Konzeption und Realisierung einer Umgebung zur Koordination rechnergestützter Tätigkeiten in kooperativen Arbeitsprozessen

Martina Wulf

Diplomarbeit

Universität Hamburg
Fachbereich Informatik

Juli 1995

Erstbetreuer:
Prof. Dr. Heinz Züllighoven

Zweitbetreuer:
Prof. Dr. Horst Oberquelle

Diese Diplomarbeit wurde am Fachbereich Informatik der Universität Hamburg zur teilweisen Erfüllung der Anforderungen zur Erlangung des Titels Diplom-Informatikerin eingereicht.

Erklärung

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

Hamburg, den 26. Juli 1995

Martina Wulf

Grootmoor 90
D-22175 Hamburg

Betreuung

Prof. Dr. Heinz Züllighoven (Erstbetreuer)
Prof. Dr. Horst Oberquelle (Zweitbetreuer)

Prof. Dr. Heinz Züllighoven

Fachbereich Informatik
Arbeitsbereich Softwaretechnik
Universität Hamburg
Vogt-Kölln-Str. 30
D-22175 Hamburg

Prof. Dr. Horst Oberquelle

Fachbereich Informatik
Arbeitsbereich Angewandte und Sozialorientierte Informatik
Universität Hamburg
Vogt-Kölln-Str. 30
D-22175 Hamburg

Inhaltsverzeichnis

1 Einführung 1

1.1 Motivation und Zielsetzung 1

1.2 Aufbau der Arbeit 2

1.3 Graphische und textuelle Konventionen 3

2 Leitbilder und Metaphern für die Entwicklung einer Softwareumgebung 5

2.1 Die Softwareentwicklungmethode WAM 5

2.2 Das Leitbild vom Arbeitsplatz für qualifizierte menschliche Tätigkeit 6

2.3 Die Entwurfsmetaphern des WAM-Repertoires 8
- 2.3.1 Werkzeug 9
- 2.3.2 Material 10
- 2.3.3 Automat 11
- 2.3.4 Umgebung 12

2.4 Der Weg zur kooperationsunterstützenden Softwareumgebung 13

3 Der Austausch arbeitsteilig genutzter Materialien zwischen Softwareumgebungen 17

3.1 Szenario: Bearbeitung einer Kundenbeschwerde 17

3.2 Koordination auf der Basis von Arbeitsgegenständen 18

3.3 Die softwaretechnische Modellierung des Materialaustausches 20
- 3.3.1 Postkörbe 21
- 3.3.2 Der Versandautomat 23
- 3.3.3 Der Aspekt Transportierbar 24
- 3.3.4 Transportbehälter 25
- 3.3.5 Das Zusammenspiel zwischen Versandautomat und Postkörben 26
- 3.3.6 Die Materialverwaltung 29
- 3.3.7 Der Versandautomat als verteilte Architekturkomponente 31

3.4 Zusammenfassung 33

4 Die rechnergestützte Koordination routinisierter, arbeitsteiliger Prozesse 35

4.1 Szenario: Kreditvergabe in einer Bank 35

4.2 Koordination auf der Basis von Plänen 36
- 4.2.1 Koordinationspläne als implementierbare Vorschriften 38
  - 4.2.1.1 Büroprozedursysteme 38
  - 4.2.1.2 Workflow Management Systeme 42
- 4.2.2 Koordinationspläne als handlungsorientierende Ressourcen 44

4.3 Zur Verwendung von Koordinationsplänen innerhalb der Softwareumgebung 47
- 4.3.1 Materialüberblick 47
- 4.3.2 Vorgangsmappen 49
- 4.3.3 Prozeßmuster 50
- 4.3.4 Das Prozeßmusterrepertoire 53
- 4.3.5 Die Kontrolldimension von Vorgangsmappen 56
  - 4.3.5.1 Aktive Koordination der Zusammenarbeit 56
  - 4.3.5.2 Informiertes kooperatives Handeln 56
  - 4.3.5.3 Transparenz im Umgang mit gemeinsamen Arbeitsgegenständen 56

4.4 Die softwaretechnische Modellierung von Prozeßmustern 59
- 4.4.1 Die Materialklasse Prozeßmuster 59

4.4.2 Tätigkeitsbeschreibungen und Zuständigkeiten 61
4.4.3 Tätigkeitennetze 61
4.4.4 Der Behälterklasse ProzeßmusterRepertoire 66
4.5 Zusammenfassung 69

5 Integration von Materialtransport- und Koordinationskonzepten 71
5.1 Der automatisierte Versand von Vorgangsmappen 71
5.2 Rollenauflösung 74
5.3 Parallelität 76
5.4 Zusammenfassung 77

6 Der Prototyp 79
6.1 Entwicklung des Prototyps 79
6.2 Werkzeugüberblick 81
6.2.1 Auflister: Prozeßmusterrepertoire 81
6.2.2 Auflister: Repertoire der Tätigkeitsbeschreibungen 82
6.2.3 Editor: Tätigkeitsbeschreibung 82
6.2.4 Editor: Prozeßmuster 83
6.2.5 Editor: Tätigkeitennetz 83
6.2.6 Info-Werkzeug 83
6.2.7 Vorgangsmappen-Verwaltung 84
6.3 Handhabungsvisionen 84
6.3.1 Das Prozeßmuster für eine neue Vorgangsmappe festlegen 85
6.3.2 Eine Vorgangsmappe weiterreichen 87
6.3.3 Eine Vorgangsmappe empfangen 88
6.3.4 Ein Prozeßmuster bearbeiten 89
6.3.5 Den aktuellen Ort einer Vorgangsmappe ermitteln 90
6.4 Technischer Werkzeugentwurf 91
6.4.1 Trennung von Funktion und Interaktion 92
6.4.2 Werkzeugkomposition 93
6.4.3 Ereignismechanismus 95
6.5 Zusammenfassung 98

7 Zusammenfassung und Ausblick 99

Literaturverzeichnis 103

1 Einführung

Die geeignete Unterstützung von kooperativen Arbeitsprozessen durch Software ist ein aktuelles Forschungs- und Anwendungsfeld der Informatik, das sich in einer rasanten Entwicklung befindet. Die vorliegende Arbeit soll einen softwaretechnischen Beitrag zu dieser Thematik leisten. Sie befaßt sich mit der Konzeption und Realisierung einer nach objektorientierten Prinzipien entworfenen Software-Arbeitsumgebung, die es Einzelnen neben der Erledigung ihrer individuellen Tätigkeiten erlaubt, zusammen mit anderen Benutzern[1] arbeitsteilig an der Erledigung routinisierter Aufgaben zu arbeiten. Dieses Kapitel schildert Motivation und Zielsetzung der Arbeit, die im Überblick präsentiert wird.

1.1 Motivation und Zielsetzung

Die vorliegende Arbeit ist vor dem Hintergrund der Softwareentwicklungsmethode WAM entstanden, die am Arbeitsbereich Softwaretechnik des Fachbereiches Informatik der Universität Hamburg in Forschung und Lehre vertreten wird. Die Methode kann als Ausprägung einer objektorientierten und evolutionären Sichtweise verstanden werden, deren besondere Eigenschaft in den zur Verfügung gestellten Metaphern **W**erkzeug, **A**utomat und **M**aterial bei der Entwicklung von interaktiven Anwendungssystemen liegt. Das zugrundeliegende Leitbild ist der Arbeitsplatz für qualifizierte menschliche Tätigkeit (vgl. [GZ92]). Der Benutzer wird als Experte seines Fachs verstanden, dem (in Analogie zu traditionellen Arbeitsmitteln und Arbeitsgegenständen) Werkzeuge, Automaten und Materialien in der Form zur Verfügung gestellt werden, daß sie sich entsprechend den situativen Erfordernissen seines Arbeitskontextes zur qualifizierten Erledigung von Tätigkeiten verwenden lassen.

Die Entwicklung der "Werkzeug und Material" - Metapher wurde von Reinhard Budde, Karl-Heinz Sylla und Heinz Züllighoven Ende der achtziger Jahre im Rahmen des WOK-Projektes der GMD vorangetragen (vgl. [BZ90, BCS92, BZ92]). Mittlerweile ranken sich um diesen Ansatz verschiedene weiterführende Diskussionen, die vom Arbeitsbereich Softwaretechnik, Wissenschaftlern am Projektbereich Hardware/Software Codesign der GMD und dem GeBOs-Projekt der RWG in Stuttgart gemeinsam getragen werden. Sie haben u.a. die Notwendigkeit der Erweiterung des Metaphern-Repertoires um das Konzept des Automaten offengelegt (vgl. [Gry95]).

Im Laufe der letzten Jahre sind sowohl in der industriellen Projektpraxis als auch in studentischen Projekten eine ganze Reihe von Erfahrungen über den Entwurf und die Konstruktion von Anwendungssystemen nach der WAM-Methode gesammelt worden (vgl. [BGZ93, BGZ95]). Das Interesse galt dabei in erster Linie der Entwicklung von aufgabenangemessenen *Einzelarbeitsplatzumgebungen.*

Sehr häufig steht die Arbeit Einzelner jedoch in einem fachlichen Zusammenhang mit der Arbeit anderer Menschen. Da Softwareunterstützung sich folglich nicht allein auf individuelle Tätigkeiten beschränken kann, entwickelte sich im vergangenen Jahr am Arbeitsbereich Softwaretechnik ein Dissertationsvorhaben zum Thema "WAM und Kooperation". Das Ergebnis

---

[1] Als Zugeständnis an die Lesegewohnheiten werden im folgenden meist nur männliche Rollenbezeichnungen benutzt. Frauen sind wie immer mitgemeint.

dieses Vorhabens wird in Kürze mit der Arbeit von G. Gryczan vorliegen (vgl. [Gry95]). Neben WAM bildet sie den engeren fachlichen Kontext, in dem das Thema der vorliegenden Arbeit angesiedelt ist, sich entwickelt und konkretisiert hat.

Das Ziel dieser Arbeit ist, aufbauend auf den in [Gry95] erarbeiteten Konzepten für die situierte Koordination rechnergestützter Tätigkeit Realisierungsmöglichkeiten für eine *kooperative* Software-Arbeitsumgebung vorzustellen. Aus softwaretechnischer Perspektive soll dargelegt werden, welche Infrastruktur aus Werkzeugen, Automaten und Materialien den Benutzenden bereitgestellt werden muß, um ihnen die Einbindung in kooperative Arbeitsprozesse zu ermöglichen. Im Vordergrund der Betrachtung stehen der Austausch von gemeinsamen Materialien und die Koordination der Zusammenarbeit. Die vorgetragenen Ideen werden an einem Prototyp demonstriert.

An Beiträgen zur Thematik der rechnergestützten Kooperation mangelt es nicht. Die Vielfalt der vorgetragenen Ansätze wird maßgeblich von den Bemühungen einer noch recht jungen, doch mittlerweile etablierten Forschungsrichtung beeinflußt. Sie vereint unter dem Schlagwort CSCW (**C**omputer **S**upported **C**ooperative **W**ork) die verschiedensten wissenschaftlichen Disziplinen (z.B. Informatik, Arbeitspsychologie, Organisations- und Kommunikationswissenschaft), um interdisziplinär Fragen der rechnergestützten kooperativen Arbeit anzugehen. Während das erforschte Spektrum an Kooperationsformen in der CSCW-Diskussion offen ist, läßt sich die Beschäftigung mit diesem Thema für die vorliegende Arbeit eingrenzen. Sie befaßt sich ausschließlich mit der Unterstützung routinisierter und arbeitsteiliger Kooperationssituationen, in denen die Beteiligten auf bereits etablierte Formen der (zumeist räumlich und zeitlich getrennten) Zusammenarbeit zurückgreifen können.

Geregelte arbeitsteilige Prozesse stehen auch im Fokus der aktuellen Diskussion um sog. Workflow Management Systeme (vgl. [Jab95]). Diese Arbeit wird sich klar gegenüber den dort vertretenen Ansätzen abgrenzen und zur Beschreibung und Modellierung der Prozesse auf das alternative, in [Gry95] begründete Konzept der *Prozeßmuster* zurückgreifen.

Prozeßmuster werden verwendet, um auf einer fachlichen Basis die erforderliche Koordination von arbeitsteilig durchgeführten Tätigkeiten zu beschreiben und zu vergegenständlichen. Koordination kann auf diese Weise selbst zum Gegenstand der Arbeit, d.h. zum Gegenstand situativer, benutzergesteuerter Anpassung werden. Damit wird auch schon ein Hinweis auf das wesentliche Gestaltungsanliegen bei der Entwicklung der Softwareumgebung gegeben: in Einklang mit dem zugrundeliegenden Leitbild soll die Kontrolle über die Koordination bei den Beteiligten selbst belassen und auf unnötige Annahmen und Vorgaben für die fachliche Ausgestaltung der koordinierten Tätigkeiten verzichtet werden.

1.2 Aufbau der Arbeit

Kapitel 2 stellt die Softwareentwicklungsmethode WAM vor und differenziert dabei zwischen Vorgehensweise, Leitbild, Entwurfsmetaphern und Konstruktionstechniken. Das Leitbild vom Arbeitsplatz für qualifizierte menschliche Tätigkeit und die passend gewählten Entwurfsmetaphern Werkzeug, Automat, Material und Umgebung werden herausgegriffen und ausführlich erläutert. Der Weg zur kooperationsunterstützenden Softwareumgebung wird mit einer Festlegung auf die relevante Begrifflichkeit (Kooperation, Koordination) eingeleitet.

Kapitel 3 erörtert den Austausch von gemeinsam genutzten Materialien zwischen zwei Softwareumgebungen. Es wird argumentiert, daß zwei kooperierende Personen ihre Zusammenarbeit in beträchtlichem Maße auf der Basis von Materialien koordinieren, die sie als Bestandteil von erzielten oder zu erzielenden Arbeitsergebnissen untereinander austauschen. Softwaretechnische Bausteine für die Modellierung des Materialtransports - der Postkorb, der Versandautomat und der Transportbehälter - werden beschrieben und mit den Metaphern von WAM interpretiert.

Kapitel 4 schildert, wie zwei oder mehr kooperierende Personen über den geregelten Materialaustausch hinaus bei der Koordination ihrer Zusammenarbeit unterstützt werden können. In diesem Zusammenhang wird untersucht, welcher Stellenwert Plänen als Beschreibungsmittel für durchzuführende Tätigkeiten, ihre fachlichen Abhängigkeiten und personellen Zuständigkeiten beigemessen werden kann. Das Verständnis von Plänen als implementierbare und ausführbare Prozeßabbilder - wie es etwa für die gegenwärtige Workflow-Diskussion charakteristisch ist - wird kritisiert. Alternative Ansätze, die den Stellenwert von Plänen relativieren und sie anstelle von verbindlichen Vorschriften als handlungsorientierende Ressourcen interpretieren, werden diskutiert. Das in [Gry95] erarbeitete Konzept der *Prozeßmuster* wird vor diesem Hintergrund eingeordnet und erläutert. Prozeßmuster werden als materialisierte Koordinationspläne verstanden, die von kooperierenden Benutzern erstellt und verwendet werden, um ihre Zusammenarbeit eigenverantwortlich zu koordinieren. Die technischen Bausteine für die Modellierung der Prozeßmuster werden beschrieben; besonderes Interesse gilt dabei der Frage, wie das Konzept der situativen Anpaßbarkeit umgesetzt werden kann. Der Zusammenhang zwischen einem als musterhaft erkannten (und durch ein Prozeßmuster reifizierten) Prozeß und seinen Materialien wird thematisiert und auf den Begriff der *Vorgangsmappe* gebracht.

Kapitel 5 zeigt, wie sich die zuvor diskutierten Konzepte integrieren lassen, um den automatisierten Transport von Vorgangsmappen zwischen einer prinzipiell *beliebigen* Zahl kooperierender Benutzer bzw. ihrer Softwareumgebungen zu ermöglichen. Es wird untersucht, welche zusätzlichen Anforderungen sich für den in Kapitel 3 vorgestellten Versandautomaten ergeben, wenn nicht einzelne Personen, sondern Rollenträger als Empfänger einer Vorgangsmappe vorgesehen werden. Da Vorgangsmappen sich konzeptionell nicht gleichzeitig in verschiedenen Softwareumgebungen bearbeiten lassen, wird zum Problem der Parallelität Stellung bezogen.

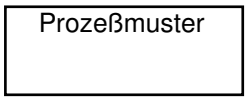
Kapitel 6 dokumentiert die Entwicklung und Handhabung des Prototyps, der praktischer Bestandteil dieser Arbeit ist. Die zuvor diskutierten Entwurfskonzepte finden in diesem Prototyp eine erste Erprobung bezüglich ihrer technischen Realisierbarkeit. Es wird ein kurzer Überblick über die verwendeten Konstruktionsprinzipien gegeben.

Kapitel 7 schließlich faßt die wesentlichen Punkte dieser Arbeit zusammen, die noch einmal kritisch reflektiert werden und Hinweise auf zukünftige Entwicklungen geben können.

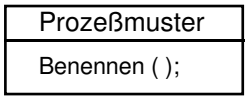
1.3 Graphische und textuelle Konventionen

Die konstruktiven Teile dieser Arbeit basieren maßgeblich auf den Konzepten der objektorientierten Anwendungsentwicklung. Die verwendeten Begriffe entstammen der Welt statisch getypter objektorientierter Programmiersprachen. Gesprochen wird u.a. von konkreten und abstrakten Klassen, von Ober- und Unterklassen und von Vererbung. Die verwendete Notation

für objektorientierte Entwurfsdiagramme basiert auf der OMT-Notation (vgl. [RBP+91]) und ist im folgenden dargestellt:

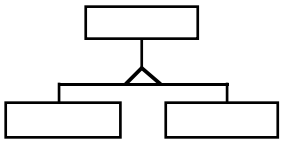


Klassen werden durch Rechtecke dargestellt. Ist der Schriftzug kursiv gesetzt, so ist die Klasse abstrakt.

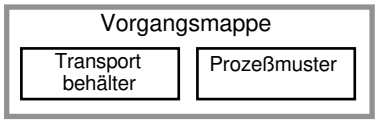


Die Operationen einer Klasse werden unterhalb einer Trennlinie, die sie vom Namen der Klasse abtrennt, aufgeführt. Ist der Schriftzug kursiv gesetzt, so wird die Operation nur abstrakt spezifiziert.

Objekte werden durch Rechtecke mit abgerundeten Ecken dargestellt. Falls vorhanden, erscheint der Name eines Objektes in Anführungszeichen unterhalb seines Typs.



Die Vererbungsbeziehung wird durch gerade Striche dargestellt, die die Klassen verbinden. Das Dreieck zeigt mit der Spitze auf die Oberklasse.



Klassen (Objekte) können logisch zu einer Gruppe von Klassen (Objekten) zusammengefaßt werden. Eine Gruppe wird durch einen dicken grauen Rahmen dargestellt.

Die Benutzt-Beziehung wird durch einen einfachen Pfeil dargestellt.

`Prozeßmuster` Entwurfskomponenten werden innerhalb des Textes in `Courier`-Schrift dargestellt.

2 Leitbilder und Metaphern für die Entwicklung einer Softwareumgebung

Die vorliegende Arbeit ist vor dem Hintergrund der am Arbeitsbereich Softwaretechnik in Forschung und Lehre vertretenen Methode WAM entstanden. WAM steht für **W**erkzeug, **A**utomat, **M**aterial, aber auch für **W**erkzeug, **A**spekt, **M**aterial, und verkörpert als Methode eine objektorientierte und evolutionäre Herangehensweise bei der Softwareentwicklung. Sie basiert auf der Verwendung allgemeinverständlicher Metaphern bei Entwurf, Konstruktion und Einsatz interaktiver Anwendungssysteme und ist mit einer grundsätzlichen Sichtweise vom Stellenwert menschlichen Arbeitshandelns verknüpft, die als *Leitbild vom Arbeitsplatz für qualifizierte menschliche Tätigkeit* diskutiert wird (vgl. [GZ92, Gry95]).

Dieses Kapitel erläutert den Softwareentwicklungsansatz WAM, beschreibt die Annahmen des ihm zugrundegelegten Leitbildes und klärt, welche Entwurfsmetaphern zur konstruktiven Umsetzung dieser Annahmen beitragen.

2.1 Die Softwareentwicklungmethode WAM

WAM vertritt als Methode einen menschenzentrierten und anwendungsnahen Ansatz. Softwareentwicklung wird als ein wechselseitiger Kommunikations- und Lernprozeß zwischen Entwicklern und Anwendern anerkannt, in dem Herstellung und Einsatz des Produktes Software miteinander verschränkt sind. Die gewählte *Vorgehensweise* basiert auf evolutionärem und partizipativem Prototyping (vgl. [Flo84, BKK+92]).

Der Systementwicklung wird als *Leitbild* der Arbeitsplatz für qualifizierte menschliche Tätigkeit zugrundegelegt (vgl. Abschnitt 2.2). Benutzer werden als Experten ihres Anwendungsbereiches verstanden, die über das notwendige Wissen und ausreichende Erfahrung verfügen, um ihre Arbeit selbständig und qualifiziert zu erledigen. Durch den Einsatz von Technik sollen flexibel verwendbare Software-Artefakte bereitgestellt werden, die den Menschen bei der Bearbeitung seiner Aufgaben unterstützen.

Die *Entwurfsmetaphern* des WAM-Repertoires (vgl. Abschnitt 2.3) helfen, die Annahmen des Leitbildes konstruktiv in ein zukünftiges Softwaresystem umzusetzen. Metaphern dienen als Analysemittel bei der Interpretation und Strukturierung eines Anwendungsbereiches, erleichtern die gedankliche Vorwegnahme des zu entwickelnden Systems und können aus Sicht des Benutzers hilfreich sein, um den Umgang mit einem Softwaresystem zu erlernen und zu meistern. Entwurfsmetaphern werden in der Regel passend zu einem zugrundeliegenden Leitbild gewählt. Verschiedene Leitbilder und Metaphern der Softwareentwicklung diskutieren Maaß und Oberquelle (vgl. [MO92]), Maaß (vgl. [Maa94]) und Coy (vgl. [Coy95]). Die zentralen Entwurfsmetaphern von WAM sind Werkzeug und Material; sie wurden mittlerweile um die Automaten- und die Umgebungsmetapher ergänzt (vgl. [Gry95]).

Im Laufe des Softwareentwicklungsprozesses sind eine Vielzahl von Dokumenten zu erstellen. WAM stellt verschiedene *Konstruktionstechniken* bereit, die Richtlinien und Orientierungshilfen für die Erstellung und Weiterführung einzelner Dokumente geben. Ein Überblick über die verschiedenen Dokumente findet sich in [KGZ93]. Für die vorliegende Arbeit sind im wesentlichen drei Dokumenttypen von Bedeutung: Szenarien, Systemvisionen und Entwurfsdokumente.

Als *Szenarien* werden solche Entwicklungsdokumente bezeichnet, die im Sinne einer Ist-Analyse Arbeitssituationen, etwa typische Arbeitsabläufe oder Handlungsstudien, beschreiben. Szenarien werden von Entwicklern geschrieben; sie beinhalten noch keine Vorstellungen über das zukünftige System und haben einen weitgehend informellen und episodischen Charakter. Wesentliche Begriffe der Fachsprache werden anhand von Szenarien herausgearbeitet und mit den Anwendern diskutiert. Szenarien bilden somit den Ausgangspunkt für die Herausbildung eines von Entwicklern und Anwendern gemeinsam getragenen Verständnisses der Anwendungswelt.

Systemvisionen antizipieren einen bestimmten Ausschnitt des zukünftigen Systems. Entwickler beschreiben darin, wie sie sich ausgewählte Arbeitssituationen nach der Einführung des Anwendungssystems vorstellen. Es existieren verschiedene Typen von Systemvisionen mit jeweils anderen Schwerpunkten; eine spezielle Systemvision, die Handhabungsvision, schildert den antizipierten Umgang eines Benutzers mit dem Softwaresystem.

Das *Entwurfsdokument* schließlich beschreibt die Softwarearchitektur, indem es die verwendeten Subsysteme, Module, Klassen und Objekte sowie ihre Beziehungen untereinander erläutert. Im Zusammenhang mit Design und Dokumentation von Softwarearchitekturen stehen derzeit sogenannte Entwurfsmuster ("design pattern") hoch im Kurs. Einen Überblick über ausgewählte Entwurfsmuster, die bei der Entwicklung von Software nach der WAM-Methode zum Tragen kommen, gibt [Rie95].

2.2 Das Leitbild vom Arbeitsplatz für qualifizierte menschliche Tätigkeit

Ein Leitbild beschreibt die grundsätzliche Sichtweise, mit der Menschen einen Ausschnitt von Realität wahrnehmen, verstehen und gestalten. Mit jedem Leitbild sind immer auch bewußte oder unbewußte Wertvorstellungen verbunden. Sie gilt es mit besonderer Achtsamkeit bei der Gestaltung von Softwaresystemen auszuloten, die in *menschliches Arbeitshandeln* eingebettet werden. Entwicklungsleitbilder geben Aufschluß über das bei der Softwareentwicklung zugrundeliegende Menschenbild und klären das Verhältnis der zukünftigen Benutzer zu dem zu entwickelnden System (vgl. [MO92, Maa94]).

Für den weiteren Argumentationsgang sind zwei Sichtweisen von besonderer Bedeutung. Sie laufen auf eine Gegenüberstellung der Automatisierung menschlicher Arbeit durch Maschinen (*Automatisierungssicht*) und ihrer aufgabenangemessenen Unterstützung mit Hilfe geeigneter Werkzeuge (*Unterstützungssicht*) hinaus (vgl. [Maa94, BZ92, Coy95]).[2]

Für die Automatisierungssicht ist typisch, daß Operationen implementiert werden, um menschliches Arbeitshandeln zu regeln und zu steuern. Hierzu werden Pläne verwendet, die im Idealfall durch Algorithmen auf Maschinen ausführbar sind und den Menschen gegebenenfalls auch ganz ersetzen sollen. Coy bemerkt zur Leitidee des Automaten (vgl. [Coy95], S. 35):

> "Der wesentliche Punkt zur Unterscheidung dieser beiden Sichtweisen ist freilich die unterschiedliche Beziehung von Arbeitssubjekt und Arbeitsmittel. Die Vorstel-

[2] [Maa94] gibt einen umfassenden Überblick über die verschiedenen, bei der Gestaltung von Softwaresystemen identifizierbaren Leitbilder, u.a. auch über das hier thematisierte *Maschinen-* bzw. *Werkstatt-*Leitbild.

> lung des Computers als Automaten weist dem Arbeitsmittel eine eigenständige, eben automatisierte Rolle im Arbeitsprozeß zu. Der an dieser Maschine arbeitende Mensch soll diese automatisierten Phasen vor- und nachbereiten, ihren Ablauf darf er freilich nicht beeinflussen, soll der vorgedachte programmierte Prozeß nicht gestört werden."

Die Maschine wird in dieser Sicht dem arbeitenden Menschen gleich- oder gar vorgesetzt, seine Arbeitstätigkeit auf die Bedienung der Maschine reduziert (vgl. [Maa94], S. 4):

> "Der mit der Maschine arbeitende Mensch wird zum potentiellen Störfaktor, der möglichst weitgehend zu eliminieren ist. Soweit seine Mitarbeit nötig ist, wird er zum Maschinenbediener degradiert, der die Vorgänge, die er anstößt, nicht durchschaut. Die Entwickler sehen sich als Ingenieure und wirken gleichzeitig als Maschineneinrichter für die Bediener, die an der vorgefundenen Bedienungsschnittstelle nichts verändern können und sollen."

Das Leitbild führt in seinen Auswirkungen u.a. dazu, daß Softwaresysteme entwickelt werden, die Arbeitsabläufe durch eine Sequenz von Menüs und Bildschirmmasken fest vorgeben und fachlich inkonsistente Systemzustände prinzipiell nicht zulassen.

Demgegenüber werden dem Anwender bei einer unterstützenden Sichtweise Hilfsmittel an die Hand gegeben, deren Einsatz er zu jedem Zeitpunkt selbst bestimmt. Anwender werden in diesem Sinne als Experten ihres Arbeitsgebiets verstanden, die Qualifikation, Kenntnisse und Erfahrung in ihr Arbeitshandeln einbringen. Dieses kann zwar von einem Softwaresystem unterstützt, aber keinesfalls ersetzt werden. Zum "Werkzeugcharakter" des Computers bemerkt Coy (vgl. [Coy95], S. 36):

> "Es gilt also, Rechner und Programme so zu gestalten, daß sie sinnvolle Möglichkeiten der Entscheidung und des Eingriffs ermöglichen und neue eröffnen. Diese Eigenschaften können als Werkzeugcharakter des Computers bezeichnet werden. Erfahrungen, Kenntnisse und Geschick werden so nicht (weg-) automatisiert, sie lassen sich bewahren, erweitern, vertiefen. Fähigkeiten, die im Umgang mit solchen Werkzeugen gewonnen werden, können in Qualität der Produkte und des Arbeitsprozesses umgesetzt werden."

Nach der WAM-Methode entwickelte Softwaresysteme basieren auf der Annahme, daß zukünftige Benutzende in ihrer Tätigkeit ausreichend qualifiziert sind, um angebotene Freiheitsgrade sinnvoll und verantwortlich zu nutzen. Der einzelne Arbeitsplatz wird daher um eine Softwareumgebung erweitert (und zum Teil auch darin modelliert), die auf die *Unterstützung*, nicht die Automatisierung qualifizierter menschlicher Arbeitstätigkeit abstellt.

Das hier zugrundegelegte Leitbild, der Arbeitsplatz für qualifizierte menschliche Tätigkeit (vgl. [GZ92, Gry95]), läßt sich durchaus in den erweiterten Kontext der aktuellen betriebswirtschaftlichen Diskussion um neue Organisationsmodelle für Unternehmen einordnen. Die Aufspaltung und Sequentialisierung menschlicher Arbeit wird mittlerweile von vielen Autoren heftig kritisiert und für ganze Aufgabenbereiche als nicht mehr adäquat betrachtet (vgl. [War92, Brö94]). In der Konsequenz führen diese Betrachtungen zu der Erkenntnis, daß der Mensch selbst wieder mehr in den Mittelpunkt der Arbeit gestellt werden muß. Für die Softwareentwicklung ergeben sich daraus u.a. die folgenden Anforderungen:

- Software sollte die Qualifikation und Kreativität der Benutzenden zum Tragen kommen lassen und sie darüberhinaus noch fördern.

Der Einsatz vom Software muß in seinen Auswirkungen durchschaubar sein, um einen verantwortungsvollen Einsatz zu ermöglichen.

Um Software den konkreten Erfordernissen einer Arbeitssituation entsprechend einsetzen zu können, muß sie flexibel durch den Benutzer steuerbar sein. Dies setzt voraus, daß Arbeitsabläufe nicht unnötig durch das System vordefiniert werden.

Die Bearbeitung anspruchsvoller Aufgaben verlangt individuell einstellbare und organisierbare Softwareumgebungen. Während der Aufgabenbearbeitung stabile Hilfsmittel (Räume, Werkzeuge, Steuerungen) sollten als Reaktion auf erkannte lokale Bedürfnisse während ihrer Nutzung *anpaßbar* sein (vgl. [Obe94], S. 34).

Mit der zuletzt genannten Forderung nach anpaßbaren Softwareumgebungen ist die entscheidende Einsicht verbunden, daß Software, zumal wenn sie in menschliche Arbeitszusammenhänge eingebettet ist, in ständiger Wechselwirkung mit einem sich wandelnden Einsatzkontext steht und ihre situative Nutzung daher nicht auf der Basis traditioneller Anforderungsanalysen vollständig antizipiert und auf Dauer festgelegt werden kann (vgl. [Obe94], S. 31).

2.3 Die Entwurfsmetaphern des WAM-Repertoires[3]

Die Umsetzung der erläuterten Leitbild-Ansprüche kann nicht erst bei Entwurf und Konstruktion des zukünftigen Softwaresystems ansetzen. Fundamentale Entscheidungen für seine weitere Entwicklung werden bereits bei der Analyse des Anwendungsbereiches getroffen, die Entwickler bestimmte Dinge bewußt wahrnehmen und andere prinzipiell ignorieren läßt. Metaphern bilden in dieser Hinsicht ein hilfreiches, aber auch richtungsweisendes Analysemittel, mit dem ein Anwendungsbereich interpretiert und gedanklich strukturiert werden kann. Maaß und Oberquelle bemerken in diesem Zusammenhang (vgl. [MO92], S. 234f.):

> "Metaphors applied in system design and use stand for certain perspectives. These perspectives, however, may be unconscious to the person who uses a metaphor. [...] Looking at things from a certain perspective means highlighting certain aspects while hiding others, means seeing certain relations and not others. It provides a frame of reference and consequently influences design and use: To someone who has a hammer everything looks like a nail. So the application of some particular perspective helps one to see more clearly in one respect while blinding to other respects."

Um vorhandene Arbeitszusammenhänge verstehen zu können, hat es sich als tragfähig erwiesen, die *Arbeitsgegenstände*, die zur Erledigung alltäglicher Aufgaben im Anwendungsbereich verwendet werden, und die *Arbeitsmittel*, mit denen diese Gegenstände bearbeitet werden, zu analysieren. Softwareentwicklung nach der WAM-Methode orientiert sich daher an dem fachlich motivierten Umgang mit solchen Arbeitsmitteln und -gegenständen und modelliert rechnergestützte Arbeitsumgebungen durch die sinnvolle Zusammenstellung von (Software-) Werkzeugen und (Software-) Materialien. Die Anwendungsdomäne sind dementsprechend

---

[3] Die in diesem Abschnitt vorgestellten Konzepte gehen zurück auf die Arbeiten von H. Züllighoven und R. Budde (vgl. [BZ90], [BZ91]). Sie können ausführlich in [KGZ93] nachgelesen werden. Die Entwurfsmetaphern *Automat* und *Umgebung* werden in [Gry95] eingeführt.

büro- und werkstattartige Umgebungen, in denen der Fokus auf den Tätigkeiten qualifizierter Mitarbeiter ruht (zur "Werkstatt-Perspektive", vgl. [MO92]).

Werkzeug und Material sind nicht nur Metaphern für die Interpretation und Analyse eines Anwendungsbereiches. Sie helfen uns gleichermaßen bei der konstruktiven Gestaltung von Systemen, die den Ansprüchen des zugrundeliegenden Leitbildes - dem Arbeitsplatz für qualifizierte menschliche Tätigkeit - genügen. Benutzenden erleichtern sie darüberhinaus die Handhabung des Softwaresystems, dessen fachliche Konzepte eng an ihre Arbeitgewohnheiten angelehnt sind.

Letzteres gilt nicht nur speziell für die Metaphern Werkzeug und Material, sondern für die Verwendung von Bildern bei der Benutzung von Computersystemen allgemein (vgl. [MO92], S.234):

> "Metaphors, carefully chosen and used to design the interface, can also serve as a learning vehicle in situations of computer use. They make systems self-explanatory or may be used explicitly in user training to explain new systems or additional aspects of systems already in use. The idea is to exploit users' prior knowledge and increase their initial familarity with the system by presenting its handling in familiar terms. Typical examples are the well-known *folders* and *trashcans* in desktop systems or the idea of providing *menus* to choose actions or objects from."

Leitbild und Entwurfsmetaphern werden passend zueinander gewählt und begleiten ein Softwaresystem auf diese Weise von der Analyse über den Entwurf bis hin zu seiner Verwendung. *Werkzeug* und *Material* sind die zentralen Metaphern des WAM-Ansatzes. Sie schaffen die entscheidende Verbindung von den bisher am Arbeitsplatz vorhandenen und gewohnten Arbeitsmitteln und -gegenständen zu den neuen Komponenten einer Softwaresystems. Das WAM-Repertoire ist mittlerweile um die Automaten- und die Umgebungsmetapher ergänzt worden. *Automaten* sollen helfen, dem Menschen lästige Sequenzen von Tätigkeiten so zusammenzufassen, daß sie ohne menschliche Eingriffe ausführbar sind. Die *Umgebung* schließlich bildet als Entwurfsmetapher den räumlichen Abschluß, in dem Werkzeuge, Automaten und Materialien "ihren Platz haben". Die verschiedenen Entwurfsmetaphern werden im folgenden einzeln dargestellt und ausführlich erläutert.

2.3.1 Werkzeug

Werkzeuge dienen der interaktiven Bearbeitung von Materialien. Mit ihnen können Materialien dargestellt, verändert und sondiert werden. Im Rahmen einer Anwendung werden Werkzeuge damit zu *Arbeitsmitteln*. Mit jedem Werkzeug ist seine fachliche Funktionalität, eine bestimmte Art der Handhabung und die Präsentation an der Benutzungsschnittstelle verbunden. Seine Funktionalität bietet das Werkzeug als einen Satz von Kommandos an, mit denen ausgewählte Routinen bei der Erledigung von Aufgaben im Anwendungsbereich modelliert werden. Die Umsetzung von Werkzeugkommandos führt zu einer Veränderung des Materials selbst oder seiner Darstellung. Jede Aktivität des Werkzeugs wird durch eine Benutzungsaktion ausgelöst, d.h. Werkzeuge verhalten sich stets reaktiv. Auswahl und Reihenfolge von Werkzeugreaktionen werden dabei durch den Benutzer festgelegt.

Ein Werkzeug kann zur Bearbeitung verschiedener Materialien geeignet sein, und umgekehrt kann ein Material auch von verschiedenen Werkzeugen bearbeitet werden. Dieser

Verwendungszusammenhang wird über *Aspekte* ausgedrückt, die das Zusammenspiel von Werkzeugen und Materialien modellieren. Aspekte definieren das Protokoll, dem ein Material genügen muß, damit es von einem Werkzeug bearbeitet werden kann. Werkzeuge erkennen die zu ihnen *passenden* Materialien an dieser Protokollschnittstelle, die über ihre Eigenschaften das Funktionieren des Werkzeuges garantiert. Mit objektorientierten Techniken wird das Protokoll in Form von Aspekt*klassen* realisiert, die eine Entkopplung von Werkzeugen und Materialien schaffen. Werkzeuge benötigen kein Wissen über konkrete Materialien, sondern *benutzen* Aspekte, um auf sie zuzugreifen, während Materialien über die von ihren Aspekten *ererbten* Schnittstellen von prinzipiell beliebig vielen Werkzeugen bearbeitbar werden, ohne spezifische Annahmen über diese zu treffen.

2.3.2 Material

Materialien werden von Werkzeugen bearbeitet. Im Rahmen einer Anwendung werden sie damit zum *Arbeitsgegenstand* oder *Bestandteil eines Arbeitsergebnisses*. Typische Materialien der Arbeitswelt sind Anträge und Formulare, aber auch abstraktere Gegenstände wie Konten, Zinssätze und Vertragskonditionen. Ob ein Ding Werkzeug oder Material ist, hängt von seinem jeweiligen Verwendungszusammenhang in einer konkreten Arbeitssituation ab. Ein Bleistift ist in einer Situation Werkzeug zum Schreiben und wird in einer anderen zum Material eines Anspitzers (vgl. [KGZ93], S. 31).

Materialien haben genau wie Werkzeuge eine eigene fachliche Funktionalität, die Benutzern des Anwendungssystems allerdings nur indirekt, nämlich über die zur Bearbeitung und Sondierung des Materials bereitgestellten Werkzeuge zugänglich ist. Die fachliche Funktionalität spiegelt die Umgangsformen mit dem Material wieder, d.h. die Art und Weise, wie mit dem Gegenstand in unterschiedlichen Aufgabenbereichen gearbeitet wird. Dabei werden vor allem die folgenden Eigenschaften betrachtet:

> Welche Informationen können an den Gegenständen "abgelesen" werden ?
>
> Welche Veränderungen werden an den Gegenständen vorgenommen, ohne daß sie zerstört oder in andersartige Gegenstände transformiert werden ?

Diese Betrachtung der Gegenstände führt zu den relevanten Begriffen der Anwendungswelt und den dahinterstehenden Verwendungskonzepten. Die fachlich motivierten Zusammenhänge zwischen Materialien werden durch ihre Anordnung in Begriffshierarchien zum Ausdruck gebracht. Ähnlichkeiten in den Umgangsformen mit Gegenständen werden generalisierend in einem Oberbegriff ausgedrückt, Spezialisierungen schlagen sich in Unterbegriffen nieder.

Behälter

Eine spezielle Materialart sind *Behälter*, die dazu benutzt werden, Arbeitsgegenstände (gegebenenfalls auch geordnet) aufzuheben. Behälter sind dadurch gekennzeichnet, daß sie explizit Material aufnehmen und herausgeben können. Über die Art dieser Materialien werden dabei prinzipiell keine Annahmen gemacht. In der Regel handelt es sich jedoch um spezialisiertes oder zumindest fachlich zusammengehöriges Material. Behälter sind ein wesentliches Konzept der Arbeitswelt, da wir es als Menschen gewohnt sind, unsere Arbeitsgegenstände (z.B. Formulare oder Briefe) mit Hilfe von Ordnern, Mappen, Akten und anderem Büro-behältermaterial zu verwalten.

2.3.3 Automat

Automaten ergänzen die Metaphern von *Werkzeug* und *Material* um ein weiteres fachlich motiviertes Ausdrucksmittel. Automaten kapseln maschinelle Vorgänge, die über längere Zeiträume und ohne äußere Eingriffe durch einen Benutzer ablaufen können. Sie realisieren lästige und formalisierbare Routinetätigkeiten oder Abläufe, die weitgehend ohne interaktive Steuerung auskommen. Einmal eingestellt laufen sie ab und produzieren vorab festgelegte Ergebnisse.

Von Gryczan werden u.a. die folgenden Unterschiede zwischen Werkzeugen und Automaten als wesentlich herausgearbeitet (vgl. [Gry95], S. 129):

> Automaten *ersetzen* den formalisierten Anteil menschlicher Handlungen; Werkzeuge *unterstützen* qualifizierte menschliche Handlungen.
>
> Automaten produzieren vorab festgelegte Ergebnisse, während Werkzeuge als unabhängige Komponenten auf Anforderungen des Benutzers reagieren.
>
> Werkzeuge unterstützen rasch wechselnde Handlungen von hoher Komplexität, die wachsende Qualifikation seitens des Benutzers erfordern. Demgegenüber implementieren Automaten eine Routine, die eine lästige Routinehandlung von geringer Komplexität insgesamt erledigt.
>
> Werkzeuge treffen keinerlei Annahmen über den Kontext, in dem sie verwendet werden. Demgegenüber müssen Automaten zumindest mit einem minimalen Modell des Kontextes ausgestattet werden, um ihren fachlichen Algorithmus ausführen zu können (sie müssen beispielsweise die Behälter kennen, in denen sie verarbeitete Materialien abgelegen sollen).

Automaten werden durch das System gestartet, laufen im Hintergrund ab und stellen ihren Service für die gesamte Laufzeit des Systems zur Verfügung. In der Regel werden sie mit Materialströmen versorgt, die sie auf der Basis ihres fachlichen Algorithmus verarbeiten. Der von einem Automaten angebotene Service sollte möglichst so beschaffen sein, daß er von mehreren Werkzeugen oder auch anderen Automaten (wieder-) verwendet werden kann. Einstellung und, falls nötig, Kontrolle eines Automat können in aller Regel mit Hilfe eines Einstellwerkzeuges interaktiv vorgenommen werden. Nach Start und Einstellung des Automaten benötigt dieser keine weitere Benutzeraktion.

Hier wird deutlich, daß Automatisierungs- und Unterstützungssicht (vgl. Abschnitt 2.2) zwei extreme Pole bei der Entwicklung von Anwendungssystemen kennzeichnen. Die Vorstellung vom Computer als Automat oder Werkzeug ist nicht notwendigerweise kontradiktorisch, sondern kann sich als komplementär erweisen (vgl. [Coy95], S. 36):

> "Zwei widersprüchliche Randbedingungen charakterisieren den Nutzen des Rechnereinsatzes: Die Automatisierung kann komplizierte Arbeit vereinfachen, der Werkzeugcharakter mag Entscheidungsspielräume erweitern. Erfolgreicher Rechnereinsatz kann die Qualität der Arbeit verbessern, indem störende oder schwer beherrschbare Komplexitäten reduziert, Entscheidungsspielräume und Lern- und Erfahrungsmöglichkeiten erweitert werden."

Daß die beiden Sichten in einem Anwendungssystem verträglich nebeneinander existieren können, setzt jedoch voraus, daß lediglich ein *vertretbarer Anteil* menschlichen Handelns

automatisiert wird. Welche Anteile menschlichen Handelns prinzipiell automatisierungswürdig und -fähig sind und welche Anteile sich grundsätzlich einer solchen Formalisierung entziehen, ließe sich ausführlich aus human- und arbeitswissenschaftlicher Sicht begründen, kann aber im Rahmen dieser Arbeit nicht ausgelotet werden. Der interessierte Leser sei hier etwa auf die Arbeiten von Suchman und Volpert verwiesen (vgl. [Suc87, Vol94]). Im Kontext von WAM werden Automaten jedenfalls nie als technischer Ersatz für qualifizierte menschliche Tätigkeit verwendet; sie können allenfalls dem Menschen lästige Handlungssequenzen zusammenfassen, die auch ohne seinen Eingriff ausführbar sind. Eine ausführliche Darstellung der in diesem Sinne verwendeten Automaten-Metapher findet sich in [Gry95].

2.3.4 Umgebung

Die zuvor diskutierten Entwurfsmetaphern geben uns Ausdrucksmittel für die Modellierung einzelner Werkzeuge und Automaten sowie der von ihnen bearbeiteten Materialien an die Hand. Ein Softwaresystem zur Unterstützung qualifizierter Tätigkeiten wird aber in der Regel nicht mit einem Werkzeug oder einem Automaten auskommen. In [Gry95] wird daher als weitere Entwurfsmetapher die *Umgebung* eingeführt (vgl. [Gry95], S. 138):

> "Eine Arbeitsumgebung zur Unterstützung qualifizierter menschlicher Tätigkeiten ist der Ort für eine anwendungsfachlich motivierte Zusammenstellung von Werkzeugen, Automaten und Materialien. Durch die Arbeitsumgebung werden keine Reihenfolgebedingungen für die Verwendung von Werkzeugen und Arbeitsmitteln festgelegt."

Die Softwareumgebung als Entwurfsmetapher orientiert sich an dem traditionellen Büroarbeitsplatz, den sich Menschen entsprechend ihrer Gewohnheiten und Ordnungsprinzipien einrichten, um mit den dort bereitgelegten Hilfsmitteln ihre Aufgaben wahrzunehmen. Folgende Eigenschaften sind charakteristisch für die Softwareumgebung:

> Innerhalb der einzelnen Umgebung sind keinerlei Abläufe definiert, d.h. die Entscheidung, wann welche Werkzeuge benutzt werden und mit welchen Materialien gearbeitet wird, liegt ausschließlich beim Anwender.
>
> Die Umgebung als rechnergestütztes Modell eines Arbeitsplatzes ist unabhängig von einem zugrundeliegenden Organisationsmodell, insofern sie keine Aussage darüber trifft, für welche Komplexität von Arbeit die Zusammenstellung der Werkzeuge, Automaten und Materialien erfolgt. Softwareumgebungen können für die Erledigung sehr begrenzter, einfacher Aufgaben, aber auch zur Unterstützung hochkomplexer Aufgabenstellungen arrangiert werden.

Als Ort, an dem Werkzeuge, Automaten und Materialien "ihren Platz haben", stellt die Umgebung eine Begrenzung dar, die es ermöglicht, *einen Arbeitsplatz von einem anderen zu unterscheiden*. Die Softwareumgebung bildet den nach außen sichtbaren "Rand" eines Arbeitsplatzes. Weder kann in den indiviuellen Arbeitsprozeß eines Anwenders steuernd von außen eingegriffen werden, noch können Materialien, die in dieser Umgebung bearbeitet werden, gleichzeitig von einem Werkzeug in einer anderen Umgebung verändert werden.

Innerhalb von Anwendungssystemen wird die Entwurfsmetapher *Umgebung* mit Hilfe eines *Umgebungsobjektes* umgesetzt. Eine ausführliche Darstellung der Aufgaben des Umgebungs-

objektes gibt [Rie95]. Hier sei nur soviel gesagt, daß (Software-) Werkzeuge, Automaten und Materialien stets in ein Umgebungsobjekt eingebettet sind, durch das sie erzeugt, verwaltet und auch wieder zerstört werden.

2.4 Der Weg zur kooperationsunterstützenden Softwareumgebung

Durch die geeignete Zusammenstellung von Werkzeugen, Automaten und Materialien lassen sich recht geradlinig Softwareumgebungen konstruieren, die *Einzelne* bei der Erledigung qualifizierter Arbeitstätigkeiten unterstützen. Häufig stehen die Tätigkeiten eines Einzelnen jedoch in einem fachlichen Zusammenhang mit denen anderer Mitarbeiter. Dies trifft beispielsweise für eine Vielzahl der Sachaufgaben zu, die traditionell in dienstleistungsorientierten Unternehmen bearbeitet werden.

Während über die Strukturierung von Einzelarbeitsplätzen in (Software-) Werkzeuge, Automaten und Materialien bereits eine ganze Reihe von Erfahrungen vorliegen, bleibt noch auszuloten, um welche Komponenten die einzelne Softwareumgebung sinnvoll erweitert werden kann, um Anwendern die Einbindung in kooperative Arbeitsprozesse zu ermöglichen. Die vorliegende Arbeit soll einen Beitrag zu dieser Thematik leisten.

Fragen der rechnergestützten Kooperation werden traditionell in der CSCW-Forschung (**C**omputer **S**upported **C**ooperative **W**ork) diskutiert. Folgende Definition von Oberquelle, die Ausdruck eines sehr allgemeinen Verständnisses von kooperativer Arbeit ist, wird für diese Arbeit zugrundegelegt (vgl. [Obe91a], S. 4):

Definition 2-1: Kooperative Arbeit

"Unter kooperativer Arbeit sollen Arbeitssituationen verstanden werden, in denen mehrere Personen zusammenarbeiten zwecks Erreichung eines Ergebnisses, welches unter den gegebenen Randbedingungen nur gemeinsam, aber nicht einzeln erzielt werden kann.
Für eine solche Situation sind folgende Eigenschaften bestimmend:

mindestens partielle Übereinstimmung der Ziele der beteiligten Personen,
gemeinsame Nutzung knapper Ressourcen durch Austausch oder gleichzeitige Nutzung
Koordination der Einzelhandlungen gemäß vereinbarten Konventionen
Verständigung über Ziele und Konventionen der Zusammenarbeit zwecks Aufrechterhaltung eines gegenseitigen Verständnisses und flexibler Anpassung.

Alle an einer gemeinsamen Aufgabe zusammenarbeitenden Personen bilden eine kooperative Gruppe."

Ein spezieller Zweig der CSCW-Forschung beschäftigt sich mit der Bereitstellung von Mehrbenutzer-Software, sogenannter Groupware, die auf die Unterstützung kooperativer Arbeit abstellt und basierend auf der Definition 2-1 von Oberquelle wie folgt definiert wird (vgl. [Obe91a], S.5):

Definition 2-2: Groupware

"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."

Der Groupware-Begriff ist ausreichend allgemein gehalten, um so verschiedene Softwaresysteme wie Gruppeneditoren, Videokonferenz-Systeme, Entscheidungs- und Planungsunterstützungssysteme oder Büroprozedursysteme zur Steuerung betrieblicher Abläufe darunter fassen zu können. Die vorliegende Arbeit erhebt selbstverständlich nicht den Anspruch, beliebige Formen der Kooperation zu behandeln. Sie fokussiert vielmehr auf die Unterstützung *routinisierter, arbeitsteiliger* Formen der Zusammenarbeit. Dies soll im folgenden näher erläutert werden.

Routinisierte Zusammenarbeit ist ihrem Wesen nach regelmäßig, aber wohlgemerkt nicht regelgeleitet. Der entscheidende Unterschied ist, daß Menschen bei der Erledigung einer gemeinsamen Aufgabe zwar immer wiederkehrende Muster im Sinne von etablierten Formen der Zusammenarbeit wahrnehmen und erkennen, sie diese aber mit dem gebotenen Entscheidungs- und Handlungsspielraum den Erfordernissen einer konkreten Arbeitssituation anzupassen wissen und ausgestalten. "Routinisiert" ist demnach weder mit "automatisierbar" noch mit "unqualifiziert" gleichzusetzen.

Mit dem Begriff der *Arbeitsteilung* wird gemeinhin die Zerlegung einer gemeinsamen Aufgabe in einzelne Tätigkeiten bezeichnet, die jeweils nur von bestimmten Personen gemäß ihrer spezifischen Qualifikationen und Kompetenzen erledigt werden können. Wenngleich die verschiedenen Tätigkeiten in einem anwendungsfachlichen Gesamtzusammenhang stehen, können sie in aller Regel sowohl zeitlich als auch räumlich getrennt voneinander bearbeitet werden, d.h. aus der Sicht des Einzelnen setzt die routinisierte Erledigung einer Tätigkeit nicht grundsätzlich die Anwesenheit anderer (kooperierender) Mitarbeiter voraus. Solche Formen der Zusammenarbeit, die sich in dienstleistungsorientierten Unternehmen wie Banken und Versicherungen genauso wie im administrativen Bereich finden lassen, unterscheiden sich demnach grundlegend von der (räumlich nicht getrennten) Arbeit an einer gemeinsamen "Werkbank", an der zwei oder mehr Personen gleichzeitig und aufeinander abgestimmt dasselbe Material bearbeiten. Für die arbeitsteilige Zusammenarbeit ist vielmehr typisch, daß Ergebnisse in Form von Arbeitsmaterialien erstellt, weitergereicht und -verarbeitet werden, die Materialien selbst aber gerade nicht gleichzeitig von mehreren Personen bearbeitet werden.

Die routinisierte, arbeitsteilige Erledigung einer gemeinsamen Aufgabe erfordert eine arbeitsplatzübergreifende *Koordination* der anfallenden Tätigkeiten (in Def. 2-1: Einzelhandlungen). Bei Crowston und Malone findet sich folgende, sehr allgemeine Definition von Koordination (vgl. [MC94], S.90):

Definition 2-3: Koordination

"Coordination is managing dependencies between activities."

Koordination setzt demnach Abhängigkeiten zwischen Tätigkeiten voraus, die es aufeinander abzustimmen gilt. Die Begriffsdefinition ist ausreichend allgemein gehalten, um sowohl die Koordination *individueller Arbeit* als auch die *kooperativer Arbeit* abzudecken. Uns geht es nicht darum, wie Einzelne die von ihnen zu erledigende Arbeit organisieren. Gegenstand der vorliegenden Arbeit ist vielmehr die rechnergestützte Koordination routinisierter arbeitsteiliger (d.h. kooperativer) Prozesse, die wie folgt definiert sei:

Definition 2-4: Koordination routinisierter, arbeitsteiliger Prozesse

> Die Koordination routinisierter, arbeitsteiliger Prozesse umfaßt die Festlegung einzelner Tätigkeiten und ihrer wechselseitigen Abhängigkeiten, die Benennung einer zuständigen Person bzw. eines Rollenträgers für einzelne Tätigkeiten sowie die bei ihrer Erledigung benötigten bzw. erzeugten Arbeitsgegenstände.

Koordination selbst ist nur *ein* Merkmal von kooperativer Arbeit (vgl. Definition 2-1). Sie kann für eine kooperative Arbeitssituation als Ganzes nicht ausreichend sein. Die Koordination routinisierter, arbeitsteiliger Prozesse setzt vielmehr voraus, daß die Randbedingungen (von der Organisation festgelegte Verfahrensvorschriften, gesetzliche Bestimmungen), Konventionen und gemeinsamen Ziele den kooperierenden Personen ausreichend transparent sind, sei es aus Erfahrung oder durch entsprechende (persönliche) Aushandlungsprozesse. Damit wird insbesondere auch deutlich, daß eine Softwareumgebung, die auf die Unterstützung der *Koordination* qualifizierter menschlicher Tätigkeiten in arbeitsteiligen Prozessen abstellt, nur einen gewissen Anteil von kooperativer Arbeit unterstützen kann und damit auch nur als Ergänzung, nicht als Ersatz zu weiteren Formen der Zusammenarbeit denkbar wird (vgl. [Obe91b], S.50).

Die im Rahmen dieser Arbeit vorgestellte Softwareumgebung soll Benutzende bei der arbeitsteiligen Erledigung routinisierter Aufgaben unterstützen, ohne dabei mit dem zugrundegelegten Leitbild, dem Arbeitsplatz für qualifizierte menschliche Tätigkeit, zu brechen. Wesentliche Konzepte und Realisierungsmöglichkeiten werden in den folgenden Kapiteln diskutiert.

3 Der Austausch arbeitsteilig genutzter Materialien zwischen Softwareumgebungen

Das Thema dieses Kapitels ist der koordinierte Austausch gemeinsam, aber nicht gleichzeitig genutzter Materialien zwischen zwei Softwareumgebungen. Als Anschauungsmaterial für die weitere Diskussion wird zunächst eine typische Kooperationssituation beschrieben, die in einem städtischen Abfallentsorgungsbetrieb bei Eingang einer Kundenbeschwerde über nicht erbrachte Leistungen der Gehwegreinigung entsteht. Anhand des Szenarios werden typische Merkmale der routinisierten, arbeitsteiligen Zusammenarbeit zwischen zwei Personen herausgearbeitet. Es wird argumentiert, daß ihre Zusammenarbeit sich in erheblichem Maße an wechselseitig ausgetauschten Arbeitsgegenständen orientiert. Darauf aufbauend wird gezeigt, wie sich der Materialaustausch an einem rechnergestützten Arbeitsplatz mit Hilfe von Posteingangs- und Postausgangskörben sowie einem Versandautomat, der die Postkörbe bedient, unterstützen läßt.

3.1 Szenario: Bearbeitung einer Kundenbeschwerde

Für die Beantwortung einer Kundenbeschwerde über nicht erbrachte Leistungen der Gehwegreinigung ist der *Betriebsleiter* eines städtischen Abfallentsorgungsbetrieb verantwortlich. Er empfängt das Schreiben, stellt anhand des Absenders den genauen Wohnbezirk des Beschwerdeverfassers fest und ermittelt auf diese Weise den für den betreffenden Wohnbezirk zuständigen *Revierleiter*.

Der Betriebsleiter fügt dem Beschwerdeschreiben zunächst einen kurzen Vermerk "mit Bitte um Sachstandsbericht" bei und legt es dann je nach Dringlichkeit in eine rote (dringlich) oder grüne (nicht dringlich) Mappe, in der es dem Revierleiter zwecks Prüfung der Sach- und Rechtslage überstellt werden kann.

Damit die Mappe über den Hauspostweg zum zuständigen Revierleiter gelangen kann, vermerkt der Betriebsleiter den Namen des Revierleiters in einem dafür vorgesehenen Feld auf der Mappenvorderseite und legt die Mappe in seinen Postausgang. Dort wird sie später von dem hausinternen Postdienst abgeholt und in den Posteingang des adressierten Revierleiters befördert.

Nach Empfang der Mappe verschafft der zuständige Revierleiter sich ein Bild von der Sachlage und prüft, ob ein Anspruch auf Gebührenrückerstattung geltend gemacht werden kann. Dies ist nur möglich, wenn die von dem Abfallentsorgungsbetrieb vorgesehenen Gehwegreinigungen aus betrieblichen Gründen nicht vorgenommen worden sind. Die Prüfung der Sachlage umfaßt in der Regel verschiedene Punkte: es muß zum Beispiel sichergestellt werden, ob die Beschwerde auf einen von dem Abfallentsorgungsbetrieb zu reinigenden Gehweg Bezug nimmt; Reinigungspläne müssen mit den Arbeitsnachweisen des Reinigungspersonals hinsichtlich eventueller Ausfälle verglichen werden, gegebenenfalls muß eine örtliche Kontrolle des Gehwegzustands veranlaßt werden.

Der Revierleiter entscheidet in Abhängigkeit vom Sachverhalt, welche Tätigkeiten er im einzelnen wahrzunehmen hat. Sein Ziel ist die Erstellung eines auf den gesammelten Informationen basierenden, informellen Sachstandsberichtes. Den fertigen Bericht fügt er der Mappe mit dem Beschwerdeschreiben bei und reicht sie über den hausinternen Postdienst zurück an den

Betriebsleiter. Dieser verfaßt nach Durchsicht des Berichtes das Antwortschreiben an den Kunden und legt anschließend den Beschwerdebrief in einem nach Eingangsdatum sortierten Beschwerdeordner ab.

3.2 Koordination auf der Basis von Arbeitsgegenständen

Anhand des Szenarios lassen sich verschiedene Merkmale der Zusammenarbeit zwischen dem Betriebsleiter und dem Revierleiter herausarbeiten:

> Bei der Erledigung einer gemeinsamen Aufgabe gehen die Kooperationspartner mit gemeinsamen Materialien (Beschwerdeschreiben, Sachstandsberichte) um, die sie erstellen, verändern oder an denen sie sich einfach nur informieren.
>
> Materialien werden als Bestandteil von erzielten Arbeitsergebnissen oder zu Informationszwecken zwischen den Kooperationspartnern ausgetauscht. Die Übergabe der Materialien erfolgt auf Veranlassung derjenigen Person, die das Material aktuell auf ihrem Schreibtisch bearbeitet.
>
> Die Arbeit mit dem gemeinsamen Material erfolgt nie gleichzeitig, sondern stets arbeitsteilig.
>
> Zu jedem Zeitpunkt hat ein Material genau *einen* festen Ort. Dies kann der Schreibtisch des Betriebsleiters oder der Posteingang des Revierleiters sein.
>
> Können die auszutauschenden Arbeitsgegenstände nicht persönlich überreicht werden, so sind Orte zu vereinbaren, an denen Materialien zur Weitergabe an den jeweiligen Kooperationspartner bereitgestellt werden (Postausgang) und an denen Materialien sichtbar zum Abholen durch den Partner bereitliegen (Posteingang).
>
> Einzelne Materialien, die über den Postweg zwischen zwei kooperierenden Arbeitsplätzen ausgetauscht werden sollen, können sinnvoll in einem adressierbaren Transportbehälter (z.B. in einer Mappe) zusammengestellt werden, in dem sie ohne Eingreifen der beiden Kooperationspartner von einem Transportdienst beförderbar sind.
>
> Eigenschaften der ausgetauschten Gegenstände können den Kooperationspartnern Informationen über den Arbeitsprozeß vermitteln, beispielweise den Grad seiner Bearbeitungsdringlichkeit.

Zusammenfassend kann festgehalten werden, daß die beiden Kooperationspartner ihre Zusammenarbeit *implizit* auf der Basis gemeinsamer Arbeitsgegenstände koordinieren. Mit impliziter Koordination ist hier gemeint, daß die Kooperationspartner sich nicht anhand von Arbeitsplänen explizit über anfallende Bearbeitungsschritte, Bearbeitungsfolgen und Zuständigkeiten verständigen, sondern diese koordinativen Informationen unausgesprochen über die gemeinsamen Materialien vermitteln, die sie wechselseitig untereinander austauschen.

Die prinzipielle Bedeutung von gemeinsam genutzten Materialien für die Koordination kooperativer Arbeit wurde bereits von Sørgaard thematisiert (vgl. [Sør88], S.321):

> "Two ways of coordinating cooperative work can be identified. One is by explicit communication about how the work is to be performed, another is less explicit, mediated by the *shared material* used in the work process. A simple example is the way two people carry a table. A part of the coordination may take place as explicit communication, for example in a discussion about how to get the table through a

> door. When the table is carried, however, the two people can follow each others´ actions because the actions get mediated through the shared material."

Interessant ist der Hintergrund, vor dem Sørgaards Arbeit über *shared material* entstanden ist. Ein primär vorgangsorientierter Zweig der CSCW-Forschung beschäftigte sich seit etwa Mitte der achtziger Jahre mit der Implementierung sogenannter Büroprozedursysteme (*office procedure systems)*. Den Ansatzpunkt für diese Systeme bildeten stark-strukturierte Aufgaben, wie sie an sachfallorientierten Arbeitsplätzen mit relativ geringem Dispositionsspielraum und starker Einbindung in formale Ablaufstrukturen anfallen. Mit der Einführung von Büroprozedursystemen sollten diese Aufgaben formal spezifiziert und in der Ablaufsteuerung weitestgehend automatisiert werden (vgl. Abschnitt 4.2.1.1).

Zu dem Zeitpunkt, als Sørgaard seinen Artikel veröffentlichte, erwuchsen angesichts der normierenden Wirkung auf das menschliche Arbeitshandeln bereits die ersten Kritiken an Büroprozedursystemen. Sørgaard schlägt daher vor, anstelle von Arbeitsabläufen die in einer Kooperationssituation gemeinsam bearbeiteten Materialien zu modellieren. Darüberhinaus legt er nahe, das Konzept des *shared material* ganz allgemein als Entwurfsmetapher bei der Entwicklung von Anwendungssystemen zu verwenden, die auf die Unterstützung kooperativer Arbeit abstellen (vgl. [Sør88], S.322):

> "If we look at how shared material is used we will get design ideas which are appropriate for computer supported cooperative work. In this way the concept of shared material is used as a design metaphor [...]."

Die von Sørgaard vorgeschlagene Entwurfsmetapher läßt sich in begrenzter Hinsicht mit der Material-Metapher des WAM-Repertoires vergleichen. Software-Materialien (*computerized materials*) werden als Objekte mit einem Satz zugehöriger Operationen interpretiert, die sich an traditionellen Umgangsformen mit den modellierten Gegenständen orientieren (vgl. [Sør88], S. 325):

> "Care must be taken in identifying the material and the primitive operations to model. Much inspiration can be obtained from the different ways the corresponding traditional material is being manipulated [...]."

Die an fachlichen Konzepten orientierte Modellierung von Softwaresystemen endet jedoch bei den gemeinsam genutzten Materialien. Sørgaard gibt keine greifbare und "plastische" Orientierungshilfe für die Strukturierung von Anwendungsmodellen insgesamt. Es fehlt insbesondere ein klares Konzept für die rechnergestützte Bearbeitung von Software-Materialien, wie es etwa durch die Werkzeug-Metapher des WAM-Repertoires bereitgestellt wird (vgl. Abschnitt 2.3.1). Die kooperative Bearbeitung von *shared material* unterliegt prinzipiell keinerlei räumlich-zeitlichen Einschränkungen; es kann sowohl gleichzeitig auf einer scheinbar gemeinsamen Benutzungsoberfläche als auch zeitversetzt und exklusiv von mehreren Benutzenden bearbeitet werden.

Sørgaards zentrale These vom Stellenwert gemeinsam genutzter Materialien für die Koordination kooperativer Arbeitsprozesse wird für die weiteren Überlegungen aufgegriffen. Da für uns die Unterstützung arbeitsteiliger Formen der Zusammenarbeit im Vordergrund steht, können jedoch bestimmte Realisierungsformen der rechnergestützten Kooperation von vornherein ausgeschlossen werden. Hierzu zählt beispielsweise die Modellierung von Werkzeugen und Materialien auf Basis der implizit bereits erwähnten WYSIWIS-Metapher (*what you see is what*

I see). Benutzenden soll auf diese Weise der Eindruck vermittelt werden, daß sie auf einer gemeinsamen Benutzungsoberfläche mit einem quasi "verteilten" Werkzeug gleichzeitig dasselbe Material bearbeiten (vgl. [SBF+87]). WYSIWIS setzt voraus, daß die Benutzer zur selben Zeit und möglichst auch im selben Raum anwesend sind, um die parallele Bearbeitung des Materials an ihrer gemeinsamen Software-Werkbank zu koordinieren. Bei arbeitsteiligen Formen der Zusammenarbeit können wir von diesen Annahmen gerade nicht ausgehen.

Der folgende Abschnitt stellt verschiedene Entwurfskomponenten zur Erweiterung der einzelnen Softwareumgebungen vor, die ihren Benutzern den koordinierten, elektronischen Austausch von Arbeitsgegenständen ermöglichen. Die Modellierung der neuen Softwarekomponenten orientiert sich an den oben herausgebildeten Merkmalen der arbeitsteiligen Kooperation und greift dabei auf die Entwurfsmetaphern des WAM-Repertoires zurück.

3.3 Die softwaretechnische Modellierung des Materialaustausches

Anhand des Szenarios "Bearbeitung einer Kundenbeschwerde" wurde bisher argumentiert, daß zwei kooperierende Personen ihre routinisierte Zusammenarbeit in erheblichem Maße auf der Basis gemeinsam genutzter Arbeitsgegenstände koordinieren, die sie wechselseitig untereinander austauschen.

Für die weiteren Überlegungen wollen wir davon ausgehen, daß die betreffenden Arbeitsgegenstände in Form von Software-Materialien vorliegen. Darüberhinaus können wir annehmen, daß zur anwendungsfachlichen Bearbeitung dieser und anderer Materialien alle notwendigen Werkzeuge in der einzelnen Softwareumgebung bereitliegen, die von dem Benutzenden bei der Erledigung seiner Arbeitstätigkeiten verwendet werden.

Prinzipiell kommen nun drei Lösungsmöglichkeiten in Frage, um den Austausch von Materialien zwischen den Kooperationspartnern zu ermöglichen:

Austausch von Materialien über traditionelle Postwege.

Selbstverständlich ließe sich der erforderliche Materialaustausch auch weiterhin über die traditionellen Transportwege einer Organisation abwickeln; dies erforderte jedoch, daß von sämtlichen weiterzureichenden (Software-) Materialien physikalische Abzüge, etwa in Form von Papierausdrucken, angefertigt werden. Diese zweifelsohne unkomfortable Lösung wird untragbar, sobald Materialien nicht nur zu Informationszwecken ausgetauscht, sondern darüberhinaus auch von mehreren Personen arbeitsteilig verändert werden sollen. Wir benötigen also eine elektronische Verbindung zwischen zwei Softwareumgebungen.

Zugriff auf gemeinsame Materialbestände und Koordination des Zugriffs per Email.

Eine relativ einfache Realisierungsmöglichkeit besteht im Einsatz elektronischer Postsysteme ("Email"), kombiniert mit einem gemeinsam zugreifbaren Materialbestand. Die Kooperationspartner senden sich elektronische Nachrichten zu, in denen sie sich wechselseitig darüber informieren, welchen Materialien jeweils von dem anderen zu bearbeiten sind.

Diese Herangehensweise stellt insofern einen Bruch mit der bisherigen Modellierung der Softwareumgebung dar, als daß Dateien gewiß keine Gegenstände sind, die Menschen zur Erledigung alltäglicher Aufgaben verwenden und mit denen sie anwendungsfachliche Umgangsformen verbinden.

Hier scheint eine Problematik durch, die mittlerweile unter dem Schlagwort "Medienbruch" diskutiert wird. Medienbrüche entstehen durch ungenügende Integration neuer Softwarekomponenten in die individuelle Arbeitsumgebung, d.h. durch mangelnde oder unzureichende Abstimmung mit der sonstigen Infrastruktur. Im Kontext rechnergestützter Kooperation treten Medienbrüche insbesondere dann auf, wenn Benutzer bei der Erledigung von (primären) Produktions- bzw. Dienstleistungsaufgaben und (sekundären) Aufgaben zur Schaffung und Anpassung der zugehörigen Arbeitsbedingungen zwischen verschiedenen Modellwelten wechseln müssen (vgl. [Obe91b], S. 44f.). Der Austausch von gemeinsamen Materialien kann in diesem Sinne als Sekundäraufgabe verstanden werden; ihre technische Unterstützung sollte sich bruchlos in die vorhandene Softwareumgebung einfügen.

Von dieser Problematik einmal abgesehen erweist sich der Einsatz herkömmlicher elektronischer Postsysteme aber auch in anderer Hinsicht als problematisch, da sich Nachrichten mit ihnen stets nur von einem Ort zum nächsten versenden lassen. Dies mag für die 4-Augen-Kooperation ausreichend sein, stößt jedoch im Rahmen komplexer arbeitsteiliger Prozesse, in denen Materialien gewöhnlich in vielen verschiedenen Softwareumgebungen und ggf. in einer vorab festgelegten Reihenfolge bearbeitet werden sollen, an seine Grenzen.

Austausch von Materialien über elektronische Postkörbe.

In dieser Arbeit wird eine Lösung vorgeschlagen, die den Transport von Materialien zwischen Softwareumgebungen mit Hilfe von elektronischen Posteingangskörben bzw. Postausgangskörben realisiert. Sie werden von einem Versandautomaten bedient, der die Ausgangskörbe leert und die daraus entnommenen Materialien in die Eingangskörbe ihres nächsten Empfängers transportiert. In den folgenden Abschnitte werden diese Entwurfskomponenten motiviert und näher erläutert.

3.3.1 Postkörbe

Aus dem Bürobereich sind wir gewohnt, den gezielten Austausch von Materialien zwischen Arbeitsplätzen über Eingangs- und Ausgangskörbe abzuwickeln. Dieses fachliche Konzept soll in die Softwareumgebung übernommen werden. Der Postausgangskorb ist ein Platz, an dem wir *bewußt* Materialien zur Weitergabe an andere bereitlegen. Analog ist der Posteingangskorb der Ort, an dem für uns *sichtbar* wird, ob uns neue Materialien zur Bearbeitung von außen bereitgestellt worden sind. Erst durch die *Vereinbarung* solcher Plätze wird der automatisierte Versand der in den Postkörben enthaltenen Materialien ermöglicht (vgl. Abschnitt 3.3.2).

Postkörbe sind stets lokal zu einer Softwareumgebung. Über sie wird eine kontrollierte Verbindung zwischen ansonsten voneinander "abgeschotteten" Softwareumgebungen hergestellt. Gegenüber anderen Materialien zeichnen sich die Postkörbe einer Umgebung dadurch aus, daß "von außen" verändernd auf sie zugegriffen werden kann und soll. Dieser Zugriff ist jedoch nicht beliebig. Von einem Eingangskorb erwarten wir zwar, daß andere uns dort etwas mit Hilfe eines Postdienstes hineinlegen lassen; neu eingegangenes Material werden wir jedoch nur selbst daraus entnehmen wollen. Umgekehrt nehmen wir nicht an, daß jemand anderes als wir selbst unsere Ausgangskörbe füllt; sehr wohl aber erwarten wir, daß sie irgendwann von dem Postdienst geleert werden. Mit dem Entwurfsrepertoire von WAM modellieren wir Postkörbe als Behälter (vgl. Abschnitt 2.3.1). Abbildung 3-1 zeigt die Umgangsformen der Behälterklasse `Postkorb`.

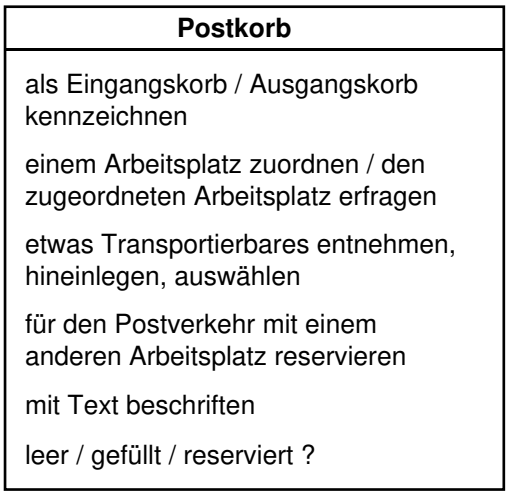


Abb. 3-1: Umgangsformen der Klasse `Postkorb`.

Im Entwurf fällt auf, daß Posteingangs- und Postausgangskörbe nicht durch getrennte Klassen modelliert werden. Diese Eigenschaft wird vielmehr durch ein Attribut zum Ausdruck gebracht, das ein `Postkorb`-Objekt entweder als Behälter für eingehende oder als Behälter für ausgehende Post kennzeichnet. Diese Entwurfsentscheidung läßt sich dadurch motivieren, daß Ein- und Ausgangskörbe in ihren grundlegenden Strukturen und Umgangsformen nicht voneinander unterscheidbar sind.[4] Letztere lassen sich wie folgt beschreiben:

> Wie alle Behälter sind auch Postkörbe dadurch gekennzeichnet, daß sie explizit Material aufnehmen und herausgeben können (vgl. Abschnitt 2.3.1). In aller Regel *entnehmen* Benutzer einem Korb für eingehende Post *selektiv* neu zugegangene Materialien und legen sie sich nach eigenen Ordnungsprinzipien auf ihrem (Software-) Schreibtisch zurecht (z.B. auf einem aktuellen Arbeitsstapel). Weiterzureichende Materialien *fügen* sie einem Korb für ausgehende Post *hinzu*, aus dem sie später von einem Postdienst entnommen werden.
>
> In jeder Softwareumgebung können prinzipiell beliebig viele Posteingangs- und Postausgangskörbe bereitstehen. Dies ergibt insofern einen Sinn, als daß wir aus dem Bürobereich gewohnt sind, Ein- und Ausgangskörbe für den Postverkehr mit anderen Arbeitsplätzen zu *reservieren*. So kann es an dem Arbeitsplatz des Revierleiters einen Eingangskorb geben, der ausschließlich für Postzugang des Betriebsleiters bestimmt ist. Ein- und Ausgangskörbe sind in diesem Sinne sowohl als allgemeine Postkörbe als auch

[4] Der Unterschied zwischen Eingangs- und Ausgangskörben besteht vielmehr darin, *wer* Materialien entnehmen und einfügen darf. Während der Postdienst in die Eingangskörbe nur Materialien einfügen bzw. aus den Ausgangskörben nur Materialien entnehmen darf, unterliegt der Benutzer im Umgang mit seinen eigenen Postkörben keinerlei Einschränkungen. Solche "Zugriffsrechte" entziehen sich der Modellierung durch Umgangsformen und werden auf einer anderen Ebene behandelt werden (vgl. Abschnitt 3.3.4).

zum Aufbau einer (gerichteten) *rohrpostartigen* Verbindung zwischen zwei Softwareumgebungen verwendbar. Das Einrichten solcher Postkörbe liegt in der Verantwortlichkeit des Benutzers und ist Teil der "Ordnung", die er sich auf seinem elektronischem Schreibtisch schafft.

Postkörbe werden *beschriftet*; sie können auf diese Weise innerhalb der einzelnen Softwareumgebung voneinander unterschieden werden. Oft gibt die Beschriftung eines Postkorbes Auskunft über seinen Verwendungszweck (z.B. *Post für den Betriebsleiter*).

Die Modellierung von Ein- und Ausgangskörben in Softwaresystemen ist prinzipiell keine neue Erfindung; sie findet sich bereits in verschiedenen Systemen zur Steuerung von Büroabläufen ("Büroprozedursysteme", "Vorgangssteuerungssysteme"). Oft ist das Bild jedoch brüchig, weil sich die Handhabung der elektronischen Postkörbe nicht an ihren realen Umgangsformen orientiert. Dies läßt sich beispielhaft an dem von Karbe et al. entwickelten System ProMInanD verdeutlichen, das ein grundlegend anderes Verständnis von elektronischen Postkörben, hier insbesondere von Ausgangskörben, reflektiert [KRW90].

ProMInaD unterstützt die arbeitsteilige Bürokooperation durch die Bereitstellung von elektronischen Umlaufmappen, die beliebige Zusammenstellungen von Dokumenten beinhalten. Geplante Bearbeitungswege werden an der Umlaufmappe selbst notiert und ermöglichen so ihren automatischen Transport durch die Organisation. Nach ihrer Bearbeitung gelangt eine Umlaufmappe auf Veranlassung des Benutzers in seinen elektronischen Ausgangskorb (out-tray).

Ausgangskörbe implementieren in ProMInanD sowohl *behälter-* als auch *werkzeugartige* Funktionalität. Ihr Behältercharakter kann darin gesehen werden, daß sie zur Sammlung und Verwaltung aller Umlaufmappen verwendet werden, die den elektronischen Schreibtisch des Benutzers bereits verlassen haben, aber insgesamt noch nicht fertig bearbeitet sind. Ihr Werkzeugcharakter ermöglicht einem Benutzer, sich über den gegenwärtigen Ort von Umlaufmappen zu informieren und noch nicht in Bearbeitung genommene Umlaufmappen aus dem Eingangskorb eines anderen Benutzers zurück auf den eigenen Schreibtisch zu holen. Im Ergebnis führt diese Modellierung somit zu einem technischen Artefakt, das zwar nach einem realen Bürobehälter benannt wurde, sich jedoch nicht wie dieser handhaben läßt.

3.3.2 Der Versandautomat

Im täglichen Leben begegnen wir einer Vielzahl von Situationen, in denen wir den Transport von Materialien veranlassen, aber selbst nicht aktiv am Prozeß des Transportierens beteiligt sind. Wir adressieren Briefe und werfen sie in einen Briefkasten, legen Dokumentmappen in unseren Postausgang oder verpacken Schriftstücke und kleinere Gegenstände in Rohrpostbüchsen. In all diesen Fällen nehmen Post- und Paketdienste oder technische Förderanlagen uns die eigentliche Transportarbeit ab.

Dieses Prinzip soll auf die Softwareumgebungen übertragen werden. Für den Transport von arbeitsteilig genutztem Material zwischen Umgebungen wird daher ein `Versandautomat` bereitgestellt. Grundlage für die Modellierung ist die Automaten-Metapher von WAM (vgl. Abschnitt 2.3.3). Der Versandautomat wird in einstellbaren Zeitabständen im Hintergrund aktiv und verarbeitet über einen prinzipiell beliebig langen Zeitraum auf der Basis seines fachlichen Algorithmus automatisch, d.h. ohne menschliche Eingriffe, zu transportierende Materialströme.

Der fachliche Algorithmus läßt sich wie folgt beschreiben: Der Versandautomat kennt die von ihm zu bedienenden Postkörbe. In regelmäßigen Zeitabständen leert er die Postausgangskörbe der verschiedenen Softwareumgebungen, interpretiert die Empfängeradressen der entnommenen Materialien und transportiert sie in den Posteingangskorb ihres nächsten Empfängers.

Die Empfängeradresse läßt sich als ein durch das Material referenziertes Objekt verstehen, das den Bezug zu einer anderen Softwareumgebung und darüber auch zu ihren (lokalen) Posteingangskörben herstellt. Transportierbare Materialien müssen dem Versandautomaten auf Anfrage ihre Empfängeradresse bereitstellen. Dies erst ermöglicht dem Automaten, den passenden Eingangskorb für ein Material zu finden. Im Entwurf wird dieser Zusammenhang durch einen *Aspekt* zum Ausdruck gebracht, der im folgenden Abschnitt näher erläutert wird.

3.3.3 Der Aspekt Transportierbar

Das Zusammenspiel zwischen dem Versandautomaten und den zu transportierenden Materialien wird durch den Aspekt `Transportierbar` beschrieben. Der Aspekt formalisiert die minimale Beziehung zwischen Versandautomat und Material, die für ihr "Zusammenpassen", d.h. das Funktionieren des Versandautomaten garantiert: transportierbare Materialien müssen auf Anfrage die Adresse ihres nächsten Empfängers bereitstellen.

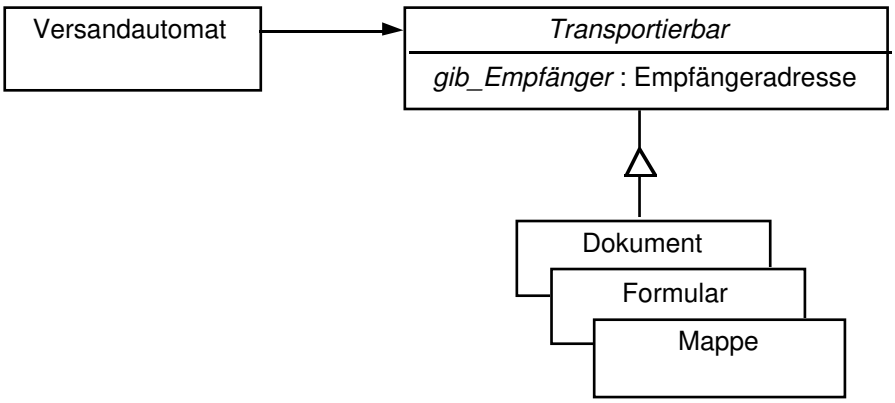


Abb. 3-2: Das Zusammenspiel zwischen dem Versandautomaten und zu transportierendem Material.

Die Verwendung von Aspekten läßt sich allgemein mit der herbeigeführten Entkopplung von Werkzeugen bzw. Automaten und den von ihnen bearbeiteten Materialien motivieren (vgl. Abschnitt 2.3.1). In unserem Fall besteht die Entkopplung zum einen darin, daß ein arbeitsteilig genutztes Material, das gegebenenfalls mit den verschiedensten Werkzeugen anwen-

dungsfachlich bearbeitet werden kann, für den Materialtransport nicht seine volle Schnittstelle offenlegen muß. Tatsächlich ist für das Funktionieren des Versandautomaten nur ein Ausschnitt der fachlichen Funktionalität relevant, die die transportierten Materialien insgesamt an ihrer Schnittstelle anbieten: auf Anforderung sollen sie ihre Empfängeradresse bereitstellen. Daß der Inhalt einer Mappe darüberhinaus auflistbar ist, ein Formular ausgefüllt und ein Dokument editiert werden kann, ist für den Verwendungszusammenhang des Materialtransports unerheblich. Um dieses auch im Entwurf explizit zu machen, wird die relevante Funktionalität als Protokoll abstrahiert und in einer eigenständigen Entwurfskomponente, der Aspektklasse `Transportierbar`, beschrieben. Aspektklassen werden generell als abstrakte Klasse definiert, d.h. sie spezifizieren ein Protokoll, ohne es durch eine Implementation einzulösen. Materialien, die mittels des Versandautomaten zwischen zwei Umgebungen ausgetauscht werden sollen, müssen diesem Protokoll genügen. Zu diesem Zweck erben sie von der Aspektklasse und implementieren ihre Schnittstelle.

In anderer, implizit bereits erwähnter Hinsicht bedeutet Entkopplung darüberhinaus, daß der Versandautomat beliebige Materialtypen (Dokument, Formular, Mappe) verarbeiten kann, solange diese nur das `Transportierbar`-Protokoll implementieren. Der Transport beliebiger Materialien ist für den weiteren Verlauf der Arbeit jedoch nicht von Bedeutung und soll daher auch nicht weiter ausgelotet werden. Wir gehen vielmehr davon aus, daß Materialien, die im Rahmen rechnergestützter Kooperationen zwischen Softwareumgebungen ausgetauscht werden sollen, sich sinnvoll in aufgabenbezogenen Transportbehältern zusammenstellen und zusammenhalten lassen. Sie werden im folgenden Abschnitt erläutert.

3.3.4 Transportbehälter

Materialien, die im Rahmen der kooperativen Aufgabenbearbeitung zwischen zwei Softwareumgebungen auszutauschen sind, werden in aufgabenbezogenen Behältern zusammengestellt, in denen sie sich von dem Versandautomaten transportieren und ihrem nächsten Empfänger zustellen lassen. Abbildung 3-3 zeigt die Umgangsformen eines Transportbehälters.

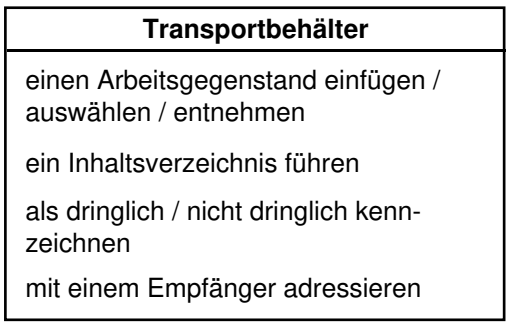


Abb. 3-3: Umgangsformen der Klasse `Transportbehälter`.

Arbeitsgegenstände können in den Transportbehälter eingefügt, darin ausgewählt und wieder aus dem Behälter entnommen werden; die aktuell in dem Transportbehälter enthaltenen Arbeitsgegenstände werden in einem Inhaltsverzeichnis geführt. Wie das Beschwerde-Szenario beispielhaft gezeigt hat, kann die äußere Form eines Transportbehälters (z.B. seine Farbe) Aufschluß über die Bearbeitungsdringlichkeit der zu erledigenden Aufgabe geben. Im Klassenentwurf bringen wir dies zum Ausdruck, indem wir erlauben, einen Transportbehälter als dringlich bzw. nicht dringlich zu kennzeichnen. Eine charakteristische Eigenschaft von Transportbehältern ist darüberhinaus, daß sie sich adressieren lassen. Damit ihre Objekte von dem Versandautomaten transportiert werden können, erbt die Klasse `Transportbehälter` von dem Aspekt `Transportierbar` und implementiert dessen Protokoll, indem sie auf Anfrage die Adresse ihres nächsten Empfängers zurückliefert (vgl. Abbildung 3-2).

Wir werden das Konzept des Transportbehälters in Kapitel 4 erneut aufgreifen und erweitern, wenn es um die rechnergestützte Koordination komplexer arbeitsteiliger Prozesse geht. In diesem Zusammenhang ergeben sich sowohl für den Versandautomaten als auch für die Transportbehälter zusätzliche Anforderungen, die sich bei dem gezielten Materialaustausch zwischen zwei Softwareumgebungen zunächst nicht stellen. Dabei wird es darum gehen, die mit einer Aufgabe verbundenen Tätigkeiten (z.B. "Erstellung eines informellen Sachstandsberichtes") sowie ihre Einbindung in einen teil- oder vollstrukturierten Ablauf an dem Transportbehälter selbst zu *vergegenständlichen*. Solche "koordinativen" Informationen sind für den Einzelnen derzeit nur indirekt aus dem Inhalt der beigefügten Materialien, Betreff-Feldern oder angehefteten Notizen ersichtlich.

3.3.5 Das Zusammenspiel zwischen Versandautomat und Postkörben

Postkörbe sind lokal zu einer Softwareumgebung, zeichnen sich gegenüber anderen Materialien einer Umgebung jedoch dadurch aus, daß sie "von außen" durch einen Versandautomaten zugreifbar sind (vgl. Abschnitt 3.3.1). Daß dieser Zugriff nicht beliebig sein kann, wurde bereits am Rande erwähnt und soll nun detaillierter ausgeführt werden.

Der Versandautomat kann in einen Eingangskorb Material hineinlegen, darf aber keines daraus entnehmen. Umgekehrt kann der Automat Ausgangskörbe leeren, aber niemals Materialien in sie einfügen. Dieses jeweils spezifische Verständnis von Ein- und Ausgangskörben wird durch den Entwurf des Materials Postkorb *nicht* reflektiert, was darauf zurückgeführt werden kann, daß sich die fachlichen Umgangsformen von Eingangs- und Ausgangskörbe, die bei der Modellierung von Materialklassen bekanntlich im Vordergrund stehen (vgl. Abschnitt 2.3.2), nicht voneinander unterschieden (vgl. Abschnitt 3.3.1).

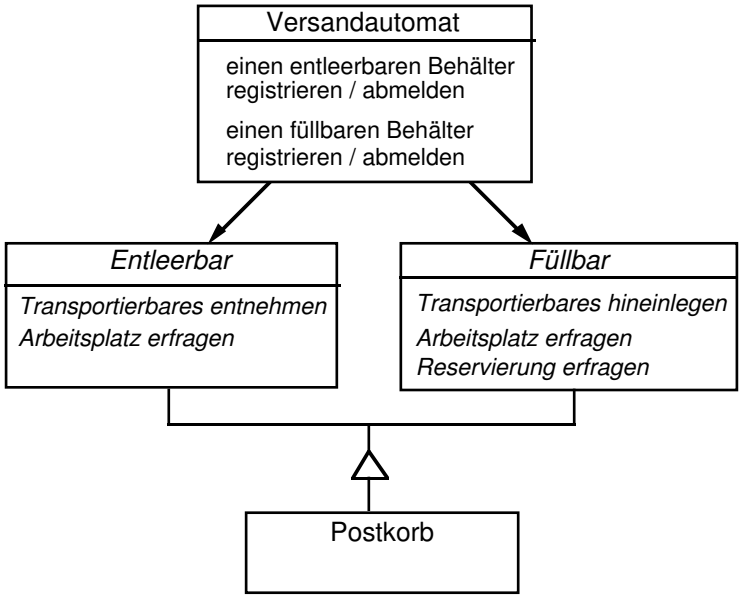


Abb. 3-4: Postkörbe werden dem Versandautomaten unter ihrer abstrakten Schnittstelle `Entleerbar` bzw. `Füllbar` bekannt gemacht.

Sowohl Eingangs- als auch Ausgangskörbe nehmen transportierbares Material auf und geben es wieder heraus; der Unterschied zwischen beiden besteht lediglich darin, *wer* die Materialien entnehmen darf. Während Benutzer mit den ihnen zur Verfügung stehenden Werkzeugen den Inhalt ihrer eigenen Postkörbe prinzipiell beliebig manipulieren können, sind für den Versandautomaten klare Grenzen aufgezeigt worden.

Um die Zugriffsmöglichkeiten des Versandautomaten auf die von ihm zu bedienenden Postkörbe im Entwurf zu modellieren, schränken wir seine Sicht auf das Material mit Hilfe der Aspekte `Entleerbar` bzw. `Füllbar` ein. Ausgangskörbe, die von dem Automaten geleert werden sollen, werden ihm unter ihrer abstrakten Schnittstelle `Entleerbar` bekannt gemacht.

Eingangskörbe, in die der Versandautomat Materialien hineinlegen soll, werden bei ihm unter der Aspekt-Schnittstelle `Füllbar` registriert (vgl. Abbildung 3-4).

Das Registrieren von Postkörben kann als ein Parameter in der Einstellbarkeit des Versandautomaten verstanden werden. Die Einstellung erfolgt konzeptionell über ein Einstellwerkzeug, das Benutzenden innerhalb ihrer Softwareumgebung zur Verfügung gestellt wird. Mit ihm können Postkörbe bei dem Versandautomaten *angemeldet*, aber auch wieder *abgemeldet* werden. Das Einstellwerkzeug benutzt die entsprechenden Operationen, die der Versandautomat an seiner Schnittstelle bereitstellt (vgl. Abbildung 3-4).[5]

Das `Entleerbar`-Protokoll besteht konzeptuell aus zwei Operationen, mit dem der Versandautomat zu transportierende Materialien entnehmen sowie den Ort, d.h. die Softwareumgebung, des entleerbaren Behälters erfragen kann. Letztere Information verwendet der Automat, um den jeweils *passenden* Eingangskorb des nächsten Empfängers zu ermitteln. Was damit gemeint ist, läßt sich am Beschwerde-Szenario veranschaulichen.

Nehmen wir an, der Betriebsleiter will dem zuständigen Revierleiter eine Beschwerdemappe

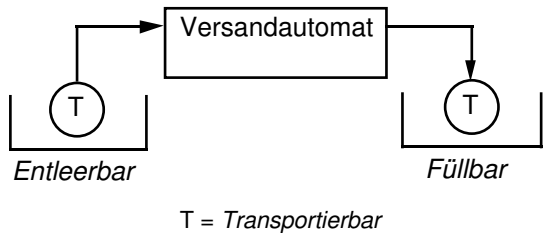


Abb. 3-5: Der Versandautomat entnimmt einem *entleerbaren* Behälter transportierbares Material und überstellt es in einen *füllbaren* Behälter des nächsten Empfängers; dieser wird von dem Versandautomaten anhand der Empfängeradresse des transportierbaren Materials ermittelt.

---

5 Bei der Entwicklung des Prototyps (vgl. Kapitel 6) wurde der Einfachheit halber auf die Implementierung eines separaten Einstellwerkzeuges verzichtet. Postkörbe werden dort automatisch nach ihrer Erzeugung durch den Benutzer bei dem Versandautomaten registriert und vor ihrer Zerstörung wieder abgemeldet.

überstellen, der mehrere Körbe für eingehende Post auf seinem elektronischen Schreibtisch eingerichtet und einen davon für Postzugang vom Betriebsleiter reserviert hat. Der Versandautomat muß nun dafür Sorge tragen, daß die Beschwerdemappe nach ihrer Entnahme aus dem

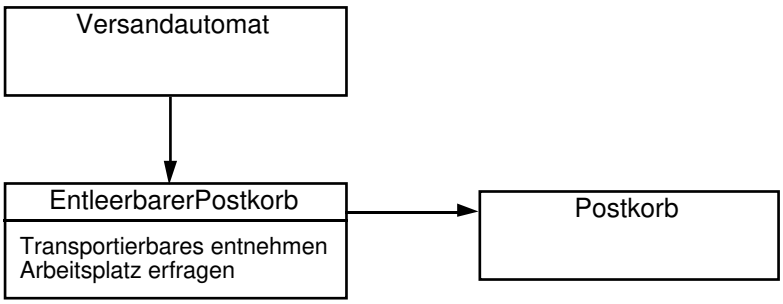


Abb. 3-6: `Postkorb`-Objekten wird durch Umwicklung mit einem `EntleerbarerPostkorb`-Objekt dynamisch die Eigenschaft zugeordnet, von dem Versandautomaten entleerbar zu sein.

Postausgangskorb des Betriebsleiters nicht in den allgemeinen Posteingang des Revierleiters gelangt, sondern vielmehr in den von ihm vorgesehenen Eingangskorb transportiert wird. Bei der Entleerung eines Ausgangskorbes ermittelt der Automat daher zunächst die Softwareumgebung, der der Korb zugeordnet ist und prüft dann, ob der nächste Empfänger einen Eingangskorb bei ihm registriert hat, der speziell für den Postzugang aus dieser Softwareumgebung reserviert wurde. Über das `Füllbar`-Protokoll kann der Versandautomat daher nicht nur den Ort eines Eingangskorbes feststellen und neue Materialien hineinlegen, sondern auch seine eventuelle Reservierung erfragen. Sämtliche Operationen des Protokolls werden in den Aspektklassen `Entleerbar` und `Füllbar` abstrakt spezifiziert und von der Materialklasse `Postkorb` implementiert.

Problematisch erscheint an der beschriebenen Konstruktion, daß einzelne `Postkorb`-Objekte via Mehrfachvererbung gleichzeitig von dem Versandautomaten `Füllbar` und `Entleerbar` zu sein scheinen (vgl. Abbildung 3-4). Der statische Klassenentwurf reflektiert demnach nicht, daß ein `Postkorb`-Objekt *entweder* als entleerbarer *oder* als füllbarer Behälter bei dem Versandautomaten registriert ist. Im folgenden soll eine mögliche Antwort auf diese Problematik vorgestellt werden.

In der objektorientierten Programmierung werden einer Klasse durch Vererbung die Eigenschaften ihrer Basisklasse(n) statisch zugeordnet. Vererbung ist jedoch nicht die einzige Möglichkeit für die Zuordnung von Eigenschaften; eine Alternative besteht im Einsatz von Objektkomposition. Eine Klassenart, die sich speziell für diesen Zweck eignet, sind die sog. *Umwicklerklassen* oder *Wrapper*; Gamma et al. bezeichnen sie auch als *Decorator* (vgl. [Gam92, GHJ+95]). Umwicklerklassen ermöglichen es, einem individuellen Objekt dynamisch ein bestimmtes Verhalten zuzuordnen, d.h. es an einen neuen Kontext anzupassen.

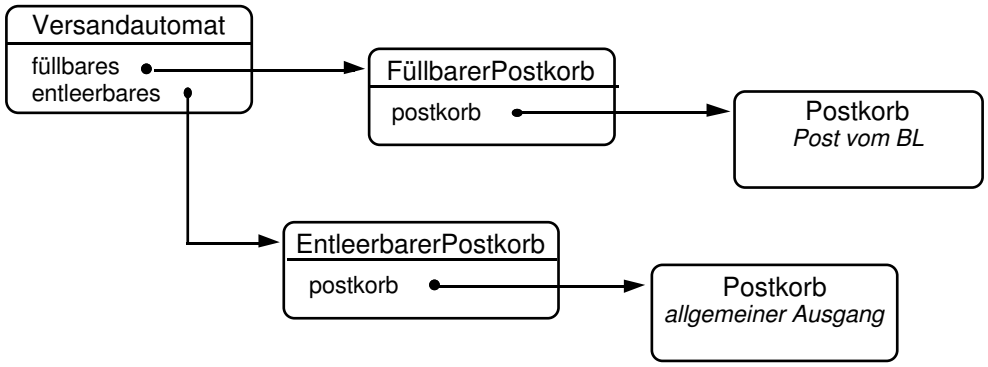


Abb. 3-7: Die dynamische Objektbeziehung reflektiert, daß einzelne `Postkorb`-Objekte entweder als entleerbare oder als füllbare Postkörbe bei dem Versandautomaten registriert sind.

In Abbildung 3-6 stellt die Klasse `EntleerbarerPostkorb` eine Umwicklerklasse für Objekte der Klasse `Postkorb` dar. `Postkorb`-Objekten wird durch Umwicklung mit einem `EntleerbarerPostkorb`-Objekt dynamisch die Eigenschaft zugeordnet, von dem Versandautomaten entleerbar zu sein. Im Unterschied zur Aspektklasse `Entleerbar` aus Abbildung 3-4 ist `EntleerbarerPostkorb` nun keine abstrakte Klasse mehr; ihre Objekte implementieren die an der Schnittstelle spezifizierten Operationen, indem sie sie an das von ihnen umwickelte `Postkorb`-Objekt weiterleiten. Analog kann einem `Postkorb`-Objekt via Umwicklung mit einem `FüllbarerPostkorb`-Objekt dynamisch die Eigenschaft zugeordnet werden, von dem Versandautomaten füllbar zu sein. Wenngleich ein `Postkorb`-Objekt sich prinzipiell sowohl entleeren als auch füllen ließe, können wir mit dieser Konstruktion zum Ausdruck bringen, daß es zu einem Zeitpunkt entweder als entleerbarer oder als füllbarer Behälter bei dem Versandautomaten registriert ist. Abbildung 3-7 zeigt eine beispielhafte dynamische Objektbeziehung.

Abschließend sei noch angemerkt, daß Bezeichnungen wie *Materialtransport* oder *Materialaustausch* natürlich in einem bildhaften Sinne verwendet werden; hier geht es nicht um die

Bewegung physikalischer Datenblöcke. Die Modellierung einer rechnergestützten Software-Arbeitsumgebung bringt unvermeidlich eine Umsetzung räumlicher Organisationsformen in logische Dimensionen mit sich. Räumliche Beziehungen zwischen Arbeitsplatz, Postkörben und Materialien ("steht auf", "liegt in") werden durch logische Beziehungen zwischen der Softwareumgebung, ihren elektronischen Postkörben und den Software-Materialien modelliert. Als Komponenten eines objektorientiert entworfenen Systems sind sie eindeutig identifizierbare Objekte, die über Verweise in ein logisches Beziehungsgeflecht zueinander gesetzt werden. Reale räumliche Veränderungen, wie sie etwa durch den Transport von Arbeitsgegenständen herbeigeführt werden, können in der Softwareumgebung entsprechend durch das Zerstören, Erzeugen oder "Umbiegen" von Objekt-Referenzen nachvollzogen werden, etwa zwischen einem `Postkorb`-Objekt und einem `Transportbehälter`-Objekt. Als eine wesentliche Aufgabe kommt der Softwareumgebung daher zu, den Werkzeugen und Automaten zunächst einmal Zugriff auf die von ihnen bearbeiteten Materialien zu verschaffen, die gegebenenfalls in einem Netzwerk verteilt bzw. in einem Datenbanksystem gespeichert sind. Diese Thematik wird im folgenden Abschnitt näher erläutert.

3.3.6 Die Materialverwaltung

Damit der Versandautomat über einen längeren Zeitraum ohne äußere Eingriffe ablaufen kann, muß er bekanntlich mit einem minimalen Wissen über seinen einbettenden Kontext ausgestattet werden, d.h. er muß die Postkörbe kennen, die er zu bedienen hat und denen er in regelmäßigen Abständen transportierbare Materialien entnimmt oder hinzufügt. Dies geschieht über die Registrierung der zu bedienenden Postkörbe. Wie jedoch kann der Automat im laufenden System Zugang zu den Postkörben erhalten, die möglicherweise in einem Netzwerk verteilt oder in einem Datenbanksystem gespeichert sind?

Als Lösungansatz wird in [WW94] die Modellierung einer `Materialverwaltung` vorgeschlagen. Hinter dieser Entwurfskomponente steht ein Konzept, das auf die einheitliche Versorgung der Werkzeuge und Automaten einer Umgebung mit Materialien abstellt und dabei die konkrete Herkunft und Lebensdauer der Materialien zu verbergen sucht.

Die `Materialverwaltung` bietet eine ausreichend mächtige Anfrageschnittstelle, über die Materialien spezifiziert und aktiviert, d.h. aus der Datenbank in den Hauptspeicher geladen werden können. Anfragen werden durch die Werkzeuge oder Automaten einer Umgebung an die Materialverwaltung gerichtet. Als Ergebnis von Anfragen werden Materialsammlungen zurückgeliefert, die alle verfügbaren und der Anfrage entsprechenden Materialien enthalten (vgl. Abbildung 3-8).

Der `Materialverwaltung` obliegt die eigentliche Beschaffung, effiziente Verwahrung und Bereitstellung von Materialien innerhalb einer Softwareumgebung. Die Materialbeschaffung

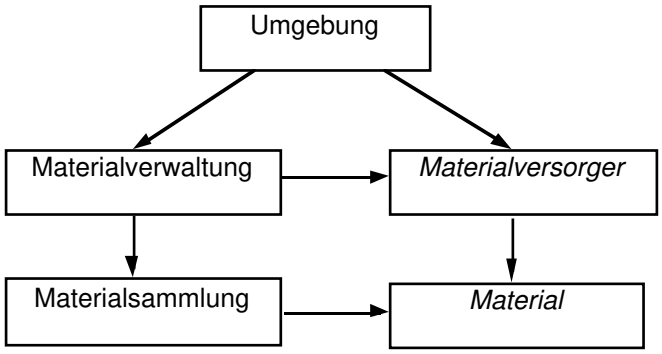


Abb. 3-8: Die Umgebung versieht die `Materialverwaltung` mit den vorhandenen `Materialversorgern`. Aktivierte Materialien werden in Materialsammlungen zusammengestellt.

erfolgt in der Regel am Rand der Umgebung durch Zugriff auf externe Datenspeicher, z.B. objektorientierte, relationale oder hierarchische Datenbanksysteme. Bei diesem Vorgang bedient sich die `Materialverwaltung` selbst weiterer Automaten, die ihr erlauben, von konkreten Datenbankschnittstellen zu abstrahieren. In [WW94] werden diese Automaten gemäß ihrer Funktion als `Materialversorger` bezeichnet und im Kontext der jeweiligen Datenbanken zu konkreten `Datenbankautomaten` spezialisiert.

Aus der Sicht der `Materialverwaltung` kapseln `Materialversorger` jeweils einzelne externe Dienste. Die Anfrageschnittstelle für Materialien wird so entworfen, daß die Inanspruchnahme spezifischer Dienste in der Materialspezifikation selbst nicht explizit gemacht werden muß, sondern alle Materialversorger von der Materialverwaltung einheitlich behandelt werden können. Die Umsetzung der Materialanfragen in datenbankspezifische Anfragen erfolgt eben erst innerhalb der spezialisierten Datenbankautomaten.

In jeder Softwareumgebung können prinzipiell beliebig viele Materialversorger existieren. Im konkreten Anwendungsfall hängt ihre Anzahl davon ab, wieviel externe Dienst insgesamt einzubinden sind. Materialverwaltung und Materialversorger werden gleichermaßen von der Softwareumgebung erzeugt. Bei ihrer Erzeugung zum Zeitpunkt der Systeminitialisierung macht die Umgebung der Materialverwaltung alle existierenden Versorger bekannt. Damit bleibt das Wissen um die Außenwelt bei der Umgebung und hinter den Materialversorgern gekapselt. Letztere sind insbesondere nicht für die Werkzeuge und Automaten einer Umgebung sichtbar.

Materialien, die einmal über einen Materialversorger beschafft worden sind, werden von der Materialverwaltung (zentral) verwahrt und auf Anforderung beliebigen, anfragenden Werkzeugen und Automaten in einer Materialsammlung zur Bearbeitung bereitgestellt. Die Materialsammlung ermöglicht zunächst nur das Sondieren der Materialien. Will ein Werkzeug oder ein Automat dann mit einem Material arbeiten, muß es dieses explizit aus der Materialsammlung entnehmen. Dies geschieht gleichfalls mit Hilfe der Materialverwaltung, die das Material entweder als Original (mit dem Recht zu ändern) oder als Kopie (nur zur Sondierung) herausreicht.

Damit Werkzeuge und Automaten überhaupt Anfragen an die Materialverwaltung formulieren und einzelne Materialien als Originale bzw. Kopien abfordern können, erhält jedes Werkzeug und jeder Automat bei seiner Erzeugung durch die Umgebung einen Verweis auf die Materialverwaltung. Entsprechend *benutzt* der Versandautomat die Materialverwaltung, um Postkörbe, die bei ihm zur Bedienung registriert worden sind, anzufordern und zu bearbeiten, indem er Transportierbares entnimmt bzw. hinzufügt (vgl. Abbildung 3-9).

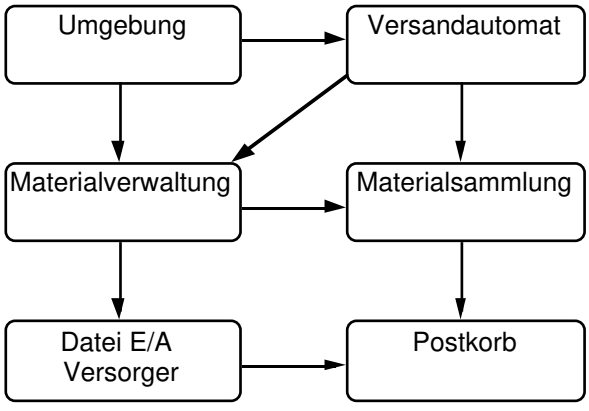


Abb. 3-9: Im Entwicklungsprototypen dient ein einfacher Datei-E/A basiert Materialversorger zum Aktivieren und Passivieren von Materialien.

3.3.7 Der Versandautomat als verteilte Architekturkomponente

Bei der konzeptionellen Darstellung des Versandautomaten haben wir bisher davon abstrahiert, daß seine korrespondierende technische Komponente zur Laufzeit in einem zentralen Prozeß angesiedelt werden muß, der unabhängig von den einzelnen Software-Arbeitsumgebungen läuft. Dies ist insofern notwendig, als daß sonst nicht aus *verschiedenen* Softwareumgebungen heraus auf den Automaten zugegriffen werden könnte, etwa um über ein Einstellwerkzeug die von ihm zu bedienenden Postkörbe an- und abzumelden (vgl. Abschnitt 3.3.5). Zudem ergibt sich die zentrale Anordnung des Versandautomaten recht natürlich aus seiner Aufgabe, Materialien *zwischen* verschiedenen Umgebungen zu transportieren. In diesem Abschnitt soll skizziert werden, wie der Versandautomat als verteilte Architekturkomponente entworfen werden kann.

Eine tragfähige Technik bei der Konstruktion verteilter Softwarearchitekturen ist das sog. Dienstleistungsprinzip (*Client/Services*-Prinzip, vgl. [KGZ93], S. 49). Einzelne Architekturkomponenten realisieren eine festgelegte implementations- und *ortsunabhängige* Dienstleistung, deren Schnittstelle sie als Dienstleistungsanbieter nach außen exportieren und so beliebigen Leistungsabnehmern verfügbar machen. Ein Leistungsabnehmer der Dienste des Versandautomaten ist beispielsweise das Einstellwerkzeug, mit dem aus den einzelnen Softwareumgebungen heraus Postkörbe bei dem Automaten registriert werden können. Die Beziehung zwischen einem Leistungsanbieter und einem Leistungsabnehmer wird im technischen Entwurf durch eine *Benutzt*-Beziehung zum Ausdruck gebracht, d.h. der Abnehmer benutzt die an der Schnittstelle des Anbieters festgelegten Dienste. Durch die Verteilung der beiden Komponenten auf voneinander unabhängige Prozeßräume entstehen im Zuge des Leistungsaustausches Kommunikationsprobleme, die sich nur unter Rückgriff auf die Mechanismen der Inter-Prozeß-Kommunikation bewältigen lassen.

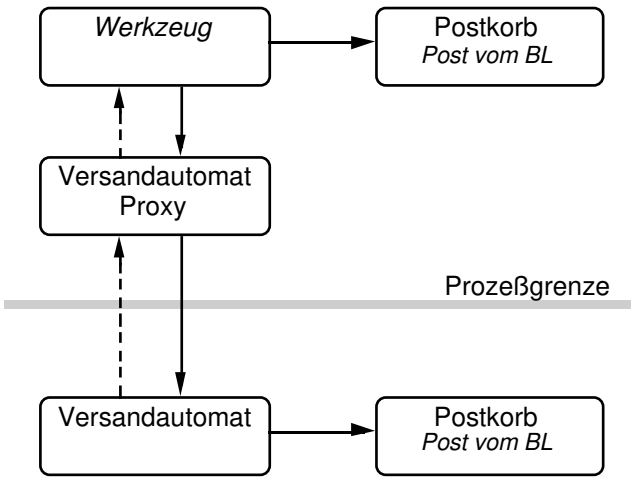


Abb. 3-10: Ein Werkzeug meldet sich als (anonymi-sierter) Beobachter bei dem Proxy an und kann dann von diesem über Änderungen an den Postkörben benachrichtigt werden. Der Versandautomat und das Werkzeug erhalten ihren Zugriff auf die Postkörbe mit Hilfe der lokalen Materialverwaltung.

Es scheint sinnvoll, notwendige Kommunikationsvorgänge, die sich auf unterster Ebene etwa durch Socketverbindungen oder RPC-Calls realisieren lassen, möglichst frühzeitig im Entwurf zu kapseln, um davon auf höheren Ebenen abstrahieren und Konzepte gewissermaßen über Prozeßränder hinaus modellieren zu können. Der Grundgedanke ist daher, das fachliche Konzept des Automaten nicht 1:1 in eine technische Komponente zu überführen, sondern auf der Seite der Softwareumgebungen mit sog. Stellvertreter-Objekten (*Proxies*) zu arbeiten, die den Versandautomaten lokal repräsentieren.

Gamma et al. diskutieren solche Stellvertreter-Objekte unter dem Begriff "remote proxy"; Coplien nennt sie "Ambassador-Objekte" (vgl. [GHJ+95, Cop92])[6]. Für das `Versand-automat`-Proxy sind folgende Eigenschaften charakteristisch:

> Das Proxy ist ein lokaler Repräsentant der Versandautomaten, d.h. er verbirgt vor den Werkzeugen der Softwareumgebung, daß der Automat in einem anderen Adreßraum angesiedelt ist.

---

[6] Proxies werden auch in der gegenwärtigen Diskussion um Standards für verteilte, objektorientierte Anwendungen hoch gehandelt. Die Diskussion wird maßgeblich von den Bemühungen der *Object Management Group* (OMG) beeinflußt, die sich die Schaffung eines Industriestandards für die Verwaltung und den Zugriff auf netzwerkweit verteilte Objekte zum Ziel gesetzt hat (vgl. [OMG92, OMG92]). Zentrales Element des CORBA-Standards ist der sogenannte *Object Request Broker* (ORB), ein Serverprozeß auf Betriebssystemebene, mit deren Hilfe sich Anwendungssysteme Proxies beschaffen, um auf beliebige, im Netz verteilte Objekte zuzugreifen. Operationenaufrufe werden über den Broker an die "realen" Objekte weitergeleitet, der u.a. für das Auffinden von verteilten Objekten im Netz verantwortlich ist.

Das Proxy bietet nur solche Operationen an, die auch der Versandautomat an seiner Schnittstelle exportiert, und ist dafür verantwortlich, Operationenaufrufe zusammen mit ihren Parametern geeignet zu kodieren und dem Versandautomaten zuzusenden. Umgekehrt müssen die Rückgabewerte des Versandautomaten von dem Proxy entschlüsselt und in aufbereiteter Form den anfragenden Werkzeugen zurückgegeben werden.

Werkzeuge erhalten ausschließlich über das Proxy Zugriff auf den "echten" Versandautomaten.

Die Werkzeuge der Softwareumgebung benutzen das Proxy-Objekt, um Dienste des Versandautomaten in Anspruch zu nehmen. Einige von ihnen *beobachten* das Proxy-Objekt darüberhinaus und werden von ihm durch ein entsprechendes Ereignis *benachrichtigt*, wenn der Versandautomat den Zustand eines Postkorbes, beispielsweise durch Materialentnahme, verändert hat. Dieser *Ereignismechanismus* ist etwa für solche Werkzeuge von Bedeutung, mit denen Benutzende sich den Inhalt ihrer Postkörben anzeigen lassen. Diese Werkzeuge müssen über Änderungen an einem Postkorb benachrichtigt werden, damit sie die Darstellung seines (von außen) geänderten Inhaltes an der Benutzungsschnittstelle aktualisieren können. Wir werden auf diesen Punkt noch einmal ausführlicher in Kapitel 6 zurückkommen, wenn die wesentlichen Techniken zur Werkzeugkonstruktion vorgestellt werden (vgl. Abschnitt 6.4). Die in diesem Abschnitt beschriebenen Zusammenhänge werden in Abbildung 3-10 veranschaulicht.

3.4 Zusammenfassung

Abschließend zu diesem Kapitel soll der wesentliche Argumentationsgang noch einmal stichpunktartig nachvollzogen werden:

Ausgehend von dem *Szenario* "Bearbeitung einer Kundenbeschwerde" wurden zunächst charakteristische Merkmale der routinisierten, arbeitsteiligen 4-Augen-Kooperation herausgearbeitet.

Sie führten zu der *These*, daß zwei kooperierende Personen ihre Zusammenarbeit in erheblichem Maße auf der Basis gemeinsamer Materialien koordinieren, die sie als Bestandteil von erzielten bzw. noch zu erzielenden Arbeitsergebnissen untereinander austauschen.

Vor diesem Hintergrund wurde das *Ziel* formuliert, in einer Softwareumgebung unterstützende Bausteine bereitzustellen, die zwei kooperierenden Benutzern den koordinierten, rechnergestützten Austausch von Materialien ermöglichen.

Als *Lösung* wurde vorgeschlagen, die Materialien in aufgabenbezogenen, adressierbaren Transportbehältern zusammenzustellen, in denen sie zwischen den kooperierenden Softwareumgebungen weitergereicht werden können. Es ließ sich argumentieren, daß der Materialtranport weitestgehend automatisiert werden kann, wenn explizite Orte vereinbart werden, an denen die Benutzer Materialien in Transportbehältern zur Bearbeitung entgegennehmen bzw. zur Weitergabe an den Kooperationspartner bereitlegen. Zu diesem Zweck wurden elektronische Posteingangskörbe bzw. Postausgangskörbe eingeführt; sie werden von einem Versandautomaten bedient, der die Ausgangskörbe leert und die daraus entnommenen Transportbehälter automatisch in den passenden Eingangskorb ihres nächsten Empfängers transportiert.

Die in diesem Kapitel vorgestellten Entwurfskomponenten bilden die Grundlage für die weitere Entwicklung der Softwareumgebung. Sie werden im folgenden Kapitel um neue Bausteine ergänzt, die Benutzer über den gezielten Materialaustausch hinaus bei der rechnergestützten Koordination ihrer (nicht notwendigerweise auf zwei Personen beschränkten) arbeitsteiligen Prozesse unterstützen.

4 Die rechnergestützte Koordination routinisierter, arbeitsteiliger Prozesse

Im zurückliegenden Kapitel wurde die routinisierte arbeitsteilige Zusammenarbeit von zwei Personen betrachtet. Sie kann als ein Spezialfall verstanden werden und soll im folgenden in einen allgemeineren Zusammenhang gestellt werden. Gegenstand dieses Kapitels sind Arbeitssituationen, in denen größere Personengruppen an der routinemäßigen Bearbeitung einer gemeinsamen Aufgabe mitwirken. In diesem Zusammenhang stellt sich in verstärktem Maße das Problem der *arbeitsplatzübergreifenden Koordination*. Charakteristisch für diese Form der Koordination ist die Festlegung von einzelnen Bearbeitungsschritten, ihren Abhängigkeiten und Zuständigkeiten. Ziel dieses Kapitels ist darzulegen, welche Voraussetzungen in einer Softwareumgebung geschaffen werden müssen, damit Benutzende über den gezielten Materialaustausch hinaus (vgl. Kapitel 3) bei der Koordination ihrer arbeitsteiligen Prozesse unterstützt werden können.

Als Anschauungsmaterial für die weitere Diskussion dient ein Szenario, das einen typischen Vorgang in einer Bank, die Vergabe eines Kundenkredits, beschreibt. Ausgehend von diesem Beispiel soll im Anschluß daran gezeigt werden, welche Rolle *Plänen* bei der Koordination von Tätigkeiten in arbeitsteiligen Prozessen zukommen kann. Verschiedene Alternativen aus dem CSCW-Bereich, dem Gebiet des Workflow Management und der Arbeitspsychologie werden dargestellt und vor dem Hintergrund des hier zugrundegelegten Leitbildes (vgl. Abschnitt 2.2) ausgelotet. Dies geschieht mit dem Ziel, Konzepte für die Gestaltung und Verwendung von Koordinationsplänen herauszuarbeiten, die für die Softwareumgebung fruchtbar gemacht werden können und sich schließlich in der Modellierung sogenannter *Prozeßmuster* verfestigen.[7]

4.1 Szenario: Kreditvergabe in einer Bank

Für die Kreditvergabe ist ein Kundenberater zuständig. Er führt das Gespräch mit dem Kunden und füllt im Anschluß daran ein entsprechendes Kreditantragsformular aus. Die Abwicklung der Kreditvergabe erfordert die routinemäßige Zusammenarbeit mit weiteren Mitarbeitern der Bank. In der Regel fallen hierbei verschiedene voneinander abzugrenzende Bearbeitungsschritte an.

Zum einen muß der ausgefüllte Kreditantrag genehmigt werden. Für die Genehmigung des Kredits zieht der zuständige Kundenberater einen weiteren Berater hinzu. Dieser kann den Antrag jedoch nicht genehmigen, bevor nicht die Sicherheiten des Kunden eingeholt worden sind. Diese Tätigkeit kann von einem Sachbearbeiter erledigt werden, wenn ihm der ausgefüllte Kreditantrag vorliegt. Liegen alle Sicherheiten vor und hat der zweite Kundenberater daraufhin die verlangte Genehmigung erteilt, muß noch eine Restkreditversicherung abgeschlossen werden. Sie stellt sicher, daß der Kredit auch dann an die Bank zurückgezahlt wird, wenn der Kreditnehmer zahlungsunfähig wird. Der Abschluß der Restkreditversicherung wird in der Regel vom zuständigen Kundenberater selbst vorgenommen. Auch hierfür ist Voraussetzung,

---

[7] Die in diesem Kapitel vorgestellten Konzepte gehen maßgeblich auf die Arbeit von G. Gryczan zurück (vgl. [Gry95]). Hier wird auch der Begriff des *Prozeßmusters* eingeführt und motiviert. Auf die dahinterstehenden Konzepte, die ausführlich in [Gry95] nachgelesen werden können, wird an späterer Stelle noch zurückzukommen sein.

daß die Höhe des beantragten Kredits, die beispielsweise aus dem ausgefüllten Kreditantragsformular hervorgeht, sowie andere Sicherheiten, die bereits einen Teil des Kreditrisikos abdecken, bekannt sind. Davon unabhängig erfolgt das Anlegen eines Kontos zur Auszahlung des Kredits. Diese Tätigkeit wird von einem Sachbearbeiter erledigt und ist prinzipiell zu einem beliebigen Zeitpunkt während der Kreditvergabe möglich.

Wurde die Restkreditversicherung erfolgreich abgeschlossen, kann der Kredit vom zuständigen Kundenberater oder von einem Sachbearbeiter über das angelegte Konto an den Kunden ausgezahlt werden.

4.2 Koordination auf der Basis von Plänen

Das Kreditvergabe-Szenario beschreibt eine routinemäßig zu erledigende Aufgabe, die die Zusammenarbeit mehrerer Mitarbeiter einer Bank (Kundenberater, Sachbearbeiter) erforderlich macht. In der Zusammenarbeit läßt sich ein bestimmtes Muster erkennen, das sich aus den bei der Kreditvergabe anfallenden Tätigkeiten ("Genehmigung erteilen"), ihren anwendungsfachlichen Abhängigkeiten ("Sicherheiten einholen vor Genehmigung erteilen"), benötigten Arbeitsgegenständen ("der Kreditantrag") sowie bestimmten Verantwortlichkeiten ("für die Genehmigung des Kredits muß ein zweiter Kundenberater hinzugezogen werden") zusammensetzt.

Über anfallende Bearbeitungsschritte, potentielle Bearbeitungsfolgen, Zuständigkeiten sowie benötigte Arbeitsgegenstände zu entscheiden, ist inhärenter und notwendiger Bestandteil der kooperativen Aufgabenbearbeitung und wurde in Abschnitt 2.4 unter den Begriff der *Koordination routinisierter, arbeitsteiliger Prozesse* gefaßt (vgl. Def. 2-4). Die koordinativen Anteile bei der Vergabe eines Kundenkredits lassen sich wie folgt charakterisieren:

> Im Rahmen der Zusammenarbeit fallen verschiedene Bearbeitungsschritte oder Tätigkeiten an, die voneinander abgrenzbar sind und benannt werden können: "Sicherheiten einholen", "Kreditgenehmigung erteilen", "Restkreditversicherung abschließen", "Auszahlungskonto anlegen".
>
> Zwischen diesen Tätigkeiten existieren anwendungsfachliche Abhängigkeiten, die bei der Kreditvergabe zu berücksichtigen sind. Zunächst müssen die notwendigen Sicherheiten des Kunden eingeholt werden, dann erst kann die Kreditgenehmigung erteilt werden, die selbst wiederum die Voraussetzung für den erfolgreichen Abschluß der Restkreditversicherung bildet. Im Unterschied hierzu erfolgt das Anlegen des Auszahlungskontos vollkommen unabhängig.
>
> Den einzelnen Tätigkeiten werden zuständige Personen ("Kundenberater X") oder Rollenträger ("ein Kundenberater") zugeordnet.
>
> Für einzelne Tätigkeiten lassen sich Arbeitsgegenstände benennen, die bei ihrer Erledigung benötigt werden. Beispielsweise kann die Kreditgenehmigung nur erfolgen, wenn dem zweiten Kundenberater das ausgefüllte Antragsformular vorliegt. Materialien müssen folglich als Bestandteil von erzielten Arbeitsergebnissen oder zu Informationszwecken zwischen den kooperierenden Personen ausgetauscht werden.

Als Zwischenergebnis können wir festhalten, daß sich die Zusammenarbeit zwischen den verschiedenen Kooperationspartner auch weiterhin an den gemeinsam bearbeiteten Arbeits-

gegenständen orientiert (etwa an dem ausgefüllten Kreditantrag), ihr Austausch jedoch nur *ein* Aspekt der Zusammenarbeit ist.

	Automatisierungssicht	*Unterstützungssicht*
Stellenwert von Koordinationsplänen	Koordinationspläne sind die Vorschriften, nach denen kooperatives Arbeitshandeln idealerweise abläuft und kontrolliert oder ersetzt wird.	Koordinationspläne sind Muster dekontextualisierter Erfahrungen. Sie geben punktuelle Anleitung und Orientierung bei der arbeitsteiligen Erledigung einer gemeinsamen Aufgabe.
Verortung der Kontrolle	Koordinationspläne werden auf einer Maschine implementiert und ausgeführt, die Arbeitsplätze und Arbeitstätigkeiten einander fest räumlich-zeitlich zuordnet. (*externe Handlungskoordination*)	Die an der Zusammenarbeit mitwirkenden Personen stimmen sich untereinander und wechselseitig über Bearbeitungsreihenfolgen und Zuständigkeiten ab. Sie formen und verändern Koordinationspläne unter Berücksichtigung formaler (organisatorischer, gesetzlicher) Bestimmungen. Die Kontrolle der Koordination liegt bei den koordinierten Personen selbst. (*interne Handlungskoordination*)
Stellenwert menschlicher Zusammenarbeit	Menschliche Zusammenarbeit ist grundsätzlich defizitär. Je genauer sie durch einen Koordinationsplan festgelegt und geregelt wird, desto besser läßt sie sich kontrollieren und (teil-) automatisieren.	Menschliche Zusammenarbeit zeichnet sich durch Kreativität, Erfahrung und hohe Fähigkeit zur Einbeziehung des Kontextes aus und ist dem maschinellen Ablauf in vielen Bereichen überlegen. Situatives Handeln wird angestrebt.

Tabelle 4-1: Zwei konträre Sichtweisen bei der rechnergestützten Koordination arbeitsteiliger Prozesse (in Anlehnung an [Gry95], S. 83).

Mit der beschriebenen Form der Koordination werden - genau wie die Definition 2.4 - zunächst noch keinerlei Annahmen bezüglich der *Kontrolle* von Koordination getroffen. Friedrich unterscheidet in diesem Zusammenhang zwischen *interner* und *externer Handlungskoordination* (vgl. [Fri94]). Die interne Koordination setzt voraus, daß Gruppenmitglieder sich

untereinander über einzelne Tätigkeiten, Abhängigkeiten und Zuständigkeiten abstimmen. Die Kontrolle der Koordination liegt also bei den koordinierten Personen selbst. Demgegenüber ist sie im Falle der externen Handlungskoordination bei einer "fremden" Instanz verortet, die Arbeitsplätze und Arbeitstätigkeiten einander fest räumlich-zeitlich zuordnet.

Diese Unterscheidung ist für die weitere Diskussion von großer Bedeutung. In der Literatur finden sich mittlerweile ganz verschiedene Ansätze zur rechnergestützten Koordination routinisierter, arbeitsteiliger Formen der Zusammenarbeit. Einer Vielzahl von ihnen ist gemein, daß sie auf *Koordinationsplänen* basieren, mit denen Tätigkeiten, die zwischen ihnen geltenden Abhängigkeiten, Zuständigkeiten sowie benötigte und erzeugte Arbeitsgegenstände beschrieben werden. Existierende Softwaresysteme und Entwicklungsprototypen machen auf ganz unterschiedliche Weise Gebrauch von solchen Koordinationsplänen. Sie können sinnvoll entlang der in Abschnitt 2.2 aufgezeigten Automatisierungs-Unterstützungs-Dimension diskutiert und eingeordnet werden, da sie sich maßgeblich darin unterscheiden, mit welchen Zielvorstellungen sie bestimmte Anteile von Koordinationsarbeit zu automatisieren versuchen und welchen Stellenwert sie dabei den Plänen - Vorschrift versus Muster - beimessen. In Tabelle 4-1 werden die verschiedenen Wertvorstellungen einander kontrastiv gegenübergestellt.

In den folgenden Abschnitten werden die verschiedenen Möglichkeiten der Verwendung von Koordinationsplänen genauer ausgelotet. Darauf aufbauend läßt sich anschließend die Modellierung der sogenannten *Prozeßmuster* motivieren, die Koordinationspläne vergegenständlichen und als Software-Materialien innerhalb der rechnergestützten Arbeitsumgebung bereitgestellt werden.

4.2.1 Koordinationspläne als implementierbare Vorschriften

Die folgenden Abschnitte stellen die wesentlichen Merkmale zweier Systemtypen, der *Büroprozedursysteme* und der *Workflow Management Systeme*, dar. An ihnen läßt sich beispielhaft demonstrieren, in welcher Weise Pläne bei der rechnergestützten Aufgabenbearbeitung zur Steuerung und Kontrolle der System-Benutzer verwendet werden können. Beide Systemtypen sind Ausdruck einer automatisierenden, ablaufsteuernden Sichtweise (vgl. Abbildung 4-1); ihnen ist gemein, daß sie Pläne als implementierbare und ausführbare Modelle arbeitsteiliger Büroprozesse interpretieren.

4.2.1.1 Büroprozedursysteme

Ein primär vorgangsorientierter Zweig der CSCW-Forschung beschäftigt sich mit der Spezifikation und Implementierung sogenannter *Büroprozedursysteme*. Seit dem Einsetzen der CSCW-Diskussion zu Beginn der achtziger Jahre haben zahlreiche Entwicklungsprojekte zu einer ganzen Fülle einschlägiger Prototypen geführt, die hier unmöglich vollständig aufgeführt werden können. Repräsentative Systeme sind AMIGO (vgl. [Pan89]), DOMINO (vgl. [KLS+84, KHK+91]) und ProMInanD (vgl. [KRW90]).

Den historischen Ansatzpunkt für Büroprozedursysteme (*office procedure systems)* bilden stark-strukturierte und arbeitsteilig zu erledigende Aufgaben, wie sie an sachfallorientierten Arbeitsplätzen mit relativ geringem Dispositionsspielraum und starker Einbindung in formale Ablaufstrukturen anfallen. Diesen Anwendungssystemen liegt ein Verständnis zugrunde, das Kubicek und Höller wie folgt beschreiben (vgl. [KH91], S. 168):

> "Die an einer Büroprozedur beteiligten Benutzer nehmen eine oder mehrere Rollen im Vorgangsvollzug wahr. Sie arbeiten gemeinsam an Datenobjekten. Auf diesen führen sie Operationen durch. Die Zuordnung von Operationen zu Rollen und die Abfolge der Arbeitsschritte werden durch Regeln festgelegt."

Die für Büroprozeduren formulierten Regeln haben nach Kubicek und Höller zwei Funktionen (vgl. [KH91], S. 169):

> *Funktionsregeln* stellen den Zusammenhang zwischen Rollen, Datenobjekten und Operationen her. Sie legen fest, welche organisatorische Rolle welche Operationen ausführen darf bzw. soll, und werden daher oft in die Rollendefinition eingearbeitet. Funktionsregeln können sich stets erst im speziellen Anwendungskontext eines Prozedursystems konkretisieren, da sie Bezug auf ein zugrundeliegendes Organisationsmodell nehmen.
>
> *Ablaufregeln* definieren die Ablaufreihenfolge, nach der Rollenträger Operationen ausführen sollen.

Dem Design von Büroprozedursystemen liegen demnach die Annahmen zugrunde,

> daß sich arbeitsteilige Formen der Zusammenarbeit durch die Spezifikation elementarer (rechnergestützter) Arbeitsschritte und der zwischen ihnen geltenden Ein- und Ausgabebeziehungen in einem organisatorischen Kontext koordinieren lassen,
>
> und daß sie sinnvoll im Rahmen eines formalen mathematischen Modells (etwa mit Hilfe von Petrinetzen) als *Prozeduren* repräsentiert werden können. Im Sinne von Kubicek und Höller können solche Prozeduren als präskriptive Funktions- und Ablaufregeln verstanden werden, die einzelne Arbeitsschritte benennen, sie an zuständige Personen oder Rollenträger binden sowie die in einem Arbeitsschritt benötigten und erzeugten Ressourcen definieren.

Büroprozeduren werden durch ein Softwaresystem *implementiert* und *ausgeführt*. Eine zentrale Steuerung informiert Benutzer über die von ihnen zu erledigenden Arbeitsschritte und sorgt für den automatischen Weitertransport und die Bereitstellung der jeweils benötigten (elektronischen) Materialien. Büroprozeduren sind Koordinationspläne, nach denen - wohlgemerkt im Idealfall - die tatsächlichen Arbeitsprozesse ablaufen sollen (vgl. [KLS+84], S. 34):

> "A coordination procedure is a plan describing how to carry out a certain task, and as such it represents the class of all possible executions of this plan."

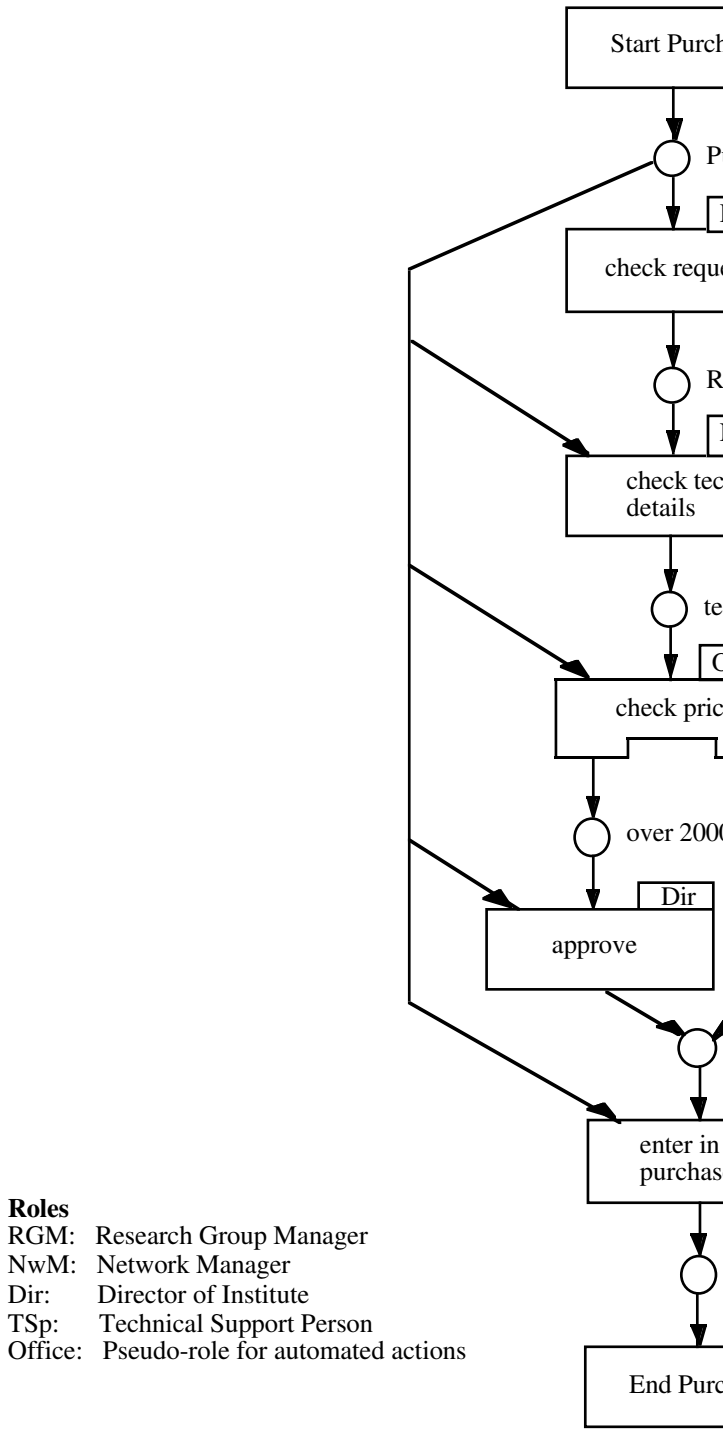


Abb. 4-1: Die Petrinetz-basierte Repräsentation eines Bestellprozesses (Quelle: [KHK+91], S. 119)[8]

Abweichungen vom "idealen" Prozeß werden zwar anerkannt, aber eher als unliebsame Störfaktoren wahrgenommen. Um ihnen zu begegnen, werden sie so vollständig wie möglich

[8] Einzelne Bearbeitungsschritte werden durch rechteckige Kästen, Ressourcen durch Kreise dargestellt. Ein- bzw. ausgehende Pfeile an den Kästen verdeutlichen, welche Ressourcen bei der Erledigung des jeweiligen Schrittes benötigt bzw. erzeugt werden.

antizipiert und im Rahmen einer ausgefeilten Ausnahmebehandlung formalisiert, so daß sie letztenendes selbst wieder zu regelhaften Bestandteilen der Prozeßausführung werden.

Die Automatisierbarkeit des Arbeitsprozesses hofft man zu erreichen, indem man die menschliche (Zusammen-) Arbeit in so feine Teilschritte zerlegt, daß für die am Prozeß Beteiligten kaum noch Handlungs- und Entscheidungsspielraum entsteht. Die läßt sich beispielhaft an der petrinetz-basierten Modellierung eines Bestellprozesses demonstrieren, der während einer Erprobung des DOMINO-Prototyps (vgl. [KLS+84, KHK+91]) bei der Gesellschaft für Mathematik und Datenverarbeitung (GMD) implementiert wurde (vgl. Abbildung 4-1).

Ein typischer Bestellvorgang für Hard- oder Software besteht aus insgesamt vier Arbeitsschritten. Nach dem Ausfüllen eines Standardformulars muß die von einem Angestellten gewünschte Bestellung zunächst vom zuständigen Gruppenleiter (Research Group Manager) unter Berücksichtigung des vorhandenen Budgets genehmigt werden. Liegt die erwartete Genehmigung vor, kann das Bestellformular an den Netzwerkverantwortlichen (Network Manager) des Instituts weitergereicht werden. Er prüft die technischen Details des Bestellgutes bezüglich seiner Verträglichkeit mit bereits vorhandenen technischen Komponenten (z.B. Drucker, Betriebssysteme, Anwendungssoftware) und unter Einhaltung eventueller organisatorischer Bestellrichtlinien. Falls die Bestellkosten einen bestimmten Grenzbetrag überschreiten, muß die Bestellung nach erfolgreicher Prüfung noch zur endgültigen Genehmigung dem Leiter des Forschungsinstituts (Director of Institute) vorgelegt werden. Andernfalls ist die Unterschrift des Gruppenleiters ausreichend. Im letzten Schritt schließlich wird das Bestellformular an einen zuständigen Mitarbeiter des technischen Personals (Technical Support Person) weitergereicht, der für die Budgetverwaltung und die Inventarüberwachung verantwortlich ist. Er erstellt eine Papierkopie des elektronischen Bestellformulars, daß nach seiner Gegenzeichnung das Institut verläßt und zu der für die eigentliche Bestellabwicklung zuständigen Abteilung gelangt.

Typisch für Büroprozedursysteme ist die enge Verknüpfung der Koordination individueller Arbeitsschritte mit deren *inhaltlicher Ausgestaltung*. Die formale Modellierung erfordert eine vollständige Spezifikation des *Kontrollflusses*, d.h. für je zwei elementare Bearbeitungsschritte muß definiert sein, ob sie in einer seriellen, parallelen oder alternativen Ausführungsreihenfolge zueinander stehen. Dies läßt sich beispielsweise mit den von höheren Programmiersprachen bereitgestellten Kontrollstrukturen ausdrücken, findet aber auch seine Entsprechung in der Netztheorie. Solche Kontrollflußspezifikationen werden rasch komplex und undurchschaubar, zumal wenn versucht wird, komplizierteren Arbeitsabläufe durch eine Flut alternativer Kontrollflußbeschreibungen beizukommen. Bei der alternativen Ausführung stellt sich das Problem, daß eine *Ablaufbedingung* darüber entscheidet, welcher Teilschritt nachfolgend abzuarbeiten ist.

Aus dem Beispiel des Bestellprozesses wird etwa ersichtlich, daß der Preis des zu bestellenden Gutes darüber entscheidet, ob eine Bestellanforderung zur endgültigen Genehmigung dem Direktor des Instituts vorgelegt werden muß oder direkt weitergereicht werden kann. Die Formulierung dieser Alternative impliziert zwei Annahmen. Zum einen wird vorausgesetzt, daß Bestellformulare über eine Art Einzelpreis-Feld verfügen; zum anderen muß dieses Feld spätestens bei der technischen Prüfung ausgefüllt werden, da das System ansonsten keine automatische Entscheidung fällen kann. Die enge Kopplung von koordinativen Systemfunktionen und inhaltlichen, d.h. anwendungsfachlichen Aspekten der Aufgabenbearbeitung läßt das System hochgradig unflexibel gegenüber veränderten Randbedingungen reagieren ("der Bestellpreis ist nicht bekannt").

Die Formalisierung bei der inhaltlichen Ausgestaltung von Arbeitsschritten wird in einigen Büroprozedursystemen (etwa ProMInanD, vgl. [KRW90]) noch dadurch vorangetrieben, daß bei der Definition der Arbeitsschritte bereits auf konkrete Applikationen (*step programs*, vgl. [KRW90]) Bezug genommen wird. Sie dienen der Erledigung eines Bearbeitungsschrittes und werden automatisch für den Benutzer gestartet und mit den benötigten Materialien (Dokumenten) geladen. Über alternative Bearbeitungswege entscheiden im Anschluß an die Bearbeitung gegebenenfalls weitere, applikationsabhängige Programme (*application-dependant decision programs*, vgl. [KRW90]).

Von ihrem Ansatz her sind Büroprozedursysteme die Vorläufer heutiger Vorgangsbearbeitungssysteme; sie werden im folgenden Abschnitt vorgestellt.

4.2.1.2 Workflow Management Systeme

Die Anforderungen an Unternehmen werden maßgeblich durch die Marktsituation und das Kundenverhalten geprägt. Mit der Globalisierung und Internationalisierung der Märkte wird der Konkurrenzdruck auf Unternehmen zunehmend stärker, zumal sich in den einzelnen Branchen eine fortschreitende Konzentration von wirtschaftlicher Macht in Form von internationalen Großkonzernen mit einem weit diversifizierten Produktangebot abzeichnet. Kunden treten zunehmend kostenbewußter und anspruchsvoller auf und fordern eine stärkere Kundenorientierung von Unternehmen. Dienstleistungsorientierte Firmen streben daher die verbesserte Nutzung ihrer Produktivitätsreserven und eine strategische Neuausrichtung an, um ihre Wettbewerbsfähigkeit zu steigern. In diesem Zusammenhang wird immer häufiger das Schlagwort "Business Process Reengineering" in die Diskussion gebracht. Geschäftliche Prozesse sollen überdacht und nötigenfalls radikal umgeformt werden, um Leistungsdaten wie Kosten, Qualität, Service und Geschwindigkeit drastisch zu verbessern (vgl. [Dav93, HC93, HC94]).

Der Schlüssel für eine zukunftsweisende, kundenorientierte Organisationsentwicklung scheint in der spezifischen Bündelung von Unternehmensaktivitäten zu liegen, die auf eine "Rundum-Sachbearbeitung" abstellt und so mit der traditionellen Spartentrennung bricht [KGZ93]. Als Folge dieser organisatorischen Veränderungen erwächst aus unternehmerischer Sicht zunehmend Bedarf an Softwaresystemen, mit denen eine kundenorientierte Sachbearbeitung koordiniert und unterstützt werden kann.

Auf dem kommerziellen Softwaresektor stehen hier derzeit die sogenannten Workflow Management Systeme (Vorgangsbearbeitungssysteme) hoch im Kurs, die in mittlerweile fast nicht mehr überschaubarer Anzahl auf den Markt drängen. Im Zuge der Neuorganisation betrieblicher Abläufe sollen sie zur *effizienten und optimierten Ausführung unternehmensweiter Geschäftprozesse* herangezogen werden (vgl. [Jab94, Jab95, Rei93]). Bei McCarthy und Bluestein findet sich folgende Definition von Workflow Management Systemen (vgl. [MB91, zitiert nach Jab95]):

Definition 4-3: Workflow Management System

"Workflow management software is a proactive computer system which manages the flow of work among participants, according to a defined procedure consisting of a number of tasks. It coordinates user and system participants, together with the appropriate data resources, which may be accessible directly by the system or off-line, to achieve defined objectives by set deadlines. The coordination involves

passing tasks from participant to participant in correct sequence, ensuring that all fulfil their required contributions, taking default actions when necessary."

Im Hinblick auf ihre Formalisierungsbemühungen treten heutige Workflow Management Systeme zweifelsohne in die Fußstapfen früherer Büroprozedursysteme. Workflows sind ausführbare Abbilder von Geschäftsvorgängen. Sie können ähnlich wie ihre Vorgänger, die Büroprozeduren, als implementierte Koordinationspläne verstanden werden, die Benutzer bei der fristgemäßen Erledigung ihrer Aufgaben koordinieren, steuern und überwachen. Trotz vielerorts erhobener Kritik an der Automatisierung arbeitsteiliger Büroarbeit (vgl. Abschnitt 4.2.2) wird die besondere Wirksamkeit von Workflow Management Systemen gerade in der strikten Befolgung der vordefinierten Ablaufbeschreibungen gesehen. Wirtschaftliche Erfolge verspricht man sich davon vor allem in den folgenden Bereichen:

Reduzierung der Durchlaufzeiten

Die Benutzung elektronischer Post soll die Versand- und Liegezeiten drastisch reduzieren. Durch geeignete Rollendefinitionen soll darüberhinaus eine günstige Lastverteilung auf mehrere Bearbeiter erreicht werden, so daß insgesamt eine Produktivitätssteigerung erzielt werden kann.

Erhöhung des Anteils automatisierter Schritte

Die Wirtschaflichkeit soll durch die Erstellung von Programmen für einzelne Arbeitsschritte einer komplexen Aufgabe weiter erhöht werden; die Verweilzeiten innerhalb eines Arbeitsschrittes verkürzen sich, da für automatisierte Arbeitsschritte kein Bearbeiter mehr gebraucht wird.

Absicherung von Verfahren durch zwangsgesteuerte Schritte

Dieser Bereich verspricht eine Qualitätssteigerung, die ebenfalls wirtschaftliche Aspekte haben kann (verstärkte Kundenbindung). Durch die Steuerung des Prozesses in Verbindung mit der Aufbauorganisation ist eine weitaus höhere Sicherheit bei der Durchführung organisatorischer Verfahren gegeben als bei papiergestützten Vorschriften und Regeln.

Erhöhte Kontrollfähigkeit über Menge und Status von Prozessen

Insbesondere bei Prozessen mit längerer Laufzeit und starker Arbeitsteilung entsteht ein Informationsbedarf über den Verbleib und den Status einzelner Prozesse. Auf Basis dieser Information besteht dann die Möglichkeit für Eingriffe bei Krankheit oder Urlaub von Bearbeitern.

Die Modellierung von Workflows orientiert sich an vier konstituiven Aspekten, mit denen das Regelwerk der Büroprozedursysteme weiter ausgearbeitet und modularisiert wird (vgl. Abschnitt 4.2.1.1).

Die *aktivitätenbezogenen Aspekte* eines Workflows beschreiben, *was* getan werden muß. Sie legen für die elementaren Workflows eines gegebenenfalls komplex verschachtelten Geschäftsvorganges Verweise auf Applikationen fest, mit denen die eigentliche Funktionalität der elementaren Bearbeitungsschritte erbracht werden soll. Die Vergabe eines Kundenkredites ließe sich in einem Workflow Management System als sogenannter Topworkflow "Kreditvergabe" modellieren, der aus mehreren, ausreichend fein zerglie-

derten Subworkflows (z.B. "Kreditantrag ausfüllen", "Genehmigen") besteht, die jeweils ein ausführbares Programm referenzieren.

Die *aktorenbezogenen Aspekte* beschreiben, *wer* einen elementaren Workflow ausführt. Als Aktoren eines elementaren Workflows können sowohl Menschen als auch Maschinen bzw. Programmsysteme benannt werden. Die geeignete Zuordnung von Aktoren zu Workflows erfolgt vor dem Hintergrund eines formalen Organisationsmodells, in dem Benutzer als Träger von Rollen mit organisationsbezogenen Kompetenzen ermittelt werden können.

Ressourcenbezogene Aspekte legen fest, *welche Informationen* (Dokumente) ein Aktor bei der Ausführung eines elementaren Workflows benötigt bzw. erzeugt.

Die *abhängigkeitsbezogenen Aspekte* schließlich bestimmen, *wann* (Sub-) Workflows ausgeführt werden. Prinzipiell stehen zwei Möglichkeiten zur Spezifikation der Ablaufkontrolle zur Verfügung. Eine präskriptive Ablaufkontrolle läßt alternative Ausführungsreihenfolgen nur über die Auswertung von vordefinierten Ablaufbedingungen zu; eine deskriptive Ablaufkontrolle unterstützt demgegenüber eine (festgelegte) Menge äquivalenter Ausführungsreihenfolgen. Die Abbildung von Kontrollflüssen erfolgt mit Hilfe formaler mathematischer Modelle, in der Regel durch Petri-Netze oder geeignete Regelsprachen (vgl. [Jab94]).

Abschließend halten wir fest, daß sowohl Büroprozedursysteme als auch Workflow Management Systeme Koordinationspläne implementieren und als ausführbare Abbilder arbeitsteiliger Büroprozesse einsetzen. Der folgende Abschnitt stellt einen alternativen Ansatz der CSCW-Forschung vor, der sich arbeitspsychologisch motivieren läßt und Pläne als Ressourcen menschlichen Arbeitshandelns interpretiert.

4.2.2 Koordinationspläne als handlungsorientierende Ressourcen

Durch Büroprozeduren bzw. Workflows festgelegte Ablaufbeschreibungen können in aller Regel zur Laufzeit des Anwendungssystems nicht mehr modifiziert werden. Die mangelnde Flexibilität rechtfertigt man damit, daß die zu modellierenden Büroprozesse stark-strukturiert und formalisierbar seien und ihre Ausführung auf einem festen Regelsatz beruhe. Daß es für die Koordination von arbeitsteiligen Prozessen keine derartig vollständige Formalisierung geben kann, wird jedoch mittlerweile von breiten Wissenschaftskreisen als erwiesen betrachtet (vgl. [GS86, Sør88, KS90, SB92, Rob93]) und hat sich im Zuge der praktischen Erprobung von Büroprozedursystemen aus der Sicht von Anwendern bestätigt (vgl. [KHK+91]).

In ihrem Erfahrungsbericht über den Einsatz des DOMINO-Prototyps bei der Gesellschaft für Mathematik und Datenverarbeitung konstatieren Kreifelts et al. kritisch (vgl. [KHK+91], S. 125ff.):

> "Especially officials felt themselves 'fenced in' by the system features they had to use. The most common complaint was the missing possibility for letting another person have a look at the purchase form and get it back with her or his comments in an informal way (ad-hoc comment feature)."

Ein weiterer Strang der Kritik entfaltet sich angesichts mangelnder Kontextinformationen (und damit wohlgemerkt auch mangelnder Kontrolle) bei der Bearbeitung von Teilschritten:

> "Even in the initial testing phase, uses did not very much playfully explore the system [...]. The main reason seems to be that the current system does not give too much cooperative context (no explicit representation of procedures, of people involved) so that users may be in doubt which is the next or previous station, who exactly is receiving their comment or complaint."

Die CSCW-Forschung hat die erhobene Kritik aufgegriffen und fokussiert bereits seit einigen Jahren auf die Entwicklung von Softwaresystemen, die weniger auf die Steuerung und Kontrolle, denn auf die *Unterstützung* routinisierter Formen der Zusammenarbeit bedacht sind. Dafür sprechen mehrere Gründe:

Büroprozesse bestehen aus einer Synthese von situationsabhängigem "Problemlösen" und routinemäßiger Erledigung von Aufgaben, die nicht getrennt voneinander betrachtet werden können. Problemlösungsstrategien entwickeln sich in der Regel aus bereits bewährten Arbeitsroutinen, die in manchen Fällen nur minimal anzupassen sind und in anderen Fällen als "Bausteine" zu einer *ad hoc* Lösung zusammengesetzt werden. Zur Formalisierbarkeit von Büroarbeit bemerken Gerson und Star (vgl. [GS86], S. 266):

> "Every real-world system is an open system: It is impossible, both in practice and in theory, to anticipate and provide for every contingency which might arise in carrying out a series of tasks. No formal description of a system (or plan for its work) can thus be complete. Moreover, there is no way of guaranteeing that some contingency arising in the world will not be inconsistent with a formal description or plan for the system. These two points apply to every system, whether it be a human social organization or an information system. *Every real-world system thus requires articulation* to deal with the unanticipated contingencies that arise. Articulation resolves these inconsistencies by packaging a compromise that "gets the job done", that is, that *closes the system locally and temporarily* so that work can go on."

Artikulationsarbeit (*articulation work*) umfaßt alle Tätigkeiten, die zur Koordination bei der Erledigung einer kooperativen Aufgabe notwendig sind. Sie erkennt an, daß Büroarbeit eine Folge dezentralisierter Verhandlungen ist und kein "richtiger" Weg der Aufgabenbearbeitung bindend vorgegeben werden kann. Büroarbeit ist nicht statisch und kann daher auch keinen starren Regeln folgen. Weder Büroprozeduren noch Workflows erfassen die Vielfalt und Reichhaltigkeit menschlicher Arbeit. Koordinationspläne sind deshalb nicht wertlos, ihr Stellenwert muß jedoch überdacht werden (vgl. [Suc87], S. 52):

> "[...] plans are resources for situated action, but do not in any strong sense determine its course. While plans presuppose the embodied practice and changing circumstances of situated action, the efficiency of plans as representations comes precisely from the fact, that they do not represent those practices and circumstances in all of their concrete detail."

Im Sinne von Suchman dürfen *Koordinationspläne* demnach nicht als algorithmisierbare, d.h. implementierbare Vorschriften und Regeln verstanden werden. Koordinationspläne sind vielmehr *Ressourcen* für menschliches Arbeitshandeln und stellen als solche ein analytisches Instrumentarium bereit, mit der eine Folge von Bearbeitungsschritten **vor** und **nach** einer situativen Handlung betrachtet werden kann. Koordinationspläne geben auf diese Weise punktuelle Anleitung und Orientierung bei der arbeitsteiligen Erledigung einer Aufgabe, ohne daß eine im formalen Sinne geschlossene Handlungsabfolge vorliegt (vgl. [SB92], S. 19):

> "... the problem with incorporating models of *organizational procedures* is not that organizational procedures and structures are fictious. Rather, they serve a heuristic function in action by identifying constraints, pitfalls and strategic positions in the field of work."

Dies setzt jedoch voraus, daß Koordinationspläne - als "geronnene" Erfahrung arbeitsteiliger Formen der Zusammenarbeit - in geeigneter Form vergegenständlicht und visualisiert werden. Vergegenständlichte Koordinationspläne verschaffen den Benutzern eines Systems die notwendigen Kontextinformationen, an denen sie sich informieren und einen *Überblick* über den Arbeitsprozeß verschaffen können.

Im Ergebnis führen diese Überlegungen zur Konstruktion von Softwaresystemen, die die Kontrolle der Koordination zu den koordinierten Personen selbst verlagern und Benutzern flexible Anpassungsmöglichkeiten von Koordinationsplänen bieten (vgl. [SB92], S. 20):

> "[...] organization models in CSCW applications should be conceived of as *resources* for competent and responsible workers. That is, the system should make the underlying model accessible to users and, indeed, support users in interpreting the procedure, evaluate its rationale and implications. It should support users in applying and adapting the model to the situation at hand. It should allow users to tamper with the way it is instantiated in the current situation, execute it or circumvent it, etc. The system should even support users in modifying the underlying model and creating new models in accordance with the changing organizational realities and needs."

Um Koordinationspläne *anpaßbar* zu machen, werden die während der Anpassung veränderlichen Aspekte vergegenständlicht ("reifiziert", vgl. [Obe94], S. 37).[9] Diese materiellen Repräsentationen lassen sich dann mit geeigneten Hilfsmitteln grafisch aufbereiten, so daß sie interaktiv, d.h. zur Laufzeit des Systems, abgewandelt werden können. Veränderliche Aspekte von Koordinationsplänen sind alle Größen, die in Büroprozedursystemen und Workflow Management Systemen noch durch statische Regeln festgeschrieben werden: die Benennung einzelner Bearbeitungsschritte und der für sie zuständigen Personen oder Rollenträger, die Festlegung von Abhängigkeiten zwischen Bearbeitungsschritten, die Zuweisung von benötigtem oder erzeugten Material, etc.

Erste Ansätze in Richtung einer situativen Anpassung von Koordinationsplänen finden sich bereits im ProMInanD-System, das zur (indirekten) Modifikation des definierten Bearbeitungsweges einer elektronischen Umlaufmappe einen Satz von Operationen bereitstellt (vgl. KRW90]); andere Systeme bieten mittlerweile direkte grafische Manipulationsmöglichkeiten an ihrer Benutzungsschnittstelle an (vgl. [HKW92, KHW93]).

4.3 Zur Verwendung von Koordinationsplänen innerhalb der Softwareumgebung

Thema von Abschnitt 4.2 waren die gegensätzlichen Verwendungsmöglichkeiten von Plänen bei der Koordination arbeitsteiliger Prozesse. Es wurde argumentiert, daß ein Verständnis von Koordinationsplänen als programmierbare und ausführbare Prozeßabbilder insofern problema-

---

[9] Diese spezielle Form der Anpaßbarkeit von Software wird von Oberquelle als *Anpaßbarkeit der zeitlichen Bedingungen* klassifiziert. Hiermit werden Anpassungen an Steuerungen (Koordinationspläne, Makros, Skripten) bezeichnet, die den Ablauf von Arbeitsprozessen beeinflussen (vgl. [Obe94], S. 37).

tisch ist und an seine Grenzen stoßen muß, als daß die betrachteten Prozesse sich einer geschlossenen Formalisierung entziehen und Benutzenden darüber hinaus der Kontrollverlust über eine Arbeitssituation droht. Alternative Ansätze legen nahe, den Stellenwert von Plänen zu relativieren und sie anstelle von Vorschriften als handlungsorientierende Ressourcen in Softwaresystemen einzusetzen.

Dieses Verständnis "paßt" in das hier angenommene Leitbild vom Arbeitsplatz für qualifizierte menschliche Tätigkeit. Es wird von Gryczan in [Gry95] in spezifischer Form aufgegriffen und im Kontext von WAM zu einem Konzept ausgearbeitet, dessen zentrale Idee es ist, Koordinationspläne zu vergegenständlichen und als Materialien innerhalb einer Softwareumgebung bereitzustellen, wo sie selbst zum Gegenstand der Arbeit werden können. Im Anlehnung an die Gryczan'sche Begriffsbildung werden diese Materialien im folgenden als *Prozeßmuster* bezeichnet. Sie sollen in diesem Abschnitt erläutert und ihr Bezug zu weiteren Materialien der kooperationsunterstützenden Softwareumgebung geklärt werden.

4.3.1 Materialüberblick

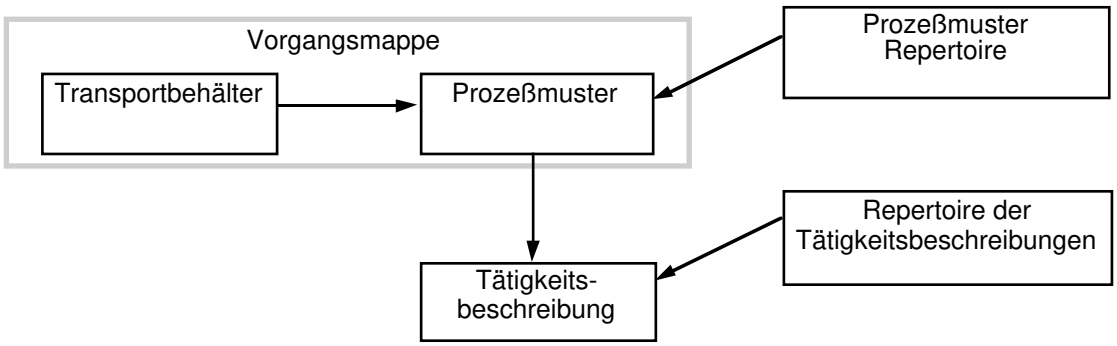


Abb. 4-2: Die "Koordinationsmaterialien" der Softwareumgebung.

Als vergegenständlichten Plänen kommt den Prozeßmustern eine zentrale Stellung bei der Koordination qualifizierter menschlicher Tätigkeiten zu; sie sind jedoch nicht die einzigen "Koordinationsmaterialien", die Benutzenden in ihrer Softwareumgebung bereitgestellt werden. Als Ausgangspunkt für die weitere Diskussion wird in diesem Abschnitt ein erster Überblick über die verschiedenen Materialien bzw. Behälter und ihre Zusammenhänge vermittelt (vgl. Abbildung 4-2).

Prozeßmuster

Prozeßmuster vergegenständlichen Formen der Zusammenarbeit, die sich bei der arbeitsteiligen Erledigung routinemäßig anfallender Aufgaben bewährt und etabliert haben. Zu diesem Zweck legen sie die im Rahmen der Aufgabenbearbeitung zu erledigenden Tätigkeiten und ihre anwendungsfachlich motivierten Abhängigkeiten fest, benennen verantwortliche Personen oder Rollenträger und geben die bei der Bearbeitung einer Tätigkeit benötigten bzw. erzeugten Materialien an. Tätigkeiten werden in Prozeßmustern in Form von Tätigkeitsbeschreibungen reifiziert. Die Erstellung und situative Anpassung der Prozeßmuster erfolgt durch die an der Kooperation beteiligten Personen.

Vorgangsmappe

In Abschnitt 3.3.4 ist das Konzept des adressierbaren Transportbehälters vorgestellt worden. Materialien, die arbeitsteilig und mit Hinblick auf eine spezifische, gemeinsam zu erledigende Aufgabe bearbeitet werden sollen, werden darin zusammengestellt und von dem Versandautomaten zwischen Softwareumgebungen weitergereicht. Das Konzept wollen wir nun erweitern, indem wir ermöglichen, *einen Transportbehälter mit einem Prozeßmuster zu behaften*; auf diese Gesamtheit werden wir uns im folgenden mit dem Begriff der *Vorgangsmappe* beziehen (vgl. Abbildung 4-2). Mit den Prozeßmustern werden die bei der Aufgabenbearbeitung anfallenden Tätigkeiten und ihre Einbindung in einen teil- oder vollstrukturierten Ablauf an dem Transportbehälter selbst vergegenständlicht. Vorgangsmappen greifen auf diese Weise den Zusammenhang zwischen einem als musterhaft erkannten Prozeß (dem Vorgang) und seinen Materialien auf.

Prozeßmusterrepertoire

Mit der Softwareumgebung sollen bekanntlich etablierte Formen der Zusammenarbeit unterstützt werden. Es kann angenommen werden, daß sich in vergleichbaren Kooperationssituationen, etwa bei der Kreditvergabe, auch vergleichbare koordinative Muster herausbilden. Sie sollen als *prototypische* Prozeßmuster vergegenständlicht und im Sinne von Vorlagen in einem allgemein zugreifbaren Materialbehälter, dem Prozeßmusterrepertoire, bereitgestellt werden. Benutzer greifen auf die Vorlagen des Repertoires zurück, um sich für die Koordination ihrer konkreten Arbeitssituation ein passendes Prozeßmuster zu erzeugen.

Tätigkeitsbeschreibung

Tätigkeiten ("Genehmigung erteilen"), die im Rahmen der kooperativen Bearbeitung von Aufgaben anfallen, werden durch Tätigkeitsbeschreibungen reifiziert. Sie sind die *Beschreibungsprimitiva* arbeitsteiliger Prozesse und werden in Prozeßmustern nach anwendungsfachlichen Gesichtspunkten miteinander kombiniert. Mit den Beschreibungen werden keine operationalisierbaren Vorgaben für die Ausgestaltung der benannten Tätigkeiten getroffen.

Repertoire der Tätigkeitsbeschreibungen

Tätigkeitsbeschreibungen sollen, vergleichbar den Prozeßmuster-Vorlagen, in einem erweiterbaren Repertoire als prototypische Bausteine zur Verfügung gestellt werden, auf die Benutzende bei Erstellung und Anpassung ihrer Prozeßmuster zurückgreifen.

Die folgenden Abschnitte widmen sich einer detaillierten Diskussion der hier zunächst nur skizzierten fachlichen Konzepte; Möglichkeiten ihrer technischen Umsetzung werden mit ausgewählten Schwerpunkten in Abschnitt 4.4 dargelegt.

4.3.2 Vorgangsmappen

Etablierte Zusammenarbeit hat viele Facetten. Sie setzt zum einen voraus, daß den Beteiligten einer kooperierenden Gruppe der *Prozeß* in Form eines immer wiederkehrenden Musters bekannt ist. Zum anderen müssen aber auch die bei der Durchführung des Prozesses benötigten *Materialien* verfügbar sein. Was das im Einzelnen bedeutet, kann an dem eingangs beschriebenen Szenario der Kreditvergabe veranschaulicht werden.

Unter der Annahme, daß das in einer Bank übliche Kreditvergabeverfahren als solches allen Beteiligten bekannt ist, wird das Funktionieren der Zusammenarbeit zwischen dem zuständigen Kundenberater, dem von ihm zur Genehmigung des Kredits herangezogenen zweiten Berater und den Sachbearbeitern dadurch sichergestellt, daß

- der zuständige Kundenberater seine Ansprechpartner bzw. die jeweils kompetenten Rollen kennt,
- er die einzelnen Tätigkeiten, die diese erledigen sollen ("Sicherheiten einholen", "Genehmigung erteilen") und die bei der Erledigung der Tätigkeit benötigten Materialien ("Kreditantrag für den Kunden X") benennen kann,
- die Materialien vorhanden sind und an die verantwortlichen Arbeitsplätze durch Weiterreichen überstellt werden können,
- Übereinstimmung bezüglich der fachlichen Abhängigkeiten zwischen einzelnen Tätigkeiten herrscht.

Den Zusammenhang zwischen einem als musterhaft erkannten Prozeß und seinen Materialien greifen wir mit dem Konzept der *Vorgangsmappe* auf, worunter wir im zurückliegenden Abschnitt begrifflich einen Transportbehälter mit einem anhaftenden Prozeßmuster gefaßt haben (vgl. Abbildung 4-2). Die Reifikation von Tätigkeiten und ihren potentiellen Bearbeitungsfolgen durch ein Prozeßmuster hilft den Beteiligten, ein etabliertes Koordinationsmuster wiederzuerkennen und zeigt damit "auf einen Blick", *in welchem Bearbeitungszusammenhang* die in dem Transportbehälter liegenden Materialien an einen bestimmten Arbeitsplatz überstellt werden. Grundvoraussetzung bleibt selbstverständlich, daß die von der Organisation und durch gesetzliche Randbedingungen bestimmten Verfahren allen Beteiligten bekannt sind, da andernfalls der übergeordnete Zweck, das "Umzu", des koordinierten Materialaustausches nicht deutlich werden kann.

Eine wichtige Eigenschaft der Vorgangsmappen besteht darin, daß sie den *Zustand* des Arbeitsprozesses reflektieren, in dessen Kontext sie verwendet werden. Der Zustand ist durch den gegenwärtigen Ort der Vorgangsmappe (eine Softwareumgebung), ihren bisher durchlaufenen Pfad (die Historie), bereits erledigte Tätigkeiten sowie weitere Bearbeitungsmöglichkeiten bestimmt. Ein Benutzer beeinflußt den Prozeßfortschritt, indem er eine Tätigkeit erledigt (sie wird an dem zugehörigen Prozeßmuster als durchgeführt gekennzeichnet), den geplanten

Bearbeitungsweg modifiziert oder die Vorgangsmappe über einen seiner Ausgangskörbe an andere Mitarbeiter weiterreicht.

Die Modellierung von Vorgangsmappen orientiert sich an den in Büroorganisationen traditionell verwendeten Umlaufmappen. Sie werden im engeren Sinne verwendet, um den am Umlauf beteiligten Mitarbeitern Arbeitsunterlagen zur Kenntnisnahme und Einsicht vorzulegen. In dieser Hinsicht kann der Umlauf als *eine* etablierte Form der (gleichwohl degenerierten) "Zusammenarbeit" zwischen Personen interpretiert und, bezogen auf die Software-Arbeitsumgebung, als Muster vergegenständlicht werden. Im Hinblick auf eine Metaphernbildung ist für uns jedoch eher das allgemeinere Konzept der Umlaufmappe von Bedeutung. Umlaufmappen dienen der Zusammenstellung von Dokumenten, die verschiedene Personen arbeitsteilig und mit Hinblick auf eine spezifische, gemeinsam zu erledigende Aufgabe bearbeiten sollen. Zur Festlegung ihres Transportweges werden Umlaufmappen mit einem *Laufzettel* behaftet; die mit einer Umlaufmappe korrelierten Tätigkeiten und deren Einbindung in eine bestimmte Ablaufstruktur sind daraus jedoch nicht ersichtlich. Sie ergeben sich für den Einzelnen in der Regel nur aus dem Inhalt der beigefügten Dokumente und anderen, zumeist informellen Beschreibungen (Betreff-Felder, angeheftete Notizen). Mit der Modellierung von Vorgangsmappen als Transportbehälter mit anhaftendem Prozeßmuster werden die koordinativen Informationen einer expliziten und formatierten Vergegenständlichung zugeführt, womit ergänzende Informationsquellen nicht ausgeschlossen werden sollen.

4.3.3 Prozeßmuster

Durch die geeignete Zusammenstellung von Werkzeugen und Materialien läßt sich eine Softwareumgebung konstruieren, die von Sequenzen der Bearbeitung absieht und sich stattdessen auf die Gegenstände konzentriert, mit denen eine Aufgabe situationsgemäß erledigt werden kann. Diese Form der Modellierung erkennt als wesentliches Merkmal qualifizierter Arbeit an, daß der Handelnde selbst Kontrolle über seine Handlungsfolgen besitzt. Demgegenüber zeigt das Kreditvergabe-Szenario, daß arbeitsteilige Formen der Zusammenarbeit durchaus Anteile enthalten, die von vorgegebenen Reihenfolgen der Bearbeitung ausgehen. Diesen scheinbaren Konflikt gilt es unter Beibehaltung der Grundannahmen zu lösen.

Zu diesem Zweck wird von G. Gryczan das Konzept des *Prozeßmusters* eingeführt (vgl. [Gry95], S. 176ff.). Prozeßmuster sind die Vergegenständlichung (Reifikation) von Arbeitsweisen, die sich im Rahmen arbeitsteiliger Kooperationen bewährt und etabliert haben. Es wird davon ausgegangen, daß routinisiertes, kooperatives Arbeitshandeln generell Muster in der Zusammenarbeit herausbildet und daß Menschen ihr zukünftiges Handeln daran orientieren und ausrichten.

Im folgenden sollen die Prozeßmuster einer detaillierteren Betrachtung unterzogen und verschiedene Eigenschaften am Beispiel des Kreditvergabe-Szenarios veranschaulicht werden. Die Eigenschaften werden nach einzelnen Punkten gegliedert.

Prozeßmuster tragen einen Namen.

Prozeßmuster sind ein Hilfsmittel für die Koordination kooperativer Aufgaben ("Kreditvergabe"). Sie werden stets von Benutzenden erstellt, die für ihre Benennung verantwortlich sind. Der Name eines Prozeßmusters ("Kreditvergabe für den Kunden X") sollte mit

Bezug auf den zu koordinierenden Prozeß gewählt werden, so daß die beteiligten Mitarbeiter an dem Namen "auf einen Blick" erkennen können, um *welche* gemeinsame Aufgabe es geht. Aus der Sicht von Benutzenden identifiziert der Name eines Prozeßmusters den Transportbehälter (vgl. Abschnitt 3.3.4), dem es anhaftet.

Prozeßmuster benennen eine oder mehrere zu erledigende Tätigkeiten.

Prozeßmuster benennen die verschiedenen Tätigkeiten (z.B. "Genehmigung erteilen"), die bei der arbeitsteiligen Erledigung einer gemeinsamen Aufgabe regelmäßig anfallen und von den beteiligten Personen koordiniert werden müssen.

Im Sinne von Floyd tragen *Tätigkeiten* zur Erfüllung von vorgegebenen Aufgaben (z.B. "Kreditvergabe") bei; sie erfolgen im Rahmen von Arbeitsabläufen, die sich aus einer strukturierten Gesamtheit einzelner Tätigkeiten zusammensetzen (vgl. [Flo91], S.8). Hauptbestandteil der menschlichen Arbeitstätigkeit sind die sie realisierenden Handlungen. Mit Oberquelle können wir Tätigkeiten in diesem Sinne als "zusammenhängende Gesamtheiten von sequentiellen Handlungen einer Rolle mit gemeinsamem Zweck und bestimmter Methode" verstehen (vgl. [Obe87], S.16).

Handlungen erfolgen demnach im Rahmen von sequentiellen Handlungsfolgen. Qualifizierte Tätigkeit zeichnet sich dadurch aus, daß weder einzelne Handlungen noch Handlungsfolgen fest vorgegeben werden können, da Menschen mit dem gebotenen Gestaltungsspielraum selbständig und situationsabhängig über alternativ mögliche Handlungsfolgen entscheiden. Mit der Benennung von Tätigkeiten in einem Prozeßmuster werden deshalb auch bewußt keine operationalisierbaren Vorgaben für ihre Ausgestaltung durch den Benutzer getroffen. Der Name einer Tätigkeit ("Genehmigung erteilen") hat ein Stück weit sprachbildenden Charakter im Hinblick auf die zu erledigende Tätigkeit, d.h. Einzelne können an ihm erkennen, *was* zu tun ist. Über das *Wie* entscheiden sie jedoch selbst und in Abhängigkeit von der konkreten Arbeitssituation.

Aus diesen Überlegungen können wir zwei Dinge schließen: zum einen sind es immer Menschen, die über das "Erledigt-Sein" der in einem Prozeßmuster benannten Tätigkeiten entscheiden; zum anderen werden Tätigkeiten nicht durch ein Prozeßmuster implementiert. In dieser Hinsicht unterscheiden letzter sich etwa grundlegend von Workflows, die einzelne Arbeitsschritte (in der Workflow-Terminologie sogenannte *elementare Workflows*, vgl. Abschnitt 4.2.1.2) mit Bezug auf konkrete Applikationen implementieren.

Die in einem Prozeßmuster benannten Tätigkeiten werden in Form von sogenannten *Tätigkeitsbeschreibungen* vergegenständlicht; sie bilden die Beschreibungsprimitiva kooperativer Prozesse und werden als prototypische Bausteine in einem Repertoire bereitgestellt, auf das Benutzer bei Erstellung und Anpassung von Prozeßmustern zurückgreifen können.

Die Beschreibung einer Tätigkeit "endet" bei der Softwareumgebung eines Benutzers, in dessen individuelle Arbeitsprozesse nicht steuernd eingegriffen werden kann (vgl. Abschnitt 2.3.4). Wir können davon ausgehen, daß in der Umgebung eines kreditgenehmigenden Kundenberaters die notwendigen Werkzeuge zur Bearbeitung von Kreditanträgen bereitliegen, treffen jedoch keinerlei Annahmen darüber, in welcher Reihenfolge er diese Werkzeuge benutzt. Die Umgebung als der Ort, an dem qualifizierte Sachbearbeitung stattfindet, stellt somit eine klare Trennlinie dar, die es uns erlaubt, zwischen der arbeitsplatzübergreifenden Koordination von

(kooperativen) Tätigkeiten und ihrer jeweils individuellen Ausgestaltung durch den Menschen zu unterscheiden.

Die von einem Prozeßmuster benannten Tätigkeiten werden nach anwendungsfachlichen Gesichtspunkten miteinander kombiniert.

Für die routinisierte Zusammenarbeit ist nicht nur wichtig, daß die Beteiligten ein gemeinsames Verständnis über Sinn und Zweck einzelner Tätigkeiten tragen; sie müssen darüber hinaus auch ihre Kombination zu komplexeren Mustern der Zusammenarbeit verstehen und erkennen. Die Einbettung in einen Gesamtprozeß ("Kreditvergabe für den Kunden X") liefert oft erst den notwendigen Kontext, in dem einzelne Tätigkeiten ("Genehmigung erteilen") interpretiert und qualifiziert erledigt werden können.

Bei der Kreditvergabe existieren beispielsweise die folgenden Abhängigkeiten: zunächst müssen die Sicherheiten des Kunden eingeholt werden, dann erst kann die Kreditgenehmigung erteilt werden, die selbst wiederum Voraussetzung für den erfolgreichen Abschluß der Restkreditversicherung bildet. Das Auszahlungskonto kann demgegenüber unabhängig von allen anderen Tätigkeiten angelegt werden.

Prozeßmuster benennen nicht nur einzelne Tätigkeiten, sondern beschreiben darüberhinaus ihre anwendungsfachlichen Abhängigkeiten. Hierin zeigt sich insbesondere ihre Eigenschaft als vergegenständlichte Koordinationspläne (vgl. Abschnitt 4.2.2). Die Vergegenständlichung eines Koordinationsplans durch ein Prozeßmuster ermöglicht seine situative Anpassung im konkreten Arbeitskontext, d.h. Pläne können selbst zum Gegenstand von Arbeit gemacht werden. Anpaßbarkeit ist eine wichtige Eigenschaft, die den Benutzenden *Kontrolle* über die Koordination ihrer Zusammenarbeit garantiert. Auf die Kontrolldimension von Prozeßmustern wird in Abschnitt 4.3.5 noch ausführlicher zurückzukommen sein.

Für jede von einem Prozeßmuster benannte Tätigkeit wird eine zuständige Person beziehungsweise ein (anonymer) Rollenträger festgelegt. Darüberhinaus werden die bei ihrer Realisierung benötigten und/oder erzeugten Materialien beschrieben.

Für die Koordination der Zusammenarbeit ist es nicht allein ausreichend, einzelne Tätigkeiten und ihre Abhängigkeiten festzulegen. Die Erledigung von Aufgaben findet in einem organisatorischen Kontext statt, in dem anfallende Tätigkeiten verantwortlich an bestimmte Personen ("der Kundenberater Y") oder Rollen ("ein Kundenberater") gebunden werden. Darüberhinaus werden bei der Bearbeitung einzelner Tätigkeiten bestimmte Materialien ("Kreditantrag") benötigt bzw. erzeugt. Diese koordinativen Informationen müssen bei der Modellierung von Prozeßmustern berücksichtigt werden; sie werden zum Bestandteil der jeweiligen Tätigkeitsbeschreibung. Letztere können wir jetzt präziser verstehen als Software-Materialien, die eine Tätigkeit benennen, ihre Zuständigkeit festlegen und bei der Erledigung der Tätigkeit benötigte bzw. erzeugte Materialien beschreiben.

Prozeßmuster reflektieren den Fortschritt der Aufgabenbearbeitung.

Es wurde bereits festgehalten, daß nur der Benutzer über den erfolgreichen Abschluß, das "Erledigt-Sein", einzelner Tätigkeit entscheiden kann. Prozeßmuster sollen solche Bearbeitungsfortschritte widerspiegeln; zu diesem Zweck markieren Benutzer die von ihnen abschließend durchgeführten Tätigkeiten an einem Prozeßmuster als erledigt.

Abschließend zu diesem Abschnitt sollen Prozeßmuster in Bezug zu einer jüngeren Arbeit aus dem CSCW-Bereich gesetzt werden. Prozeßmuster besitzen in vielerlei Hinsicht die Charakteristika der von Robinson thematisierten *common artifacts*, mit denen Sørgaards Begriff vom *shared material* aufgegriffen und fortgedacht wird (vgl. [Rob93]). Unter einem "common artifact" versteht Robinson alltägliche Artefakte, die alle wesentlichen Dimensionen kooperativer Arbeit abdecken. Für den Einzelnen müssen sie in Struktur und Umgang durchschaubar sein, in der Gruppe die implizite Kommunikation und Koordination durch gemeinsame Nutzung und Bearbeitung fördern. Sie sollen den Fokus für Diskussions- und Aushandlungsprozesse stellen und Überblick über den kooperativen Prozeß liefern. Als anschauliches Beispiel für ein "common artifact" führt Robinson das Schlüsselbord einer Hotelrezeption an (vgl. [Rob93], S. 191):

> "The keyrack behind the reception desk in a hotel is an example. Guests can leave and collect their keys; can see which other guests are in or out, and leave messages in the pigeonholes. Hotel staff use it to communicate with their colleagues, and place bills, faxes etc. to be given out to guests. The presence of keys, or the contents of pigeonholes, conveys information, and may be the subject of question or discussion. Some operations are considered legitimate, while others are not: usually only the receptionist can place keys or messages; keys have to be hung over appropriate numbers; etc. The keyrack is a model of the hotel, mapping the rooms. A glance at it in the late evening gives overview of the hotel occupancy.
>
> Yet this keyrack is not foolproof, nor is it "active". There is nothing to prevent keys being hung in the wrong place, or lost. It can be used in many idiosyncratic ways - probably violating the recognised procedures. Conversely, there are certain things about a keyrack that a fixed, like the positions of the hooks. It is simply not possible to hang a key between two hooks, as there is nothing to hang it on. So potential uses are a result of physical properties, local conventions and rules, and situated activities."

Wie Sørgaard interpretiert auch Robinson seine "common artifacts" nicht mit Bezug auf ein Gesamtmodell der Anwendung; er kann daher auch keinerlei plastische Orientierungshilfe geben, *in welcher Form* sie in einem Anwendungssystem bereitgestellt werden könnten. Robinson beschränkt sich vielmehr auf die Feststellung, daß die für "common artifacts" charakteristischen Dimensionen kooperativer Arbeit in CSCW-Anwendungen reflektiert werden sollten (vgl. [Rob93], S. 199):

> "Common artifact is a label for a set of work dimensions that need to be reflected in CSCW applications."

Die Idee der "common artifacts" wird mit dem Konzept des Prozeßmusters in spezifischer Form aufgegriffen und zu Software-Materialien einer Arbeitsumgebung konkretisiert.

4.3.4 Das Prozeßmusterrepertoire

Im zurückliegenden Abschnitt wurde argumentiert, daß die Benutzer Prozeßmuster erstellen und verwenden, um ihre routinisierte Zusammenarbeit rechnergestützt zu koordinieren. In diesem Zusammenhang wurden Tätigkeitsbeschreibungen als die Bausteinprimitiva von Prozeßmustern eingeführt. Dabei haben wir angenommen, daß die verschiedenen, prototypischen Tätigkeitsbeschreibungen den Benutzern in einem gemeinsam genutzten und erweiterbaren

Repertoire bereitgestellt werden, auf das sie bei Erstellung und Anpassung ihrer Prozeßmuster zurückgreifen.

Routinisierte Zusammenarbeit impliziert, daß *wiederholt* ähnliche Arbeitssituationen in einer Organisation auftreten und sich dabei auch vergleichbare koordinative Muster herausbilden (etwa bei der Vergabe von Kundenkrediten). Benutzenden sollte daher die Möglichkeit geboten werden, häufig verwendete Prozeßmuster im Sinne von *Prototypen* bzw. *Vorlagen* an einem Ort zu sammeln, an dem sie den verschiedenen Softwareumgebungen zugänglich sind. Auf diese Weise bleibt es Einzelnen erspart, für vergleichbare Kooperationssituationen wieder und wieder quasi "identische" Prozeßmuster aus den (elementaren) Tätigkeitsbeschreibungen aufzubauen. Mit Gryczan fassen wir die Gesamtheit dieser prototypischen Prozeßmuster unter den Begriff des *Prozeßmusterrepertoires* (vgl. [Gry95], S. 191):

> "Für die unterschiedlichen kooperativen Arbeitsprozesse wird eine Menge von Prozeßmustern zur Verfügung gestellt, die jeweils in einer bestimmten Anwendungssituation verwendet werden können. Dafür existiert ein Prozeßmusterrepertoire in jeder Arbeitsumgebung. Ein Prozeßmusterrepertoire ist die Menge der Prozeßmuster, die für kooperative Tätigkeiten zur Verfügung stehen. Es wird durch einen Behälter für Prozeßmuster vergegenständlicht."

Mit den Vorlagen des Repertoires vergegenständlichen wir demnach die bewährten Muster zur Koordination kooperativer Tätigkeiten (z.B. bei der Kreditvergabe in einer Bank) und abstrahieren dabei von der einzelnen Kooperationssituation (z.B. die Kreditvergabe für den Kunden X). Umgekehrt können wir solche bewährten Muster aber auch verwenden, um daran unser kooperatives Arbeitshandeln in der Situation zu orientieren bzw. auszurichten.

Einzelne Vorlagen gewinnen wir durch Abstraktion; sie dienen uns später zum *Erzeugen und Erkennen* einer prinzipiell beliebigen Anzahl gleichartiger Prozeßmuster, die wir in konkreten Kooperationssituationen verwenden, um unsere Zusammenarbeit mit anderen zu koordinieren. Dabei werden alle Eigenschaften einer Vorlage auf ihre Exemplare übertragen. Veranschaulichen wir uns die Verwendung der Prozeßmuster-Vorlagen am Beispiel der Kreditvergabe:

Ein Kunde X wendet sich mit einem dringlichen Kreditwunsch an seinen zuständigen Kundenberater. Die Kreditvergabe erfordert die routinemäßige Zusammenarbeit mit weiteren Mitarbeitern der Bank. Der angesprochene Kundenberater kann daher auf die bereits definierte *Kreditvergabe*-Vorlage des Prozeßmusterrepertoires zurückgreifen und auf Basis dieser Vorlage das Prozeßmuster *Kreditvergabe für den Kunden X* erzeugen. Dabei werden alle Eigenschaften der Vorlage auf das Prozeßmuster übertragen. Der Kundenberater greift sich nun eine leere rote Mappe, die er mit dem Prozeßmuster *Kreditvergabe für den Kunden X* behaftet[10]. Nach dem Abschluß des Kundengespräches füllt der Berater das Kreditantragsformular für den Kunden X aus und fügt es der Mappe hinzu, die er anschließend zum Weitertransport in seinen Postausgangskorb legt.

---

[10] In einer Softwareumgebung ließen sich verschiedene Sorten von Mappen in einem *Mappen-Pool* zusammenstellen, der für den Benutzer - ähnlich einem Büromaterial-Schrank - einen gewissermaßen unerschöpflichen Vorrat an leeren Mappen bereithält. Bei der Entwicklung des Prototypen wurde der Einfachheit halber auf die Implementierung eines solchen Mappen-Pools verzichtet. Stattdessen wird zusammen mit einem Prozeßmuster immer auch ein leerer (und einheitlicher) Transportbehälter erzeugt (vgl. Abschnitt 3.3.4), dem das Muster automatisch anhaftet. Diese Form der Modellierung verhindert wohlgemerkt, daß Informationen über den Arbeitsprozeß, z.B die Bearbeitungsdringlichkeit der gemeinsamen Aufgabe, anhand der speziellen Eigenschaften des gewählten Transportbehälters, z.B. seine Farbe, vermittelt werden.

Eine besondere Vorlage des Prozeßmusterrepertoires ist die "leere" Vorlage, auf die Benutzer zugreifen, wenn *keine* der anderen Vorlagen ihre spezielle Arbeitssituation adäquat beschreibt. In diesem Fall muß der durch das Prozeßmuster reifizierte Koordinationsplan von Grund auf neu erstellt werden. Benutzenden steht zu diesem Zweck das Repertoire von Tätigkeitsbeschreibungen zur Verfügung, aus dem sie einzelne Bausteine auswählen und zu einem passenden Muster kombinieren. Das Erstellen neuer Muster auf der Basis der leeren Vorlage wird typischerweise der Fall sein, wenn das Prozeßmusterrepertoire selbst noch im Aufbau befindlich ist. Steht den Benutzenden jedoch erst einmal ein solider Grundstock an prototypischen Mustern zur Verfügung, kann davon ausgegangen werden, daß sich in der Regel passende Vorlagen finden lassen, die die eingetretene Kooperationssituation zutreffend oder zumindest annähernd erfassen.

Prozeßmuster können jederzeit situativ den Erfordernissen eines Arbeitskontextes angepaßt werden. Der durch ein Prozeßmuster vergegenständlichte Koordinationsplan wird demnach nicht *statisch* durch seine Vorlage geprägt. Die Ausrichtung des Handelns an erkannten Mustern und ihre inkrementelle Anpassung im Kontext konkreter Arbeitssituationen kann vielmehr zum Erkennen "verbesserter" Formen der Zusammenarbeit und - hier schließt sich der Kreis - zur Herausbildung neuer oder veränderter Vorlagen führen. Benutzende greifen also nicht nur auf die Vorlagen des Prozeßmusterrepertoires zu, um sich für die Koordination einer Arbeitssituation ein passendes Prozeßmuster zu erzeugen, sondern überführen umgekehrt auch neue bzw. adaptierte Prozeßmuster in das Repertoire, wenn sie sich bei der Koordination kooperativer Tätigkeiten als pragmatisch erwiesen haben und ihre Wiederverwendung in weiteren Kooperationssituationen plausibel erscheint. Das Einfügen neuer Vorlagen ist, wie gesagt, insbesondere für die Aufbauphase eines Prozeßmusterrepertoires charakteristisch.[11]

Abschließend wollen wir festhalten:

> Benutzer verwenden Prozeßmuster, um ihre Zusammenarbeit rechnergestützt zu koordinieren.
>
> Ein Prozeßmuster wird auf der Basis einer mehr oder weniger stark spezialisierten Vorlage des Prozeßmusterrepertoires erzeugt.

4.3.5 Die Kontrolldimension von Vorgangsmappen

In diesem Abschnitt wird eine wesentliche Dimension qualifizierter Zusammenarbeit thematisiert, die bisher nur am Rande erwähnt wurde, hier aber noch einmal einer ausführlicheren Darstellung zugeführt werden soll. Dabei handelt es sich um die Frage der *Kontrolle* in arbeitsteiligen Prozessen. Es sollen drei Aspekte von *Kontrolle* aufgezeigt werden, die sich aus der

---

[11] Abbott und Sarin diskutieren im Zusammenhang mit der von ihnen antizipierten "Fourth Generation of Workflow Technology" einen konzeptionell verwandten Ansatz (vgl. [AS94], S. 119):

> "Users should be able to easily browse through a 'library' of available process (and process component) definitions to find the one appropriate to what they are doing. [...] It should be possible to build the process definition 'library' or hierarchy based on experience in executing processes. On completion of a process (or before that, if necessary) it should be possible to review how it was performed and to abstract information from it in order to 'save' a process definition, or components thereof, in the library."

Sicht von Benutzern maßgeblich aus dem Umgang mit Vorgangsmappen respektive ihren Prozeßmustern ergeben:

Kontrolle im Sinne von *aktiver Koordination der Zusammenarbeit*,

Kontrolle im Sinne *informierten kooperativen Handelns*,

Kontrolle im Sinne von *Transparenz im Umgang mit gemeinsam genutzten Arbeitsgegenständen*.

4.3.5.1 Aktive Koordination der Zusammenarbeit

Prozeßmuster ermöglichen den Benutzern, ihre Tätigkeiten bei der Bearbeitung einer gemeinsamen Aufgabe rechnergestützt zu koordinieren. Entscheidend ist dabei, daß die Koordination der Zusammenarbeit *aktiv* durch Benutzende gestaltet wird. Die Kontrolle über die Koordination liegt demnach bei den koordinierten Personen selbst.

Die entscheidenden Voraussetzungen hierfür ergeben sich direkt aus dem Umgang mit Prozeßmustern, die *von Benutzenden erstellt und angepaßt* werden. Dies ist möglich, weil die Regeln, nach denen die Koordination funktioniert, durch das Prozeßmuster vergegenständlicht (und nicht etwa implementiert) werden. Die Regeln werden auf diese Weise für alle Prozeßbeteiligten transparent und vor allem beherrschbar, d.h. Benutzende können die Regeln aktiv gestalten, indem sie ein Prozeßmuster zum Gegenstand ihrer Arbeit machen.

4.3.5.2 Informiertes kooperatives Handeln

Kontrolle bedeutet nicht nur, daß Benutzende den kooperativen Arbeitsprozeß steuern, indem sie Bearbeitungswege planen und entsprechend dynamischer Anforderungen ändern. Kontrolle beginnt aus der Sicht eines Benutzers bereits damit, daß ein Prozeßmuster den *Kontext* vergegenständlicht, in dem er Tätigkeiten interpretieren und qualifiziert erledigen kann. Zu diesem Kontext zählt nicht nur die (statische) Einbettung einzelner Tätigkeit in eine fachlich vernetzte Tätigkeitenstruktur, sondern beispielsweise auch der Bearbeitungsfortschritt des Prozesses, der durch das Prozeßmuster reflektiert wird (vgl. Abschnitt 4.3.3). Prozeßmuster vermitteln den beteiligten Personen Überblick über den Prozeß als Ganzes und bilden in dieser Hinsicht einen Ausgangspunkt für informiertes Arbeitshandeln.

4.3.5.3 Transparenz im Umgang mit gemeinsamen Arbeitsgegenständen

Kontrolle wird bereits von Gordon und Blair als eine wesentliche Dimension der Zusammenarbeit anerkannt (vgl. [GB91]) und auf interessante Weise in Bezug zu technischen Kontrollansätzen im Bereich verteilter Systeme gestellt. Ausgangspunkt ihrer Argumentation ist, daß Softwaresysteme, die auf die Unterstützung und Koordination kooperativer Arbeit abstellen, per se verteilt sind und daher in fundamentaler Weise auf Technologien angewiesen sind, die durch verteilte Systemplattformen angeboten werden (z.B. Email, client/server-basierte Datenbanken und Betriebssysteme).

Transparency	**Central issue**	**Result of Transparency**

Location	The location of an object in a distributed environment	User unaware of the location of services
Access	The method of access to objects in a distributed system	All objects are accessed in the same way
Migration	The re-location of an object in a distributed environment	Objects may move without the user being aware
Concurrency	Shared access to objects in a distributed environment	Users do not have to deal with problems of concurrent access
Replication	Maintaining copies of an object in a distributed environment	System deals with the consistency of copies of data
Failure	Partial failure in a distributed environment	Problems of failure are masked from the user

Tabelle 4-2: Das technikorientierte Transparenzverständnis in verteilten Systemen (Quelle: [GB91], S. 59)

Traditionell wird in verteilten Systemen *Kontrolle* mit *Transparenz* gleichgesetzt, wobei Transparenz hier verstanden wird als das Ziel, charakteristische Merkmale des verteilten Rechnens sowohl vor Applikationen als auch Benutzenden zu verstecken (vgl. [GB91], S. 59):

> "Distribution transparency is now seen as a collective name for the masking out of various features of a distributed computation."

In Tabelle 4-2 werden verschiedene Formen der Transparenz in verteilten Systemen, ihr jeweils zentrales Anliegen und die Konsequenzen aus der Sicht von Benutzenden zusammengestellt.

Interessant ist nun, daß dieses technikorientierte Verständnis von Transparenz ganz offensichtlich den Bedürfnissen einer kooperationsunterstützenden Softwareumgebung zuwiderläuft. Die Existenz anderer Benutzer soll dort gerade *nicht* vor dem Einzelnen verborgen werden; sie gilt vielmehr als notwendige Voraussetzung für die Zusammenarbeit, im Rahmen derer Menschen mit gemeinsamen Arbeitsgegenständen (Beschwerdeschreiben, Sachstandsberichte, Kreditantragsformulare) umgehen, die sie koordiniert untereinander austauschen.

An dieser Stelle tritt in vollem Umfang die *Doppeldeutigkeit des Transparenz-Begriffs* zutage. Dem technikorientierten Ansatz läßt sich nun eine Sichtweise gegenüberstellen, derzufolge Transparenz genau gegensätzlich (und wohlgemerkt im wörtlichen Sinne) verstanden wird als das Ziel, den Umgang mit gemeinsamen Software-Materialien aus Sicht der Benutzenden *durchschaubar* und *kontrollierbar* zu gestalten. Die Auswirkung dieser Sichtweise auf die verschiedenen Transparenzformen werden in Tabelle 4-3 dargestellt.

Transparenz	**Resultat aus der Sicht von Benutzenden**

Location	Der Ort eines gemeinsamen Materials ist zu jedem Zeitpunkt eindeutig durch die Softwareumgebung bestimmt, in die es zuletzt - eingebettet in eine Vorgangsmappe - überstellt wurde. Benutzer können sich über den Ort eines Materials informieren.
Access	Der Ort eines gemeinsamen Materials regelt auch die erlaubten Zugriffe: Benutzende können nur dann verändernd auf ein Material zugreifen, wenn es sich in ihrer eigenen Softwareumgebung befindet (sie verfügen über das Material-*Original* mit dem Recht zum Ändern).
Migration	Der Austausch gemeinsamer Materialien zwischen Softwareumgebungen wird explizit gemacht: Benutzende reichen Materialien über ihre Postausgangskörbe weiter und nehmen sie über ihre Posteingangskörbe in Empfang.
Concurrency	Das Problem des konkurrierenden Material-Zugriffs wird demnach auf einer nicht-technischen Ebene durch die kooperierenden Personen selbst gelöst, die sich die gemeinsamen Materialien untereinander zur *exklusiven* Bearbeitung bereitstellen.
Replication	In Anlehnung an das Bild von *Original und Kopien* gehen wir davon aus, daß jederzeit genau ein Materialoriginal (mit dem Recht zum Ändern), aber beliebig viele Materialkopien (nur zum Sondieren) existieren können (vgl. [WW94]). Das Original liegt stets dort, wo das Material "seinen Ort hat". Soll aus einer anderen Softwareumgebung heraus auf das Material zugegriffen werden (etwa zu Informationszwecken), wird eine Material-Kopie bereitgestellt. Änderungen an einer Kopie wirken sich nie auf das Original aus; auch Materialinkonsistenzen lassen sich demnach auf nicht-technischer Ebene ausschließen. Benutzern wird an der Systemschnittstelle deutlich gemacht, ob sie mit einem Material-Original oder einer Material-Kopie umgehen.
Failure	Scheitert der automatisierte Versand von Materialien zwischen zwei Umgebungen - etwa weil der vorgesehene Empfänger gerade keinen Posteingangskorb bei dem Versandautomaten registriert hat - so bleibt die Vorgangsmappe mit dem darin eingebetteten Material *für den Absender sichtbar* in seinem Postausgangskorb liegen. Dieser kann entweder abwarten, ob möglicherweise der nächste Zustellversuch glückt, oder aber einen neuen Empfänger bestimmen.

Tabelle 4-3: Das kooperationsorientierte Transparenzverständnis vom Umgang mit gemeinsamen Materialien.

4.4 Die softwaretechnische Modellierung von Prozeßmustern

Im zurückliegenden Abschnitt wurden die fachlichen Konzepte *Prozeßmuster* (vgl. Abschnitt 4.3.3) und *Prozeßmusterrepertoire* (vgl. Abschnitt 4.3.4) dargestellt. Dieser Abschnitt behandelt ihre Umsetzung in einen softwaretechnischen Entwurf.

4.4.1 Die Materialklasse Prozeßmuster

Vor dem Hintergrund des von WAM bereitgestellten Entwurfsrepertoires lassen sich Prozeßmuster als Software-Materialien interpretieren. Im technischen Entwurf sollen sie durch Objekte der Materialklasse `Prozeßmuster` modelliert werden. Im Fokus dieses Abschnitts steht die Frage, wie die Klasse `Prozeßmuster` in Verhalten und Struktur zu gestalten ist, damit sich beliebige Muster zur Koordination kooperativer Tätigkeiten mit ihr modellieren lassen.

Bei der Beschreibung des *fachlichen Konzepts* Prozeßmuster sind bereits alle wesentlichen Eigenschaften solcher Muster herausgearbeitet worden. Sie gilt es in einen geeigneten Entwurf der Materialklasse umzusetzen. Es werden daher die folgenden Umgangsformen modelliert:

> `Prozeßmuster`-Objekte können benannt werden. Namen helfen, das vergegenständlichte Koordinationsmuster kurz und bündig zu charakterisieren und lassen Benutzende so "auf einen Blick" erkennen, worum es geht.
>
> In einem `Prozeßmuster`-Objekt lassen sich Tätigkeitsbeschreibungen dynamisch ergänzen, entfernen und nach anwendungsfachlichen Gesichtspunkten miteinander kombinieren.
>
> Die von einem `Prozeßmuster`-Objekt benannten Tätigkeiten können als erledigt markiert werden.
>
> `Prozeßmuster`-Objekte bieten die notwendige Funktionalität zum Sondieren des (vergegenständlichten) Prozeßzustands an: welche Tätigkeiten sind bereits erledigt? welche Bearbeitungsschritte stehen noch aus?

Neben dieser "Kernfunktionalität" besitzen Prozeßmuster eine Reihe weiterer nützlicher Eigenschaften, die an dieser Stelle kurz erwähnt werden sollen, im folgenden aber nicht weiter vertieft werden.

> `Prozeßmuster`-Objekte lassen erkennen, wann welcher Mitarbeiter den Prozeß angestoßen, d.h. das Prozeßmuster zur Koordination der anfallenden Tätigkeiten erzeugt hat.
>
> Sie geben Aufschluß darüber, welcher Mitarbeiter den Vorgang zuletzt in Bearbeitung hatte und wann der Transportbehälter, dem das Prozeßmuster anhaftet, in die aktuelle Umgebung überstellt wurde.
>
> Ein `Prozeßmuster`-Objekt kann mit einem Kommentar versehen werden, der grob die Anwendungssituation umreißt und bei Bedarf Hinweise, Anmerkungen oder Informationen zum aktuellen Prozeß bzw. Prozeßverlauf enthält.
>
> Benutzenden sollte die Möglichkeit geboten werden, Bearbeitungsprioritäten und Deadlines für die durch Prozeßmuster vergegenständlichten Kooperationsprozesse festzulegen.[12]

---

[12] Die entsprechenden Umgangsformen mit `Prozeßmuster`-Objekten sind derzeit noch nicht realisiert.

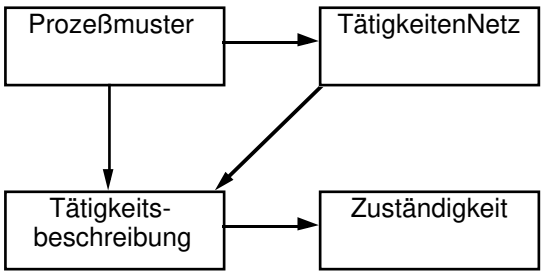


Abb. 4-3: `Prozeßmuster` realisieren einen gewissen Anteil ihrer Funktionalität durch das Zusammenspiel mit weiteren Entwurfskomponenten.

Zur Umsetzung der oben beschriebenen Kernfunktionalität arbeitet die `Prozeßmuster`-Klasse mit weiteren Klassen zusammen, an die sie Verantwortlichkeit delegiert. Abbildung 4-3 gibt einen Überblick über das Zusammenspiel der verschiedenen Komponenten.

Die Verantwortlichkeiten der neuen Komponenten lassen sich wie folgt skizzieren:

`Tätigkeitsbeschreibung`

vergegenständlicht qualifizierte menschliche Arbeitstätigkeiten ("Sicherheiten einholen").

benennt die für die Erledigung der Tätigkeit zuständige Person beziehungsweise einen verantwortlichen (anonymen) Rollenträger.

beschreibt informell, d.h. verbal die bei der Erledigung des Bearbeitungsschrittes benötigten und/oder erzeugten Arbeitsgegenstände.

`Zuständigkeit`

stellt eine Tätigkeit in einen organisatorischen Kontext. Zuständigkeiten sind Größen eines formalisierten Modells der Organisation. Sie benennen den Arbeitsplatz einer einzelnen Person ("Mitarbeiter X") oder eine Rolle ("Kundenberater"), die im Kontext des Organisationsmodells zu konkreten Arbeitsplätzen aufgelöst werden kann.

`TätigkeitenNetz`

verwaltet die anwendungsfachlichen Abhängigkeiten zwischen den Tätigkeiten, die in einem Prozeßmuster benannt werden

kennt den Bearbeitungsstatus (erledigt/unerledigt) einzelner Tätigkeiten.

Die verschiedenen Komponenten sollen in den folgenden Abschnitten einer genaueren Betrachtung unterzogen werden.

4.4.2 Tätigkeitsbeschreibungen und Zuständigkeiten

`Tätigkeitsbeschreibungen` vergegenständlichen einzelne Bearbeitungsschritte, die bei der arbeitsteiligen Erledigung von Aufgaben anfallen. Sie sind bekanntlich die Beschreibungsprimitiva arbeitsteiliger Prozesse und werden in `Prozeßmustern` nach anwendungsfachlichen Gesichtspunkten zu vergegenständlichten Koordinationsplänen kombiniert. Mit den Beschreibungen werden keine Vorgaben für die Ausgestaltung der benannten Tätigkeiten getroffen (vgl. Abschnitt 4.3.3).

Vor dem Hintergrund des von WAM bereitgestellten Entwurfsrepertoires lassen sich `Tätigkeitsbeschreibungen` als Software-Materialien interpretieren. In Kapitel 6 wird ein Werkzeug vorgestellt, mit dem dieses Material bearbeitet werden kann.

Die Vergegenständlichung einer Tätigkeit beginnt mit ihrer Benennung. Der Name einer `Tätigkeitsbeschreibung` ("Sicherheiten einholen") läßt die Beteiligten einer Kooperationssituation "auf einen Blick" erkennen, was zu erledigen ist. Über das Wie entscheiden sie selbst und unter Berücksichtigung des gegebenen Arbeitskontextes. `Tätigkeitsbeschreibungen` informieren über Bearbeitungsschritte, die sie informell beschreiben. Die Beschreibung kann in kurzen Worten die Tätigkeit umreißen, Arbeitsgegenstände benennen, die bei ihrer Erledigung in der Regel benötigt und erzeugt werden, oder Hinweise auf zeitliche Bearbeitungsfristen geben. Entscheidend ist, daß keine dieser Informationen formale Bedeutung trägt: sie werden von Menschen interpretiert, die sie zur Kenntnis nehmen, beherzigen, aber auch ignorieren und umgehen können.

Darüberhinaus besitzen `Tätigkeitsbeschreibungen` jedoch auch eine formal interpretierbare Eigenschaft. Dies ist ihre Referenz auf ein `Zuständigkeit`-Objekt. Die Referenz stellt die Tätigkeit in einen organisatorischen Kontext, indem sie sie verantwortlich an den Arbeitsplatz eines bestimmten Mitarbeiters oder an eine Rolle bindet. `Zuständigkeit`-Objekte sind wohlgemerkt Größen eines formalisierten Modells, der Organisation.

Im Vorgriff auf den folgenden Abschnitt sei hier bereits erwähnt, daß der automatisierte Versand von Vorgangsmappen (vgl. Abschnitt 4.3.2) zwischen kooperierenden Softwareumgebungen auf der formalen Interpretierbarkeit von `Zuständigkeiten` beruht. Sie stellen den Bezug zu Umgebungen und ihren lokalen Posteingangskörben her und konkretisieren die in Abschnitt 3.3.2 skizzierte *Empfängeradresse*. Dies erklärt, warum zumindest eine rudimentäre Modellierung des organisatorischen Kontextes, in dem Kooperationsprozesse stattfinden, existieren muß. Auf diesen Punkt wird in Abschnitt 5.2 ausführlich zurückzukommen sein.

4.4.3 Tätigkeitennetze

Zwischen den einzelnen Tätigkeiten, die durch ein Prozeßmuster benannt werden, bestehen unterschiedliche Beziehungen: Tätigkeiten können sich bedingen ("der Abschluß einer Restkreditversicherung setzt voraus, daß die Kreditgenehmigung bereits erteilt worden ist") oder unabhängig voneinander sein ("das Auszahlungskonto kann unabhängig von weiteren Tätigkeiten der Kreditvergabe angelegt werden"). Die Beschreibung dieser *anwendungsfachlich motivierten* Abhängigkeiten ist ein wichtiger Bestandteil von Prozeßmustern; sie gibt einen Rahmen für die möglichen Bearbeitungswege des Transportbehälters vor, dem sie anhaftet (vgl. Abschnitt 5.1).

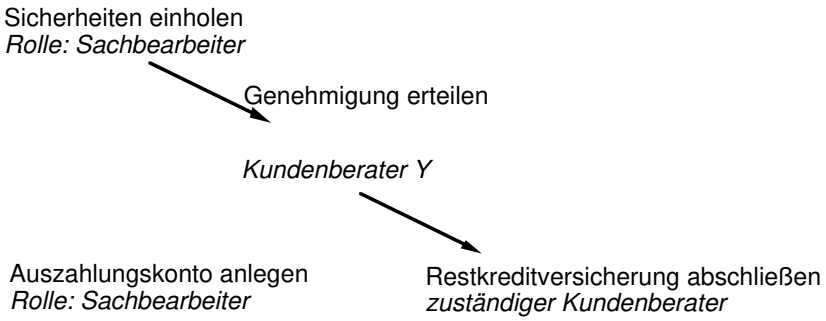


Abb. 4-4: Das Tätigkeitennetz des Prozeßmusters *Kreditvergabe* in der Graph-Darstellung; Pfeile legen anwendungsfachlich motivierte Abhängigkeiten zwischen Tätigkeiten fest.

Das Abhängigkeitsgefüge der von einem Prozeßmuster benannten Tätigkeiten wird durch ein sogenanntes Tätigkeitennetz beschrieben; Prozeßmuster *delegieren* die Verwaltung von Abhängigkeiten an ihre eingebettetes Tätigkeitennetz. Abbildung 4-4 stellt beispielhaft das Tätigkeitennetz für das Prozeßmuster *Kreditvergabe* dar.

Die Bedeutung einer Präzedenz läßt sich wie folgt beschreiben: eine Tätigkeit setzt für ihre eigene, sinnvolle Bearbeitbarkeit das "Erledigt-Sein" all jener Tätigkeiten voraus, von denen sie anwendungsfachlich abhängt. Da einzelne Tätigkeiten nicht ihre eigene Bearbeitung voraussetzen können, sind Tätigkeitennetze stets zyklenfrei.

Tätigkeitennetze sollen keine präzise Festlegung bezüglich der zeitlichen Ordnung von Tätigkeiten treffen. Sie setzen ihr jedoch einen Rahmen, insofern sie den Raum möglicher Bearbeitungswege aufspannen. Stehen alle Tätigkeiten in einer paarweisen Abhängigkeitsbeziehung, so ist der Bearbeitungsweg sogar eindeutig vorbestimmt. Andernfalls ergeben sich mehrere denkbare Bearbeitungswege ("das Auszahlungskonto kann *vor* oder *nach* Abschluß

der Restkreditversicherung angelegt werden"), durch die anwendungsfachlich unabhängige Tätigkeiten in eine zeitliche Bearbeitungsreihenfolge gebracht werden. Implizit wird hiermit folgendes ausgedrückt: der von einem Prozeßmuster behaftete Transportbehälter (die Vorgangsmappe) hat zu jedem Zeitpunkt genau einen festen *Ort* und kann deshalb auch nur zeitlich nacheinander von einer Softwareumgebung zur nächsten weitergereicht werden. Auf diesen Punkt wird noch ausführlicher in Kapitel 5 zurückzukommen sein.

Die Struktur eines Tätigkeitennetzes ändert sich dynamisch, wenn in einem Prozeßmuster

(a) eine neue Tätigkeitsbeschreibung eingefügt,
(b) eine vorhandene Tätigkeitsbeschreibung entfernt,
(c) eine neue Abhängigkeit ergänzt,
(d) eine existierende Abhängigkeit

gelöst wird. Tätigkeitennetze lassen sich demnach nicht durch Petri-Netze modellieren, die stets von statischen Strukturen ausgehen. Bei der Modellierung von Tätigkeitennetzen können wir jedoch an ein niederes Netzmodell, das *Kausalnetz*, anknüpfen und es für unsere Zwecke so erweitern, daß Änderungen an der Netzstruktur möglich, d.h. semantisch interpretierbar werden.

Petri-Netze sind mathematische Strukturen mit zwei Sorten von Elementen und einer Relation, die nur verschiedenartige Elemente, genannt Stellen und Transitionen, verbinden kann (vgl. [JV87], S.9). Netze können interpretiert werden, indem man Stellen und Transitionen mit passenden Konzeptpaaren verbindet und die Relation sinnvoll erklärt. *Kausalnetze* sind ein spezieller, zyklen- und stellenverzweigungsfreier Petrinetz-Typ, in dem Stellen als einmalige Bedingungen und Transitionen als einmalige Ereignisse aufgefaßt werden. Die Kausalrelation drückt logische Abhängigkeiten zwischen den Netztransitionen aus.

Netze eignen sich nicht nur als graphische Beschreibungsmittel; sie haben darüberhinaus auch ein definiertes zeitliches Verhalten, das durch das Schalten von Transitionen zum Ausdruck gebracht wird. In einem Kausalnetz werden beim Schalten einer Transition Vorbedingungen bei eventueller Gültigkeit von Nebenbedingungen in Nachbedingungen überführt. Die Gültigkeit einer Bedingung wird durch eine Markierung in der entsprechenden Stelle ausgedrückt. Ein Spezialfall sind *sequentielle* Kausalnetze, die keine verschiedenen nebenläufigen, d.h. logisch unabhängigen Elemente besitzen und deshalb eine eindeutige zeitliche Ordnung für das Schalten ihrer Transitionen vorgeben.

Für unseren Verwendungszusammenhang können wir Tätigkeitennetze zunächst als Kausalnetze auffassen. Dabei interpretieren wir eine Transition als Tätigkeit; das Schalten einer Transition entspricht dem "Abhaken" der vergegenständlichten Tätigkeit, die damit als durchgeführt gekennzeichnet ist. Wie läßt sich das traditionelle Modell nun aber in Hinblick auf die vier oben genannten Fälle dynamischer Restrukturierbarkeit semantisch erweitern? Das Einfügen einer neuen Tätigkeitsbeschreibung erweist sich als unproblematisch; demgegenüber müssen beim Entfernen von Tätigkeiten sowie beim Hinzufügen und Lösen von Abhängigkeiten bestimmte Bedingungen eingehalten werden. Dies sollen im folgenden anhand eines Beispielfalls erläutert werden.

Ein Kunde X wendet sich mit einem Kreditwunsch an seinen zuständigen Berater. Dieser empfiehlt, den Kredit über ein Grundpfandrecht an der Eigentumswohnung des Kunden zu sichern, da diese Kreditform aufgrund der Sicherheiten niedriger verzinst werden muß, als ein Kredit, der lediglich auf der Bonität des Kunden beruht. Der zuständige Kundenberater greift

auf das Prozeßmusterrepertoire zu und läßt sich ein Prozeßmuster auf Basis der *Kreditvergabe*-Vorlage erzeugen. Wir nehmen nun an, daß auf den Abschluß einer Restkreditversicherung verzichtet werden kann, da der Kredit vollständig durch die Hypothek abgedeckt wird. Die für die Zusammenarbeit erforderliche Koordination wird in diesem Fall also nicht adäquat durch das *Kreditvergabe*-Muster vergegenständlicht. Der zuständige Berater wandelt deshalb das Prozeßmuster ab, indem er die überflüssige Tätigkeitsbeschreibung (*Restkreditversicherung abschließen*) entfernt. Beim Entfernen von Tätigkeiten sind grundsätzlich folgende Randbedingungen einzuhalten:

(1) Die Bearbeitungshistorie eines Prozeßmusters darf nicht verfälscht werden, indem bereits als erledigt gekennzeichnete Tätigkeiten entfernt werden. Darüberhinaus muß durch das im Prozeßmuster eingebettete Tätigkeitennetz sichergestellt werden, daß sämtliche Abhängigkeiten einer zu entfernenden Tätigkeit zu weiteren Bearbeitungsschritten (im Beispiel: *Genehmigung erteilen*) gelöst werden. Abhängigkeiten dürfen nicht "ins Leere" verweisen.

Desweiteren wollen wir annehmen, daß die vom Kunden gewünschte Kredithöhe eine Genehmigung durch insgesamt zwei Kundenberater erforderlich macht. Der zuständige Berater fügt dem Prozeßmuster daher eine weitere Tätigkeitsbeschreibung (*Zweitgenehmigung erteilen*) hinzu. Alle genannten Änderungen werden wohlgemerkt im initialen Bearbeitungszustand des Prozeßmusters vorgenommen, in dem noch keine der Tätigkeiten erledigt worden sind.

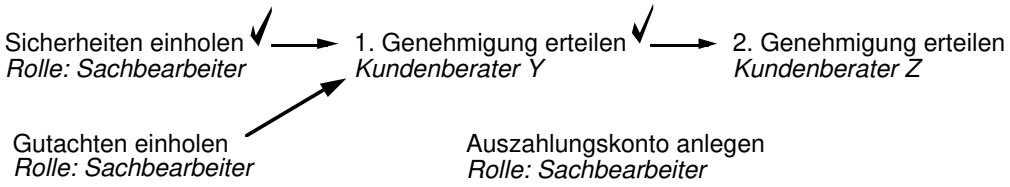


Abb. 4-5: Die dynamische Restrukturierung eines Prozeßmusters führt zu semantisch nicht interpretierbaren Konstellationen, wenn Abhängigkeiten so festgelegt werden, daß bereits als erledigt gekennzeichnete () Tätigkeiten (*1. Genehmigung erteilen*) von unerledigten abzuhängen scheinen (*Gutachten einholen*).

Entwickeln wir das Beispiel weiter. Die Kreditsicherheiten sind bereits eingeholt und der erste Kundenberater hat seine Genehmigung erteilt. Die Unterlagen gelangen nun zum zweiten

Kundenberater. Er stellt fest, daß der verantwortliche Sachbearbeiter als Sicherheitengrundlage ein altes Gutachten über den Schätzwert der Eigentumswohnung herangezogen hat, das bereits im Zusammenhang mit einem früheren Kredit erstellt wurde. Der Kundenberater hält es demgegenüber für notwendig, ein Gutachten über den aktuellen Schätzwert der Immobilie einzuholen. Dies erledigt er nicht persönlich. Er betraut damit einen Sachbearbeiter, der sich mit einem unabhängigen Gutachter in Verbindung setzt und nach Eingang des Gutachtens die Kundenunterlagen der Bank aktualisiert. Aufgrund der neuen Informationen wird die Genehmigung des ersten Kundenberaters hinfällig, so daß ihm die Kreditunterlagen erneut zur Genehmigung überstellt werden müssen.

Soll die beschriebene Situation auch weiterhin über das Prozeßmuster koordiniert werden, sind offensichtlich Anpassungen notwendig. Sie erfordern besondere Achtsamkeit, insofern ein Teil der durch das Prozeßmuster benannten Tätigkeiten (*Sicherheiten einholen*, *1. Genehmigung erteilen*) bereits als erledigt markiert worden ist. Erweiterungen dürfen nicht dazu führen, daß das Prozeßmuster in einen semantisch nicht mehr interpretierbaren Zustand überführt wird. Damit ist folgendes gemeint: wenn wir davon ausgehen, daß eine als durchgeführt markierte Tätigkeit das "Erledigt-Sein" all jener Tätigkeiten voraussetzt, von denen sie anwendungsfachlich abhängt, sollte das Festlegen neuer Abhängigkeiten im Ergebnis nicht dazu führen, daß erledigte Tätigkeiten von noch unerledigten abzuhängen scheinen. Diese Konstellation ist, im buchstäblichen Sinne, *bedeutungslos*. Der dynamischen Restrukturierbarkeit eines Prozeßmusters sind somit Grenzen gesetzt. Deshalb:

(2) Bei der Festlegung einer neuen Abhängigkeit zwischen zwei Tätigkeiten eines Prozeßmusters muß das eingebettete Tätigkeitennetz dafür sorgen, daß die Sinnhaftigkeit des Prozeßmusters gewahrt bleibt, so daß als erledigt markierte Tätigkeiten stets auch das "Erledigt-Sein" all jener Tätigkeiten implizieren, von denen sie anwendungsfachlich abhängen.

Prinzipiell sind zwei Verfahren denkbar. Konsistenz kann durch das Tätigkeitennetz gewahrt werden, indem als erledigt markierte Tätigkeiten gegebenenfalls auf den Status "unerledigt" zurückgesetzt werden (im Beispiel: *Erstgenehmigung erteilen*, vgl. Abbildung 4-5), um eine insgesamt sinnvolle Konstellation wiederherzustellen. Dieser Ansatz ist problematisch, insofern das Prozeßmuster dann einen Teil seiner Bearbeitungshistorie nicht *sichtbar* widerspiegelt. Für Benutzende ist an dem Prozeßmuster selbst nicht mehr erkennbar, daß die Erstgenehmigung des Kredits aufgrund der speziellen Arbeitssituation ein zweites Mal durch den Kundenberater Y erteilt werden muß.

Es erscheint sinnvoller, die Festlegung neuer Abhängigkeiten von vornherein so zu beschränken, daß eine sinnlose Konstellationen überhaupt gar nicht erst entsteht. Abbildung 4-6 zeigt, wie das Prozeßmuster den Erfordernissen der Situation entsprechend erweitert werden kann. Als neue Tätigkeiten werden *Gutachten einholen* und *1. Genehmigung erteilen* eingefügt. An dem Prozeßmuster ist direkt ablesbar, daß der Kreditantrag dem Kundenberater Y insgesamt zweimal zur Genehmigung überstellt wird. Das Prozeßmuster reflektiert auf diese Weise sowohl den gesamten bisherigen als auch den im weiteren geplanten Bearbeitungsprozeß.

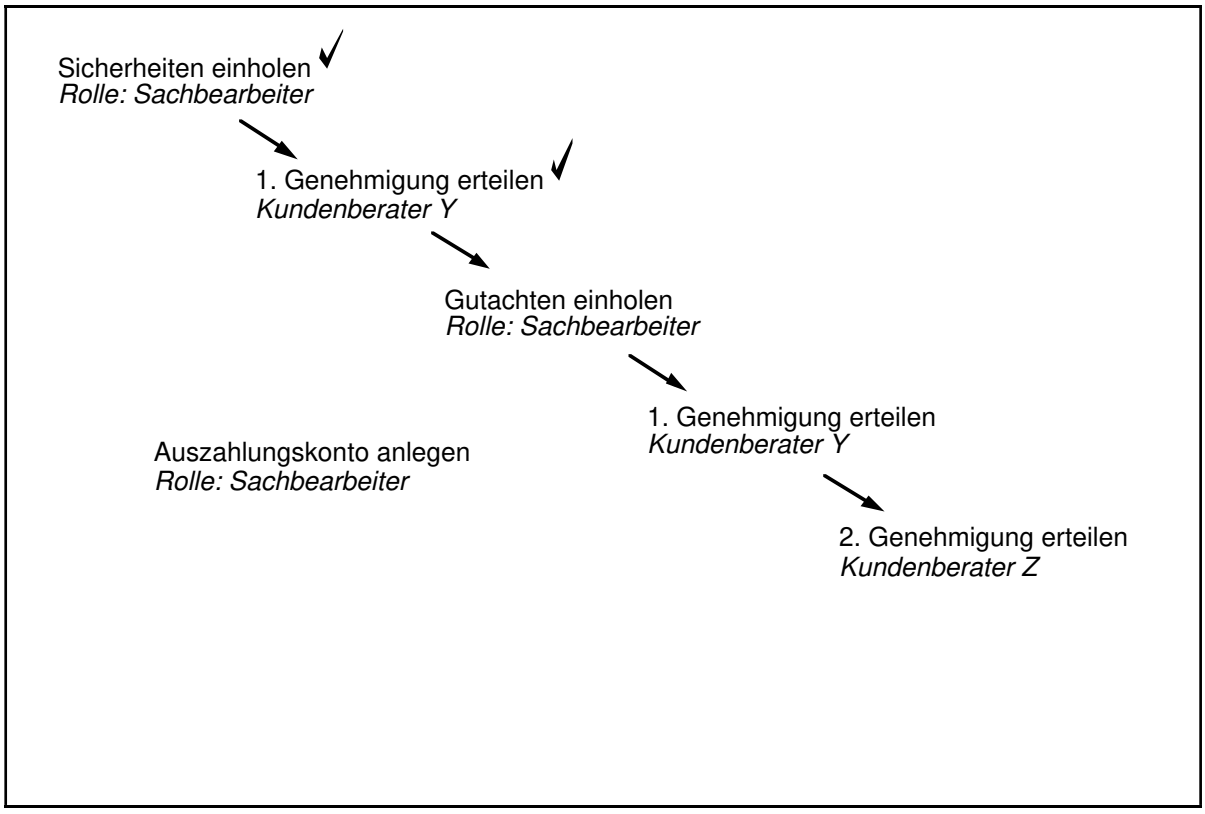


Abb. 4-6: Das situativ erweiterte Prozeßmuster *Kreditvergabe für den Kunden X*.

Der Vollständigkeit halber sei im Zusammenhang mit dem Einfügen neuer Abhängigkeiten außerdem die oben bereits erwähnte Forderung nach der Zyklenfreiheit von Tätigkeitennetzen angeführt:

(3) Das Hinzufügen einer neuen Abhängigkeit zwischen zwei Tätigkeiten darf zu keinem Zyklus innerhalb des Abhängigkeitsgefüges führen, da einzelne Tätigkeiten unmöglich ihre eigene Bearbeitung voraussetzen können.

Nun können wir noch den Fall betrachten, daß die Reihenfolge, in der Erst- und Zweitgenehmigung erteilt werden, unerheblich ist und die Abhängigkeit zwischen *1. Genehmigung erteilen* und *2. Genehmigung erteilen* daher gelöst werden kann. Dieses Vorhaben erweist sich bei dem in Abbildung 4-6 dargestellten Prozeßfortschritt als unproblematisch. Allgemein ist jedoch folgendes zu beachten:

(4) Das Lösen einer Abhängigkeit zwischen zwei Tätigkeiten A und B, wobei B für ihre eigene sinnvolle Bearbeitbarkeit das "Erledigt-Sein" von A voraussetzt, ist nur erlaubt, wenn B selbst noch unerledigt ist.

Dies bedeutet beispielsweise, daß die Abhängigkeit zwischen *Sicherheiten einholen* und *1. Genehmigung erteilen* in Abbildung 4-6 nicht mehr gelöst werden darf. Diese Einschränkung läßt sich folgendermaßen motivieren: wenn Prozeßmuster jederzeit den bisherigen Bearbeitungsprozeß widerspiegeln sollen, dürfen Abhängigkeiten, die während der Aufgabenbearbeitung berücksichtigt wurden und daher auch Aufschluß über Bearbeitungsreihenfolgen geben, nicht nachträglich entfernt werden.

4.4.4 Der Behälterklasse ProzeßmusterRepertoire

Ein Prozeßmusterrepertoire ist ein fachlicher Behälter für spezielle, aber gleichartige Materialien (Prozeßmuster) und wird durch ein `ProzeßmusterRepertoire`-Objekt technisch umgesetzt. Im Vordergrund der weiteren Betrachtungen steht die Frage, wie auf technischer Ebene Prozeßmuster auf der Basis von Vorlagen des Prozeßmusterrepertoires erzeugt werden können (vgl. Abschnitt 4.3.4).

Eine wichtige Grundlage für die Modellierung des Prozeßmusterrepertoires bildet das "Prototype"-Designpattern. Designpattern sind Muster, die als Vorlage für die Konstruktion eines softwaretechnischen Entwurfes dienen[13]. Sie beschreiben das Zusammenspiel technischer Elemente wie Klassen, Objekte und Operationen. Struktur und Dynamik eines Designpattern klären die beteiligten Komponenten, ihre Zusammenarbeit und die Verteilung von Verantwortlichkeit. Designpattern sind effiziente Vorlagen für den technischen Entwurf, weil sie von unwichtigen Details abstrahieren und den Blick auf das Wesentliche richten. Als "Mikroarchitekturen", die typische, im Entwurfsprozeß auftretende Probleme lösen, lassen sie sich flexibel zu softwaretechnischen Gesamtarchitekturen kombinieren und erleichtern dabei Kommunikation, Verständlichkeit und Dokumentation eines Entwurfs (vgl. [GHJ+95, Rie95]).

Das Prototype-Pattern charakterisieren Gamma et al. wie folgt (vgl. [GHJ+95], S. 117):

> "Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype."

Das Prototype-Pattern adressiert dabei die folgende Problemstellung: Nicht immer wird die "Erzeugungsschablone" für ein neues Objekt ausreichend durch seine Klassenbeschreibung erfaßt. Einen potentiellen Anwendungsfall bilden anspruchsvolle Grafikeditoren, mit denen Benutzer einfache Zeichenelemente zu komplexeren Grafikbausteinen kombinieren können, um sie im Sinne von Makros wiederzuverwenden. Solche Bausteine werden dynamisch, d.h. zur Laufzeit des Systems von den Benutzenden definiert und dienen dann als Vorlage für die Erzeugung einer beliebigen Anzahl gleichartiger Zeichenelemente. Die verschiedenen Grafikbausteine lassen sich offenkundig nicht durch Klassen modellieren. Klassen sind die relevanten Programmkonstrukte bei der Übersetzung eines Systems. Als *statische* Erzeugungsmuster definieren sie die ihren Objekten zugeordneten Operationen und deren interne Realisierung durch Algorithmen und Datenstrukturen; Struktur und Verhalten eines Objekte sind während seiner Lebenszeit jedoch *unveränderlich*. Die Grafikbausteine müssen demgegenüber als dynamisch erzeugbare und veränderliche Objekte bereitgestellt werden, die gleichzeitig als Spezifikation für konkret zu erzeugende (Makro-) Zeichenobjekte dienen. Das Prototype-Pattern liefert eine operationalisierbare Lösung dieses Problems:

> Spezifiziere die verschiedenen Arten von Objekten, die erzeugt werden sollen, jeweils mit Hilfe eines *prototypischen Objektes* und erzeuge neue Objekte einer Art durch Kopieren des Prototyps.

Prototypen exportieren zu diesem Zweck eine Operation zum Duplizieren ("Klonen") ihrer selbst. Sie werden bei einer Komponente registriert, die neue Objekte erzeugt, indem sie einzelne

---

[13] Im Deutschen hat sich für die Bezeichnung *design pattern* mittlerweile der Begriff des *Entwurfsmusters* eingebürgert. Im folgenden wird darauf jedoch bewußt verzichtet, um eine klare Trennlinie zum Muster-Begriff bei der Koordination kooperativer Tätigkeiten zu ziehen.

Prototypen zum Anlegen eines Duplikats auffordert. Mit dem Prototype-Pattern lassen sich somit dynamisch, d.h. zur Laufzeit des Systems neue Erzeugungsschablonen für Objekte bereitstellen, ohne daß neue Klassen definiert werden müssen. Dem Kopieren von Prototypen kommt in dieser Hinsicht eine der Klasseninstanziierung vergleichbare Bedeutung zu (vgl. [GHJ+95])[14].

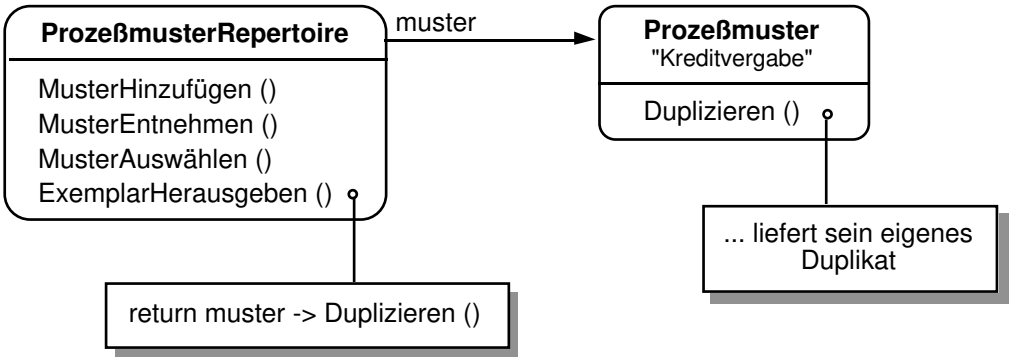


Abb. 4-7: Das `ProzeßmusterRepertoire`-Objekt erzeugt Exemplare, indem es prototypische Musterobjekte (*Kreditvergabe*) kopiert.

Das Erzeugen von Exemplaren auf der Basis von Vorlagen des Prozeßmusterrepertoires kann als Ausprägung des Prototype-Designpattern verstanden werden (vgl. Abbildung 4-7). Offensichtlich eignen Klassen sich nicht zur Modellierung spezifischer Prozeßmuster, da sie in einem laufenden Softwaresystem sowohl das Definieren neuer Prozeßmuster als auch das Verändern bereits vorhandener Muster durch Benutzende ausschließen. Dies sind jedoch wesentliche Eigenschaften, die wir für Prozeßmuster gefordert haben. Aus diesem Grund wurde in Abschnitt 4.4.1 die Materialklasse `Prozeßmuster` eingeführt, die ausreichend "generisch" ist, um mit ihr beliebige Muster der Koordination vergegenständlichen zu können.

`Prozeßmuster`-Objekte, die sich für die Koordination arbeitsteiliger Prozesse als brauchbar erwiesen haben und darüberhinaus routinemäßig von Benutzern verwendet werden, werden in

[14] Das Prototype-Pattern hat mittlerweile auch Anwendung beim Entwurf "klassenloser" objektorientierter Programmiersprachen gefunden. Solche Sprachen assoziieren mit jedem Typ ein prototypisches Objekt anstelle einer Klasse. Neue Objekte dieses Typs werden dementsprechend durch Klonen des Prototyps erzeugt, und nicht - wie es in den meisten klassenbasierten objektorientierten Sprachen (z.B. C++) der Fall ist - mit Hilfe spezieller Konstruktoren. Ein Beispiel hierfür ist die Sprache Omega (vgl. [Bla91]).

dem `ProzeßmusterRepertoire`-Objekt für die verschiedenen Software-Arbeitsumgebungen zugänglich gemacht (vgl. Abschnitt 4.3.4), damit ihre Benutzer sich bei Bedarf Exemplare der Vorlagen erzeugen können, anstatt wiederholt dieselben Muster aus elementaren Bausteinen, den Tätigkeitsbeschreibungen, zu definieren. Mit Hilfe des Prototype-Pattern läßt sich dies nun folgendermaßen technisch realisieren:

> Die verschiedenen Muster zur Koordination kooperativer Tätigkeiten werden als *prototypische* `Prozeßmuster`-Objekte in das `ProzeßmusterRepertoire`-Objekt eingefügt. Neue Prototypen können zur Laufzeit erzeugt und in das Prozeßmusterrepertoire integriert werden.

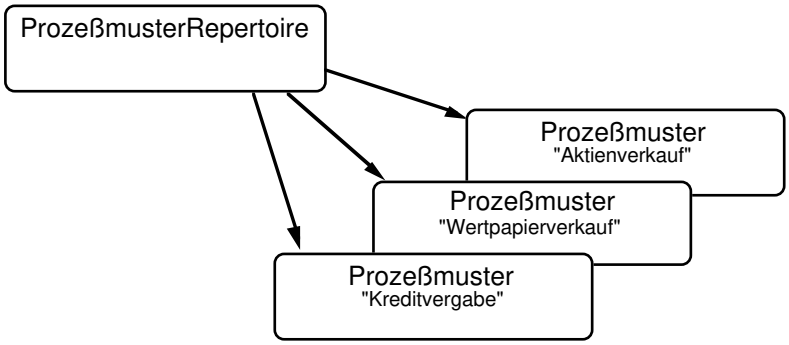


Abb. 4-8: Prozeßmuster werden als prototypische `Prozeßmuster`-Objekte in dem `ProzeßmusterRepertoire`-Objekt zusammengestellt.

> Das `ProzeßmusterRepertoire`-Objekt verwaltet die prototypischen `Prozeßmuster`-Objekte und gibt auf Anfrage Exemplare von einzelnen Vorlagen heraus. Bei der Erzeugung eines Exemplars wird das prototypische Musterobjekt kopiert.

Die verschiedenen prototypischen Muster für die Koordination kooperativer Tätigkeiten werden bei dem `ProzeßmusterRepertoire`-Objekt registriert (vgl. Abbildung 4-8). Auf Anforderung kann es Exemplare von ausgewählten Vorlagen erzeugen und herausgeben, indem es den jeweiligen Prototyp dazu veranlaßt sich zu duplizieren. `Prozeßmuster`-Objekte bieten eine entsprechende Operation an ihrer Schnittstelle an, mit der sie eine 1:1 Kopie ihrer selbst liefern (vgl. Abbildung 4-7).

4.5 Zusammenfassung

Der wesentliche Argumentationsgang dieses Kapitels soll abschließend noch einmal stichpunktartig nachvollzogen werden:

Ausgehend von dem Szenario "Kreditvergabe in einer Bank" wurden die wesentlichen Merkmale der Koordination routinisierter, arbeitsteiliger Prozesse herausgearbeitet und diskutiert, welcher Stellenwert Plänen in diesem Zusammenhang zukommen kann.

Es wurde die These vertreten, daß ein Verständnis von Koordinationsplänen als implementierbare und ausführbare Prozeßabbilder nicht mit dem Leitbild vom Arbeitsplatz für qualifizierte Tätigkeit vereinbar ist, da die betrachteten Arbeitsprozesse sich einer geschlossenen Formalisierung entziehen und Benutzenden darüberhinaus der Kontrollverlust über eine Arbeitssituation droht.

Vor diesem Hintergrund wurden alternative Ansätze vorgestellt, die den Stellenwert von Plänen relativieren und sie anstelle von Vorschriften als Ressourcen menschlichen Arbeitshandelns verstehen. In Einklang mit diesen Überlegungen formierte sich das Ziel, Benutzern in ihrer Softwareumgebung unterstützende Bausteine bereitzustellen, die ihnen Handlungsorientierung bei der Bearbeitung kooperativer Aufgaben geben, ohne dabei gleichzeitig bindende Handlungsvorgaben zu machen.

Als Lösung wurde ein von G. Gryczan konzeptionell erarbeiteter und in [Gry95] ausführlich motivierter Ansatz vorgetragen, demzufolge Pläne als handlungsorientierende Muster interpretiert und in der Softwareumgebung in Form von Materialien, den sogenannten Prozeßmuster, vergegenständlicht werden. Sie zeichnen sich dadurch aus, daß sie sich im Arbeitskontext situativ erstellen bzw. anpassen lassen und darüberhinaus den notwendigen Kontext liefern, in dem einzelne Tätigkeiten interpretiert und qualifiziert erledigt werden können.

Schließlich wurde gezeigt, daß Prozeßmuster in Bezug zu weiteren Materialien der Softwareumgebung zu sehen sind. In diesem Zusammenhang wurden ausführlich die Vorgangsmappe und das Prozeßmusterrepertoire diskutiert. Vorgangsmappen greifen den Zusammenhang zwischen einem als musterhaft erkannten Prozeß (vergegenständlicht durch ein Prozeßmuster) und seinen Materialien (zusammengestellt in einem Transportbehälter) auf; das Prozeßmusterrepertoire hilft, die bewährten Prozeßmuster für die Koordination kooperativer Tätigkeiten allgemein zugänglich bereitzustellen.

An dieser Stelle sind die wesentlichen Konzepte für die Automatisierung des Materialtransports zwischen Softwareumgebungen und die Unterstützung von Benutzern bei der Koordination ihrer kooperativen Tätigkeiten bekannt. Das folgende Kapitel soll zeigen, wie die verschiedenen Konzepte integriert sind.

5 Integration von Materialtransport- und Koordinationskonzepten

In den beiden vorangehenden Kapiteln wurden Konzepte für den Austausch von Materialien zwischen Softwareumgebungen (Kapitel 3) und für die Koordination von Tätigkeiten in arbeitsteiligen Prozessen (Kapitel 4) diskutiert. Dabei hat sich gezeigt, daß routinisierte Zusammenarbeit beide Aspekte beinhaltet:

> *Materialien*, die während der arbeitsteiligen Erledigung einer gemeinsamen Aufgabe benötigt und erzeugt werden, müssen zwischen Softwareumgebungen ausgetauscht werden.
>
> Für die am *Prozeß* Beteiligten müssen die zu erledigenden Tätigkeiten, ihre anwendungsfachlichen Abhängigkeiten und Zuständigkeiten erkenntlich und nachvollziehbar sein.

Den Zusammenhang zwischen einem als musterhaft erkannten Prozeß und seinen Materialien haben wir mit dem Konzept der *Vorgangsmappe* aufgegriffen, womit wir uns begrifflich auf einen Transportbehälter mit anhaftendem Prozeßmuster beziehen (vgl. Abschnitt 4.3.2). Vorgangsmappen bilden demnach das wesentliche Bindeglied zwischen den beiden Ebenen der Zusammenarbeit. Als vergegenständlichte Pläne für die Koordination routinisierter, arbeitsteiliger Zusammenarbeit zeigen *Prozeßmuster* mögliche Bearbeitungswege auf, die gegebenenfalls das (veränderliche) Ergebnis wechselseitiger und den Kooperationsprozeß begleitender Abstimmungsarbeit sind, im "stabilen" Zustand aber gerade den automatisierten Versand der in dem behafteten *Transportbehälter* enthaltenen Materialien ermöglichen. Dieser Punkt ist bisher nur am Rande erwähnt worden und soll daher im Rahmen dieses Kapitels detaillierter ausgearbeitet werden.

Zunächst wird der *Versandautomat* (vgl. Abschnitt 3.3.2) erneut aufgegriffen. Dabei stehen zwei Aspekte im Vordergrund der Betrachtung. Zum einen geht es darum zu klären, wie der automatisierte Versand von Vorgangsmappen zwischen Softwareumgebungen realisiert wird. Zum anderen soll aufgezeigt werden, welche zusätzlichen Anforderungen an den Automaten erwachsen, wenn nicht Personen, sondern anonyme Rollenträger als Empfänger einer Vorgangsmappe vorgesehen werden. Im Anschluß daran wird zum Problem der *Parallelität* Stellung bezogen. Vorgangsmappen schließen aus verschiedenen Gründen die gleichzeitige Bearbeitung von fachlich unabhängigen Tätigkeiten in unterschiedlichen Softwareumgebungen aus und bieten in dieser Hinsicht eine prinzipielle Angriffsfläche.

5.1 Der automatisierte Versand von Vorgangsmappen

Als eine wichtige Eigenschaft von Vorgangsmappen wurde in Abschnitt 4.3.2 ihr *Behältercharakter* herausgearbeitet. Vorgangsmappen sind Transportbehälter für Materialien, die im Rahmen rechnergestützter arbeitsteiliger Kooperationen zwischen Softwareumgebungen weitergereicht werden sollen. Materialien lassen sich mit Hilfe der Vorgangsmappen nicht nur aufgabenbezogen zusammenstellen (und zusammenhalten), sondern können darin vor allem automatisiert zwischen den kooperierenden Softwareumgebungen weitergereicht werden. Der automatisierte Versand einer Vorgangsmappe ist möglich, weil durch das zugehörige Prozeß-

muster die möglichen Bearbeitungswege und Zuständigkeiten aufgezeigt werden; letztere sind durch den Versandautomaten (vgl. Abschnitt 3.3.2) interpretierbar.

Die Zuständigkeit stellt eine Tätigkeit in einen organisatorischen Kontext, d.h. sie bindet sie verantwortlich an eine Person oder an eine Rolle. Aus der Sicht des Versandautomaten werden Zuständigkeiten mit *Orten* korreliert, an die eine Vorgangsmappe nacheinander und in einer bestimmten Reihenfolge zu überstellen ist. Zuständigkeiten stellen also den Bezug zu Software-Arbeitsumgebungen und darüber auch zu ihren lokalen Posteingangskörben her und konkretisieren die in Abschnitt 3.3.2 skizzierte *Empfängeradresse*.

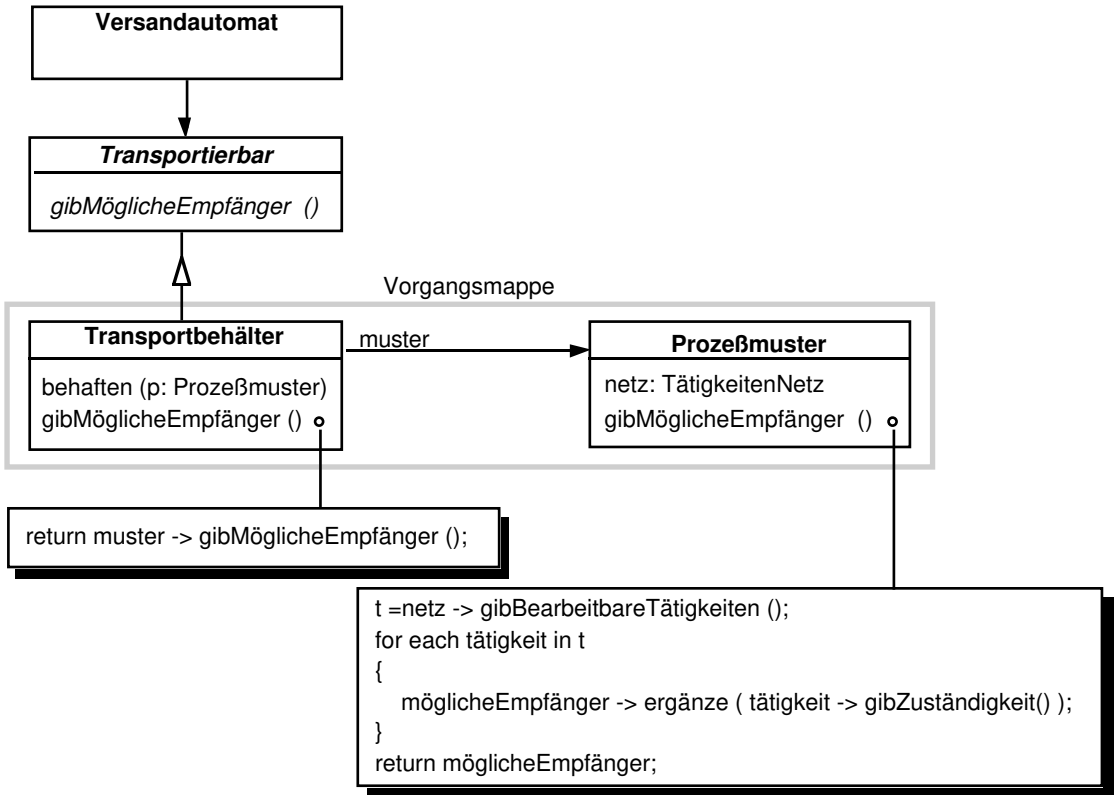


Abb. 5-1: Das Zusammenspiel zwischen `Versandautomat` und `Transportbehälter`, entkoppelt über das abstrakte `Transportierbar`-Protokoll.

Die Aspektklasse `Transportierbar` beschreibt den Verwendungszusammenhang, in dem Versandautomat und Transportbehälter zusammengeführt werden. Gegenüber seiner früheren Darstellung in Abschnitt 3.3.3 ist das `Transportierbar`-Protokoll abgewandelt worden: transportierbare Materialien stellen dem Versandautomaten auf Anfrage nicht notwendigerweise einen, sondern gegebenenfalls eine ganze Liste ihrer möglichen Empfänger bereit. Dies bedarf der Erläuterung.

Die Festlegung anwendungsfachlicher Abhängigkeiten zwischen den von einem Prozeßmuster benannten Tätigkeiten setzt einen Rahmen für die prinzipiell möglichen Bearbeitungswege der Vorgangsmappe; sie macht jedoch keine präzisen Vorgaben für die zeitliche Ordnung der Tätigkeiten. In manchen Fällen mag die Bearbeitungsreihenfolge eindeutig sein, weil alle Tätigkeiten paarweise voneinander abhängen. Häufig ergeben sich jedoch mehrere gangbare

Wege ("das Kreditauszahlungskonto kann *vor* oder *nach* dem Abschluß der Restkreditversicherung angelegt werden").

Wenn wir davon ausgehen, daß eine Vorgangsmappe zu jedem Zeitpunkt genau einen festen *Ort* hat und deshalb auch nur nacheinander von einer Softwareumgebung zur nächsten weitergereicht werden kann (vgl. Abschnitt 5.3), so müssen anwendungsfachlich unabhängige Tätigkeiten notwendigerweise in eine zeitliche Bearbeitungsreihenfolge gebracht werden. Ist der weitere Bearbeitungsgang einer Vorgangsmappe nicht eindeutig, gibt es also mehrere mögliche Empfänger, so liegt es in der Verantwortlichkeit des Versandautomaten, die Alternativen zu einer eindeutigen Bearbeitungsreihenfolge aufzulösen.

Die Klasse `Transportbehälter` erbt von der Aspektklasse `Transportierbar`, deren abstrakt spezifiziertes Protokoll `gibMöglicheEmpfänger()` sie durch eine Implementation einlösen muß. `Transportbehälter`-Objekte *benutzen* zu diesem Zweck eine gleichnamige Operation, die ihr `Prozeßmuster`-Objekt an seiner Schnittstelle exportiert und die als Pseudo-Code in Abbildung 5-1 skizziert wird. Die Idee des dahinterstehenden Algorithmus soll im folgenden dargestellt werden; sie beruht im wesentlichen auf zwei Eigenschaften der Prozeßmuster:

(1) Tätigkeitennetze (vgl. Abschnitt 4.4.3) verwalten die anwendungsfachlich motivierten Abhängigkeiten zwischen den Tätigkeiten, die in einem Prozeßmuster benannt werden. Darüberhinaus wissen sie um den Bearbeitungsstatus (erledigt/ unerledigt) einzelner Tätigkeiten, die von Benutzenden als durchgeführt gekennzeichnet werden können. Das Abschließen einer Arbeitstätigkeit verändert den Gesamtbearbeitungszustand des Kooperationsprozesses und beeinflußt insbesondere die weiteren Bearbeitungsmöglichkeiten.

    Ein Tätigkeitennetz kann abhängig vom jeweiligen Zustand Auskunft über die weiteren Bearbeitungsmöglichkeiten erteilen, und zwar im Sinne nachfolgend zu erledigender Tätigkeiten. Dies läßt sich beispielhaft an dem Kreditvergabe-Muster nachvollziehen. Im initialen Bearbeitungszustand sind *Sicherheiten einholen* und *Auszahlungskonto anlegen* sinnvoll bearbeitbare Tätigkeiten, da sie von keiner anderen Tätigkeiten abhängen. Wird nun das Konto angelegt und die Tätigkeit vom zuständigen Sachbearbeiter als erledigt markiert, beschränken sich die Bearbeitungsmöglichkeiten im nächsten Schritt auf das Einholen der Sicherheiten. Alternativ könnten zunächst die Sicherheiten eingeholt werden, so daß für die weitere Bearbeitung sowohl *Genehmigung erteilen* als auch *Auszahlungskonto anlegen* zur Wahl stehen.

    Prozeßmuster greifen im ersten Schritt auf die beschriebene Funktionalität ihrer eingebetteten Tätigkeitennetze zu, um sich eine Liste der aktuell bearbeitbaren Tätigkeiten geben zu lassen.

(2) Sind diese Tätigkeiten erst einmal ermittelt, so ergeben sich im zweiten Schritt aus der für jede Tätigkeit festgelegten *Zuständigkeit* (eine Person oder ein Rolle) die möglichen nächsten Empfänger der Vorgangsmappe. Die Liste dieser Zuständigkeiten wird als Ergebnis der im `Transportierbar`-Protokoll spezifizierten Operation `gibMöglicheEmpfänger()` zurückgeliefert.

Der Versandautomat verarbeitet dieses Ergebnis in zweierlei Hinsicht. Zum einen muß er verantwortliche *Rollen* im Kontext eines aufbauorganisatorischen Modells auflösen, d.h. er muß

Personen als mögliche Träger von Rollen und damit auch als mögliche Empfänger des transportierbaren Materials ermitteln. Auf diesen Punkt wird detaillierter in Abschnitt 5.2 zurückzukommen sein. Gibt es mehrere mögliche Empfänger für das Material, muß er darüberhinaus zwischen den verschiedenen Transportalternativen wählen. Hier sei dahingestellt, welche Strategie der Automat anwendet; er kann eine zufällige Wahl treffen oder auch abhängig von der Auslastung einzelner Softwareumgebungen entscheiden. Daß "Paralleltransporte" grundsätzlich ausgeschlossen werden, d.h. Vorgangsmappen nur sequentiell den verschiedenen Softwareumgebungen zur Bearbeitung bereitgestellt werden können, bietet Anlaß zur Kritik, zu der in Abschnitt 5.3 ausführlich Stellung bezogen wird.

5.2 Rollenauflösung

Die Benennung von verantwortlichen *Rollen* für einzelne Tätigkeiten erfordert notwendigerweise ihre Auflösung im Kontext eines aufbauorganisatorischen Modells, in dem Benutzende als Träger von Rollen ermittelt werden können. Die Behandelbarkeit dieses Problems ist beim Einsatz von Technik nur dann gewährleistet, wenn ein zumindest teilformalisiertes Modell der Aufbauorganisation vorliegt.

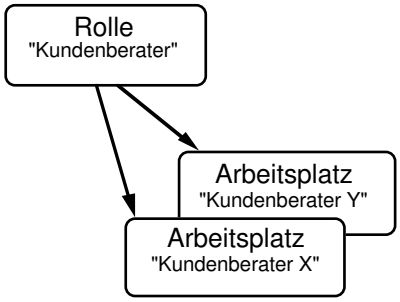


Abb. 5-2: Rollen referenzieren Arbeitsplatzumgebungen, deren Benutzer potentielle Rollenträger sind.

Möglichkeiten der Realisierung werden derzeit intensiv im Zusammenhang mit Workflow Management Systemen (vgl. Abschnitt 4.2.1.2) ausgelotet. Konzeptionell modellieren sie die Basisentitäten einer Organisationsstruktur durch eine Menge allgemeiner *Objekttypen* (Angestellter, Gruppe, Abteilung), die durch *Beziehungstypen* (ist Vorgesetzter von, berichtet an, ist Untereinheit von, gehört zu Bereich) untereinander korreliert werden (vgl. [Jab95], S. 16ff.). Hierzu orthogonal werden *Rollen* durch eine Menge von *Kompetenzen* beschrieben, die für potentielle Rollenträger definiert sein müssen. Erweiterte Rollenbeschreibungen stellen

schließlich den Bezug von Rollen zu organisatorischen Strukturen her, indem sie die Menge der möglichen Rollenträger durch Prädikate über Objekt- und Beziehungstypen einschränken. Bei der Rollenauflösung gilt es dann, neben den Kompetenzen einer Person auch ihren organisatorischen Kontext zu berücksichtigen ("die Person soll Abteilungsleiter sein und in der gleichen Abteilung wie die Person P arbeiten").

Bei der prototypischen Implementierung des hier vorgestellten Konzepts wurde ein stark vereinfachtes Modell gewählt, das personenbezogene Software-Arbeitsumgebungen ("der Arbeitsplatz des Kundenberaters Y") und Rollen durch `Arbeitsplatz`- bzw. `Rolle`-Objekte repräsentiert (vgl. Abbildung 5-2). Rollen besitzen Referenzen auf all jene Softwarearbeitsplätze, deren Benutzer potentielle Rollenträger sind. Die für einzelne Tätigkeiten festgelegten Zuständigkeiten (vgl. Abschnitt 4.4.2) werden entsprechend mit `Rolle`- bzw. `Arbeitsplatz`-Objekten parametrisiert (vgl. Abbildung 5-3).

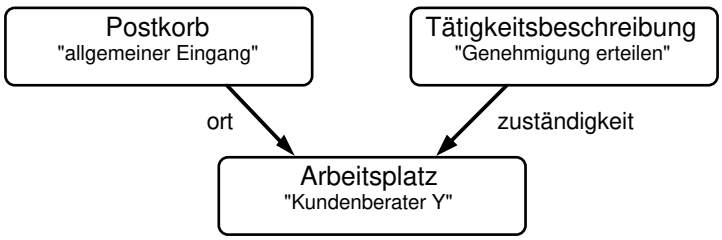


Abb. 5-3: Arbeitsplatz-Objekte stellen einen indirekten Bezug zwischen aufgabenbezoge und organisationstopologischen Objekten her.

Die Verwaltung von Rollen und Arbeitsplätzen wird in einen eigenständigen Automaten, den *Rollenverwalter*, ausgelagert, dessen Dienste dem Versandautomaten zugänglich sind (vgl. Abbildung 5-4). Mit Hilfe des Rollenverwalters kann der Versandautomat einzelne Rollen zu personenbezogenen Softwareumgebungen auflösen und so die möglichen, nächsten Empfänger für ein zu transportierendes Material ermitteln. Steht die Zielumgebung einmal fest, läßt sich auch der passende Eingangskorb finden: Postkörbe sind stets lokal zu einer Software-Arbeitsumgebung und besitzen daher gleichfalls eine Referenz auf ein Arbeitsplatz-Objekt, das mit der Zielumgebung identisch sein muß (vgl. Abbildung 5-3).

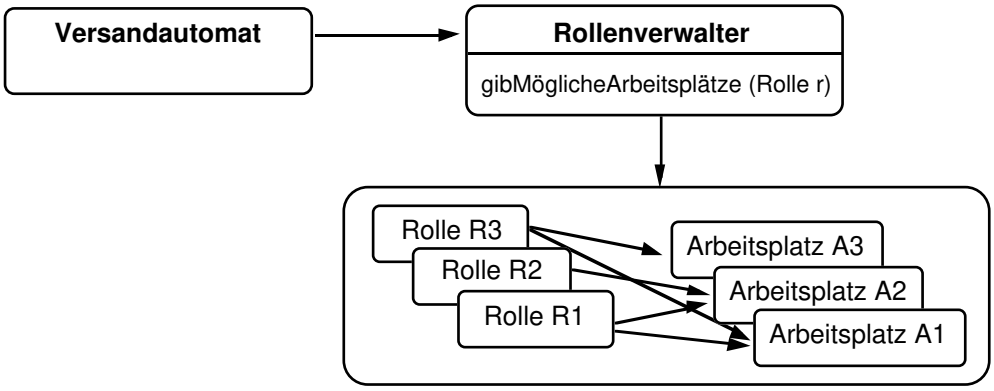


Abb. 5-4: Der `Rollenverwalter` ist für die Verwaltung von Rollen und Arbeitsplätzen zuständig. Bei der Erzeugung des `Versand-automaten` erhält dieser einen Verweis auf den `Rollenverwalter`.

Dieses einfache Modell gerät zweifelsohne an seine Grenzen, wenn in einer Organisation gleiche Tätigkeiten von einer Vielzahl unterschiedlicher Mitarbeiter erledigt werden können oder wenn die Organisation über ausgeprägte hierarchische Strukturen verfügt, die es bei der Rollenauflösung zu berücksichtigen gilt. Hier sind leistungsfähigere Auflösungsstrategien gefordert. Eine detaillierte Ausarbeitung dieser Frage sprengt jedoch den Rahmen der vorliegenden Arbeit und bleibt daher weiteren Arbeiten vorbehalten.

5.3 Parallelität

Die Modellierung von Vorgangsmappen schließt eine *parallele* Bearbeitung von anwendungsfachlich unabhängigen Tätigkeiten in verschiedenen Softwareumgebungen aus. Dieses Problem ist nicht nur für Vorgangsmappen spezifisch. Es gilt ganz allgemein für Modellansätze, bei denen der Bearbeitungspfad eines Materials, das in einer geordneten Reihenfolge unterschiedlichen Bearbeitungsstellen zugeführt werden soll, an dem Material selbst notiert wird. Solche Modellansätze werden in [Rei93] als *Objektmigrationsmodelle* klassifiziert und den sogenannten *Vorgangsmodellen* bzw. *ablauforientierten Modellen* gegenübergestellt, für die eine zentralisierte und von den Materialien des Anwendungssystems losgelöste Steuerung und Überwachung der Bearbeitung typisch ist.[15]

[15] Pankoke-Babatz et al. treffen eine ähnliche Unterscheidung. Sie differenzieren zwischen "procedure oriented models" und "form orient models" (vgl. [Pan89], S. 172ff). Letztere binden Bearbeitungspfade direkt an die zu bearbeitenden Datenobjekte, die auf diese Weise lokal interpretierbar und tranportierbar werden. Im

Den Objektmigrationsmodellen liegt die Vorstellung zugrunde, daß eine elektronische Umlaufmappe durch ein Netz von Bearbeitungsstellen migriert und dabei bearbeitet wird. Da eine Umlaufmappe konzeptionell *unteilbar* ist, wird ihre gleichzeitige, d.h. parallele Bearbeitung an verschiedenen Bearbeitungsstellen in dem Modell nicht geregelt (vgl. [Rei93], S.64). Ablauforientierte Modelle, die nicht von einem solchen Bild "fachlich zusammengehöriger" Materialien ausgehen, lösen das Problem, indem sie verschiedene Stellen parallel über anstehende Aufgaben benachrichtigen, die gerade benötigten Materialien bereitstellen und ihre getrennte Bearbeitung anschließend synchronisieren (vgl. [Jab95], S. 18).

Neuere Ansätze für Objektmigrationsmodelle versuchen, Nebenläufigkeit durch eine Partitionierung der Objekte oder durch abhängige Umlaufmappen zu unterstützen. Eine repräsentative Beispielimplementierung wurde etwa im Rahmen des ProMInanD-Projekt erarbeitet (vgl. Abschnitt 3.3.1). Eine elektronische Umlaufmappe verfügt dort über eine sogenannte Migrationsbeschreibung, die ihren automatischen Transport durch die Organisation ermöglicht. Eine Migrationsbeschreibung ist in gewisser Hinsicht vergleichbar mit dem Prozeßmuster einer Vorgangsmappe (vgl. Abschnitt 4.4.3). Sie benennt eine Reihe von durchzuführenden Bearbeitungsschritten sowie die zwischen ihnen geltenden anwendungsfachlichen Abhängigkeiten und bestimmt auf diese Weise die möglichen Bearbeitungswege der Umlaufmappe.

Parallelität kommt im Umgang mit elektronischen Umlaufmappen ins Spiel, wenn in ihrer Migrationsbeschreibung abhängige Umlaufmappen (*dependant electronic circulation folders*) definiert werden, die sich zu einem festgelegten Zeitpunkt von einem Router automatisch aus der einbettenden Umlaufmappe heraus starten und später wieder mit ihr zusammenführen lassen. Abhängige Umlaufmappen existieren zunächst losgelöst von ihrer Erzeugermappe, insofern sie parallel zu ihr bearbeitet werden und selbst weitere, abhängige Umlaufmappen starten können. Ergebnisse aus der Bearbeitung abhängiger Umlaufmappen fließen jedoch in aller Regel in die Erledigung eines späteren Arbeitsschrittes ihrer "Elternmappe" ein (vgl. [KRW90], S. 116).

Das von Karbe et al. entwickelte Konzept ließe sich technisch betrachtet sicherlich auch auf Vorgangsmappen übertragen, indem abhängige Vorgangsmappen definiert und zum gewünschten Zeitpunkt automatisch von dem Versandautomaten in Umlauf gebracht werden. Bei der prototypischen Realisierung der Softwareumgebung wurde dieser Ansatz jedoch nicht gewählt, da er dem charakteristischen Konzept von Vorgangsmappen - die Kapselung der Materialien eines Prozesses mit der Beschreibung ihrer (geplanten) Bearbeitungswege - zuwiderläuft. Dadurch wird selbstverständlich nicht ausgeschlossen, daß Einzelne bei der Erledigung eines Bearbeitungsschritts mit weiteren Benutzern zusammenarbeiten und zu diesem Zweck auch neue, "parallele" Vorgangsmappen erzeugen und in Umlauf bringen. Vorgangsmappen werden jedoch ausschließlich *auf Veranlassung von Benutzenden* erzeugt und müssen, falls ein fachlicher Zusammenhang zwischen ihnen besteht, von ihnen verwaltet und synchronisiert werden.

5.4 Zusammenfassung

Zum Abschluß dieses Kapitels halten wir zusammenfassend fest:

---

prozedurorientierten Modell werden demgegenüber Steuerung und Kontrolle durch eine zentrale Komponente wahrgenommen, die allein den Zusammenhang zwischen Datenobjekten und Bearbeitungswegen kennt.

Über die festgelegten Zuständigkeiten für einzelne Tätigkeiten eines Prozeßmusters kann der Transport von Vorgangsmappen zwischen kooperierenden Softwareumgebungen von dem Versandautomaten übernommen werden. Aus Sicht des Versandautomaten stellen Zuständigkeiten den Bezug zu Arbeitsplätzen und deren lokalen Posteingangskörben her.

Die Benennung von zuständigen Rollenträgern für Tätigkeiten erfordert, daß der Versandautomat Rollen im Kontext eines aufbauorganisatorischen Modells auflöst. Das Wissen um den Zusammenhang zwischen Rollen und Arbeitsplätzen einzelner Mitarbeiter wird konzeptionell in einem Rollenverwalter angesiedelt. Er versorgt den Versandautomaten auf Anfrage mit einer Liste derjenigen Arbeitsplätze, an denen eine bestimmte Rolle wahrgenommen werden kann.

Mögliche Alternativen für den Weitertransport einer Vorgangsmappe werden von dem Versandautomaten in eine eindeutige, d.h. sequentielle Transportreihenfolge aufgelöst. Automatisierte Paralleltransporte müssen ausgeschlossen werden, wenn die konzeptionelle Unteilbarkeit von Vorgangsmappen nicht durchbrochen werden soll.

In der vorangegangenen Diskussion sind Transportbehälter und Prozeßmuster in einen Zusammenhang mit dem Versandautomaten gestellt worden. Das folgende Kapitel stellt nun die *Werkzeuge* vor, mit denen Benutzenden die interaktive Bearbeitung der Materialien ermöglicht wird.

6 Der Prototyp

Das Ziel dieses Kapitels ist, eine plastische Vorstellung von der Softwareumgebung zu vermitteln, die prototypisch im Rahmen der vorliegenden Arbeit entwickelt wurde. Sie soll Benutzern ermöglichen, neben der Erledigung ihrer individuellen Tätigkeiten an der arbeitsteiligen Erledigung routinisierter, kooperativer Aufgaben mitzuwirken.

Der Betrachtungsschwerpunkt hat bisher auf dem Versandautomaten und den Materialien der Softwareumgebung gelegen. Anhand einer prototypischen Beschreibung, die auch graphische Darstellungen enthält, sollen nun die *Werkzeuge* zur Bearbeitung der verschiedenen Materialien in den Vordergrund gerückt werden. Dabei geht es nicht nur um die softwaretechnische Konstruktion der Werkzeuge, sondern insbesondere auch um ihre Handhabung aus der Sicht von Benutzenden.

Die Handhabung der Software-Werkzeuge wird mit Hilfe sog. Handhabungsvisionen antizipiert, die eine spezielle Ausprägung des Dokumententyps *Vision* darstellen. Visionen helfen im Entwicklungsprozeß, fachliche Anforderungen in softwaretechnische Beschreibungen zu überführen. Sie liefern eine wesentliche Grundlage für die Erstellung von Oberflächenprototypen und bilden in dieser Hinsicht den ersten Schritt in Richtung des technischen Entwurfs.

Im weiteren wird zunächst ein kurzer Überblick über die Entwicklung des Prototyps gegeben. Es folgt eine Beschreibung der einzelnen Werkzeuge nach ihrem groben Verwendungszweck; Aussehen und Verwendung verschiedener Werkzeuge sollen mit Hilfe von Handhabungsvisionen auszugsweise veranschaulicht werden. Das Kapitel wird mit einer Darstellung des technischen Werkzeugentwurfs abgeschlossen.

6.1 Entwicklung des Prototyps

Einleitend wird ein kurzer Überblick über die Entwicklung des Prototyps gegeben, in dem die Ziele und Schwerpunkte bei seiner Erstellung herausgearbeitet werden.[16]

Der Prototyp wurde im Zeitraum von November 1994 bis März 1995 stufenweise ausgebaut und während dieser Zeit mehrfach revidiert. Die Revisionen sind zu einem maßgeblichen Teil darauf zurückzuführen, daß sich das Begriffsverständnis von Prozeßmustern (vgl. Abschnitt 4.3.3) erst während des Erstellungsprozesses selbst geformt und konkretisiert hat.

Bei der Entwicklung des Prototyps konnte auf rahmenwerkartige Klassenbibliotheken zurückgegriffen werden, die im Laufe der letzten Jahre als Ergebnis von Studien- und Diplomarbeiten am Arbeitsbereich Softwaretechnik entstanden sind. Die Bibliotheken reflektieren wichtige Entwurfserfahrungen, die bei der technischen Konstruktion von Anwendungssystemen nach den Metaphern Werkzeug, Automat und Material gewonnen werden konnten: die Trennung von Interaktion und Funktion, der Ereignismechanismus sowie Prinzipien der Werkzeug- bzw. Automat- und Material-Kopplung und der Werkzeugkomposition. Einen detaillierten Überblick über diese und weitere Entwurfsprinzipien, die auch die Grundlage für die Konstruktion der im weiteren vorgestellten Werkzeuge bilden, gibt [Rie95].

---

16 Perpektiven für weiterführende Entwicklungen werden im abschließenden Kapitel dieser Arbeit aufgezeigt.

Die Klassenbibliotheken wurden in der objektorientierten Programmiersprache C++ entwickelt. Sie umfassen u.a. eine sogenannte Interaktionstypen-Bibliothek, die einen Satz vorgefertigter Bausteine für die Realisierung von grafischen Benutzungsschittstellen anbietet (Knöpfe, Textfelder, Auswahlboxen, etc.). Sie wurde speziell für X Windows Applikation entwickelt und basiert auf dem Motif Toolkit (vgl. [Rie93]). Für die Entwicklung des Prototyps wurde sie durch eine mit dem Tcl/Tk Toolkit realisierte Oberflächenbibliothek ersetzt. Als Interpreter-basierte Skriptsprache eignet sich Tcl/Tk wesentlich besser zum Prototyping von Benutzungs-oberflächen und gestattet eine effiziente Programmierung und Änderung der interaktiven Komponenten eines Systems (vgl. [Ous94]). Die Umstellung der vorhandenen IAT-Bibliothek war mit beträchtlichem Portierungsaufwand an weiteren, abhängigen Bibliotheksteilen verbunden und hat einen nicht unerheblichen Teil der gesamten Entwicklungszeit eingenommen.

Ziel bei der Erstellung des Prototyps war es, beispielhaft zu veranschaulichen, welche *grundsätzlichen* Aufgaben durch eine aus Werkzeugen, Automaten und Materialien bestehende Infrastruktur unterstützt werden müssen, um Benutzenden die Einbindung in routinisierte, arbeitsteilige Kooperationsprozesse zu ermöglichen. Insbesondere kann und soll nicht der Anspruch erhoben werden, daß die Oberflächengestaltung und Handhabung einzelner Werkzeuge bei ihrem gegenwärtigen Entwicklungsstand allen software-ergonomischen Qualitätsansprüchen gerecht wird.

Relativ frühzeitig hat sich abgezeichnet, daß bei der Implementierung des Prototyps Schwer-punkte gesetzt und bestimmte Aspekte ganz ausgeklammert werden müssen. Im Vordergrund der Betrachtung standen die Prozeßmuster, d.h. ihre Erstellung, Anpassung und grafische Aufbereitung, sowie, darauf aufbauend, die technische Realisierung des Materialtransports zwischen Softwareumgebungen.

Die vorrangige Beschäftigung mit Prozeßmustern läßt sich darauf zurückführen, daß der Einstieg in das Thema dieser Arbeit über die Beschäftigung mit Workflow Management Systemen erfolgte. Die charakteristische Eigenschaft solcher Systeme liegt bekanntlich in der strikten Befolgung vordefinierter Ablaufbeschreibungen und gestattet daher gerade *nicht* ihre flexible Anpassung an veränderte Arbeitskontexte (vgl. Abschnitt 4.2.1.2). In dieser Arbeit wurde demgegenüber eine Sichtweise eingenommen, die Pläne nicht als handlungsregulierende Steuerungen, sondern als handlungsorientierende Muster versteht und anerkennt, daß für Büroarbeit kein "richtiger" Weg der Aufgabenbearbeitung bindend vorgegeben werden kann (vgl. Abschnitt 4.2.2). Daraus entstand schließlich der Wunsch, ein Werkzeug zu konzipieren, mit dem Benutzende dynamisch Muster für die Koordination ihrer Zusammenarbeit definieren und anpassen können. Um dies realisieren, mußte zunächst einmal eine geeignete materielle Repräsentation der Muster gefunden werden, die selbst zum Gegenstand der Arbeit gemacht und durch ein Werkzeug grafisch dargestellt, sondiert und bearbeitet werden kann.

Der im weiteren Verlauf dieses Kapitels vermittelte Überblick über die vorhandenen Software-Werkzeuge, ihre Handhabung und technische Konstruktion versteht sich vor dem Hintergrund dieser gewählten Schwerpunktsetzung.

6.2 Werkzeugüberblick

Abb. 6-1: Der Werkzeugkasten der Softwareumgebung. Die Korbikone signalisiert das Eintreffen neuer Vorgangsmappen.

Für den Einzelnen mischen sich bei seiner Einbindung in Kooperationsprozesse notwendigerweise (primäre) Aufgaben, die unmittelbar der Erbringung von Dienstleistungen in einem konkreten Arbeitskontext dienen, mit (sekundären) koordinativen Aufgaben zur Schaffung, Reflektion und Anpassung zugehöriger Arbeitsbedingungen (vgl. [Obe91b]). Werkzeuge, die ein Benutzer bei der Erledigung seiner anwendungsfachlichen Aufgaben, z.B. beim Ausfüllen eines Kreditantrags, benötigt, sind stets abhängig von einem spezifischen Anwendungsgebiet zu bestimmen und werden hier ausdrücklich *nicht* behandelt. Wir können davon ausgehen, daß sie in der Softwareumgebung bereitliegen. Im Fokus der Betrachtung stehen vielmehr solche Werkzeuge, mit denen Benutzer "Koordinationsarbeit" verrichten, d.h. ihre routinisierte, arbeitsteilige Zusammenarbeit mit anderen Benutzern *steuern* und *kontrollieren*.

Mit dem Versandautomaten, den Postkörben und den Vorgangsmappen wurden bereits wesentliche Automaten und Materialien der Software-Arbeitsumgebung erläutert. Nun sollen die verschiedenen Werkzeuge zu ihrer Bearbeitung vorgestellt werden. Sie lassen sich von dem Benutzer in einem "Werkzeugkasten" auswählen und öffnen. Dabei können gleiche oder verschiedene Werkzeuge in einer selbstbestimmten Anzahl und Auswahl zur selben Zeit geöffnet sein.

6.2.1 Auflister: Prozeßmusterrepertoire

Auflister werden in der Softwareumgebung eingesetzt, um den Inhalt von fachlichen Materialbehältern anzuzeigen; der *Prozeßmusterrepertoire-Auflister* gibt einen Überblick über die bewährten Prozeßmuster zur Koordination kooperativer Tätigkeiten, die in dem *Prozeßmusterrepertoire* (vgl. Abschnitt 4.3.4) prototypisch bereitgestellt werden. Wie in Abbildung 6-2 angedeutet, kann das Auflister-Werkzeug außerdem verwendet werden, um neue Prototypen in das Repertoire einzufügen oder nicht länger benötigte daraus zu entfernen.

Das Prozeßmusterrepertoire enthält bekanntlich nur solche Muster, die sich bei der Koordination kooperativer Tätigkeiten als brauchbar erwiesen haben und deren Wiederverwendung in vergleichbaren Situationen naheliegt. Es soll daher von verschiedenen Softwareumgebungen gemeinsam genutzt werden. Einzelne erhalten mit Hilfe des Auflisters eine *Sicht* auf den gemeinsamen Behälter, dessen Materialien sie in dem Auflister selektieren und sich bei Bedarf zur Bearbeitung oder einfach nur zur Information am Bildschirm anzeigen lassen können. Bei einer gemeinsamen Nutzung des Repertoires kann ein konkurrierender Zugriff auf die darin verwalteten Materialien nicht ausgeschlossen werden.

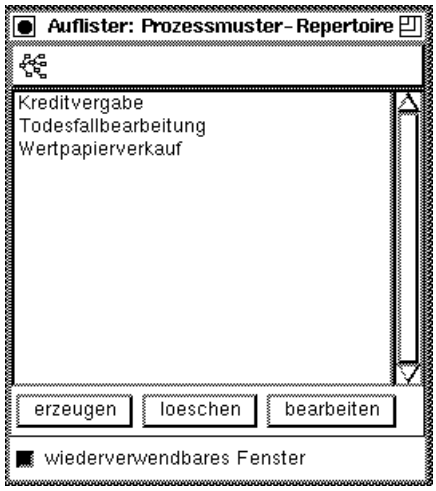


Abb. 6-2: Ein Auflistwerkzeug für das Prozeßmusterrepertoire.

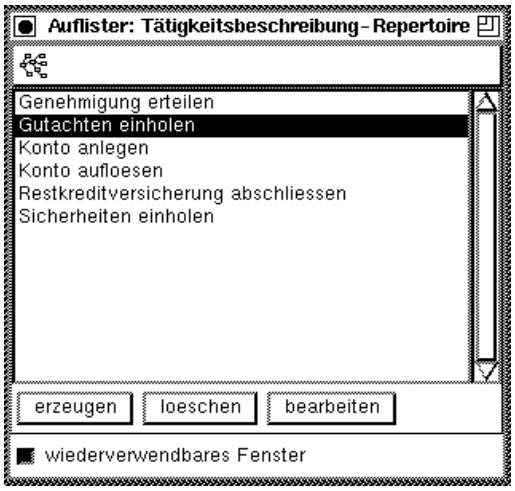


Abb. 6-3: Ein Auflistwerkzeug für das Repertoire der Tätigkeitsbeschreibungen

Um *gleichzeitige Änderungen* derselben Vorlage in verschiedenen Software-Arbeitsumgebungen zu vermeiden, wird eine exklusive Zugriffskontrolle realisiert, der das Bild von *Original und Kopien* zugrundeliegt (vgl. Abschnitt 4.3.5.3). Es wird davon ausgegangen, daß von einem Prozeßmuster in dem Repertoire jederzeit höchstens ein Original, aber beliebig viele Kopien existieren. Das Original kann nur in einer Softwareumgebung zur Zeit vorliegen und auch nur in dieser geändert werden; andere Umgebungen greifen währenddessen allenfalls sondierend auf das Material zu. In [WW94] wird dargestellt, wie diese zunächst fachlich orientierte Sichtweise vom gleichzeitigen Umgang mit gemeinsamen Materialien technisch umgesetzt werden kann.

6.2.2 Auflister: Repertoire der Tätigkeitsbeschreibungen

Tätigkeitsbeschreibungen sind bekanntlich die Bausteinprimitiva von Prozeßmustern und werden gleichfalls in einem Repertoire zusammengestellt (vgl. Abschnitt 4.3.1). Benutzer greifen bei der Erstellung und Anpassung von Prozeßmustern für die Koordination ihrer Zusammenarbeit auf diesen erweiterbaren Grundstock von Tätigkeitsbeschreibungen zurück. Wie das Prozeßmusterrepertoire ist auch das Repertoire der Tätigkeitsbeschreibungen ein von den kooperierenden Softwareumgebungen gemeinsam genutzter Behälter, dessen Inhalt von einem *Auflister* angezeigt wird (vgl. Abbildung 6-3). Neu erstellte Tätigkeitsbeschreibungen können mit Hilfe des Werkzeuges in das Repertoires eingefügt bzw. bereits vorhandene daraus entfernt werden.

6.2.3 Editor: Tätigkeitsbeschreibung

Der *Editor für Tätigkeitsbeschreibungen* dient zum Erzeugen, Ansehen und Bearbeiten einer Tätigkeitsbeschreibung. Mit Hilfe des Editors lassen sich sowohl die informellen als auch die interpretierbaren Anteile einer Tätigkeitsbeschreibung bearbeiten (vgl. Abschnitt 4.4.2). Zur informellen Charakterisierung einer Tätigkeit beschreibt ein Benutzer in wenigen Worten die Tätigkeit selbst, benennt die bei ihrer Erledigung benötigten und/oder erzeugten Materialien und gibt nötigenfalls Hinweise auf zeitliche Bearbeitungsfristen. Darüberhinaus ordnet er der Tätigkeit eine (durch den Versandautomaten interpretierbare) Zuständigkeit zu, indem er sie verantwortlich an den Arbeitsplatz eines Mitarbeiters bzw. eine Rolle bindet. Mit dem Editor

lassen sich sowohl die Materialien des Repertoires (vgl. Abschnitt 6.2.2) als auch die in einem Prozeßmuster aggregierten Tätigkeitsbeschreibungen bearbeiten.

6.2.4 Editor: Prozeßmuster

Der *Prozeßmuster-Editor* dient zum Erzeugen, Ansehen und Verändern eines Prozeßmusters. Das Werkzeug zeigt Informationen und Kommentare zu dem vergegenständlichten Prozeß an, die größtenteils von Benutzenden erstellt und erweitert werden. Mit Hilfe des eingebetteten Tätigkeitennetz-Editors (vgl. Abschnitt 6.2.5) lassen sich die durchzuführenden Tätigkeiten und ihre wechselseitigen Abhängigkeiten beschreiben. Das Werkzeug schlägt dem Benutzer darüberhinaus eine oder gegebenenfalls auch mehrere Tätigkeiten vor, die sich im gerade aktuellen Prozeßzustand sinnvoll bearbeiten lassen. Nach ihrer Erledigung hakt der Benutzer einzelne, von dem Prozeßmuster benannte Tätigkeiten ab und kennzeichnen sie damit als durchgeführt. Das Werkzeug eignet sich sowohl zur Bearbeitung der prototypischen Vorlagen des Prozeßmusterrepertoires als auch zur Bearbeitung ihrer Exemplare, mit denen die Benutzer ihre Zusammenarbeit aktiv steuern und kontrollieren.

6.2.5 Editor: Tätigkeitennetz

Der *Tätigkeitennetz-Editor* ist ein eingebettetes Subwerkzeug des Prozeßmuster-Editors und läßt sich nur aus ihm heraus öffnen. Mit Hilfe dieses Werkzeuges werden Tätigkeitsbeschreibungen in ein Prozeßmuster eingefügt, daraus entfernt und nach anwendungsfachlichen Gesichtspunkten miteinander kombiniert. Bei Erstellung und Anpassung eines Tätigkeitennetzes werden dem Benutzer nur solche Bearbeitungsmöglichkeiten angeboten, die das Netz in einem semantisch interpretierbaren Zustand belassen (vgl. Abschnitt 4.4.3). Benutzende, die im Rahmen eines kooperativen Arbeitsprozesses in aller Regel "nur" für die Erledigung einzelner Tätigkeiten zuständig sind, verschaffen sich mit Hilfe des Tätigkeitennetz-Editors einen *visuellen* Überblick über den Gesamtprozeß. Dieser grafischen Darstellung kann insbesondere der bisherige Verlauf und gegenwärtige Zustand des Prozesses entnommen werden, da bereits durchgeführte Tätigkeiten mit einem Haken versehen sind.

6.2.6 Info-Werkzeug

Das Info-Werkzeug ermöglicht Benutzenden, einzelne Vorgangsmappen, die ihre Software-Arbeitsumgebung bereits verlassen haben, aufzufinden und sich über ihren gegenwärtigen Ort und Bearbeitungszustand zu informieren.

6.2.7 Vorgangsmappen-Verwaltung

Vorgangsmappen werden in einer Softwareumgebung empfangen, bearbeitet und weitergeleitet. Die *Vorgangsmappen-Verwaltung* unterstützt in diesem Zusammenhang zwei Aufgaben. Zum einen verwaltet sie die lokalen Postkörbe, über die ein Benutzer den Austausch von Vorgangsmappen mit anderen Softwareumgebungen regelt. Zum anderen gibt sie einen Überblick über die Vorgangsmappen, die auf seinem Schreibtisch zur Bearbeitung bereitliegen. Mit Hilfe des Werkzeuges erzeugt und vernichtet ein Benutzer Ein- bzw. Ausgangskörbe; er beschriftet einzelne Körbe und reserviert sie für den Postverkehr mit anderen Softwareumge-

bungen (vgl. Abschnitt 3.3.1); er läßt sich den Inhalt ausgewählter Postkörbe anzeigen und selektiert darin enthaltene Materialien, um sie auf seinen Schreibtisch zu bewegen. Umgekehrt bewegt er Vorgangsmappen, die weitergereicht werden können, von seinem Schreibtisch in den gewünschten Postausgangskorb. Über neu eingetroffene Vorgangsmappen wird der Benutzer durch die invertierte Darstellung einer "Korb-Ikone" informiert.

6.3 Handhabungsvisionen

Ausgehend von einer fiktiven Kreditvergabesituation wird in diesem Abschnitt auszugsweise die Verwendung der zuvor dargestellten Werkzeuge aus der Sicht von Benutzenden beschrieben. Als Beschreibungsmittel dienen Handhabungsvisionen, deren Anschaulichkeit durch die Integration von Bildern der Werkzeug-Protoypen erhöht werden soll.

Die Ausgangssituation gestaltet sich wie folgt. Herr Meyer möchte für den Eigenausbau seiner Doppelhaushälfte einen Kredit in Höhe von DM 30.000 aufnehmen. Er bespricht sich mit seiner Kundenberaterin Frau Schmidt, die ihm vorschlägt, den Kredit über ein Grundpfandrecht an der Doppelhaushälfte zu sichern. Herr Meyer willigt ein. Die Kreditvergabe erfordert die Zusammenarbeit mit weiteren Mitarbeitern der Bank und soll deshalb auf der Basis eines Prozeßmusters koordiniert werden. Frau Schmidt verschafft sich mit Hilfe des Prozeßmuster-repertoire-Auflisters einen Überblick über die bereits vorhandenen Standardmuster und stellt fest, daß ein Muster für die *Kreditvergabe* zwar existiert, aber in der vorliegenden Form nicht genau auf ihre Situation paßt, da auf den Abschluß einer Restkreditversicherung verzichtet werden kann (der Kredit wird vollständig durch die Hypothek abgedeckt). Frau Schmidt beschließt, dennoch auf das *Kreditvergabe*-Muster zurückzugreifen und es anschließend den Erfordernissen ihrer Situation anzupassen.

Für die Handhabungsvisionen wurden die folgenden Benutzungssituationen ausgewählt:

bei der Erzeugung einer neuen Vorgangsmappe das zugehörige Prozeßmuster für die erforderliche Koordination der Zusammenarbeit festlegen,

eine Vorgangsmappe für den Weitertransport in die nächste Arbeitsumgebung bereitstellen,

ein Vorgangsmappe in Empfang nehmen,

ein Prozeßmuster bearbeiten,

Informationen über den gegenwärtigen Ort einer Vorgangsmappe einholen, das die eigene Softwareumgebung bereits verlassen hat.

6.3.1 Das Prozeßmuster für eine neue Vorgangsmappe festlegen

Frau Schmidt öffnet die Vorgangsmappen-Verwaltung (vgl. Abbildung 6-7) und klickt mit der Maus auf den Knopf "neue Vorgangsmappe erzeugen". Daraufhin wird der Prozeßmuster-Editor geöffnet, mit dem Frau Schmidt das zugehörige Prozeßmuster der neuen Vorgangsmappe festlegen kann.

Abb. 6-4: Der Prozeßmuster-Editor.

Der Prozeßmuster-Editor zeigt zunächst ein leeres Prozeßmuster an.

Die Kundenberaterin wählt die Option "Vorlage" des "Bearbeiten"-Menüs. Als Vorlage für das Prozeßmuster wird ihr eine Liste der verschiedenen Prototypen des Prozeßmusterrepertoires angeboten. Frau Schmidt wählt den Eintrag "Kreditvergabe". In dem Editor wird nun das bewährte Muster für die Vergabe eines Kundenkredits angezeigt. Die Kundenberaterin ergänzt zunächst den Standardtext des Kommentar-Feldes um die aus ihrer Sicht wesentlichen Informationen zum vorliegenden Kreditvergabefall. Anschließend klickt sie mit der Maus auf den Knopf "Tätigkeiten-Netz". Auf dem Bildschirm erscheint nun ein weiterer Editor, in dem das Tätigkeitennetz des Prozeßmusters in einer grafischen Darstellung angezeigt wird.

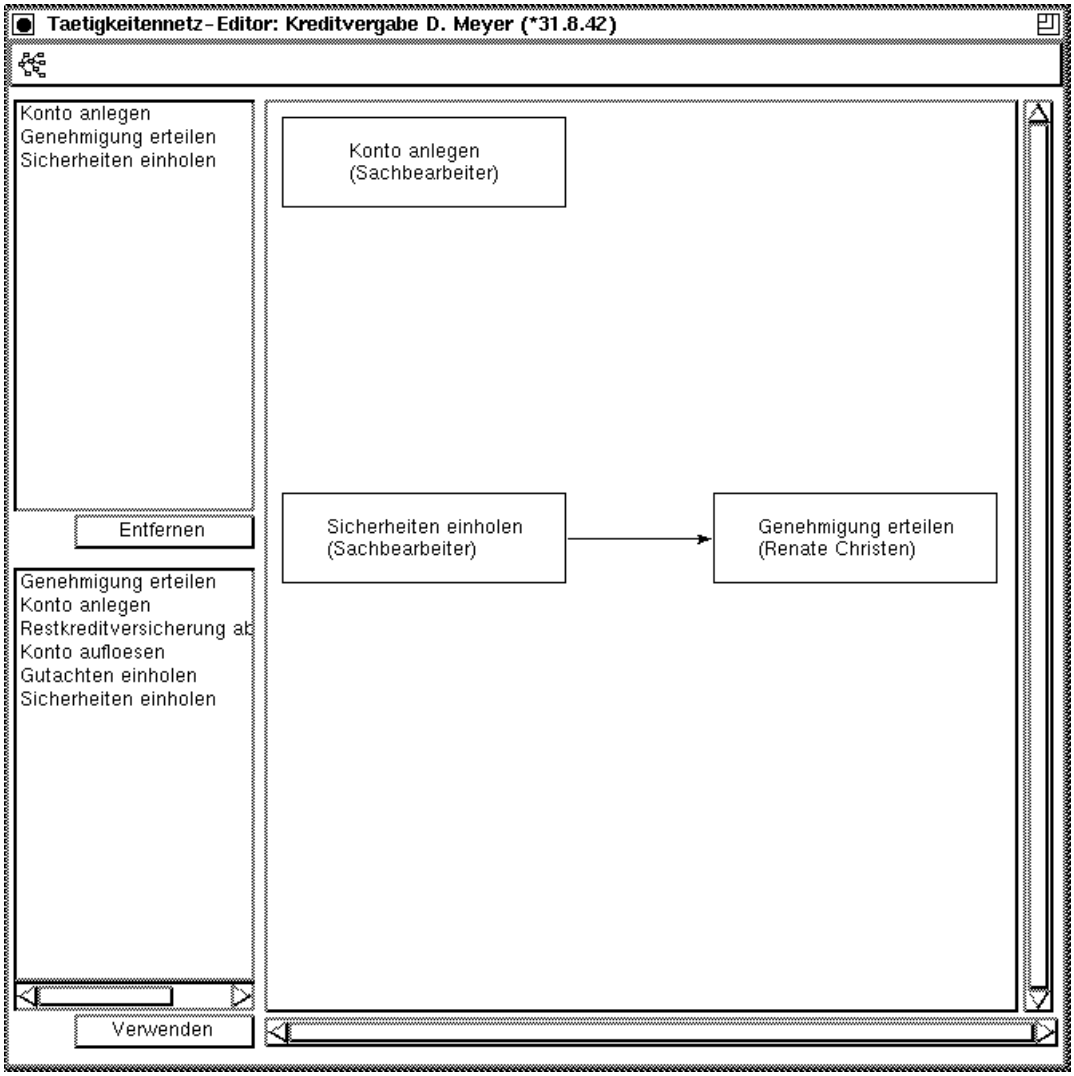


Abb. 6-5: Der Tätigkeitennetz-Editor.

Einzelne in dem Prozeßmuster benannte Tätigkeiten, die sich entfernen lassen, ohne das Netz in einen semantisch nicht mehr interpretierbaren Bearbeitungszustand zu überführen (vgl. Abschnitt 4.4.3), werden in einer separaten Liste angezeigt (oben links im Bild). Darüberhinaus werden auch die Tätigkeitsbeschreibungen des Repertoires, die bei der Erstellung bzw. situierten Anpassung des Prozeßmusters zur Verwendung stehen, in einer Liste aufgeführt (unten links im Bild).

Um das Prozeßmuster an die Erfordernisse ihrer Situation anzupassen, selektiert Frau Schmidt in der oberen Liste den Eintrag "Restkreditversicherung abschließen" und wählt den Knopf "Entfernen". Die Tätigkeitsbeschreibung wird daraufhin aus der Liste gelöscht und ist auch in der grafischen Darstellung des Tätigkeitennetzes nicht mehr sichtbar; darüberhinaus ist die anwendungsfachliche Abhängigkeit mit der gleichfalls von dem Prozeßmuster benannten Tätigkeit "Genehmigung erteilen" gelöst worden.

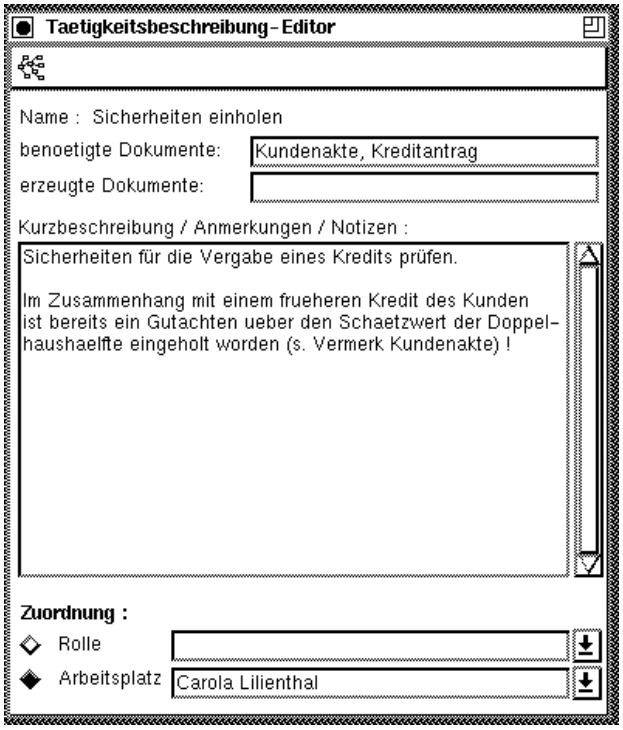


Abb. 6-6: Der Editor für Tätigkeitsbeschreibungen.

In der grafischen Darstellung erscheint unter dem Namen einer Tätigkeit die für ihre Erledigung zuständige Person bzw. Rolle. Frau Schmidt möchte die Zuordnung von "Sicherheiten einholen" ändern und führt zu diesem Zweck einen Doppelklick darauf aus. Die Tätigkeitsbeschreibung wird jetzt in einem Editor angezeigt. Anstatt der bisherigen Einstellung "Rolle: Sachbearbeiter" wählt die Kundenberaterin den "Arbeitsplatz"-Knopf und entscheidet sich in der Auswahlliste für den Arbeitsplatz von Frau Lilienthal. Sie fügt bei dieser Gelegenheit außerdem einen kurzen Bearbeitungshinweis für die Sachbearbeiterin an. Nachdem Frau Schmidt die Anpassung des Prozeßmusters abgeschlossen hat, sichert sie es über das "Bearbeiten"-Menü unter dem Namen "Kreditvergabe für D. Meyer (geb. 31.8.42)" und schließt den Prozeßmuster-Editor. Die neue Vorgangsmappe erscheint nun unter diesem Namen in der Liste der auf dem Schreibtisch von Frau Schmidt befindlichen Vorgangsmappen (vgl. Abbildung 6-7).

Bevor die Vorgangsmappe sinnvoll an den nächsten Arbeitsplatz weitergereicht werden kann, füllt die Kundenberaterin das Antragsformular aus, das sie anschließend zusammen mit der Kundenakte von Herrn Meyer in die leere Vorgangsmappe einfügt.

6.3.2 Eine Vorgangsmappe weiterreichen

Solange die Kundenberaterin die Vorgangsmappe nicht in einen Ausgangskorb bewegt, aus dem der Versandautomat sie entnehmen und an den nächsten Arbeitsplatz überstellen kann, bleibt sie auf ihrem Schreibtisch liegen. Um sie zu versenden, öffnet Frau Schmidt das Werkzeug zur Verwaltung ihrer Vorgangsmappen (vgl. Abbildung 6-7). Der untere Bereich des Werkzeuges zeigt an, welche Vorgangsmappen derzeit auf dem Schreibtisch von Frau Schmidt liegen. Die Kundenberaterin selektiert mit einem einfachen Mausklick die Vorgangsmappe "Kreditvergabe für D. Meyer (*31.8.42)" und wählt dann im Menü "Bewegen - zum Ausgangskorb" ihren allgemeinen Postausgangskorb. Die Vorgangsmappe wird daraufhin vom Schreibtisch in den gewünschten Ausgangskorb bewegt.

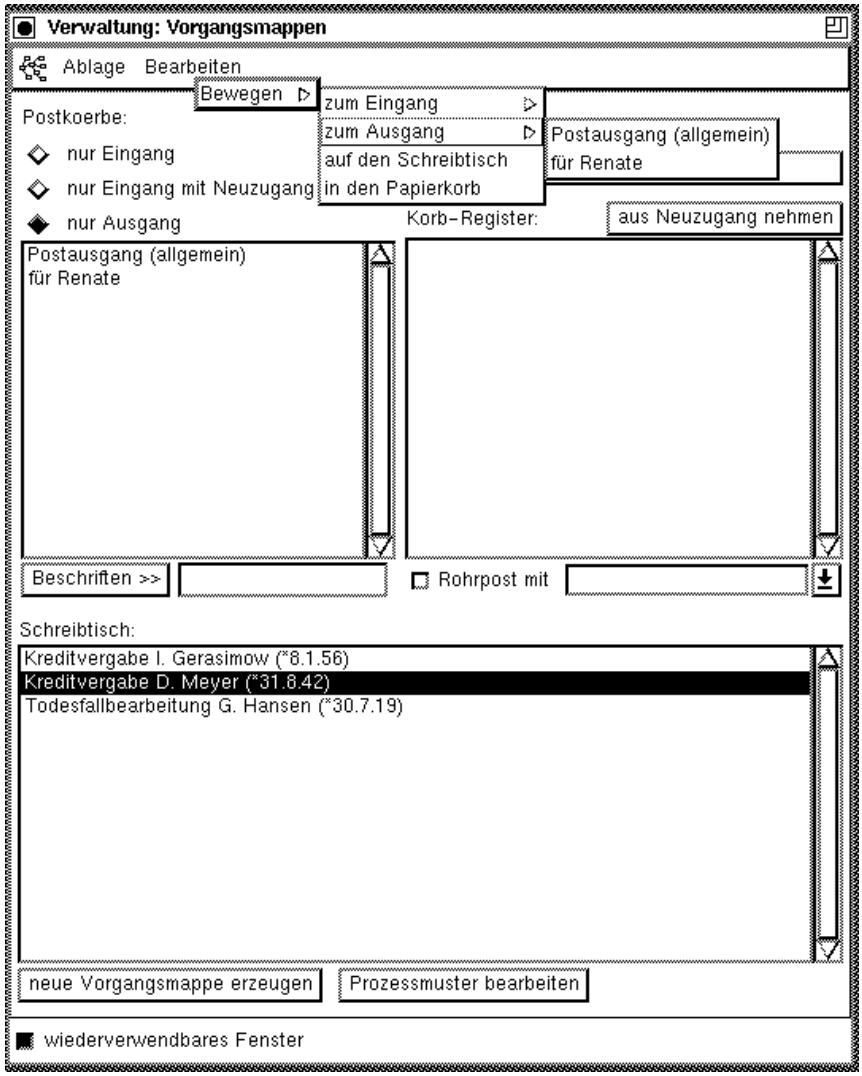


Abb. 6-7: Das Werkzeug zur Verwaltung der Vorgangsmappen einer Softwareumgebung.

6.3.3 Eine Vorgangsmappe empfangen

Der Versandautomat überstellt die Vorgangsmappe im nächsten Schritt in die Software-Arbeitsumgebung der Sachbearbeiterin Frau Lilienthal. Sie wird durch die invertierte Darstellung der Korbikone über das Eintreffen der neuen Vorgangsmappe informiert und öffnet daraufhin das Werkzeug zur Verwaltung ihrer Vorgangsmappen.

Das Werkzeug stellt im oberen Bereich die verschiedenen Postkörbe der Sachbearbeiterin dar und ist so voreingestellt, daß in dem linken Auflister nur die Eingangskörbe mit Neuzugängen angezeigt werden. Die Sachbearbeiterin erkennt auf diese Weise schnell, in welche Eingangskörbe ihr neue Vorgangsmappen zur Bearbeitung überstellt worden sind. Sie selektiert den angezeigten Korb "Posteingang (allgemein)", woraufhin in der rechten Listbox sein Inhalt angezeigt wird. Auch hier ist das Werkzeug so voreingestellt, daß es nicht den gesamten Inhalt des Postkorbes, sondern nur seine Neuzugänge anzeigt. Die Sachbearbeiterin wählt die Vorgangsmappe "Kreditvergabe für D. Meyer (*31.8.42)" aus und legt es sich mit Hilfe des "Bewegen"-Menüs auf ihren Schreibtisch.

6.3.4 Ein Prozeßmuster bearbeiten

Abb. 6-8: Im Prozeßmuster-Editor werden der Sachbearbeiterin Tätigkeiten zur Bearbeitung vorgeschlagen.

Die Sachbearbeiterin wählt auf ihrem Schreibtisch die Vorgangsmappe "Kreditvergabe für D. Meyer (*31.8.42)" aus und klickt mit der Maus auf den Knopf "Bearbeiten". Das zugehörige Prozeßmuster der Vorgangsmappe wird ihr daraufhin im Prozeßmuster-Editor angezeigt. Um sich einen Überblick über den Prozeß und seinen Bearbeitungsfortschritt zu verschaffen, läßt die Sachbearbeiterin sich zunächst das Tätigkeitennetz des Prozeßmuster anzeigen (vgl. Abbildung 6-9). Anschließend informiert sie sich über die von ihr zu erledigende(n) Tätigkeit(en).

Im unteren Bereich des Prozeßmuster-Editors (vgl. Abbildung 6-8) wird der Sachbearbeiterin die Tätigkeit "Sicherheiten einholen" zur Bearbeitung vorgeschlagen. Bevor sie damit beginnt, lädt sie die Tätigkeitsbeschreibung mit einem Doppelklick in den Tätigkeitsbeschreibung-Editor (vgl. Abbildung 6-6). Sie nimmt die Bearbeitungshinweise der Kundenberaterin Frau Schmidt zur Kenntnis und beginnt dann, mit den ihr zur Verfügung stehenden (anwendungsfachlichen) Werkzeugen und Materialien die Kundensicherheiten zu prüfen. Nach Abschluß dieses Bearbeitungsschritts markiert sie die Tätigkeit in dem Prozeßmuster als erledigt, indem sie "Sicherheiten einholen" in der Liste der vorgeschlagenen Folgetätigkeiten selektiert und mit der Maus auf den Knopf "Abhaken" klickt. Im Tätigkeitennetz-Editor wird "Sicherheiten einholen" jetzt mit einem Haken versehen.

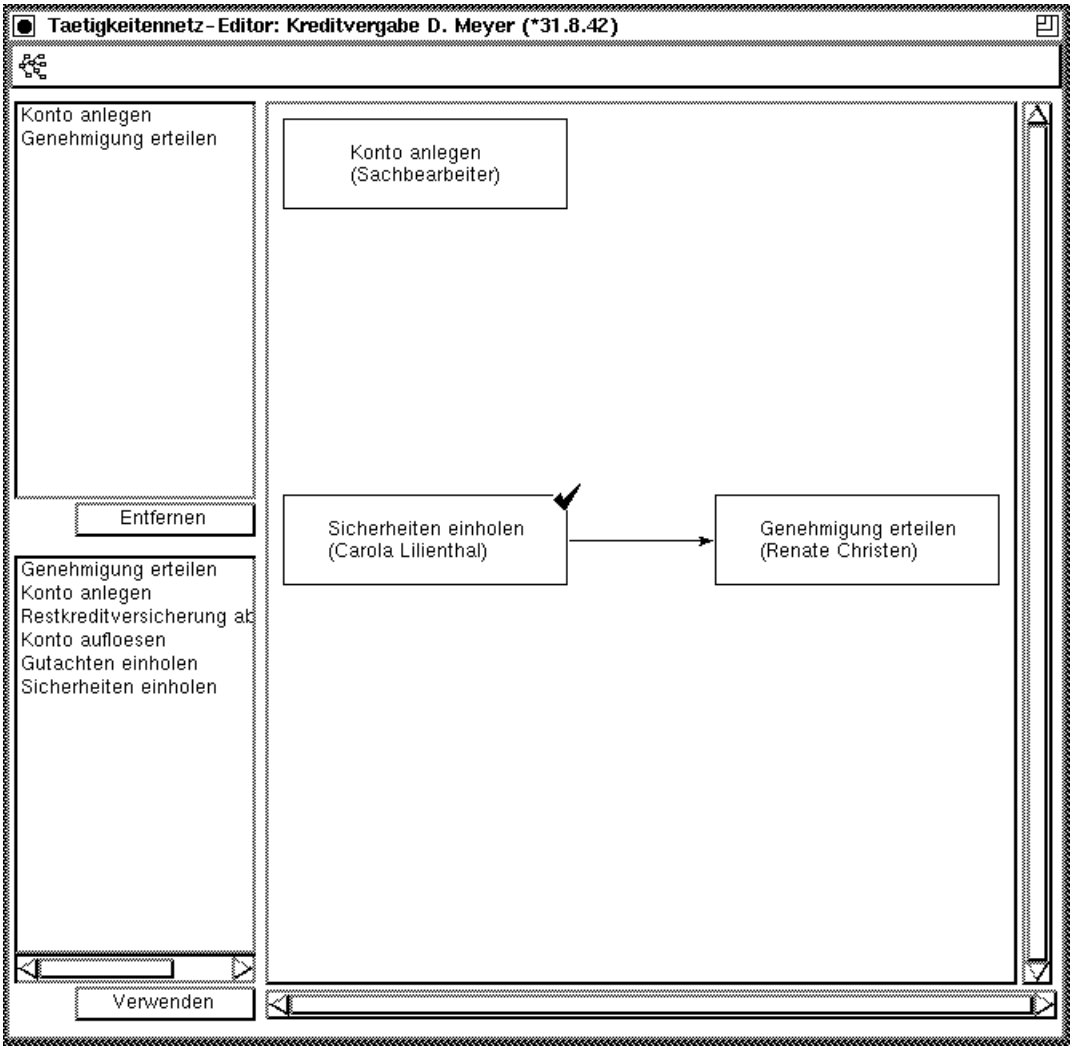


Abb. 6-9: Als durchgeführt markierte Tätigkeiten werden in dem Tätigkeitennetz mit einem Haken versehen.

6.3.5 Den aktuellen Ort einer Vorgangsmappe ermitteln

Frau Schmidt möchte sich über den gegenwärtigen Ort des Vorgangsmappe "Kreditvergabe D. Meyer (*31.8.42)" erkundigen und öffnet das Info-Werkzeug. Sie gibt als Suchmuster die Zeichenkette "Kreditvergabe*" ein und wählt als Suchkriterium "Name der Vorgangsmappe". Mit einem einfachen Mausklick auf den Knopf "Suche aktualisieren" läßt sie sich die Liste sämtlicher Vorgangsmappen anzeigen, die das Suchmuster erfüllen. Frau Schmidt selektiert in der Ergebnisliste die Vorgangsmappe "Kreditvergabe D. Meyer (*31.8.42)". Im unteren Bereich des Werkzeuges wird ihr daraufhin neben weiteren Informationen zu der Vorgangsmappe auch ihr gegenwärtiger Ort angezeigt.

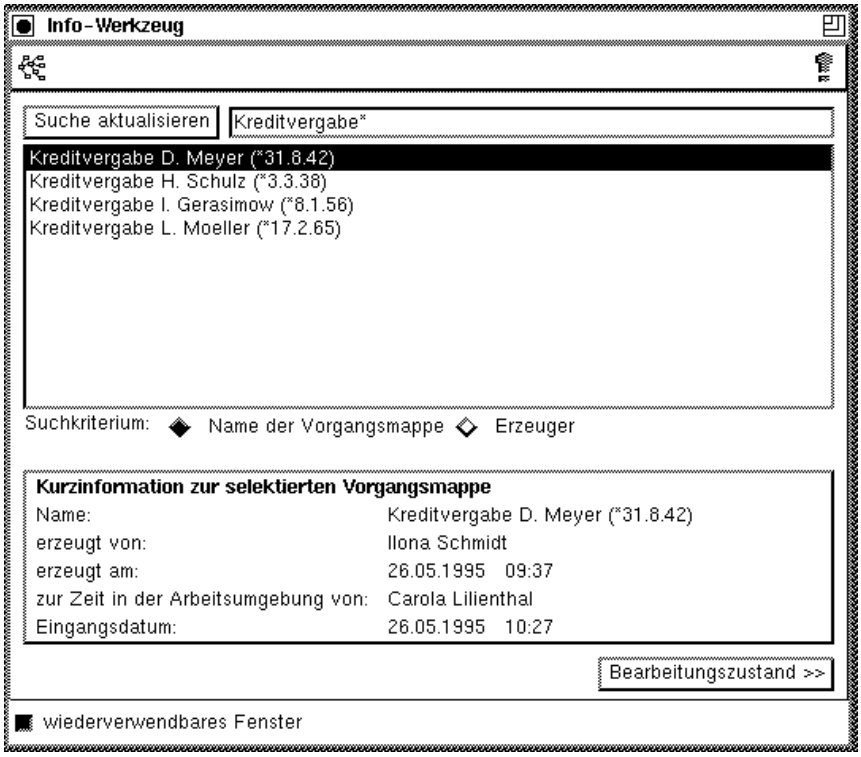


Abb. 6-10: Das Info-Werkzeug.

6.4 Technischer Werkzeugentwurf

Ziel dieses Abschnitts ist, anhand von Entwurfsbeispielen die technische Konstruktion der vorgestellten Werkzeuge nachzuzeichnen. Dabei geht es weniger darum, die implementierte Systemstruktur (vgl. Abbildung 6-11) so vollständig wie möglich zu dokumentieren, als vielmehr die verwendeten Entwurfsmuster - die *Trennung von Interaktion und Funktion*, die *Komposition von Werkzeugen* und der *Ereignismechanismus*[17] - in ihrer Anwendung auf den entwickelten Prototyp zu veranschaulichen. Interessierte Leser finden in [Rie95] eine ausführliche Darstellung dieser und weiterer Entwurfsmuster, die sich bei der Konstruktion von Anwendungssystemen nach den Metaphern Werkzeug, Automat und Material als tragfähig erwiesen haben.

Als Anschauungsbeispiel für die weitere Diskussion eignen sich der Prozeßmuster-Editor (vgl. Abschnitt 6.2.4) und der Tätigkeitennetz-Editor (vgl. Abschnitt 6.2.5). Die an ihnen verdeutlichten Konstruktionsprinzipien sind gleichermaßen auch auf die Implementierung der übrigen Werkzeuge bzw. Automaten angewandt worden.

Abbildung 6-11 zeigt die grobe Systemstruktur mit Werkzeug-, Automat- und Materialklassen. Materialien werden in einfachen Kästen, Werkzeuge und Automaten in unterlegten Kästen dargestellt. Pfeile repräsentieren die *Benutzt*-Beziehung zwischen Klassen. Bei Gruppierungen (▭) wurde der Übersichtlichkeit halber auf den Pfeil zwischen Werkzeug/Automat und

---

[17] Ein weiteres wichtiges Entwurfsmuster ist die Entkopplung von Werkzeugen bzw. Automaten und Materialien mit Hilfe von Aspekten. Da wir Aspekte bereits ausführlich im Zusammenhang mit dem Versandautomaten behandelt haben (vgl. Abschnitt 3.3.3 und 3.3.5), soll an dieser Stelle nicht erneut darauf eingegangen werden.

Material verzichtet. Aspektklassen sind in der Grobstruktur nicht eingezeichnet. Der grau hervorgehobene Teil der Systemstruktur wird im Rahmen der folgenden Kapitel näher erläutert.

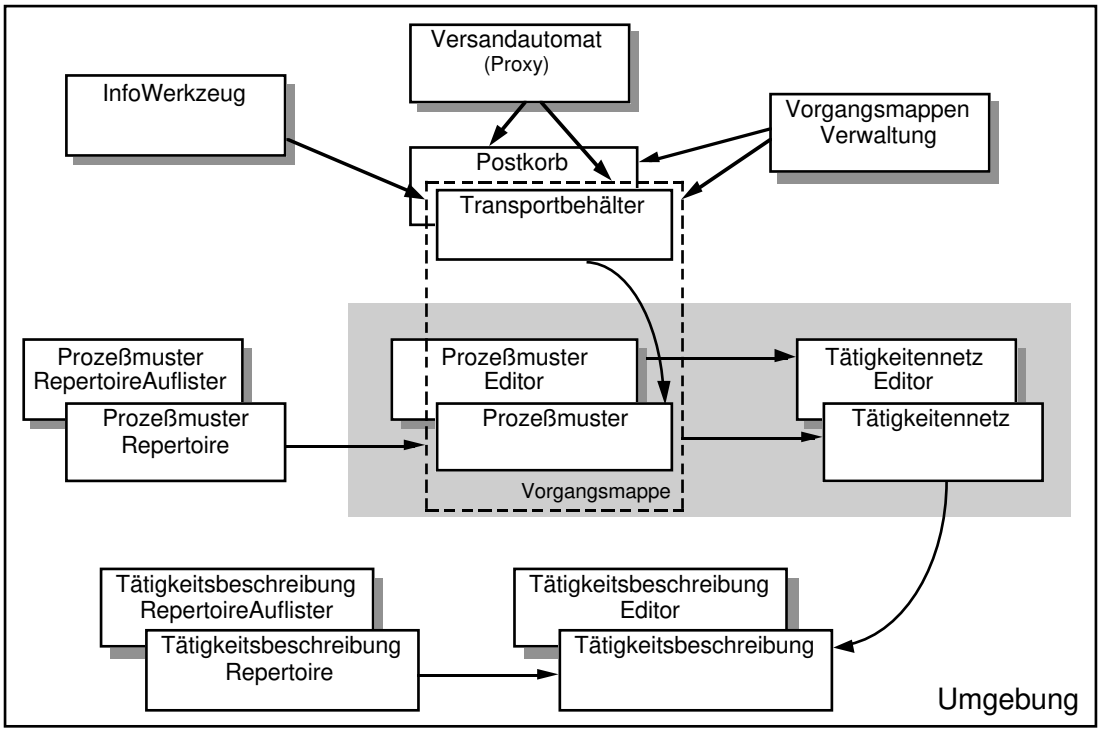


Abb. 6-11: Die grobe Systemstruktur mit Werkzeug-, Automat- und Materialklassen.

6.4.1 Trennung von Funktion und Interaktion

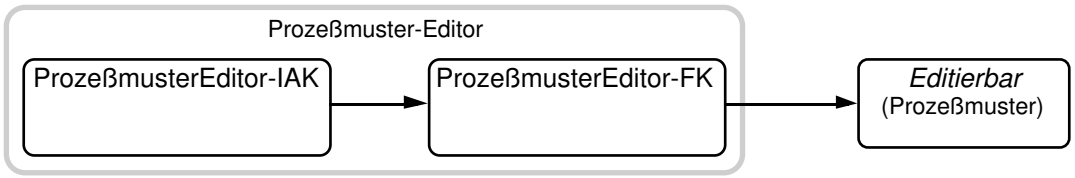


Abb. 6-12: Trennung von Interaktion und Funktion im Prozeßmuster-Editor. Die Funktionskomponente des Werkzeuges bearbeitet das editierbare Material (Prozeßmuster).

Ein wesentliches Prinzip bei der Konstruktion von Werkzeugen liegt in der Trennung von *Funktion* und *Interaktion*. Während die Präsentation eines Werkzeuges sowie die mit ihm stattfindende Interaktion von der gewählten Benutzungsumgebung abhängen (Großrechnerterminal, grafische Arbeitsplatzrechner, vorhandene Eingabegeräte), ist seine Funktionalität davon weitgehend unabhängig. Sie wird in erster Linie durch den fachlichen Arbeitszusammenhang bestimmt, in dem das Werkzeug mit seinen Materialien zusammengeführt wird.[18] Werkzeuge werden deshalb nicht als eine Werkzeug-Klasse entworfen, sondern durch eine Struktur aus zwei Komponenten technisch umgesetzt. Abbildung 6-12 zeigt dies am Beispiel des "abstrakten" Werkzeugs Prozeßmuster-Editor.

Technisch besteht der Editor daher aus zwei Objekten: einer *Interaktionskomponente* und einer *Funktionskomponente*. Die Interaktionskomponente (kurz: IAK) ist für die Präsentation und Handhabung des Werkzeugs an der Benutzungsschnittstelle zuständig und stellt das editierbare Material (Prozeßmuster) dar. Die Funktionskomponente (kurz: FK) realisiert unabhängig von der gewählten Oberflächenschnittstelle die von dem Werkzeug angebotene Funktionalität und ermöglicht die Sondierung und Manipulation des Materials. Darüberhinaus schreibt sie das Werkzeuggedächtnis fort.

---

[18] Die Trennung von Funktionalität auf der einen Seite und Präsentation und Handhabung auf der anderen Seite ist vergleichbar mit dem aus Smalltalk bekannten Model-View-Controller Paradigma (vgl. [Gol90]).

Von der IAK realisierte Handhabungsformen werden in Benutzungsaufrufe der FK umgesetzt. Die FK realisiert diese Aufrufe gegebenenfalls, in dem sie selbst wiederum (manipulierende oder sondierende) Operationen ruft, die das Material an der für sie sichtbaren Aspektschnittstelle (`Editierbar`) zur Verfügung stellt.

An ihrer Schnittstelle bietet die Funktionskomponente des Prozeßmuster-Editors eine Operation `TätigkeitAbhaken` an. Diese fachliche Operation wird dem Benutzer mit Hilfe eines Knopfs (vgl. Abbildung 6-4, unten links im Bild) von der IAK präsentiert. Wird der Knopf von einem Benutzer betätigt, setzt die IAK dies in einen Aufruf der Operation `TätigkeitAbhaken` um. Als Argument übergibt die IAK der FK die selektierte Zeichenkette in der Liste der vorgeschlagenen Folgetätigkeiten. Die FK ruft daraufhin eine entsprechende Operation des Materials, mit der die von dem Prozeßmuster benannte Tätigkeit als erledigt markiert werden kann.

Interaktionskomponenten werden auf Basis von Interaktionstypen (Fenster, Knöpfe, Menüs) realisiert, aus denen sich die Werkzeugoberfläche nach dem Baustein-Prinzip zusammensetzen läßt. Damit eine IAK an wechselnde Benutzungsschnittstellen (sog. *Graphical User Interfaces*, z.B. Motif, Tcl/Tk) angepaßt werden kann, ohne daß die zugehörige FK geändert werden muß, darf die geltende *Benutzt*-Beziehung zwischen einer IAK und ihrer FK nicht umgekehrt (d.h. bidirektional) werden. Eine FK trifft daher keinerlei Annahmen über die IAK, die ihre Funktionalität benutzt.

6.4.2 Werkzeugkomposition

Werkzeuge können rekursiv auf Subwerkzeugen zusammengesetzt werden. Daraus resultiert ein Werkzeugbaum, in der jedem Werkzeug eine bestimmte Aufgabe zukommt. Einfache Werkzeuge realisieren die ihnen zugedachte Aufgabe durch eigene Leistung; zusammengesetzte Werkzeuge delegieren Teilaufgaben an Subwerkzeuge, deren Ergebnisse sie in Hinblick auf ihre eigene Aufgabe auswerten.

Der Prozeßmuster-Editor ist ein solches zusammengesetztes Werkzeug. Bei Erstellung und Veränderung eines Prozeßmusters bedient er sich des Tätigkeitennetz-Editors, mit dem Tätigkeitsbeschreibungen in ein Prozeßmuster eingefügt, daraus entfernt und nach anwendungsfachlichen Gesichtspunkten miteinander kombiniert werden können (vgl. Abschnitt 6.2.5). Der Tätigkeitennetz-Editor ist ebenfalls ein zusammengesetztes Werkzeug; er benutzt zwei Auflister-Subwerkzeuge, mit denen er die Liste aller entfernbaren Tätigkeitsbeschreibungen sowie die verwendbaren Tätigkeitsbeschreibungen anzeigen läßt (vgl. Abbildung 6-5). Die Kopplung des sog. *Kontextwerkzeuges* Prozeßmuster-Editor mit dem Subwerkzeug Tätigkeitnetz-Editor wird in Abbildung 6-13 veranschaulicht.

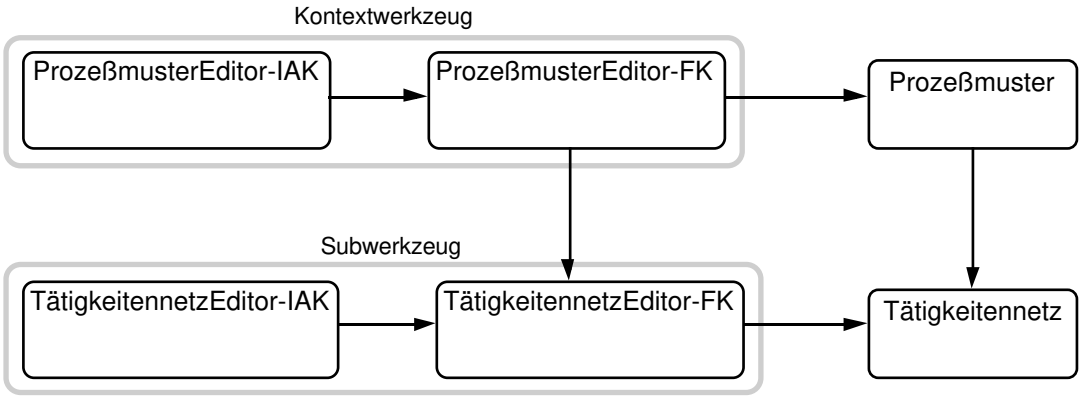


Abb. 6-13: Die Kopplung von Kontext- und Subwerkzeugen am Beispiel des Prozeßmuster-Editors und Tätigkeitennetz-Editors.

Durch die Trennung von Interaktion und Funktion (vgl. Abschnitt 6.4.1) werden Werkzeuge durch zwei Objekte realisiert. Die Funktionskomponente gilt dabei als technischer "Stellvertreter" des Werkzeuges und ist als solche für das Erzeugen und Löschen von Subwerkzeugen verantwortlich. Der konzeptuelle Werkzeugbaum wird dabei auf einen isomorphen FK-Baum abgebildet, in denen IAK's ihren zugehörigen FK's zugeordnet sind (vgl. Abbildung 6-13). Falls notwendig, lassen sich auch die IAK's untereinander zu einem IAK-Baum verbinden, der den FK-Baum nachvollzieht.

An ihrer Schnittstelle bietet die Funktionskomponente des Prozeßmuster-Editors eine Operation `OeffneTaetigkeitennetzEditor` an. Diese fachliche Operation wird dem Benutzer mit Hilfe eines Knopfes (vgl. Abbildung 6-4, unten rechts im Bild) von der Interaktionskomponente präsentiert. Betätigt der Benutzer den Knopf, setzt die IAK dies in einen Aufruf der Operation `OeffneTaetigkeitennetzEditor` um. Die FK des Prozeßmuster-Editors erzeugt daraufhin die FK des Tätigkeitennetz-Editors, der sie anschließend das eingebettete Tätigkeitennetz des bearbeiteten Prozeßmusters übergibt. Das Erzeugen der zugehörigen Sub-IAK erfolgt nach dem Erzeugen der Sub-FK und wird innerhalb der verwendeten, rahmenwerkartigen Klassenbibliotheken mit Hilfe von *Später Erzeugung* (vgl. [RZ94]) anonym und automatisch ausgeführt. Dies ist möglich, weil die Verwaltung von Interaktions- und Funktionskomponenten und ihrer Zusammengehörigkeit als generische Funktionalität in den Oberklassen des Anwendungsrahmens realisiert wird, von denen alle IAK's und FK's eines Systems erben.

6.4.3 Ereignismechanismus

Die Präsentation von Werkzeug und Material durch die Interaktionskomponente eines Werkzeuges hängt von dem Zustand der Funktionskomponente und ihrer Materialien ab. Die IAK muß deshalb über relevante Zustandsänderungen benachrichtigt werden, damit sie die Präsentation an der Benutzungsschnittstelle aktualisieren kann. Der dabei entstehende Benachrichtigungsfluß läuft der Benutzt-Beziehung zwischen IAK und FK entgegen. Damit die FK (softwaretechnisch) von der IAK unabhängig bleibt, muß vermieden werden, daß sie expliziten Bezug auf die fachliche Schnittstelle der zu benachrichtigenden IAK nimmt.

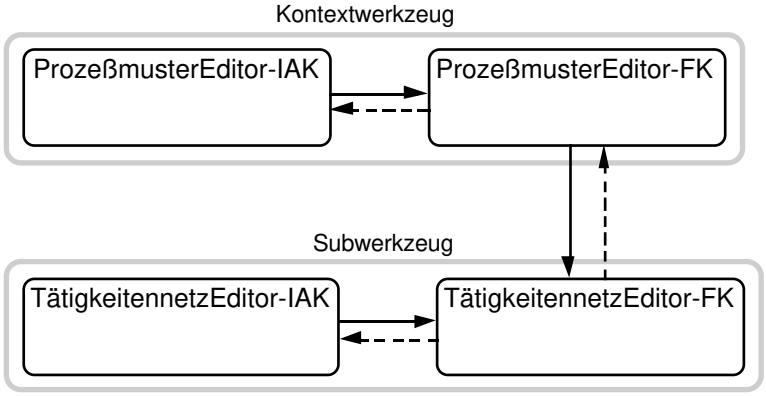


Abb. 6-14: Unabhängige Komponenten werden von abhängigen Komponenten *beobachtet* (); Zustandsänderungen werden in Form von *Ereignissen* an die (anonymisierten) Beobachter *gemeldet* (---).

Dieses Problem kann gelöst werden, indem unabhängige Komponenten sich von anonymisierten, abhängigen Komponenten *beobachten* lassen, auf deren Schnittstelle sie nur insoweit Bezug nehmen, als daß sie ihnen Zustandsänderungen in Form von Ereignissen melden. Dieser Ereignismechanismus ist nicht nur geeignet, um eine bidirektionale Benutzt-Beziehung zwischen IAK und FK zu vermeiden. Er greift gleichfalls für die Kopplung von Kontext-FK und Sub-FK (vgl. Abschnitt 6.4.2). Abbildung 6-14 zeigt dies am Beispiel der Prozeßmuster-Editor/Tätigkeitennetz-Editor Beziehung[19].

Die IAK des Tätigkeitennetz-Editors muß beispielsweise von ihrer FK benachrichtigt werden, wenn eine Tätigkeitsbeschreibung aus dem Netz entfernt wurde. Die IAK reagiert auf das Ereignis `Tätigkeitsbeschreibung_entfernt`, indem sie das Netz in seiner

[19] Die Entkopplung von Komponenten mit Hilfe des Ereignismechanismus wird von Gamma et al. als *Observer*-Pattern diskutiert (vgl. [GHJ+95], S. 293ff).

aktualisierten Form neu zeichnet. Gleichzeitig ist aber auch die Kontext-FK von der Zustandsänderung ihrer Sub-FK zu benachrichtigen. Dies kann beispielsweise dazu führen, daß die FK des Prozeßmuster-Editors die Liste der vorgeschlagenen Folgetätigkeiten aktualisiert (vgl. Abbildung 6-8).

Ereignisse lassen sich im einfachsten Fall als Konstanten modellieren; durchschaubarere Entwürfe entstehen, wenn sie als Ereignis-Objekte in die öffentliche Schnittstelle einer FK eingebettet werden. Unabhängig von der gewählten Realisierung muß Beobachtern die Möglichkeit geboten werden, sich unter einer abstrakten (d.h. anonymisierten) Schnittstelle als zu informierende Interessenten anzumelden. In [Rie95] wird vorgeschlagen, daß Beobachter sich darüberhinaus mit einer im Falle des Ereignisses aufzurufenden Operationen anmelden. Tritt das Ereignis ein, wird das Melden des Ereignisses in einen Aufruf der registrierten Operation umgesetzt. Jenachdem, ob mit Ereignis-Konstanten oder Ereignis-Objekten gearbeitet wird, ist die Funktionskomponente oder aber das Ereignis-Objekt selbst für die Verwaltung und Benachrichtigung seiner Beobachter verantwortlich. Die bei der Implementierung des Prototyps verwendeten Klassenbibliotheken unterstützen derzeit nur die Benachrichtigung über Ereignis-Konstanten.

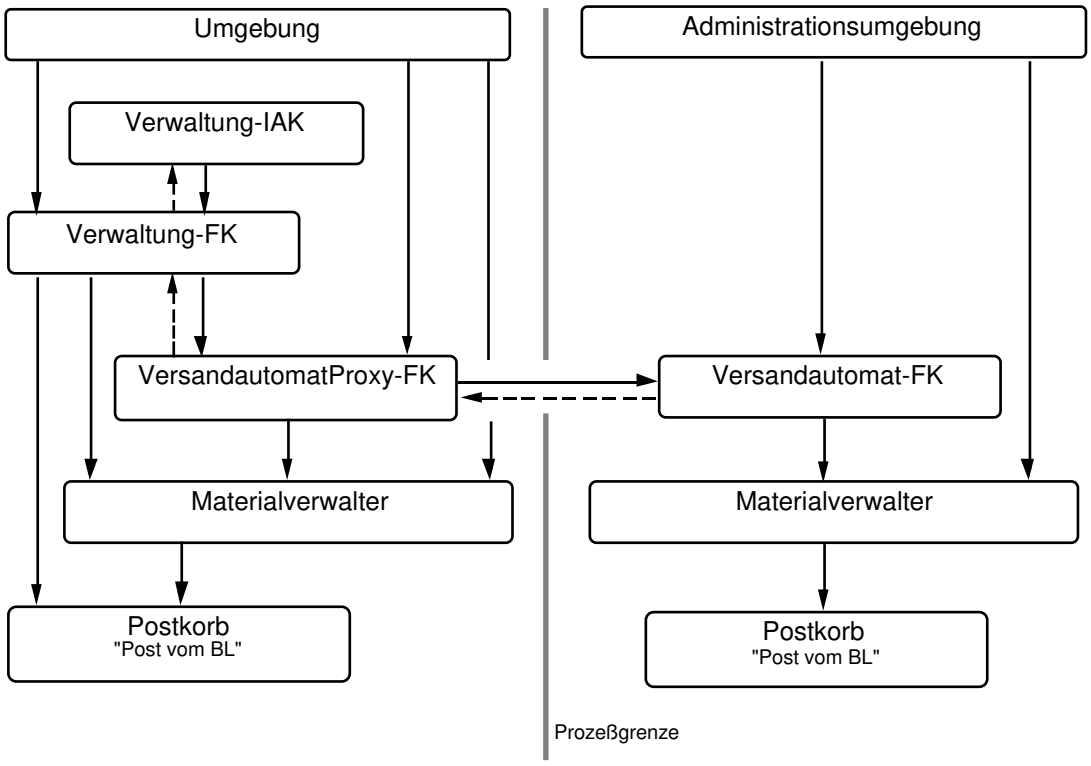


Abb. 6-15: Die Kopplung des Verwaltungswerkzeuges mit dem lokalen Versandautomat-Proxy. Das Proxy besteht nur aus einer Funktionskomponente, deren Einstellung konzeptionell mit Hilfe eines Einstellwerkzeuges erfolgt (vgl. Abschnitt 3.3.5).

Der Ereignismechanismus findet insbesondere auch bei der *Kopplung von Werkzeugen und Automaten* seine Verwendung. Bei der Darstellung des Versandautomaten als verteilte Architekturkomponente (vgl. Abschnitt 3.3.7) wurde dieses Kopplungsmuster bereits skizziert;

wir können es nun am Beispiel des Transports einer Vorgangsmappe präzisieren (vgl. Abbildung 6-15).

Die Vorgangsmappen-Verwaltung (vgl. Abschnitt 6.2.7) verwaltet u.a. die lokalen Postkörbe, über die ein Benutzer den Austausch von Vorgangsmappen mit anderen Softwareumgebungen regelt. Postkörbe werden nach ihrer Erzeugung über das lokale Proxy (vgl. Abschnitt 3.3.7) bei dem Versandautomaten zur Bedienung registriert. Zu diesem Zweck benutzt die FK des Verwaltungswerkzeuges die Dienste des Versandautomat-Proxies, das die Anfragen der FK über Prozeßgrenzen hinweg an den "echten" Versandautomaten weiterleitet. Dieser ist in einer eigenständigen Umgebung angesiedelt, die hier zur Unterscheidung von den (kooperierenden) Softwareumgebungen als Administrationsumgebung bezeichnet wird. Alle FK's erhalten ihren Zugriff auf die Postkörbe jeweils mit Hilfe einer lokalen Materialverwaltung, die die Materialbestände der Umgebung verwaltet (vgl. Abschnitt 3.3.6). Die Bedeutung des Ereignismechanismus kann am Versenden eines Transportbehälters veranschaulicht werden. Dabei ergibt sich prinzipiell der folgende Ablauf:

> Über die FK des Verwaltungswerkzeuges wird der Transportbehälter in einen Postausgangskorb gelegt.
>
> Das Versandautomat-Proxy überwacht die Postkörbe seiner Umgebung. Findet er einen nicht-leeren Postausgangskorb, stellt er eine Verbindung zum Versandautomaten in der Administrationsumgebung her.
>
> Der Versandautomat stellt sicher, daß er exklusiven Zugriff auf den Ausgangskorb erhält, entnimmt den Transportbehälter und fügt ihn (ebenfalls bei exklusivem Zugriff) in den Eingangskorb des nächsten Empfängers ein.
>
> Verläuft dieser Vorgang erfolgreich, sorgt der Versandautomat dafür, daß die Materialverwaltungen der einzelnen Umgebungen über die durch den Transport herbeigeführten Materialveränderungen benachrichtigt werden, damit sie ihre Materialbestände aktualisieren können.[20]
>
> Desweiteren benachrichtigt der Versandautomat die Versandautomat-Proxies der betroffenen Umgebungen über die Veränderung an ihren Postkörben.
>
> Das jeweilige Versandautomat-Proxy leitet dieses Ereignis im nächsten Schritt an die bei ihm als Beobachter angemeldeten FK des Verwaltungswerkzeuges weiter. Die FK reagiert auf dieses Ereignis, indem sie sich über die Materialverwaltung den aktualisierten Postkorb beschafft und selbst wiederum ihre IAK benachrichtigt, die für eine Aktualisierung der Darstellung des betreffenden Postkorbs an der Benutzungsschnittstelle sorgt.

Eine vergleichbare Konstruktion verbindet auch das Info-Werkzeug (vgl. Abschnitt 6.2.6) mit dem Versandautomat-Proxy, über dessen Dienst die Werkzeug-FK bestimmte Vorgangsmappen suchen und Informationen über sie einholen kann (z.B. ihren gegenwärtigen Ort). Veralten

---

[20] Die Benachrichtigung wird über einen sog. *Materialkoordinator* abgewickelt. Dabei handelt es sich - dem Versandautomaten vergleichbar - um einen verteilten Automaten, der in der Administrationsumgebung angesiedelt ist. Seine Aufgabe beschränkt sich darauf, konkurrierende Zugriffe der Software-Umgebungen auf Materialien nach dem Bild von Original (mit dem Recht zu ändern) und Kopien (mit dem Recht zu lesen) zu koordinieren und die Umgebungen über Änderungen an dem gemeinsam zugreifbaren Materialbestand zu informieren (vgl. [WW94], S. 11ff).

Suchergebnisse oder Informationen, so wird die FK des Info-Werkzeuges entsprechend vom Proxy benachrichtigt.

6.5 Zusammenfassung

Vor dem Hintergrund der zu unterstützenden Kooperationssituationen sind in diesem Kapitel die verschiedenen Werkzeuge einer Arbeitsumgebung vorgestellt worden, die Benutzenden die situierte Koordination ihrer kooperativen Tätigkeiten ermöglichen. Nach einer überblicksartigen Beschreibung der Werkzeugfunktionalität wurde, ausgehend von einer fiktiven Kreditvergabesituation, die Handhabung der Werkzeuge mit Hilfe von Visionen beschrieben. Entlang ausgewählter Entwurfsbeispiele sind anschließend die wesentlichen Muster zur Werkzeugkonstruktion erläutert worden.

7 Zusammenfassung und Ausblick

Dieses Kapitel reflektiert die Arbeit als Ganzes, indem es rückblickend ihre wesentlichen Stationen und Ergebnisse zusammenfaßt sowie Themen für weiterführende Arbeiten aufzeigt.

Zentrales Thema dieser Arbeit war die Entwicklung einer aus (Software-) Werkzeugen, Automaten und Materialien bestehenden Arbeitsumgebung, die es Einzelnen erlaubt, zwecks Erledigung einer gemeinsamen Aufgabe koordiniert mit anderen Benutzern zusammenzuarbeiten. Betrachtet wurden arbeitsteilige Aufgaben, die mit einer gewissen Regelmäßigkeit in Organisationen anfallen und bei deren Bearbeitung die kooperierenden Personen auf bereits etablierte Muster der Zusammenarbeit zurückgreifen können.

An den Anfang dieser Arbeit wurde die These gestellt, daß zwei kooperierende Personen ihre Zusammenarbeit implizit auf der Basis gemeinsamer Materialien koordinieren, die sie als Bestandteil von bereits erzielten bzw. noch zu erzielenden Arbeitsergebnissen untereinander austauschen. Die These ließ sich anhand eines Szenarios veranschaulichen und durch ein in der CSCW-Forschung diskutiertes Koordinationsmodell erhärten. Zunächst wurden daher die softwaretechnischen Bausteine für die Modellierung des *Materialaustausches* vorgestellt. Materialien wurden aufgabenbezogen in adressierbaren Transportbehältern zusammengestellt, in denen sie sich von einem Versandautomaten zwischen zwei Umgebungen weiterreichen ließen. Zum Empfang und zur Weitergabe der Transportbehälter wurden den Kooperationspartnern Eingangs- und Ausgangskörbe bereitgestellt.

Im Anschluß daran wurde argumentiert, daß kooperierende Benutzer über den gezielten Materialaustausch hinaus auch bei der expliziten *Koordination ihrer Zusammenarbeit* unterstützt werden können. In diesem Zusammenhang wurde untersucht, welche Rolle Koordinationsplänen als Beschreibungsmittel für die durchzuführenden Tätigkeiten, ihre fachlichen Abhängigkeiten und organisatorischen Zuständigkeiten zukommen kann. Es wurde die Position eingenommen, daß ein Verständnis von Plänen als implementierbare und ausführbare Prozeßabbilder, wie es etwa für die derzeitige Diskussion um Workflow Management Systeme charakteristisch ist, nicht mit den Gestaltungsanliegen dieser Arbeit vereinbar ist, da sich qualifizierte - individuelle und kooperative - Arbeitstätigkeit einer geschlossenen Formalisierung entzieht und involvierten Benutzern darüberhinaus der Kontrollverlust über ihre (Zusammen-) Arbeit droht. Diese Position ließ sich durch verschiedene Beiträge der CSCW-Diskussion untermauern.

Vor diesem Hintergrund wurden alternative Ansätze vorgestellt, die den Stellenwert von Koordinationsplänen relativieren und sie anstelle von handlungsregelnden Vorschriften als Ressourcen des (kooperativen) Arbeitshandelns verstehen. In Einklang mit diesen Überlegungen wurde im weiteren ein Ansatz verfolgt, demzufolge Koordinationspläne als handlungsorientierende *Prozeßmuster* interpretiert und vergegenständlicht werden. Die Besonderheit der Prozeßmuster bestand darin, daß sie sich situativ im Arbeitskontext von Benutzenden erstellen und anpassen lassen, die sie zur eigenverantwortlichen Koordination und Kontrolle ihrer Zusammenarbeit verwenden.

Im folgenden wurden deshalb die Bausteine für die Modellierung der Prozeßmuster vorgestellt. *Tätigkeitsbeschreibungen* ließen sich zur Vergegenständlichung der in einem Prozeßmuster benannten Tätigkeiten verwenden, ohne damit bindende Vorgaben für ihre fachliche Ausgestal-

tung zu machen. Mit den *Tätigkeitennetzen* wurden die fachlichen Abhängigkeiten zwischen diesen Tätigkeiten sowie deren Bearbeitungsstatus verwaltet. Besonderes Interesse galt hier der Frage der situativen Anpaßbarkeit; als Lösung wurde vorgeschlagen, ein Tätigkeitennetz als semantisch erweitertes Kausalnetz zu modellieren, das - in Grenzen - eine dynamische Restrukturierung der kausalen Beziehungen gestattet. *Zuständigkeiten* ermöglichten, eine Tätigkeit verantwortlich an eine Person oder einen Rollenträger zu binden.

Weitere Materialien, die in engem Bezug zu den Prozeßmustern zu sehen sind, wurden detailliert erläutert. Mit der *Vorgangsmappe* - einem Transportbehälter mit anhaftendem Prozeßmuster - konnte der Zusammenhang zwischen einem als musterhaft erkannten Prozeß und seinen Materialien aufgegriffen werden. Das *Prozeßmusterrepertoire* diente dem Zweck, die bewährten Prozeßmuster für die Koordination kooperativer Tätigkeiten allgemein zugänglich bereitzustellen.

Mit der *Integration von Materialtransport- und Koordinationskonzepten* schließlich wurden die Transportbehälter und der Versandautomat erneut thematisiert. Es ließ sich zeigen, daß ein Transportbehälter auf der Basis des ihm anhaftenden Prozeßmusters automatisiert zwischen einer prinzipiell beliebigen Anzahl kooperierender Umgebungen transportiert werden kann. Für den Versandautomaten ergab sich daraus die zusätzliche Anforderung, einen als Empfänger vorgesehenen Rollenträger in einem gegebenen organisatorischen Kontext zu einer konkreten Einzelperson aufzulösen. Aus der konzeptionellen Unteilbarkeit einer Vorgangsmappe mußte der Schluß gezogen, daß die darin enthaltenen Materialien nur nacheinander, nicht jedoch parallel in verschiedenen Softwareumgebungen bearbeitet werden können.

Die folgenden Punkte sollen, gleichsam stationenübergreifend, als wesentliche Ergebnisse der vorliegenden Arbeit zusammengefaßt werden:

In Hinblick auf die betrachtete Kooperationsform (arbeitsteilig, routinisiert) unterstützt die Softwareumgebung *zwei Ebenen der Zusammenarbeit*. Sie ermöglicht kooperierenden Benutzern zum einen, die bei der Aufgabenbearbeitung benötigten Materialien untereinander auszutauschen, und erlaubt ihnen zum anderen, ihre Zusammenarbeit arbeitsplatzübergreifend zu koordinieren.

Bei der Realisierung der erforderlichen technischen Infrastruktur wurde erkennbar *zwischen formalisierbaren und nicht-formalisierbaren Anteilen getrennt*. Während der Materialaustausch weitestgehend automatisiert werden konnte, ist deutlich geworden, daß eine vergleichbare Algorithmisierung der koordinativen Aufgaben weder wünschenswert noch akzeptabel umsetzbar ist.

Durch die Vergegenständlichung regelmäßiger Muster in der Zusammenarbeit sind verschiedene *Spannungsfelder gelöst* worden. Benutzer bewahren auch dann noch die Kontrolle über ihre Handlungsfolgen, wenn fachlich vorgegebene Reihenfolgen bei der Aufgabenbearbeitung existieren. Vorgangsmappen lassen sich auch nach einer Anpassung ihres vorgesehenen Bearbeitungsganges weiterhin automatisiert zwischen Umgebungen transportieren.

Durch die klare *Trennung zwischen fachlichen (d.h. inhaltlich gestalterischen) und koordinativen Anteilen* der Zusammenarbeit ist eine infrastrukturell offene Softwareumgebung entstanden, die im Unterschied zu "monolithischen" Systemen die bruchlose Einbindung beliebiger anwendungsfachlicher Werkzeuge, Automaten und Materialien

möglich macht. Individuelle und kooperative Arbeitstätigkeiten lassen sich demnach gleichermaßen in der Softwareumgebung unterstützen.

Die verschiedenen Ergebnisse sind an einem Prototyp erarbeitet bzw. veranschaulicht worden. Das vorrangige Interesse galt dabei den Prozeßmustern, d.h. ihrer Erstellung, Anpassung und grafischen Aufbereitung, sowie, darauf aufbauend, der Realisierung des Materialversands. Letzteres beinhaltete auch den Empfang und die Weitergabe von Vorgangsmappen über die lokalen Postkörbe einer Umgebung sowie das Nachforschen über Verbleib und Bearbeitungsstatus einzelner Vorgangsmappen. Die für diesen Zweck entwickelten Werkzeuge sind in Handhabung und technischer Konstruktion präsentiert worden.

Aufgrund der schwerpunktmäßigen Beschäftigung mit den Prozeßmustern wurden die Transportbehälter bei der Entwicklung nachrangig behandelt. Dies ergab sich insofern recht natürlich, als daß der Prototyp losgelöst von anwendungsfachlichen Werkzeugen, Automaten und vor allem Materialien konzipiert und implementiert wurde und demnach auch keine aufgabenbezogenen Materialien für den Austausch zwischen Softwareumgebungen zur Verfügung standen. Desweiteren versprach der Umgang mit Transportbehältern keinen besonderen Erkenntnisgewinn für das Thema dieser Arbeit, so daß in diesem Bereich auf eine Beispielimplementierung verzichtet wurde.

Aufgrund der genannten Schwerpunktsetzung sind darüberhinaus die folgenden Bereiche vernachlässigt worden:

die Konzeption eines *Rollenwerkzeuges*, mit dem Benutzer interaktiv Arbeitsplätze und Rollen registrieren und einander zuordnen können, sowie darauf aufbauend die *Einnahme von rollenspezifischen Sichten* innerhalb der Softwareumgebung[21],

die Deutung und Regelung von *Kompetenzen* bei Definition und Modifikation von Prozeßmustern.

Beide Bereiche sind bereits im Hinblick auf eine stärkere Einbeziehung des organisatorischen Kontextes zu werten, in dem Menschen tätig sind und kooperieren. Während Prozeßmuster konzeptionell unabhängig von einem spezifischen Organisationsmodell (z.B. hierarchisch-bürokratisch, professionell) und der konkreten Aufbauorganisation sind, rücken diese Bereiche zunehmend in den Vordergrund, wenn - gerade in großen Organisationen - ausgereifte Rollenauflösungsstrategien für die Aufrechterhaltung des automatisierten Materialtransports erforderlich werden.

Die Beschäftigung mit diesen Themenbereichen lag jenseits der Möglichkeiten dieser Arbeit und bleibt daher weiteren Arbeiten vorbehalten. Es wird die Herausforderung der nächsten Zeit sein, die bereits vorhandenen Bausteine weiterzuentwickeln und sie mit den skizzierten, neuen Komponenten zu integrieren, um die notwendige Grundlage für eine praktische Erprobung und empirische Bewertung des Gesamtansatzes zu schaffen, die bislang ausbleiben mußte.

---

[21] Die konzeptuelle und technische Umsetzung des Rollenwerkzeuges wird derzeit im Rahmen einer Studienarbeit ausgearbeitet.

Literaturverzeichnis

[AS94] K.R. Abbott, S.K. Sarin: *Experiences with Workflow Management: Issues for the Next Generation*. In: ACM Proceedings of the Conference on Computer Supported Cooperative Work, Chapel Hill, North Carolina (USA), 1994, 113-120.

[BCS92] R. Budde, M.-L. Christ-Neumann, K.-H. Sylla: *Tools and Materials, an Analysis and Design Metaphor.* In: Proceedings of the TOOLS 7, New York, London, Prentice Hall, 1992.

[BCS+93] R. Budde, M.-L. Christ-Neumann, K.-H. Sylla, H. Züllighoven: *Das Leitbild qualifizierter Arbeit und die Metaphern "Werkzeug" und "Material" für Herstellung und Einsatz von Computeranwendungen*. In: W. Müller, E. Senghaas-Knobloch (Hrsg.). Arbeitsgerechte Softwaregestaltung. Leitbilder, Methoden, Beispiele. Band 1. Lit, Münster, Hamburg, 1993, 125-138.

[BGZ92] U. Bürkle, G. Gryczan, H. Züllighoven: *Erfahrungen mit der objektorientierten Vorgehensweise in einem Bankenprojekt*. In: Informatik-Spektrum, 15 (5), 1992, 273-281.

[BGZ95] D. Bäumer, G. Gryczan, H. Züllighoven: *Objektorienterte Softwareentwicklung für Banken - Methodik und Erfahrungen aus einer mehrjährigen Projektpraxis.* In: OBJEKTspektrum, 3/95 (Mai/Juni 1995), 45-53.

[BKK+92] R. Budde, K. Kautz, K. Kuhlenkamp, H. Züllighoven: *Prototyping*. Springer, Berlin, Heidelberg, New York, 1992.

[Bla91] G. Blaschek: *Type-Safe Object-Oriented Programming with Prototypes - the Concepts of Omega*. In: Structured Programming, 12 (4), Springer, 1991, 217-225.

[Brö94] P. Brödner: *Gestaltung von Arbeit, Technik und Organisation zur Rückgewinnung von Wettbewerbsfähigkeit im Maschinenbau*. In: Der GMD-Spiegel, 3/94, 20-22.

[BS91] L. Bannon, K. Schmidt: *CSCW: Four Characters in Search of a Context*. In: J. Bowers & S. Benford (Hrsg.). Studies in Computer Supported Cooperative Work. Theory, Practice and Design. North-Holland, Amsterdam, New York, Oxford, Tokyo, 1991, 3-16.

[BZ90] R. Budde, H. Züllighoven: *Software-Werkzeuge in einer Programmierwerkstatt*. Berichte der Gesellschaft für Mathematik und Datenverarbeitung, Nr. 182, Oldenbourg, München, Wien, 1990.

[BZ92] R. Budde, H. Züllighoven: *Software Tools in a Programming Workshop*. In: C. Floyd, H. Züllighoven, R. Budde, R. Keil-Slawik (Hrsg.). Software Development and Reality Construction. Springer, Berlin, Heidelberg, New York, 1992, 252-268.

[Cop92] J.O. Coplien: *Advanced C++ Programmings Styles and Idioms*. Addison-Wesley, Reading, Massachusetts, 1992.

[Coy95] W. Coy: *Automat - Werkzeug - Medium*. In: Informatik-Spektrum, 18 (1), 1995, 31-38.

[Dav93] T.H. Davenport: *Process Innovation - Reengineering Work through Information Technology*. Harvard Business School Press, Boston, 1993.

[Flo84] C. Floyd: *A Systematic Look at Prototyping*. In: R. Budde, K. Kuhlenkamp, L. Mathiassen und H. Züllighoven (Hrsg.). Approaches to Prototyping. Springer, Berlin, Heidelberg, New York, 1984, 1-18.

[Flo91] C. Floyd: *Arbeitsunterlagen zur Lehrveranstaltung 'Einführung in die Softwaretechnik'*. Universität Hamburg, Fachbereich Informatik, 1991.

[Fri94] J. Friedrich: *Defizite bei der software-ergonomischen Gestaltung computergestützter Gruppenarbeit*. In: A. Hartmann, T. Herrmann, M. Rohde, V. Wulf (Hrsg.). Menschengerechte Groupware - Software-ergonomische Gestaltung und partizipative Umsetzung. Teubner, Stuttgart, 1994, 15-30.

[Gam92] E. Gamma: *Objektorientierte Software-Entwicklung am Beispiel von ET++. Design-Muster, Klassenbibliotheken, Werkzeuge*. Springer, Berlin, Heidelberg, New York, 1992.

[GHJ+95] E. Gamma, R. Helm, R. Johnson, J. Vlissides: *Design Patterns - Elements of Reusable Object-Oriented Software*. Reading, Massachusetts: Addison-Wesley, 1995.

[Gol90] A. Goldberg: *Information Models, Views, and Controllers*. Dr. Dobb's Journal, Juli 1990, 54-61.

[Gry95] G. Gryczan.: *Situierte Koordination computergestützter qualifizierter Tätigkeit über Prozeßmuster*. Dissertation, Universität Hamburg, Fachbereich Informatik, 1995 (in Vorbereitung).

[GS86] E.M. Gerson, S.L. Star: *Analyzing Due Process in the Workplace*. In: ACM Transactions on Office Information Systems, 4 (3), 1986, 257-270.

[GZ92] G. Gryczan, H. Züllighoven: *Objektorientierte Systementwicklung - Leitbild und Entwicklungsdokumente*. In: Informatik-Spektrum , 15 (5), 1992, 264-272.

[HC93] M. Hammer, J. Champy: *Reengineering the Corporation. A Manifesto for Business Revolution*. Breadly, London, 1993.

[HC94] M. Hammer, J. Champy: *Business Reengineering: Die Radikalkur für das Unternehmen*. Campus, Frankfurt/Main, New York, 1994.

[Her94] T. Herrmann: *Grundsätze ergonomischer Gestaltung von Groupware*. In: A. Hartmann, T. Herrmann, M. Rohde, V. Wulf (Hrsg.). Menschengerechte Groupware - Software-ergonomische Gestaltung und partizipative Umsetzung. Teubner, Stuttgart 1994, 65-107

[HKE92] P. Hennessy, T. Kreifelts, U. Ehrlich: *Distributed work management: activity coordination within the EuroCoOp project*. In: Computer Communications, 15 (8), 1992, 477-488.

[Jab94] S. Jablonski: *MOBILE: A Modular Workflow Model and Architecture*. erscheint in: Proceedings of the Fourth International Working Conference on Dynamic Modelling and Information Systems, September 1994, Noordwijkerhout, Niederlande.

[Jab95] S. Jablonski: *Workflow-Management-Systeme: Motivation, Modellierung, Architektur*. In: Informatik-Spektrum , 18 (1), 1995, 13-24.

[JV87] E. Jessen, R. Valk: *Rechensysteme - Grundlagen der Modellbildung*. Springer, Berlin, Heidelberg, 1987.

[KGZ93] K. Kilberth, G. Gryczan, H. Züllighoven: *Objektorientierte Anwendungsentwicklung. Konzepte, Strategien, Erfahrungen*. Vieweg, Braunschweig, Wiesbaden, 1993.

[KH91] H. Kubicek, H. Höller: *Teilautonome Arbeitsgruppen als Leitbild für Groupware-Systeme*. In: H. Oberquelle (Hrsg.). Kooperative Arbeit und Computerunterstützung - Stand und Perspektiven. Verlag für Angewandte Psychologie, Göttingen, Stuttgart, 1991.

[KHK+91] T. Kreifelts, E. Hinrichs, K.-H. Klein, P. Seuffert, G. Woetzel: *Experiences with the DOMINO Office Procedure System*. In: L. Bannon, M. Robinson & K. Schmidt (Hrsg.). Proc. ECSCW '91, Amsterdam, 1991, 117-130.

[KHW93] T. Kreifelts, E. Hinrichs, G. Woetzel: *Sharing To-Do Lists with a Distributed Task Manager*. In: G. de Michelis, C. Simone, K. Schmidt (Hrsg.). Proc. ECSCW '93. Milan, 1993, 31-46.

[KLS+84] T. Kreifelts, U. Licht, P. Sueffert, G. Woetzel: *DOMINO: A system for the specification and automation of cooperative office processes*. In: B. Myrhaug, D.R. Wilson (Hrsg.). Proc. EUROMICRO '84. North-Holland, Amsterdam, 1984, 33-41.

[KLS+94] T. Kirsche, R. Lenz, H. Schuster, H. Wedekind: *Extending Workflow Management Techniques to Control the Course of Database Conversations.* In: A. Hartmann, T. Herrmann, M. Rohde, V. Wulf (Hrsg.). Menschengerechte Groupware - Software-ergonomische Gestaltung und partizipative Umsetzung. Teubner, Stuttgart 1994, 237-258.

[KRW90] B. Karbe, N. Ramsberger, P. Weiss: *Support of Cooperative Work by Electronic Circulation Forlders*. In: ACM SIGOIS Bulletin, 11 (2-3), 1990, 109-117.

[KS90] J.L. King, S.L. Star: *Conceptual Foundations for the Development of Organizational Decision Support Systems*. In: Proceedings of the 23rd Annual Hawaii International Conference on System Sciences, Kailua-Kona. IEEE Computer Society Press, Vol. 3, 1990, 143-151.

[Maa91] S. Maaß: *Computergestützte Kommunikation und Kooperation*. In: H. Oberquelle (Hrsg.). Kooperative Arbeit und Computerunterstützung - Stand und Perspektiven. Verlag für Angewandte Psychologie, Göttingen, Stuttgart, 1991, 11-35.

[Maa94] S. Maaß: *Maschine, Partner, Medium, Welt ... Eine Leitbildgeschichte der Software-Ergonomie*. Erscheint in: H.D. Hellige (Hrsg.). Leitbilder der Informatik- und Computer-Entwicklung. Tagung der GI-Fachgruppe 'Historische Aspekte von Informatik und Gesellschaft' und des Deutschen Museums, München, Vordruck des Tagungsbandes, Bremen, 1994 (in Vorbereitung).

[MB91] J.C. McCarthy, W.M. Bluestein: *The Computing Strategy Report: Workflow's Progress*. Forrester Research Inc., Cambridge, 1991.

[MC94] T.W. Malone, K. Crowston: *The Interdisciplinary Study of Coordination*. In: ACM Computing Surveys, 26 (1),1994, 87-120.

[MO92] S. Maaß, H. Oberquelle: *Perspectives and Metaphores for Human-Computer Interaction*. In: C. Floyd, H. Züllighoven, R. Budde, R. Keil-Slawik (Hrsg.). Software Development and Reality Construction. Springer, Berlin, Heidelberg, New York, 1992, 233-251.

[Obe87] H. Oberquelle: *Sprachkonzepte für benutzergerechte Systeme*. Springer, Berlin, Heidelberg, New York, 1987.

[Obe91a] H. Oberquelle: *Kooperative Arbeit und meschengerechte Groupware als Herausforderung für die Software-Ergonomie*. In: H. Oberquelle (Hrsg.). Kooperative Arbeit und Computerunterstützung - Stand und Perspektiven. Verlag für Angewandte Psychologie, Göttingen, Stuttgart, 1991.

[Obe91b] H. Oberquelle: *CSCW- und Groupware-Kritik*. In: H. Oberquelle (Hrsg.). Kooperative Arbeit und Computerunterstützung - Stand und Perspektiven. Verlag für Angewandte Psychologie, Göttingen, Stuttgart, 1991.

[Obe94] H. Oberquelle: *Situationsbedingte und benutzerorientierte Anpaßbarkeit von Groupware*. In: A. Hartmann, T. Herrmann, M. Rohde, V. Wulf (Hrsg.). Menschengerechte Groupware - Software-ergonomische Gestaltung und partizipative Umsetzung. Stuttgart, 1994, 31-50.

[OMG91] Object Management Group. *The Common Object Request Broker: Architecture and Specification*. Revision 1.1 (OMG Document 91.12.1), 1991.

[OMG92] Object Management Group. *Object Management and Architecture Guide*. 2nd Edition (OMG Document 91.11.1), 1992.

[Ous94] J.K. Ousterhout: *Tcl and the Tk toolkit*. Reading, Massachusetts, Addison-Wesley, 1994.

[Pan89] U. Pankoke-Babatz: *Computer Based Group Communication - the AMIGO activity model*. Ellis Horwood Limited, Chichester, 1989.

[Pie91a] U. Piepenburg: *Rechnergestütztes kooperatives Arbeiten*. Mitteilung Nr. 197, Fachbereich Informatik, Universität Hamburg, 1991.

[Pie91b] U. Piepenburg: *Ein Konzept von Kooperation und seine Implikation für die technische Unterstützung kooperativer Prozesse*. In: H. Oberquelle (Hrsg.). Kooperative Arbeit und Computerunterstützung - Stand und Perspektiven. Verlag für Angewandte Psychologie, Göttingen, Stuttgart, 1991.

[RB91] T. Rodden, G. Blair: *CSCW and Distributed Systems: The Problem of Control*. In: L. Bannon, M. Robinson & K. Schmidt (Hrsg.). Proc. ECSCW '91, Amsterdam, 1991, 49-63.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen: *Object-Oriented Modeling and Design*. London: Prentice-Hall, 1991.

[Rei93] B. Reinwald: *Entwurf und Betrieb geregelter arbeitsteiliger Anwendungssysteme*. Teubner, Stuttgart, Leipzig, 1993.

[Rie93] D. Riehle: *Interaktionstypen Dokumentation zur IATMotif v1.1 Bibliothek*. Arbeitsbereich Softwaretechnik, Fachbereich Informatik, Universität Hamburg, 1993.

[Rie95] D. Riehle: *Muster am Beispiel der Werkzeug und Material Metapher*. Diplomarbeit, Fachbereich Informatik, Universität Hamburg, 1995.

[Rob93] M. Robinson: *Design for unanticipated use*. In: G. de Michelis, C. Simone, K. Schmidt (Hrsg.). Proc. ECSCW '93, Milan, 1993, 187-202.

[RZ94] D. Riehle, H. Züllighoven: *Späte Erzeugung*. 39. Internationales Wissenschaftliches Kolloquium. Technische Universität Ilmenau (Thüringen), 1994, 3-9.

[RZ95] D. Riehle, H. Züllighoven: *A Pattern Language for Tool Construction and Integration Based on the Tools & Materials Metaphor*. In: J.O. Coplien, D.C. Schmidt (Hrsg.). Pattern Languages of Programs. Addison-Wesley, Reading, Massachusetts, 1995.

[SB92] L. Bannon, K. Schmidt: *Taking CSCW Seriously: Supporting Articulation Work*. In: Computer Supported Cooperative Work (CSCW). An International Journal, 1(1), 1992, 1-31.

[SBF+87] M. Stefik, D.G. Bobrow, G. Foster, S. Lanning, D. Tatar: *WYSIWIS revised: early experiences with multiuser interfaces*. In: ACM Transactions on Office Information Systems, 5 (2), April 1987, 146-167.

[Sør88] P. Sørgaard: *Object Oriented Programming and Computerized Shared Material*. In : Proc. ECOOP '88. Oslo (Norwegen). Springer, Berlin, Heidelberg, New York, London, Paris, Tokyo, 1988, 319-334.

[Suc87] L. Suchman: *Plans and Situated Actions*. Cambridge University Press, 1987.

[Vol94] W. Volpert: *Wider die Maschinenmodelle des Handelns - Aufsätze zur Handlungsregulationstheorie*. Pabst Science Publishers, 1994.

[War92] H.-J. Warnecke: *Die Fraktale Fabrik: Revolution in der Unternehmenskultur*. Springer, Berlin, Heidelberg, New York, 1992.

[Wes95] D. Weske: *Zum Stand der Diskussion über verteilte, nach der Werkzeug-Material-Metapher entworfene Werkzeugumgebungen*. Unveröffentlichtes Papier zum Verteilungs-Workshop, Arbeitsbereich Softwaretechnik, Fachbereich Informatik, Universität Hamburg, Januar 1995.

[WW94] D. Weske, M. Wulf: *Konzepte zur Materialversorgung verteilter Werkzeugumgebung am Beispiel der Anbindung einer objektorientierten Datenbank*. Studienarbeit, Universität Hamburg, Fachbereich Informatik, März 1994.