softwareentwickler sind dagegen die Maschinenkonstrukteure, die Planer der Fabrik.

Leitbild Arbeitsplatz für eigenverantwortliche Expertentätigkeit

Das Leitbild vom Arbeitsplatz für eigenverantwortliche Expertentätigkeit ist *das* zentrale Leitbild des WAM-Ansatzes. Es zielt darauf ab, einen für qualifizierte Arbeit geeigneten Arbeitsplatz bereitzustellen. Der Anwender wird als fachlicher Experte betrachtet, der in seiner Tätigkeit nicht angeleitet oder gesteuert, sondern optimal unterstützt werden muss. Die daraus folgende Handlungs- und Entscheidungsfreiheit des Anwenders fordert vom Entwickler die Konstruktion flexibel verwendbarer Arbeitsgegenstände und Softwarewerkzeuge.

Die Leitbilder “Fabrik” und “Arbeitsplatz für eigenverantwortliche Expertentätigkeit” stehen einander antithetisch gegenüber. Die damit verbundenen verschiedenen Sichtweisen, die *ablaufsteuernde* Sichtweise und die *unterstützende* Sichtweise, werden in Abschnitt 3.1.2 (“Workflow-Management-Systeme”, ab Seite 31) und Abschnitt 0 (“Prozessmuster”, ab Seite 41) formuliert. Sie schlagen sich in den dort dargestellten, gegensätzlichen Konzepten für die Organisation von kooperativen Arbeitsprozessen nieder.

[5] Eine ausführlichere Betrachtung von direkter Manipulation (und speziell Drag and Drop) aus Sicht des WAM-Ansatzes findet sich in [Lip99] ab Seite 33.

2.1.3 Entwurfsmetaphern

Leitbilder sollen dem Softwareentwickler dabei helfen, das fachliche Modell aus einem bestimmten Anwendungskontext in den Entwurf einer zukünftigen Software zu überführen. Diese Funktion kann wesentlich durch die Verwendung von Entwurfsmetaphern unterstützt werden, die ein Leitbild ausdeuten und ergänzen:

Definition 3: Entwurfsmetapher[6]
Eine Entwurfsmetapher ist eine bildhafte, gegenständliche Vorstellung, die ein Leitbild fachlich und konstruktiv "ausgestaltet", d.h. konkretisiert.
Eine Entwurfsmetapher strukturiert die Wahrnehmung und trägt zur Begriffsbildung bei. Sie leitet die Vorstellung und Kommunikation über das, was fachlich analysiert, modelliert und technisch realisiert werden soll.
Eine Entwurfsmetapher dient der Gestaltung von Softwaresystemen, indem sie Handhabung und Funktionalität für die Beteiligten verständlicher macht.
Eine Entwurfsmetapher hat im WAM-Ansatz immer auch eine technisch konstruktive Interpretation in Form von Konstruktionsanleitungen und Entwurfsmustern.

Eine Entwurfsmetapher verbindet im WAM-Ansatz demnach eine bildhafte Vorstellung mit Konstruktionsanleitungen und Entwurfsmustern[7]. Damit schlägt die Entwurfsmetapher eine entscheidende Brücke zwischen fachlicher Sicht und konstruktiver Umsetzung.

Der WAM-Ansatz gestaltet das für ihn maßgebliche Leitbild vom Arbeitsplatz für eigenverantwortliche Expertentätigkeit mit mehreren Entwurfsmetaphern aus, die in diesem Abschnitt kurz beschrieben werden.

Von den für den WAM-Ansatz namensgebenden Entwurfsmetaphern gehe ich als erstes auf die Werkzeugmetapher ein:

Definition 4: Werkzeug (Entwurfsmetapher)[8]
Werkzeuge sind Gegenstände, mit denen Menschen im Rahmen einer Aufgabe Materialien verändern oder sondieren können.
Werkzeuge eignen sich meist für verschiedene fachliche Zwecke und für die Arbeit an unterschiedlichen Materialien. Sie müssen geeignet gehandhabt werden.
Werkzeuge vergegenständlichen wiederkehrende Arbeitshandlungen.
Viele konzeptionelle Eigenschaften von (Hand-)Werkzeugen lassen sich auf Softwarewerkzeuge übertragen. Eine direkte Abbildung der Handhabung und Gestalt ist aber selten sinnvoll.

Es zeigt sich am Beispiel der Werkzeugmetapher, dass bei der Verwendung einer Entwurfsmetapher sehr genau festgelegt werden muss, welche Merkmale tatsächlich übertragen und angewendet werden sollen. Bei der Werkzeugmetapher etwa steht im Hinblick auf Softwarewerkzeuge im Vordergrund, dass sie Materialien sondieren oder bearbeiten können. Softwarewerkzeuge sollten auch mit ähnlichen Freiheitsgraden verwendbar sein wie physische Werkzeuge: Sie müssen ergriffen werden können, der Anwender muss sie aber auch beiseite legen können.

---

[6] aus [Zül98], Seite 79

[7] Der Begriff Entwurfsmuster ist hier so zu verstehen, wie er in dem Standardwerk [GHJ+] beschrieben wurde.

[8] aus [Zül98], Seite 84

Nicht direkt übertragen werden können aber in der Regel Handhabung und Gestalt eines physischen Werkzeugs. So macht es keinen Sinn, einen Bleistift in Handhabung und Gestalt nachzubilden. Es geht vielmehr darum, die Funktionalität physischer Werkzeuge in Softwarewerkzeugen abzubilden und auch zusammenfassen (bei Bleistift und Radiergummi: schreiben, unterstreichen, durchstreichen, ausradieren usw.)

Aufgrund des eher passiven Charakters von Materialien (sie "werden bearbeitet") erlaubt die Entwurfsmetapher "Material" eine direktere Abbildung:

Definition 5: Material (Entwurfsmetapher)[9]
Materialien sind Gegenstände, die im Rahmen einer Aufgabe Teil des Arbeitsergebnisses werden. Materialien werden durch Werkzeuge und Automaten bearbeitet und verkörpern fachliche Konzepte. Sie müssen für die Bearbeitung geeignet sein.
Die Eigenschaften vorhandener Arbeitsgegenstände lassen sich oft sinnvoll auf Softwarematerialien übertragen.

Bei nach dem WAM-Ansatz durchgeführten Projekten erfahren die am Arbeitsplatz vorhandenen Materialien tatsächlich häufig eine Umsetzung in Software-Materialien.

Die noch fehlende Entwurfsmetapher aus dem Akronym "WAM" ist der Automat:

Definition 6: Automat (Entwurfsmetapher)[10]
Automaten sind im Rahmen einer zu erledigenden Aufgabe ein Arbeitsmittel, um Material zu bearbeiten. Sie erledigen lästige Routinetätigkeiten als eine definierte Folge von Arbeitsschritten mit festem Ergebnis ohne weitere äußere Eingriffe.
Automaten laufen unauffällig im Hintergrund, wenn sie einmal vom Benutzer oder von der Arbeitsumgebung gestartet sind.
Sie können auf ihren Zustand überprüft und im vorgegebenen Rahmen eingestellt werden.

Automaten arbeiten wie Werkzeuge auf Materialien, werden aber nicht im eigentlichen Sinne von Benutzern gehandhabt. Sie funktionieren sozusagen nur "auf Knopfdruck" und führen dann eine Routinetätigkeit mit genau bekanntem Ergebnis durch. Sie können eingestellt, konfiguriert werden.

Auch die Entwurfsmetapher "Automat" fügt sich gut zum Leitbild vom Arbeitplatz für eigenverantwortliche Expertentätigkeit, denn die Kontrolle über den Einsatzes des Automaten liegt beim Anwender.

Ich möchte hier noch eine letzte, wichtige WAM-Entwurfsmetapher aufführen, die für die Kooperationsunterstützung eine besondere Rolle spielt, die Arbeitsumgebung:

[9] aus [Zül98], Seite 86

[10] aus [Zül98], Seite 89

Definition 7: Arbeitsumgebung (Entwurfsmetapher)[11]
Die Arbeitsumgebung ist der Ort, wo Werkzeuge, Materialien und andere Gegenstände, die bei der Erledigung von Aufgaben griffbereit sein müssen, fachlich motiviert angeordnet sind. Dabei findet die eigentliche Arbeit am Arbeitsplatz statt, während zur Umgbung noch die Orte gehören, die unmittelbar zugänglich sind.
Der (individuelle) Arbeitsplatz ist gegen den Zugriff von außen geschützt. Wenn nur die Arbeit eines einzelnen Benutzer unterstützt werden soll, fallen Arbeitsplatz und –umgebung meist zusammen.

Die Arbeitsumgebung integriert die durch die anderen bisher genannten Entwurfsmetaphern beschriebenen Gegenstände, bietet ihnen einen Platz. Mit der Betrachtung der Unterstützung kooperativer Arbeit gewinnt die Unterscheidung zwischen Arbeitsplatz und Arbeitsumgebung an Wichtigkeit; dann nämlich wird einerseits der Transport von Gegenständen aus einem exklusiven Bereich (individuellen Arbeitsplatz) in einen anderen diskutiert und andererseits das "Delta" zwischen Arbeitsplatz und Arbeitsumgebung sichtbar: in Gestalt gemeinsam zugänglicher Orte (siehe hierzu Kapitel 3 ab Seite 26).

2.1.4 Wichtige WAM-Entwurfsmuster

Die Definition des Begriffes Entwurfsmetapher enthält die Aussage, dass jede Entwurfsmetapher nach dem WAM-Ansatz auch eine technisch konstruktive Interpretation in Form von Konstruktionsanleitungen und Entwurfsmustern hat. Dementsprechend gibt es insbesondere zu den Entwurfsmetaphern Material und Werkzeug einige zentrale Entwurfsmuster des WAM-Ansatzes, über die ich hier einen kurzen Überblick geben möchte. Eine Zusammenfassung derjenigen WAM-Entwurfsmuster, die sich auf Werkzeuge beziehen, findet man außer in [Zül98] auch in [Rie97].

2.1.4.1 Werkzeug- und Materialkopplung

Wenn Software-Werkzeuge und –Materialien konstruiert werden, stellt sich die Frage, auf welche Weise das "Zueinander-Passen" ausgedrückt werden kann, dass sich bei physischen Werkzeugen und Materialien häufig unmittelbar zeigt (mit einem bestimmten Schraubendreher kann man nur Schrauben bestimmter Größen drehen, ein Dosenöffner eignet sich nicht für Weinflaschen usw.)

Wie also werden Werkzeuge und Materialien gekoppelt? Der WAM-Ansatz beschreibt als Lösung für dieses Problem die Verwendung von *Aspekten.*

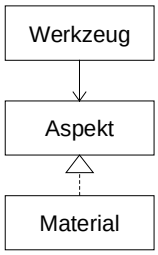


Abbildung 2: Kopplung von Werkzeug und Material über einen Aspekt

[11] aus [Zül98], Seite 87

Ein Aspekt repräsentiert den Verwendungszusammenhang zwischen einem Werkzeug und den durch das Werkzeug bearbeitbaren Materialien. Der WAM-Ansatz beschreibt die Möglichkeiten, dieses für verschiedene objektorientierte Programmiersprachen konstruktiv umzusetzen. Für Java empfiehlt es sich, Aspekte als Interfaces umzusetzen.

2.1.4.2 Trennung von Funktion und Interaktion

Die Trennung von Funktion und Interaktion ist ein Architekturkonzept für interaktive Anwendungen, die bereits im Model-View-Controller-Paradigma des Smalltalk-Systems zu finden ist. Bei der Anwendung auf Softwarewerkzeuge nach WAM wird unter der Funktion eines Werkzeuges seine fachliche Funktionalität und unter der Interaktion seine Präsentation und Handhabung verstanden.

Der WAM-Ansatz vertritt die Auffassung, dass Werkzeuge in eine Interaktions- und eine Funktionskomponente zerlegt werden sollten, mit dem Ziel, insbesondere die Interaktionskomponente (also Präsentation und Handhabung des Werkzeuges) unabhängig von der Funktionskomponente abändern zu können.

Ein ebenso wichtiger, struktureller Vorteil bei der Trennung von Funktion und Interaktion ist die Möglichkeit, das Wissen um Abhängigkeiten zwischen an der Benutzeroberfläche angezeigten und möglicherweise änderbaren Merkmalen eines Materials nur in der Funktionskomponente anzusiedeln.

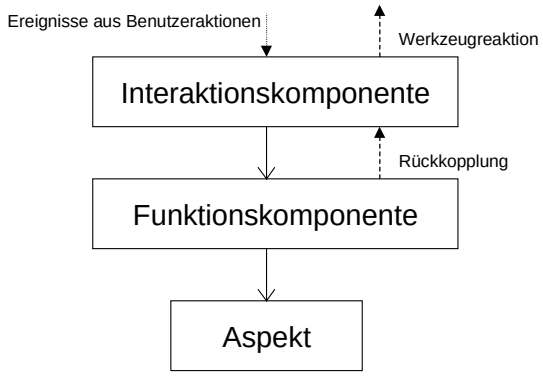


Abbildung 3: Rückkopplung zwischen Funktions- und Interaktionskomponente

Um dies konstruktiv umzusetzen, kann das Beobachtermuster[12] verwendet werden (die Interaktionskomponente tritt dabei der Funktionskomponente gegenüber als Beobachter auf). Die Interaktionskomponente reagiert auf Benutzereingaben und ruft die entsprechenden Operationen an der Funktionskomponente auf, um über diese ein Material zu modifizieren. Treten dadurch Änderungen an relevanten Merkmalen auf, informiert die Funktionskomponente ihre(n) Beobachter. Die Interaktionskomponente kann nun die Änderungen sondieren und ihre Anzeige anpassen.

---

[12] Das Beobachtermuster ist in [GHJ+] dargestellt und wird dort so beschrieben:
"Definiere eine 1-zu-n-Abhängigkeit zwischen Objekten, so dass die Änderung des Zustands eines Objekts dazu führt, dass alle abhängigen Objekte benachrichtigt und automatisch aktualisiert werden."

Das Beobachtermuster wird übrigens auch in der Model-View-Controller-Archtitektur des Smalltalk-Systems verwendet, das im Zusammenhang mit der Trennung von Funktion und Interaktion bereits genannt wurde.

2.1.4.3 Werkzeugkomposition

Der WAM-Ansatz beschreibt Entwurfsmuster, um Softwarewerkzeuge aus bereits existierenden oder auch aus neu zu konstruierenden Werkzeugen zusammenzusetzen. Die für diese Arbeit relevanten Aspekte der Werkzeugkomposition sind in Abschnitt 5.1.11.4 (“Exkurs: Werkzeugkomposition in JWAM”, ab Seite 85) dargestellt.

2.1.4.4 Fachwerte

Die meisten objektorientierten Programmiersprachen stellen sogenannte “Basisdatentypen” bereit, die Wertsemantik besitzen (z.B. Ganz- und Gleitkommazahltypen, Zeichen oder Zeichenketten), also

- (konzeptionell) zeit- und ortlos existieren
- auf Gleichheit verglichen werden können, aber keine Objektidentität besitzen
- unveränderlich sind

Bei der Modellierung großer Anwendungssysteme können sogenannte *Fachwerte* (dies sind domänenspezifische Werttypen wie Kontonummer, Geldbetrag usw.) identifiziert werden, die keine Entsprechung in Basisdatentypen objektorientierter Programmiersprachen haben.

Wenn diese Werttypen in der softwaretechnischen Umsetzung tatsächlich typisiert und nicht nur irgendwie in eine Zeichenkette oder einen anderen Basisdatentyp kodiert werden sollen, müssen sie in objektorientierten Programmiersprachen als Klassen modelliert werden. Der WAM-Ansatz macht hierzu einige Konstruktionsvorschläge.

Ein Java-Framework nach dem WAM-Ansatz: JWAM

Dieser Abschnitt beginnt mit einer kurzen Einleitung zu objektorientierten Frameworks und gibt dann einen Überblick über das JWAM-Framework.

2.1.5 Objektorientierte Frameworks

Eines der Versprechen, mit denen die Objektorientierung angetreten ist, ist eine größere Wiederverwendbarkeit der mit dieser Technik konstruierten Software. Bereits mit den ersten objektorientierten Programmiersprachen kamen die Klassenbibliotheken auf, Sammlungen von Klassen, die für die Wiederverwendung vorgesehen waren. Angesichts immer größerer und komplexerer Anwendungen ist allerdings eine Wiederverwendbarkeit, die sich ausschließlich auf diese Ebene beschränkt, nicht befriedigend.

Objektorientierte Frameworks werden mit dem Ziel konstruiert, Wiederverwendung auf einer höheren, makroskopischeren Ebene zu ermöglichen: Ein objektorientiertes Framework dient im Gegensatz zu einer Klassenbibliothek dazu, nicht einzelne Klassen, sondern ganze softwaretechnische Entwürfe, die jeweils ein Geflecht von Objekten und deren Zusammenspiel festlegen, als wiederverwendbare Einheiten bereitzustellen.

Mehrere solche Entwürfe, die Lösungen für ähnliche Probleme (für eine Klasse von Problemen) darstellen, werden in einem Framework zusammengefasst. Gebräuchlich sind beispielsweise Frameworks für eher technische Gebiete wie grafische Benutzerschnittstellen oder Persistenz, aber auch für bestimmte Domänen, etwa den Bank- oder Logistikbereich.

Dabei wird zwischen sogenannten *Black-Box-Frameworks* und *White-Box-Frameworks* unterschieden (vgl. [JF88]). Ein Black-Box-Framework ist dadurch gekennzeichnet, dass es nur

auf Basis seiner offengelegten Schnittstellen verwendet wird; der Entwickler hat hier keine Kenntnis von der Implementation oder soll sie mindestens nicht nutzen. Bei einem White-Box-Framework dagegen liegt die Implementation offen und der Entwickler kann und soll durch Vererbung davon Gebrauch machen. Größere Frameworks bestehen aus zahlreichen Teil- oder Sub-Frameworks und sind meist als Mischformen aus Black- und White-Box-Framework anzusehen.

In welcher Weise werden die in einem Framework abstrahierten Entwürfe durch den Entwickler für eine konkrete Anwendung angepasst und eingesetzt?

Ein Framework besteht aus "halbfertigen" Anwendungsteilen, bei denen an bestimmten Stellen, sogenannten *Hot Spots*, eine Konfigurierung bzw. Spezialisierung (meist unter Ausnutzung von Polymorphie) vorgesehen ist. Schmid schreibt hierzu:

"A hot spot lets you 'plug in' an application-specific class or subsystem, either by selection from a set of those supplied with a black-box framework or by programming a class or subsystem in a white-box framework" ([Sch97], Seite 49)

Die folgende Abbildung verdeutlicht dies:

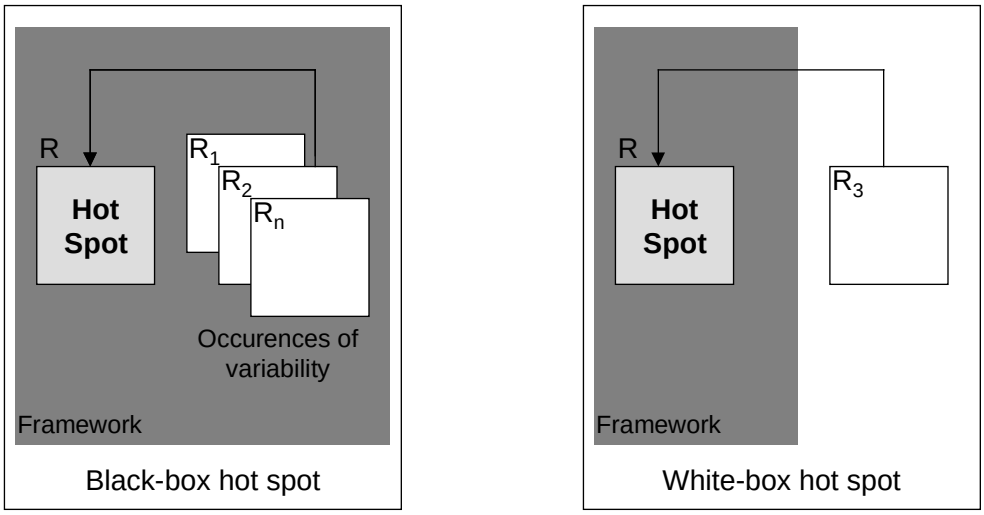


Abbildung 4: Black- und White-Box Hot Spots (nach [Sch97])

Die Abbildung zeigt zwei Möglichkeiten, einen Hot Spot anwendungsspezifisch anzupassen. Die Anpassung wird in beiden Fällen dadurch vorgenommen, dass ein Exemplar einer anwendungsspezifischen Klasse in den Hot Spot "eingeklinkt" wird. Diese Klasse muss Subtyp eines Typs R sein, der durch den Hot Spot vorgegeben ist.

Der Hot Spot selbst ist durch ein sogenanntes *Hot-Spot-Subsystem* (s. [Sch96]) realisiert, dass einen Entwurf für ein Exemplar von R und die anderen beteiligten Objekte und ihr Zusammenspiel vorgibt. Ein einfaches Beispiel für einen Hot Spot ist etwa die Verwendung eines `ActionListener` (in der Rolle von R) im Java-GUI-Framework *AWT*. Ein solcher `ActionListener` kann bei einem GUI-Element des Typs `Button` registriert werden und wird bei Betätigung des Buttons informiert. Der Entwickler stellt eine eigene Implementation von `ActionListener` bereit, um auf den Knopfdruck passend reagieren zu können[13]. `Button`, `ActionListener` und ihr Zusammenspiel bilden das Hot-Spot-Subsystem

[13] Dieses Beispiel illustriert auch einen gewissen Zusammenhang zwischen Entwurfsmustern und Hot Spots, denn das Zusammenspiel von `Button` und `ActionListener` entspricht dem Beobachtermuster (s. [GHJ+]). Die

In einem Black-Box-Framework werden zu einem solchen Hot Spot mehrere konkrete Klassen (in der Abbildung R_1 bis R_n) bereitgestellt, die Subtypen von R sind. Der Anwendungsentwickler kann aus den vorgesehenen Klassen eine auswählen und ein entsprechendes Exemplar mit dem vorgesehenen Objektgeflecht instantiieren.

Bei einem *White-Box-Framework* geschieht die Anpassung dagegen durch Subklassenbildung. Es ist der Anwendungsentwickler, der für den Hot Spot eine konkrete Klasse (die Subtyp von R ist) bereitstellen muss. Die Variabilität ist beim White-Box Hot Spot größer, doch hat der Entwickler mehr Arbeitsaufwand und muss erheblich mehr Wissen über die Implementationsdetails des Frameworks besitzen, um eine entsprechende Klasse zu implementieren.

Da mit der Framework-Architektur auch große Teile des Kontrollflusses für die darauf basierenden Programme vorgegeben werden, spricht man auch von der "Umkehrung des Kontrollflusses"[14].

2.1.6 JWAM in der Übersicht

Im vorigen Abschnitt wurde dargestellt, wie Frameworks generische Lösungen für verwandte Probleme implementieren. Die im WAM-Ansatz beschriebenen Ideen und Entwurfsmuster bieten solche Lösungsansätze für die Konstruktion interaktiver Anwendungssysteme. Das auf dem WAM-Ansatz basierende, in Java implementierte Framework JWAM setzt dementsprechend die wichtigsten Entwurfsmuster des WAM-Ansatzes konstruktiv um.

In [Zül98] wird (in Kapitel 3) ein Schichtenmodell für WAM-basierte Anwendungssysteme vorgeschlagen. Das JWAM-Framework ist in Anlehnung an diese Schichtenarchitektur strukturiert worden, wie die folgende Abbildung verdeutlicht:

---

meisten in [GHJ+] beschriebenen Entwurfsmuster bieten sich für eine analoge Verwendung in Hot-Spot-Subsystemen an.

[14] Die "Inversion Of Control" ist allerdings nicht nur bei objektorientierten Frameworks anzutreffen. Auch prozedurale *Application Programming Interfaces* können dieses Charakteristikum aufweisen, das auch als "HOLLYWOOD-Prinzip" ("Don't call us, we'll call you") bezeichnet wird — siehe beispielsweise [PY96], Seite 60.

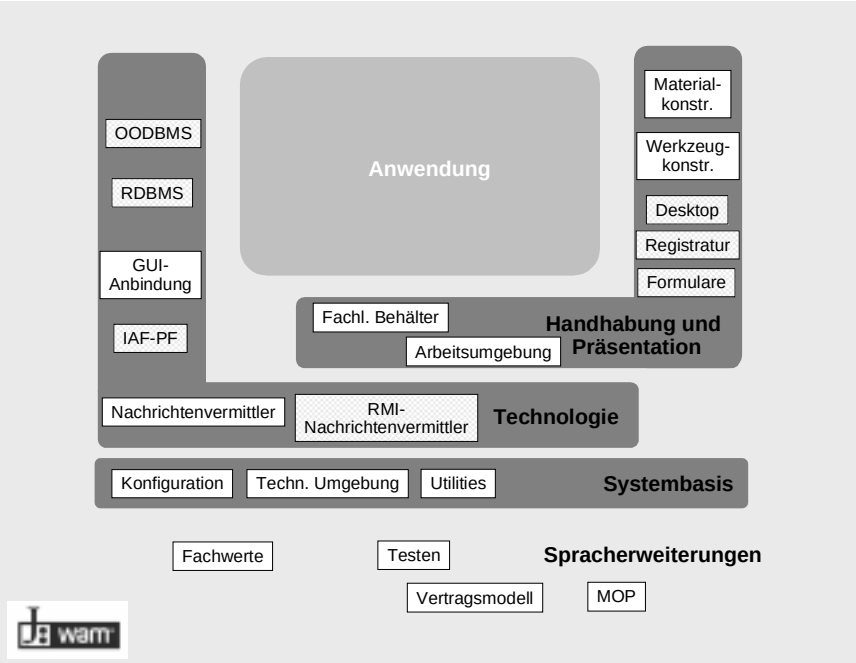


Abbildung 5: Das JWAM-Framework (Schichtenarchitektur)

Die Abbildung zeigt die Schichten *Spracherweiterungen, Systembasis, Technologie, Handhabung und Präsentation.* Darin sind (als helle Rechtecke) die in den jeweiligen Schichten enthaltenen Sub-Frameworks dargestellt, wobei die etwas dunkleren unter diesen *optionale* Bestandteile des JWAM-Frameworks sind, sogenannte *Framework-Komponenten* (s. Abschnitt 2.1.6.2, “Framework-Komponenten”, ab Seite 25). Das von den oberen Schichten gebildete “U” soll den Rahmen darstellen, den das Rahmenwerk den darauf aufsetzenden Anwendungen bzw. domänenspezifischen Schichten bietet.

Jeder der Schichten ist eine Menge von Klassen zugeordnet. Das grundsätzliche Prinzip einer Schichtenarchitektur (und ihr Vorteil) besteht darin, dass sich nur die jeweils höheren Schichten auf die jeweils niedrigeren abstützen; niemals darf eine niedrigere Schicht (also aus objektorientierter Sicht die in ihr enthaltenen Klassen) von einer höheren Schicht abhängig sein. Aus dieser Forderung folgt, dass Klassen einer bestimmten Schicht nur Klassen (Typen) niedrigerer Schichten benutzen oder von diesen erben dürfen.

Dieses Prinzip ermöglicht bei Änderungen im System eine Begrenzung der notwendigen Folgeänderungen (niedrigere Schichten sind durch Änderungen an höheren Schichten nicht tangiert) und erleichtert damit auch die Dekomposition des Systems in einzeln auslieferbare Teile[15].

Der gezeigte Schichtenaufbau strukturiert das JWAM-Framework entlang der Dimension “Anwendungsnähe”: Die untersten Schichten stellen Basisdienste bereit oder kapseln technische Schnittstellen, während die oberen Schichten fachliche[16] Konzepte und abstrakte Umsetzungen von Entwurfsmustern enthalten.

Die Schichten im Einzelnen:

[15] Die explizit modellierten Schichten bilden natürlich nicht die einzigen möglichen Dekompositionslinien, es sind z.T. auch schichtenübergreifende Zerlegungen sinnvoll (s. hierzu z.B. [Zül98] ab Seite 369).

[16] Da JWAM ein Framework ist, das als Anwendungsgebiet die Konstruktion interaktiver Anwendungen nach dem WAM-Ansatz hat, sind in diesem Fall WAM-Konzepte als fachliche Konzepte zu verstehen.

Spracherweiterungen

In der Spracherweiterungsschicht sind solche Teile des Frameworks gruppiert, die als wünschenswerte Erweiterungen der Programmiersprache Java betrachtet werden können. Wichtige Bestandteile:

- Eine Bibliothek mit Klassen, die das Überprüfen Klasseninvarianten und von Vor- und Nachbedingungen entsprechend dem von Bertrand Meyer (z.B. in [Mey97]) formulierten *Vertragsmodell* ermöglichen[17].
- Ein Sub-Framework für das Testen des Frameworks auf Basis von Unit Tests[18].
- Schnittstellen und abstrakte Basisklassen für *Fachwerte* sowie einige konkrete Implementationen.

Die Verwendung der Spracherweiterungen ist im Framework ubiquitär. Deshalb ist diese Schicht auf der Abbildung nicht wie die anderen Schichten als ein solider Block dargestellt, sondern erscheint wie eine Atmosphäre oder Flüssigkeit, die die anderen Schichten umgibt (und damit an jede dieser Schichten angrenzt).

Systembasis

Die Systembasisschicht kapselt technische, plattformabhängige Komponenten, die für die Realisierung des Frameworks und der darauf aufsetzenden Anwendungen notwendig sind. Die hohe Portabilität der Programmiersprache Java (inklusive der GUI-Frameworks *AWT* und *Swing*) und die gute Ausstattung mit standardisierten Anschlüssen für externe Systeme (Filesystem, Netzwerkanbindung, JDBC-API für den Anschluss relationaler Datenbanken) dünnen diese Schicht allerdings auf wenige Klassen aus.

Technologie

Die Technologieschicht umfasst Modelle der verwendeten Technik (wie Persistenz) und allgemein verwendbare softwaretechnische Gegenstände, die keine anwendungsfachliche Interpretation besitzen.

Handhabung und Präsentation

Die Handhabungs- und Präsentationsschicht enthält abstrakte Implementationen der wichtigsten auf Handhabung und Präsentation bezogenen WAM-Entwurfsmuster und orientiert sich an den bereits beschriebenen WAM-Entwurfsmetaphern. Zentrale Bestandteile sind:

- Ein Sub-Framework für die Werkzeug- und Materialkonstruktion. Die Werkzeugkonstruktion wird eingehender beschrieben in Abschnitt 5.1.11.4(“Exkurs: Werkzeugkomposition in

[17] Solche Vor- und Nachbedingungen sind in der Programmiersprache *Eiffel* übrigens ein integraler Sprachbestandteil.

[18] Möglichst zu jeder Klasse im Framework soll eine Klasse geschrieben werden, die diese testet und so gewissermaßen zu einem Teil der Spezifikation wird. Die Testklassen gehören zu denselben Schichten und Java-Packages wie ihre Gegenstücke und erben aus dem Test-Framework, welches es ermöglicht, das gesamte Framework automatisiert durchzutesten. Das Test-Framework in JWAM ist gegenwärtig noch eine dünne Kapsel um *JUnit* von Kent Beck und Erich Gamma (siehe [BG98]).

JWAM", ab Seite 85), die Materialkonstruktion in Abschnitt 5.1.11.1 (" Materialien in JWAM", ab Seite 80).

- Die Framework-Komponenten für
  - den JWAM-Desktop, der einen Einzelarbeitsplatz mit Werkzeugen und Materialien entsprechend der Werkzeugmetapher visualisiert (dieser ist beschrieben in [Lip99])
  - die Registratur, in der Materialien persistent aufbewahrt werden können. Mit Blick auf Kooperationsunterstützung wird die Registratur in Abschnitt 3.1.3 (ab Seite 34) kurz beschrieben.
  - das Formularwesen, das eine Formularklasse und passende Werkzeuge enthält. Ein Formular ist besitzt in Analogie zum physichen Formular benannte Felder, die ausgefüllt werden können (hier nur mit Fachwerten).

Software- und Framework-Komponenten in JWAM

Der Fremdwörter-Duden übersetzt das Wort "Komponente" mit "Bestandteil eines Ganzen". In diesem Abschnitt beschreibe ich mit den Begriffen *Software-Komponente* und *Framework-Komponente* zwei unterschiedliche Interpretationen dieses Konzepts und stelle dar, wie sich das JWAM-Framework zu ihnen positioniert.

2.1.6.1 Software-Komponenten

Software-Komponenten sind Einheiten, aus denen Anwendungssysteme zusammengefügt werden können; es handelt sich damit auch um Einheiten der Software-Wiederverwendung. Szyperski und Pfister definieren eine solche Software-Komponente wie folgt:

"A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties" ([Szy97], Seite 34)

Diese Definition betont einerseits, dass eine Software-Komponente ihre Schnittstellen im Sinne eines Vertrags spezifizieren und den von ihnen benötigten Kontext angeben muss. Dies bedeutet, dass die Komponente neben den von ihr angebotenen Dienstleistungen auch diejenigen angibt, die sie (vom Kontext) selbst in Anspruch nehmen muss, um ihre Leistung zu erbringen. Andererseits hebt die Definition hervor, dass eine Software-Komponente unabhängig ausgeliefert werden kann und von dritter Seite mit anderen Komponenten zu einer Anwendung zusammengesetzt wird. Szyperski fügt dieser Definition an anderer Stelle noch hinzu, dass Software-Komponenten *binäre* Einheiten seien, die zusammenarbeiten, um ein funktionierendes System zu bilden (vgl. [Szy97], Seite 3).

Diese Beschreibung von Software-Komponenten läuft auf ablauffähige "Plug-And-Play"-Bausteine hinaus und zielt auf die Vision eines Komponenten-Marktes, auf dem Hersteller technisch austauschbarer, aber qualitativ durchaus unterschiedlicher Komponenten miteinander konkurrieren. Entsprechend nennt Szyperski als ein Beispiel einer erfolgreichen Komponenten-Architektur moderne Betriebssysteme und die darauf ablauffähigen Programme (diese stellen die Komponenten dar) oder die heutigen Web-Browser und die zu ihnen von Drittfirmen erhältlichen sogenannten Plug-Ins.

Das JWAM-Framework stellt zum gegenwärtigen Zeitpunkt keine Unterstützung für Software-Komponenten im obigen Sinne bereit. Eine mit JWAM konstruierte Anwendung bietet deshalb in der Regel keinen Mechanismus, um sie zur Laufzeit durch neue Komponenten zu erweitern oder solche Komponenten auszutauschen. Dabei gäbe es mit Werkzeugen, Materialien und Automaten

(bzw. den zugehörigen Java-Class-Dateien) geeignete binäre Einheiten (mehrere Werkzeuge, Materialien und Automaten gemeinsam in einer *Java Archive* (JAR)-Datei würden eine Software-Komponente bilden). Die Weiterentwicklung des Frameworks in diese Richtung ist aber möglich und naheliegend.

2.1.6.2 Framework-Komponenten

Bei der Entwicklung einer konkreten Anwendung auf Basis eines objektorientierten Frameworks werden meist nicht alle Bestandteile dieses Frameworks genutzt. Es ist deshalb von Vorteil, wenn in diesem Fall möglichst viele der nicht verwendeten Bestandteile weggelassen werden können[19].

Um dies für das JWAM-Framework zu ermöglichen, wurde das Framework in einen Kern und optionale Bestandteile partitioniert. Diese optionalen Bestandteile werden als *Framework-Komponenten* bezeichnet. Beispiele für Framework-Komponenten von JWAM sind der Desktop, die Registratur und das Formularwesen. In Abbildung 5 “Das JWAM-Framework (Schichtenarchitektur)” sind diejenigen Bestandteile, die Framework-Komponenten sind, durch eine dunklere Farbe hervorgehoben.

Die Aufteilung des JWAM-Frameworks in Kern und Komponenten spiegelt sich in der Package-Struktur[20] des Frameworks wieder: Die Klassen des Kerns befinden sich unterhalb des Packages `de.jwam` (für Klassen der Werkzeugkonstruktion z.B. unterhalb `de.jwam.handling.basicconstruction.toolconstruction`), während sich die Klassen der Framework-Komponenten unter `de.jwamx` einordnen (für den JWAM-Desktop z.B. unterhalb `de.jwamx.handling.desktop`).

Der Kern lässt sich ohne die Framework-Komponenten vollständig kompilieren. Mit einer auf JWAM basierenden Anwendung müssen demnach nur der JWAM-Kern und die tatsächlich benutzten JWAM-Framework-Komponenten ausgeliefert werden.

Wie verhalten sich nun Framework-Komponenten zu Software-Komponenten?

Grundsätzlich ist ihr Komponentencharakter auf unterschiedlichen Ebenen angesiedelt, denn die Entscheidung über die Verwendung oder Nichtverwendung einer Framework-Komponente wird zur Konstruktionszeit einer Anwendung getroffen, während eine Software-Komponente potenziell zur Laufzeit einer Anwendung hinzugefügt oder aus ihr entfernt wird.

Für den Fall, dass JWAM aber einmal Software-Komponenten vorsieht, sind Framework-Komponenten die richtige Form, um passende Implementationen gemeinsam mit dem Framework auszuliefern. Das Kompilat einer solchen Framework-Komponente wäre dann eine Software-Komponente.

---

[19] Unter “Weglassen” ist zu verstehen, dass die Anwendung ohne die betreffenden Klassen ausgeliefert werden kann.

[20] In Java werden die Klassen in Packages gruppiert. Die Packages bilden eine Modul- und Namenshierarchie.

3 Kooperationsunterstützung für JWAM

Das Ziel dieses Kapitels ist es, in kurzer Form die CoJAC-Arbeiten, und die aus ihnen hervorgegangenen Framework-Komponenten, die JWAM um Kooperationsunterstützung erweitern sollen, vorzustellen. Dabei möchte ich sie anhand von bewährten Kriterien aus dem Forschungsgebiet CSCW und Begriffen des WAM-Ansatzes klassifizieren.

Um dies zu ermöglichen, gebe ich zunächst eine kurze Übersicht über CSCW (und gehe auch auf den durch kommerzielle Systemanbieter dominierten Bereich der Workflow-Management-Systeme ein). Dies hat außerdem die Funktion, fundierte Begriffe für die folgenden Kapitel bereitzustellen.

Daran anschließend folgt die kurze Darstellung der CoJAC-Arbeiten und ihre Klassifikation.

Kooperation und Software

In diesem Abschnitt gebe ich jeweils kurze Übersichten zu den Themen *Computer-Supported Cooperative Work (CSCW)* und *Workflow-Management-Systeme (WfMS)* und führe wichtige Begriffe zur Kooperationsunterstützung ein, die aus dem WAM-Ansatz stammen.

3.1.1 Computer-Supported Cooperative Work (CSCW)

Mit der Unterstützung kooperativer Arbeit befasst sich das Forschungsgebiet *CSCW (Computer-Supported Cooperative Work)*. Der Name, geprägt durch Irene Greif und Paul Cashman anlässlich eines Workshops im Jahr 1984 (s. [Grei88]), ist so allgemein formuliert, dass das damit bezeichnete Forschungsgebiet sehr groß und vielgestaltig ist.

Es wurde vielfach versucht, den Begriff CSCW enger und konkreter zu fassen. Doch schon die Interpretation des Teilbegriffs „kooperative Arbeit“ ist nicht unumstritten, wie Schmidt und Bannon ausführen (s. [SB92], S. 12) — die für mich maßgebliche Definition von Züllighoven wird später in diesem Abschnitt wiedergegeben.

Weniger eine Einschränkung als vielmehr eine Fokussierung des Forschungsgebietes CSCW verfolgen Schmidt und Bannon, indem sie drei Kerngebiete benennen (in [SB91]):

- „articulating cooperative work“
- „sharing an information space“
- „adapting the technology to the organisation and *vice versa*“

Das erste Gebiet untersucht die Abstimmungsprozesse zwischen Personen, die miteinander kooperativ zusammenarbeiten. Eine besondere Betonung legen die Autoren hierbei auf die Betrachtung der informalen, sozial eingebetteten Kommunikation, die neben der formalen Ebene vorhanden ist („double level language“).

Der zweite Komplex beschäftigt sich mit der Frage, wie „information spaces“ (Informationsräume), also Bereiche, in denen von verschiedenen Seiten Informationen hinterlegt und abgerufen werden können, für kooperative Arbeit genutzt werden können und wie dies durch Computersysteme unterstützt werden kann.

Das dritte Gebiet schließlich betrachtet die Wechselwirkungen zwischen dem Einsatz von CSCW-Systemen in einer Organisation und deren innerer Struktur.

Aus den von Schmidt und Bannon formulierten Kerngebieten wird deutlich, dass CSCW eine interdisziplinäre Forschungsdisziplin ist, die neben der Informatik auch die Gebiete

Organisationslehre, Psychologie und Soziologie einbezieht. Demgegenüber erscheint der ebenfalls populäre Begriff *Groupware* deshalb geradezu „zupackend“, indem er einfach nur „Software für Gruppen“ (oder für *eine* Gruppe) in den Mittelpunkt stellt und sich damit stärker in Richtung der Konstruktion von Software ausrichtet.

Ich möchte für Groupware die folgende Definition von Oberquelle verwenden:

Definition 8: Groupware[21]
Groupware ist Mehrbenutzer-Software, die zur Unterstützung von kooperativer Arbeit entworfen und genutzt wird und die es erlaubt, Informationen und (sonstige) Materialien auf elektronischem Wege zwischen den Mitgliedern einer Gruppe koordiniert auszutauschen oder gemeinsame Materialien in gemeinsamen Speichern koordiniert zu bearbeiten.

Die von Oberquelle gewählte Definition stellt die gemeinsame Nutzung von Informationen und Materialien in den Mittelpunkt, ob dies nun über eine elektronische Versendung oder einen gemeinsamen Speicher geschieht. Viele Anwendungen, die als Groupware klassifiziert werden, sind Erweiterungen von Einzelplatzsystemen um Gruppenfunktionalität, die genau diese gemeinsame Nutzung von Materialien ermöglicht. Die hinzugefügte Funktionalität geht je nach System aber auch darüber hinaus. So werden Textverarbeitungssysteme um Funktionen erweitert, die Änderungen durch verschiedene Benutzer deutlich machen („Collaborative Writing“) oder sogar gleichzeitiges Editieren ermöglichen.

Der Begriff Groupware hat unter anderem aufgrund dieser Entwicklung die Konnotation, die Zusammenarbeit aus der Sicht des Individuums zu erschließen und deshalb den Kooperierenden viel Handlungsfreiheit zu lassen.

Zusammenfassend ist zur Abgrenzung der Begriffe CSCW und Groupware zu sagen, dass CSCW die Bezeichnung eines Forschungsbereiches ist, während Groupware eine bestimmte Kategorie daraus hervorgehender Produkte bezeichnet. Schmidt und Bannon merken hierzu an:

„While many commercial reports still refer to the Groupware field, the term CSCW has come to be preferred in the research community due to its more comprehensive remite,...“ ([SB92], S. 9)

Es gibt eine Reihe von Möglichkeiten, CSCW-Systeme zu klassifizieren. Ich möchte in diesem Abschnitt zwei davon vorstellen und beginne mit dem “3K-Modell” (so nennt es Borghoff in [BS98] , Seite 127).

Die drei “K’s” stehen für Kommunikation, Koordination und Kooperation. Die damit bezeichneten Handlungen bauen aufeinander auf. Teufel gibt im Zusammenhang mit Gruppenarbeit folgende Definition für Kommunikation

Definition 9: Kommunikation[22]
Kommunikation ist die Verständigung mehrerer Personen untereinander.

Die in der Definition geforderte Verständigung zwischen den Personen kann auf einer technischeren Ebene als “verlässlicher und hinreichend schneller Austausch von Informationsobjekten” übersetzt werden (vgl. [TSM+95], Seite 26).

Kommunikation zwischen den betroffenen Personen ist die Voraussetzung für *Koordination*. Malone und Crowston definieren Koordination als “managing dependencies between activities”

[21] aus [Obe91], Seite 5

[22] aus [TSM+95], Seite 12

(in [MC94], Seite 90). Dabei unterscheiden sie verschiedene Typen von Abhängigkeiten zwischen Tätigkeiten, unter anderem:

- *Shared resources:* Dieser Abhängigkeitstyp ist dadurch gekennzeichnet, dass für mehrere Tätigkeiten dasselbe begrenzt vorhandene Gut benötigt wird (gleich, ob es sich nun um ein bestimmtes Dokument, um Geld, Speicherplatz oder das Expertenwissen einer bestimmten Person handelt). Dieses Gut muss in einem solchen Konfliktfall nach einem sinnvollen Prinzip zugeteilt werden. Auf den ersten Blick scheint dies nur eine Frage der Zuteilungsreihenfolge zu sein, bei einem Gut, welches verbraucht werden kann, geht es allerdings auch darum, wer überhaupt etwas davon erhält.
- *Producer/Consumer:* Diese Art von Abhängigkeit besteht zwischen zwei Tätigkeiten, bei denen das Produkt oder Arbeitsergebnis der einen für die Durchführung der anderen benötigt wird. Typische Beispiele für das Auftreten dieser Abhängigkeitsform sind Fertigungsprozesse, bei denen ein Produkt Schritt um Schritt weiterverarbeitet wird, oder Vorgänge in Unternehmen, bei denen eine Person für ihre Tätigkeit Informationen benötigt, die eine anderer Person bereitstellen muss.

Züllighoven gibt für Koordination eine etwas engere Definition als Crowston und Malone:

Definition 10: Koordination[23]
Koordination ist der Prozess oder der Mechanismus zur Abstimmung von Arbeitsteilung bei kooperativer Arbeit. Koordination kann auf wechselseitigen Konventionen oder ausdrücklichen Regeln beruhen.

Durch die Verwendung des Begriffs “Arbeitsteilung” wird der Begriff der Koordination hier auf Arbeit verengt, die in arbeitsteiligen Prozessen organisiert wird. In diesen Prozessen besteht zwischen den einzelnen Tätigkeiten typischerweise eine Producer/Consumer-Abhängigkeit, wie sie oben beschrieben wurde. Die Definition konzentriert sich damit auf den für die Themenstellung dieser Arbeit relevanten Bereich, denn die Framework-Komponente *Vorgangsmappen* (deren Entwicklung ja beschrieben werden soll) hat die Unterstützung genau solcher Arbeitsprozesse als Ziel.

Die eben gegebene Definition benutzt bereits den Begriff der kooperativen Arbeit, der von Züllighoven so beschrieben wird:

Definition 11: Kooperative Arbeit[24]
Bei kooperativer Arbeit arbeiten verschiedene Personen geplant und koordiniert zusammen, um ein gemeinsames Ergebnis zu erreichen.

Diese Definition impliziert, dass die Arbeitsbeiträge der Beteiligten sich zu einem Ganzen (dem gemeinsamen Ergebnis) addieren oder sogar aufeinander aufbauen. Daraus folgt:

- Die Beteiligten müssen sich abstimmen, damit die Arbeitsbeiträge zusammenpassen.

[23] aus [Zül98], Seite 428

[24] aus [Zül98], Seite 428

- Wenn die Arbeitsbeiträge der zusammenarbeitenden Personen aufeinander aufbauen, müssen die Arbeitsergebnisse und –gegenstände zwischen ihnen weitergegeben oder zugänglich gemacht werden.

Die Klassifizierung von CSCW-Systemen nach dem 3K-Modell basiert auf den gerade diskutierten Begriffen Kommunikation, Koordination und Kooperation. In einem CSCW-System stellt die Kommunikationsunterstützung die notwendige Basis dar, auf der eine mögliche Koordinationsunterstützung aufsetzt (die Abstimmung der Arbeitsteilung zwischen beteiligten Personen stellt eine spezielle Form der Kommunikation dar). Kooperationsunterstützung wiederum beinhaltet sowohl Kommunikations- als auch Koordinationsunterstützung.

Je nach Betonung der einen oder anderen Unterstützungsfunktion lassen sich die verschiedenen Applikationstypen in dem in Abbildung 6 gezeigten Dreieck positionieren.

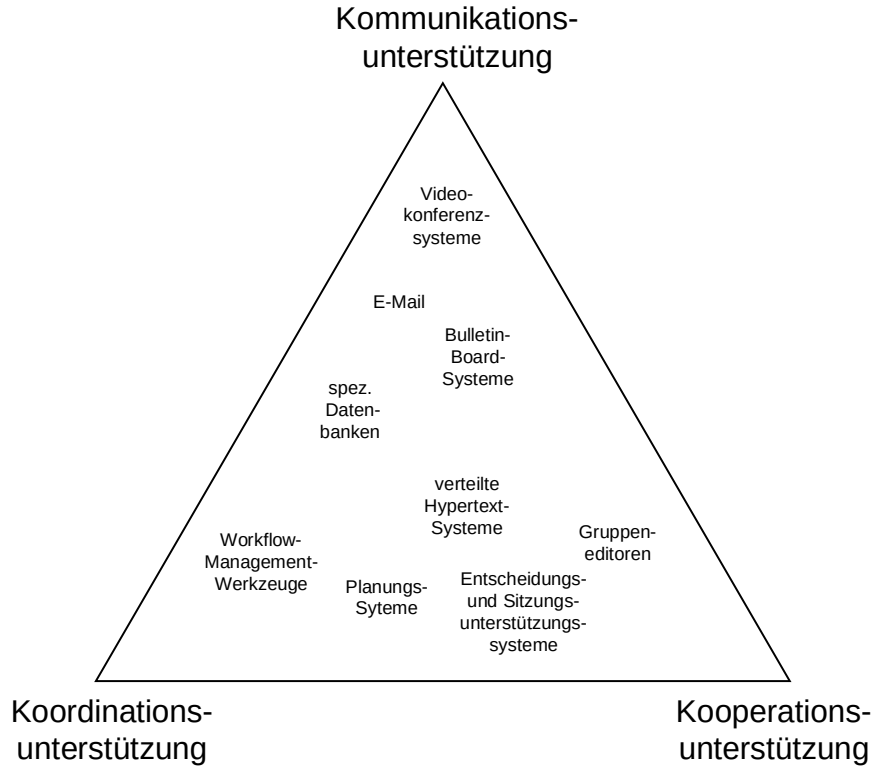


Abbildung 6: Klassifikationsschema nach Unterstützungsfunktionen (nach [TSM+95])

Abbildung 6 zeigt verschiedene CSCW-Systeme je nach Tendenz eingeordnet zwischen den drei "Polen" Kommunikations-, Koordinations-, Kooperationsunterstützung. So sind etwa Videokonferenzsysteme oder e-Mail konzipiert, um Kommunikation zu unterstützen, während Koordination und Kooperation nicht durch diese Systeme direkt unterstützt werden. Gruppeneditoren hingegen, bei denen das von einer Gruppe gemeinsam zu erreichende Ergebnis (z.B. in Form eines Textes) den Mitgliedern ja stets vor Augen ist, haben ihren Schwerpunkt klarerweise bei der Kooperationsunterstützung (vgl. [TSM+95], Seite 27/28).

Eine zweite Klassifikation von CSCW-Systemen zeigt die folgende Matrix:

Time / Place	Same	Different but predictable	Different and unpredictable
Same	Meeting facilitation	Work shifts	Team rooms
Different but predictable	Tele/video/desktop conferencing	Electronic mail	Collaborative writing
Different and unpredictable	Interactive multicast seminars	Computer bulletin boards	Workflow

Abbildung 7: „Time/Space Matrix“ (nach [Gru94], S. 25)

Die Matrix betrachtet die Dimensionen Ort und Zeit. Der Ort bezieht sich auf den Aufenthaltsort der beteiligten Benutzer bei der rechnergestützten Kooperation. Es wird danach unterschieden, ob sich die an der Kooperation beteiligten Personen am selben Ort oder an verschiedenen Orten befinden. Befinden sie sich an verschiedenen Orten, wird noch einmal danach unterschieden, ob diese Orte (für die jeweils anderen Kooperationsbeteiligten) vorhersagbar oder aber nicht vorhersagbar und damit unbekannt sind. Die zeitliche Dimension wird analog untergliedert.

Zur Erläuterung der Matrix[25] betrachte ich hier Workflow und Electronic Mail:

Workflow[26]: Workflow-Management-Systeme zerlegen Arbeitsprozesse in Einzelschritte und weisen diese in wohldefinierter Weise (und häufig über Rollen) den Bearbeitern zu. Workflow-Management-Systeme organisieren die Arbeitsprozesse in tayloristischer Weise, diese werden zergliedert und teilweise dekontextualisiert. Die verschiedenen Bearbeiter sind örtlich verteilt, und der einzelne Bearbeiter hat in der Regel keinen Einblick, wer genau nach ihm die Bearbeitung eines Prozesses fortführt, und wann und wo dies geschieht. Darum sind sowohl Zeit als auch Ort als verschieden und unvorhersagbar gekennzeichnet.

Electronic Mail: Menschen, die miteinander über e-Mails kommunizieren, tun dies zu verschiedenen Zeiten und an verschiedenen Orten. Typischerweise ist für jemand, der zu diesem Zweck eine e-Mail verschickt, der Zeitpunkt, wann diese Mail gelesen (und möglicherweise beantwortet) wird, eingrenzbar: Er erwartet etwa, dass die e-Mail spätestens am folgenden Tag zur Kenntnis genommen wird. Typischerweise weiß er auch, wo der Adressat die Mail lesen wird: an seinem Arbeitsplatz, in seinem Büro, dessen Ort bekannt ist. Damit wird Electronic Mail hinsichtlich räumlicher wie zeitlicher Determiniertheit als verschieden, aber vorhersagbar eingeordnet.

Grudin hebt in dem Artikel, in dem er diese Matrix vorstellt, selbst hervor, dass eine eindeutige Zuordnung nach der obigen Matrix nicht immer möglich ist. Bei Electronic Mail beispielsweise ist es aus heutiger Sicht (die dargestellte Matrix mit ihren Einträgen stammt aus dem Jahr 1994) bei zunehmender Nutzung von *Mail Forwarding[27]*, *SMS[28]* und ähnlichen Diensten abhängig von

[25] Eine Erläuterung dieser Matrix, die (außer Workflow) *alle* aufgeführten Anwendungen kurz beschreibt, findet sich z.B. in [RW98] ab Seite 59.

[26] Siehe hierzu den folgenden Abschnitt 3.1.2 („Workflow-Management-Systeme“) ab Seite 31.

[27] *Mail Forwarding* bezeichnet einen Mechanismus, der für eine e-Mail-Adresse eingehende Mails an andere Adressen weiterleitet

[28] *SMS* steht für *Short Message Service*. Dieser Dienst erlaubt es, Kurznachrichten mit Mobiltelefonen zu empfangen, wo sie auf dem Display gelesen werden können. Die Kurznachrichen können von anderen Mobiltelefonen aus oder aus dem Internet versendet werden.

der konkreten Ausgestaltung der Zusammenarbeit, ob der Ort, an dem jemand e-Mails liest, vorhersagbar ist oder nicht.

3.1.2 Workflow-Management-Systeme

Workflow-Management-Systeme (WfMS) sind Softwaresysteme, die die Organisation eines Unternehmens (einer Verwaltung) unter dem Blickwinkel betrieblicher Abläufe betrachten und die Steuerung des Arbeitsflusses (*Workflow)* zwischen den beteiligten Stellen übernehmen.

Die *Workflow Management Coalition* (WfMC)[29] beschreibt WfMS wie folgt:

„A Workflow Management System is one which provides procedural automation of a business process by management of the sequence of work activities and the invocation of appropriate human and/or IT resources associated with the various activity steps (...)

Workflow Management System: A system that completely defines, manages and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic."

([Wor95], Seite 6)

Die Aufgabe eines WfMS liegt demnach in der Automatisierung von Geschäftsprozessen; das sind diejenigen Vorgänge innerhalb eines Unternehmens, die einen wesentlichen Beitrag zu einem (nicht notwendigerweise ökonomischen) Unternehmenserfolg leisten[30].

Ein Geschäftsprozess wird bei Einsatz eines WfMS auf einen Workflow abgebildet, der eine (vollständige oder teilweise) Automatisierung des Geschäftsprozesses darstellt[31]. Aufgrund der Orientierung an Geschäftsprozessen werden WfMS von vielen Unternehmen als Hilfsmittel gesehen, um ein sogenanntes *Business Process Reengineering* zu implementieren, also eine Umgestaltung der Geschäftsprozesse mit dem Ziel der Effizienzsteigerung konkret umzusetzen.

Der Workflow besteht aus den einzelnen Arbeitstätigkeiten, die den Geschäftsprozess ausmachen. Die Ausführung dieser Tätigkeiten wird vom WfMS gesteuert, wobei den Tätigkeiten menschliche oder informationstechnische "Ressourcen" zugeordnet sind, die vom WfMS eingesetzt werden. Dabei beruht die Steuerung auf einer formalen Repräsentation der Workflow-Logik, der sogenannten *Workflow-* oder *Prozessdefinition*. Ein auf Basis einer solchen Workflowdefinition ablaufender Arbeitsprozess wird als *Workflow-Instanz* bezeichnet.

Workflowdefinitionen werden in WfMS häufig durch Petrinetze (oder Konstrukte, die diesen sehr ähnlich sind) spezifiziert.

---

[29] Die 1993 gegründete WfMC ist eine Organisation, die sich zum Ziel gesetzt hat, den Einsatz der Workflow-Technologien zu fördern, indem sie auf eine einheitliche Terminologie und die Interoperabilität der einzelnen Systeme hinwirkt. Zentrales Projekt ist dabei das *Workflow Reference Model*, das im Wesentlichen die Schnittstellen von fünf Standardkomponenten für ein WfMS und ihre Zusammenspiel definiert (z.B. Workflow-Ablauf-Komponente, Werkzeuge zur Workflow-Modellierung usw.) Verschiedene konform gestaltete WfMS können über diese Schnittstellen zusammenarbeiten.

[30] Vgl. [JBS97], Seite 486.

[31] Die WfMC definiert den Workflow wie folgt: "The computerised facilitation or automation of a business process, in whole or part." ([Wor95], Seite 6)

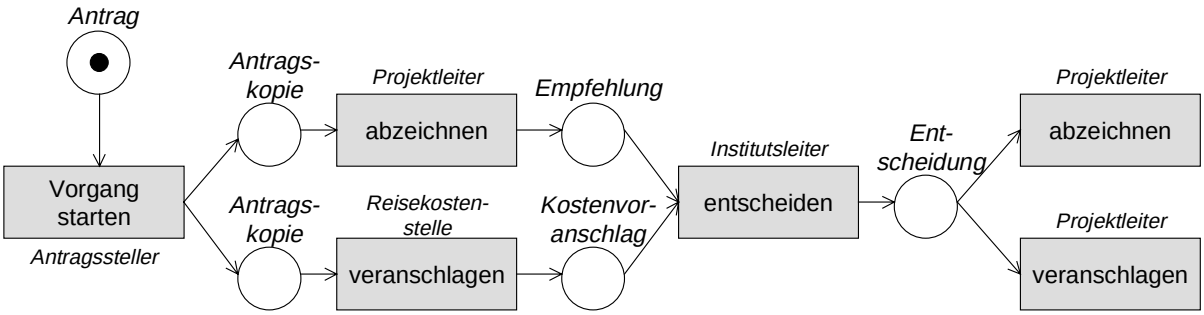


Abbildung 8: Petrinetz am Beispiel des Vorgangs eines Dienstreiseantrags (nach [BS98], Seite 359)

Abbildung 8 zeigt dies am Beispiel eines Dienstreiseantrags. In dieser Darstellung repräsentieren die Stellen (die Kreise) die jeweiligen Vorgangsinformationen und die Transitionen (die Rechtecke) die Aktionen, die durch die Akteure ausgeführt werden. Die Komplexität von Petri-Netzen und ähnlichen Modellierungsmitteln führt dazu, dass die Workflowdefinitionen i.d.R. nicht von den in die Prozesse involvierten Mitarbeitern, sondern nur von entsprechenden Spezialisten gestaltet und geändert werden können.

WfMS sind als Ausprägungen der *ablaufsteuernden Sichtweise* bei der Software-Entwicklung anzusehen, die von Züllighoven wie folgt charakterisiert wird:

Definition 12: ablaufsteuernde Sichtweise[32]
Die ablaufsteuernde Sichtweise bei der Softwareentwicklung kann durch folgendes Vorgehen charakterisiert werden:
1. *Analysiere die Arbeitsabläufe im Anwendungsbereich und erkenne die wiederholbaren Anteile.*
2. *Bilde daraus einen optimalen, allgemeingültigen Ablauf*
3. *Übertrage den algorithmisierbaren Anteil auf den Computer*
4. *Steuere die menschlichen Arbeitsanteile (Dateneingabe, Reaktion auf die Datenausgabe) durch das Programm*

Eine Steuerung der Abläufe durch den Computer, wie sie in der obigen Definition beschrieben ist, wird in allen gängigen Definitionen von WfMS explizit als Ziel genannt. Wie sehr die ablaufsteuernde Sichtweise die WfMS bestimmt, lässt sich auch an Formulierungen erkennen, die von “menschlichen oder IT-Ressourcen” sprechen und so eine “maschinelle” Ausführung von Aufträgen durch den Menschen nahelegen. Das passende Leitbild für die aus dieser Sichtweise heraus gestalteten WfMS ist deshalb das *Leitbild Fabrik*, wie es in Abschnitt 2.1.2 (“Leitbilder”, ab Seite 13) beschrieben ist.

Die WfMC illustriert den typischen Aufbau eines WfMS mit folgender Abbildung:

32 aus [Zül98], Seite 78

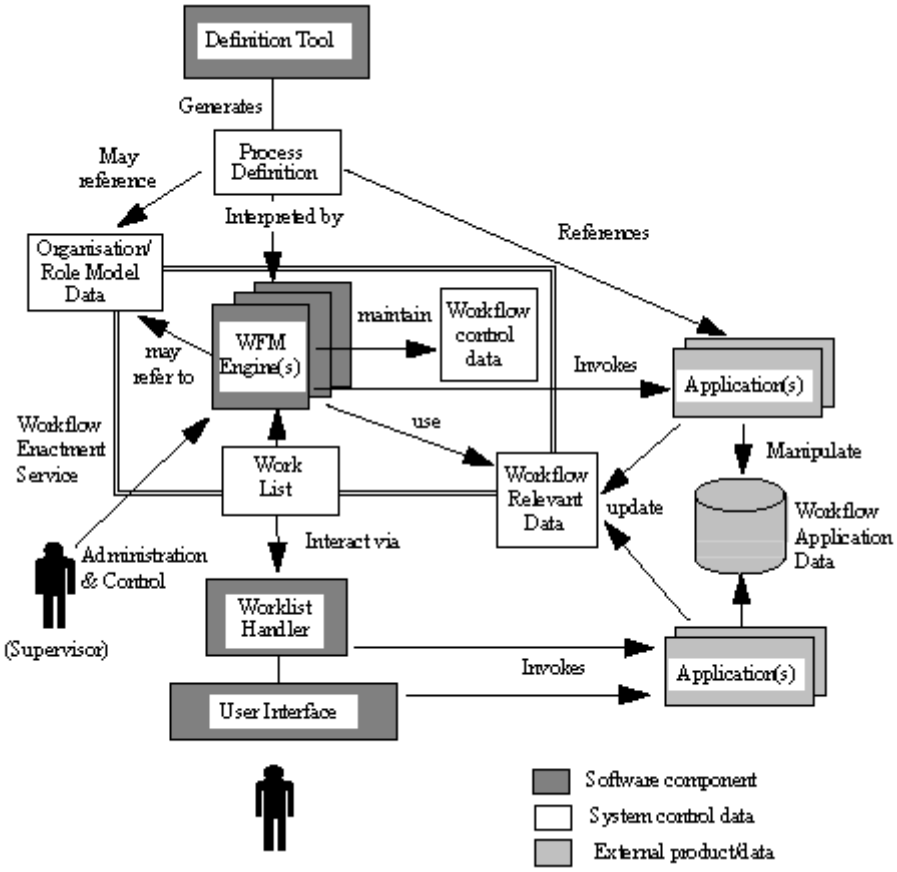


Abbildung 9: "Generic Workflow Product Structure" nach [Wor95], Seite 13

Abbildung 9 zeigt, wie die Prozessdefinition von einer *Workflow Management Engine* interpretiert wird, die sich auf ein Modell der Organisation *(Organisation/Role Model Data)* abstützt[33]:

Die einzelnen Arbeitsschritte einer Workflow-Instanz werden zum Teil automatisiert erledigt (wozu ggf. externe Anwendungen angesprochen werden müssen) oder aber Bearbeitern zugewiesen, denen die *Engine* dazu sogenannte *Work Items*[34] in eine spezielle Liste, die *Work List* stellt. Der Bearbeiter kann seine Work List vom Arbeitplatzrechner aus einsehen und abarbeiten. Bei der Abarbeitung der Work Items unsterstützt das WfMS den Bearbeiter wiederum durch eine möglichst weitgehende Integration externer Anwendungen.

WfMS sind durch ihren tayloristischen Ansatz geeignet, starre Arbeitsprozesse mit einer hohen Wiederholungsrate in einem Unternehmen zu steuern. Aus demselben Grund ist ihr Einsatz aber problematisch, wenn die ablaufenden Arbeitsprozesse hochflexibel und durch die Bearbeiter abänderbar sein müssen. Bei vielen WfMS müssen mögliche "Ausnahmen" bereits in der Workflowdefinition vorgesehen sein, um die nötige Flexibilität zu erhalten (was eine adäquate Behandlung *echter* Ausnahmen unmöglich macht).

Es gibt Bestrebungen, WfMS zu konstruieren, die mehr Flexibilität bieten, in denen auch die Bearbeiter vom vorgesehenen Prozessverlauf abweichen, die Workflowdefinition für eine einzelne Workflow-Instanz abwandeln können. Es ist allerdings fraglich, wie gut sich dies in Systeme einfügen kann, denen die ablaufsteuernde Sichtweise zugrunde liegt.

---

[33] Um Tätigkeiten zu Rollen, Abteilungen oder Personen zuordnen zu können, benötigt das WfMS Kenntnis der internen Organisationsstruktur. Deshalb enthalten die meisten WfMS ein eigenes Modell der Organisation.

[34] Die WfMC definiert ein *Work Item* als "Representation of work to be processed in the context of an activity in a workflow instance." (s. [Wor95], Seite 54)

3.1.3 Weitere WAM-Begriffe zur Kooperationsunterstützung

Der WAM-Ansatz definiert im Bezug auf kooperative Arbeit noch einige weitere Begriffe (über die in Abschnitt 3.1.1 hinaus), die ich hier darstellen möchte.

Nach dem WAM-Ansatz konstruierte Anwendungssysteme enthalten Gegenstände, die die fachlichen Konzepte des Anwendungsbereichs verkörpern. Die für Kooperationszwecke konstruierten Gegenstände werden nach Kooperationsmitteln und –medien unterschieden:

Definition 13: Kooperationsmittel[35]
Ein Kooperationsmittel ist ein fachlich motivierter Gegenstand, der die Kooperation unterstützt. Er vergegenständlicht die Kooperation oder die dabei notwendige Koordination.

Ein Beispiel für Kooperationsmittel sind Textdokumente, an denen man Änderungen nachvollziehen kann (diese sind natürlich gleichzeitig auch Materialien, auf deren Bearbeitung die kooperative Arbeit abzielt). Auch Vorgangsmappen und Laufzettel sind Kooperationsmittel, wie auch persönliche Kalender, die zur Terminplanung eingesetzt werden und den anderen Gruppenmitgliedern zur Einsicht und zur Eintragung neuer Terminvorschläge offen stehen (z.B. Netscape Calendar, Microsoft Schedule etc.).

Definition 14: Kooperationsmedium[36]
Ein Kooperationsmedium ist ein fachlich motivierter Gegenstand, der zur Realisierung von Kooperation in Anwendungssystemen dient.
Gemeinsam ist allen Kooperationsmedien, dass mit ihrer Hilfe Materialien oder Informationen ausgetauscht werden können und dass sie selbst vergegenständlicht sind. Beispiel für Kooperationsmedien sind ein elektronisches Postversandsystem, Gruppenpostfächer oder ein elektronisches Notizbrett.

Das Wort „Medium“ legt mit seiner Bedeutung als „vermittelndes Element“ oder „Träger“[37] nahe, dass ein Kooperationsmedium Informationen oder Materialien weiterleitet, ähnlich wie z.B. die Luft den Schall oder eine Flüssigkeit eine Druckwelle; Informationen bzw. Materialien wandern durch Kooperationsmedien *hindurch* (dies ist als Modellvorstellung zu verstehen, der die technische Umsetzung nicht unbedingt entsprechen muss). Während ein Kooperationsmittel die Kooperation oder Koordination selbst vergegenständlicht, dient das Kooperationsmedium nur ihrer Realisierung.

Der WAM-Ansatz unterscheidet außerdem die Kooperationsunterstützung nach dem Grad, in dem Kooperation und Koordination vergegenständlicht sind, in Unterstützung *impliziter* und *explizter Kooperation.*

---

[35] Aus [Zül98], Seite 429

[36] Aus [Zül98], Seite 429

[37] Aus *Duden Fremdwörterbuch*, 5., neu bearb. u. erw. Aufl., Mannheim; Wien; Zürich: Dudenverl., 1990:

> **Medium**
> **I.** [*lat.*; „Mitte“] *das;* -s, ...dien u. ...dia: 1. (Plural ...dia; selten) Mittel, Mittelglied; Mittler[in], vermittelndes Element. 2. [...] 3. (Plural: ...dien) Träger physikalischer oder chemischer Vorgänge, z.B. Luft als Träger von Schallwellen (Phys., Chem.). [...]

Die Unterstützung impliziter Kooperation ist die schwächere der beiden Unterstützungsformen:

Definition 15: Unterstützung impliziter Kooperation[38]
Bei der impliziten Kooperation wird der konkurrierende Zugriff mehrerer Benutzer auf gemeinsame Ressourcen im Benutzungsmodell ermöglicht und verdeutlicht. Kooperation oder Koordination selbst sind aber nicht vergegenständlicht.

Die Definition legt fest, dass das einer entsprechenden Software zugrundeliegende Benutzungsmodell es den Anwendern erlaubt, sich in Bezug auf eine *Shared Resource*-Abhängigkeit ihrer Tätigeiten zu koordinieren (im Sinne von Malone und Crowston, siehe Abschnitt 3.1.1 ab Seite 26). Dass der konkurrierende Zugriff deutlich wird, ist notwendige und hinreichende Bedingung, damit sich die Anwender in dieser Weise koordinieren können (außerhalb des Systems, denn Kooperation oder Koordination selbst sind ja nicht vergegenständlicht).

Ein Beispiel für ein Kooperationsmittel, das die implizite Kooperation unterstützt, ist die *Registratur*, deren Realisierung durch eine JWAM-Framework-Komponente Andreas Havenstein in [Hav99] beschreibt (s. auch [GHR+00]). Die Registratur ist ein fachlicher, persistenter Behälter, in dessen Benutzungsmodell ein fachliche Umgang mit Originalen und Kopien vorgesehen ist. Materialien können in die Registratur abgelegt und aus ihr entnommen werden. Änderungen an einem registrierten Material kann nur derjenige vornehmen, der das Original entleiht (dies ist immer ein exklusiver Vorgang). Dass ein Original entliehen ist, wird für andere Benutzer deutlich, wenn sie die Registratur sondieren.

An die Unterstützung expliziter Kooperation werden strengere Anforderungen gestellt als an die impliziter Kooperation:

Definition 16: Unterstützung expliziter Kooperation[39]
Bei der expliziten Kooperation wird im Benutzungsmodell deutlich, dass mehrere Benutzer kooperativ in einer gemeinsamen Arbeitsumgebung arbeiten. Geeignete Kooperationsmittel und –medien stehen für die Weitergabe von Materialien und für die Koordination bereit.

Während bei der Unterstützung impliziter Kooperation nur gefordert wurde, dass die Auswirkungen der Handlungen anderer Benutzer deutlich werden, wird nun erwaret, dass die einzelnen Kooperations- und Koordinationsvorgänge und die Benutzer selbst in der Anwendung vergegenständlicht werden.

Die CoJAC-Arbeiten und ihre Klassifikation

In diesem Abschnitt beschreibe ich in kurzer Form die CoJAC-Arbeiten (und insbesondere die aus ihnen hervorgegangenen Framework-Komponenten) unter Verwendung der in diesem Kapitel eingeführten Begriffe.

Anschließend klassifiziere ich die damit realisierbaren Groupwarefunktionalitäten nach dem 3K-Modell und anhand der Time/Space-Matrix (s. Abschnitt 3.1.1 ab Seite 26).

---

[38] aus [Zül98], Seite 430

[39] aus ([Zül98], S.437)

3.1.4 Die Raumkomponente

Entwurf und Konstruktion der Raumkomponente beschreibt Jörn Koch in seiner Diplomarbeit "JWAM-Komponenten zur Kooperationsunterstützung: Die Raumkomponente" ([Koc00])[40].

Diese Framework-Komponente stellt Klassen zur Verfügung, mit deren Hilfe ein virtuelles Gebäude mit benannten Räumen konstruiert werden kann, das mit einem passenden Werkzeug, dem *Room Explorer,* visualisiert wird und auch "begangen" werden kann.

Es geht hierbei allerdings nicht um eine dreidimensionale, fotorealistische Darstellung von Räumen. Die Räume des Gebäudes werden als zweidimensionale, rechteckige Bereiche visualisiert. Es existieren keine Türen, vielmehr können alle (zugänglichen) Räume wie bei einem Puppenhaus direkt "betreten" werden. Die Client/Server-Architektur der Raumkomponente sieht die Bereitstellung des Gebäudes auf einem Server vor, so dass es gleichzeitig von verschiedenen Benutzern von ihren Client-Rechnern aus betreten und erforscht werden kann.

In einem Raum können sich Materialien befinden. Sie können von den Benutzern mittels Drag and Drop von Raum zu Raum, aber auch auf den privaten Arbeitsplatz (Desktop) oder umgekehrt bewegt werden[41].

Durch diese Handlungsmöglichkeiten ergibt sich bereits eine Fülle möglicher Kooperationsformen. So kann durch Konvention das Platzieren eines Gegenstandes an einem bestimmten Ort koordinative Funktion besitzen (eine in einem bestimmten Raum abgelegte Notiz kann z.B. eine Person veranlassen, diese zu lesen und die Anweisungen darauf auszuführen). Koch nennt dies die "Koordination über Material und Ort".

Gemeinsame Materialien können ebenfalls gut in einem Raum aufbewahrt werden. Wenn das Material von einem Benutzer bearbeitet wird, können alle anderen Benutzer ersehen, dass es ihnen nicht mehr zur Verfügung steht.

Als in der Raumkomponente verwirklichte Konzepte, die diese Kooperationsformen unterstützen, nennt Koch:

- Das Konzept des gemeinsamen Ortes (*Shared Workspace*)
- Die Bereitstellung verschiedener Orte (Räume) mit unterschiedlicher fachlicher Bedeutung (kann durch Benennung und räumliche Anordnung ausgedrückt werden)
- Die Stabilität der Topolgie (das virtuelle Gebäude bleibt in seiner Struktur über lange Zeiträume statisch) zur Orientierung
- Die Persistenz der Orte (des Gebäudes) mit den darin enthaltenen Materialien

Die räumliche Vorstellung verbunden mit den Raumbenennungen geben den im Gebäude befindlichen Materialien eine besondere Ordnung und erlauben es den Benutzern, bei der Kommunikation bewährte Formulierungen zu wählen wie etwa: "Haben Sie schon einmal in das Sekretariat geguckt, da könnte das Formular liegen".

---

[40] Die Diplomarbeit [Koc00] ist zum Erstellungsdatum dieser Arbeit noch nicht erschienen und liegt mir auch nicht in Auszügen vor. Ich muss deshalb um Verständnis bitten, wenn einige Aussagen in diesem Abschnitt etwas vage sind oder sich nach Abschluss der Diplomarbeit von Jörn Koch gewisse Abweichungen herausstellen sollten. Die hier gemachten Aussagen beruhen auf dem mehrere Monate vor Abgabe gehaltenen Diplomarbeitsvortrag und sollen nur eine Übersicht über die Raumkomponente geben.

[41] gegenwärtig ist eine Bearbeitung der Materialien nur auf dem privaten Arbeitsplatz, dem Desktop, möglich.

Kooperation: implizit

Kooperationsmedium: Gebäude

Kooperationsmittel: keine

Die Raumkomponente unterstützt implizite Kooperation: Der konkurrierende Zugriff auf Materialien, die sich in den Räumen befinden, wird deutlich, es gibt aber keine Kooperationsmittel, die Kooperation oder Koordination vergegenständlichen. Das virtuelle Gebäude selbst stellt ein Kooperationsmedium dar, weil es dem Materialaustausch dienen kann, Kooperation oder Koordination aber nicht vergegenständlicht.

3.1.5 Das Postversandsystem

Mirko Freund beschreibt in seiner Diplomarbeit "Komponenten zur Kooperationsunterstützung: Das Postversandsystem" ([Fre00]) Entwurf und Konstruktion des Postversandsystems, einer Framework-Komponente für JWAM, die es ermöglicht, Materialien von Arbeitsplatz zu Arbeitsplatz zu versenden. Die zu versendenden Materialien müssen hierzu transportierbar sein (technisch bedeutet dies, dass sie die Schnittstellte `TransportableThing` implementieren müssen). Eine zweite Möglichkeit besteht darin, ein Material in eine Transport-Box (einen speziellen Transportbehälter) zu stecken und es darin zu versenden.

Das Postversandsystem wird von Freund in mehrere Komponenten zerlegt. Diese Zerlegung wird von mir in Abschnitt 7.1.2 ab Seite 104 ausführlicher dargestellt vorgestellt.

Freund nennt u.a. folgende Unterschiede des Postversandsystems im Vergleich zu herkömmlicher elektronischer Post:

- Die zu transportierenden Materialien werden als Materialien und nicht als Textnachrichten mit Anhang verschickt, und sie gelangen auch als Materialien zum Empfänger. Es gibt also keine Umwege über Dateien, sondern eine nahtlose Integration in JWAM-basierte Anwendungen
- Das Postversandsystem stellt sicher, dass sich ein zu transportierendes Material zu einem Zeitpunkt nur an einem Ort befindet (Prinzip der Einheit von Ort und Zeit): Während beim Verschicken einer e-Mail mit Attachment letztlich Kopien versendet werden (eine angehängte Datei verbleibt beim Absender unverändert in ihrem Verzeichnis, die e-Mail kann an mehrere Empfänger verschickt werden), wechseln mit dem Postversandsystem verschickte Materialien ihren Ort und verlassen den Arbeitsplatz des Absenders und können nur an einen Empfänger versendet werden.

Mit dem Postversandsystem können verschiedene Benutzungsmodelle realisiert werden: Im einfachsten Fall wird die Arbeitsumgebung des Benutzers nur mit einem Aus- und einem Eingangskorb ausgestattet, die etwa bei Verwendung des JWAM-Desktops auch einfach visualiert werden können. In den Ausgangskorb gelegte, korrekt adressierte Materialien werden in bestimmten Zeitintervallen versendet. Im Eingangskorb landen die für den Benutzer eintreffenden Materialien.

Weitere realisierbare Benutzungsmodelle bieten u.a. die Möglichkeit, eingehende Materialien zu filtern oder mehrere Eingangskörbe zu verwenden. Diese mächtigeren, aber auch komplizierteren Benutzungsmodelle machen dem Benutzer gegenüber die beim einfachen Modell verborgenen Posteingangs- und Postausgangsautomaten explizit, die der Benutzer mit Einstellwerkzeugen konfigurieren kann.

Kooperation: explizit

Kooperationsmedium: Postversandsystem, vergegenständlicht durch Eingangs- und Ausgangs-Postkörbe sowie Posteingangs- und –ausgangsautomat

Kooperationsmittel: Transport-Box

Das Postversandsystem unterstützt explizite Kooperation, denn durch das Auftreten der Benutzer als Empfänger und Absender von transportierbaren Materialien wird deutlich, dass mehrere Benutzer kooperativ in einer gemeinsamen Arbeitsumgebung arbeiten. Das Postversandsystem selbst ist als Kooperationsmedium einzustufen, dass dem Benutzer gegenüber durch die Postein- und –ausgangskörbe sowie — je nach Benutzungsmodell — auch durch Posteingangs- und ausgangsautomat sichtbar wird. Die Transport-Box ist als Kooperationsmittel zu betrachten.

3.1.6 Vorgangsmappen und Laufzettel

In dieser Arbeit beschreibe ich Entwurf und Konstruktion der JWAM-Framework-Komponente *Vorgangsmappen*. Die Vorgangsmappen unterstützen kooperative Arbeitsprozesse, bei denen eine Koordination im Sinne der Producer-Consumer-Abhängigkeit nach Crowston vorliegt (es handelt sich also um arbeitsteilige Prozesse, bei denen spätere Arbeitsschritte auf den Ergebnissen voriger aufbauen).

Die Arbeitsgegenstände werden dabei in der Vorgangsmappe über das Postversandsystem von Bearbeiter zu Bearbeiter verschickt. An der Vorgangsmappe ist ein Laufzettel angebracht, der den vorgesehenen Prozessverlauf (abänderbar) durch eine Liste von Tätigkeiten mit den dafür Verantwortlichen darstellt. Der Laufzettel dokumentiert aber auch den tatsächlichen Prozessverlauf nach jedem Arbeitsschritt ein Stück mehr, weil mit jedem solchen Schritt durch die Bearbeiter Notizen auf dem Laufzettel hinterlassen werden.

Im Gegensatz etwa zu WfMS haben die Benutzer bei Verwendung von Vorgangsmappen die Freiheit, den vorgesehenen Prozessverlauf jederzeit einfach und ad hoc abzuändern.

Die Vorgangsmappen und Laufzettel werden ergänzt durch einen Vorgangsmonitor, der das Suchen und Filtern nach bestimmten Vorgangsmappen, die im System “unterwegs” sind, anhand verschiedener Kriterien erlaubt und über den gegenwärtigen Zustand einer Vorgangsmappe informieren kann.

Kooperation: explizit

Kooperationsmedium: keines (das Kooperationsmedium Postversandsystem wird benutzt)

Kooperationsmittel: Vorgangsmappe und Laufzettel

Vorgangsmappen und Laufzettel unterstützen die explizite Kooperation, denn im Benutzungsmodell wird einerseits deutlich, dass mehrer Benutzer kooperativ zusammenarbeiten, andererseits sind Kooperation und Koordination in Vorgangsmappe und Laufzettel vergegenständlicht. Vorgangsmappe und Laufzettel sind als Kooperationsmittel anzusehen. Die Vorgangsmappen werden über das Kooperationsmedium Postversandsystem verschickt.

3.1.7 Klassifikation nach dem 3K-Modell

Die untenstehende Abbildung zeigt die Einordung der Framework-Komponenten aus den CoJAC-Arbeiten bereitgestellt werden, in das Dreieck des 3K-Modells anhand der durch sie bereitgestellten Groupwarefunktionalitäten (s. Abschnitt 3.1.1 ab Seite 26):

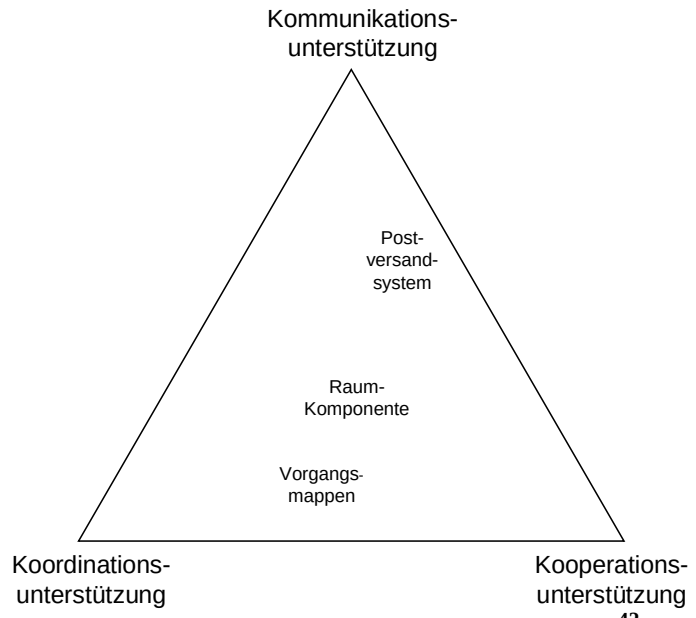


Abbildung 10:Klassifikation der CoJACs[42] nach dem 3K-Modell

Ich beschreibe im Folgenden, warum ich die Positonierung im 3K-Dreieck so vorgenommen habe, wie es Abbildung 10 zeigt.

Bei der Raumkomponente erfährt keine der Unterstützungsfunktionen eine herausgehobene Gewichtung. Am ehesten unterstützt die Komponente durch die räumliche Ordnung die Koordination und durch den Materialaustausch die Kooperation, so dass sie vom Mittelpunkt aus (leicht) in diese Richtungen verschoben ist.

Das Postversandsystem unterstützt einerseits durch die wie bei e-Mail vorhandene Möglichkeit, Textdokumente u.ä. direkt an bestimmte Empfänger zu versenden, die Kommunikation, andererseits hat der explizite Materialaustausch, der von ihm ermöglicht wird, kooperationsunterstützenden Charakter.

Die Vorgangsmappen unterstützen einerseits die Koordination durch den auf dem Laufzettel modellierten Vorgangsverlauf, andererseits wird durch die fortlaufende Vorgangsdokumentation auf dem Laufzettel und die in der Vorgangsmappe befindlichen Arbeitsgegenstände auch die Kooperationsunterstützung geleistet.

[42] Dies ist kein Schreibfehler! Wenn man das Akronym CoJAC ausschreibt, ergibt sich “Klassifikation der Cooperative JWAM Architecture Components (...)”

3.1.8 Klassifikation anhand der Time/Space-Matrix

Die folgende Tabelle zeigt die Einordung der Framework-Komponenten aus den CoJAC-Arbeiten in die Time/Place-Matrix (s. Abschnitt 3.1.1 ab Seite 26):

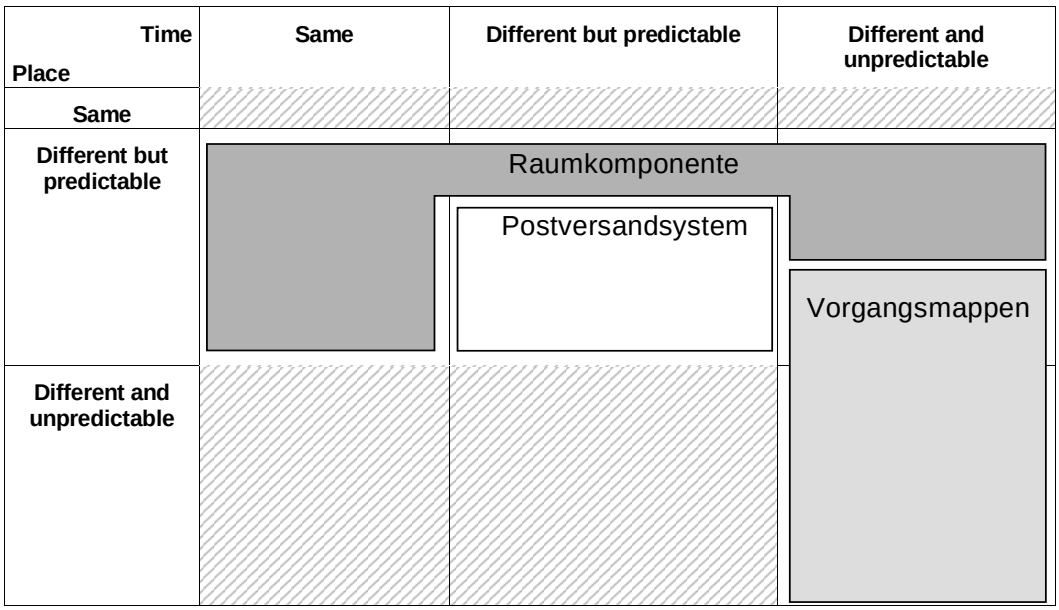


Abbildung 11: Klassifikation der CoJACs anhand der Time/Space-Matrix

Bei der Kooperation mit Hilfe der Raumkomponente befinden sich die Beteiligten an verschiedenen Orten. Ich gehe davon aus, dass die Orte, von dem aus die verschiedenen Teilnehmer Zugang zum virtuellen Gebäude haben, normalerweise bekannt sind, da sich das System eher an kleinere Arbeitsgruppen wendet. Das Arbeiten ist gleichzeitig möglich und führt dann auch zu gegenseitiger Wahrnehmung. Es ist aber auch denkbar, ein Material in einen bestimmten Raum zu legen, damit ein anderer es später bearbeitet. Ob der Zeitpunkt hierbei bekannt bzw. vorhersagbar ist, hängt von den im Arbeitsumfeld gültigen Konventionen ab.

Das Postversandsystem ist nach der Time/Space-Matrix genauso zu klassifizieren wie e-Mail: Ort und Zeitpunkt, zu dem Absender und Empfänger tätig werden, sind verschieden, aber vorhersehbar. Dies trifft mit dem Postversandsystem sogar noch eher zu als bei e-Mail, da beim Postversand-System Mail-Forwarding, WebMail, SMS usw. vermutlich keine Rolle spielen.

Die Bearbeiter von Vorgangsmappen befinden sich grundsätzlich an verschiedenen Orten. Bei der direkten Adressierung einer Vorgangsmappe an einen bestimmten Bearbeiter kann davon ausgegangen werden, dass dem Absender der Ort des Empfängers bekannt ist. Bei einer Zuordnung über Rollen sind dem Absender aber der Empfänger und sein Ort unbekannt. Die Zusammenarbeit verläuft grundsätzlich zeitversetzt, jedoch kann der Absender den Zeitpunkt der Bearbeitung durch den Empfänger in der Regel nicht vorhersagen, da er keine Kenntnis hat, wann der Empfänger dazu kommt, sich die Vorgangsmappe vorzunehmen (Die Bearbeitung einer Vorgangsmappe, d.h. das Ausführen einer entsprechenden, auf dem Laufzettel vorgesehenen Tätigkeit kann sehr viel Zeit beanspruchen).

4 Kooperationsunterstützung durch Prozessmuster

„Ja mach nur einen Plan
Sei nur ein großes Licht.
Und mach dann noch ‘nen zweiten Plan
Gehn tun sie beide nicht.“

Bertolt Brecht, „Dreigroschenoper“

Das Konzept des Prozessmusters und seine Umsetzung durch Vorgangsmappe und Laufzettel stehen im Mittelpunkt dieses Kapitels.

Das Kapitel beginnt mit einer zusammenfassenden Darstellung des Konzeptes, das Gryczan in seiner Dissertation „Prozessmuster zur Unterstützung kooperativer Tätigkeit“ formuliert hat und das für diese Arbeit von grundlegender Bedeutung ist. Das Prozessmuster-Konzept stellt einen grundsätzlichen Gegenentwurf zur der Organisation von Arbeitsprozessen dar, wie sie in Workflow-Management-Systemen vorgegeben wird.

Im Anschluss stelle ich die (physischen) Kooperationsmittel Vorgangsmappe und Laufzettel vor und zeige, dass diese das Prozessmuster-Konzept umsetzen. Mit dieser Erkenntnis verbindet sich für mich die Erwartung, dass die Umsetzung dieser Gegenstände in Software-Kooperationsmittel eine Unterstützung von Arbeitsprozessen ermöglicht, die dem Leitbildes Arbeitsplatzes für eigenverantwortliche Expertentätigkeit gerecht wird.

Zum Abschluss des Kapitels beschreibe ich die beim Umgang mit Vorgangsmappe und Laufzettel auftretenden Anwendungsfälle und modelliere diese Kooperationsmittel mit CRC-Karten. Auf diese Weise halte ich das Grundverständnis für diese Gegenstände auf Basis der bisherigen Überlegungen fest.

Prozessmuster

In diesem Abschnitt soll das Konzept des Prozessmusters motiviert, dargestellt und erläutert werden. Ich möchte die von Gryczan gegebene Definition des Begriffs Prozessmuster voranstellen, um dem Leser vor Augen zu führen, wohin die folgenden Darlegungen führen sollen.

Definition 17: Prozessmuster[43]
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.

Der Kern des Konzepts ist die Vergegenständlichung des Arbeitsprozesses. Damit wird der Arbeitsprozess sichtbar, verstehbar und anpassbar für die beteiligten Benutzer.

[43] aus [Gry96], Seite 178

Dabei kann der zusammengesetzte Begriff „Prozessmuster“. Der Wortbestandteil „Muster” ist hier in doppelter Hinsicht zu verstehen, einmal als Muster im analytischen Sinne: einer als sich regelhaft wiederholenden erkannten Struktur; zum anderen als generatives Muster: als Anleitung oder Vorlage.

Ein Prozessmuster stellt also eine einmal in bestimmten Arbeitzusammenhängen erkannte Regelhaftigkeit, eine typische Ausprägung eines Arbeitsprozesses dar und wird damit gleichzeitig zur Vorlage für neue, anpassbare Ausprägungen.

Gryczan formuliert die Überlegungen, welche schließlich zum Konzept des Prozessmusters führen, vorwiegend in Abgrenzung zur *ablaufsteuernden Sichtweise,* jene Perspektive, die algorithmische Beschreibungen als geeignet ansieht, um sie zur automatisierten Steuerung von Arbeitsprozessen mit menschlicher Beteiligung zu verwenden (siehe Definition 12 auf Seite 32).

Der ablaufsteuernden Sichtweise stellt Gryczan die *unterstützende Sichtweise* entgegen, die dem Leitbild vom Arbeitsplatz für eigenverantwortliche Expertentätigkeit entspricht:

Definition 18:Unterstützende Sichtweise[44]
Die unterstützende Sichtweise versteht Benutzer von Softwaresystemen als Experten ihres Arbeitsgebietes, die in ihrem Arbeitshandeln Computer als Arbeitsmittel einsetzen. Ein charakteristisches Merkmal dieser Betrachtungsweise ist, dass die Initiative bei der Verwendung von Computern vom Benutzer ausgeht und dass die Kontrolle über den Ablauf der Arbeitsschritte beim Benutzer liegt.

Anhand der beiden Sichtweisen beleuchtet Gryczan den Begriff des Planes von zwei Seiten und stellt dem Begriff der vorausgeplanten Handlung den der *situierten Handlung* gegenüber, welcher alltäglichem menschlichen Verhalten besser entspricht. Das Konzept des Prozessmusters als Ergebnis der Überlegungen schliesslich gibt dem situierten Handeln der Benutzer Raum und versteht sich so als Gegenentwurf zur Steuerung von Arbeitsprozessen, wie sie in Workflow-Management-Systemen implementiert ist.

4.1.1 Pläne und Situierte Handlungen

Gryczan nimmt bei seinen Überlegungen zur Planung von Arbeitsprozessen mehrmals Bezug auf das Buch „Plans and situated actions. The problem of human-machine communication“ ([Suc87]) von Lucy Suchman, das die Kommunikation von Mensch und Computer in Abgrenzung zur ablaufsteuernden Sichtweise untersucht.

Suchman untersucht darin auch die Modellvorstellung des Planens, wie sie in der künstlichen Intelligenz und der Kognitionswissenschaft vorherrscht:

„The model treats a plan as something located in the actor’s head, which directs his or her behaviour.“ ([Suc87], S.3)

„On the planning view, plans are prerequisite to and prescribe action, at every level of detail.“ ([Suc87], S. 27)

Demnach ist ein Plan ein isolierbares Gebilde im Kopf eines Handelnden, der sein Verhalten bis ins Detail steuert.

Darauf aufbauend formuliert Gryczan:

[44] dies ist eine Umformulierung des Begriffes 1.4 („Unterstützung“) aus [Gry96] auf Seite 5.

Definition 19: Plan (ablaufsteuernde Sichtweise) ([Gry96], S. 41)
Ein Plan ist eine Verfahrensvorschrift, die das menschliche Handeln vollständig steuert

Suchman stellt in ihrer Arbeit in Frage, ob eine derartige Begrifflichkeit des „Plans“ tragfähig ist für eine Modellvorstellung vom menschlichen Handeln. Sie untersucht hierzu menschliches Handeln und stellt der „geplanten Handlung“ die „situierte Handlung“ gegenüber, die vor dem Hintergrund und angepasst an einen bestimmten materiellen und sozialen Kontext vorgenommen wird und sich eben nicht direkt aus einem Plan ableitet.

Wenn man die „situierte Handlung“ als Regelfall und nicht als Ausnahme menschlicher Handlungen erkennt, trägt dies auch der potenziellen Dynamik von Arbeitssituationen Rechnung: der Möglichkeit etwa, dass sich der Arbeitskontext plötzlich und unvorhergesehen verändert, was notwendig Rückwirkungen auf das Arbeitshandeln haben muss.

Gryczan schärft den Begriff der Handlung durch die Einbeziehung von Erkenntnissen aus der Arbeitspsychologie, die dem Begriff im Wesentlichen Zielgerichtetheit, soziale Eingebundenheit und Gegenständlichkeit („tätige Auseinandersetzung mit der realen Welt“) hinzufügen. Damit kommt Gryczan zu folgender Definition:

Definition 20: situierte Handlung ([Gry96], S. 51)
Eine situierte Handlung ist eine gegenständliche Handlung, die vor dem Hintergrund der persönlichen Erfahrung eines Menschen und einer konkreten Situation durchgeführt wird.

Es ist offensichtlich, dass dies mit dem Begriff “Plan”, wie er aus ablaufsteuernder Sicht verstanden wird, nicht zusammenpasst. Suchman beschreibt das mit einem sehr schönen Bild:

„Just as it would seem absurd to claim that a map in some sense controlled the traveller’s movements through the world, it is wrong to imagine plans as controlling actions.“ ([Suc87], S.188)

Die Karte, die ein Reisender bei sich führt, dient ihm als Hilfsmittel, gibt ihm Orientierung, steuert ihn aber nicht. Vielmehr wird er sie (als situiert Handelnder) bei Veränderungen des Kontextes, bei einem Nicht-Mehr-Zusammenpassen mit der von ihm wahrgenommenen Realität (veränderte Wegverläufe etc.) anpassen, aktualisieren.

Wenn man die Landkarte als Plan begreift (es gibt hier auch wörtlich Bedeutungsnähe, man beachte die Wortbildung „Stadtplan“), dann ist dieser Plan ein Artefakt, das zur Orientierung dient, eine Richtung vorgibt, aber ständig mit der veränderten Situation abgeglichen und angepasst wird. Indem der Plan vom Reisenden gegebenenfalls verändert oder ergänzt wird, vergegenständlicht er auch dessen Ideen, Erfahrungen und Erwartungen.

Suchman betont dementsprechend die Rolle von Plänen als Artefakte, um über Handlungen nachzudenken, lehnt sie als generativen Mechanismus für Handlungen aber ab. Gryczan greift dies auf und bemerkt,

„dass retrospektive Konstruktionen (also: Pläne) Menschen dazu dienen, eine konkrete Situation ‚zu begreifen‘ (‚catch hold of‘)“ ([Gry96], S. 44)

Das führt zu dem Begriff des Planes, der aus der unterstützenden Sichtweise heraus formuliert werden kann:

Definition 21: Plan (unterstützende Sichtweise) ([Gry96], S. 44)
Ein Plan ist ein Hilfsmittel, mit dem Erfahrungen vergegenständlicht werden. Er wird in einer Situation vom handelnden Subjekt als Hilfsmittel verwendet, um punktuelle Handlungsanleitung zu geben.

Die Definition trifft keine Aussage darüber, ob der Plan ursprünglich vom handelnden Subjekt selbst oder von anderer Seite erstellt wurde (die von Suchman als Metapher gewählte Landkarte etwa ist ja heutzutage meist industriell gedruckt).

Der Fall, in dem jemand einen Plan für den eigenen Gebrauch erstellt, ist verhältnismäßig unkompliziert, denn hier tritt die Tätigkeit der Interpretation stark in den Hintergrund. Schwieriger ist es, wenn ein Handelnder einen Plan zu Hilfe nimmt, der von einem anderen erstellt worden ist, denn mit der ablaufsteuernden Sichtweise wird ja auch das einfache „Abarbeiten“ eines Planes verneint. Daraus folgt aber, dass der Handelnde sich den Plan, wie immer er auch formuliert sein mag, aneignen, ihn interpretieren muss.

Die Qualität dieser Interpretation (die natürlich auch von der Gestaltung des Planes abhängt) ist entscheidend ist für die Nutzbarkeit des Planes.

4.1.2 Vergegenständlichung des kooperativen Arbeitsprozesses durch Prozessmuster

Der im vorigen Abschnitt motivierte Begriff des Planes aus unterstützender Sichtweise ist ein Grundpfeiler für das Konzept des Prozessmusters. Ihn wendet Gryczan auf die Fragestellung der Softwareunterstützung für Arbeitsprozesse an.

Als Beispiel für einen kooperativen Arbeitsprozess stellt Gryczan die Kreditvergabe in einer Bank dar ([Gry96], S. 157ff.). In [Zül98], S. 445ff. findet sich eine etwas komprimiertere Darstellung. Diese gebe ich hier wieder:

Kreditvergabe

Kredite werden in einer Bank durch Kundenberater vergeben. Dabei wird zwischen Neu- und Altkunden unterschieden. Über Altkunden können bereits vor einem Beratungsgespräch Informationen eingeholt werden.Welcher Berater das Beratungsgespräch führt, ist bankintern durch eine Kundenzuordnung geregelt.

Befürwortet der Berater den Kreditwunsch des Kunden, füllt er teils während, teils nach dem Gespräch ein Kreditantragsformular aus. Außerdem erstellt er ein Gesprächsprotokoll.
Für die Genehmigung des Kredits wird in der Regel ein weiterer Kundenberater hinzugezogen (Vier-Augen-Prinzip). Grundlage für die Genehmigung sind der Kreditantrag und das Gesprächsprotokoll.

Wenn der Antrag vom zweiten Kundenberater gegengezeichnet wurde, geht er zum ersten zurück. Dieser prüft erneut den Kreditvertrag und verschickt die einzelnenen Kopien an die zuständigen Stellen der Kreditbearbeitung und des Controllings. Die Auszahlung, die der Berater in der Regel bereits vor der Genehmigung veranlasst, erfolgt über ein speziell eingerichtetes Verrechnungskonto, das für den Kunden bis zur endgültigen Genehmigung gesperrt ist. Die Auszahlung kann durch den Kundenberater, vom Schalterangestellten oder von einem Sekretariat durchgeführt werden.

Die folgende Tabelle zeigt die Abhängigkeiten zwischen den einzelnen Tätigkeiten (diese werden durch die Einträge in den Spalten „frühestens“ bzw. „spätestens“ ausgedrückt)

Nr.	Tätigkeit	Frühestens	Spätestens	Muss/kann
1	Kundenzuordnung	—	vor 2	M
2	Gesprächsvorbereitung	Nach 1	vor 3	K
3	Beratungsgespräch	Nach 1	vor 6	M
4	Ausfüllen des Kreditantrags	Während 3	vor 6	M
5	Protokollerstellung	Nach 3	vor 10	K
6	Weiterreichen der Kreditunterlagen	Nach 4	vor 7	M
7	Kreditgenehmigung	Nach 6	vor 8	M
8	Zurückreichen der Kreditunterlagen	Nach 7	vor 10	M
9	Verrechnungskonto einrichten	Nach 1	vor 10	M
10	Kreditauszahlung	Nach 9	vor 11	M
11	Kreditbearbeitung	Nach 8	—	M
12	Kreditkontrolle	Nach 4, 9	—	K

Abbildung 12: Abhängigkeiten bei der Kreditvergabe

Mit dem Prozessmuster soll nun für den typischen Verlauf des Arbeitsprozesses eine Beschreibung gefunden werden, die mit Blick auf das vorige Kapitel einen Plan im Sinne der unterstützenden Sichtweise darstellt (siehe Definition 21 auf Seite 44).

Ich wiederhole an dieser Stelle noch einmal die eingangs vorgezogene Definition des Prozessmusters:

Definition 17: Prozessmuster ([Gry96], S.178)
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.

Das Prozessmuster muss also

- zu erledigende Tätigkeiten beschreiben (*Tätigkeitsbeschreibung)*
- diese Tätigkeiten Personen oder Rollen zuordnen *(Zuordnung)*
- die notwendigen Dokumente benennen *(Dokumente)*
- die Abhängigkeiten zwischen den so beschriebenen Tätigkeiten ausdrücken *(Abhängigkeiten)*

Herausstellen möchte ich, dass das Prozessmuster als Material definiert wird. Damit wird sein Charakter als für den Anwender bearbeitbares Hilfsmittel betont, was der Beschreibung des Planes aus unterstützender Sicht entspricht.

Ein vereinfachtes Prozessmuster für die Kreditvergabe[45] kann in der Übersicht so dargestellt werden:

[45] Die Kreditprüfung nach dem Vier-Augen-Prinzip ist nicht abgebildet.

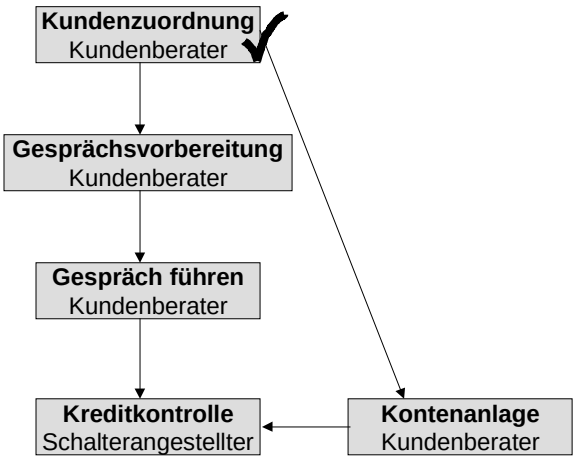


Abbildung 13: Prozessmuster für die Kreditvergabe

Hierbei repräsentiert ein Kästchen eine Tätigkeit (Benennung der Tätigkeit und des Verantwortlichen). Ein Pfeil bedeutet „ist Voraussetzung für". Der Haken an der Kundenzuordnung kennzeichnet sie als erledigt. Die notwendigen Dokumente sind in dieser Übersicht nicht enthalten.

Vorgangsmappe und Laufzettel

4.1.3 Beschreibung von Vorgangsmappe und Laufzettel

Im Folgenden möchte ich die bewährten (physischen) Kooperationsmittel Vorgangsmappe und Laufzettel[46] darstellen und zeigen, dass sie das Konzept des Prozessmusters umsetzen.

Eine *Vorgangsmappe* ist eine Mappe, die im Rahmen eines kooperativen Arbeitsprozesses zur Weiterleitung von Arbeitsmaterialien verwendet wird. Der Name ist geprägt durch die Verwendung solcher Mappen in Verwaltungen, bei denen sich die Arbeitsprozesse an Verwaltungs*vorgängen* entlang vollziehen.

Eine Vorgangsmappe enthält:

- die Materialien, die im Laufe des kooperativen Arbeitsprozesses erstellt und bearbeitet werden
- Materialien, die Informationen zum kooperativen Arbeitsprozess enthalten

An der Vorgangsmappe wird auf der Vorderseite ein *Laufzettel* befestigt. Dieser hat zwei Funktionen:

- durch Aufführen und Beschreiben von Arbeitsschritten und Zuständigkeiten wird auf dem Laufzettel ein Prozessverlauf vorgegeben
- das Eintragen einer Notiz zu einem Arbeitsschritt (durch den Bearbeiter, der diesen Arbeitsschritt durchgeführt hat) dient dazu, die wichtigsten Informationen an hervorgehobener Stelle an den (die) nächsten Bearbeiter weiterzugeben

[46] S. hierzu die Ausführungen von Prinz und Kolvenbach zur Rolle von Vorgangsmappen (*Electronic Circulation Folders)* in Ministerien (in [PK96]). Auf diese Arbeit werde ich in Abschnitt 5.1.2 („Electronic Circulation Folder (Umlaufmappen) bei POLITeam", ab Seite 56) noch einmal Bezug nehmen.

Vorgangsmappen (und ähnliche Kooperationsmittel) finden in Unternehmen und Organisationen Verwendung, die über ein internes Transportwesen (Büroboten) verfügen, das diese als „Hauspost“ befördern kann. Dabei werden die zu befördernden Materialien häufig in standardisierte Umschläge oder Behälter verpackt, auf denen der Adressat (als Person oder Abteilung) gekennzeichnet wird.

Ein Laufzettel für die Kreditvergabe (entsprechend dem Beispiel aus Abschnitt 4.1.2) könnte beispielsweise so aussehen:

Tätigkeit	*Beschreibung*	*Wer*
Kundenzuordnung	Der Kunde wird dem zuständigen Kundenbearbeiter zugeordnet.	Kundenberater
Gesprächsvorbereitung	Sofern es sich um einen Altkunden handelt, müssen die vorhandenen Unterlagen auf für den Kredit relevante Information hin durchgesehen werden. Es werden Kopien der interessierenden Unterlagen in die Vorgangsmappe gelegt.	zust. Kundenberater
Gespräch führen	Ziele des Gesprächs: ▪ Den Kreditwunsch des Kunden klären ▪ Wenn die Kreditvergabe befürwortet werden kann, werden genaue Kreditsumme, Laufzeit und Verzinsung gemeinsam festgelegt und ein Kreditantrag ausgefüllt. Dieser wird in die Vorgangsmappe gelegt. Es ist ein Protokoll anzufertigen und in die Vorgangsmappe zu legen.	zust. Kundenberater
Kreditprüfung	Der Kreditantrag und alle zur Prüfung notwendigen Dokumente werden eingesehen. Kann der Kreditantrag befürwortet werden, ist dies durch Unterschrift zu bestätigen.	Zweiter Kundenberater
Kreditauszahlung	Ein Verrechnungskonto wird eingerichtet. Der von zwei Beratern unterschriebene Antrag wird in Kopie ans Controlling verschickt. Der Betrag wird auf das Verrechnungskonto eingezahlt.	Zust. Kundenberater
Kreditkontrolle	Der Kreditantrag wird anhand der vorliegenden Dokumente einer abschließenden Prüfung unterzogen. Wenn alles in Ordnung ist, wird die Sperrung des Verrechnungskontos aufgehoben.	Back-Office-Angestellter

Erledigte Tätigkeiten

Tätigkeit	*Bemerkungen*	*Wer*	*Datum, Unterschrift*
Kundenzuordnung			
...	...	...	...
Kreditkontrolle			

Abbildung 14: Laufzettel für die Kreditvergabe

Der obere Teil des Laufzettels beschreibt einen typischen Prozessverlauf, der untere Teil dient der Dokumentation des tatsächlichen Verlaufs. Durch Änderungen und Ergänzungen am Laufzettel und die Weiterversendung an andere als die vorgegebenen Personen und Abteilungen kann der Prozessverlauf abgeändert und situativ angepasst werden.

Wie weit entspricht die Einheit aus Vorgangsmappe und Laufzettel nun der Definition des Prozessmusters? Ich greife hierbei auf die Merkmale zurück, die ich bei der Erläuterung der Definition des Prozessmusters herausgearbeitet habe:

- *Tätigkeitsbeschreibung:* Auf dem Laufzettel sind die durchzuführenden Tätigkeiten aufgeführt und beschrieben.
- *Zuordnung:* Die Tätigkeiten werden auf dem Laufzettel Personen oder Abteilungen zugeordnet.

- *Dokumente:* Auf dem Laufzettel werden in den Tätigkeitsbeschreibungen die zur Arbeit benötigten Dokumente benannt, in der Vorgangsmappe sind sie gegebenenfalls enthalten.
- *Abhängigkeiten:* Die Abhängigkeiten zwischen den so beschriebenen Tätigkeiten werden auf dem Laufzettel nur unvollständig durch die vorgegebene Reihenfolge ausgedrückt, die den Normalfall abbildet. Komplexere Abhängigkeiten, wie sie in der Tabelle „Abhängigkeiten bei der Kreditvergabe“ (Abbildung 12 auf Seite 45) oder in der grafischen Darstellung „Prozessmuster für die Kreditvergabe“ (Abbildung 13 auf Seite 45) deutlich werden, können nicht formuliert werden.

Physische Vorgangsmappen und Laufzettel setzen damit das Konzept des Prozessmusters weitgehend um. Es besteht allerdings eine Einschränkung bei der Formulierung komplexerer Abhängigkeiten zwischen den Tätigkeiten.

Damit sind anstatt der axiomatisch anmutenden Definition des Prozessmusters nun konkrete Gegenstände zuhanden, die als Ausgangspunkt einer dem Wort angemessenen „Vergegenständlichung“ dienen können.

4.1.4 Anwendungsfälle

Das untenstehende Use-Case-Diagramm bildet die wichtigsten Anwendungsfälle (Use Cases) ab, die bei Verwendung von Vorgangsmappen und Laufzetteln auftreten:

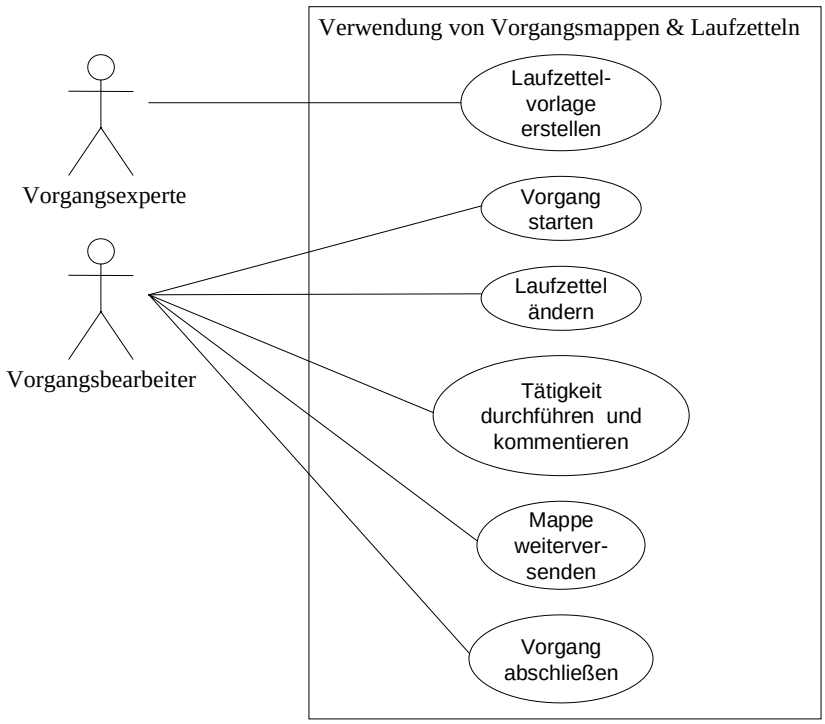


Abbildung 15: Verwenden von Vorgangsmappen und Laufzetteln (Use-Case-Diagramm)

Ich beschreibe im Folgenden diese Anwendungsfälle in kurzer Form:

Eine Laufzettelvorlage beschreibt einen *oder* den *typischen Verlauf eines wiederholt auftretenden kooperativen Arbeitsprozesses*[47]*. Eine solche Vorlage wird von einem Kenner des jeweiligen Vorgangs, dem* Vorgangsexperten*, erstellt und verfügbar gemacht. Ein Vorgangsexperte ist häufig selbst ein* Vorgangsbearbeiter*, also jemand, der im Verlauf des Arbeitsprozesses*

[47] Bei Abbildung 14 auf Seite47 („Laufzettel für die Kreditvergabe“) entspricht dies dem oberen Teil.

bestimmte Teilschritte ausführt (er hat sein Expertenwissen auf diesem Weg erworben). Die folgenden Anwendungsfälle (Use Cases) können von Vorgangsbearbeitern durchgeführt werden:

- Vorgang starten: *Wenn ein Anlass (Auslöser) für einen bestimmten Arbeitsprozess gegeben ist, kopiert ein Mitarbeiter die Laufzettelvorlage und bringt sie an einer Vorgangsmappe an, die er dann mit den passenden Dokumenten und Gegenständen bestückt. Entweder führt dieser Bearbeiter selbst die erste Tätigkeit durch, oder aber er sendet die Vorgangsmappe sofort weiter.*
- Durchführung einer Tätigkeit: *Der Mitarbeiter führt die Handlungen aus, die zum Durchführen der Tätigkeit notwendig sind. Dies schließt häufig das Bearbeiten von Unterlagen ein, die sich in der Vorgangsmappe befinden (möglicherweise auch das Entfernen von Unterlagen aus oder deren Hinzufügen zu der Mappe). Der Mitarbeiter kennzeichnet die Erledigung auf dem Laufzettel und kommentiert sein Tun schriftlich.*
- Mappe weiterversenden: *Nachdem ein Mitarbeiter eine Tätigkeit durchgeführt hat, vermerkt er den Mitarbeiter, der als nächstes tätig werden soll, auf dem Laufzettel und legt die Mappe in seinen Postausgangskorb.*
- Vorgang abschließen: *Nachdem ein Mitarbeiter die letzte Tätigkeit erledigt hat, kann er den Vorgang abschließen. Dazu archiviert er möglicherweise die gesamte Vorgangsmappe in einem oder entnimmt möglicherweise einzelne Dokumente, um sie an anderen Orten aufzubewahren.*
- Laufzettel ändern: *Weicht der Prozessverlauf vom Normalfall ab, muss dies durch Änderungen am Laufzettel deutlich gemacht werden. Diese Entscheidungen werden vom Bearbeiter situativ und vorgangsangemessen getroffen. Entsprechend ändert sich auch der Weg, den die Vorgangsmappe durch die Organisation nehmen wird.*

4.1.5 CRC-Karten

CRC-Karten sind einfache Modellierungsmittel auf dem Wege zu einem objektorientierten Entwurf, die ursprünglich von Kent Beck und Ward Cunningham stammen (vgl. [BC89]). Die Buchstaben CRC sind eine Abkürzung für **C**lass-**R**esponsibility-**C**ollaboration und stehen dafür, dass auf jeweils einer Karteikarte für eine Klasse ihre Verantwortlichkeiten (die Dienstleistungen, die sie erbringt) und diejenigen Klassen, mit denen sie zusammenarbeitet (die sie benutzt, auf die sie sich bezieht) vermerkt werden.

Ich zeige an dieser Stelle drei CRC-Karten, die ich für *Vorgangsmappe, Laufzettel* und *Tätigkeit* entworfen habe und die mein grundlegendes Verständnis der beiden Kooperationsmittel festhalten sollen:

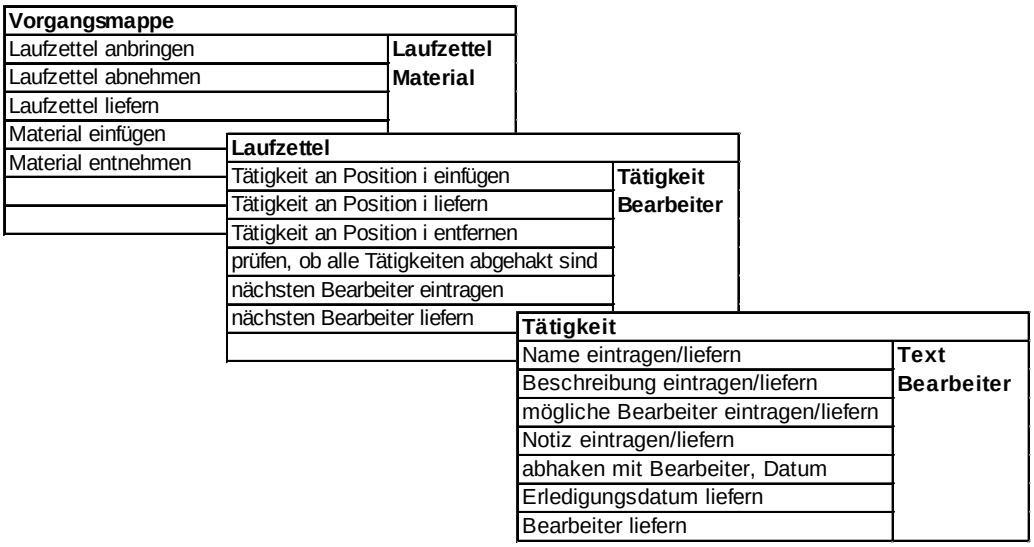


Abbildung 16: CRC-Karten für Vorgangsmappe, Laufzettel, Tätigkeit

Meine Vorstellung ist bei diesen CRC-Karten noch an den physischen Gegenständen orientiert: Ich stelle mir eine Mappe vor, auf deren Deckel der Laufzettel angebracht wird, und die anhand dieses Laufzettels von einem Hauspost-Dienst von Postkorb zu Postkorb gebracht werden.

An der CRC-Karte für die Vorgangsmappe sehe ich als Umgangsformen deshalb vor, dass ein Laufzettel angebracht, wieder entfernt, aber auch während seiner Befestigung am Mappendeckel gelesen und beschrieben werden kann. Außerdem können in die Mappe natürlich Materialien hineingelegt und herausgenommen werden. Die Klasse Vorgangsmappe bezieht sich auf die Klassen Laufzettel und Material (diese Klasse habe ich hier nicht modelliert).

Die CRC-Karte für den Laufzettel beschreibt als Umgangsformen, das Eintragen, Lesen und Entfernen von Tätigkeiten an beliebiger Position auf dem Laufzettel und eine Überprüfung, ob alle Tätigkeiten abgehakt sind. Außerdem kann auf dem Laufzettel der nächste Bearbeiter eingetragen und abgelesen werden — aufgrund dieser Informationen wird die Mappe vom Botendienst befördert. Die Klasse Laufzettel bezieht sich auf die Klassen Tätigkeit und Bearbeiter (diese Klasse habe ich auch nicht modelliert).

Die CRC-Karte für die Tätigkeit[48] schließlich sieht als Umgangsformen das Eintragen und Auslesen eines Tätigkeitsnamens, einer Beschreibung, sowie der möglichen Bearbeiter vor. Es kann auch eine Notiz zu der Tätigkeit eingetragen werden. Bei Durchführung kann die Tätigkeit abgehakt werden, wobei der Bearbeiter und das Erledigungsdatum eingetragen werden. Diese Informationen sind wiederum auch ablesbar. Die Klasse Tätigkeit bezieht sich auf die Klassen Text und Bearbeiter (die ich beide nicht modelliert habe).

[48] Es ist eine schwierige Frage, ob diese Klasse besser “Tätigkeit” oder “Tätigkeitsbeschreibung” heißen sollte. Wenn von dem Eintrag auf dem Laufzettel die Rede ist, erscheint es natürlich zu sagen, dass der Laufzettel “Tätigkeiten in einer Liste aufführt”, die “Tätigkeit wird abgehakt” (nicht die Tätigkeitsbeschreibung). Auf der anderen Seite hört es sich unpassend an, wenn man sagt, man würde eine “Notiz in die Tätigkeit eintragen”. Ich habe mich deshalb einfach für die kürzere Version als Benennung entschieden, benutze aber in dieser Arbeit auch weiter beide Worte, je nachdem, was natürlicher klingt (s.o.)

Zusammenfassung

Ich habe in diesem Kapitel das Konzept des Prozessmusters und die Umsetzung durch Vorgangsmappen und Laufzettel ausführlich dargestellt.

Die wichtigsten Ergebnisse:

- Prozessmuster sind Pläne (im Sinne der unterstützenden Sichtweise) für kooperative Arbeitsprozesse; sie erlauben den Beteiligten, ihre Aufgaben durch situierte Handlungen zu erfüllen.
- Prozessmuster organisieren wie WfMS kooperative Arbeitsprozesse bei deren Ausführung in Teilschritten, jedoch aus der unterstützenden Sichtweise heraus.
- Vorgangsmappen und Laufzettel setzen des Konzept des Prozessmusters um, können jedoch komplex strukturierte Abhängigkeiten bei Arbeitsprozessen nicht formal abbilden.
- Prozessmuster sind besonders gut geeignet für einigermaßen überschaubare, sich häufig ändernde Vorgänge, die von qualifizierten, eigenverantwortlich agierenden Experten bearbeitet werden. Bei diesen ist ein klarer Vorteil für die Prozessmuster gegenüber den recht unflexiblen WfMS zu erkennen. Für wenig variierende, Vorgänge allerdings, die von den Bearbeitern (sei es mangels Qualifikation oder aufgrund der Komplexität des Vorgangs) nicht mehr in ihrer Gänze überblickt werden können, ist der Einsatz von WfMS vermutlich vorzuziehen.

Zum Ende des Kapitels habe ich die Verwendung von Vorgangsmappen und Laufzetteln anhand von Anwendungsfällen verdeutlicht und mein Grundverständnis dieser Kooperationsmittel durch eine Modellierung mit CRC-Karten festgehalten.

5 Entwurf und Implementation von Vorgangsmappe und Laufzettel

In diesem Kapitel beschreibe ich Entwurf und Implementation einer Framework-Komponente für JWAM, die die Kooperationsmittel Vorgangsmappe und Laufzettel und dazu passende Werkzeuge bereitstellt.

Mit den CRC-Karten aus dem vorigen Kapitel sind Vorarbeiten geleistet, doch stellen diese nur eine erste grundlegende Skizze für die zu konstruierenden Materialklassen dar. Um zu einem konkreten Entwurf zu gelangen (der außerdem auch die Werkzeuge einbezieht), betrachte ich zu Beginn dieses Kapitels konkrete Anwendungssysteme, die auf Vorgangsmappen bzw. sogenannten Umlaufmappen basieren. Das Ziel dieser Betrachtung ist es, anhand der Systeme weitere Anforderungen an die Framework-Komponente zu identifizieren und einige der in ihnen verwirklichten Lösungsansätze als Anregung für den Entwurf aufzugreifen. Unter Berücksichtigung einiger zusätzlicher Anforderungen formuliere ich in diesem Sinne grundlegende Entwurfsentscheidungen, die im Weiteren ausgestaltet werden.

Im darauf folgenden Abschnitt beschreibe ich das Benutzungsmodell für die Vorgangsmappen und ihre Verwendung mit den dafür vorgesehenen Werkzeugen. Diese Beschreibung strukturiere ich entlang der Anwendungsfälle, die bereits in Abschnitt 4.1.4 (ab Seite 48) von mir herausgearbeitet worden sind und detailliere dabei den Entwurf.

Eine eingehende Beschreibung der Materialkonstruktion und der Materialschnittstellen rundet meine Darstellung des Entwurfs und der Implementation von Vorgangsmappe und Laufzettel ab.

Am Ende fasse ich den Inhalt dieses Kapitels noch einmal zusammen.

CSCW-Syteme mit Vorgangsmappen

Im Folgenden betrachte ich drei CSCW-Lösungen, die kooperative, arbeitsteilige Arbeitsprozesse ebenfalls durch Mappen unterstützen, in denen die Arbeitsmaterialien unter den Beteiligten ausgetauscht werden. Das Ziel dieser Betrachtung ist es, weitere für Entwurf und Implementation von Vorgangsmappe und Laufzettel relevante Fragestellungen und Lösungsmöglichkeiten zu identifizieren.

5.1.1 Geschäftsfallmappen beim UBS Corporate Desktop

Die UBS (Union Bank of Switzerland) hat mit dem Corporate Desktop ein WAM-basiertes System entwickelt, das dem Benutzer einen Arbeitsplatz bereitstellt, der ihn bei der Bearbeitung von bankspezifischen Geschäftsfällen unterstützt. In Version 2.12 des UBS Corporate Desktop sind dies die folgenden Geschäftsfälle:

- *Annual Review*
- *Kreditantrag*
- *Ratingänderung*

Annual Review bezeichnet den Vorgang der jährlichen Kreditüberprüfung, *Kreditantrag* den Vorgang der Neubeantragung oder der Beantragung einer Änderung eines laufenden Kredites. Eine *Ratingänderung* (Das Rating beschreibt das sogenannte Kreditrisiko) ermöglicht es, einen Kreditnehmer aufgrund ausserordentlicher Ereignisse neu zu „raten“, d.h. in Bezug auf seine „Ausfallwahrscheinlichkeit“ (Expected Default Frequency, EDF) neu zu zu bewerten.

Die drei Geschäftsfallarten und die damit verbundenen kooperativen Arbeitsprozesse werden durch sogenannte Geschäftsfallmappen und passende Werkzeuge unterstützt. Die Geschäftsfallmappen enthalten dabei die Arbeitsmaterialien.

Der voraussichtliche Prozessverlauf **ist nicht als Material vergegenständlicht.** Man geht für die drei Geschäftsfalltypen von einem recht einfachen Modell aus: Es gibt nur drei beteiligte Instanzen, nämlich *Vertriebsmitarbeiter*, das *Bilanzerfassungsteam* und den *Entscheid* (Credit Officer, Kreditmanagement). Bei allen drei o.g. Geschäftsfalltypen wird die Geschäftsfallmappe immer von einem Vertriebsmitarbeiter erstellt. Dieser sendet die Mappe an das Bilanzerfassungsteam, welches die Bilanzen erfasst und die Mappe zurück an den Vertrieb sendet. Je nach fachlicher Beurteilung stellt der Vertriebsmitarbeiter einen Antrag (Kreditantrag, Antrag zur Ratingänderung etc.) und übersendet die Mappe zum Entscheid. Dieser prüft den Antrag und bewilligt ihn oder lehnt ihn ab. Nach Abschluss eines Geschäftsfalles wird er *abgewickelt* und archiviert, so dass er dem *Service Center* als Informationsgrundlage dienen kann.

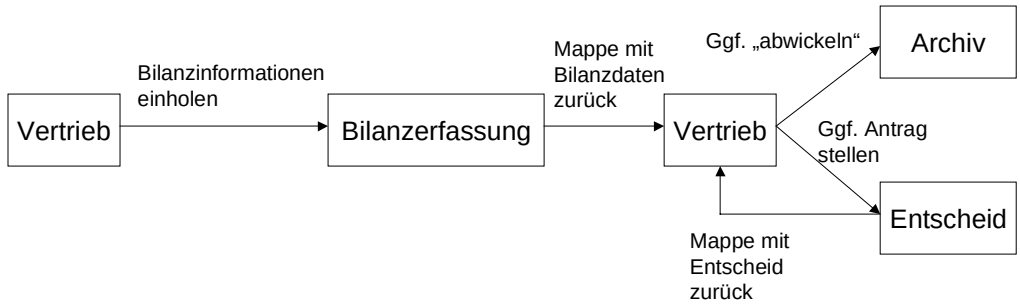


Abbildung 17: Stationen der Gschf.-Mappe beim UBS CorporateDesktop (Grobstruktur)

Diesem in Abbildung 17 dargestellten Grobablauf entsprechend sind für die Geschäftsfallmappen die explizit erkennbaren Status „in Bearbeitung“, „beantragt“, „abgelehnt“ sowie verschiedene Variationen von „bewilligt“ vorgesehen. Der Status einer Mappe kann mit Unterstützung eines Prüfautomaten geändert werden, dessen Funktion ich in Abschnitt 5.1.1.3 ab Seite 55 erläutere.

Es gibt eine gewisse Entsprechung zwischen der Information, die ein solcher Mappenstatus repräsentiert, und der Information, die in einem Laufzettel enthalten ist: Beide spiegeln das Fortschreiten der Bearbeitung wider. Der Laufzettel bezieht sich dabei auf die *Tätigkeiten*, während die Statusinfomation der Geschäftsfallmappe die wichtigsten *Tätigkeitsergebnisse* abstrahiert.

Der Verzicht auf einen Laufzettel (oder ein ähnliches Konstrukt) beim UBS Desktop führt damit im Wesentlichen dazu, dass die zu erledigenden Tätigkeiten nicht detailliert in einem Plan vergegenständlicht werden; der Grobablauf dagegen ist vorgegeben, und unterhalb dieser Ebene ist situiertes Handeln uneingeschränkt möglich. Die Protokollierung der an der Mappe vorgenommenen Änderungen (siehe „Die Geschäftsfallmappe“ weiter unten) dokumentiert den Prozessverlauf in ähnlicher Weise wie der Laufzettel es tut, während der Mappen-Status dem Benutzer Orientierung in Hinblick auf die Ergebnisse des bisherigen Prozessverlaufs gibt.

Durch die Bereitstellung spezieller Werkzeuge für die Bearbeitung der Geschäftsfallmappen der drei Typen für die verschiedenen Benutzerrollen (Vertrieb, Entscheid usw.) und die darin vergegenständlichten Umgangsformen (deren Verfügbarkeit auch vom Inhalt der Mappe und ihrem Status abhängen) findet eine weitere Unterstützung und Orientierung des Benutzers im Prozessverlauf statt.

5.1.1.1 Die Geschäftsfallmappe

Eine Geschäftsfallmappe kann für Kunden angelegt werden, wenn für diesen bereits ein sogenanntes *Dossier* besteht. Beim Anlegen werden Kopien aller zur Bearbeitung des Geschäftsfalls notwendigen Arbeitsgegenstände in die Geschäftsfallmappe gelegt. Für den Typ Annual Review sind dies beispielsweise verschiedene Abschlüsse (Vorjahresabschluss, Berichtsjahrbudget, Budget laufendes Jahr), eine Kopie des bisherigen Ratingblattes usw.

Die Geschäftsfallmappe stellt sich dem Benutzer im „Geschäftsfallmappen-Viewer“ mit zwei Registern dar: dem Inhaltsregister und dem Register Bearbeitungsverlauf.

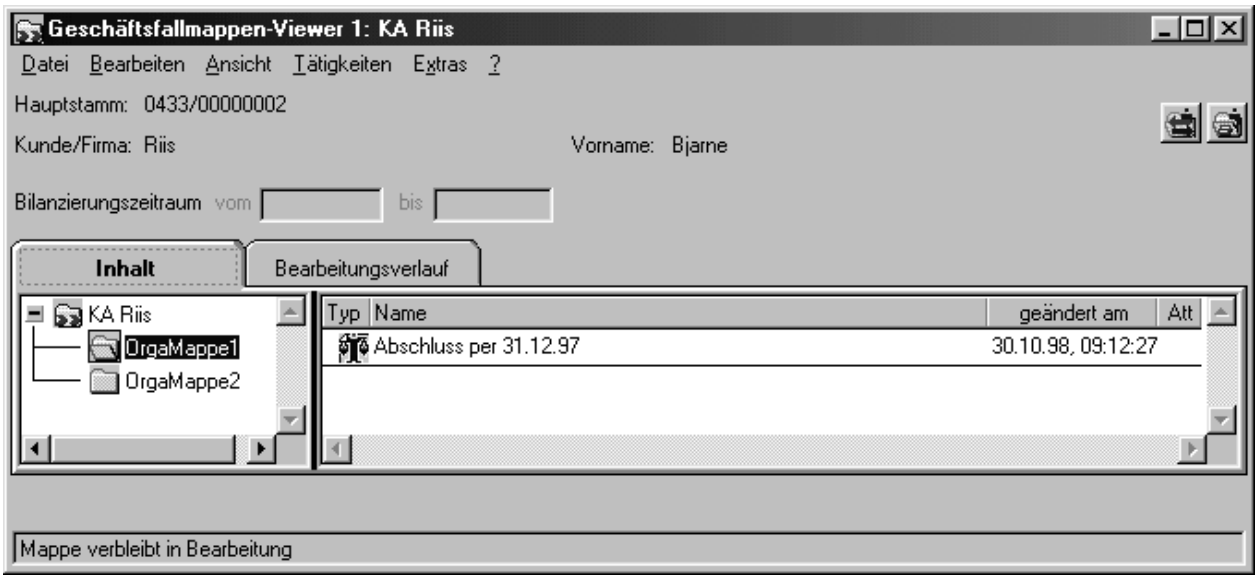


Abbildung 18: Geschäftsfallmappe mit offenem Inhaltsregister

Im Register “Inhalt” werden all jene Dokumente angezeigt, die sich in der Geschäftsfallmappe befinden. Dabei kann der Inhalt in sogenannte „Organisationsmappen“ gegliedert werden, die letztlich eine Einordnung der Dokumente wie in einem Dateibaum erlauben.

Die in der Geschäftsfallmappe enthaltenen Dokumente können ausgehend vom Geschäftsfallmappenviewer geöffnet und bearbeitet werden. Änderungen an den Dokumenten sind lokal und haben keine versteckten Seiteneffekte auf Materialien an anderen Orten des Anwendungssystems.

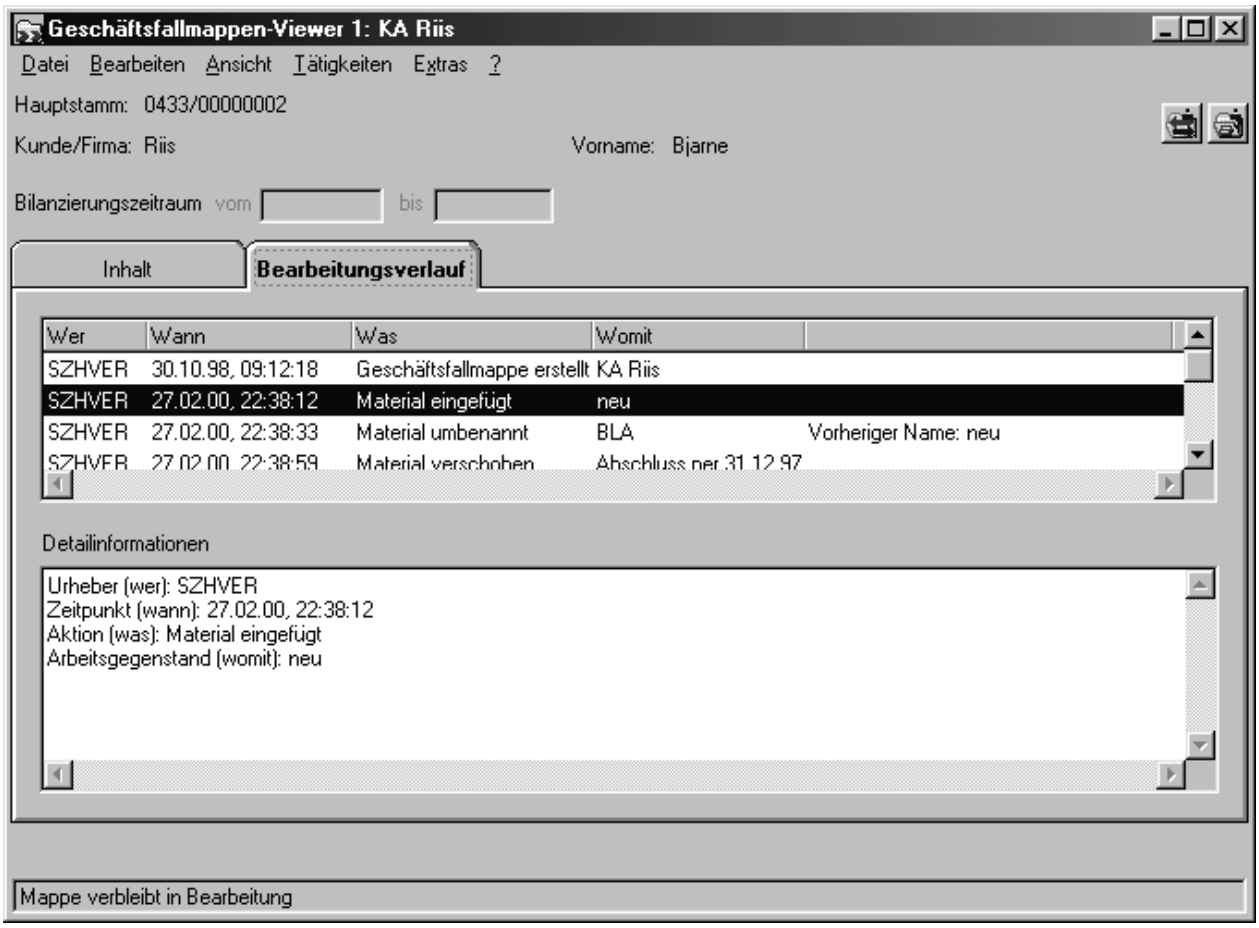


Abbildung 19: Geschäftsfallmappe mit offenem Register „Bearbeitungsverlauf“

Damit die Bearbeitung eines Geschäftsfalls jederzeit nachvollzogen werden kann, werden im Register “Bearbeitungsverlauf” wichtige Ereignisse protokolliert: Anlagen der Geschäftsfallmappe, Statusänderung, Erstellen und Ausdrucken von Protokollen, Versandweg, Prüfungen, etc. Es wird festgehalten, *wer was wann* ausgeführt hat.

5.1.1.2 Weitergabe der Geschäftsfallmappen

Die Geschäfsfallmappen können im bankinternen Postsystem frei (und unabhängig vom Status) versendet werden, mögliche Adressaten sind Einzelpersonen oder Gruppenpostkörbe. Die Benutzer verfügen über ein Postwerkzeug, dass ihnen persönliche und Gruppen-Eingangskörbe sowie eine persönliche Ablage präsentiert.

Der Corporate Desktop verwaltet Rechte (auch Stellvertreter-Rechte) für die Benutzer.

5.1.1.3 Der Prüfautomat

Der Corporate Desktop bietet zur Unterstützung von Vertrieb und Entscheid einen Prüfautomaten an. Dieser wird nie automatisch gestartet, sondern immer explizit durch den Benutzer.

Der Prüfautomat kann die Geschäftsfallmappe auf Vollständigkeit und Konsistenz prüfen. Zum Beispiel wird geprüft, ob für das Betriebsjahr genau ein definitiver Abschluss in der Mappe vorhanden ist, ob die Abschlusszahlen plausibel sind etc. Der Benutzer kann sich die verschiedenen bei der Prüfung angewendeten Kriterien anzeigen lassen. Neben dem Kriterium wird eine kleine Ampel angezeigt, die je nachdem, ob ein Kriterium als erfüllt angesehen wird, als kritisch eingestuft wird, oder aber zweifelsfrei nicht erfüllt werden kann, auf Grün, Gelb oder Rot steht.

Der Prüfautomat wird vom Vertriebsmitarbeiter verwendet, bevor er eine Mappe an den Entscheid weiterleitet. Nur wenn der entsprechende Antrag als vollständig bewertet wird, wird der Mappenstatus vom Prüfautomaten auf *beantragt* gesetzt, und normalerweise sollten nur Mappen mit diesem Status dem Entscheid vorgelegt werden.

Der sogenannte Credit Officer im Entscheid kann den Automaten nach Erfassung seines Entscheides nutzen: Je nach Gesamtergebnis der Prüfung setzt der Automat den Status der Geschäftsfallmappe um.

Für diese Statusänderung (die Bewilligung oder Ablehnung bedeutet) wie auch für die Änderung des Status auf *beantragt* (s.o.) **muss** der Prüfautomaten verwendet werden (vgl. [UBS98], Seite 26).

5.1.1.4 Abwicklung

Ein erfolgreich abgeschlossener Geschäftsfall wird von einem Mitarbeiter des *Service Centers* abgewickelt. Hierbei kommt ein *Abwicklungsautomat* zum Einsatz. Dieser akzeptiert nur erfolgreich bewilligte Geschäftsfälle.

Bei der Abwicklung werden die relevanten Arbeitsgegenstände in ein sogenanntes elektronisches *Dossier* (ein Archiv) transferiert. Dabei wird die Geschäftsfallmappe versiegelt und als Ganzes im Dossier abgelegt. Die Mappe und die in ihr abgelegten Materialien sind danach nicht mehr veränderbar, um die Nachvollziehbarkeit des Geschäftsfalles zu gewährleisten. Die Mappe verschwindet aus der persönlichen Ablage des Mitarbeiters, der sie dem Abwicklungsautomaten übergeben hat.

5.1.2 Electronic Circulation Folder (Umlaufmappen) bei POLITeam

Das Projekt POLITeam ist Teil der POLIKOM-Initiative des Bundesforschungsministeriums. Diese verfolgt den Zweck, vor dem Hintergrund der durch den Umzug der meisten Ministerien nach Berlin aufgeworfenen Probleme („zweigeteilte“ Ministerien, die zum Teil in Berlin, zum Teil in Bonn sitzen, Verbleib wichtiger Behörden in Bonn) ministerielle Abläufe durch Computer und Telekommunikation zu unterstützen. Das POLITeam-Projekt hat hierbei die Aufgabe, eine angemessene elektronische Unterstützung für kooperative ministerielle Arbeitsprozesse bereitzustellen, insbesondere solche, die die Erstellung schriftlicher Dokumente zum Ziel haben.

Im Projektverlauf wurde festgestellt, dass die Mehrzahl der durch POLITeam zu unterstützenden Prozesse zunächst „top-down“ in der Hierarchie herabwandert, dann einen kooperativen Arbeitsprozess auf niedriger Hierarchieebene auslöst und dessen Ergebnisse wiederum „bottom-up“ aufwärts in der Hierarchie durch mehrere Stufen der Zustimmung und Überarbeitung befördert.

Ein exemplarisches (noch nicht elektrifiziertes) Szenario ist die Vorbereitung einer Ministerrede. Hierzu wird eine Umlaufmappe verwendet, die zunächst an die für das Thema der Rede zuständige Abteilung weitergegeben wird. Von Abteilungsleitung und Unterabteilungsleitung mit Anmerkungen und Adressaten versehen, wandert die Mappe schließlich zu den jeweiligen Fachleuten, die in kleinen Gruppen organisiert sind und bei der Erstellung der verschiedenen Abschnitte der Rede zusammenarbeiten. Die Gruppen werden dabei jeweils von Schreibkräften unterstützt, die handgeschriebene oder in ein Diktiergerät gesprochene Texte in Druckschrift zu Papier bringen.

Wenn die getippten Entwürfe vorliegen, wandert die Umlaufmappe in der ministeriellen Hierarchie zurück nach „oben“, wird von den jeweiligen Leitern gelesen, gegebenenfalls mit Hinweisen und Anmerkungen versehen (oder aber zurückgeschickt oder stattdessen nach

telefonischer Rücksprache verändert) und schließlich weitergeleitet. Beim Minister und seinen Beratern angekommen, wird aus den vorhandenen Texten und den daran angebrachten Anmerkungen eine endgültige Version erstellt.

In dem vorgestellten Szenario werden die Umlaufmappen im Allgemeinen durch den internen Botendienst des Ministeriums befördert, während nur in dringenden Ausnahmefällen Sekretäre/Sekretärinnen diese Aufgabe persönlich übernehmen. Gelegentlich werden Dokumente auch gefaxt, jedoch wird stets das Originaldokument nachgeschickt.

Im POLITeam-Projekt wurde entschieden, dass solche Prozesse mit herkömmlichen Workflow-Management-Systemen nicht adäquat unterstützt werden könnten, da diese Systeme in zu starkem Maße verlangen, dass Prozesse im Voraus festgelegt werden und ein flexibles Abändern nicht gestatten. Folgende Anforderung an die Computerunterstützung wurde formuliert:

„As a primary requirement we can identify that a flexible coordination medium is required for the electronic support of ministerial workflows. The flexibility should encompass easy means for the modification of the workflow route and the ability to transport arbitrary documents that can be easily added and removed from the workflow."

Anwender-Interviews brachten die Entwickler schließlich zu der Überzeugung, dass eine elektronische Version der gebräuchlichen Umlaufmappen ein geeignetes Kooperationsmittel ist.

Im POLITeam-Projekt wurden die elektronischen Umlaufmappen auf Basis der Groupware *LinkWorks* implementiert. Dabei sind die Mappen Behälter, die beliebige Objekte enthalten können. Der Prozessverlauf wird auf sogenannten *Circulation Slips* dokumentiert, die auch die Parallelisierung von Arbeitsschritten unterstützen (die genaue Ausgestaltung ist [PK96] und [SPF96] leider nicht zu entnehmen). Die Circulation Slips werden mit den Umlaufmappen gemeinsam verschickt und sind änderbar.

Die Mappen werden mit der übrigen elektronischen Post in *LinkWorks* befördert. Die Groupware wurde dabei soweit angepasst, dass Umlaufmappen in den Post-Werkzeugen sofort als solche erkennbar sind und nicht etwa erst aus einer Mail „ausgepackt" werden müssen.

Zusätzlich wurden einige Besonderheiten der physischen Vorbilder übernommen, so eine farbliche Kennzeichnung der Umlaufmappen, die die Priorität der Mappe ausweist und die Verwendung verschiedener Farben bei der Anbringung von Anmerkungen und Änderungen, gestaffelt nach Hierarchieebenen.

Da die Benutzer beim POLITeam-Projekt zusätzlich auch mit einer Art *Shared Workspace* ausgestattet wurden, bildeten sich bei der Arbeit Mischformen bei der Nutzung dieses Mediums und der Verwendung der Umlaufmappen heraus.

Die Publikationen über das POLITeam-Projekt berichten von einer guten Akzeptanz der eingeführten Computerunterstützung und einer hohen Zufriedenheit der Benutzer.

5.1.3 Vorgangsmappen bei der RWG

Die RWG (Rechenzentrale der Württembergischen Genossenschaften) hat zur Unterstützung kooperativer Arbeitsprozesse im Bankenbereich ein WAM-basiertes Vorgangsmappen-Konzept in ihrem GEBOS-System umgesetzt. Die folgende kurze Beschreibung basiert auf Entwurfsdokumenten der RWG:

5.1.3.1 Vorgang und Vorgangsschrank

Die von der RWG entworfene *Vorgangsmappe* wird als Teil eines Exemplars der Klasse *Vorgang* betrachtet. Dieser kann zusätzlich eine *Checkliste* beinhalten, welche die zu erledigenden Aufgaben aufführt.

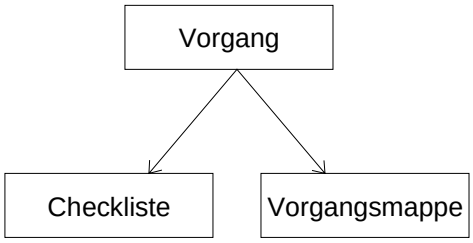


Abbildung 20: Ein Vorgangsobjekt referenziert eine Vorgangsmappe und eine Checkliste

Es wird zwischen *privaten* und *öffentlichen* Vorgängen unterschieden. Private Vorgänge werden ausschließlich von Person zu Person versendet, während öffentliche Vorgänge in einem speziellen Behälter mit dem Namen *Vorgangsschrank* aufbewahrt werden, aber natürlich auch verschickt werden. Wenn öffentliche Vorgänge durch direkten Versand von Bearbeiter zu Bearbeiter gereicht werden, sind sie zu diesem Zeitpunkt aus dem Vorgangssschrank *ausgeliehen*, wobei beim Versand sichergestellt ist, dass die als momentaner Besitzer des Vorgangs verzeichnete Person korrekt geändert wird.

Auf der anderen Seite kann der Vorgangsschrank selbst auch als ein minimaler *Shared Workspace* begriffen werden, der zur Koordination und zum Materialaustausch dient. Er stellt außerdem Informationen über den Status von Vorgängen zum Zeitpunkt des letzten Versands über sogenannte *Ausleih*- und *Infozettel* bereit, was eine Unterstützung der Verfolgung von Vorgängen bedeutet. Dieses Thema wird in Kapitel 6 von mir behandelt (ab Seite 90).

5.1.3.2 Checkliste

Auf der Checkliste sind alle Tätigkeiten eines Vorgangs aufgelistet. Dabei ist zu jeder Tätigkeit deren Status ersichtlich (unerledigt/erledigt/durchgestrichen).

Es können zu jedem Zeitpunkt von jedem Bearbeiter auf der Checkliste Tätigkeiten hinzugefügt werden. Ausserdem können Tätigkeiten (von dem Bearbeiter, der diese Tätigkeit zugefügt hat) auch wieder entfernt und/oder auch gestrichen werden, d.h. sie verbleiben sichtbar auf der Checkliste, können aber nicht mehr bearbeitet werden. Bei der Streichung einer Tätigkeit muss ein Vermerk erfolgen, der über den Grund und Initiator der Streichung Auskunft gibt. Der/die Mitarbeiter kann/können Vermerke auf der Checkliste eintragen.

5.1.3.3 Vorlagenordner

Vorlagen für Vorgänge werden zentral in einem *Vorlagenordner* verwaltet. Auf Basis einer Vorgangsvorlage kann ein konkreter Vorgang erzeugt werden.

Eine Vorgangsvorlage besteht aus:

- einer Vorgangsmappe mit einer Checkliste, die die beim zu unterstützenden Vorgangstyp i.d.R. notwendigen Arbeitsschritte aufführt.

- einem Ordner für Materialvorlagen

Vorgangsmappen werden bei der Erzeugung aus Vorlagen nämlich nicht mit bestimmten Materialien "vorbefüllt". Stattdessen bietet das Vorgangsmappenwerkzeug (s.u.) die im o.g. Materialvorlagenordner enthaltenen Vorlagen an, damit der Anwender zum richtigen Zeitpunkt auf Basis einer dieser Vorlagen ein einzelnes Material neu in die Mappe einfügen kann.

5.1.3.4 Vorgangswerkzeug

Das Vorgangswerkzeug präsentiert einen Vorgang, ähnlich der Darstellung der Geschäftsfallmappe beim UBS Corporate Desktop, durch Registerblätter, hier sind es vier: „Überblick"/„Deckblatt"/„Vorgangsmappe"/„Checkliste".

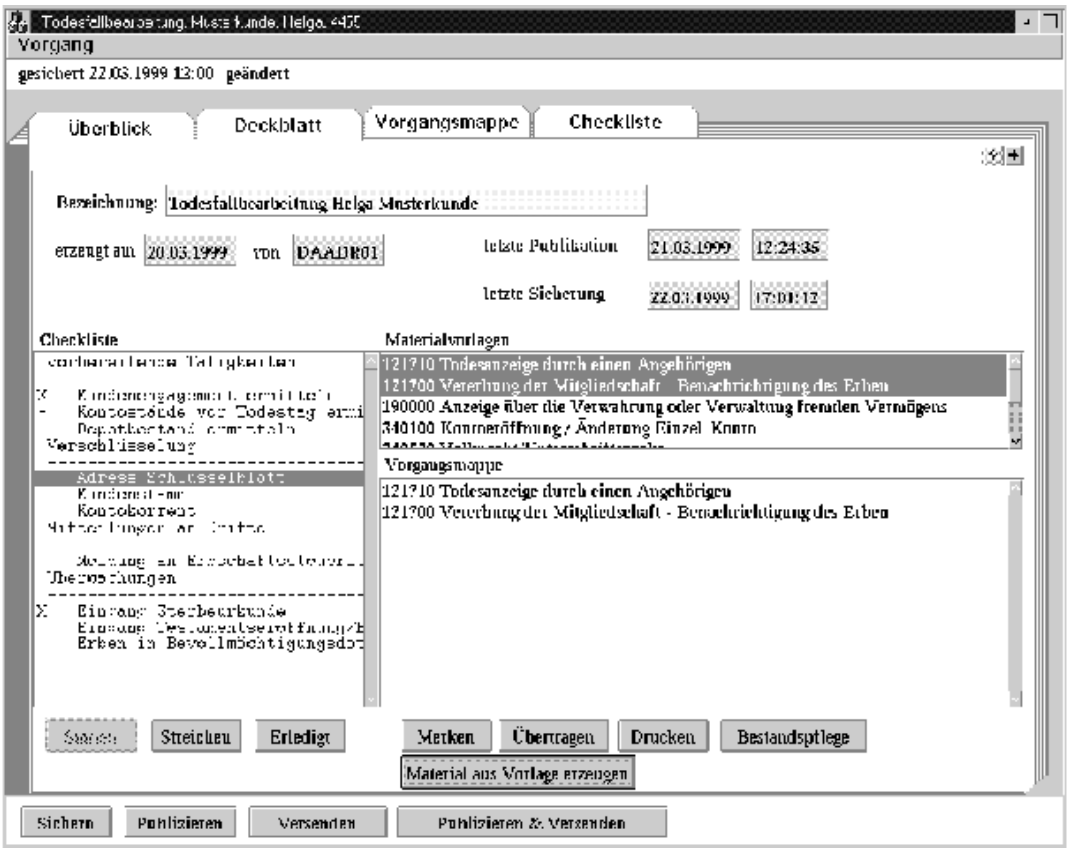


Abbildung 21: Das Vorgangswerkzeug der RWG

Die Deckblatt-Seite enthält dabei Grundinformationen über den Vorgang (Bezeichnung des Vorganges, Bezeichnung der zugrundeliegenden Vorgangsvorlage, Erzeugungsdatum, ...), während die „oben liegende" Überblick-Seite die wichtigsten Informationen aller drei anderen Registerblätter darstellt. Die Überblick-Seite soll dabei für manchen schnellen Blick genügen, so dass ein Aufblättern der dahinter liegenden Seiten überflüssig wird.

Ausgehend von der Registerseite "Vorgangsmappe" können die in der Mappe enthaltenen Materialien bearbeitet werden. Änderungen an den Materialien bleiben im selben Sinne lokal wie ich das in Abschnitt 5.1.1.1 für die Materialien in Geschäftsfallmappen des UBS Desktop beschrieben habe.

5.1.4 Vergleich und abgeleitete Entwurfsentscheidungen

In diesem Abschnitt geht es mir darum, aus den vorangegangenen Betrachtungen die Erkenntnisse zu gewinnen, die die Findung eines geeigneten Entwurfs unterstützen können. Dazu

möchte ich zunächst die Implementationen hinsichtlich einiger Merkmale vergleichen und die Alternativen bewerten.

Die folgende Tabelle bietet eine Übersicht über die wichtigsten vergleichbaren Merkmale, die im vorigen Abschnitt beschrieben wurden:

Eigenschaft	**UBS Corp. Desktop**	**POLITeam**	**RWG**
Entsprechung des Laufzettels	existiert nicht	Circulation Slip	Checkliste, Teil des Vorgangsobjekts
Entsprechung der Vorgangsmappe	Geschäftsfallmappe	Electronic Circulation Folder	Vorgangsmappe, Teil des Vorgangsobjekts
Vorlagen	für die drei vorgegebenen Geschäftsfalltypen	?*	Generisch erstellbare Vorlagen für Mappen inkl. Checkliste
initiale Ausstattung der Mappen mit Materialien	ja, für jeden der drei Mappentypen fest	Nein	Assoziation von Materialvorlagen, Festlegung an der Mappenvorlage
Bezüge von Tätigkeiten zu Mappeninhalt	Wird über den Prüfautomat hergestellt (Mappenstatus)	Nicht formalisiert	Nicht formalisiert
Weitergabe der Mappen	Elektronische Post	Elektronische Post oder *Shared Workspace*	Elektronische Post, Vorgangsschrank

* Dieser Aspekt wird in den Artikeln zu POLITeam leider nicht ausreichend beschrieben, um darüber eine Aussage zu treffen

Abbildung 22: Tabellarischer Vergleich dreier Vorgangsmappen-Umsetzungen

Entlang dieser Merkmale entwickle ich im Folgenden grundlegende Entwurfsentscheidungen für die Konstruktion von Vorgangsmappe und Laufzettel.

5.1.4.1 Laufzettel

Der UBS Corporate Desktop kennt interessanterweise kein Laufzettel-ähnliches Konstrukt. Es kann bei dieser Anwendung darauf verzichtet werden, weil die Versandwege einer Mappe jeweils sehr klar und sehr einfach vorgegeben sind, und weil über den Mappenstatus und den Zwang, diesen nach bestimmten Bearbeitungsschritten durch Verwendung des Prüfautomaten umzusetzen, eine starke Anleitung des Vorgangsverlaufes existiert. Sowohl POLITeam als auch die Umsetzung der RWG haben dagegen eine Entsprechung des Laufzettels (im ersten Fall der Circulation Slip, im zweiten die Checkliste).

Da es das Ziel dieser Arbeit ist, eine Komponente für das JWAM-Framework zu konstruieren, die das Prozessmuster-Konzept umsetzt, liegt es auf der Hand, dass ich mich in diesem Punkt nicht an der (für den Spezialfall angepassten) Umsetzung, wie sie beim UBS Corporate Desktop zu finden ist, orientieren kann. Das Prozessmuster-Konzept soll durch Vorgangsmappe *und* Laufzettel umgesetzt werden.

Bei der Checkliste der RWG gefällt mir die Idee sehr gut, Tätigkeiten nicht zu löschen, sondern zu streichen: Die gestrichenen Tätigkeiten werden als durchgestrichen in der Checkliste angezeigt, so dass diese Abänderung mit einem Blick erkennbar ist.

Die Idee, mindestens die potenzielle Parallelität im Prozessmuster auszudrücken, wie es POLITeams Circulation Slips ermöglichen, findet sich auch in der Arbeit von Guido Gryczan, die das Prozessmuster-Konzept vorstellt (vgl. [Gry96], ab S. 198 – oder auch die von Gryczan zitierte Arbeit [Wul95]). Dort sind sogenannte Tätigkeitsnetze, die grafisch editiert werden können, Grundlage der Prozessmuster.

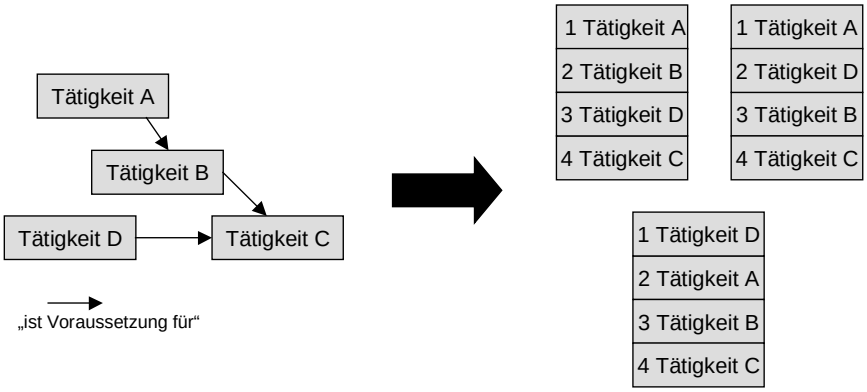


Abbildung 23: Ein Tätigkeitsnetz kann auf mehrer Sequenzialisierungen abgebildet werden

Abbildung 23 zeigt die grundsätzliche Struktur eines Tätigkeitsnetzes, an dem man für jede Tätigkeit ablesen kann, welche anderen Tätigkeiten vor ihr durchgeführt werden müssen. Daraus lässt sich nicht nur erkennen, welche Tätigkeiten parallelisierbar sind, sondern auch, welche Reihenfolgen bei einer Sequenzialisierung der Tätigkeiten zulässig sind.

Ein Tätigkeitsnetz ist eine adäquate Modellierung der realen Abhängigkeiten zwischen den einzelnen Tätigkeiten innerhalb eines arbeitsteiligen, kooperativen Arbeitsprozesses, wie man selbst in der (vergleichsweise übersichtlichen) Tabelle „Abhängigkeiten bei der Kreditvergabe“ (Abbildung 12 auf Seite 45) erkennen kann: Die dokumentierten Abhängigkeiten zwischen den Tätigkeiten bilden eine netzartige Struktur zwischen diesen aus, was sich auch in der vereinfachten Visualisierung auf Seite 46 (Abbildung 13, „Prozessmuster für die Kreditvergabe“) niederschlägt.

Es macht allerdings vielen Anwendern Schwierigkeiten, mit Netzeditoren u. ä. umzugehen, weil dies ein Verständnis für abstrakte mathematische Modelle voraussetzt. Diese Tatsache ist ein gewichtiges Argument gegen die Präsentation und Editierung von Prozessmustern als Netzen, denn der Kerngedanke des Konzeptes ist es ja gerade, die Prozessbeschreibung zu einem durch den Anwender bearbeit- und verstehbaren Material zu machen.

Aus diesem Grund entscheide ich mich gegen die Verwendung eines Tätigkeitsnetzes und für die sequenzielle Form des Laufzettels, wie sie bereits bei den CRC-Karten in Kapitel 4 modelliert ist: Der Laufzettel enthält die Tätigkeiten in einer Reihenfolge, die einen Standardfall darstellt. Streichungen sind (bei nichtabgeschlossenen Tätigkeiten) für den Anwender jederzeit situativ möglich. Der bereits dort genannte Nachteil, dass der Laufzettel dann komplexe Abhängigkeiten zwischen den Tätigkeiten nicht modellieren kann, ist hinnehmbar, weil ich voraussetze, dass die Benutzer qualifizierte Experten ihres Aufgabengebiets sind[49], die die Arbeitsvorgänge, an denen sie beteiligt sind, durchschauen.

[49] entsprechend dem WAM-Leitbild des Arbeitsplatzes für eigenverantwortliche Expertentätigkeit

Entwurfsentscheidungen

- (Es wird eine Laufzettelklasse bereitgestellt)
- Der Laufzettel enthält eine Liste von Tätigkeiten, deren Reihenfolge eine Standardreihenfolge der Erledigung angibt
- Auf dem Laufzettel können Tätigkeiten durchgestrichen werden

5.1.4.2 Vorgangsmappe

Alle drei betrachteten Systeme besitzen eine Entsprechung für die Vorgangsmappe, wie sie im letzten Kapitel vorgestellt wurde. Die Electronic Circulation Folder von POLITeam können einen Circulation Slip enthalten, während bei dem Entwurf der RWG ein Vorgangsobjekt existiert, das die Vorgangsmappe und die Checkliste aggregiert. Beim UBS Corporate Desktop existiert kein Laufzettel-ähnlicher Gegenstand.

Beim Entwurf einer Materialklasse für die Vorgangsmappe gehe ich zum physischen Vorbild zurück, das Ausgangspunkt der Modellierung ist: Bei einer physischen Vorgangsmappe handelt es sich um eine Mappe, die einen Laufzettel enthält oder auf der ein solcher angeheftet ist, und in der verschiedene Arbeitsmaterialien von Bearbeiter zu Bearbeiter geschickt werden. Aus meiner Sicht spricht nichts dagegen, diese Konfiguration auch so in den Entwurf zu übernehmen: Auch die Vorgangsmappe als Software-Material stellt für mich einfach einen Behälter für Materialien dar, an dem außerdem ein Laufzettel angebracht werden kann und der für die Versendung geeignet ist.

Dabei abstrahiere ich bewusst von der inneren Struktur der Mappe (gibt es Registerseiten, sind die enthaltenen Dokumente geordnet usw.), weil ich alle dort denkbaren sinnvollen Ausprägungen als Vorgangsmappen ansehe. Ich sehe auch keine Notwendigkeit, ein zusätzliches Vorgangsobjekt wie bei der RWG einzuführen.

Dennoch oder gerade deswegen spreche ich aber auch weiter von *dem Vorgang*, wenn ich den kooperativen Arbeitsprozess meine.

Entwurfsentscheidungen

Die Vorgangsmappe ist charakterisiert durch folgende Eigenschaften:

- Sie ist ein Behälter für Materialien
- An ihr kann ein Laufzettel angebracht werden
- Sie ist versendbar

5.1.4.3 Vorlagen für Vorgangsmappen und Laufzettel

Wenn ein bestimmter kooperativer Arbeitsprozess, genauer: ein Exemplar eines bestimmten Typs von Prozess, gestartet werden soll, ist es für den Benutzer von Vorteil, wenn er schnell und unkompliziert eine entsprechende vorkonfigurierte Vorgangsmappe erzeugen kann. Dazu muss ihm das Anwendungssystem eine Auswahl solcher spezialisierter Mappen anbieten können.

Ein Weg, um dies zu ermöglichen, ist die Bereitstellung verschiedener Mappen- und Laufzetteltypen für eine *feste Auswahl* an möglichen Prozesstypen, wie es der UBS Corporate Desktop für die drei Typen Annual Review, Kreditantrag und Ratingänderung tut (wie bereits mehrfach angemerkt, gibt es beim Corporate Desktop allerdings keine Laufzettel).

Eine andere Möglichkeit ist, dem Anwender (oder bestimmten Anwendern) die Möglichkeit zu geben, Mappen- bzw. Laufzettelvorlagen für bestimmte Prozesstypen interaktiv (also zur Laufzeit) zu erstellen und diese zentral zu verwalten. Diesen Ansatz verfolgt die Implementation der RWG.

Bei der Entwicklung der Framework-Komponente für Vorgangsmappen möchte ich beide Ansätze unterstützen.

Für die Umsetzung der zweiten Variante ist die Modellierung generischer Vorgangsmappen- und Laufzettelvorlagen notwendig. Dabei möchte ich die Mappen und Laufzettel selbst zu Vorlagen für neue Vorlagen machen, also das häufig gebrauchte Prototyp-Muster verwenden. Die zur Laufzeit entworfenen Vorlagen müssen in einem zentralen Vorlagenordner abgelegt werden können.

Derselbe Mechanismus kann vom Anwendungsentwickler auch zur Konstruktionszeit zur Bereitstellung einer festen Auswahl von Mappen verwendet werden, er kann für diesen Zweck aber natürlich auch die Framework-Klassen für Vorgangsmappe und Laufzettel spezialisieren[50].

Entwurfsentscheidungen

- Die Benutzer können generische Vorgangsmappen- und Laufzettelvorlagen erstellen
- Die Vorgangsmappen und Laufzettel stellen selbst Vorlagen dar (Prototypen)
- Die Vorlagen können in einem zentralen Vorlagenordner abgelegt werden

5.1.4.4 Initiale Ausstattung der Mappe mit Materialien

Wenn beim UBS Corporate Desktop eine Geschäftsfallmappe neu erzeugt wird, enthält sie bereits kundenspezifische Materialien (passend für den jeweiligen Geschäftsfalltyp), die auf einem Kundendossier beruhen. Auch die spätere Erzeugung und Einfügung weiterer Materialien ist zum Teil durch das Vorgangswerzkzeug direkt unterstützt (hier kann z.B. über einen Menüeintrag ein neues Ratingblatt eingefügt werden).

Wenn dagegen beim Entwurf der RWG auf Basis einer Vorgangsvorlage ein neuer Vorgang erzeugt wird, sind ihm nur Material*vorlagen* assoziiert, die bei Bedarf zum Einfügen neuer Materialien verwendet werden.

Die Liste der assoziierten Materialvorlagen „(...) stellt eine Art Materialvorschlag dar; eine unverbindliche Zusammenstellung aller Materialien, die im Kontext des Vorgangs sinnvoll sind oder sein könnten. Der Berater kann sich so die benötigten Materialien schnell zusammenstellen, indem er die betreffenden Vorlagen wählt und die konkreten Materialien einfach erzeugt“ (zitiert aus der „Materialvision Vorgang“ der RWG).

Die so eingefügten Materialien enthalten direkt nach Erzeugung natürlich noch keine kundenspezifischen Eintragungen.

Der Unterschied der beiden Systeme ist darin begründet, dass der UBS Corporate Desktop nur drei Vorgangstypen unterstützt und die Geschäftsfallmappen und Werkzeuge darauf zugeschnitten sind, während die Vorgangsmappen der RWG für eine größere Vielfalt von Vorgängen konzipiert sind und deshalb generischer gestaltet wurden.

Die Vorgangsmappen, die ich in einer Framework-Komponente bereitstellen möchte, sollen ebenfalls vielfältige Vorgangstypen unterstützen können, weshalb ich das Konzept der RWG,

[50] In Abschnitt 5.1.11.2 ab Seite 81 stelle ich mit der Materialklasse `ProcessFolderAdapter` eine Klasse vor, die die Erstellung spezialisierter Vorgangsmappen erleichtert.

einer Vorgangsmappen-Vorlage eine wohlbestimmte Menge von Materialvorlagen zu assoziieren, gern aufgreifen würde. Da sich dies jedoch für die Implementation als relativ aufwändig herausgestellt hat[51], habe ich mich für eine simplere Form der Verwendung von Materialvorlagen als "Basisversion" entschieden und möchte die RWG-Version als noch nicht implementierte Ausbaustufe ansehen.

Die simple Variante sieht so aus, dass bei der Verwendung von Vorgangsmappen als Vorlagen die in ihnen enthaltenen Materialien in die neu erzeugte Mappe kopiert werden. Es handelt sich um *fachliche* Kopien, d.h. dass die fachliche Identität der kopierten Materialien eine andere ist als die der Vorlagen.

Um es dem Anwender aber ähnlich wie beim UBS Corporate Desktop zu ermöglichen, Vorgangsmappen eines bestimmten Typs erzeugen zu können, bei denen z. B. Formulare mit Daten eines zuvor ausgewählten Kunden *vorausgefüllt* in jede neu erzeugte Mappe eingefügt werden, muss der Anwendungsentwickler eine entsprechende Werkzeugunterstützung selbst implementieren.

Entwurfsentscheidungen

- Bei Verwendung einer Vorgangsmappe als Vorlage werden fachliche Kopien der in ihr enthaltenen Materialien in die neu erzeugte Vorgangsmappe eingefügt.
- Einer Vorgangsmappen-Vorlage kann eine Menge von Materialvorlagen *assoziiert* werden (Ausbaustufe).

5.1.4.5 Bezüge der Tätigkeiten zum Mappeninhalt

Beim Corporate Desktop der UBS gibt es keine Entsprechung des Laufzettels, die von den Bearbeitern durchzuführenden Tätigkeiten werden jedoch durch die Verwendung des Prüfautomaten mit dem Mappeninhalt verknüpft. Hierbei spielt der Status der Geschäftsfallmappen die entscheidende Rolle (entweder „in Bearbeitung", „beantragt", „abgelehnt" oder „bewilligt").

Dagegen sind bei POLITeam und der Implementation der RWG die Tätigkeiten auf dem Circulation Slip bzw. der Checkliste explizit aufgeführt, es besteht aber keine formale Verknüpfung zwischen diesen Tätigkeiten und dem Mappeninhalt, nur in den textuellen Beschreibungen der Tätigkeiten gibt es Bezugnahmen auf den Mappeninhalt. Dem physischen Vorbild folgend, steht auch für mich nur die textuelle Beschreibung einer Tätigkeit auf dem Laufzettel im Vordergrund.

Den beim UBS Corporate Desktop an der Mappe modellierten Status für die Framework-Komponente aufzugreifen, erscheint mir nicht sinnvoll: Der Status hat bei den Geschäftsfallmappen deshalb ein so großes Gewicht, weil er nur durch den Einsatz des

---

[51] Es hat sich herausgestellt, dass es zum gegenwärtigen Zeitpunkt sehr schwierig ist, in JWAM Materialien, welche sich in der Umgebung (auf dem Desktop, in Behältern usw.) befinden, *anders als mit direkter Manipulation* auszuwählen, in einen anderen Behälter einzufügen usw. Alle im JWAM-Framework enthaltenen Werkzeuge basieren, was diese Aktionen angeht, ausschließlich auf direkter Manipulation (Drag'n Drop).

Da im WAM-Ansatz nicht mit Dateien, sondern mit Gegenständen umgegegangen wird, verbietet sich der Einsatz von Dateiauswahl-Dialogen, wie sie vom *Java Development Kit* standardmäßig angeboten werden. Andererseits stehen entsprechende "Material-Auswahl"-Dialoge in JWAM noch nicht zur Verfügung (sind zwar geplant, aber noch nicht fertig). Aus diesem Grund ist die Realisierung des Aktes, mit dem Materialvorlagen, die einer Mappenvorlage assoziiert werden sollen, *ausgewählt* werden können, ein echtes Problem gewesen, dessen Lösung für diese Arbeit unverhältnismäßig viel Aufwand bedeutet hätte, weil sie thematisch nicht in ihrem Zentrum steht.

Prüfautomaten geändert werden kann. Dieser Automat führt umfangreiche fachliche Überprüfungen am Inhalt der Mappe durch, die in einer Framework-Komponente nicht einmal abstrakt implementiert werden können. Einfach nur ein Attribut für eine Statusinformation an der Vorgangsmappenklasse vorzusehen, stellt aber keine wirkliche Unterstützung für den Anwendungsentwickler dar.

Entwurfsentscheidung

Die Tätigkeitsbeschreibungen nehmen nur textuell Bezug auf Materialien in der Vorgangsmappe.

5.1.4.6 Weitergabe der Vorgangsmappen

Bei allen drei betrachteten Systemen werden die Vorgangsmappen vorrangig über elektronische Post an den nächsten Bearbeiter weitergegeben. Bei POLITeam geschieht dies darüber hinaus auch über den *Shared Workspace*, einen gemeinsamen Informationsraum, bei der Implementation der RWG gibt es den Vorgangsschrank, über den ebenfalls Vorgangsmappen weitergegeben werden könnten.

Der Laufzettel führt, wenn ich auf die erste Skizzierung dieser Klasse durch CRC-Karten in Kapitel 4 zurückgreife, Tätigkeiten auf, denen mögliche Bearbeiter (oder Rollen) zugeordnet sind. Statt eine freie Festlegung des nächsten Empfängers am Laufzettel zu erlauben, wie es bei den CRC-Karten modelliert wurde (und damit die auf dem Laufzettel beschriebenen Tätigkeiten und Zuständigkeiten nicht mit der Adressierung zu verknüpfen), löse ich mich an dieser Stelle vom physischen Vorbild: Die Adressierung soll erfolgen, indem der Bearbeiter die nächste durchzuführende Tätigkeit bestimmt und aus den ihr zugeordneten, möglichen Bearbeitern/Rollen eine(n) auswählt[52].

Damit erhält die Adressierung der Vorgangsmappe eine neue Qualität, denn sie ist nicht mehr losgelöst vom eigentlichen Prozessmuster, sondern geht direkt aus der Planungstätigkeit des Anwenders hervor.

Dies macht für mich das Postversandsystem zu dem “offiziellen” Weg der Weitergabe der Vorgangsmappen, weil es die so festgelegte Adressierung interpretieren und in einen Versendevorgang umsetzen kann.

Natürlich ist es auch denkbar, eine Vorgangsmappe dem nächsten Bearbeiter etwa über die Raumkomponente (s. Abschnitt 3.1.4 ab Seite 36) weiterzugeben (durch Ablage in einem vereinbarten Raum) — eine solche Handlungsweise löst sich aber von der Unterstützung des Prozessmusters und verlangt statt dessen neue Konventionen.

Entwurfsentscheidungen

- Die Weitergabe der Vorgangsmappen soll in der Regel über das Postversandsystem erfolgen
- Die Adressierung der Vorgangsmappen wird mit der Tätigkeitsliste des Laufzettels verknüpft

Weitere Anforderungen

Ich formuliere an dieser Stelle weitere Anforderungen, die sich nicht direkt aus den vorangegangenen Betrachtungen der CSCW-Systeme ableiten lassen.

[52] Ich gehe davon aus, dass bei Verwendung von Rollen die Rollenauflösung selbstverständlich nicht vom Anwender vorgenommen wird, sondern über einen automatisierten Mechanismus (s. Abschnitt 5.1.5 ab Seite 66)

5.1.5 Rollen und Benutzergruppen

Es wurde in dieser Arbeit bisher nicht diskutiert, ob und wie weit die Modellierung von Organisationsstrukturen in der Framework-Komponente für Vorgangsmappen vorgenommen werden soll — in Workflow-Management-Systemen etwa ist dies ein Standard-Bestandteil (siehe Abschnitt 3.1.2 bzw. die darin enthaltene Abbildung der "Generic Workflow Product Structure" auf Seite 33).

Bei der Formulierung von Zuständigkeiten auf dem Laufzettel werden funktionelle Rollen[53] verwendet, denn in der Regel wird eine Tätigkeit ja nicht einer bestimmten Person, sondern eben einer Rolle wie „Kundenberater" zugeordnet. Für das Versenden einer Vorgangsmappe mit dem Postversandsystem bedeutet dies, dass entweder die ausgewählte funktionelle Rolle auf die Transportadresse *eines* bestimmten Benutzers abgebildet werden muss, oder aber auf eine Transportadresse, die zu einem Gruppenpostkorb gehört[54]. Im ersten Fall wird unter mehreren Personen, die eine bestimmte funktionelle Rolle ausfüllen, eine durch die Anwendung ausgewählt, im zweiten Fall regelt die Gruppe, die der funktionellen Rolle entspricht[55] (und der der Gruppenpostkorb gehört), selbst, wer die Vorgangsmappe bearbeiten soll.

Es wird anhand dieser Ausführungen schnell deutlich, dass es die flexible Verwendbarkeit der Vorgangsmappen sehr einschränken würde, wenn man mit ihnen ein bestimmtes Organisationsmodell vorgeben würde – beispielsweise Annahmen macht, ob es im System Gruppenpostkörbe gibt etc. Es ist auch zu berücksichtigen, dass Benutzergruppen und funktionelle Rollen häufig mit Zugriffs- und Ausführungsrechten in Softwaresystemen verkoppelt werden – einem Thema, das ich nicht mit meinem vermengen möchte.

Deshalb möchte ich diejenigen Funktionen anpassbar gestalten, die eine Auswahl von funktionellen Rollen und Benutzern anbieten (um diese einer Tätigkeit zuzuordnen) und die einen Benutzer/eine Rolle auf Transportadressen abbilden.

Wenn dies umgesetzt wird, können Anwendungen, die in unterschiedlichster Weise Organisationsstrukturen modellieren, die Vorgangsmappen verwenden.

Entwurfsentscheidung

Die Framework-Komponente für Vorgangsmappen macht keine Annahmen über die Organisationsmodellierung und ist in dieser Hinsicht für den Anwendungsentwickler anpassbar

5.1.6 Protokollierung, Vergleich

Der Prozessverlauf und die Änderungen am Laufzettel sollen so transparent und nachvollziehbar wie möglich dokumentiert werden. Mit der Möglichkeit, Tätigkeiten klar erkennbar durchzustreichen (statt zu löschen), ist bereits ein Anfang gemacht. Dies muss dadurch ergänzt werden, dass sowohl beim Durchstreichen wie beim Abhaken einer Tätigkeit der Zeitpunkt und der handelnde Benutzer festgehalten werden.

---

[53] In [Zül98] ist folgende Definition für eine funktionelle Rolle zu finden:

„Eine funktionelle Rolle ist [...] eine Sammlung von Aufgaben und Verantwortlichkeiten, die einem prototypischen Arbeitsplatz und einer Rolle zugeordnet sind. Diese Rolle kann dann von einer oder mehreren Personen ‚besetzt' werden. Die Zuordnung kann zeitlich wechselnd, dauerhaft, informell oder in der Unternehmenshierarchie festgeschrieben sein. [...]"

[54] Ein Gruppenpostkorb ist ein Postkorb, auf den eine Gruppe von Benutzern Zugriff hat.

[55] Das soll bedeuten: die Gruppe die sich aus allen Personen zusammensetzt, die die funktionelle Rolle ausfüllen.

Um eine Verfälschung einmal gemachter Eintragungen zu verhindern, dürfen einmal abgehakte oder durchgestrichene Tätigkeiten nicht wieder zurückgesetzt werden können (Wenn das Abhaken oder Durchstreichen fehlerhaft war, muss dem Laufzettel ggf. eine neue Tätigkeitsbeschreibung eingefügt werden, die dies richtigstellt).

Weitergehende Vorschläge wären eine Protokollierung aller Änderungen (z. B. in textueller Form) oder ein Werkzeug, dass Unterschiede zwischen einem Laufzettel und seiner Vorlage visualisiert.

Entwurfsentscheidungen

- Beim Abhaken oder Durchstreichen einer Tätigkeit werden Zeitpunkt und handelnder Benutzer festgehalten
- Das Abhaken oder Durchstreichen einer Tätigkeit kann (nach Bestätigung) nicht wieder rückgängig gemacht werden

Zusammenfassung der bisherigen Entwurfsentscheidungen

In den vorigen Abschnitten habe einige Entwurfsentscheidungen formuliert und begründet, die ich an dieser Stelle noch einmal zusammenfassen möchte:

Laufzettel

- Der Laufzettel enthält eine Liste von Tätigkeiten, deren Reihenfolge eine Standardreihenfolge der Erledigung angibt
- Die Tätigkeitsbeschreibungen nehmen nur textuell Bezug auf Materialien in der Vorgangsmappe
- Auf dem Laufzettel können nicht abgehakte Tätigkeiten durchgestrichen werden
- Beim Abhaken oder Durchstreichen einer Tätigkeit werden Zeitpunkt und handelnder Benutzer festgehalten
- Das Abhaken oder Durchstreichen einer Tätigkeit kann (nach Bestätigung) nicht wieder rückgängig gemacht werden

Vorgangsmappe

- Die Vorgangsmappe ist charakterisiert durch folgende Eigenschaften:
  - Sie ist ein Behälter für Materialien
  - An ihr kann ein Laufzettel angebracht werden
  - Sie ist versendbar
- Die Weitergabe der Vorgangsmappen soll in der Regel über das Postversandsystem erfolgen
- Die Adressierung der Vorgangsmappen wird mit der Tätigkeitsliste des Laufzettels verknüpft

Vorlagen

- Die Benutzer können generische Vorgangsmappen- und Laufzettelvorlagen erstellen
- Die Vorgangsmappen und Laufzettel stellen selbst Vorlagen dar (Prototypen)
- Die Vorlagen können in einem zentralen Vorlagenordner abgelegt werden

- Bei Verwendung einer Vorgangsmappe als Vorlage werden fachliche Kopien der in ihr enthaltenen Materialien in die neu erzeugte Vorgangsmappe eingefügt.
- Einer Vorgangsmappen-Vorlage kann eine Menge von Materialvorlagen *assoziiert* werden (Ausbaustufe)

Sonstiges

- Die Framework-Komponente für Vorgangsmappen macht keine Annahmen über die Organisationsmodellierung und ist in dieser Hinsicht für den Anwendungsentwickler anpassbar

Das Benutzungsmodell

Der Begriff des Benutzungsmodells wurde bereits in Abschnitt 0 (Definition 1 auf Seite 12) vorgestellt. Mit ihm wird das Modell bezeichnet, das sich der Benutzer auf den verschiedenen Ebenen von Anwendungssoftware machen kann.

In der Definition des Begriffes Benutzungsmodell werden die Ebenen *Fachliche Gegenstände, Konzepte, Abläufe* und *Handhabung und Präsentation* genannt, auf denen das Benutzungsmodell gelagert ist. Es ist im WAM-Ansatz offengelassen, in welcher Form diese Ebenen und insbesondere ihr Zusammenspiel beschrieben werden sollen, es gibt also keine speziellen Entwurfsdokumente für Benutzungsmodelle oder so etwas wie ein "Benutzungsmodelldiagramm". Vermutlich ist es auch gar nicht möglich, eine allgemeine Form für die Darstellung eines solchen Modells zu finden.

Ich habe mich entschieden, ausgehend von den Anwendungsfällen zur Verwendung von Vorgangsmappe und Laufzettel, die ich bereits in Abschnitt 4.1.4 für die physischen Kooperationsmittel in kurzer Form beschrieben habe, die verschiedenen Ebenen aufzublättern und das Benutzungsmodell in dieser Weise zu formulieren.

Der Darstellung des Benutzungsmodells anhand der Anwendungsfälle stelle ich die Erläuterung von zwei grundlegenden Konzepten voran, die sich nicht ausreichend gut einzelnen Anwendungsfällen zuordnen lassen.

5.1.7 Die "Lebensphasen" einer Vorgangsmappe

Die Erkenntnis, dass eine Vorgangsmappe verschiedene "Lebensphasen" durchläuft, ergibt sich zwangsläufig, wenn man einige Entwurfsentscheidungen zu Ende denkt, die ich formuliert habe.

Der Begriff "Lebensphase" ist deshalb passend, weil sich mit ihm die Vorstellung eines Anfangs (Erzeugen einer Vorgangsmappe), eines Endes (Abschließen eines Vorgangs, Archivieren von Mappeninhalten) und der dazwischenliegenden Phasen, in denen die Vorgangsmappe verschiedene Umgangsformen besitzt, verbindet.

Eine der Entwurfsentscheidungen, die die Modellierung solcher Phasen nahelegt, ist das Durchstreichen von Tätigkeiten auf dem Laufzettel und die daraus erwachsenden Konsequenzen: Wenn ein Anwender eine Vorgangsmappe auf Basis einer Vorlage erzeugt, muss er diese für den speziellen Vorgang anpassen können, noch bevor die eigentliche Bearbeitung des Vorgangs beginnt; beispielsweise will er eine Tätigkeit vom Laufzettel entfernen.

In diesem Fall erscheint es nicht sinnvoll, dass die Tätigkeit durchgestrichen werden *muss*, es sollte auch möglich sein, sie einfach zu löschen (denn die Vorgangsmappe wird noch auf den Vorgang *angepasst*, die *Bearbeitung* des Vorgangs hat noch nicht begonnen).

Umgekehrt darf es nicht möglich sein, eine Tätigkeit während der Bearbeitung eines Vorgangs einfach zu löschen; sie muss stattdessen durchgestrichen werden, damit so dokumentiert wird, wer wann die Tätigkeit gestrichen hat (derjenige ist außerdem aufgefordert, das "warum" selbst auf dem Laufzettel zu dokumentieren).

Diese Überlegungen führen mich dazu, von einer Phase der *Anpassung* (Customize) der Vorgangsmappe (und des Laufzettels) zu sprechen, und von einer daran anschließenden Phase der Benutzung (Use). Die letzte Phase, nach Abschluss des Vorgangs, ist die Phase der *Ablage* (File).

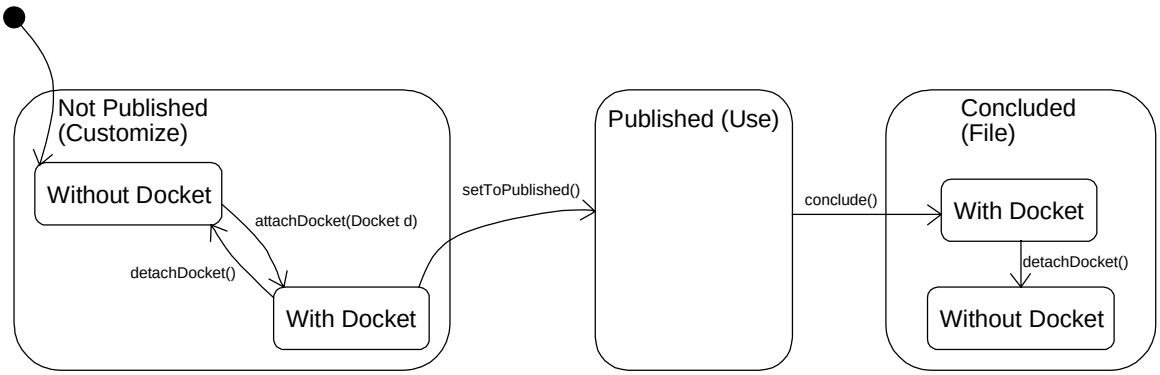


Abbildung 24: State-Chart-Diagramm für die "Lebensphasen" der Vorgangsmappe

Abbildung 24 zeigt die Phasen als State-Chart-Diagramm. Nach der Erzeugung ist die Vorgangsmappe im Zustand "Not Published (Customize)"[56] und ohne einen Laufzettel[57]. In diesem Zustand kann an der Vorgangsmappe ein Laufzettel angebracht, er kann aber auch wieder entfernt werden (der Zustand wechselt dann zwischen den Subzuständen "Without Docket" und "With Docket"). Der Laufzettel selbst kennt übrigens nur die Zustände "Publiziert" und "Nicht Publiziert" und vollzieht den Zustandswechsel gemeinsam mit der Vorgangsmappe.

Im nicht publizierten Zustand können Tätigkeiten am Laufzettel nicht nur durchgestrichen, sondern wahlweise auch gelöscht werden (das Hinzufügen und Modifizieren der Tätigkeitsbeschreibungen ist stets möglich). Wenn ein Laufzettel angebracht und die Vorgangsmappe für den speziellen Vorgang angepasst worden ist (der Laufzettel ist ggf. modifiziert worden, es sind Materialien in die Mappe eingefügt worden), kann die Vorgangsmappe *publiziert* werden und geht damit in die Phase "Published (Use)" über.

Der Laufzettel kann von der Vorgangsmappe *nicht entfernt werden*, während sie im publizierten Zustand ist, beide gehören bei der Bearbeitung des Vorgangs untrennbar zusammen. Mit dem Übergang in diese Phase können auch keine Tätigkeiten mehr vom Laufzettel gelöscht werden, wohl aber durchgestrichen (dies wurde in diesem Abschnitt bereits erläutert).

Nachdem der Vorgang bearbeitet wurde (alle auf dem Laufzettel aufgeführten Tätigkeiten müssen entweder abgehakt oder durchgestrichen sein), kann er abgeschlossen werden. Die Vorgangsmappe geht in die Phase "Concluded (File)" über. Der Laufzettel kann in dieser Phase

---

[56] Diese Benennung rührt daher, dass diese Zustände auch für die Vorgangsverfolgung, die in Kapitel 6 und 7 dargestellt wird, eine Rolle spielen. In diesem Zusammenhang ist es sinnvoll, den Übergang von der Anpassungs- in die Benutzungsphase der Vorgangsmappe als "Publizieren" der Mappe anzusehen, da die Mappe von diesem Zeitpunkt an "im System bekannt" ist, also z.B. bei Suchanfragen beachtet wird.

[57] Wenn die Vorgangsmappe auf Basis einer entsprechenden Vorlage erzeugt wird, kann natürlich bereits ein Laufzettel angebracht sein, technisch betrachtet gibt es aber einen kurzen Moment ohne Laufzettel, und der soll hier genügen, um den Anfangzustand des Diagramms zu rechtfertigen.

wieder von der Vorgangsmappe abgenommen werden (um ihn z.B. einzeln zu archivieren). An der Mappe kann kein neuer Laufzettel angebracht werden, so dass sie nicht wieder verwendet werden kann.

Während aller “Lebensphasen” können Materialien in die Vorgangsmappe eingefügt oder daraus entnommen werden.

5.1.8 Die Reihenfolge der Tätigkeiten auf dem Laufzettel

Der Ausgangspunkt dieser Überlegungen ist der schon mehrfach dargestellte Grundsatz, dass die Reihenfolge der Tätigkeiten auf dem Laufzettel einen Standardfall widerspiegeln soll. Für einen unbearbeiteten Laufzettel (gemeint ist ein Laufzettel, auf dem alle Tätigkeiten gelistet und beschrieben sind, aber keine von ihnen bearbeitet ist) ist die Bedeutung der Tätigkeiten-Reihenfolge damit ganz einfach zu verstehen. Es ist aber noch nicht eindeutig festgelegt, welche Bedeutung die Reihenfolge im Vorgangsverlauf haben soll. Es gibt hier im Wesentlichen zwei Möglichkeiten:

- Die Reihenfolge ändert sich nicht. Dann spiegelt sie stets (bis auf Einfügungen) die Planung wider, wie sie zu Beginn des Vorgangs war. Wenn Tätigkeiten abgehakt werden, kann man (an den Lücken) leicht erkennen, wo etwas entgegen dem eigentlichen Plan noch nicht erledigt wurde. Dies entspricht übrigens auch dem physischen Vorbild.
- Die Reihenfolge spiegelt den Bearbeitungsverlauf wider, d.h.: Die zuerst durchgeführte (bzw. durchgestrichene) Tätigkeit wandert an den Anfang der Liste, die als zweite durchgeführte an die zweite Stelle usw. Ein Vorteil dieser Darstellung ist, dass die noch nicht durchgeführten und nicht durchgestrichenen Tätigkeiten einen „Rest-Laufzettel“ bilden, der in seiner Reihenfolge planerisch umsortiert werden kann. Dieses Modell löst sich vom physischen Vorbild und nutzt die zusätzlichen Möglichkeiten, die eine Abbildung auf Software bietet.

Der Vorteil der ersten Möglichkeit besteht in dem hohen Wiedererkennungswert, den die Struktur des Laufzettels in diesem Fall besitzt. Bei dieser Umsetzung würde der Bearbeiter für bestimmte Vorgangstypen die Struktur des Laufzettels genau kennen und könnte (da sie bei den konkreten Vorgängen immer gleich ist) mit einem Blick ersehen, welche Tätigkeiten erledigt, unerledigt oder durchgestrichen sind. Um diesen Wiedererkennungswert nicht zu beeinträchtigten, müssten nachträglich hinzugefügte Tätigkeiten am Ende des Laufzettels auftauchen. Der Nachteil ist allerdings, dass sich das Ansinnen, noch unerledigte Tätigkeiten in einer anderen Reihenfolge als ursprünglich vorgesehen auszuführen, nicht mehr ausdrücken lässt.

Eine starre Reihenfolge bietet also einen Vorteil in Bezug auf die schnelle Lesbarkeit des Laufzettels, wenn man den entsprechenden Vorgangstyp gut kennt, beschränkt aber die Semantik des Laufzettels in seiner Rolle als veränderbarer Plan.

Die zweite Möglichkeit erlaubt es dagegen, den Grundsatz, dass die Reihenfolge der Tätigkeitenliste einen Hinweis auf eine sinnvolle Bearbeitungsreihenfolge gibt, im Prozessverlauf fortzuschreiben, wie das folgende Beispiel verdeutlicht:

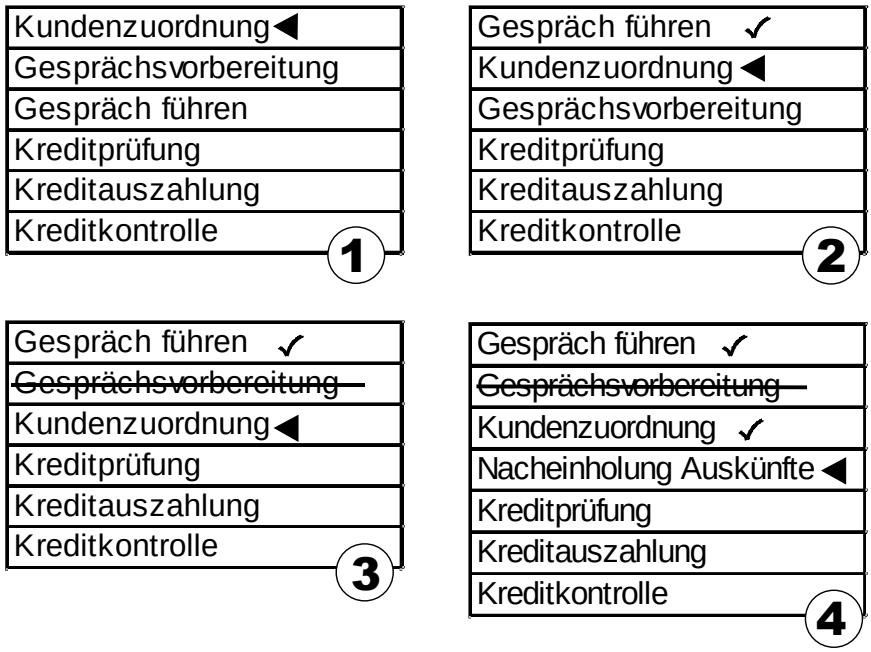


Abbildung 25: Bedeutung der Reihenfolge der Tätigkeiten auf dem Laufzettel

Zur Erläuterung der Abbildung: Die Tätigkeitenliste 1 besteht nur aus unerledigten Tätigkeiten, wobei die erste natürlich die anstehende ist (markiert durch die nach links weisende, dreieckige Spitze). Nachdem ein Bearbeiter das auf der Liste erwähnte Gespräch geführt hat - ohne dass zuvor eine geplante Kundenzuordnung stattgefunden hat, geschweige denn eine Gesprächsvorbereitung – markiert der Bearbeiter die Tätigkeit „Gespräch führen“ als erledigt (zu erkennen an dem Häkchen), die daraufhin an die erste Stelle der Liste rückt. Danach ist „Kundenzuordnung“ noch immer die anstehende Tätigkeit (beides zu sehen in Tätigkeitenliste 2).

Da das Gespräch nicht vorbereitet wurde, streicht der Bearbeiter die Tätigkeit „Gesprächsvorbereitung“ durch, was dazu führt, dass die durchgestrichene Tätigkeit auf die zweite Stelle der Liste rückt (zu sehen an Tätigkeitenliste 3). Da nach kurzer Absprache klar ist, dass der Bearbeiter den Kunden weiter betreuen wird, markiert er auch die Tätigkeit „Kundenzuordnung“ als erledigt. Um Teile der durchgestrichenen Tätigkeit „Gesprächsvorbereitung“ nachholen zu lassen, fügt der Bearbeiter die Tätigkeit „Nacheinholung Auskünfte“ ein, und zwar als die anstehende. Tätigkeitenliste 4 entspricht diesem Bearbeitungsstand.

Da ich den Laufzettel vorrangig unter dem Gesichtspunkt der Vergegenständlichung des Plans im Sinne der unterstützenden Sichtweise betrachte (dieser ist ja per Definition dazu gedacht, fortlaufend verändert und angepasst zu werden), erhält die Tätigkeiten-Reihenfolge in meinem Entwurf die Bedeutung im Sinne des zweiten Modells[58].

5.1.9 Das Benutzungsmodell anhand von Anwendungsfällen

Ich erläutere in diesem Abschnitt das Benutzungsmodell für die Verwendung von Vorgangsmappe und Laufzettel anhand der Anwendungsfälle, die ich in Abschnitt 4.1.4 für die physischen Kooperationsmittel beschrieben habe. Dabei habe ich den ersten Anwendungsfall von

[58] Es ist Werkzeug-seitig natürlich denkbar, zusätzlich eine Anzeige der „starren Reihenfolge“ vorzunehmen.

“Laufzettelvorlage erstellen” in “Vorlage erstellen” umbenannt, da bei den elektrifizierten Kooperationsmitteln auch Mappenvorlagen existieren.

Die Anwendungsfälle, die im Folgenden dargestellt werden, sind demnach

- Vorlage erstellen
- Vorgang starten
- Laufzettel ändern
- Tätigkeit durchführen und kommentieren
- Mappe weiterversenden
- Vorgang abschließen

5.1.9.1 Vorlage erstellen

Im zentralen Vorlagenorder[59] können Vorlagen für Vorgangsmappen mit Laufzettel abgelegt werden. Außerdem können auch Vorlagen für Vorgangsmappen ohne Laufzettel oder auch einzelne Laufzettel aufbewahrt werden — der Vorlagenordner enthält nämlich auch die “leeren” Vorlagen für Vorgangsmappe und Laufzettel, da das Laufzettel- und das Vorgangsmappen-Werkzeug keine Erzeugungsfunktionalität besitzen.

Der Vorlagenordner kann auf dem Desktop visualisiert werden, das Einfügen von Vorlagen geschieht dann durch Drag and Drop[60] (tatsächlich wird eine technische Kopie des Materials in den Ordner eingefügt).

Die Vorlagen sind ausgezeichnete Exemplare des Typs Vorgangsmappe oder Laufzettel. Das Erstellen einer Vorlage geschieht durch die zielgerichtete Bearbeitung eines entsprechenden Materials (also Befüllen der Vorgangsmappe mit Materialien, Eintragen von Tätigkeitsbeschreibungen auf dem Laufzettel) und das Einfügen in den Vorlagenordner.

[59] Vorlagenordner und Vorlagenordner-Werkzeug sind von mir in minimaler Form implementiert worden, um das Benutzungsmodell zu komplettieren. In diesem Kapitel ist deshalb ihre Handhabung nur soweit beschrieben, wie es für das Formulieren des Benutzungsmodells notwendig ist.

[60] Ich erkläre in Fußnote 51 auf Seite 64, warum andere Möglichkeiten nicht implementiert wurden.

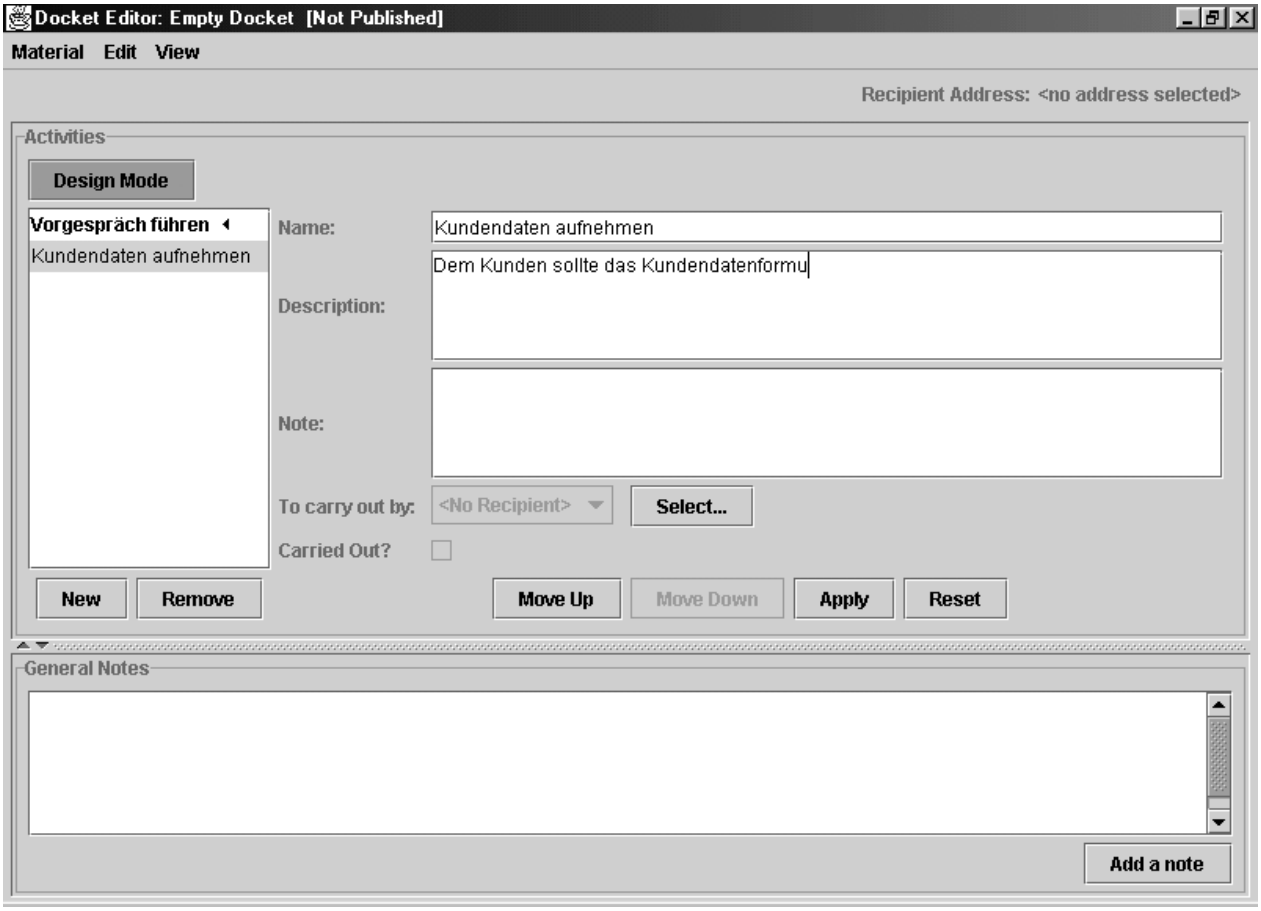


Abbildung 26: Das Laufzettelwerkzeug

Abbildung 26 zeigt das Laufzettelwerkzeug (es kann einzeln verwendet werden, ist aber auch ein Teil des Vorgangsmappenwerkzeugs, siehe Abbildung 27 auf Seite 76). In der Mitte links befindet sich der Bereich mit der Liste der Tätigkeiten. Der Editor-Teil rechts daneben zeigt die ausgewählte Tätigkeitsbeschreibung an und erlaubt ihre Bearbeitung. Im unteren Bereich können Notizen eingetragen werden, die sich auf den gesamten Vorgang beziehen.

Über der Tätigkeitsliste befindet sich ein Knopf mit der Aufschrift “Design Mode”. Dieser Knopf, der zwischen den Zuständen *hereingedrückt* und *hervorstehend* umgeschaltet werden kann (also kein normaler Druckknopf ist), schaltet das Werkzeug zwischen dem Entwurfsmodus und dem Arbeitsmodus um. Im Entwurfsmodus können genau die Tätigkeiten vorgenommen werden, die zum Erstellen einer Vorlage notwendig sind: Einfügen einer neuen Tätigkeitsbeschreibung (mit dem Knopf “New”), Löschen einer Tätigkeitsbeschreibung (“Remove”), Ändern der Reihenfolge (mit den Knöpfen “Move Up”, “Move Down”), und natürlich: Bearbeiten der Tätigkeitsbeschreibung.

Dies wiederum umfasst im Entwurfsmodus das Festlegen des Tätigkeitsnamens und der Beschreibung sowie das Bestimmen derjenigen Benutzer und/oder Rollen, die für die Durchführung der Tätigkeit geeignet sind (das Drücken des “Select”-Knopfes führt dazu, dass ein entsprechender Auswahl-Dialog angezeigt wird, der vom Anwendungsentwickler bereit gestellt werden muss — siehe Abschnitt 5.1.10 ab Seite 78). Änderungen müssen jeweils explizit durch Drücken des “Apply”-Knopfes übernommen werden.

Beim Einfügen eines “benutzten” Laufzettels als Vorlage in den Vorlagenordner (auch wenn dieser als Teil einer Vorgangsmappe eingefügt wird) werden durchgestrichene Tätigkeiten und Bearbeitungskommentare entfernt und abgehakte Tätigkeiten zurückgesetzt, so dass die Laufzettelvorlage “unbenutzt” ist.

Damit ein bereits benutzter Laufzettel auch als Grundlage des Entwurfs einer Vorlage dienen kann, stelle ich ein oberflächenloses Mini-Werkzeug, den *Vorlagenersteller* zur Verfügung (der beispielsweise über das Kontextmenü des JWAM-Desktops gestartet werden kann), welcher aus einem benutzten Laufzettel eine unbenutzte Kopie erstellt und diese in die Arbeitsumgebung legt.

5.1.9.2 Vorgang starten

Um einen Vorgang zu starten, also einen kooperativen Arbeitsprozess auf Basis einer Vorgangsmappe in Gang zu setzen, muss die Vorgangsmappe zuerst erzeugt und für den Prozess angepasst werden. Der Anwender öffnet hierzu das Werkzeug für den Vorlagenordner und erzeugt eine entsprechend vorbereitete Vorgangsmappe (die typischerweise bereits den entsprechend vorbereiteten Laufzettel enthält, andernfalls muss auch dieser aus einer Vorlage erzeugt und per Drag and Drop an der Mappe befestigt werden).

Der Anwender kann zur Anpassung der Mappe Materialien in diese einfügen und am Laufzettel Tätigkeiten streichen, löschen oder neue hinzufügen sowie Notizen anfügen (die Handhabung des Laufzettelwerkzeugs ist im vorigen Abschnitt kurz erläutert worden).

Der Vorgang wird gestartet, indem der Anwender entweder eine erste Tätigkeit durchführt und abhakt (siehe den Anwendungsfall *Tätigkeit durchführen und dokumentieren* in Abschnitt 5.1.9.4 ab Seite 75) oder die Mappe an einen anderen Bearbeiter versendet (siehe den Anwendungsfall *Mappe weiterversenden* ab Seite 77). Dies führt automatisch zum Publizieren der Vorgangmappe, wie es im Abschnitt zu den “Lebensphasen” einer Vorgangsmappe beschrieben wurde: Der Laufzettel kann nicht mehr von der Vorgangsmappe abgelöst werden, und die aufgeführten Tätigkeiten können nicht mehr gelöscht, sondern nur noch durchgestrichen werden.

Ein Vorgang könnte aber auch ganz ohne Beteiligung eines Bearbeiters gestartet werden, etwa wenn eine Anfrage direkt vom Kunden ins Anwendungssystem gelangt (z.B. bei einer Eingabe über das Internet), oder wenn bestimmte Vorgänge zeitgesteuert angestoßen werden (z.B. die Erstellung eines quartalsweise abzufassenden Berichts). Der Anwendungsentwickler muss in diesem Fall entscheiden, ob die Anpassungsphase automatisch beendet wird, bevor der erste Bearbeiter die Vorgangsmappe erhält.

5.1.9.3 Laufzettel ändern

Unter dem Ändern eines Laufzettels verstehe ich an dieser Stelle die Veränderung eines mit einer Vorgangsmappe publizierten Laufzettels, und zwar eine Veränderung an seiner Struktur, eine Abänderung des durch ihn formulierten Plans. Dabei ist die Struktur des Laufzettels in den Beschreibungen der einzelnen Tätigkeiten und ihrer Reihenfolge (die eine Standard-Bearbeitungsreihenfolge darstellt) zu sehen.

Der Laufzettel ist konzeptuell gleichermaßen Plan des zukünftigen Prozessverlaufs wie auch Dokumentation des bisherigen Prozessverlaufs. Dieser doppelten Bedeutung trage ich dadurch Rechnung, dass die als erledigt oder durchgestrichen markierten Tätigkeiten im Gegensatz zu den übrigen nicht mehr verändert werden können. Der Laufzettel spiegelt dabei den tatsächlichen Prozessverlauf wider, indem die erledigten Tätigkeiten am Beginn der Tätigkeitenliste stehen, in der Reihenfolge ihrer Erledigung. Dies ist ein Unterschied zum physischen Vorbild, der sicherstellt, dass der „erledigte/gestrichene Teil“ der Tätigkeiten in der korrekten zeitlichen Reihenfolge gelistet wird und der „unerledigte Teil“ in der geplanten Reihenfolge. Nach Abschluss des gesamten Prozesses dokumentiert der Laufzettel damit den zeitlichen Ablauf des Prozesses in seiner Reihenfolge (s. Abschnitt 5.1.8, “Die Reihenfolge der Tätigkeiten auf dem Laufzettel” ab Seite 70).

Dementsprechend ändern folgende Handlungen den Laufzettel ab:

- Einfügen einer neuen Tätigkeit in die unerledigten Tätigkeiten
- Verändern der Benennung oder Beschreibung einer Tätigkeit
- Durchstreichen einer Tätigkeit
- Ändern der Reihenfolge innerhalb der unerledigten Tätigkeiten

Diese Handlungen können mit dem Laufzettelwerkzeug durchgeführt werden (s. Abschnitt 5.1.9.1 ab Seite 72 und den folgenden Abschnitt).

Die erste nicht erledigte oder nicht gestrichene Tätigkeit auf dem Laufzettel ist die nächste durchzuführende (dies ist auch maßgeblich für die Adressierung der Vorgangsmappe, siehe Abschnitt 5.1.9.5, „Mappe weiterversenden“). Streng genommen ist deswegen auch das vorgezogene Durchführen einer Tätigkeit und ihr Abhaken eine Änderung des Laufzettels, weil es implizit zu einer Änderung der Reihenfolge führt.

Dieser Fall sollte normalerweise nur dann auftreten können, wenn ein Bearbeiter von vornherein für mehrere Tätigkeiten vorgesehen ist. Da die Bearbeiter jedoch den Laufzettel situativ abändern können, sind Ausnahmen hiervon nicht ausgeschlossen.

5.1.9.4 Tätigkeit durchführen und dokumentieren

Im Normalfall führt der Bearbeiter die jeweils nächste Tätigkeit durch (das ist die erste nicht erledigte oder nicht durchgestrichene auf dem Laufzettel), kommentiert die Erledigung und markiert die Tätigkeit als erledigt. Er kann dies aber auch mit einer auf dem Laufzettel weiter unten aufgelisteten Tätigkeit tun (dass dies die Ausnahme darstellt, wurde bereits zu Ende des vorigen Abschnitts erörtert). Nachdem die Erledigung der Tätigkeit noch einmal bestätigt wurde, wird sie entsprechend der Bearbeitungsreihenfolge hinter die zuvor erledigten Tätigkeiten eingeordnet und ist nicht mehr veränderbar.

Das Durchführen einer Tätigkeit umfasst gegebenenfalls Handlungen, die komplett außerhalb der Softwareunterstützung stattfinden wie z.B. ein persönliches Gespräch. Wenn diese allerdings eine Wichtigkeit für den Vorgang hat, werden auch die Handlungen, die ohne Softwareunterstützung stattfinden in der Regel in irgendeiner Form in der Vorgangsmappe dokumentiert werden (bei einem Gespräch wird z.B. ein Gesprächsprotokoll angefertigt und in die Mappe gelegt). Solche Materialien können im späteren Vorgangsverlauf für andere Bearbeiter wichtig sein.

Der Inhalt der Vorgangsmappe hat zum einen dokumentierenden Charakter, besteht zum anderen aber auch aus Materialien, die im Sinne einer Producer/Consumer-Abhängigkeit weitergegeben werden. Somit geht das Durchführen einer Tätigkeit häufig einher mit dem Entnehmen/Einfügen von Materialien aus der bzw. in die Vorgangsmappe und das Bearbeiten von in ihr enthaltenen Materialien.

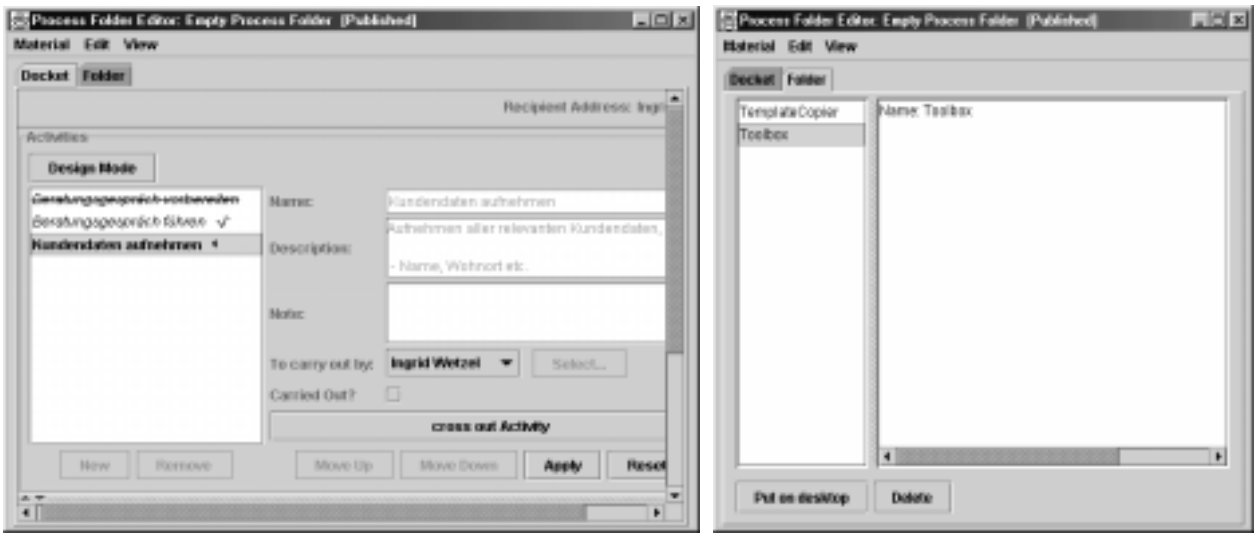


Abbildung 27: Das Vorgangsmappenwerkzeug in Laufzettel- und Mappenansicht

Abbildung 27 zeigt das Vorgangsmappenwerkzeug einmal in der Laufzettel- und das andere Mal in der Mappenansicht, die über die Registerkarten ausgewählt werden können (das Standard-Behälterwerkzeug aus dem JWAM-Framework und das Laufzettelwerkzeug sind hier als Subwerkzeuge eingebunden). Das Laufzettelwerkzeug befindet sich im Arbeitsmodus, in dem keine neue Tätigkeit in die Liste eingefügt und die Reihenfolge der Liste nicht verändert werden kann (ein Löschen ist ohnehin nicht mehr möglich, weil die Vorgangsmappe publiziert ist). Im Arbeitsmodus kann dagegen eine Notiz zur Durchführung einer Tätigkeit eingetragen und diese abgehakt oder durchgestrichen werden, was nach Bestätigung durch den "Apply"-Knopf wirksam wird. Die Abbildung zeigt in der Tätigkeitsliste auch bereits eine durchgestrichene und eine abgehakte Tätigkeit.

Die Mappenansicht zeigt die in der Mappe enthaltenen Materialien und erlaubt, sie aus der Mappe zu entnehmen und in die Arbeitsumgebung zu legen oder aber sie zu löschen. Das Einfügen in die Vorgangsmappe geschieht ausschließlich über Drag and Drop[61].

Die Vorgangsmappe realisiert dabei das *Konzept der Einheit von Ort und Zeit*. Damit ist gemeint, dass ein in der Mappe enthaltenes Material nur dort vorhanden ist – Änderungen daran haben keine sofortige Auswirkung auf Gegenstände außerhalb der Vorgangsmappe, und umgekehrt hat eine Veränderung außerhalb der Vorgangsmappe keine modifizierende Auswirkung auf Materialien in ihr. Damit verhält sich die Vorgangsmappe in dieser Hinsicht wie ihr physisches Vorbild, was zu einem gut nachvollziehbaren Benutzungsmodell führt. Zur Bearbeitung eines Materials in der Vorgangsmappe muss dieses entnommen und nach Bearbeitung wieder eingefügt werden.

Die Entwurfsentscheidung, die Einheit von Ort und Zeit umzusetzen, hat aber auch negative Konsequenzen. Als Beispiel stelle man sich eine Vorgangsmappe vor, die ein Kundenblatt mit den wichtigsten Kundendaten enthält. Wenn nun, während dieser Vorgang in Bearbeitung ist (die Mappe also von Bearbeiter zu Bearbeiter wandert), an anderer Stelle (etwa in einer zentralen Kundenregistatur) eine Adressänderung für diesen Kunden aufgenommen wird, verbleibt im Kundenblatt in der Vorgangsmappe weiterhin die veraltete Adressinformation. Das einzig Erfreuliche an diesem Umstand ist, dass dem Anwender aufgrund des einfachen Benutzungsmodells bewusst ist, dass dies vorkommen kann.

[61] Dies ist die Funktionalität des Standard-Behälterwerkzeug des JWAM-Framworks. Warum andere Möglichkeiten nicht implementiert wurden, erkläre ich in Fußnote 51 auf Seite 64.

Aus diesem Grund müssen Materialien, die im Arbeitsprozess einerseits auf einem aktuellen Stand benötigt werden, andererseits aber während des Vorgangsverlaufs veralten können, jeweils neu beschafft werden und können nicht mit der Vorgangsmappe verschickt werden.

Wenn umgekehrt Änderungen an Materialien, die bei der Bearbeitung eines Vorgangs vorgenommen werden, nicht erst beim Abschließen des Vorgangs nach außen sichtbar werden sollen, reicht es ebenfalls nicht aus, diese mit der Vorgangsmappe weiterzuverschicken, sondern die Änderung muß durch eine explizite Handlung nach außen bekanntgemacht werden.

Nun ist es einer der Vorzüge der Verwendung von Vorgangsmappen, dass der Bearbeiter dort möglichst viele der für den Vorgang relevanten Materialien vorfindet und sie sich nicht umständlich zusammensuchen muss. Die obigen Darlegungen zeigen aber, dass hin und wieder doch das Beschaffen von Materialien oder ihr Ablegen außerhalb der Mappe nötig ist.

Ein Vorschlag, um dies besser zu unterstützen, wäre die Einführung von *Abholzetteln* (gegenwärtig ist dies nicht implementiert). Ein solcher Abholzettel enthält allgemeine Informationen über das Material und seinen Aufbewahrungsort. Möglichst direkt (etwa über ein Kontextmenü) sollte für einen Abholzettel die Bearbeitung des damit beschriebenen Materials durch eines der möglichen Werkzeuge bestimmt werden können, was zur Beschaffung des Materials und das Bearbeiten mit dem Werkzeug führt.

Wenn ein solcher Abholzettel in der Vorgangsmappe verschickt wird, kann die Synchronisation zwischen dem Informationsstand, der sich in den Materialien in der Mappe manifestiert, und dem, der außerhalb des Vorgangs sichtbar ist, unterstützt werden. Das Benutzungsmodell ist in dieser Hinsicht natürlich einfacher zu verstehen, wenn ein solcher Abholzettel nicht als „Insellösung“ nur für die Vorgangsmappen verwendet wird, sondern im ganzen Anwendungssystem einheitlich eingesetzt wird[62].

5.1.9.5 Mappe weiterversenden

Wenn eine Vorgangsmappe erst einmal eine Adressierung aufweist, kann sie verschickt werden, indem sie in den Ausgangskorb gelegt wird. Doch auf welche Weise erhält die Vorgangsmappe ihre Adressierung? Wenn man die Funktion des Laufzettels reflektiert, ist es naheliegend,die Adressierung der Vorgangsmappe mit dem Laufzettel zu verknüpfen, wie ich es bereits in Abschnitt 5.1.4.6 (“Weitergabe der Vorgangsmappen”, ab Seite 65) beschrieben habe.

Der Laufzettel vergegenständlicht den bearbeitbaren Plan des Vorgangsablaufs. Die erste nicht durchgestrichene und noch nicht erledigte Tätigkeit in der Liste des Laufzettels ist die nächste zu erledigende, die *aktuelle Tätigkeit.* Dieser Tätigkeit sind mögliche Bearbeiter (Rollen) zugeordnet. Wenn eine(r) von diesen ausgewählt wird, ergibt sich sich aus der zugeordneten Postadresse[63]die Adressierung der Vorgangsmappe.

Das bedeutet, dass eine Vorgangsmappe nur dann verschickt werden kann, wenn ein Laufzettel angebracht worden ist. Dieser Laufzettel muss zudem noch mindestens eine unerledigte Tätigkeit aufweisen, und für diese muss ein Bearbeiter (bzw. eine entsprechende Rolle) bestimmt sein.

---

[62] Tatsächlich ist diese Idee aufgegriffen worden und (allerdings nach dem vorläufigen Abschluss meiner Implementationsarbeiten) eine Fachwertklasse `dvLocation` ins JWAM-Framework eingeführt worden, die ähnlich wie eine URL den Aufbewahrungsort eines Materials bezeichnet und die Grundlage für einen solchen Abholzettel sein kann.

[63] Den Mechanismus, der einem Benutzer/einer Rolle eine Transportadresse zuordnet, muss der Anwendungsentwickler bereitstellen — siehe Abschnitt 5.1.10 ab Seite 78.

Wenn das System solches bereitstellt, ist es natürlich auch möglich, eine Vorgangsmappe durch das Einpacken in einen Briefumschlag verschickbar zu machen, der „per Hand“ adressiert wird. Dies ist jedoch als ein Umgehen der intendierten Benutzungsweise der Vorgangsmappe anzusehen.

5.1.9.6 Vorgang abschließen

Voraussetzung für das Abschließen eines Vorgangs ist, dass der Laufzettel der entsprechenden Vorgangsmappe keine Tätigkeiten mehr auflistet, die weder gestrichen noch erledigt sind. Das Abschließen eines Vorgangs umfasst die Handlungen, die das durch den Vorgang erzielte Arbeitsergebnis „auswerten“. Typischerweise wird hierbei die Mappe entweder als ganzes archiviert oder aber es werden selektiv Materialien zur weiteren Aufbewahrung oder Bearbeitung entnommen. Diese Abwicklung der Mappe kann vom Anwendungssystem (spezialisiert für bestimmte Mappen- bzw. Vorgangstypen) durch einen Abwicklungsautomaten unterstützt werden, wie es beim UBS Corporate Desktop der Fall ist. Am Ende setzt der Bearbeiter (oder ggf. der Automat) den Zustand der Mappe auf „abgeschlossen“ um, indem er den Punkt “Conclude Process” aus dem Menü “Material” des Vorgangsmappenwerkzeugs auswählt.

Der Laufzettel einer abgeschlossenen Vorgangsmappe kann von ihr abgelöst werden, z.B. um ihn einzeln zu archivieren oder als Basis für eine neue Vorlage zu verwenden.

Konstruktionsdetails

In diesem Abschnitt stelle ich einige Details der Konstruktion der Framework-Komponente dar, die zeigen, wie sich die getroffenen Entwurfsentscheidungen und das explizit formulierte Benutzungsmodell in der Implementation niederschlagen.

5.1.10 Entkopplung von der Organisationsmodellierung: RoleChooser und RoleMapper

In Abschnitt 5.1.5 mit der Überschrift “Rollen und Benutzergruppen” habe ich bereits festgelegt, dass in der Framework-Komponente für die Vorgangsmappen keine Annahmen über die Struktur der Organisation, in der die Vorgangsmappen eingesetzt werden, gemacht werden sollen, also unter anderem keine Annahmen über die Gruppen, Rollen, Benutzer in der Organisation und ihre Beziehungen (Enthaltensein, Stellvertreter usw.) zueinander.

Dennoch muss das Laufzettelwerkzeug dem Benutzer gegenüber die Möglichkeit anbieten, zu einer Tätigkeit die dafür in Frage kommenden Rollen oder Benutzergruppen oder konkreten Bearbeiter auszuwählen: Dies soll geschehen, wenn am Laufzettelwerkzeug für eine Tätigkeit der Knopf “Select” betätigt wird (siehe Abbildung 26 auf Seite 73). Als Reaktion auf den Knopfdruck könnte etwa ein Dialog geöffnet werden, der Benutzergruppen und Benutzer der Organisation hierarchisch darstellt und zum Auswählen anbietet.

Aufgrund der gemachten Prämisse kann ein solcher Auswahldialog aber nicht in der Framework-Komponente implementiert werden; hier muss ein Einschub[64] für eine *Software-Komponente* (eine sehr kleine, zugegebenermaßen — siehe trotzdem Abschnitt 2.1.6.1 auf Seite 24) vorgesehen werden, und so ist das Laufzettelwerkzeug auch konstruiert.

[64] Der Begriff *Einschubkomponente*, der eine Software-Komponente bezeichnet, die für einen speziellen *Einschub* konstruiert wird, stammt von Andreas Kornstädt, der ihn im Rahmen seiner noch laufenden Dissertation über *Ein Anwendungskomponentenrahmenwerk für die musikwissenschaftliche Analyse* geprägt hat.

Bevor ich diesen Einschub beschreibe, möchte ich noch die zweite Gelegenheit erwähnen, bei der das Laufzettelwerkzeug auf diese Einschubkomponente zurückgreifen muss: Wenn ein Anwender nämlich einen Bearbeiter oder eine Rolle oder Gruppe für die nächste Tätigkeit auswählt, muss dies auf eine Transportadresse des Postversandsystems abgebildet — auch hier ist Wissen über die Organisationsmodellierung nötig, das in der Framework-Komponente nicht vorhanden sein darf.

Das Anbieten von Einschüben für die Implementationen des Anwendungsentwicklers erfolgt über die Singleton[65] -Klasse `RoleUtilitiesProvider`. Dort können Implementationen für die Schnittstellen `RoleChooser` und `RoleMapper` angemeldet werden, also — um bei diesemWort zu bleiben — in den Einschub hineingeschoben werden. Das Laufzettelwerkzeug greift auf die dort angemeldeten Objekte zu, im ersten beschriebenen Fall auf den `RoleChooser`, im zweiten auf den `RoleMapper`,.

```
public interface RoleChooser
{
  public DomainValue[] askForToCarryOutBy();
}
```

Abbildung 28: Schnittstelle `RoleChooser`

Wenn wie oben beschrieben der “Select”-Knopf am Laufzettelwerkzeug gedrück wird, ruft dieses die einzige Methode des angemeldeten `RoleChooser`. Dieser kann — je nach Implementation — einen entsprechenden Auswahldialog öffnen. Zurückliefern muss er ein Array aus Fachwerten, die die für die Tätigkeit in Frage kommenden (ausgewählten) Benutzer, Rollen oder Gruppen repräsentieren[66]. Das Laufzettelwerkzeug zeigt ganz einfach die String-Darstellungen dieser Objekte an. Es muss die Bedeutung der einzelnen Fachwerte auch nicht kennen, da es die Umsetzung in eine Transportadresse an den `RoleMapper` delegiert (das heißt aber natürlich, dass `RoleChooser` und `RoleMapper` zusammenpassen müssen).

Die Schnittstelle `RoleMapper` muss dementsprechend in der Lage sein, für jeden der Fachwerte, die der RoleChooser zurückliefert, eine Transportadresse zu liefern, damit die Vorgangsmappe in den richtigen Postkorb verschickt werden kann:

```
public interface RoleMapper
{
  public dvTransportAddress adressForRole(DomainValue role);
}
```

Abbildung 29: Schnittstelle `RoleMapper`

Indem das Laufzettelwerkzeug durch die Verwendung der Einschubkomponenten selbst keinerlei Wissen bezüglich der im konkreten Kontext vorhandenen Organisationsstruktur besitzen muss, wird es in vielen (bezüglich der Organisationsstruktur) verschiedenen Anwendungskontexten einsetzbar.

[65] Ein Singleton ist die Bezeichnung für eine Klasse, die nur ein Exemplar hat. Das Singleton ist als Muster auch in [GHJ+] beschrieben.

[66] Eine Rückgabe von `null` wird als ergebnisloser Abbruch der Auswahl interpretiert.

5.1.11 Die Materialien

An dieser Stelle stelle ich die konstruierten Materialien vor. Ich leite diese Vorstellung mit einem Abschnitt ein, in dem ich die grundsätzlichen Eigenschaften von Materialien im JWAM-Framework kurz beschreibe.

Im Anschluss stelle ich die Schnittstellen für die Vorgangsmappe *(ProcessFolder)* und die Schnittstelle für den Laufzettel *(Docket)* vor. Bei der Beschreibung der Laufzettelschnittstelle gehe ich auch auf ein besonderes Zustandsmodell für die Tätigkeiten ein.

Auf die Schnittstellenbeschreibung der Tätigkeit *(Activity)* selbst verzichte ich dann aber, da zu den Umgangsformen, die auf der CRC-Karte für die Tätigkeit in Kapitel 4 beschrieben sind, nur diejenigen hinzugekommen sind, die mit dem Zustandsmodell zu tun haben, das bereits gemeinsam mit der Laufzettelschnittstelle erläutert wird.

Am Ende dieses Abschnitts bildet ein kurzer Exkurs zur Werkzeugkonstruktion und –komposition in JWAM die Grundlage für die Vorstellung der Klasse *ProcessFolderAdapter* (einer Vorgangsmappenimplementation als Dekorierer). Mit der in [Be00] beschriebenen neuen Werkzeugkonstruktion wird es möglich, parallel zur Komposition von Materialien auch zugehörige Werkzeuge zu kombinieren, was an diesem Beispiel diskutiert werden soll.

5.1.11.1 Materialien in JWAM

Im JWAM-Framework ist eine Basisschnittstelle für alle Materialien mit dem Namen `Thing` vorgegeben. Zu dieser Schnittstelle existiert auch eine abstrakte Implementation `ThingImpl`, von der nahezu alle im Framework implementierten Materialien erben.

Das Framework definiert damit eine Reihe von Umgangsformen, die allen Materialien gemein sind. Die wichtigsten sind folgende

- *Fachliche Identität:* Ein Material hat eine eindeutige fachliche Identität, eine ID, die auch bei einem Wechsel der technischen Identität (etwa durch Transfer auf einen anderen Rechner) erhalten bleibt.
- *Technische Kopie:* mit Hilfe der Operation `newClone()` kann von jedem Material eine technische Kopie hergestellt werden. Diese ist eine maximal tiefe Kopie, hat also keine einzige Objektreferenz mit dem Original gemeinsam.
- *Fachliche Kopie:* mit Hilfe der Operation `newCopy()` kann von jedem Material eine fachliche Kopie hergestellt werden. Wie flach oder tief diese Kopie ist, muss der Implementierer selbst nach fachlichen Gesichtspunkten festlegen, die fachliche Identität der Kopie ist jedenfalls anders als die des Originals
- *Beschreibung des Materials:* Jedes Material liefert über die Operation `thingDescription()` eine Materialbeschreibung (als Fachwert) zurück, die über ID, Name und Typ des Materials Auskunft gibt und den Pfad für eine Icon-Grafik enthalten kann.

Ich habe bei meinem Entwurf grundsätzlich Wert darauf gelegt, bei der Materialkonstruktion Interfaces zu verwenden. Bei der Programmiersprache Java, die im Hinblick auf die Klassenvererbung (also die Vererbung von Implementationen) nur Einfachvererbung erlaubt, ist nur dieses Vorgehen geeignet, um dem Anwendungsentwickler die notwendigen Freiheiten zu lassen: So kann eine von ihm konstruierte Klasse ggf. von einer anderen Klasse erben (was z.B. unumgänglich sein kann, um bestimmte Persistenzmechanismen für das Material nutzen zu können) und trotzdem die geforderte Materialschnittstelle implementieren.

Die von mir implementierten konkreten Materialklassen erben alle von `ThingImpl` oder einer spezielleren Unterklasse.

5.1.11.2 Vorgangsmappe (ProcessFolder)

In Abschnitt 5.1.4.2 ab Seite 62 habe ich die Vorgangsmappe bereits als ein fachlicher, versendbarer Behälter, an dem ein Laufzettel angebracht werden kann, gekennzeichnet

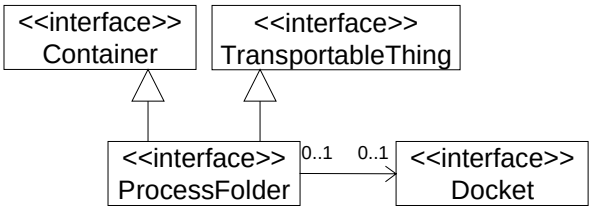


Abbildung 30: Schnittstelle `ProcessFolder` (Klassendiagramm)

Technisch bedeutet das, dass die Schnittstelle `ProcessFolder` das Interface `de.jwam.handling.containerconstruction.Container` erweitern muss, die allgemeinste Schnittstelle, die im JWAM-Rahmenwerk für fachliche Behälter vorgesehen ist[67]. Außerdem erweitert sie das Interface `TransportableThing`, um durch das Postversandsystem versendbar zu sein. Hierbei ist von besonderer Bedeutung, dass die Transportadresse, die über diese Schnittstelle erfragbar ist, aus dem angebrachten Laufzettel entnommen und durchgereicht wird. Somit ist die Vorgangsmappe nur versendbar, wenn ein Laufzettel angebracht und auf diesem für die anstehende Aufgabe ein Bearbeiter ausgewählt ist[68]. All dies zeigt Abbildung 30.

Eine konkrete Implementation von ProcessFolder in Form eines Dekorierers für beliebige JWAM-Behälter wird in Abschnitt 5.1.11.5 ab Seite 87 kurz vorgestellt.

Die wesentlichen „neuen“ (also nicht aus anderen Schnittstellen ererbten Umgangsformen) betreffen die in Abschnitt 5.1.7 dargestellten “Lebensphasen” der Vorgangsmappe und die Anbringung des Laufzettels. Zunächst zu den “Lebensphasen”:

```
public void setToPublished();
public boolean hasBeenPublished();
public void conclude();
public void hasBeenConcluded();
```

Abbildung 31: Operationen, die die “Lebensphasen” von `ProcessFolder` betreffen

Die Operationen `setToPublished()` und `conclude()` führen zu genau den Zustandsübergängen (aus *Not Published* zu *Published* und aus *Published* zu *Concluded*), die in Abbildung 24 auf Seite 69 mit diesen Operationsnamen gekennzeichnet sind. Die Methode `conclude()` hat als Vorbedingung, dass alle Tätigkeiten auf dem Laufzettel entweder abgehakt oder durchgestrichen sind (die Methode `hasBeenCompleted()` des Laufzettel muss `true` zurückliefern, s. Abschnitt 5.1.11.3).

Die Operationen `hasBeenPublished()` und `hasBeenConcluded()` dienen dazu, diese Zustände zu sondieren.

---

[67] Die wesentlichen Umgangsformen von `Container` sind das Erfragen eines Inhaltsverzeichnisses und das Einfügen und Entnehmen von Gegenständen.

[68] Das Postversandsystem betrachtet ein Objekt des Typs `TransportableThing` nur dann als versendbar, wenn an diesem eine Adresse gesetzt ist.

Zu den Methoden, die der Anbringung/Entfernung des Laufzettels dienen:

```
public void attachDocket(Docket docket);
public void detachDocket();
public boolean hasAttachedDocket();
public Docket attachedDocket();
```

Abbildung 32: Methoden zur Anbringung/Entfernung eines Laufzettels

Der Laufzettel wird mittels der Methode `attachDocket()` an der Vorgangsmappe angebracht. Voraussetzung ist, dass nicht bereits ein Laufzettel angebracht ist, was über die Methode `hasAttachedDocket()` in Erfahrung gebracht werden kann. Ein gegebenenfalls bereits angebrachter Laufzettel kann mittels `attachedDocket()` erfragt werden. Dieser kann, falls die Vorgangsmappe bereits abgeschlossen oder aber noch gar nicht publiziert ist (s.o.), durch einen Aufruf der Methode `detachDocket()` wieder entfernt werden.

5.1.11.3 Laufzettel (Docket)

Das Interface `Docket` definiert die Schnittstelle für den Laufzettel. Die Klasse `DocketImpl` ist die von mir bereitgestellte Implementationsklasse dazu

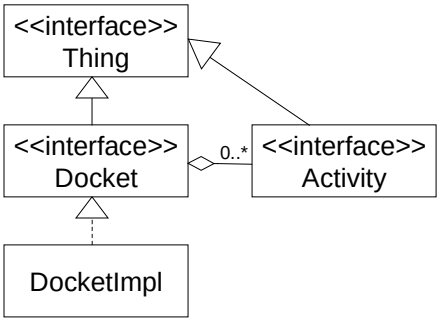


Abbildung 33: Die Schnittstelle Docket (Klassendiagramm)

Der zentrale Bestandteil des Laufzettels ist die Liste der Tätigkeiten (die zugehörige Schnittstelle heißt `Activity`, siehe Abbildung 33). Diese Tätigkeiten können aus der Liste gelöscht, sie können kommentiert, durchgestrichen oder abgehakt werden. Da der Laufzettel den Prozessverlauf dokumentieren soll, können aber einmal durchgestrichene oder abgehakte Tätigkeiten nicht mehr verändert werden. Um dies sicherzustellen, sind die Tätigkeiten im Laufzettel gekapselt und werden nur als technische Kopien herausgegeben. Änderungen an einzelnen Tätigkeiten müssen über die Operation `updateActivity()` am Laufzettel durchgeführt werden (s.u.).

Die erste naheliegende Idee, eine abgehakte oder durchgestrichene Tätigkeit auch technisch sofort endgültig zu „verriegeln“ (also weitere Veränderungen zu verhindern) habe ich deshalb nicht realisiert und stattdessen ein zweistufiges Konzept gewählt, das ich mit dem folgendem State-Chart-Diagramm illustrieren möchte:

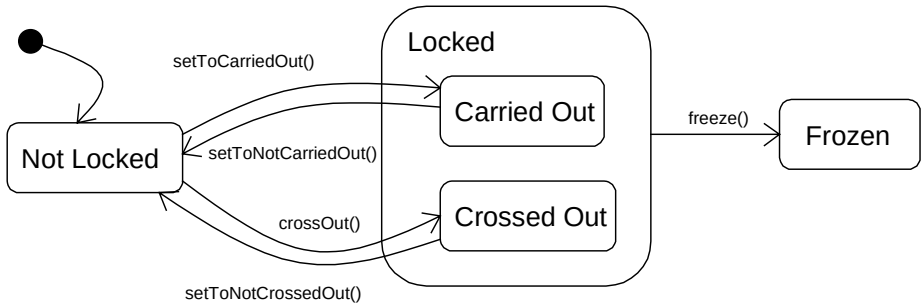


Abbildung 34: State Chart-Diagramm für eine Tätigkeit

Demnach kann sich eine Tätigkeit grundsätzlich in den Zuständen *Not Locked*, *Locked* und *Frozen* befinden. Im (initialen) Zustand *Not Locked* ist eine Tätigkeit weder durchgestrichen noch als erledigt markiert und alle Attribute (Name, Beschreibung etc.) können verändert werden. Indem die Tätigkeit durchgestrichen oder als erledigt markiert wird, gelangt sie in den Zustand *Locked*, und dort entsprechend in den Unterzustand *Carried Out* (erledigt) oder *Crossed Out* (durchgestrichen). Im Zustand *Locked* können außer den im State Chart sichtbaren Zustandsübergängen keine Veränderungen mehr an der Tätigkeit vorgenommen werden. Sowohl das Durchstreichen als auch das „Abhaken" kann allerdings noch zurückgenommen werden, womit der Zustand wieder zu *Not Locked* wechselte. Durch das Einfrieren der Tätigkeit gelangt die Tätigkeit schließlich aus dem Zustand *Locked* in den Zustand *Frozen*. In diesem Zustand ist die Tätigkeit vollständig unveränderlich.

Die `updateActivity()`-Methode am Laufzettel (s.u.) akzeptiert als Parameter dementsprechend nur nichteingefrorene Tätigkeiten und friert diese dann ein, falls sie erledigt oder durchgestrichen sind.

Im Folgenden geht es um die Operationen, die mit den in der Tätigkeitsliste enthaltenen Tätigkeiten umgehen. Dabei habe ich mich dafür entschieden, einige Methoden jeweils sowohl für einen indexbasierten als auch für einen ID-basierten Zugriff auf die Tätigkeiten auszulegen[69]. Die folgende Aufstellung enthält zunächst die in dieser Hinsicht „neutralen" Methoden (also weder indexbasiert noch ID-basiert):

```
public void updateActivity(Activity a);
public Activity[] activities();
public int activitiesCount();
public Activitiy[] frozenActivities();
public int frozenActivitiesCount();
public Activity[] notFrozenActivities();
public int notFrozenActivitiesCount();
public Activity currentActivity();
```

Abbildung 35: Allgemeine tätigkeitsbezogene Methoden

Die Methode `activitiesCount()` liefert die Anzahl der auf dem Laufzettel gelisteten Tätigkeiten zurück. Alle diese Aktvitäten liefert die Methode `activities()` als technische Kopien zurück[70]. Entsprechend der disjunkten[71] Aufteilung aller gelisteten Tätigkeiten in

[69] Gemeint ist hier die für jedes `Thing` eindeutige ID vom Typ `dvIdentificator`. (aus dem JWAM-Package `de.jwam.lang.domainvalue`).

[70] Die Reihenfolge ist die durch den indexbasierten Zugriff induzierte.

[71] Zwei Mengen A, B sind *disjunkt* genau dann, wenn $A \cap B = \emptyset$ gilt (ihre Schnittmenge leer ist).

„eingefrorene“ und „nichteingefrorene“ (s. Abbildung 34, „State Chart-Diagramm “ auf Seite 83) gibt es für diese beiden Fälle zusätzlich die Methodenpaare `frozenActivities()/frozenActivitesCount()` bzw. `notFrozenActivities()/notFrozenActivitesCount()`. Den Ausführungen zu der Reihenfolge der Tätigkeiten auf dem Laufzettel entsprechend (s. Abschnitt 5.1.8 ab Seite 70) entspricht die Liste aller Tätigkeiten der Konkatenation der eingefrorenen und der nichteingefrorenen Tätigkeiten.

Da es sich bei den Objekten des Typs `Activity`, die man von den oben beschriebenen Methoden erhalten kann, stets nur um technische Kopien handelt, haben Änderungen an diesen keine Auswirkungen auf den Laufzettel. Änderungen an den Tätigkeiten können nur übernommen werden, indem man die Methode `updateActivity()` mit der geänderten Tätigkeit als Parameter ruft. Vorbedingung ist, dass das vom Laufzettel verwaltete Gegenstück noch nicht eingefroren ist. Ist die übergebene Tätigkeit als erledigt markiert oder durchgestrichen, friert der Laufzettel sie mit dem Aufruf von `updateActivity()` ein.

Die Methode `currentActivity()` liefert die als nächstes zu erledigende Tätigkeit zurück (die erste nicht abgehakte und nicht durchgestrichene Tätigkeit auf dem Laufzettel).

Zu den ID-basierten, tätigkeitsbezogenen Methoden:

```
public boolean hasActivity(dvIdentificator id);
public Activity activity(dvIdentificator id);
public void removeActivity(dvIdentificator id);
public void moveActivityBefore(dvIdentificator toMove, dvIdentificator fix);
public void moveActivityAfter(dvIdentificator toMove, dvIdentificator fix);
```

Abbildung 36: ID-basierte tätigkeitsbezogene Methoden von `Docket`

Die Methode `hasActivity()` gibt darüber Auskunft, ob der Laufzettel eine Tätigkeit mit einer bestimmten ID auf seiner Liste hat. Wenn dies der Fall ist, kann man über die ID durch die Methode `activity()` eine technische Kopie der Tätigkeit erhalten. Mittels der Methode `removeActivity()` kann man bei Übergabe der ID eine Tätigkeit vom Laufzettel löschen (dies ist nur erlaubt, wenn die Tätigkeit noch nicht eingefroren ist). Das Löschen von Tätigkeiten ist nicht möglich, wenn der Laufzettel bereits veröffentlicht ist.

Mit Hilfe der Methoden `moveActivityBefore()` bzw. `moveActivityAfter()` kann die Reihenfolge der Tätigkeiten geändert werden. Es wird jeweils die ID der zu bewegenden Tätigkeit angegeben und die ID der Tätigkeit, vor bzw. hinter der diese eingefügt werden soll. Vorbedingung ist, dass niemals vor einer eingefrorenen Tätigkeit eingefügt bzw. niemals eine eingefrorene Tätigkeit bewegt werden darf.

Die entsprechenden Methoden für indexbasierten Zugriff sind:

```
public Activity activityAt(int pos);
public void removeActivityAt(int pos);
public void moveActivityBefore(int toMove, int fix);
public void moveActivityAfter(int toMove, int fix);
```

Abbildung 37: Indexbasierte tätigkeitsbezogene Methoden von `Docket`

Für einen gültigen Index kann durch Aufrufen der Methode `activityAt()` eine technische Kopie der Tätigkeit an diesem Listenindex erhalten. Entsprechend kann mit `removeActivityAt()` eine Tätigkeit gelöscht werden — hier gelten die gleichen Einschränkungen wie beim ID-basierten Löschen.

Die Methoden moveActivityBefore() und moveActivityAfter() funktionieren wie ihre ID-basierten Vorbilder, nur dass die betroffenen Tätigkeiten durch ihren Index ausgewählt werden.

Nun zu den Bearbeitungszustand-spezifischen Methoden:

```
public void setToPublished();
public boolean hasBeenPublished();
public Docket copyAsTemplate();
public void hasBeenCompleted();
```

Abbildung 38: Bearbeitungszustand-spezifische Methoden von Docket

Die Methode `setToPublished()` ermöglicht es, einen Laufzettel entsprechend des Zustands der Vorgangsmappe, an der der Laufzettel angebracht ist, als „veröffentlicht“ zu markieren. Diese Methode ist *ausschließlich dazu gedacht*, von der Vorgangsmappe gerufen zu werden, wenn an dieser gerade `setToPublished()` aufgerufen wurde. Mit `hasBeenPublished()` kann erfragt werden, ob der Laufzettel veröffentlicht ist - an einem veröffentlichten Laufzettel können keine Tätigkeiten mehr gelöscht, sie können nur durchgestrichen werden.

Die Methode `copyAsTemplate()` liefert eine Kopie des Laufzettels, die nicht veröffentlicht ist und nur nichteingefrorene Tätigkeiten auflistet, wie es für die Weiterverwendung als Vorlage sinnvoll ist. Die Methode `hasBeenCompleted()` schließlich stellt fest, ob alle auf dem Laufzettel aufgeführten Tätigkeiten bereits durchgeführt oder durchgestrichen sind, der Laufzettel also keine Tätigkeiten mehr aufweist, die zu erledigen wären (sie ist damit gleichbedeutend mit dem Prädikat `activitiesCount() == frozenActivitiesCount()`, s.u.) und damit als komplett angesehen werden kann.

Zum Abschluss betrachte ich die “allgemeinen” Operationen des Laufzettels:

```
public String generalNote();
public void appendToGeneralNote(String append);
public void setReceiver(dvUser rec, dvTransportAddress recAddress);
public boolean hasReceiver();
public dvUser receiver();
public dvTransportAddress receiverAddress();
```

Abbildung 39: Allgemeine Methoden

Die Methode `generalNote()` erlaubt es, die auf den ganzen Laufzettel bezogenen Notizen zu erfragen. Diesen kann mit der Methode `appendToGeneralNote()` eine neue Anmerkung hinzugefügt werden.

Mit der Methode `hasReceiver()` kann festgestellt werden, ob am Laufzettel ein Empfänger eingetragen ist. Ist dies der Fall, kann mit der Methode `receiver()` dieser erfragt werden. Mit der Methode `receiverAddress()` kann die zugeordnete Adresse in Erfahrung gebracht werden. Beides wird mit der Methode `setReceiver()` gesetzt. Dabei ist die Voraussetzung, dass der Empfänger für die nächste zu erledigende Tätigkeit zulässig ist, die über `currentActivity()` (s.u.) erfragt werden kann. Es liegt in der Verantwortung des Aufrufenden (in der Regel ein Werkzeug), die korrekte Adresse gemeinsam mit dem Empfänger zu setzen, da die Zuordnung von Benutzern/Rollen zu Adressen, etwa durch den Zugriff auf ein Adressbuch, nicht in einem (konzeptuell passiven) Material stattfinden sollte (das Laufzettelwerkzeug bedient sich hierzu einer Software-Komponente, die vom Anwendungsentwickler bereitgestellt werden muss, siehe 5.1.10 ab Seite 78)

5.1.11.4 Exkurs: Werkzeugkomposition in JWAM

Die Werkzeugkonstruktion nach WAM basiert auf zwei wesentlichen Mustern: Der Trennung von Funktion und Interaktion und der Werkzeugkomposition aus Subwerkzeugen (beschrieben in [Zül98] oder [Rie97], kurz zusammengefasst in Abschnitt 2.1.4 ab Seite 17).

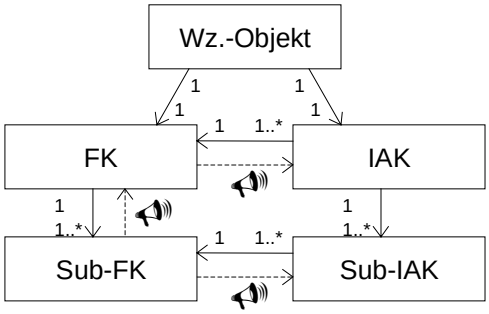


Abbildung 40: JWAM-Werkzeugkonstruktion bisher

Abbildung 40 zeigt die Werkzeugkonstruktion, wie sie von JWAM 1.4 unterstützt wird: Ein Werkzeug besteht aus Interaktionskomponente (IAK) und Funktionskomponente (FK). Die IAK kennt die FK, deren Operationen sie ruft, um Benutzereingaben in Handlungen umzusetzen. Die FK aber kennt umgekehrt die IAK nicht und kann sie nur mittels der losen Kopplung über den Ereignismechanismus von Materialänderungen unterrichten. Die Verwendung von Subwerkzeugen funktioniert auf den Ebenen Funktion und Interaktion parallel: Die Kontext-FK kennt die verwendete Sub-FK, die Kontext-IAK die verwendete Sub-IAK.

Diese Konstruktionsweise hat sich für die statische Einbindung von Subwerkzeugen grundsätzlich bewährt. Es hat sich jedoch gezeigt, dass auf die Verknüpfung von Kontext-IAK und Sub-IAK verzichtet werden kann, denn bei der Einbindung eines Subwerkzeugs spielt für das Kontextwerkzeug die durch das Subwerkzeug angebotene Funktionalität die entscheidende Rolle, während die Schnittstelle der IAK nicht von Interesse ist. Der entscheidende Beitrag der IAK liegt in der Präsentation gegenüber dem Benutzer und nicht in ihrer technischen Schnittstelle. Die Kenntniss des Typs der Sub-IAK durch die Kontext-IAK stellt somit eine unnötige Versteifung des Entwurfs dar. Aus dieser Erkenntnis hat sich für die JWAM 1.5 folgendes Modell entwickelt (s. [FÖ00]):

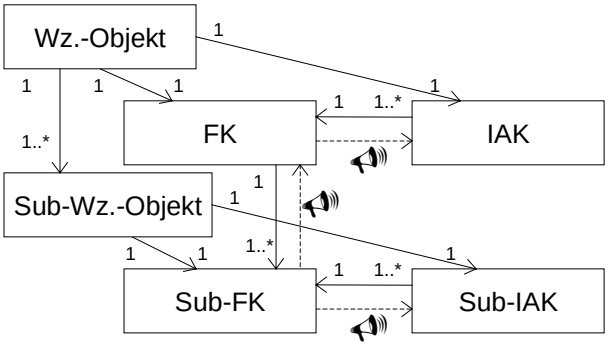


Abbildung 41: Die neue JWAM-Werkzeugkonstruktion

Hier ist zu sehen, dass auch Sub-FK und Sub-IAK unter einem Werkzeugobjekt zusammengefasst sind[72]. Damit gibt es (abgesehen von der Rollenverteilung) in der Konstruktion keine Unterschied mehr zwischen Kontextwerkzeug und Subwerkzeug. Robert Beeger liefert mit

[72] Es ist es in der Framework-Version 1.5 auch möglich, auf die FK-IAK-Trennung zu verzichten.

seiner Studienarbeit [Be00] die Grundlage, um bei der Einbindung von Subwerkzeugen deren GUI und Menüeinträge dynamisch in das Kontextwerkzeug einzubetten zu können. Die Konsequenz:

- Dasselbe Werkzeuge kann „Stand alone“ laufen oder als Subwerkzeug eingebunden werden
- Die Werkzeugkomposition kann in einfacher Weise polymorph erfolgen
- Durch die Verwendung von Fabriken (s. Fabrikmuster in [GHJ+]) ist es möglich, die konkret einzubindenden Subwerkzeuge erst zur Laufzeit festzulegen.

Bei den folgenden Schilderungen meines Entwurfs werde ich gelegentlich auf die sich hierdurch eröffnenden Möglichkeiten eingehen.

5.1.11.5 Die Materialklasse ProcessFolderAdapter

Die Klasse `ProcessFolderAdapter`, die ich als Implementation der Schnittstelle `ProcessFolder` anbiete, habe ich mit Blick auf die Entwicklung der Werkzeugkonstruktion im JWAM-Rahmenwerk (wie im vorigen Abschnitt beschrieben) entworfen.

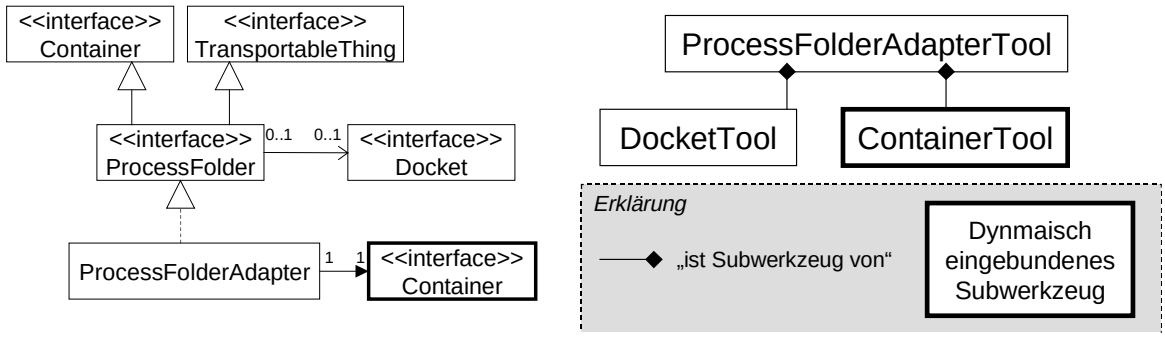


Abbildung 42: Die Klasse `ProcessFolderAdapter` (Klassendiagramm), Komposition eines passenden Werkzeuges

Es handelt sich beim ProcessFolderAdapter um einen Adapter im Sinne von [GHJ+], der einen beliebigen JWAM-Container „umwickelt“ und zu einer Vorgangsmappe macht[73].

Nach der skizzierten Umstellung der JWAM-Werkzeugkonstruktion bietet dies die Möglichkeit, ein Werkzeug zu konstruieren, das auf Materialien des Typs ProcessFolderAdapter arbeitet und dabei dem Vorgangsmappen-Werkzeug gleicht, in der Registerseite für den Mappeninhalt aber dynamisch ein passendes Subwerkzeug für den konkreten umwickelten Behälter einbindet.

Damit böte sich dem Anwendungsentwickler eine einfache Möglichkeit, spezielle, von ihm konstruierte Behälter[74] zu Vorgangsmappen zu machen (dies allein ist auch jetzt schon möglich), wobei ein von ihm dafür konstruiertes Behälterwerkzeug ohne weiteren Implementationsaufwand in das Vorgangsmappenwerkzeug integriert werden könnte. Der `ProcessFolderAdapter` stellt

[73] Alle Methoden aus dem JWAM-Interface `de.jwam.handling.containerconstruction.Container` werden dabei an den umwickelten Container delegiert, alle anderen Methoden sind in `ProcessFolderAdapter` selbst implementiert.

[74] Ein spezieller Behälter kann beispielsweise stets einen Satz ganz bestimmter Dokumente enthalten oder eine bestimmte innere Ordnung haben.

somit eine Hot Spot (vgl. Abschnitt 2.1.5 ab Seite 19) dar, der durch vom Anwendungsentwickler erstellte Klassen auf Material- und Werkzeugebene konfiguriert werden kann.

Zusammenfassung und Ausblick

Mit dem Ende dieses Kapitels ist auch sein zu Anfang definiertes Ziel erreicht: Ich habe einen softwaretechnischer Entwurf für die JWAM-Komponente „Vorgangsmappen“ erarbeitet und implementiert. Auf dem Weg dahin habe ich Anregungen aus CSCW-Systemen, denen (dem Prozessmuster) verwandte Konzepte zugrundeliegen, aufgegriffen und ein Benutzungsmodell formuliert, das direkt in die Werkzeug- und Materialkonstruktion eingeflossen ist.

Für die konstruierten Materialien ergibt sich folgendes Gesamtbild:

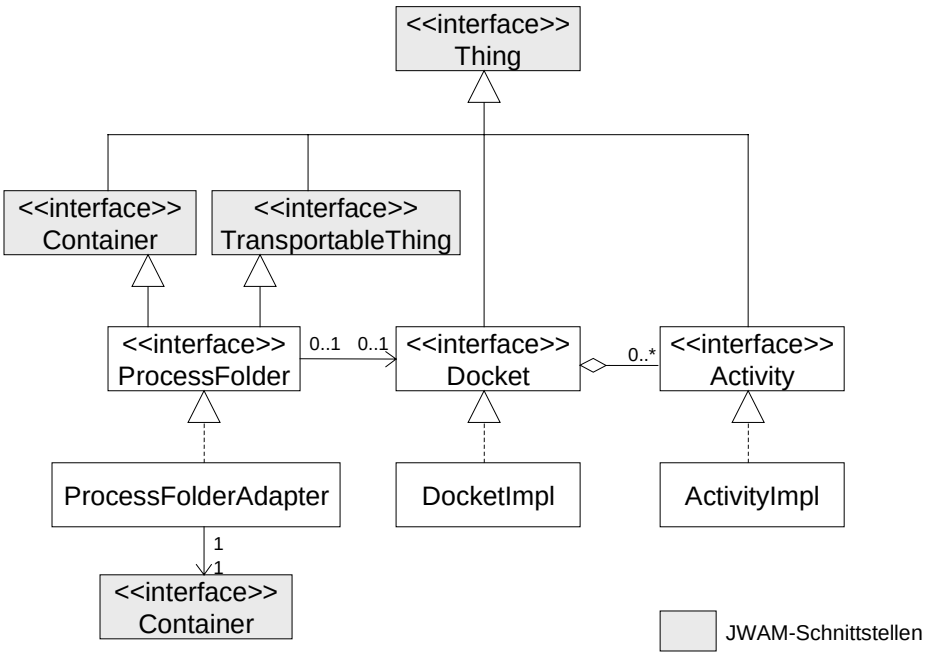


Abbildung 43: Überblick über die konstruierten Materialien (Klassendiagramm)

Diese Abbildung fügt die unterschiedlichen Einzeldiagramme aus diesem Kapitel zusammen (und außerdem die von mir bereitgestellte konkrete Implementation von `Activity`, `ActivityImpl`, hinzu). Dass alle konkreten Klassen (`ProcessFolderAdapter, DocketImpl, ActivityImpl`) von `ThingImpl` erben (um `Thing` zu implementieren), ist der Übersichtlichkeit halber nicht dargestellt.

Die Framework-Komponente, deren Entwurf ich in diesem Kapitel beschrieben habe, stellt mit Vorgangsmappe und Laufzettel hochflexible Kooperationsmittel bereit, die zur Laufzeit für verschiedene Arbeitsprozesstypen konfiguriert und in Form von Vorlagen wiederverwendet werden können. Die Konfigurationsmöglichkeiten umfassen sowohl die generische Beschreibung typischer Prozessverlaufs durch eine Liste von Tätigkeitsbeschreibungen auf dem Laufzettel als auch eine Grundausstattung der Vorgangsmappe mit bestimmten Materialien. Damit sind diese Bestandteile in Art eines Black-Box-Frameworks für den Anwendungsentwickler verwendbar.

Für die Erleichterung der Anpassbarkeit zur Konstruktionszeit habe ich insbesondere eine Vorgangsmappen-Implementation als Dekorierer vorgestellt, die einen beliebigen speziellen

Behälter dekoriert[75]. Mit der neuen JWAM-Werkzeugkonstruktion muss bei Vorhandensein eines entsprechenden Behälterwerkzeugs nicht einmal ein spezielles Vorgangsmappenwerkzeug konstruiert werden, das Behälterwerkzeug wird einfach in das Vorgangsmappenwerkzeug integriert. Damit stellt dieser konkrete Dekorierer einen klassischen Hot Spot (auf Werkzeug- und Materialebene!) dar, der durch "Einklinken" der vom Anwendungsentwickler entwickelten Klassen konfiguriert werden kann[76].

Indem das Wissen über die Strukturen der Organisation, in der die Vorgangsmappen eingesetzt werden, an definierte Software-Komponenten delegiert wird, für die spezielle "Einschübe" vorgesehen sind, ist kann Framework-Komponente in sehr unterschiedlich strukturierten Organisationen gleichermaßen gut eingesetzt werden..

Alle Materialschnittstellen wurden als Java-Interfaces und nicht als Klassen definiert, was die Implementationsfreiheit für den Anwendungsentwickler, der eigene Vorgangsmappen und Laufzettel bereitstellen will, sehr erhöht.

In ihrer Funktion beschrieben, jedoch nicht im Detail dargestellt, wurden der Vorlagenordner (und das zugehörige Werkzeug), in dem die Vorgangsmappen- und Laufzettelvorlagen aufbewahrt werden, sowie der Vorlagenersteller, der aus einem benutzten Laufzettel eine Vorlage erstellen kann. Die zugehörigen Klassen wurden von mir nur in minimaler Form implementiert, so dass ich für die Darstellung in diesem ohnehin langen Kapitel keinen Raum verwenden wollte.

In dieser Arbeit bisher nicht angesprochen wurden jene Fragen, die unmittelbar ins nächste Kapitel überleiten: Wie kann man sicherstellen, dass ein Vorgang nicht irgendwo liegenbleibt und verlorengeht? Wie kann man herausfinden, bei wem sich eine bestimmte Vorgangsmappe gerade befindet?

Es geht mithin um die Verfolgung einmal gestarteter Vorgänge, die ein Bearbeiter nicht mit der dazugehörigen Vorgangsmappe aus den Augen verlieren möchte. Welche Anforderungen an eine solche Vorgangsverfolgung zu stellen sind und wie das mit der bisherigen Konstruktion zusammenpasst, genau damit beschäftigen sich die folgenden Kapitel.

---

[75] ...wenn er die JWAM-Behälter-Schnittstelle erfüllt.

[76] Wenn zum JWAM-Framework weitere Behälter mit Standardwerkzeugen hinzukommen, erlaubt dies natürlich auch eine Black-Box-Verwendung dieses Hot Spots

6 Verfolgung von Vorgängen

Mit den vorangegangenen Kapiteln ist die Basis einer Framework-Komponente für JWAM, die Vorgangsmappen, Laufzettel und passende Werkzeuge bereitstellt, vorhanden. Dabei wurde bisher ausführlich erörtert, was Benutzer mit Vorgangsmappen tun, wenn diese in ihrem Postkorb bzw. auf ihrem Schreibtisch landen.

Die Frage, wie sich ein Anwender über den Bearbeitungsstand eines bestimten Vorgangs informieren oder sich einen Gesamtüberblick über die laufenden Vorgänge verschaffen kann, wurde noch nicht aufgeworfen. In diesem Kapitel geht es deshalb um die *Verfolgung von Vorgängen*, die dem Anwender ermöglicht werden soll. Es sollen konzeptionelle Vorarbeiten geleistet werden, welche die Grundlage für eine konstruktive Umsetzung bilden sollen, die im nächsten Kapitel dargestellt wird.

Motivation für die Vorgangsverfolgung

Mit Vorgangsmappen und Laufzetteln stehen Kooperationsmittel bereit, die geeignet sind, arbeitsteilige kooperative Prozesse zu unterstützen. Sie dienen dazu, Arbeitsmaterialien und Prozessbeschreibung mit jedem Arbeitsschritt von Bearbeiter zu Bearbeiter zu schicken und setzen das Prinzip der Einheit von Ort und Zeit um. Dies bedeutet, dass trotz der weitergehenden technischen Möglichkeiten bei der Konstruktion elektronifizierter Vorgangsmappen eine Mappe (und ihr Inhalt) zu einem Zeitpunkt konzeptionell nur an einem Ort befindlich sein kann. Eine Vorgangsmappe, die zu einem bestimmten Bearbeiter geschickt wurde, steht diesem also exklusiv zur Verfügung.

Dennoch können andere Mitarbeiter ein berechtigtes Interesse haben, sich über den Bearbeitungsfortschritt eines bestimmten Vorganges zu informieren, dessen zugehörige Mappe sich nicht auf ihrem Arbeitsplatz befindet, oder aber sie möchten einen Vorgang erst anhand bestimmter Kriterien auffinden.

Im Folgenden sollen beispielhaft einige Situationen beschrieben werden, in denen Anwender auf verschiedene Weisen an solchen Informationen interessiert sind. Die Entwicklung einer Framework-Komponente wirft dabei im Gegensatz zur Entwicklung einer konkreten Anwendung an dieser Stelle das Problem auf, dass eben kein konkreter Einsatzkontext vorliegt und keine Anwender, mit denen man Interviews führen könnte, so dass die Anforderungsanalyse zwangsläufig etwas unscharf bleiben muss. Ich habe mich jedoch bemüht, Beispiele auszuwählen, die etwa im Kontext des Kreditvergabebeispiels (dargestellt in Abschnitt 4.1.2 ab Seite 44) plausibel sind, das ja bereits von Gryczan (in [Gry96]) bei der Entwicklung des Prozessmusterkonzepts herangezogen wurde:

1. Ein Kunde möchte über den Stand eines ihn betreffenden Vorgangs informiert werden (etwa, wie sein Kreditwunsch beurteilt wird und wann er definitive Nachricht erhalten wird) und wendet sich an einen Ansprechpartner (sein persönlicher Berater oder z.B. ein Call-Center-Mitarbeiter). Dieser Ansprechpartner muss in der Lage sein, sich über den Vorgang zu informieren (wo befindet sich die Vorgangsmappe, wie lange ist sie dort schon, welche Tätigkeiten sind bereits ausgeführt worden), um dem Kunden Auskunft erteilen zu können.
2. Ein Bearbeiter, der eine besondere Verantwortung für einen bestimmten Vorgang trägt, möchte bei jedem „Anfassen“ des Vorgangs (Weiterschicken der Vorgangsmappe, Einfügen eines Dokuments etc.) informiert werden. Gegebenenfalls nimmt er bei Erhalt einer entsprechenden Benachrichtigung die Vorgangsmappe in Augenschein.

3. Einem Bearbeiter, der eine Vorgangsmappe am vorigen Tag weitergeschickt hat, wird klar, dass er darin einen Eintrag zu machen vergessen hat. Er muss den Vorgang über die Eingabe gewisser Kriterien auffinden können, um dem momentanen Bearbeiter eine entsprechende Nachricht zu schicken.

4. Um unzumutbar lange Bearbeitungszeiten zu verhindern, soll ein Verantwortlicher benachrichtigt werden, wenn Vorgänge (eines bestimmten Typs) länger als eine bestimmte Zeit „leben“ oder nicht bearbeitet werden, oder bei demselben Bearbeiter verweilen (Alarmfunktion).

Die Beispiele zeigen die Bandbreite der gewünschten Informationen und auch der verschiedenen Formen, in denen diese zum Benutzer gelangen sollen. Der erste Fall, in dem ein Kunde über den aktuellen Stand eines ihn betreffenden Vorgangs informiert werden will, kommt der Kapitelüberschrift *Verfolgung von Vorgängen* wörtlich am nächsten: Der Verlauf eines bestimmten Vorgangs soll nachverfolgt, nachvollzogen werden. Um dies zu ermöglichen, muss der Vorgangsverlauf gut dokumentiert werden.[77]

Der zweite Fall, wo ein Bearbeiter bei jedem „Anfassen“ eines bestimmten Vorgangs informiert werden will, ist dem ersten in der Hinsicht ähnlich, dass auch hier ein einzelner Vorgang verfolgt werden soll. Der entscheidende Unterschied ist, dass im ersten Fall der Interessierte selbst aktiv wird und sich die Informationen beschafft, während im zweiten Fall ein einmalig angemeldetes Interesse genügt, um „auf dem Laufenden“ gehalten zu werden. Diese Art von Benachrichtigung (synchron und ohne dass der Benutzer sich aktiv erkundigen muss) wird häufig mit dem Schlagwort *Awareness* bezeichnet: Der Begriff *Awareness* entstammt der CSCW-Forschung und beschreibt das Bewusstmachen der Tätigkeiten anderer Benutzer gegenüber einem Anwender, der ein CSCW-System einsetzt.[78] Der Begriff wird meist im Zusammenhang mit *Shared Workspaces* und synchronen CSCW-Anwendungen verwendet (kann aber auch auf den asynchronen Fall angewendet werden[79]).

Der dritte als Beispiel genannte Fall, bei dem ein Benutzer die Menge der Vorgänge anhand bestimmter Kriterien durchsuchen können soll, ist nur eine von vielen denkbaren Situationen, in denen diese Fähigkeit gefordert ist. Je flexibler und differenzierter die Suchmöglichkeiten gestaltet sind, um so nützlicher können sie auch in Anwendungsfällen sein, die bei der Entwicklung der Suchfunktionalität noch gar nicht vorgedacht waren.

Der letzte Fall, der eine Art von Alarm-Funktion umsetzt, ermöglicht wie der zweite, dass der Benutzer nach der Bekundung von Interesse passiv auf die entsprechenden Benachrichtigungen wartet. Jedoch bezieht sich das Interesse hier nicht auf einen einzelnen Vorgang, sondern auf eine Menge von Vorgängen.

Die dargestellten Fälle lassen zwei Dimensionen der Vorgangsverfolgung erkennen:

- *aktiv/passiv:* Entweder muss der Benutzer jeweils aktiv werden, um an die Vorgangsinformationen zu gelangen, oder aber er meldet einmal ein Interesse an, bleibt von nun an passiv und wird ereignisbezogen informiert

---

[77] Man findet in diesem Zusammenhang auch häufig den Begriff der *Historie* oder *Historisierung*. Damit ist gemeint, die „Geschichte“ eines Gegenstandes, also alle Änderungen an ihm, zu speichern.

[78] Dourish etwa definiert *Group Awareness* als *„understanding of the activities of others, which provides a context for your own activity“* ([DB92])

[79] Dourish merkt dazu an: *„In fact, I think the distinction between synchronous and asynchronous systems is essentially spurious, and simply an artifact of the architectural approaches we have become accustomed to using. Real work moves freeling between different degrees of synchrony and coordination“* ([Dou97])

- *spezieller Vorgang/Menge von Vorgängen:* Der Benutzer möchte entweder Informationen zu einem bestimmten Vorgang, oder er interessiert sich für eine Menge von Vorgängen, typischerweise für alle Vorgänge eines bestimmten Typs. Der letztere Fall ist insbesondere von Interesse, wenn aus Beobachtung und Auswertung von vielen Vorgängen eines Typs Schlüsse über die Umgestaltung dieses Vorgangstyps (also eine Anpassung der Vorlage) getroffen werden sollen.

Gefordert ist außerdem die Fähigkeit, nach Vorgängen anhand bestimmter Kriterien zu suchen.

Vorgangsverfolgung bei CSCW-Systemen mit Vorgangsmappen

In dem für diese Arbeit grundlegenden Text von Gryczan findet sich zum Thema Vorgangsverfolgung nur ein recht kurzer Abschnitt zur "Lokalisierung von Vorgangsmappen" ([Gry96], ab Seite 202). Dort wird ein sogenanntes "Info-Werkzeug" vorgestellt. Dieses ermöglicht einem Benutzer unter den Vorgangsmappen (an denen er bereits gearbeitet hat oder für die er als Bearbeiter in der Zukunft zugeordnet ist) anhand der Mappenbenennung oder des Mappen-Erzeugers zu suchen. Damit wird ein denkbarer Weg dargestellt, die Verfolgung von Vorgängen zu ermöglichen.

In Kapitel 5 habe ich zu Beginn CSCW-Systeme betrachtet, die kooperative Arbeitsprozesse durch Vorgangs- oder Umlaufmappen unterstützen, und für meinen Entwurf zahlreiche Anregungen erhalten und aufgegriffen.

Es ist naheliegend, genau diese Anwendungssysteme nun auch in Bezug darauf zu untersuchen, wie bei Ihnen die Vorgangsverfolgung und Suchfunktionen realisiert sind.

6.1.1 Geschäftsfallmappen suchen beim UBS Corporate Desktop

Der UBS Corporate Desktop bietet dem Benutzer die Möglichkeit, nach Geschäftsfallmappen zu suchen. Ein Werkzeug mit dem Namen *Geschäftsfallmappen-Sucher* kann zu diesem Zweck aus der Werkzeugkiste des Corporate Desktop heraus gestartet werden.

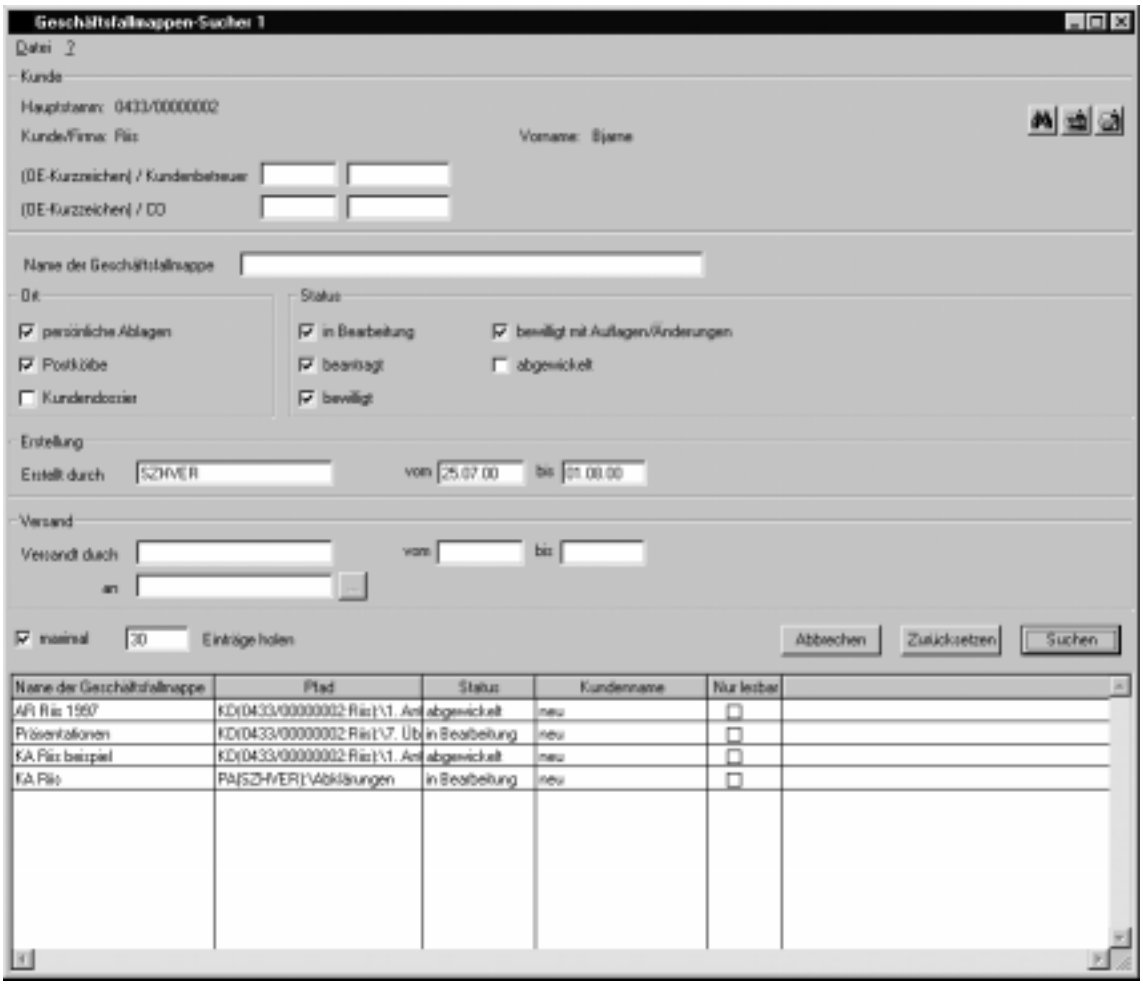


Abbildung 44: Der Geschäftsfallmappen Sucher des UBS Corporate Desktop

Im oberen Teil des Geschäftsfallmappen-Suchers können die Suchkriterien eingegeben werden. Für eine erfolgreiche Suche ist dabei ein Mindestsatz an spezifizierten Kriterien (Kunde, Ersteller, Erstellungszeitraum, Absender, Versandzeitraum) von Vorteil. Nach Eingabe der Suchkriterien kann die Suche mit dem *Suchen*-Knopf gestartet werden.

Die Geschäftsfallmappen, die den Suchkriterien entsprechen, werden im unteren Teil des Fensters in Form einer Liste ausgegeben. Sie können durch den Bearbeiter angezeigt, kopiert und verschoben werden, wenn er die dazu jeweils nötigen Rechte besitzt.

6.1.2 POLIAwac – der Awareness-Client von POLITeam

Wie bereits in Abschnitt 5.1.2 ab Seite 56 dargestellt, erfolgt in POLITeam die Kooperation über Umlaufmappen *(Electronic Circulation Folders)* und gemeinsame Arbeitsbereiche. Für das gesamte System, insbesondere also *beide* Formen der Kooperation, wurde ein „Awareness-Client“ mit dem Namen *PoliAwac* konstruiert, der in [SPF96] vorgestellt und diskutiert wird.

Ein Anwender des POLITeam-Systems kann die Auswirkungen der Handlungen anderer Systemteilnehmer über mehrere Mechanisman wahrnehmen:

- *Symbolische Statusdarstellung:* In POLIAwac sind Objektsymbole allgegenwärtig. Diese Symbole werden auch verwendet, um Gruppenwahrnehmungsinformationen darzustellen. So werden Objekte, die durch einen bestimmten Benutzer modifiziert worden sind, durch die diesem Benutzer zugeordnete Farbe eingefärbt, wobei diese Einfärbung mit der Zeit

zurückgeht. Die Art der Modifikation (Deleted, Moved, Edited...) wird durch ein zweites, überlagertes Symbol angedeutet.

- *Ereignisleiste:* Die Ereignisleiste besteht aus einer Dropdowntextliste und einer Reihe von Schaltflächen. In der Liste wird eine Historie der den Benutzer interessierenden Ereignisse angezeigt, wobei die Reihenfolge chronologisch ist. Die Ereignisse werden durch das Symbol des betroffenen Objektes (s.o.) sowie eine textuelle Beschreibung repräsentiert. Durch die Benutzung der zusätzlichen Schaltflächen kann der Anwender auch ausdrückliche Nachrichten zu einem von ihm ausgewählten Objekt versenden, die dann als Ereignisse für andere Anwender sichtbar werden.
- *Symbolvergrößerung:* Zusätzlich zu den zuvor beschriebenen Symbolveränderungen wird auch die Symbolgröße verändert. Symbole werden hierbei auf bis zu 200 Prozent ihrer ursprünglichen Fläche vergrößert, bei baumartigen Anzeigen wird diese Vergrößerung zur Wurzel hin propagiert, wenn das betroffene Symbol nicht selbst sichtbar ist.
- *Pop-Up-Dialog:* Ereignisse mit höchster Priorität werden in einem Pop-Up-Dialog angezeigt.

Dabei kann der Anwender durch die Beschreibung eines sogenannten „Interessensprofils“ frei wählen, über welche Ereignisse er in welcher der oben angegebenen Formen informiert werden möchte:

„Ein Interessensprofil beschreibt für eine Menge von Objekten, über welche Operationen man in welcher Arbeitssituation informiert werden möchte (z.B immer dann, wenn ein Behälterobjekt geöffnet wird, wird man über Änderungen an darin enthaltenen Objekten informiert).“ ([SPF96], in Abschnitt 3.1)

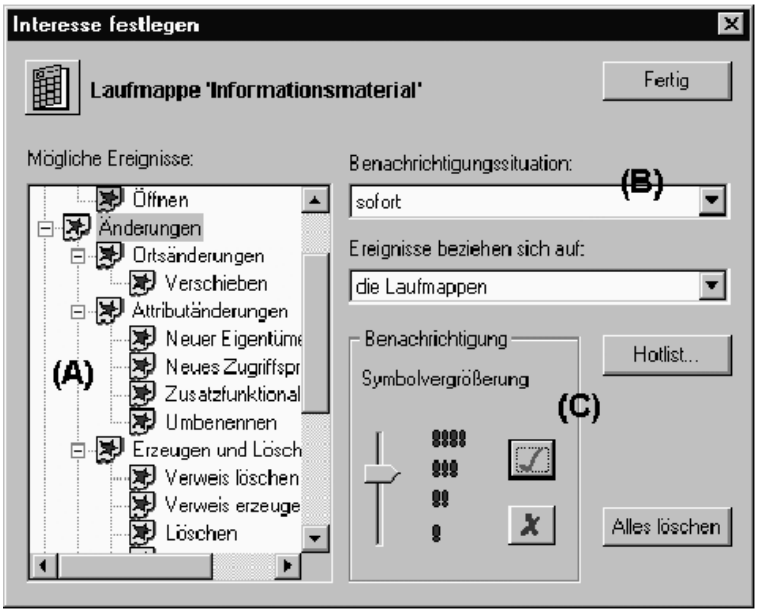


Abbildung 45: Editor für ein Interessensprofil bei POLIAwac

Die obige Abbildung zeigt den Editor für das Interessensprofil im Bezug auf eine bestimmte Laufmappe mit der Bezeichnung “Informationsmaterial”. Im linken Bereich (A) ist eine hierarchische Auflistung der Ereignisse zu sehen, die im Zusammenhang mit der Laufmappe auftreten können. Zu jedem Ereignis kann, das links ausgewählt wird, können auf der rechten Seite Einstellungen vorgenommen werden.

Im rechten Bereich kann (bei (B)) die Situation ausgewählt werden, in der Benachrichtigungen stattfinden sollen (der Anwender kann sofort zum Zeitpunkt der Modifikation informiert werden, oder aber erst dann, wenn er selbst auf ein Objekt zugreift, das sich auf denselben Vorgang bezieht wie die Laufmappe). Schließlich kann in Bereich (C) ausgewählt werden, wie die Benachrichtigung erfolgen soll. Hierfür sind vier Intensitätsstufen vorgesehen (gekennzeichnet mit ein bis vier Ausrufungszeichen), die die Bedeutung 1) symbolische Statusdarstellung, 2) Anzeige in der Ereignisleiste, 3) Symbolvergrößerung und 4) Pop-Up-Dialog tragen, wobei die höheren Intensitätsstufen die Benachrichtigungsformen der niedrigeren jeweils enthalten. Außerdem kann die Benachrichtigung natürlich auch abgeschaltet werden.

6.1.3 Vorgangsverfolgung bei der RWG

Wie schon in Abschnitt 5.1.3.1 (“Vorgang und Vorgangsschrank”, ab Seite 58) dargestellt, werden die Vorgänge (und mit ihnen die Vorgangsmappen) beim Entwurf der RWG von Bearbeiter zu Bearbeiter verschickt, die öffentlichen (publizierten) Vorgänge haben ihren Platz im Vorgangsschrank, wo sie als ausgeliehen gekennzeichnet sind, während sie von Bearbeiter zu Bearbeiter wandern.

Der Vorgangsschrank enthält für jeden Vorgang einen sogenannten *Infozettel*, der über die im Vorgang enthaltenen Materialien und den Bearbeitungsstatus informiert. Für ausgeliehene Vorgänge enthält er statt des Infozettels einen *Ausleihzettel*, der dem Infozettel gleicht, zusätzlich aber noch Informationen zu dem Ausleihenden, dem Zeitpunkt des Ausleihens usw. enthält.

Bei jedem Versendevorgang wird der Ausleihzettel aktualisiert, so dass der Vorgangsschrank ausreichend Informationen enthält, um darin erfolgreich suchen zu können. Für diesen Zweck gibt es den Vorgangssucher.

Über den Vorgangssucher kann ein Benutzer publizierte Vorgänge suchen. Folgende Suchmöglichkeiten stehen zur Verfügung:

- Alle Vorgänge einer Person
- Alle offenen Vorgänge
- Alle Vorgänge eines bestimmten Typs
- Alle in einem bestimmten Zeitraum abgeschlossenen Vorgänge

Der Vorgangssucher sucht ausschließlich im Vorgangsschrank. Wird ein Vorgang gefunden, für den ein Ausleihzettel existiert, kann sich der Benutzer den Ausleihzettel zu diesem Vorgang anschauen und sich so über den Stand und Lokalität des Vorgangs informieren.

Die mir vorliegenden Entwurfsdokumente enthalten leider keine genaueren Angaben zu den Suchfunktionen und ihrer Umsetzung.

6.1.4 Zusammenfassung

Die betrachteten Systeme stellen recht unterschiedliche Werkzeuge zur Vorgangsverfolgung bereit, die sich alle in dem in Abschnitt 0 eingangs gesteckten Rahmen bewegen. So ist im UBS Corporate Desktop ausschließlich Suchfunktionalität vorgesehen. Wegen der im Abschnitt “Die Geschäftsfallmappe” (ab Seite 54) beschriebenen Registerseite “Bearbeitungsverlauf” (die sich in der Geschäftsfallmappe befindet) kann durch das Studium einer mit dem Geschäftsfallmappen-Sucher gefundenen Mappe auch der betreffende Vorgang nachverfolgt werden. Es ist dem

Anwender allerdings nicht möglich, sich für eine Mappe zu interessieren und über deren Werdegang fortlaufend informiert zu werden.

POLIAwac legt dagegen mit seinen Interessenprofilen und den verschiedenen Formen der Benachrichtigung ein starkes Gewicht auf die Präsentation der Awareness-Informationen. Leider ist ansonsten die Beschreibung der Vorgangsverfolgungs-Funktionen sehr knapp ausgefallen.

Der Entwurf der RWG wiederum sieht ein einfaches Suchwerkzeug vor, das auf dem Vorgangsschrank arbeitet. Hier muss wie beim UBS Corporate Desktop der Anwender jede Information aktiv anfordern.

Damit erlauben zwei der drei betrachteten Systeme den Anwendern, Mappen anhand fachlicher Merkmale aufzufinden und sie dann (beim RWG-Entwurf ggf. den zugehörigen Infozettel) anzusehen. Auch wenn dies im Fall von POLIAwac nicht explizit formuliert ist, gehe ich davon aus, dass alle drei betrachteten Systeme genausowie wie das von Gryczan beschriebene Info-Werkzeug die Lokalisierung einer dem Anwender bekannten Vorgangsmappe erlauben. Dies muss demnach als besonders wichtige Funktionalität festgehalten werden.

Augenfällig ist, dass für keines der betrachteten Systeme Werkzeuge beschrieben wurden, die dazu dienen, die Gestaltung von Vorgängen zu evaluieren, indem etwa über alle Vorgänge eines bestimmten Typs Durchschnitts- und andere Kennwerte gebildet würden. Dies liegt nach meinem Dafürhalten daran, dass die drei Systeme als CSCW-Anwendungen mit Blick auf kleine bis mittlere Arbeitsgruppen und einer starken Anwenderzentrierung anzusehen sind. Für solche Systeme ist typisch, dass sie den Anwendern (auch in Bezug auf die Prozessgestaltung) fachliche Kompetenz und weitgehende Entscheidungsfreiheit zumessen, weswegen die Fragestellung der Prozessoptimierung bzw. der Überwachung einer effizienten Prozessgestaltung auf Basis statistischer Daten nicht im Fokus der Anwendungen liegt.

Eine solche Funktionalität passt viel besser zur ablaufsteuernden Sichtweise, wie sie WfMS vertreten. Mit den dort vorgesehenen Mechanismen zur Vorgangsverfolgung befasst sich der folgende Abschnitt.

Arbeitsablauf-Monitoring und –Controlling bei WfMS

Einen kurzen Abriss zu Workflow-Management-Systemen habe ich bereits in Abschnitt 3.1.2 ab Seite 31 gegeben. Dort habe ich darauf hingewiesen, dass WfMS Ausprägungen der ablaufsteuernden Sichtweise sind. Der Gedanke der Steuerung und Kontrolle legt nahe, dass WfMS über Vorgangsverfolgungs-Funktionalität verfügen sollten. Diese wird aufgeteilt in Arbeitsablauf-Monitoring und Arbeitsablauf-Controlling.

6.1.5 Die Begriffe Arbeitsablauf-Monitoring und Arbeitsablauf-Controlling

Als Arbeitsablauf-Monitoring wird laut Jablonski ([JBS97], ab Seite 201) die Veranschaulichung von Arbeitsabläufen auf Basis der Laufzeitdaten eines WfMS verstanden.

Die Workflow Management Colation prägt demgegenüber den Begriff *Workflow Monitoring* als *“The ability to track and report on workflow events during workflow execution”*.

Durch das Arbeitsablauf-Monitoring ist es z.B. möglich, Kunden über den aktuellen Bearbeitungsfortschritt ihres Auftrags zu informieren. Es kann aber auch dazu dienen, nicht nur aktuelle Daten auszuwerten, sondern auch Annahmen über zukünftige Entwicklungen zu machen, um z.B. eine lastabhängige Rollenauflösung vorzunehmen.

Das Arbeitsablauf-Controlling dagegen verdichtet Istdaten zu Kennzahlen und stellt sie Solldaten gegenüber. Damit soll die Erkennung von Diskrepanzen zwischen Ablauf-Entwurf und -Realisierung erkannt werden und gegebenenfalls eine Arbeitsablauf-Reorganisation initiiert werden. Das Arbeitsablauf-Controlling liefert also unter anderem durchschnittliche, minimale und maximale Durchlaufzeiten eines Workflows und hilft dabei, Verzweigungen oder Regeln in Workflows anzupassen ("Wenn die Rechnungen von Lieferant A x-mal geprüft wurden und die Fehlerquote unter 1% lag, durchlaufen sie fortan den Pfad ohne manuelle Prüfung").

Jablonski betrachtet Arbeitsablauf-Monitoring und –Controlling in erster Linie als Teil des Workflow-Zyklus (Workflow Management Cycle, Workflow-Life-Cycle), eines phasenbasierten Vorgehensmodells zur Entwicklung von Informationssystemen: sie stellen für ihn ein Mittel der Evaluierung in diesem Vorgehensmodell dar.

Laut Jablonski ist der Realisierungsstand derzeitiger Workflow-Management-Systeme in Bezug auf Arbeitsablauf-Monitoring und besonders das Arbeitsablauf-Controlling noch nicht besonders weit fortgeschritten. Insbesondere sieht kaum ein WfMS die für das Controlling notwendige Pflege von Solldaten vor. Die Bereitstellung von Laufzeitdaten erfolgt meist durch sogenannte *Audit Trails*, das sind spezielle Logdateien[80]. Diese Audit Trails können über passende Werkzeuge ausgewertet und visualisiert werden.

Als Beispiel für die Realisierung von Arbeitsablauf-Monitoring und –Controlling bei einem speziellen WfMS betrachte ich nun das COSA-Workflow-System.

6.1.6 Realisierung in COSA Workflow

COSA Workflow ist ein Workflow Management System, dessen Workflowschemata auf Petri-Netzen beruhen. Der Bearbeitungsverlauf einer Workflowinstanz entspricht somit einer Abfolge von Schaltungen der Transitionen des zugrundeliegenden Netzes. Vorgangsverfolgung kann in COSA über zwei Tools erfolgen: den *Netzwerk-Statusanzeiger* und das *COADMenu*. Der Netzwerk-Statusanzeiger ist ein GUI-basiertes Werkzeug, das sich auch an die Anwender richtet, während COADMenu maskenbasiert ist und nur für Administratoren gedacht ist.

6.1.6.1 Der Netzwerk-Statusanzeiger

Der Netzwerk-Statusanzeiger erlaubt es einem Bearbeiter, eine bestimmte Workflowinstanz (bei COSA auch *Ablaufinstanz* genannt) vor dem Hintergrund ihrer Definition zu betrachten (wenn er die entsprechenden Zugriffsrechte besitzt): Das zugehörige Petri-Netz wird in der COSA-Darstellung abgebildet, in dem momentanen Zustand, d.h. mit einer bestimmten Anordnung der Marken auf den Stellen (Bedingungen). Eine mit Marken belegte Bedingung gilt als erfüllt und weist darauf hin, dass die folgende Tätigkeit (Transition) ausführbar ist bzw. geschaltet werden kann. Anhand der Position der einzelnen Token kann der Anwender den jeweiligen Bearbeitungszustand ablesen. Die folgende Abbildung zeigt die grafische Repräsentation des Netzes im Netzwerk-Statusanzeiger:

[80] Die Workflow Management Coalition definiert Audit Trails als "A historical record of the state transitions of a workflow process instance from start to completion or termination".

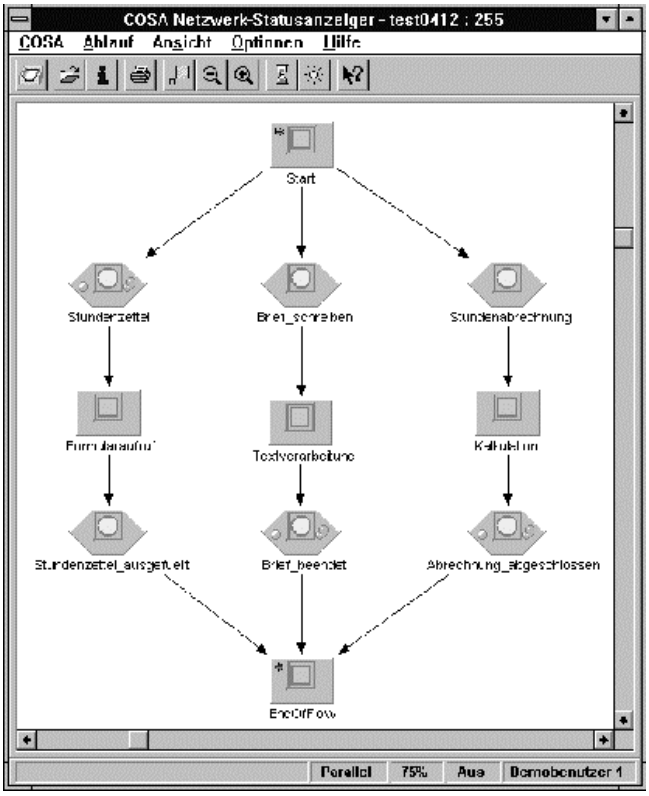


Abbildung 46: Der Netzwerk-Statusanzeiger von COSA Workflow

Die Auffindung eines bestimmten Prozesses (mit dem Ziel diesen dann im Netzwerk-Statusanzeiger zu betrachten) wird durch das Werkzeug nur sehr primitiv unterstützt: Alle vorhandenen Ablaufinstanzen können in einer Tabelle angezeigt und gefiltert werden, nach Vorgangsbezeichnung (ein Vorgang umfasst bei COSA Workflow mehrere zusammengehörige Abläufe, also mehrere Workflows) und Ablaufdefinition (Workflowdefinition) — das bedeutet, dass im Wesentlichen nur anhand eines Namens unter Ablaufinstanzen eines bestimmten Typs gesucht werden kann. Durch Auswahl aus der Tabelle kann die entsprechende Ablaufinstanz im Netzwerk-Statusanzeiger betrachtet werden, wie es aus der obigen Abbildung ersichtlich ist.

6.1.6.2 COADMenu

COADMenu ist das maskenbasierte Administrationstool zu COSA Workflow, das u.a. Systeminstallation, Zugriffsrechteverwaltung und Benutzerverwaltung ermöglicht. Im Hinblick auf Vorgangsverfolgung ist die Reportfunktionalität von COADMenu zu betrachten.

Die Reportfunktion erlaubt es, SQL-Abfragen auf den internen Datenbanktabellen auszuführen, in denen COSA Workflow Ablaufdefinitionen und die Ablaufinstanzen ablegt. Dabei werden die Ergebnismengen in Dateien ausgegeben. Auf diese Weise können Kennzahlen für Ablaufinstanzen, die auf bestimmten Ablaufdefinitionen beruhen, ermittelt werden.

COSA Workflow bietet mit den Funktionen in COADMenu jedoch keine Unterstützung für Arbeitsablauf-Controlling, wie es Jablonski definiert: Es gibt keine Eingabe- und Pflegemöglichkeit von Solldaten und in der Folge natürlich auch keine Unsterstützung, um auf der Basis eines Vergleichs von Ist- und Solldaten Ablaufdefinitionen statisch oder gar dynamisch abzuändern.

6.1.7 Zusammenfassung

Aus den abstrakten Betrachtungen und anhand des Beispieles COSA Workflow ist zu erkennen, dass der Begriff des Arbeitsablauf-Monitoring (je nach Interpretation) einen Teil der Vorgangsverfolgung abdecken könnte, die in den betrachteten CSCW-Systemen anzutreffen war. Die Funktionalität ist in der Literatur zu Workflow Management Systemen jedoch stärker mit Blick auf Workflow-Entwickler und Administratoren denn mit Blick auf die Anwender beschrieben. So umfasst denn bei COSA die dem Benutzer angebotene Funktionalität auch nur die Möglichkeit, nach einem bestimmten Vorgang ("Ablaufinstanz") über seinen Namen zu suchen und dann den Bearbeitungsfortschritt dieses Vorgangs in Form einer Petri-Netz-Darstellung zu betrachten.

Das Arbeitsablauf-Controlling dagegen, das auch in COSA Workflow nur in Ansätzen möglich ist (mit dem Zugriff auf die internen Tabellen ist eher eine technische Vorraussetzung erfüllt als eine Unterstützung vorhanden), verlangt bestimmte technische Fertigkeiten (bei COSA: Kenntnis der internen Tabellenstrukturen, SQL-Kenntnisse) und kann sich schon deshalb nur an entsprechend qualifizierte Administratoren wenden.

Zusammenfassung und Ausblick

In diesem Kapitel habe ich ausgehend von einer Reihe beispielhaft formulierter Situationen, in denen Anwender sich über Vorgänge/Vorgangsmappen, die im System "unterwegs" sind, informieren wollen, Anforderungen und Realisierungsmöglichkeiten für eine Vorgangsverfolgung beleuchtet.

Dabei habe ich anhand der Beispiele zunächst zwei Dimensionen der Verfolgung von Vorgängen identifiziert: Zum einen kann die Initiative bei der Verfolgung vom Anwender ausgehen (er erkundigt sich aktiv nach den gewünschten Informationen), oder aber sie liegt beim System (der Anwender bekundet einmalig sein Interesse an bestimmten Informationen und bekommt sie von nun an ohne eigenes Zutun angezeigt), zum anderen kann der Anwender den Wunsch haben, einen einzelnen oder aber eine Gruppe oder Klasse von Vorgängen zu betrachten. Neben diesen Dimensionen steht als wichtigste Anforderung die Durchsuchbarkeit der Vorgänge anhand bestimmter Merkmale.

Der Blick auf die bereits in Kapitel 5 betrachteten Systeme und auf die in WfMS umgesetzten Ansätze zeigt, dass die aus der unterstützenden Sichtweise gestalteten Anwendungen hauptsächlich Werkzeuge zum Durchsuchen der Menge der Vorgänge aufweisen und die Nachvollziehbarkeit dieser Vorgänge dann in Gestalt der Kooperationsmittel Vorgangsmappe (bzw. Laufzettel/Checkliste) selbst gegeben ist. Bei den WfMS sind die Ansätze zur Vorgangsverfolgung dagegen nicht anwenderzentriert, sondern wenden sich eher an die Administratoren der Workflow-Systeme. Dies zeigt sich beim konkret betrachteten WfMS COSA Workflow zum einen in einem Werkzeug mit sehr primitiver Suchfunktionalität für den Anwender und in der Tatsache, dass der ausgewählte Vorgang (die Ablaufinstanz) auf Basis einer Petri-Netz-Darstellung visualisiert wird. Dem steht ein nur von Fachleuten benutzbares, dafür aber umso mächtigeres Administrationswerkzeug mit vollem Zugriff auf die Repository-Datenbanktabellen gegenüber.

Aus Sicht des WAM-Ansatzes kann sich damit die zu gestaltende Vorgangsverfolgung allenfalls an den drei Ansätzen orientieren, die mit Vorgangmappen arbeiten. Hierbei sind die Lokalisierung von dem Anwender bekannten Vorgangsmappen sowie die Bereitstellung differenzierter Suchfunktionalität die wichtigsten Aspekte.

Im nächsten Kapitel, in dem ich die Konstruktion des Vorgangsmonitors beschreibe, der die Vorgangsverfolgung realisieren soll, wird es aber auch darum gehen müssen, das bisher formulierte Benutzungsmodell so zu erweitern, dass inbesondere das Konzept der Einheit von Ort und Zeit um ein Modell der Informationssammlung ergänzt wird, das dem Benutzer verdeutlicht, wann welche Informationen für den Vorgangsmonitor bereitgestellt werden.

7 Entwurf und Implementation eines Vorgangsmonitors

In diesem Kapitel beschreibe ich Entwurf und Implementation eines Vorgangsmonitors, der die technologische Basis für die Entwicklung von Werkzeugen für die Vorgangsverfolgung bereitstellen soll, deren Nutzen und mögliche Anforderungen im vorigen Kapitel erläutert wurden.

Die durch die Vorgangsverfolgung erreichbare Transparenz der Arbeitsprozesse und die einhergehende Kontrolle im konkreten Einsatz dürfte Arbeitnehmerrechte berühren und ganz allgemein den sensiblen Bereich Datenschutz. Dieses Thema wird von mir in dieser Arbeit aber nicht behandelt.

Dies hat zwei Gründe: Zum einen ist der von mir beschriebene Vorgangsmonitor nicht in der Handhabungs- und Präsentationsschicht des JWAM-Frameworks anzusiedeln, sondern in der Technologieschicht. Der Einsatz eines mächtigen Automaten auf der technologischen Ebene sagt aber noch nichts darüber aus, welche Funktionen im System tatsächlich genutzt und über Werkzeuge den Benutzern zur Verfügung gestellt werden. Der Zugang zu Funktionen mag eingeschränkt sein, etwa nur bestimmten Benutzern oder verschiedenen Benutzern nur in bestimmtem Abstufungen zur Verfügung stehen.

Genauere Vorstellungen werden von mir zu diesem Thema aber auch deshalb nicht formuliert, weil JWAM zur Zeit im Rahmen einer Diplomarbeit um ein auf Schlössern und Schlüsseln basierendes Sicherheitskonzept erweitert wird (s. [Bei00]). Eine Konfiguration des Zugangs zum Vorgangsmonitor müsste auf diesem noch nicht fertig formulierten Konzept beruhen und kann auch deshalb nicht hier beschrieben werden.

Interessierende Merkmale von Vorgängen

In diesem Abschnitt möchte ich die Merkmale eines Vorgangs zusammentragen und lokalisieren, die für die Vorgangsverfolgung von Interesse sind. Mit Blick auf die in Kapitel 6 geschilderten Beispiele lassen sich zunächst drei wichtige Informationsbedürfnisse beschreiben:

1) Untersuchung des Bearbeitungsfortschritts eines bestimmten Vorgangs

2) Erkennen der Überschreitung bestimmter zeitlicher Restriktionen für einen oder mehrere Vorgänge

3) Filterung der Vorgänge nach bestimmten Kriterien.

Das erste genannte Informationsbedürfnis kann relativ leicht befriedigt werden, indem eine Kopie (eine möglichst neue, aber eben nicht wirklich aktuelle Momentaufnahme, ein *Snapshot*) der zugehörigen Vorgangsmappe an den interessierten Bearbeiter ausgehändigt wird, weil Vorgangsmappe und Laufzettel Kooperationsmittel sind, die den Vorgangsverlauf außerordentlich gut dokumentieren.

Da von einem bestimmten Vorgang die Rede ist und der Vorgangsmonitor ein Automat sein soll, der als Basis z.B. für Suchwerkzeuge fungiert, gehe ich davon aus, dass dem Vorgangsmonitor die ID der Vorgangsmappe übergeben wird und er dafür die passende Mappenkopie zurückliefern muss.

Das zweite Informationsbedürfnis erfordert ein Durchsuchen der im Umlauf befindlichen Vorgangsmappen anhand eines *generischen Merkmals* (wie in diesem Fall z. B. der letzte Versendezeitpunkt). Als generische Merkmale bezeichne ich diejenigen Merkmale, die sich

direkt über die Schnittstelle der Vorgangsmappe, des Laufzettels oder dessen Tätigkeiten auslesen lassen.

Das dritte formulierte Informationsbedürfnis erfordert ein Durchsuchen der im Umlauf befindlichen Vorgangsmappen entweder anhand *generischer* oder aber *fachlicher Merkmale.* Unter einem fachlichen Merkmal verstehe ich ein Merkmal eines Vorgangs, das eben nicht generisch ist, z.B. die Höhe eines Kreditantrags (bei einem Kreditbearbeitungsvorgang).

Für die Befriedigung der im letzten Kapitel formulierten Informationsbefürfnisse muss der Vorgangsmonitor also

1) zu der ID einer Vorgangsmappe einen möglichst aktuellen Snapshot dieser Mappe zurückliefern können

2) die im Umlauf befindlichen Vorgangsmappen nach *generischen* Merkmalen filtern können

3) die im Umlauf befindlichen Vorgangsmappen nach *fachlichen* Merkmalen filtern können

Beispiele für generische Merkmale:

- der Bearbeitungszustand der einzelnen Tätigkeiten (abgehakt/durchgestrichen/noch unbearbeitet) — diese Information kann an den Tätigkeiten des Laufzettels erfragt werden
- der Bearbeiter, der die Mappe in seinem Besitz hat — dieser kann auch von dem Laufzettel abgelesen werden, *aber nur zum Zeitpunkt des Versands mit dem Transportsystem* (denn die Adressierung sagt ja nur im Moment des Versands etwas über den Ort der Mappe danach aus)
- das letzte Versendedatum der Mappe — dies kann an der Vorgangsmappe ausgelesen werden, über die Schnittstelle `TransportableThing`, die sie implementiert.

Beispiele für fachliche Merkmale:

- Entscheidungen im Verlauf des Vorgangs (wurde ein Kredit genehmigt oder abgelehnt?)
- Kreditsumme
- Name des Antragstellers

Alle diese fachlichen Merkmale sind in der Regel als Attribute an Materialien zu finden, die in der Vorgangsmappe liegen.

Diskussion eines Konstruktionsvorschlags

Der vorige Abschnitt hat gezeigt, dass die für die Vorgangsverfolgung interessierenden Merkmale alle unmittelbar oder mittelbar an oder in der Vorgangsmappe zu finden sind, und dass sie dabei z. T. in Abhängigkeit von Versendevorgängen interpretiert werden müssen. Außerdem habe ich festgehalten, dass der Vorgangsmonitor für die in Umlauf befindlichen Vorgangsmappen möglichst aktuelle Snapshots vorhalten muss.

7.1.1 Ein erster Konstruktionsvorschlag

Die Folgerung aus den obigen Bemerkungen ist, dass der Vorgangsmonitor seiner Aufgabe gerecht werden kann, wenn er jeweils zum Zeitpunkgt des Versandes einer Vorgangsmappe mit einem solchen Snapshot versorgt wird.

Eine relativ einfache Möglichkeit, dieses Ziel zu erreichen, zeigt folgende Abbildung:

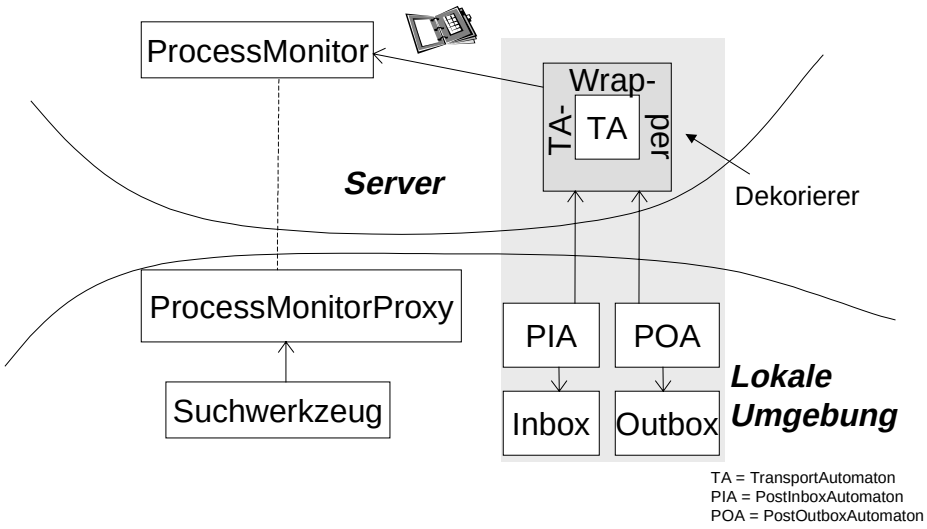


Abbildung 47: Ein Konstruktionsvorschlag für die Vorgangsverfolgung

Die Grundidee bei der in Abbildung 47 gezeigten Konstruktion liegt darin, den Transportautomaten, der die Serverkomponente des Postversandsystems darstellt (die Bestandteile des Postversandsystems sind in der Abbildung grau unterlegt), durch einen speziellen Dekorierer[81], den `TransportAutomatonWrapper`, zu "umwickeln", der bei jedem Versendevorgang, den der Transportautomat durchführt, das versendete Material daraufhin prüft, ob es eine Vorgangsmappe ist, und für jede versendete Vorgangsmappe einen Snapshot erzeugt und an den Vorgangmonitor (`ProcessMonitor`) weitergibt. Die Abbildung deutet außerdem noch die Existenz eines Suchwerkzeuges an, das den Vorgangsmonitor (über einen Proxy) zur Realisierung seiner Funktionalität verwendet.

Ein Vorteil der gewählten Architektur soll nicht unerwähnt bleiben: Indem der Vorgangsmonitor als passives, von anderen zu rufendes Objekt konzipiert wird, ist es relativ einfach, gegebenenfalls noch andere Automaten oder Werkzeuge zu verwenden, um ihn mit Informationen zu versorgen. Wenn beispielsweise das Raumsystem aus der Arbeit von Jörn Koch (s. [Koc00]) adressierbare Postkörbe in den Räumen enthielte, könnte das Platzieren einer Vorgangsmappe in einen solchen Postkorb durch direktes Hineinlegen ("Drag and Drop") wie ein Versendevorgang an den Vorgangsmonitor gemeldet werden.

[81] Das Dekorierer-Muster ist ein Entwurfsmuster, das in [GHJ+] beschrieben ist. Dort heißt es zum Dekorierer: "Erweitere ein Objekt dynamisch um Zuständigkeiten. Dekorierer bieten eine flexible Alternative zur Unterklassenbildung, um die Funktionalität einer Klasse zu erweitern".
Der Dekorierer (auch: "Gebundener Umwickler") referenziert ein Objekt und ist vom selben oder von einem Supertyp dieses "umwickelten" Objektes. Er erbringt seine eigene Funktionalität, indem er die Operationen des umwickelten Objekts aufruft und zusätzliche Funktionalität hinzufügt.

7.1.2 Technische Umsetzung des Vorschlags

Um die Rolle des Transportautomaten im Postversandsystem zu erklären, stelle ich hier kurz die Komponentenzerlegung des Postversandsystems dar, die Freund in [Fre00] beschreibt:

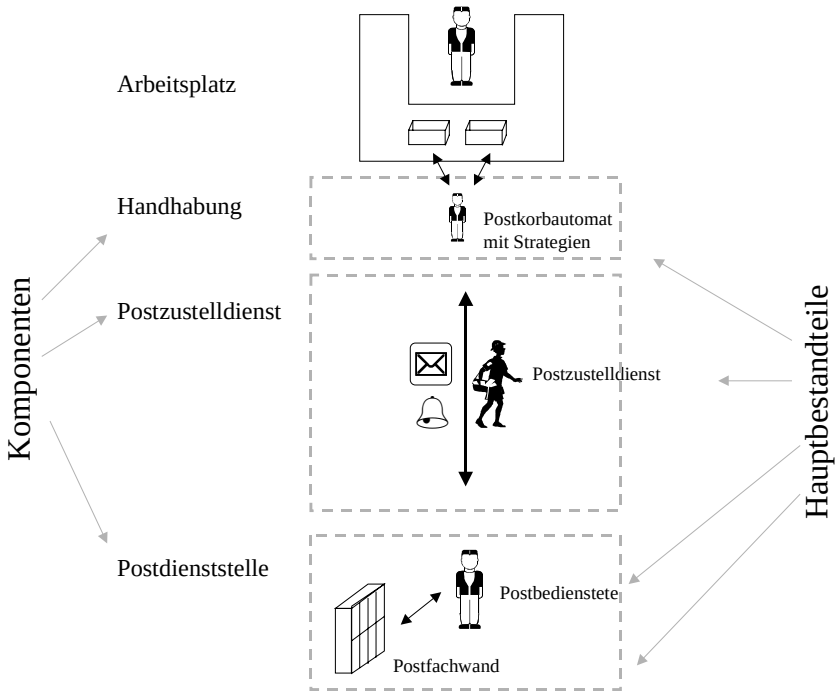


Abbildung 48: Architektur des Postversandsystems nach [Fre00]

Die Komponenten bezeichnet Freund als *Postdienststelle, Postzustelldienst* und *Handhabungskomponente*, wobei der Transportautomat Bestandteil des Postzustelldienstes ist. Die Aufgabenteilung unter den Komponenten lässt sich so beschreiben:

- Die *Postdienststelle* verwaltet Postfächer für alle angemeldeten Adressen. Diese kann man sich wie einer Postfachwand angeordnet vorstellen. Eingehende Sendungen (also Materialien vom Typ `TransportableThing`) werden in das passende Empfängerfach abgelegt und können unter Angabe der Adresse von dort abgeholt werden.
- Der *Postzustelldienst* kann Materialien versenden, d.h. er nimmt sie entgegen und liefert sie bei der Postdienststelle ab, wo sie im passenden Postfach abgelegt werden. Er kann umgekehrt auch die Sendungen für einen bestimmten Addressaten von der Postdienststelle abholen. Außerdem benachrichtigt er die Handhabungskomponente ggf. über das Eintreffen von Sendungen für bestimmte Adressaten. Der Postzustelldienst überwindet die Rechnergrenzen und kapselt die verwendete Middleware.
- Die *Handhabungskomponente* umfasst Automaten (Posteingangs-, Postausgangsautomat), die auf Initiative des Anwenders bzw. zu definierten Zeitpunkten den Postzustelldienst mit Sendungen beschicken oder als Reaktion auf eine Benachrichtigung den Postzustelldienst die neu eingetroffenen Sendungen abholen lassen. Dabei greift die vorliegende Implementation auf ausgezeichnete Posteingangs- und Postausgangskörbe in der lokalen Benutzerumgebung zu.

Die Schnittstelle `TransportAutomaton` stellt in dieser Aufteilung die Schnittstelle dar, die der Postzustelldienst gegenüber der Handhabungskomponente anbietet:

```
public interface TransportAutomaton
{
  ...
  public void registerAddress (dvTransportAddress a);
  public Vector addresslist ();
  public boolean isSendable (dvTransportableDescription desc);
  public void send (TransportableThing t);
  public dvTableOfContents tableOfContents (dvTransportAddress address);
  ...
}
```

Abbildung 49: Schnittstelle `TransportAutomaton` (Ausschnitt, ohne Exceptions)

Die Handhabungskomponente kann mit `registerAddress()` bei dem Transportautomaten Adressen registrieren, für die dieser dann bei der Postdienstelle ein Postfach einrichten lässt und kann mittels `addressList()` alle registrierten Adressen erfragen. Mittels `tableOfContents()` kann ein Inhaltsverzeichnis aller Sendungen für einen bestimmten Adressaten erfragt werden.

Für die Benachrichtigung des Vorgangsmonitors interessant ist der Sendevorgang: Mit der Methode `isSendable()` kann für die Beschreibung eines Materials vom Typ `TransportableThing` überprüft werden, ob das Material versendbar ist. Trifft dies zu, kann es mit der Methode `send()` abgeschickt werden. Genau an dieser Methode setzt nun der für die Vorgangsmappen konstruierte Dekorierer `TransportAutomatonWrapper` an.

```
public class TransportAutomatonWrapper implements TransportAutomaton
{
  ...
  // constructor
  public TransportAutomatonWrapper(TransportAutomaton automaton,
     ProcessMonitor monitor)
  {
    _automaton = automaton;
    _monitor = monitor;
  }

  // send the TransportableThing
  public void send (TransportableThing t) throws TransportException
  {
    _automaton.send(t);

    if (t instanceof ProcessFolder)
    {
      // t is ProcessFolder
      _monitor.announceForwarding((ProcessFolder) t);
    }
  }
  ...
}
```

Abbildung 50: Klasse `TransportAutomatonWrapper` (Ausschnitt, ohne Exception Handling)

Der `TransportAutomatonWrapper` erhält zur Konstruktionszeit einen Transportautomaten, den er umwickelt, und einen Vorgangsmonitor (`ProcessMonitor`), den er bei Versendung von Vorgangsmappen informiert. Alle Methoden der Schnittstelle `TransportAutomaton` außer `send()` werden durch direkte Delegation an den umwickelten Automaten abgewickelt. Nur in der Methode `send()` ist zusätzliche Funktionalität implementiert, die den Vorgangsmonitor mit Snapshots versorgt.

Die Implementation von `send()` ruft zunächst `send()` am umwickelten Transportautomaten, um das übergebene Material verschicken. Anschließend prüft sie, ob es sich bei der Sendung um eine

Vorgangsmappe handelt. Wenn dies der Fall ist, wird die Operation `announceForwarding()`[82] am Vorgangsmonitor mit einer Snapshot-Kopie der Mappe aufgerufen, um die Versendung mitzuteilen.

Der `TransportAutomatWrapper` kann Client-seitig den dort an der Umgebung registrierten TransportAutomaten ersetzen. Wenn dieser allerdings ein Proxy ist, es also ein Server-seitiges Gegenstück gibt[83], empfiehlt es sich eher, den Umwickler auf dem Server zu installieren, da dann ein möglicherweise auf dem selben Server laufender Vorgangsmonitor lokal angesprochen werden kann.

7.1.3 Diskussion des Vorschlags

Zunächst einmal kann man gegen den gemachten Konstruktionsvorschlag natürlich einwenden, es sei zu aufwendig, so häufig eine komplette Vorgangsmappe an den Vorgangsmonitor zu geben. Dem muss ich aber entgegenhalten, dass der Vorgangsmonitor nur so möglichst aktuelle Snapshots der Vorgangsmappen vorhalten kann.

Es gibt jedoch zwei echte Problemfälle:

- Wenn ein Benutzer mehrere Tätigkeiten hintereinander erledigt, die Mappe also nicht weitergeleitet werden muss, bleibt dieser Bearbeitungsfortschritt vom Vorgangsmonitor völlig unbemerkt
- Das Abschließen des Vorgangs bleibt ebenso unbemerkt (wie auch das Publizieren unbemerkt bleibt bis zum ersten Versenden)

7.1.4 Ein zweiter Konstruktionsvorschlag

Aus den obigen Überlegungen folgt, dass es die Qualität der Vorgangsverfolgung deutlich erhöht, wenn auch das Vorgangsmappenwerkzeug den Vorgangsmonitor mit der Vorgangsmappe versorgt, sobald eine Mappe publiziert oder geschlossen werden soll oder eine Tätigkeit als erledigt oder durchgestrichen markiert wird.

Damit ergibt sich schematisch folgendes Bild:

[82] Die Schnittstelle des Vorgangsmonitors soll hier noch nicht en detail dargestellt werden. Sie wird in Abschnitt 0 ab Seite 110 beschrieben.

[83] Der Sinn der Unterteilung des Transportsystems in die drei Komponenten ist es ja gerade, z.B. den Postzustelldienst austauschen zu können. So ist zwar die momentan vorhandene Implementation Proxy-basiert (d.h. es gibt Server-seitig einen Transportautomaten), aber es wäre natürlich auch denkbar, hinter dem Zustelldienst einen gebräuchlichen Mailserver zu verbergen (es gäbe dann auch keine “Postdienststelle” mehr) — ohne dass es einen Server-seitigen Transportautomen gibt.

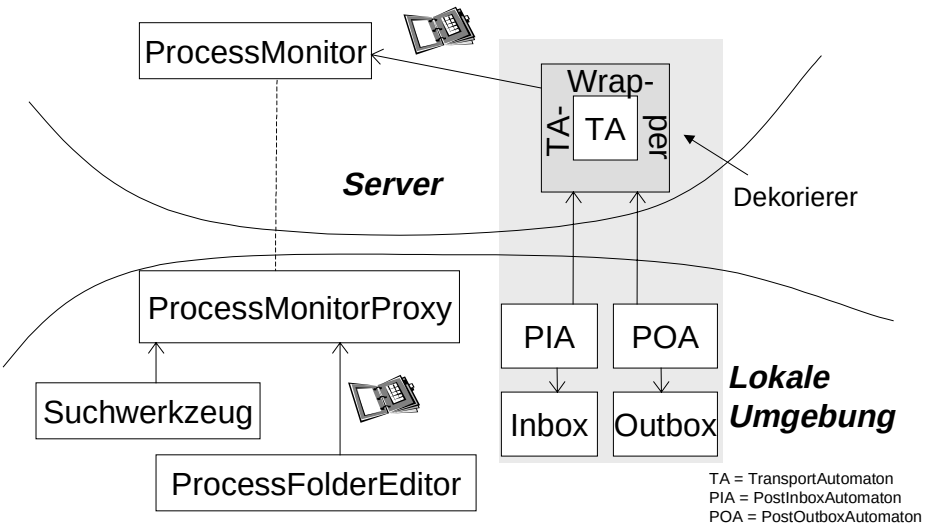


Abbildung 51: Der zweite Konstruktionsvorschlag

In derselben Weise, wie es bisher schon der Umwickler des Transportautomaten mit dem Vorgangsmonitor tat, versorgt das Vorgangsmappenwerkzeug (`ProcessFolderEditor`) den Proxy des Vorgangsmonitors mit Snapshots.

Die gerade beschriebene Architektur zieht einige Veränderungen an dem Benutzungsmodell zur Verwendung von Vorgangsmappen und Laufzettel nach sich, das in Kapitel 5 formuliert worden war.

Das veränderte Benutzungsmodell

Beim Entwurf von Vorgangsmappe und Laufzettel in Kapitel 5 habe ich mich bemüht, das Benutzungsmodell im Bezug auf diese Kooperationsmittel deutlich werden zu lassen. Mit der Einführung des Vorgangsmonitors erweitert sich dieses Benutzungsmodell zwangsläufig.

Die gravierendste Änderung im Vergleich zum bestehenden Benutzungsmodell erzwingt der Vorgangsmonitor durch die erwünschte Fähigkeit, eine Kopie des jeweils letzten registrierten Standes einer Vorgangsmappe bereitstellen zu können. Doch auch die Fähigkeit des Suchens anhand bestimmter Kriterien allein sprengt das bisherige Benutzungsmodell.

Die Vorstellung einer Vorgangsmappe, die der Benutzer erst in seinem Besitz hat und im rechten Moment — über das Postversandsystem vermittelt, aber *direkt* — dem nächsten Bearbeiter zusendet, ist nicht mehr angemessen: Mit der Übergabe an das Postversandsystem muss sich nun auch die Vorstellung verbinden, dass die Vorgangsmappe auf dem Weg zum Adressaten einer Art Kontrollinstanz, einem *Monitor*[84] übergeben wird, der für sich eine Kopie der Vorgangsmappe erstellt und eifrig Notizen macht, bevor die Mappe zum eigentlichen Bestimmungsort weitergeleitet wird.

---

[84] *Monitor* bedeutet laut dem Fremdwörter-Duden (*Duden Fremdwörterbuch*, 5., neu bearb. u. erw. Aufl., Mannheim; Wien; Zürich: Dudenverl., 1990) auch “Aufseher” (veraltet). Diese Vermenschlichung möchte ich hier nicht übermäßig betonen, aber im Zusammenhang mit der Vorstellung physischer Vorgangsmappen und eines Transportsystems mit Büroboten mag dies ein hilfreiches Bild sein.

Wird, wie in Abschnitt 7.1.4 vorgeschlagen, auch das Vorgangsmappen-Werkzeug so modifiziert, dass es den Vorgangsmonitor mit Snapshots versorgt, muss dem Benutzer zusätzlich klar sein, dass auch das Abhaken oder Durchstreichen einer Tätigkeit auf dem Laufzettel sogleich zu dem Versenden einer Kopie der Vorgangsmappe an den Vorgangsmonitor führt, und dass die Ergebnisse dieser Handlungen damit für andere Benutzer potenziell sichtbar werden.

Das veränderte Benutzungsmodell hat eine ebenso wichtige Bedeutung für diejenigen Anwender, die den Vorgangsmonitor nutzen, um Vorgänge zu verfolgen, denn der Zeitpunkt, zu dem der Monitor mit Informationen versorgt wird, und die Art, wie dies geschieht, ist wiederum entscheidend für die Interpretation der von ihm bereitgestellten Informationen.

Suchen

In Abschnitt 0 (ab Seite 101) habe ich die für den Vorgangsmonitor interessierenden Merkmale eines Vorgangs in generische und fachliche Merkmale unterteilt und festgestellt, dass für beide Arten von Merkmalen Such- bzw. Filterfunktionen durch den Vorgangsmonitor bereitgestellt werden müssen.

7.1.5 Suchen anhand generischer Merkmale

Es gibt etliche Fragestellungen, unter denen man die generischen Merkmale einer Vorgangsmappe zum Zweck des Filterns betrachten kann (z.B. Suche anhand des Zeitpunkt der letzten Versendung, Suche nach allen Kreditvorgängen, bei denen die Tätigkeit “Auszahlung” noch nicht erfolgt ist usw.). Ich stelle eine begrenzte Auswahl der entsprechender Abfragen über die Schnittstelle des Vorgangsmonitors bereit (s. Abschnitt 7.1.8 ab Seite 111).

Obgleich im Rahmen dieser Arbeit nicht mehr implementiert, stelle ich im Folgenden eine Skizze für ein simples Suchwerkzeug vor, das das Filtern von Vorgängen anhand generischer Merkmale erlaubt.

Dem Werkzeug liegt folgende grafische Benutzungsschnittstelle zu Grunde:

Abbildung 52: Suchwerkzeug für generische Merkmale (GUI-Entwurf)

In der obigen Abbildung kann durch die mit *Abfragetyp* beschriftete Kombobox eine der zur Verfügung stehenden Suchmöglichkeiten ausgewählt werden. Im Beispiel ist es die Suche nach allen Vorgangsmappen, die sich an einer bestimmten Transportadresse befinden.

Der darunter befindliche umrandete Bereich zeigt Felder an, in die die zu dem Abfragetyp passenden Suchkriterien eingetragen werden können. Im Beispiel kann die Suche auf Vorgangsmappen eines bestimmten Typs eingeschränkt werden, und natürlich muss die Transportadresse angegeben werden.

Nach Eingabe der Suchkriterien kann mittels des *Search...*-Knopfes die Suche durchgeführt werden. In der mit der Überschrift *Matches* betitelten Liste werden die gefundenen Vorgangsmappen aufgeführt[85]. Durch Auswahl einer der gefundenen Mappen und Drücken des Knopfes *Put Copy On Desktop...* kann eine Kopie der ausgewählten Mappe auf den Desktop abgelegt werden. Diese kann dann mit dem Vorgangsmappenwerkzeug betrachtet werden, um sich ein genaues Bild von dem Vorgang zu machen.

7.1.6 Suchen anhand fachlicher Merkmale

Das Problem bei der Realisierung einer Suchfunktion anhand fachlicher Merkmale besteht darin, dass die Merkmale qua Definition an keiner dem Vorgangsmonitor bekannten Schnittstelle (Vorgangsmappe, Laufzettel, Tätigkeit) ablesbar sind.

Die naheliegende Schlussfolgerung hieraus ist, dass die Schnittstelle `ProcessFolder` der Vorgangsmappe so erweitert werden muss, dass auf die fachlichen Merkmale zugegriffen werden kann (demnach ebenfalls über ein generisches Protokoll). Dieser Weg wird von mir auch

[85] Hier könnte natürlich auch eine Tabellendarstellung gewählt werden, die noch mehr Informationen über die Vorgangsmappe darstellt.

verfolgt[86] und ist in Abschnitt 7.1.9 ab Seite 112 beschrieben. Vereinfacht dargestellt werden die verschiedenen fachlichen Merkmale als Name-Wert-Paare an der Vorgangsmappenschnittstelle verfügbar gemacht.

Damit ist das Problem aber noch nicht gelöst, denn auf Ebene eines generischen Protokolls verlieren die fachlichen Merkmale ihre Semantik, so dass am Vorgangsmonitor hierfür natürlich keine Suchoperation für bestimmte Fragestellungen angeboten werden kann. Darum muss auch die Suchfunktionalität für die fachlichen Merkmale am Vorgangsmonitor so gestaltet sein, dass eine maximale Freiheit bei der Formulierung von Bedingungen für die fachlichen Merkmale besteht.

Ich habe dies ermöglicht, indem ich eine SQL-artige Abfragesprache definiert habe, mit der die Suchbedingungen formuliert werden können (s. Abschnitt 7.1.9 ab Seite 112 für die Details).

Die Schnittstelle des Vorgangsmonitors

In diesem Abschnitt stelle ich die Schnittstelle des Vorgangsmonitors vor, die ich aus den vorangegangenen Überlegungen heraus entwickelt habe. Das zugehörige Java-Interface trägt den Namen `ProcessMonitor`.

Es gibt es eine zentrale Operation an der Schnittstelle `ProcessMonitor`, die es ermöglicht, den Snapshot einer Vorgangsmappe zu erhalten, deren eindeutiger Identifikator als Parameter übergeben wird:

```
public ProcessFolder folderCopy(dvIdentificator folderID);
```

Abbildung 53: Die Operatrion `folderCopy()` der Schnittstelle `ProcessMonitor`

Der zum Aufruf dieser Methode benötigte Identifikator kann unter anderem durch Verwendung der übrigen Operationen des Vorgangsmonitors gefunden werden, die ich im Folgenden ausführlich vorstellen will.

7.1.7 Listener Part

Als den *Listener Part* bezeichne ich den Teil der Schnittstelle, über die Informationen zum Vorgangsmonitor gelangen, mit dem der Vorgangsmonitor "zuhört". Dies geschieht über die folgenden Methoden:

```
public void publishProcessFolder(ProcessFolder folder);
public void closeProcessFolder(ProcessFolder folder);
public void announceChange(ProcessFolder folder);
public void announceForwarding(ProcessFolder folder);
```

Abbildung 54: Listener Part der Schnittstelle `ProcessMonitor`

Wie ich in Abschnitt 0 festgestellt habe, ist der Vorgangsmonitor in der Lage, alle relevanten Informationen zu einem bestimmten Vorgang aus dessen Vorgangsmappe zu entnehmen. Aus diesem Grund ist bei allen vier dargestellen Methoden der einzige Parameter vom Typ `ProcessFolder` (also eine Vorgangsmappe).

[86] Allerdings stellt sich hierbei heraus, dass die Schnittstelle `ProcessFolder` gar nicht erst verändert werden muss, wenn bei der Implementation eine bestimmte Konvention beachtet wird.

Die beiden ersten Methoden spielen eine Rolle in Bezug auf die "Lebensphasen" der Vorgangsmappe (s. Abschnitt 5.1.7 ab Seite 68), denn mit `publishProcessFolder()` wird die Vorgangsmappe zum ersten Mal für die Vorgangsverfolgung sichtbar, während mit `closeProcessFolder()` der Vorgang als abgeschlossen erklärt wird und aus der Vorgangsverfolgung verschwindet.

Die vorletzte Methode `announceChange()` muss z.B. durch das Vorgangsmappenwerkzeug gerufen werden, wenn eine Tätigkeit auf dem Laufzettel abgehakt oder gestrichen worden ist. Die Methode `announceForwarding()` dagegen muss gerufen werden, um eine Versendung der Vorgangsmappe zu kennzeichnen. Sie unterscheidet sich von `announceChange()` dadurch, dass die Adressinformationen an der Vorgangsmappe anders interpretiert werden: Wird `announceForwarding()` gerufen, wird davon ausgegangen, dass sich die Mappe danach bei der auf ihr angegebenen Adresse befindet, während diese Annahme bei `announce Change()` nicht gemacht werden kann.

7.1.8 Informer Part (Generische Merkmale)

Als den *Informer Part* der Schnittstelle ProcessMonitor bezeichne ich diejenigen Operationen, die es erlauben, Informationen zu Vorgängen zu erhalten. Zunächst behandle ich daraus denjenigen Teil, der die Suche anhand generischer Merkmale ermöglicht.

Betrachtet man die Anzahl von Merkmalen, wird deutlich, dass sich durch die Vielzahl der möglichen logischen Verknüpfungen eine unübersehbare Vielfalt an Fragestellungen formulieren lässt. Da dem Entwurf kein konkretes Anwendungsbeispiel zugrunde liegt, gibt es keine gesicherte Grundlage, um diese möglichen Fragestellungen zu bewerten. Ich war deshalb dazu gezwungen, die meiner Meinung nach wichtigsten auszuwählen, auch anhand der Überlegungen in Abschnitt 0 ("Motivation für die Vorgangsverfolgung"):

```
public dvTableOfContents foldersWithMatchingActivities(dvIdentificator
                           templateID, dvIdentificator[] carriedOut,
                           dvIdentificator[] notCarriedOut);
public dvTableOfContents foldersNotCarriedOutInTime(dvIdentificator
                           templateID, dvIdentificator startActivity,
                           dvIdentificator[] toCarryOut,
                           int daysMaxAllowed);
 public dvTableOfContents foldersLastChangedBetween(Date after,
                           Date before);
 public dvTableOfContents foldersLastSentBetween(Date after, Date before);
 public dvTableOfContents foldersAtAddress(dvTransportAddress address);
 public dvTableOfContents foldersAtRole(DomainValue role);
```

Abbildung 55: Informer Part der Schnittstelle `ProcessFolder` (generische Merkmale), Ausschnitt

Alle aufgeführten Operationen liefern ein Objekt vom Typ `dvTableOfContents` zurück. Dies ist ein im JWAM-Framework definierter zusammengesetzter Fachwert, der ein Inhaltsverzeichnis modelliert (in diesem Fall das Inhaltsverzeichnis der Ergebnismenge für eine Suchanfrage). Ein Exemplar dieser Klasse enthält eine Liste von Fachwerten des Typs `dvThingDescription`. Ein Exemplar von `dvThingDescription` stellt die Beschreibung eines Materials dar[87], und so beschreibt das zurückgelieferte Inhaltsverzeichnis in seiner Gesamtheit die Ergebnismenge der jeweiligen Suchanfrage.

[87] Die Schnittstelle für Materialien in JWAM, `Thing`, enthält eine Methode `thingDescription()`, die eine solche Beschreibung für ein Material liefert. In der solchen Beschreibung sind der eindeutige Identifikator des Materials, sein Name, sein dynamischer Typ (als Objekt vom Typ `Class`) sowie der Verweis auf ein Icon enthalten.

Die ersten beiden aufgeführten Operationen sind auch die kompliziertesten, denn bei ihnen wird auf der Ebene einzelner Tätigkeiten das Suchkriterium definiert. Auf Basis einer bestimmten Mappenvorlage[88], deren eindeutiger Identifikator als erster Parameter verwendet wird, können der Operation `foldersWithMatchingActivities()` die Identifikatoren erledigter und die Identifikatoren nicht erledigter Tätigkeiten[89] übergeben werden. Zurückgeliefert wird ein Verzeichnis, das all jene Vorgangsmappen listet, die auf der gegebenen Vorlage basieren und bei denen die über die Parameter festgelegten Tätigkeiten erledigt bzw. unerledigt sind.

Die Operation `foldersNotCarriedOutInTime()` ist dagegen geeignet, um Zeitüberschreitungen festzustellen (also die bereits in Abschnitt 0 angesprochene Alarmfunktion zu implementieren). Die Ergebnismenge enthält alle Vorgangmappen eines bestimmten Typs, bei der zwischen dem Erledigen einer bestimmten Tätigkeit und dem Erledigen einer Menge von Folgetätigkeiten eine gewisse Zeitspanne überschritten wird. Die Vorlage, auf der die Mappe basiert, die "Start-Tätigkeit" sowie die Folgetätigkeiten (genauer: die entsprechenden Identifikatoren) und die Zeitspanne in Tagen sind die Parameter dieser Operation.

Zu den verbleibenden vier Operationen gibt es in der Schnittstelle ProcessMonitor jeweils noch ein Gegenstück, das die Suche auf Mappen einschränkt, die auf bestimmten Vorlagen basieren (diese sind nicht aufgeführt). Die Operationen `foldersLastChangedBetween()` bzw. `foldersLastSentBetween()` liefern ein Inhaltsverzeichnis mit all jenen Vorgangsmappen zurück, die zwischen zwei bestimmten Daten, die als Parameter übergeben werden, zuletzt versendet bzw. geändert worden sind.

Die Operationen `foldersAtRole()` bzw. `foldersAtAddress()` liefern all jene Mappen zurück, die beim letzten Versendevorgang an eine bestimmte Rolle oder eine bestimmte Transportadresse, die als Parameter übergeben wird, versendet worden sind.

7.1.9 Informer Part (Fachliche Merkmale)

In Abschnitt 7.1.6 ("Suchen anhand fachlicher Merkmale") habe ich bereits die Notwendigkeit beschrieben, fachliche Merkmale über die Schnittstelle ProcessFolder auslesbar zu machen. Tatsächlich bedarf es dazu keiner Schnittstellenänderung, stattdessen muss nur ein bestehender Mechanismus genutzt werden:

Jedes Material (`Thing`) liefert in JWAM über die Operation `thingDescription()` ein Exemplar der Klasse `dvThingDescription`, das eine Beschreibung dieses Materials darstellt. Teil dieser Beschreibung ist auch eine Liste von benannten Attributen, genauer: eine Abbildung von Namen auf Fachwerte.

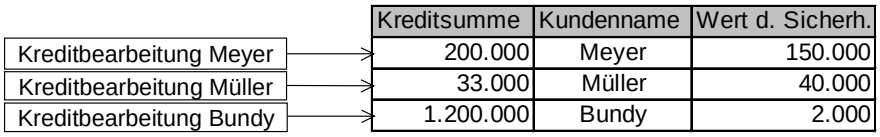
Die fachlichen Merkmale eines Vorgangs können also dem Vorgangsmonitor dadurch zugänglich gemacht werden, dass ein Objekt des Typs `ProcessFolder` als Rückgabewert der Operation `thingDescription()` eine Materialbeschreibung liefert, die als Attribute eben diese fachlichen Merkmale (in Gestalt von Fachwerten) aufweist. Wenn man die Konvention einführt, dass Vorgangsmappen, die auf derselben Vorlage beruhen, auch dieselben Attribute in ihrer

---

[88] Im Zuge der Einführung des Vorgangsmonitors mussten die Materialien Vorgangsmappe, Laufzettel und Tätigkeit so verändert werden, dass die aus einer Vorlage erstellten Objekte sich jeweils die IDs ihrer Urbilder merken und zurückliefern können.

[89] Die Identifikatoren beziehen sich dabei auf die Tätigkeitsobjekte der Vorlage; die Verbindung zu den Tätigkeitsobjekten, die in einer Mappe vorhanden sind, welche auf jener Vorlage basiert, kann hergestellt werden, weil deren Tätigkeiten wiederum den Identifikator ihrer jeweiligen Vorlage kennen (siehe Fußnote 88).

Materialbeschreibung enthalten müssen, kann man sich die Attribute aus einer Liste von Materialbeschreibungen solcher Mappen wie eine Tabelle vorstellen:



Kreditsumme	Kundenname	Wert d. Sicherh.
200.000	Meyer	150.000
33.000	Müller	40.000
1.200.000	Bundy	2.000

Abbildung 56: Fachliche Merkmale in einer Liste von Materialbeschreibungen

Um in dieser Weise angepasste Materialbeschreibungen zurückliefern zu können, muss für solche Vorgangsmappentypen, die eine Suche anhand fachlicher Merkmale gestatten sollen, dafür gesorgt werden, dass bestimmte Merkmale z.B. aus in der Mappe befindlichen Formularen ausgelesen und in die Materialbeschreibung eingefügt werden. Der simpelste Weg, so etwas zu implementieren, ist, für bestimmte Vorgangstypen jeweils spezielle Implementationen von `ProcessFolder` anzubieten[90].

Eine zweite Möglichkeit bestünde darin, eine `ProcessFolder`-Implementation zu erstellen, die Attribute bestimmter Mappeninhalte immer in ihe Materialbeschreibung einfügt: So könnten beispielsweise alle Formularfelder von Formularen des Typs *Kundenstammblatt*[91], die sich in der Mappe befinden, stets in die Materialbeschreibung der Vorgangsmappe übernommen werden.

Die zweite Lösung hat den Vorteil, dass dann mehrere Vorgangstypen mit derselben Vorgangsmappen-Implementation abgedeckt werden können.

Wenn nun die fachlichen Merkmale über die Schnittstelle ProcessFolder ausgelesen werden können, wie könnte anhand ihrer die Vorgangsmappen durchsucht werden? Ich habe in Abschnitt 7.1.6 bereits darauf hingewiesen, dass für das Suchen auf den fachlichen Merkmalen äußerste Flexibilität nötig ist, da auf der Ebene von Attributen in einer Materialbeschreibung die Semantik der Merkmale nicht mehr erkennbar ist. Darum habe ich eine einfache Abfragesprache definiert, die Attributvergleiche mit boolescher Logik verknüpft. Im Ergebnis liest sich das so ähnlich wie die Standard-Abfragesprache für relationale Datenbanken, SQL (Structured Query Language). Beispiel:

```
“(name = ‘Meyer’ or name = ‘Meier’) and kreditsumme > ‘20000’”
```

(Jeder, der SQL nur oberflächlich kennt, wird die oben formulierte Bedingung sofort verstehen.)[92]

Die beschriebene Lösung hat zwei wesentliche Vorteile:

- Ein Anwendungsprogrammierer kann für einen bestimmten Vorgangstyp und die zugehörigen Vorgangsmappen ein spezialisiertes Suchwerkzeug erstellen, dass seine Ergebnisse vom Vorgangsmonitor erhält, indem es aus den vom Benutzer eingegebenen Suchkriterien einen

---

[90] Die Klasse `ProcessFolderAdapter` ist so angepasst worden, dass sie die Attribute aus der Materialbeschreibung des umwickelten Behälters ausliest und in die eigene Materialbeschreibung einfügt. Damit ist es “nur” nötig, jeweils spezielle Implementationen des umwickelten Behälters zu verwenden.

[91] Sogenannte Formulare sind Bestandteil des JWAM-Frameworks: Es handelt sich hierbei um Materialien, die benannte Felder besitzen, in die fachliche Werte durch den Benutzer eingetragen werden können. Die Präsentation und Handhabung lehnt sich dabei weitgehend an phyisische Formulare an. Ausführliche Informationen zu Formularen in JWAM kann man in [Thie98] finden.

[92] Aus dieser Idee erwächst noch eine weitere Bedingung für die für Attribute verwendbaren Fachwerte, nämlich dass sie aus einem String erzeugt werden können (denn die in Hochkommata eingeschlossenen Werte müssen ja auf irgendeine Weise in Fachwerte gewandelt werden).

entsprechenden Abfragetext generiert und den Monitor damit aufruft. Die Schnittstelle des Vorgangsmonitors (die gleich vorgestellt wird) muss nicht angepasst weden.

- Die benutzte Abfragesprache ist so einfach, dass vom Benutzer definierte Abfragen unkompliziert in SQL oder die Abfragesprache eines objektorientierten Datenbanksystems zu transformieren sind, zumal die von mir definierte Abfragesprache den dort verwendeten Sprachen nicht unähnlich ist. Damit kann der Vorgangsmonitor leicht mit verschiedenen Persistenzmedien verbunden werden. Meine einfache Beispielimplementation ist allerdings File-basiert und verwendet einen einfachen Parser und Auswerter[93].

Kommen wir zu der Operation, die zum Suchen anhand fachlicher Merkmale verwendet wird:

```
public dvTableOfContents foldersMatching(dvIdentificator templateID,
                                         String conditionString);
```

Abbildung 57: Informer Part der Schnittstelle `ProcessFolder` (fachliche Merkmale)

Die Methode `foldersMatching()` erfordert als Parameter den eindeutigen Identifikator der Mappenvorlage, die den gesuchten Vorgangstyp festlegt, und den Abfragetext. Die Ergebnismenge wird wiederum durch ein Examplar von `dvTableOfContents` beschrieben.

Zusammenfassung

In diesem Kapitel habe ich den Entwurf für einen Vorgangsmonitor entwickelt, der als passive Komponente konzipiert ist und bei Veränderungen an Vorgangsmappen im System jeweils mit einem neuen Snapshot (einer Mappenkopie) versorgt werden muss.

Zur Bereitstellung dieser Snapshots zwei wichtige Quellen benannt: Zum einen eine Komponente des Postversandsystems, die von Freund als *Postzustelldienst* bezeichnet wird und deren Schnittstelle `TransportAutomaton` mit einem von mir implementierten, passenden Dekorierer versehen werden kann, der den Vorgangsmonitor über alle transportierten Vorgangsmappen auf dem Laufenden hält; zum anderen das Vorgangsmappenwerkzeug, das zu bestimmten Anlässen (Publizieren einer Mappe, Abhaken einer Tätigkeit) den Vorgangsmonitor informiert.

Ich habe in diesem Abschnitt generische von fachlichen Merkmalen unterschieden, auch als Suchkriterien. Am Vorgangsmonitor habe ich dementsprechend verschiedene Abfrageschnittstellen entworfen: Während für die generischen Merkmale eine Reihe von Filteroperationen angeboten wird, kann über eine spezielle Abfragesprache anhand fachlicher Merkmale gesucht werden.

Bei all dem bin ich mir der Problematik bewusst, dass der so entwickelte Vorgangsmonitor nicht auf einer Anforderungsermittlung beruht, die gemeinmsam mit Anwendern eines konkreten Systems durchgeführt worden wäre. Insbesondere bei den Operationen des Vorgangsmonitors, die ein Filtern nach den generischen Merkmalen erlauben, ist es denkbar, dass sie für einen konkreten Einsatz in Anwendungsprojekten angepasst werden müssten. Die für das Filtern nach fachlichen Merkmalen gefundene Lösung (Verwendung einer einfachen Abfragesprache) ist dagegen vermutlich flexibel genug, um in den verschiedensten Anwendungskontexten in dieser Form Verwendung zu finden.

Das Ziel, eine Komponente zu schaffen, die möglichst weitgehend ohne Vererbung einsetzbar ist, ist bis auf einen Punkt erreicht worden: die Suche anhand fachlicher Merkmale. Die hierfür

[93] Parser und Auswerter sind nach dem Prinzip des rekursiven Abstiegs konstruiert, wie es etwa in [Aho88] beschrieben ist.

notwendige Bereitstellung von Vorgangsmappen, die in ihrer Materialbeschreibung fachliche Attribute ausweisen, erfordert die Konstruktion neuer Klassen. Für eine konkrete Anwendung wird mindestens *ein* neuer Subtyp geschaffen werden müssen, der Attribute aus speziellen, für die Mappe obligatorischen Materialien (z.B. Kundenstammblatt) generisch entnimmt und in die Materialbeschreibung einfügt. Außerdem wird ein Anwendungsentwickler spezielle Werkzeuge zur Bereitstellung fachlicher Suchfunktionalität konstruieren wollen, da Anwender typischerweise nicht gezwungen sein sollten, eine spezielle Abfragesprache zu lernen. Zur Unterstützung des Anwendungsentwicklers könnte jedoch ein Testwerkzeug sinnvoll sein, das auf direkter, textueller Eingabe von Abfragen beruht. Ein solches Werkzeug fehlt in der Framework-Komponente noch.

Trotz dieser Einschränkungen ist die in dieser Arbeit konstruierte Framework-Komponente für JWAM aus meiner Sicht mit dem Vorgangsmonitor komplettiert worden. Ich stelle in der Framework-Komponente mit der Klasse `ProcessMonitorImpl` eine File-basierte, prototypische Implementation bereit.

8 Zusammenfassung und Ausblick

In dieser Arbeit habe ich den Entwurf und die Implementation einer Framework-Komponente für das auf dem WAM-Ansatz basierende, objektorientierte Java-Framework JWAM beschrieben, welche die Kooperationsmittel Vorgangsmappe und Laufzettel und einen Mechanismus zur Verfolgung der damit unterstützten kooperativen Arbeitsprozesse bereitstellt. Das konkreteste Ergebnis der Arbeit ist deshalb die gegenwärtig in einer Alpha-Version in das JWAM-Framework integrierte Komponente.

In den ersten Kapiteln dieser Arbeit habe ich vornehmlich ihre Grundlagen dargestellt: den WAM-Ansatz, das JWAM-Framework, grundlegende Begriffe aus den Bereichen CSCW und WfMS.

Daran anschließend habe ich die CoJAC-Arbeiten vorgestellt und anhand von Kriterien und Begriffen aus dem Bereich CSCW und dem WAM-Ansatz klassifiziert. Das darauffolgende Kapitel stellte eingehend das Konzept des Prozessmuster und seine Umsetzung durch Vorgangsmappe und Laufzettel dar.

Als ein Ziel dieser Arbeit habe ich an ihrem Beginn formuliert, dass der Entwurf der Komponente schrittweise und nachvollziehbar entwickelt werden solle; daran habe ich mich bei der Strukturierung der Arbeit orientiert. Deshalb habe ich die Kooperationsmittel Vorgangsmappe und Laufzettel zunächst mit dem ganz einfachen Mittel der CRC-Karten modelliert, um ein Basisverständnis dieser Gegenstände anhand der physischen Vorbilder festzuhalten, auf das ich im Lauf der Arbeit mehrfach zurückgreifen konnte.

Diese einfache, in Kapitel 4 vorgenommene Modellierung identifiziert drei elementare Klassen: die Vorgangsmappe, den Laufzettel und die Tätigkeit. Diese drei Klassen sind auch die wichtigsten Materialklassen der schließlich konstruierten Framework-Komponente. Die grundsätzliche Charakterisierung der Vorgangsmappe als einen versendbaren Materialbehälter, an dem ein Laufzettel angebracht werden kann, hat sich ebenfalls bis zum Ende durchgesetzt.

Trotz dieser seit dem ersten einfachen Entwurf unverändert gebliebenen Entwurfsentscheidungen hat sich der Entwurf natürlich verändert und wurde von mir detailliert. Dazu habe ich in Kapitel 5 zunächst andere CSCW-Systeme, die Vorgangsmappen zur Kooperationsunterstützung einsetzen, betrachtet und die darin enthaltenen Entwürfe bewertet. Die Erkenntnisse daraus sind in meine anschließend formulierten Entwurfsentscheidungen eingeflossen. So habe ich z. B. von dem Entwurf, den die RWG für die Vorgangsmappen in ihrem GEBOS-System beschrieben hat, die Idee übernommenen, das Durchstreichen von Tätigkeiten auf dem Laufzettel zu erlauben.

Auch das zu Beginn benannte Ziel, ein konsistentes Benutzungsmodell zu formulieren, das Entwicklern und Anwendern eine gemeinsame Sichtweise auf die Software bietet, habe ich hier verwirklicht, indem ich das Modell anhand von Anwendungsfällen (Use Cases) beschrieb, wobei ich sowohl die Handhabung von Werkzeugen als auch Materialeigenschaften und Konzepte wie die “Lebensphasen” einer Vorgangsmappe dargestellt habe[94].

Mit dem Abschluss des 5. Kapitels durch die Beschreibung einiger Konstruktionsdetails sind die grundlegenden Kooperationsmittel der Framework-Komponente und die zugehörigen Werkzeuge beschrieben und implementiert. Sie fügen sich als Materialien und Werkzeuge in die Handhabungsschicht des JWAM-Frameworks ein.

---

[94] Bei den Anwendungsfällen konnte ich ein weiteres Mal auf das 4. Kapitel zurückgreifen, indem ich ein Use-Case-Diagramm für die physischen Kooperationsmittel Vorgangsmappe und Laufzettel vorgestellt und erläutert hatte.

Im Anschluss an Kapitel 5 habe ich mit dem Vorgangsmonitor einen Mechanismus entworfen und implementiert, der eine Verfolgung der im System durch Vorgangsmappen unterstützten Arbeitsprozesse, eine Vorgangsverfolgung, ermöglicht. Der Vorgangsmonitor ist ein zentral auf einem Server laufender Automat, der dem Anwendungsentwickler eine Schnittstelle bietet, die er benutzen kann, um Werkzeuge zu konstruieren, die eine Übersicht über die ablaufenden Arbeitsprozesse (z. B. eines bestimmten Typs wie "Kreditprüfung") erlauben oder die im Umlauf befindlichen Vorgangsmappen anhand bestimmter fachlicher oder generischer Merkmale durchsuchen können. Auch die Bereitstellung von Informationen zu einem speziellen Vorgang wird unterstützt, indem ein Snapshot, eine potenziell veraltete Kopie der entsprechenden Mappe vom Vorgangsmonitor angeboten wird.

Das Benutzungsmodell, welches die Verwendung von Vorgangsmappen und Laufzetteln beschreibt, habe ich in Kapitel 7 bei Einführung des Vorgangsmonitors so erweitert, dass deutlich wird, wann der Vorgangsmonitor über den jeweils aktuellen Vorgangs-Zustand informiert wird. Dies ist für den Anwender sowohl als Bearbeiter in einem Arbeitsprozess von Interesse wie als Benutzer z. B. eines Suchwerkzeugs, das auf dem Vorgangsmonitor aufsetzt.

Beim Entwurf des Vorgangsmonitors habe ich dieselben CSCW-Systeme wie beim Entwurf der Vorgangsmappen danach untersucht, welche Formen der Vorgangsverfolgung sie anbieten. Besonderen Wert legten die betrachteten Systeme hierbei auf das Durchsuchen der Vorgänge anhand fachlicher Merkmale.

Während sich das Filtern der Vorgänge anhand der Struktur des Laufzettels als (technisch) relativ einfach herausstellte, musste ich mich bei der Bereitstellung einer Suchfunktionalität anhand fachlicher Merkmale (wie Kontostand, Alter, etc.) dafür entscheiden, eine SQL-ähnliche Abfragesprache für den Vorgangsmonitor zu definieren. Damit diese eingesetzt werden kann, müssen allerdings durch den Anwendungsentwickler spezielle Vorgangsmappen konstruiert werden, die die Attribute, die als Suchkriterien dienen können, in Form einer Materialbeschreibung bereitstellen.

Mit den Kooperationsmitteln Vorgangmappe und Laufzettel und den zugehörigen Werkzeugen enthält die Framework-Komponente für Vorgangsmappen Klassen, die vom Anwendungsentwickler im Sinne eines Black-Box-Frameworks benutzt werden können[95]. Dagegen verlangt der Einsatz des Vorgangsmonitors die Spezialisierung von Framework-Klassen durch den Anwendungsentwickler (im Sinne eines White-Box-Frameworks), wenn die fachlichen Suchfunktionen genutzt werden sollen.

Integration mit den anderen CoJAC-Arbeiten

Die Integration der Framework-Komponente für Vorgangsmappen mit den Framework-Komponenten, die aus den anderen CoJAC-Arbeiten hervorgegangen sind, ist beim Postversandsystem offensichtlich: Die Vorgangsmappen sind versendbare Gegenstände, die mit Hilfe des Postversandsystems verschickt werden können, und der Vorgangsmonitor "belauscht", wenn auch indirekt über einen Dekorierer, den Transportautomaten des Postversandsystems.

Mit der Raumkomponente ist die in dieser Arbeit konstruierte Framework-Komponente dagegen nicht verkoppelt. Vorgangsmappen und Laufzettel können in den dort vorhandenen gemeinsamen Räumen allein über ihre Typisierung als Material abgelegt und verwendet werden.

[95] Dass auch die Spezialisierung für den Anwendungsentwickler in vielen Situation sinnvoll ist, wird in Kapitel 5 mehrfach hervorgehoben.

Ausblick

Im Verlauf dieser Arbeit sind einige Fragen aufgetaucht, die mit der Entwicklung des JWAM-Frameworks zusammenhängen und deren Beantwortung auch für die in dieser Arbeit konstruierte Komponente wesentlich Verbesserungen ermöglichten.

Hier ist etwa die Fragestellung zu nennen, ob das JWAM-Framework ein (Software-) Komponenten-Modell benötigt und wie dieses zu gestalten wäre. Ich habe in Abschnitt 2.1.6.1 angedeutet, wie eine mögliche Antwort auf diese Frage lauten könnte. Davon ist übrigens auch die eigentlich zu große Abhängigkeit der Framework-Komponente für Vorgangsmappen vom Postversandsystem betroffen: Wenn nur die Außenschnittstelle des Postversandsystems in den Kern des JWAM-Frameworks überführt und die Implementation von Freund zur Software-Komponente würde, wäre die von mir konstruierte Framework-Komponente nur noch vom JWAM-Kern abhängig (und die Verwendung von Vorgangsmappe und Laufzettel könnte mit einer *beliebigen* Postversandsystem-Implementation erfolgen).

Ein anderes in JWAM noch nicht gelöstes Problem ist die Bereitstellung von Standard-Werkzeugen zur Auswahl und Ablage von Materialien in der Arbeitsumgung (ähnlich den entsprechenden JDK-Standarddialogen für Dateien). Ohne eine solche Unterstützung werden allzuviele Handlungen unpassenderweise über Drag and Drop erledigt. Auch die in dieser Arbeit beschriebenen Werkzeuge würden von einer entsprechenden Umgestaltung profitieren.[96]

Eine letzte Bemerkung:

Es ist für die Konstruktion von Frameworks per definition ein Problem, wenn nicht Erfahrungen aus konkreten Anwendungsprojekten abstrahiert werden, sondern “auf der Grünen Wiese” konstruiert wird (es heißt nicht umsonst: “Use before Reuse”).

Diesem Problem habe ich in dieser Arbeit zu begegnen versucht, indem ich mehrere CSCW-Systeme, die mit Vorgangsmappen-ähnlichen Konstrukten umgehen, untersucht habe. Die daraus gewonnenen Erkenntnisse sind in die Konstruktion eingeflossen.

Dennoch ersetzt dies nicht Erkenntnisse aus dem Einsatz in Anwendungsprojekten. Aus diesem Grund bin ich sehr gespannt, wie die Erfahrungen mit der in JWAM eingebundenen Framework-Komponente ausfallen werden, wenn diese (nachdem sie Alpha- und Beta-Stadium hinter sich gelassen hat) in Projekten zum Einsatz kommt.

[96] u.a. aufgrund dieser Anregungen wird gegenwärtig ein entsprechende Werkzeug, der *MaterialChooser*, für JWAM implementiert.

Literaturverzeichnis

[BBS+95]

D. Bäumer, R. Budde, K.-H. Sylla, G. Gryczan, H. Züllighoven: *Objektorientierte Konstruktion von Software-Werkzeugen und –Materialien.* Informatik-Spektrum, Band 18, Heft 4, Berlin, Heidelberg: Springer-Verlag, S. 203-210, August 1995.

[BC89]

K. Beck, W. Cunningham: *A Laboratory For Teaching Object-Oriented Thinking.* Proceedings of OOPSLA ’89, Seite 1-6, 1989.

[Bee00]

R.F. Beeger: *Einbettung und Wiederverwendung von Oberflächen bei der Werkzeugkomposition*, Studienarbeit am Fachbereich Informatik der Universität Hamburg, Herbst 2000.

[Bei00]

C. Beis: *Integration von Zugriffskontrollen in das JWAM-Framework.* Diplomarbeit am Fachbereich Informatik der Universität Hamburg, Dezember 2000.

[BGK+97]

D. Bäumer, G. Gryczan, R. Knoll, C. Lilienthal, D. Riehle, H. Züllighoven: *Framework Development For Large Systems.* In *Communications Of The ACM,* Vol.40, No. 10, S. 52-59, October 1997.

[BRJ99]

G. Booch, J. Rumbaugh, I. Jacobson: *The Unified Modeling Language User Guide.* Addison-Wesley, 1999.

[BS98]

U. M. Borghoff, J.H. Schlichter: *Rechnergestützte Gruppenarbeit. Eine Einführung in Verteilte Anwendungen.* Berlin, Heidelberg, New York: Springer Verlag, 1998.

[BG98]

K. Beck, E. Gamma: *Test Infected: Programmers Love Writing Tests.* In *Java Report,* Vol.3, Nr. 7, Seite 37-50, Juli 1998.

[BZ90]

R. Budde, H. Züllighoven: *Software-Werkzeuge in einer Programmierwerkstatt.* Berichte der GMD, Nr. 182, München, Wien: Oldenbourg, 1990.

[DB92]

P. Dourish, S. Bly: *Portholes: Supporting Awareness in a distributed work group.* In *Proceedings of the ACM Conference on Human Factors in Computer Systems (INTERCHI'92),* SIGCHI, S. 541-547. ACM Press, New York, NY, Mai 1992.

[Dou97]

P. Dourish: *Extending Awareness Beyond Synchronous Collaboration.* Position paper for the CHI'97 Workshop on Awareness in Collaborative Systems. Atlanta, Georgia, März 97

[Fre00]

M. Freund: *Komponenten zur Kooperationsunterstützung: Das Postversandsystem*. Diplomarbeit am Arbeitsbereich Softwaretechnik, Fachbereich Informatik, Universität Hamburg, Juli 2000.

[FÖ00]

J. Felski, E. Özdan: *Frameworkbasierte Werkzeugkomponenten* Studienarbeit, Universität Hamburg, Fachbereich Informatik, Arbeitsbereich Softwaretechnik, 2000.

[GHJ+96]

E. Gamma, R. Helm, R. Johnson, J. Vlissides: *Entwurfsmuster – Elemente wiederverwendbarer objektorientierter Software.* Übersetzung von D. Riehle, Bonn: Addison-Wesley, 1996.

[GHR+00]

G. Gryczan, A. Havenstein, S. Roock, I. Wetzel, H. Züllighoven: *Kooperation mit persistenten fachlichen Behältern.* In *OBJEKTSpektrum,* SIGS Conferences GmbH, Heft 1, Seite 82-87, Januar/Februar 2000.

[Grei88]

I. Greif: *Introduction* in *Computer-Supported Cooperative Work: A Book of Readings,* ed. I. Greif, 783. San Mateo, California: Morgan Kaufmann Publishers.

[Gru94]

J. Grudin: *CSCW: History and Focus.* In *IEEE Computer,* 27 (1994) 5, S. 19-26

[Gry96]

G. Gryczan: *Prozessmuster zur Unterstützung kooperativer Tätigkeit.* Dissertation am Arbeitsbereich Softwaretechnik, Fachbereich Informatik, Universität Hamburg. Deutscher Universitätsverlag, Wiesbaden, 1996.

[Hav99]

A. Havenstein: *Unterstützung kooperativer Arbeit durch eine Softwareregistratur*. Studienarbeit am Fachbereich Informatik der Universität Hamburg, 1999.

[JBS97]

S. Jablonski, M. Böhm, W. Schulze (Hrsg.): *Workflow-Management. Entwicklung von Anwendungen und Systemen; Facetten einer neuen Technologie.* dpunkt-Verlag, Heidelberg, 1997.

[JF88]

R. Johnson, B. Foote: *Designing Reusable Classes.* In *The Journal of Object-Oriented Programming*, Vol.1, No. 2, Seite 22-35, Juni/Juli 1988

[Koc00]

J. Koch: *JWAM-Komponenten zur Kooperationsunterstützung: Die Raumkomponente.* Diplomarbeit am Arbeitsbereich Softwaretechnik, Fachbereich Informatik, Universität Hamburg. Veröffentlichung voraussichtlich September 2000.

[Lip99]

Martin Lippert: *Die Desktop-Metapher in Systemen nach dem Werkzeug- und Material-Ansatz.* Diplomarbeit am Arbeitsbereich Softwaretechnik, Fachbereich Informatik, Universität Hamburg, Oktober 1999.

[MC94]

T. Malone, K.Crowston: *The Interdisciplinary Study of Coordination.* In *ACM Computing Surveys*, Vol. 26, No. 1, Seite 87-120, 1994.

[Obe91]

H. Oberquelle: *Kooperative Arbeit und menschengerechte Groupware als Herausforderung für die Software-Ergonomie.* In *Kooperative Arbeit und Computerunterstützung – Stand und Perspektiven.* Hrsg.: H. Oberquelle. Verlag für Angewandte Psychologie, 1991. S. 1-10.

[PK96]

W. Prinz, S. Kolvenbach: *Support for Workflows in a Ministerial Environment.* In: M. S. Ackermann (ed.): *Proceedings of the ACM 1996 Conference on Computer Supported Cooperative Work, Nov. 16-20,* 1996, Boston Massachusetts, USA. New York: ACM Press, 1996, pp. 199-208.

[PY96]

C. Petzold, P. Yao: *Windows 95 Programmierung.* Microsoft Press Deutschland, Unterschleißheim, 1996.

[Rie97]

D. Riehle: *Entwurfsmuster für Softwarewerkzeuge. Gestaltung und Entwurf von Anwendungen mit grafischer Benutzeroberfläche.* Addison-Wesley, 1997.

[RW98]

S. Roock, H. Wolf: *Die Raummetapher zur Entwicklung kooperationsunterstützender Softwaresysteme für Organisationen,* Diplomarbeit am Arbeitsbereich Softwaretechnik, Fachbereich Informatik, Universität Hamburg, Februar 1998.

[SB91]

K. Schmidt, L. Bannon: *CSCW: Four Characters in Search Of A Context.* In J.M. Bowers, S.D. Benford (ed.): *Studies in Computer Supported Cooperative Work.* Elsevier Science Publishers B.V. North-Holland, 1991.

[SB92]

K. Schmidt; L. Bannon: *Taking CSCW Seriously: Supporting Articulation Work.* In *Computer Supported Cooperative Work: An International Journal.* 1 (1992) 1, S.7-40

[Sch97]

H. A. Schmid: *Systematic Framework Design By Generalization; How to deduce hot spot implementation from its specification.* In *Communications Of The ACM,* Vol.40, No. 10, S. 52-59, October 1997.

[SPF96]

M. Sohlenkamp, W. Prinz, L. Fuchs: *POLIAwac — Design und Evaluation des POLITeam Awareness-Client. ...*

[Suc87]

L. Suchman: *Plans and situated actions. The problem of human-machine communication.* Cambridge University Press, 1987.

[Szy98]

C. Szyperski: *Component Software. Beyond Object-Oriented Programming.* Addison-Wesley, 1998.

[TSM+95]

Stephanie Teufel, C.Sauter, T.Mühlherr, K.Bauknecht: *Computerunterstützung für die Gruppenarbeit.* Addison-Wesley, 1995.

[UBS98]

United Bank Of Switzerland: *Handbuch: Corporate Desktop 2.12.* 1998

[Wan99]

D. Wang: *Konzeption und Realisierung eines Frameworks für Workflow Management Systeme in verteilten Umgebungen.* Diplomarbeit am Arbeitsbereich Verteilte Systeme, Fachbereich Informatik, Universität Hamburg, November 1999.

[Wor95]

The Workflow Management Coalition (D. Hollingsworth): *The Workflow Reference Model.* Document No. TC00-1003, Document Status - Issue 1.1 (erhältlich unter http://www.wfmc.org), Januar 1995

[Wul95]

M. Wulf: *Konzeption und Realisierung einer Umgebung zur Koordination rechnergestützter Tätigkeiten in kooperativen Arbeitsprozessen*. Diplomarbeit, Universität Hamburg, Fachbereich Informatik, Arbeitsbereich Softwaretechnik, September 1995

[Zül98]

H. Züllighoven: *Das objektorientierte Konstruktionshandbuch nach dem Werkzeug & Material-Ansatz.* dpunkt-Verlag, Heidelberg, 1998

Abbildungsverzeichnis