Workflow-Unterstützung für den universitären Lehrplanungsprozeß: Möglichkeiten und Grenzen bei der Umsetzung mit „FlowMark“

Studienarbeit

Universität Hamburg
Fachbereich Informatik
Arbeitsbereich Softwaretechnik

Juni 1999

Autoren:
Timmy Blank und
Olaf Tesmer

Betreuung:
Dr. Ingrid Wetzel und
Wolf-Gideon Bleek

Timmy Blank
Auf der Haide 11
21039 Börnsen
Matr. Nr.: 4628041

Olaf Tesmer
Barmbeker Str. 163
22299 Hamburg
Matr. Nr.: 4346687

Inhaltsverzeichnis

1 Einleitung

Der organisatorische Aufbau der Unternehmen hat sich vornehmlich aus der Taylorschen Theorie zur systematischen Untersuchung von Arbeitsmethoden und der Trennung von Routinearbeiten und Planungs- und Kontrollarbeiten aus dem Jahre 1911 (Taylors Scientific Management [Taylor_11]) entwickelt. Es wurde versucht, die Arbeit (sowohl im Büro als auch in der Fabrik) in kleinste Einheiten zu zerlegen und diese dann von Angestellten und Arbeitern ausführen zu lassen. Die Nachteile dieses Ansatzes wurden vielfach diskutiert (vgl. [Schäl_96, S. 7]).

In [Keller_97] wird ausgeführt, daß die aus der Taylorschen Theorie resultierende hierarchische Unternehmensstruktur (auch Bürokratie genannt) nicht mehr der Dynamik der heutigen Weltmärkte gewachsen ist. Diese Organisationsform, die als funktionsorientiert bezeichnet wird, bedingt starre Informationsflüsse, führt zu langen Liegezeiten von internen als auch externen Aufträgen und es kommt zu Abstimmungsschwierigkeiten an den jeweiligen Schnittstellen. Der Grund hierfür ist in der Aufgabenteilung als solches zu sehen, da sie den Mitarbeitern den Blick für den gesamten Herstellungsprozeß verbaut. Das Ziel der Mitarbeiter ist es, ihr Abteilungsergebnis zu verbessern. Der Controllingexperte will z.B. die Kosten minimieren, der Entwicklungsleiter hat seinen Blickwinkel auf der Technologieoptimierung und der Vertriebsmitarbeiter auf der Umsatzmaximierung. Diese unterschiedlichen Zielsetzungen können dazu führen, daß die Mitarbeiter kontraproduktiv arbeiten.

Dieses lange Zeit vorherrschende bürokratische Organisationsprinzip ging von relativ konstanten Umweltbedingungen aus. Unter Umweltbedingungen versteht man in diesem Zusammenhang die technische, ökologische, soziale als auch die politische Umgebung eines Unternehmens. Diese Bereiche unterliegen heutzutage einem raschen und selten prognostizierbaren Wandel. Somit wird von heutigen Unternehmen eine rasche Anpaßbarkeit gefordert, welche nicht mehr mit der alten Organisationsstruktur in Einklang gebracht werden kann. Produkte und Dienstleistungen müssen hochwertig sein und individuell auf den Kunden zugeschnitten werden, um sich von der Konkurrenz abheben zu können. Dies erfordert eine hohe Flexibilität und kurze Produktzyklen. Somit mußte nach neuen Organisationsmöglichkeiten gesucht werden.

Der gewählte Ansatz wird als Prozeß Management bezeichnet. Prozeß Management wird als der Weg verstanden, „ ... die traditionelle bürokratische Organisation in eine marktorientierte Prozessorganisation umzuwandeln.“ [Schäl_96, S. 9]. Das Unternehmen wird nicht mehr nach der Idee aufgebaut, daß der industrielle Arbeitsprozeß in kleinste Teile zerlegt werden muß. Die Idee ist nun vielmehr, zusammengehörige Teile in sogenannten Geschäftsprozessen zusammenzufassen. Unter einem Geschäftsprozeß versteht man laut [Jablonski_97a, Glossar] einen „ ... Vorgang in Wirtschaftseinheiten (Unternehmen, Verwaltungen, etc.), der einen wesentlichen Beitrag zu einem nicht notwendigerweise ökonomischen Unternehmenserfolg leistet. Dabei läuft er in der Regel funktions-, hierarchie- und standortübergreifend ab, kann die Unternehmensgrenzen überschreiten und erzeugt einen meßbaren, direkten Kundennutzen.“ Wichtig ist, daß Geschäftsprozesse in Unternehmen abteilungsübergreifend ablaufen. Somit ist der Ansatz der strikten Aufgabentrennung nicht mehr haltbar. Man orientiert sich am Gesamtprozeß und versucht diesen zu optimieren.

Es wurden eine Vielzahl von Ansätzen zur Gestaltung von geschäftsprozeßorientierten Unternehmensstrukturen entwickelt. Zu nennen sind u.a. Business (Re-)Engineering, Vernetztes Denken, Lean Management und Workflow-Management. Einen Überblick findet

man in [Keller_97, S. 32 ff.]. Resultate dieser Umgestaltung sind eine Abflachung der Hierarchien, eine größere Fokussierung auf Gruppenarbeit, Erhöhung der Kompetenzen der einzelnen Mitarbeiter, Bildung von virtuellen Unternehmen, usw.

Aufgrund der Tatsache, daß Computertechnologie in heutigen Unternehmen weit verbreitet ist, liegt der Ansatz nahe, die vorhandene IT-Infrastruktur (sowohl Hard- als auch Software) zu nutzen, um die identifizierten Geschäftsprozesse zu optimieren. Außerdem muß beachtet werden, daß Geschäftsprozesse nicht nur die Prozesse selbst, sondern auch Personen (oder Maschinen), die diese Prozesse ausführen und in den Prozessen zu verarbeitende Daten beschreiben. Neu und flexibel strukturierte Geschäftsprozesse verlangen nach entsprechend flexiblen Ausführungssystemen, die ihre Ausführung unterstützen und dabei auch die Aufbauorganisation des Unternehmens berücksichtigen. Unter dem Begriff der Aufbauorganisation versteht man laut [Brockhaus_93b] den „Beziehungszusammenhang von Stellen und Abteilungen eines Unternehmens, einer Behörde oder Verwaltung einschließlich der Regelung der Weisungsbefugnisse (Leitungsstruktur) und der Informationswege (Kommunikationsstruktur).“ Man verspricht sich von sogenannten Workflow-Management-Systemen (WfMS), daß sie die gestellten Anforderungen erfüllen können (vgl. [Jablonski_94, S. 14]).

Zur Zeit sind eine Vielzahl kommerzieller WfMS am Markt erhältlich (vgl. z.B. [Jablonski_97a], [Alonso_96], [Weiß_96]). Laut [Weiß_96] sind diese Produkte in zwei Gruppen aufzuteilen. Es handelt sich zum einen um Programme, die aus anderen Programmen wie Dokumenten-Management-Systemen, E-mail-Systemen, Textverarbeitungs-Systemen, Groupware-Produkten, etc. durch Erweiterung um die erforderlichen Funktionalitäten entstanden sind. Zum anderen gibt es dann noch die sogenannten originären WfMS, die vollständig neu entwickelt wurden.

Bei WfMS handelt es sich um aktive, also ablaufsteuernde Softwaresysteme, wobei der Ablauf durch ein sogenanntes Workflow-Schema (oder Workflow-Modell) festgelegt wird. Ein WfMS hat also zwei Aufgaben: Es muß zum einen die Möglichkeit bieten, Ablaufspezifikationen zu formulieren und zum anderen für die korrekte Ausführung dieser Spezifikationen sorgen.

Um ein Modell der Vorgänge einer Anwendungssituation erstellen zu können, welches als Grundlage zur Ablaufsteuerung verwendet werden soll, bedarf es detaillierter Kenntnisse der Struktur dieser Vorgänge. Arbeitsvorgänge in der Realität (die ja Bestandteil von Geschäftsprozessen sind, die von WfMS ausgeführt werden sollen) zeichnen sich allerdings auch durch unstrukturierte Teile aus, die ebenfalls in den Anwendungen berücksichtigt werden müssen. Inwieweit dies von dem jeweiligen WfMS unterstützt wird (und ob eine Unterstützung überhaupt möglich ist), hängt von den individuellen Fähigkeiten des Systems ab. Laut [Jablonski_97a, S. 6] werden die Fähigkeiten eines WfMS bzw. einer Workflow-Management-Anwendung hinsichtlich der Unterstützung von Arbeitsvorgängen nachhaltig dadurch bestimmt, welche Möglichkeiten zur Darstellung der Geschäftsprozesse angeboten werden. Diese Möglichkeiten werden wiederum durch die verwendete Workflow-Sprache des jeweiligen Produktes bestimmt, wodurch die Sprache eines WfMS als Beurteilungsgrundlage der Leistungsfähigkeit verwendet werden kann.

1.1 Ziele der Arbeit

In dieser Arbeit wird das WfMS FlowMark (Version 2.3) von IBM untersucht. Es ist ein Produkt, welches der Gruppe der originären WfMS angehört. Laut [IBM_98] handelt es sich

um ein WfMS, welches das Business (Re-)Engeneering in vielen Bereichen der Industrie, in denen die Automation von Arbeitsabläufen erforderlich ist, unterstützt.

Wir wollen in dieser Arbeit untersuchen, um welche Arbeitssituationen es sich hier genau handelt. Hierzu haben wir unserer Arbeit ein reales Anwendungsbeispiel, namentlich der Lehrplanungsprozeß der Universität Hamburg (aus Sicht des Arbeitsbereichs Softwaretechnik des Fachbereichs Informatik), zugrundegelegt, welches sich durch das Vorkommen verschiedener Arten von Arbeitsabläufen auszeichnet. Unser Ziel ist es herauszustellen, welche Arbeitsabläufe dies sind und welche hiervon durch FlowMark unterstützt werden können.
Hierzu ist es einerseits notwendig, Möglichkeiten zur Charakterisierung kooperativer Arbeitssituationen zu finden, um diese dann auf das Anwendungsbeispiel anzuwenden.
Andererseits sind die FlowMark zugrundeliegenden Konzepte und System-Komponenten zu untersuchen. Mit dem Wissen über die zur Verfügung stehenden Mittel wollen wir die Umsetzungsmöglichkeiten des Anwendungsbeispiels mit FlowMark eingrenzen, damit die Eignung des Systems zur Unterstützung bestimmter Kooperationsformen eingeschätzt werden kann.

1.2 Gliederung der Arbeit

Die Arbeit gliedert sich wie folgt:
In Kapitel 2 wird zunächst das Anwendungsbeispiel anhand von Kooperationsbildern und Wozu-Tabellen dargestellt. Kapitel 3 charakterisiert das Anwendungsbeispiel anhand von Kooperationsmerkmalen und diskutiert die Unterstützungsmöglichkeiten durch CSCW-Applikationen. Dazu wird nach einer Einführung in die Konzepte von WfMS eine Positionierung des Anwendungsbeispiels in ein Workflow Kontinuum vorgenommen.

Kapitel 4 setzt sich mit dem System FlowMark auseinander. Im Mittelpunkt stehen die von FlowMark angebotenen Modellierungsmöglichkeiten und die Architektur des Systems. Hierzu werden zunächst einige allgemeine an die Architektur von WfMS gestellte Anforderungen genannt und dann die Architektur von FlowMark hinsichtlich deren Erfüllung untersucht.

Kapitel 5 beschreibt die wichtigsten Aspekte der Umsetzung des Anwendungsbeispiels. Außerdem werden die Möglichkeiten und Grenzen FlowMarks zur Unterstützung der Kooperationsformen im Anwendungsbeispiel herausgearbeitet.

Kapitel 6 faßt die Ergebnisse, die im Verlauf dieser Arbeit gewonnen werden konnten, zusammen und gibt einen allgemeinen Ausblick.

2 Das Anwendungsbeispiel

Bei dem dieser Arbeit zugrunde liegendem Anwendungsbeispiel handelt es sich um den Lehrplanungsprozeß der Universität Hamburg. Die Struktur dieses Prozesses (beteiligte Funktionelle Rollen, Arbeitsschritte und verwendetet Dokumente) wurde durch Führung mehrerer Interviews mit der AB-Lehrplanungsbeauftragten (AB steht für Arbeitsbereich) des Arbeitsbereichs Softwaretechnik der Universität Hamburg, Dr. Ingrid Wetzel, identifiziert. Da uns keine weiteren Informationen über den Ablauf der Lehrplanungsprozesse in anderen Arbeits- und Fachbereichen zur Verfügung standen, beschreiben wird den Lehrplanungsprozeß aus Sicht des Arbeitsbereiches Softwaretechnik des Fachbereichs Informatik. Wir gehen allerdings von der Annahme aus, daß sich die Vorgänge bezüglich der Lehrplanung in den einzelnen Arbeitsbereichen des Fachbereichs Informatik gleichen.

Grundsätzlich teilt sich der Lehrplanungsprozeß in zwei voneinander zeitlich getrennt ablaufende Abschnitte ein. Der erste Abschnitt wird im folgenden als „Lehrplanungsprozeß für das übernächste Semester“ bezeichnet und der zweite Abschnitt als „Lehrplanungsprozeß für das nächste Semester“. Die beiden Abschnitte unterscheiden sich darin, daß das Ergebnis des Lehrplanungsprozesses für das übernächste Semester ein vorläufiges Lehrangebot des jeweiligen Fachbereiches für das übernächste Semester darstellt und das Ergebnis des Lehrplanungsprozesses für das nächste Semester das endgültige, eventuell überarbeitete, Lehrangebot. Eine Überarbeitung kann dann von Nöten sein, wenn z.B. aktuelle Forschungsrichtungen berücksichtigt oder neue Mitarbeiter in die Lehre miteinbezogen werden sollen. Außerdem kann es auch zu Überschneidungen hinsichtlich der Lehrangebote verschiedener Arbeitsbereiche kommen, welches eine Anpassung erforderlich macht.

Im folgenden wird die Ist-Situation des Lehrplanungsprozesses beschrieben. Dies geschieht unter Verwendung von Kooperationsbildern mit deren Hilfe die beteiligten funktionellen Rollen (als Kästen dargestellt) und deren Kooperationsbeziehungen untereinander durch Darstellung der von ihnen ausgetauschten Dokumente verdeutlicht wird. Direkte Kommunikation der Beteiligten (z.B. Sitzungen), wird durch Sterne dargestellt. Dies ist eine Erweiterung der von [Krabbel_96] entwickelten Kooperationsbilder, die nachfolgend definiert werden. Es ist zu beachten, daß es sich bei der Darstellung um die Beschreibung des idealen Ablaufs handelt. Ausnahmensituationen werden nicht betrachtet.

„Ein Kooperationsbild (s. [Krabbel et al_96]) ist eine visuelle Darstellung einer kooperativen Arbeitssituation, besonders bei übergreifenden Aufgaben. Dabei werden allgemeinverständliche Piktogramme verwendet. Kooperationsbilder stellen dar, wie Personen arbeitsteilig zusammenarbeiten.“ [Züllighoven_98, S. 645] Um die Sequentialität bzw. Parallelität der Arbeitsschritte zu kennzeichnen, wird in den hier verwendeten Kooperationsbildern auf Sequenznummern zurückgegriffen. Aufeinanderfolgende Nummern beschreiben eine sequentielle Abfolge, gleiche Nummern gefolgt von unterschiedlichen Buchstaben deuten eine mögliche parallele Abfolge an.

Zur detaillierteren Darstellung der Aufgaben, der in diesem Kontext beteiligten funktionellen Rollen, werden Wozu-Tabellen eingesetzt. „Eine Wozu-Tabelle [Krabbel et al_96] ist eine Tabelle, in der für eine ausgewählte Kooperationsform festgehalten wird, wer was mit wem oder womit macht und - namengebend für diese Art von Tabellen - wozu das ganze gut ist. Die Frage ‘wozu’ zielt explizit auf den Zweck einer Tätigkeit oder den kooperativen Verwendungszweck eines Gegenstandes. ...“ [Züllighoven_98, S. 653].

2.1 Der Lehrplanungsprozeß für das übernächste Semester

Der Lehrplanungsprozeß für das übernächste Semester wird von den FB-Lehrplanern (FB steht für Fachbereich) angestoßen. Diese fordern von den einzelnen AB-Lehrplanungsbeauftragten des jeweiligen Fachbereiches die sogenannte mittelfristige Lehrplanung an (s. Abb. 1, 1a). In dem hierbei verwendeten Dokument wird für die folgenden fünf Jahre festgehalten, wann welcher Lehrende welche Pflichtveranstaltung halten wird. Es handelt sich hierbei um Grundstudiums- und Kerngebietsvorlesungen. Des weiteren wird angegeben, wann Forschungsfreisemester geplant sind. Außerdem finden sich Bereitschaftserklärungen hinsichtlich der Übernahme von Vorlesungszyklen durch Lehrende.

In jedem Arbeitsbereich hat der dort tätige Sekretär die Aufgabe, dem AB-Lehrplanungsbeauftragten die Liste der betreuten Studien- und Diplomarbeiten (s. Abb. 1, 1b) zukommen zu lassen. Hier wird festgehalten, welcher Mitarbeiter des AB welche Studien- und Diplomarbeiten betreut. Der Sekretär ist dafür zuständig, diese auf dem aktuellsten Stand zu halten.

Diese beiden aufgeführten Dokumente enthalten wichtige Informationen. Aus ihnen geht hervor, wieviel Lehre von den jeweiligen Mitarbeitern des Arbeitsbereiches noch angeboten werden muß, damit diese ihre Lehrdeputate erfüllen können. Die Lehrdeputate legen fest, wieviel Stunden Lehre jeder einzelne im Semester abzuleisten hat. Hierzu zählen allerdings nicht nur die angebotenen Lehrveranstaltungen sondern auch die betreuten Studien- und Diplomarbeiten. Außerdem besteht die Möglichkeit, daß im vorherigen Semester erbrachte Mehrleistungen bis zu einem Anteil von fünfzig Prozent der aktuellen Leistung verrechnet werden.

Im nächsten Schritt findet in jedem Arbeitsbereich eine Lehrplanungssitzung statt (s. Abb.1, 2). Teilnehmer sind der AB-Lehrplanungsbeauftragte, der AB-Leiter und die Veranstalter des Arbeitsbereichs. In dieser Sitzung stimmen die Teilnehmer, unter Berücksichtigung der Lehrdeputate, das Lehrangebot für das übernächste Semester ab. Falls mehrere Veranstalter eine Veranstaltung gemeinsam anbieten wollen, ist abzuklären, wer diese für sein Lehrdeputat angerechnet bekommt.

Das in der AB-Lehrplanungssitzung ausgehandelte Lehrangebot des Arbeitsbereiches wird dann zusammen mit den Lehrdeputaten vom AB-Lehrplanungsbeauftragten an den FB-Lehrplaner weitergeleitet (s. Abb. 1, 3a), der eine Überprüfung der Angebote hinsichtlich der sogenannten Kapazitätsauslastung vornimmt. Hierzu gleicht er die angebotenen Veranstaltungen mit der momentanen Studentenzahl und der daraus resultierenden Anzahl an notwendigen Veranstaltungen ab, damit den Studierenden ein Studium nach ihrem Studienplan ermöglicht wird. Als Berechnungsgrundlage verwendet der FB-Lehrplaner allerdings nicht nur die Studentenzahl, sondern auch Erfahrungswerte hinsichtlich der Gewohnheiten und Vorlieben der Studierenden, die er im Laufe der Jahre aufgebaut hat.

Der FB-Lehrplaner überprüft auch, ob Lehrberechtigungsanträge für geplante Veranstaltungen vorliegen. Diese Anträge sind erforderlich, wenn ein Mitarbeiter eines Arbeitsbereiches eigenverantwortlich eine Lehrveranstaltung anbieten möchte, allerdings noch keine Lehrberechtigung hat. In diesem Dokument muß aufgeführt sein, warum der Antragsteller diese Veranstaltung anbieten möchte, welchen Inhalt sie haben soll und warum er sich für fähig dafür hält. Der AB-Leiter muß den Lehrberechtigungsantrag durch Unterschrift bestätigen (s. Abb. 1, 3b-2). Dieser bestätigte Lehrberechtigungsantrag wird dann von dem betroffenen Veranstalter an den FB-Lehrplaner gesandt (s. Abb.1, 3b-3a). Eine Kopie geht an den AB-Lehrplanungsbeauftragten (s. Abb.1, 3b-3b). Liegt der Lehrberechtigungsantrag vor,

so leitet der FB-Lehrplaner ihn an den FB-Prüfungsamtsvorsitzenden weiter. Liegt er nicht vor, so fordert er ihn über den zuständigen AB-Lehrplanungsbeauftragten an, der diese Anforderung an den betroffenen Antragsteller weiterleitet.

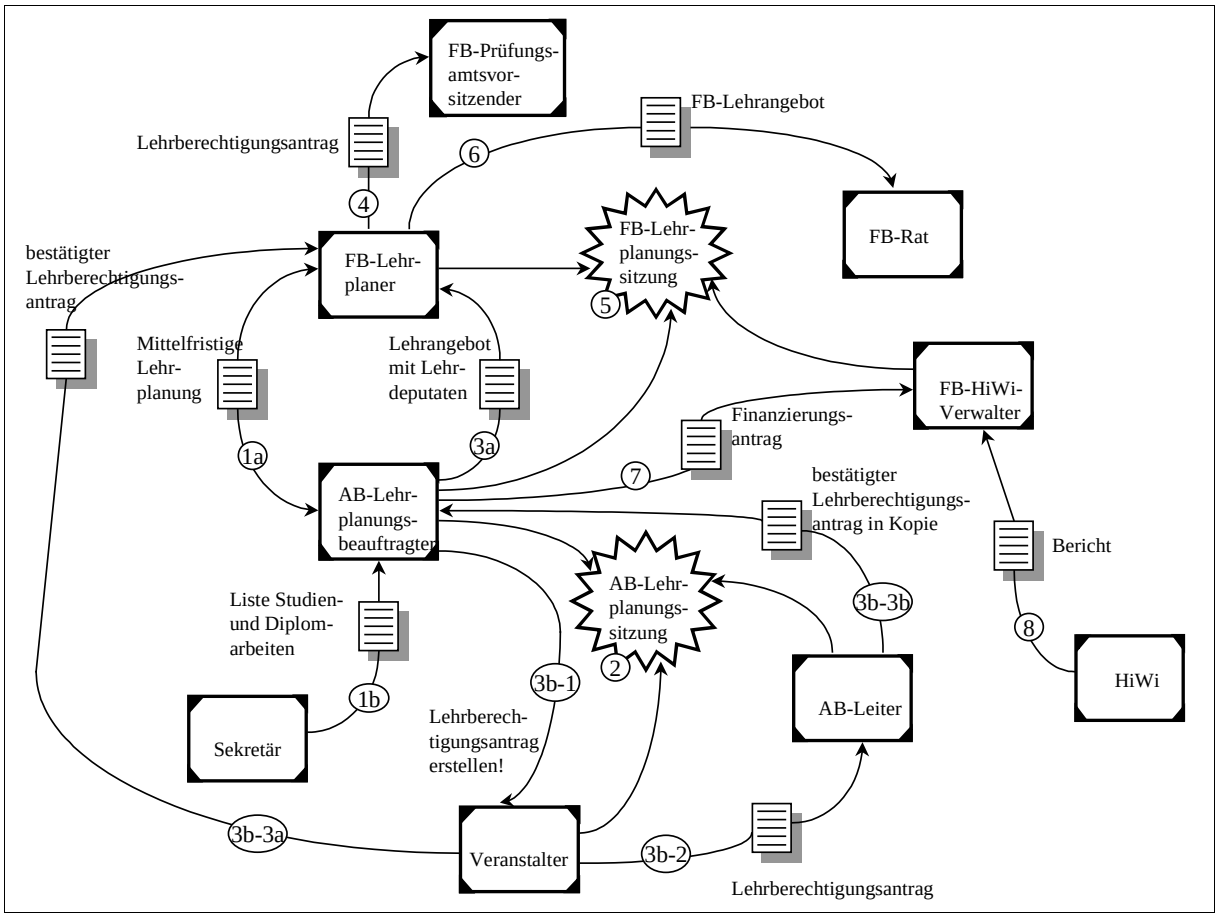


Abbildung 1: Kooperationsbild des Lehrplanungsprozesses für das übernächste Semester

Nach der Überprüfung der Lehrangebote durch den FB-Lehrplaner findet eine FB-Lehrplanungssitzung statt (s. Abb. 1, 5). Teilnehmer sind die AB-Lehrplanungsbeauftragten, der FB-Lehrplaner, der FB-Raum-/Zeitplaner und der FB-HiWi-Verwalter (HiWi steht für Hilfswissenschaftler). In dieser Sitzung wird eine letzte Überprüfung des gesamten Lehrangebotes des Fachbereiches hinsichtlich Vollständigkeit und eventueller inhaltlicher Überschneidungen vorgenommen. Entspricht das Angebot den Vorstellungen der Beteiligten, so wird das FB-Lehrangebot beschlossen und im nächsten Schritt vom FB-Lehrplaner an den FB-Rat zur Verabschiedung weitergeleitet (s. Abb. 1, 6).

Da das Lehrangebot nun feststeht, kann der AB-Lehrplanungsbeauftragte bestimmen, wieviele Übungsgruppen im übernächsten Semester angeboten werden müssen und daraus folgend, wieviele HiWis zur Betreuung der Übungsgruppen benötigt werden. Damit diese entsprechend entlohnt werden können muß ein Finanzierungsantrag an den FB-HiWi-Verwalter gestellt werden (s. Abb. 1, 7).

Die HiWis müssen nach Abschluß ihrer Lehrtätigkeit einen Bericht über ihre erbrachten Leistungen anfertigen. Dieser geht dann an den FB-HiWi-Verwalter (s. Abb. 1, 8). Dieser Schritt steht nicht im direkten (zeitlichen) Zusammenhang mit der Lehrplanung für das übernächste Semester, findet aber der Vollständigkeit halber hier Berücksichtigung.

Zur Zusammenfassung der Aufgaben der an dem Lehrplanungsprozeß für das übernächste Semester beteiligten Funktionellen Rollen ist nachfolgend eine Wozu-Tabelle aufgeführt.

	Wer	Erledigt was	Mit wem/was	Wozu
1a	AB-Lehrplanungsbeauftragter	Erstellt mittelfristige Lehrplanung und sendet diese an den FB-Lehrplaner	Veranstaltern, die Pflichtveranstaltungen übernehmen	Um Informationen zu Lehrdeputaten zu erhalten und dem FB-Lehrplaner Gewißheit über die Betreuung der Pflichtveranstaltungen zu geben
1b	Sekretär	Pflegt Liste der Studien- und Diplomarbeiten und übergiebt sie dem AB-Lehrplanungsbeauftragten	Liste der Studien- und Diplomarbeiten	Um dem AB-Lehrplanungsbeauftragten Informationen zu Lehrdeputaten zur Verfügung zu stellen
2	AB-Lehrplanungsbeauftragter	Ruft AB-Lehrplanungssitzung ein	AB-Leiter, Veranstalter	Um den Lehrvorschlag für das übernächste Semester auszuhandeln
3a	AB-Lehrplanungsbeauftragter	Übergibt AB-Lehrangebot mit Lehrdeputaten an FB-Lehrplaner	AB-Lehrangebot mit Lehrdeputaten	Damit der FB-Lehrplaner die AB-Lehrangebote hinsichtlich der Kapazitätsauslastung überprüfen und das FB-Lehrangebot erstellen kann. Außerdem Überprüfung, ob erforderliche Lehrberechtigungsanträge vorliegen.
3b-2	Veranstalter	Erstellt Lehrberechtigungsantrag und sendet ihn an den AB-Leiter	Lehrberechtigungsantrag	Um eigenverantwortlich eine Lehrveranstaltung anbieten zu können, wobei der Antrag vom AB-Leiter unterschrieben werden muß.
3b-3a	Veranstalter	Sendet bestätigten Lehrberechtigungsantrag an FB-Lehrplaner	Bestätigter Lehrberechtigungsantrag	Damit der FB-Lehrplaner überprüfen kann, ob ein geforderter Lehrberechtigungsantrag vorliegt und damit er diesen an den FB-Prüfungsamtsvorsitzenden weiterleiten kann
4	FB-Lehrplaner	Sendet Lehrberechtigungsanträge an den FB-Prüfungsamtsvorsitzenden	Lehrberechtigungsanträge	Zur Überprüfung
5	FB-Lehrplaner	Ruft FB-Lehrplanungssitzung ein	AB-Lehrplanungsbeauftragte der ABs des FB, FB-HiWi-Verwalter	Um das FB-Lehrangebot zu beschließen und eventuelle letzte Änderungen vorzunehmen.
6	FB-Lehrplaner	Sendet FB-Lehrangebot an den FB-Rat	FB-Lehrangebot	Damit der FB-Rat das FB-Lehrangebot verabschieden kann.
7	AB-Lehrplanungsbeauftragter	Erstellt Finanzierungsantrag und sendet ihn an den FB-HiWi-Verwalter	Finanzierungs-Antrag	Damit die Finanzierung der HiWis gesichert ist.
8	HiWi	Erstellt Bericht und sendet ihn an den FB-HiWi-Verwalter	Bericht	Zu Dokumentationszwecken.

Tabelle 1: Wozu-Tabelle der Lehrplanung für das übernächste Semester

2.2 Der Lehrplanungsprozeß für das nächste Semester

Ausgangspunkt der Lehrplanung für das nächste Semester ist die Erstellung eines vorläufigen Campus Raum-/Zeitplans durch den FB-Raum-/Zeitplaner. Dieser tritt mit den Lehrenden, die Veranstaltungen am Campus anbieten, in Kontakt, um zu erfahren, wann und wo diese gerne ihre Veranstaltungen am Campus stattfinden lassen würden (s. Abb. 2, 1). Der vorläufige Campus Raum-/Zeitplan wird dann an den Campus Raum-/Zeitplaner weitergeleitet, damit dieser die Wünsche in seiner Planung berücksichtigen kann (s. Abb. 2, 2).

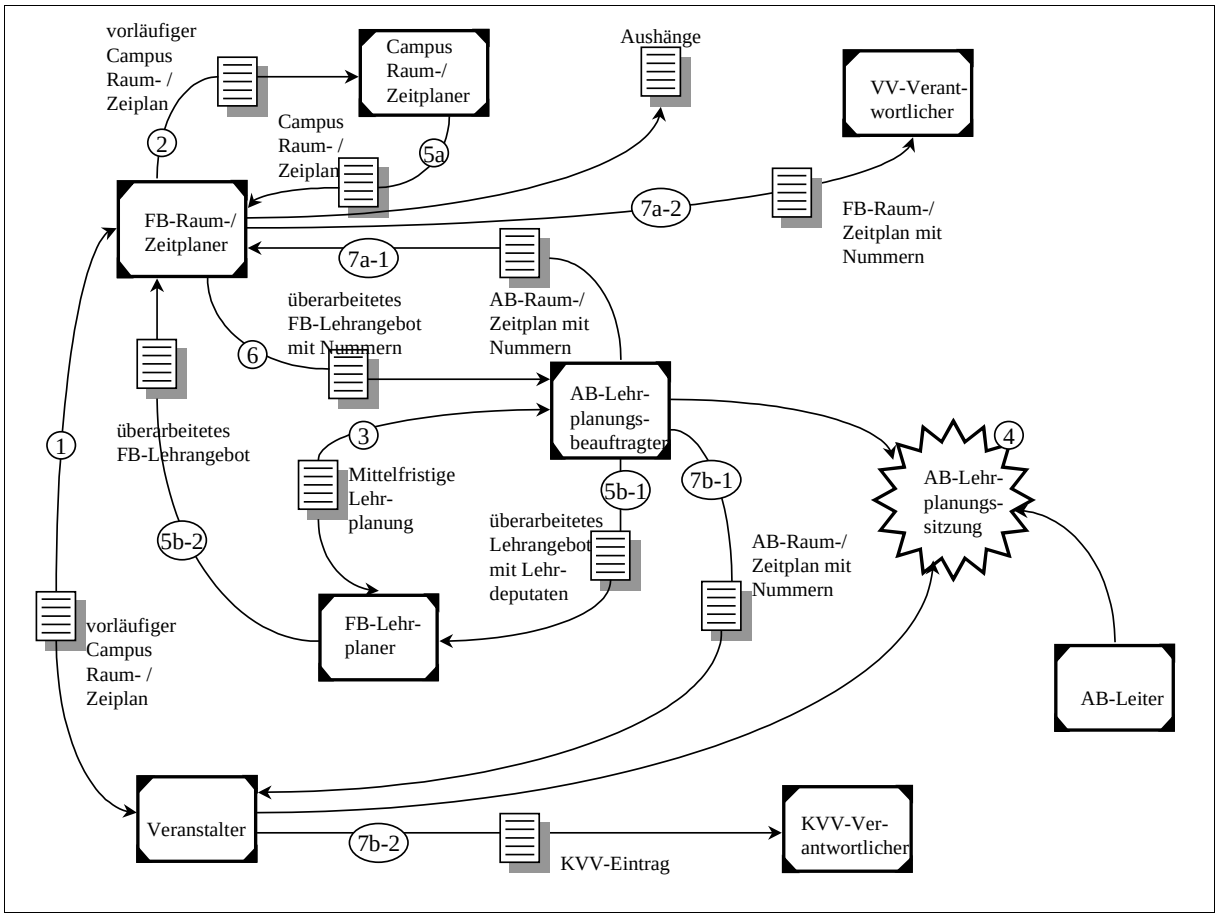


Abbildung 2: Kooperationsbild des Lehrplanungsprozesses für das nächste Semester

Wie schon beim Lehrplanungsprozeß für das übernächste Semester wird auch in diesem Abschnitt des Lehrplanungsprozesses vom FB-Lehrplaner die mittelfristige Lehrplanung angefordert (s. Abb. 2, 3).

In jedem Arbeitsbereich findet wiederum eine AB-Lehrplanungssitzung statt (s. Abb. 2, 4). Teilnehmer sind der AB-Lehrplanungsbeauftragte, der AB-Leiter und die Veranstalter. Inhaltlich geht es um eine eventuelle Anpassung des Lehrangebotes für das übernächste Semester hinsichtlich der Berücksichtigung aktueller Forschungsthemen oder neuer Mitarbeiter. Dieses überarbeitete Lehrangebot mit Lehrdeputaten wird dann an den FB-Lehrplaner gesendet (s. Abb. 2, 5b-1). Dieser sammelt alle überarbeiteten Lehrangebote und stellt das gesamte Lehrangebot des FB zusammen. Danach leitet er es an den FB-Raum-/ Zeitplaner weiter (s. Abb. 2, 5b-2).

Der FB-Raum-/Zeitplaner ordnet den Veranstaltungen daraufhin Nummern zu und sendet das überarbeitete FB-Lehrangebot mit Nummern zusammen mit dem Campus Raum-/Zeitplan, welchen er im Vorwege vom Campus Raum-/Zeitplaner erhalten hat (s. Abb. 2, 5a), an die AB-Lehrplanungsbeauftragten zurück (s. Abb. 2, 6). Diese erstellen dann jeweils für ihren Arbeitsbereich den AB-Raum-/Zeitplan. Hierzu müssen sie sich mit den Veranstaltern abstimmen, um deren Wünsche berücksichtigen zu können. Des weiteren ist auch eine Abstimmung mit AB-Lehrplanungsbeauftragten anderer Arbeitsbereiche erforderlich, wenn diese dieselben Veranstaltungsräume nutzen. Nach Fertigstellung gehen die AB-Raum-/ Zeitpläne wieder zurück an den FB-Raum-/Zeitplaner und an die Veranstalter der jeweiligen Arbeitsbereiche (s. Abb. 2, 7a-1 und 7b-1).

Der FB-Raum-/Zeitplaner stellt dann den gesamten FB-Raum-/Zeitplan zusammen und sendet ihn an den Uni-VV-Koordinator (VV steht für Vorlesungsverzeichnis), damit dieser die benötigten Informationen zur Erstellung des Uni-VV erhält (s. Abb. 2, 7a-2). Außerdem extrahiert der FB-Raum-/Zeitplaner aus dem FB-Raum-/Zeitplan Aushänge für die einzelnen Veranstaltungsräume, aus denen hervorgeht, wann welche Veranstaltung in dem jeweiligen Raum stattfindet.

Die Veranstalter können nach Erhalt des AB-Raum-/Zeitplans mit Nummern ihren KVV-Eintrag (KVV steht für Kommentiertes Vorlesungsverzeichnis) erstellen. Falls mehrere Veranstalter eine Veranstaltung betreuen, erfordert dies eine Abstimmung darüber, wer von ihnen den KVV-Eintrag erstellt. Der KVV-Eintrag wird dann an den KVV-Verantwortlichen gesandt, der die Druckvorlage zur Erstellung des KVV erzeugt (s. Abb. 2, 7b-2).

Zur Zusammenfassung der Aufgaben der an dem Lehrplanungsprozeß für das nächste Semester beteiligten Funktionellen Rollen ist nachfolgend eine Wozu-Tabelle aufgeführt.

	Wer	Erledigt was	Mit wem / was	Wozu
1	FB-Raum-/ Zeitplaner	Handelt vorläufigen Campus Raum-/Zeitplan aus	Veranstalter, die Veranstaltungen am Campus haben	Um einen vorläufigen Campus Raum-/Zeitplan zu erstellen.
2	FB-Raum-/ Zeitplaner	Sendet vorläufigen Campus Raum-/Zeit-plan an den Campus Raum-/Zeitplaner	Vorläufiger Campus Raum- / Zeitplan	Damit der Campus Raum-/Zeit-planer die Wünsche der Veranstalter bei der Erstellung der Campus Raum-/Zeitplans berücksichtigen kann.
3	AB-Lehrpla-nungsbeauf-tragter	Erstellt mittelfristige Lehrplanung und sendet diese an den FB-Lehr-planer	Veranstaltern, die Pflichtveranstal-tungen übernehmen	Um Informationen zu Lehrdeputaten zu erhalten und dem FB-Lehrplaner Gewißheit über die Betreuung der Pflichtveranstaltungen zu geben
4	AB-Lehrpla-nungsbeauf-tragter	Ruft AB-Lehrplanungs-sitzung ein	AB-Leiter, Veranstalter	Um den Lehrvorschlag für das übernächste Semester zu überarbeiten.
5a	Campus Raum-/ Zeitplaner	Sendet Campus Raum-/ Zeitplan an die FB-Raum-/ Zeitplaner	Campus Raum-/ Zeitplan	Damit diesen Veranstaltungen Veranstaltungsnummern zugeordnet werden können und der FB-Lehr-plan vervollständigt werden kann.
5b-1	AB-Lehrpla-nungsbeauf-tragter	Sendet überarbeitetes AB-Lehrangebot an den FB-Lehrplaner	Überarbeitetes AB-Lehrangebot	Damit der FB-Lehrplaner das überarbeitete Lehrangebot des FB zusammenstellen kann
5b-2	FB-Lehr-planer	Sendet überarbeitetes FB-Lehrangebot an den FB-Raum-/Zeitplaner	Überarbeitetes FB-Lehrangebot	Damit FB-Raum-/Zeitplaner den Veranstaltungen des FB Veran-staltungsnummern zuordnen kann.
6	FB-Raum-/ Zeitplaner	Sendet das überarbeitete FB-Lehrangebot mit	Überarbeitete FB-Lehrangebot mit	Damit die AB-Lehrplanungs-Beauftragten die AB-Raum-/

		Nummern an die AB-Lehrplanungsbeauf-tragten	Nummern	Zeitpläne erstellen können
7a-1	AB-Lehrpla-nungsbeauf-tragter	Sendet AB-Raum-/ Zeitplan an den FB-Raum-/Zeitplaner	AB-Raum-/Zeitplan	Damit der AB-Raum-/Zeitplaner den FB-Raum-/Zeitplan erstellen kann
7a-2	FB-Raum-/ Zeitplaner	Sendet FB-Raum-/ Zeit-plan mit Nummern an den VV-Verantwortlichen	FB-Raum-/ Zeitplan mit Nummern	Damit der VV-Verantwortliche das VV erstellen kann.
7b-1	AB-Lehrpla-nungsbeauf-tragter	Sendet AB-Raum-/Zeit-plan an die Veranstalter	AB-Raum-/Zeitplan	Damit die Veranstalter die KVV-Einträge erstellen können.
7b-2	Veranstalter	Sendet KVV-Eintrag an den KVV-Verant-wortlichen	KVV-Eintrag	Damit der KVV-Verantwortliche die Vorlage für den Druck des KVV erstellen kann.

Tabelle 2 Wozu-Tabelle der Lehrplanung für das nächste Semester

3 Charakterisierung und Positionierung des Anwendungsbeispiels

3.1 Kooperative Arbeit

In den letzten Jahren finden in Unternehmen strategische Umbrüche der Organisationsstrukturen statt. Die Entwicklung geht dabei weg von hoch abeitsteiligen und hierarchischen, hin zu prozeßorientierten und integrierten Geschäftsprozessen (vgl. Kapitel 1).

Es entsteht eine neue Organisationsform mit flexiblen technischen und sozialen Netzwerken, interdisziplinären, autonomen Projektgruppen und Informations- und Berichtswegen jenseits des originären Organigramms. In immer stärkerem Maße führt dieses Konzept zu gegenseitiger Beeinflussung und Abhängigkeit aller an den Geschäftsprozessen Beteiligten im Sinne einer dynamischen betrieblichen *Zusammenarbeit* [Riggert_98].

Seit relativ leistungsfähige Arbeitsplatzrechner in Labors und Büros in lokalen Netzen miteinander verbunden wurden und überregionale Rechnernetze genutzt werden können, bestehen vielfältige neue technische Möglichkeiten, um die Zusammenarbeit zwischen Personen und Gruppen zu unterstützen. In der Informatik hat sich daraus unter dem Schlagwort *computergestützte kooperative Arbeit* ("computer-supported cooperative work", CSCW) ein neues Anwendungsfeld entwickelt.

Da es nicht sinnvoll ist, über Computerunterstützung nachzudenken, solange nicht geklärt ist, was genau unter *kooperativer Arbeit* zu verstehen ist, stellen wir in diesem Kapitel dazu zunächst Klärungsversuche aus der Literatur vor (3.1.1). Danach stellen wir *Merkmale kooperativer Systeme* heraus und benennen deren mögliche Ausprägungen (3.1.2). Schließlich charakterisieren wir das Anwendungsbeispiel hinsichtlich der vorgestellten Merkmale (3.1.3).

3.1.1 Begriffsklärung

Kooperation wird häufig sehr allgemein als das planvolle Zusammenwirken einer Mehrzahl von Personen im selben Produktionsprozeß oder in verschiedenen, aber verbundenen Produktionsprozessen verstanden, manchmal auch nur als die aufgabenbezogene Kommunikation von Teams oder Gruppen [Oberquelle_91].

[Bannon et al_89] bemängeln das Fehlen von allgemein akzeptierten Definitionen *kooperativer Arbeit* und fordern:
"The focus is to *understand*, so as to better *support,* cooperative work."

Weiterhin befassen sie sich ausführlich mit dem Begriff Cooperative Work und grenzen ihn gegenüber verwandten Ausdrücken wie *Collaborative* oder *Collective Work* ab. Sie plädieren für eine offene Definition von Cooperative Work:

"In sum, the term cooperative work is the general and neutral designation of multiple persons working together to produce a product or service. It does not imply specific forms of interaction or organization such as comradely feelings, equality of status, formation of a distinct group identity etc." [Bannon et al_89]

In Verbindung mit Kooperation fallen häufig die Begriffe *Koordination, Kommunikation, Zusammenarbeit* und *Information.* [Bair_89] grenzt diese Begriffe voneinander ab, indem er nach der Intensität der *Interaktion* zwischen Personen Ebenen der Zusammenarbeit definiert:

1. Information - informieren: Es kann anonym Information ausgetauscht werden, ohne daß sich die Beteiligten kennen müssen.

2. Koordination - koordinieren: Hier dient Kommunikation der gemeinsamen Nutzung von Information und anderer Ressourcen. Es ist nicht notwendig, gemeinsame Arbeitsziele zu verfolgen. Arbeitstätigkeiten können ineinandergreifen, um den Zugriff auf gemeinsam genutzte Ressourcen zu regeln. Die Beteiligten müssen sich flüchtig kennen.
3. Zusammenarbeit - zusammenarbeiten: Die Beteiligten nehmen an demselben Arbeitsprozeß teil. Sie sind jedoch meistens nicht gleichberechtigt. Obwohl es ein gemeinsames Ergebnis geben kann, wird die Leistung jedes Einzelnen getrennt bewertet. Jeder Einzelne ist meist Mitglied mehrerer Gruppen. Die Häufigkeit der Interaktion wechselt. Die Beteiligten stehen in einer Kollegenbeziehung.
4. Kooperation - kooperieren: Die Beteiligten arbeiten an einem gemeinsamen Ergebnis. Individuelle Ziele werden dem Gruppenziel untergeordnet, Entscheidungen werden gemeinsam getroffen. Die Leistung der Gruppe wird insgesamt bewertet, die Konkurrenz ist minimal. Die Gruppe lebt von der häufigen Interaktion und wird durch die gemeinsamen Ziele zusammengeschweißt.

Parallel zur Intensität der Zusammenarbeit nimmt nach [Lubich_95] die Verteilung des aufgabenbezogenen Wissens zu und es sinkt die Möglichkeit eines Einzelnen, mit individuellem Wissen die Aufgabe selbstständig zu lösen. Er stellt deshalb die *Kommunikation* als Basis jeder Kooperation dar (vgl. Abb. 3).

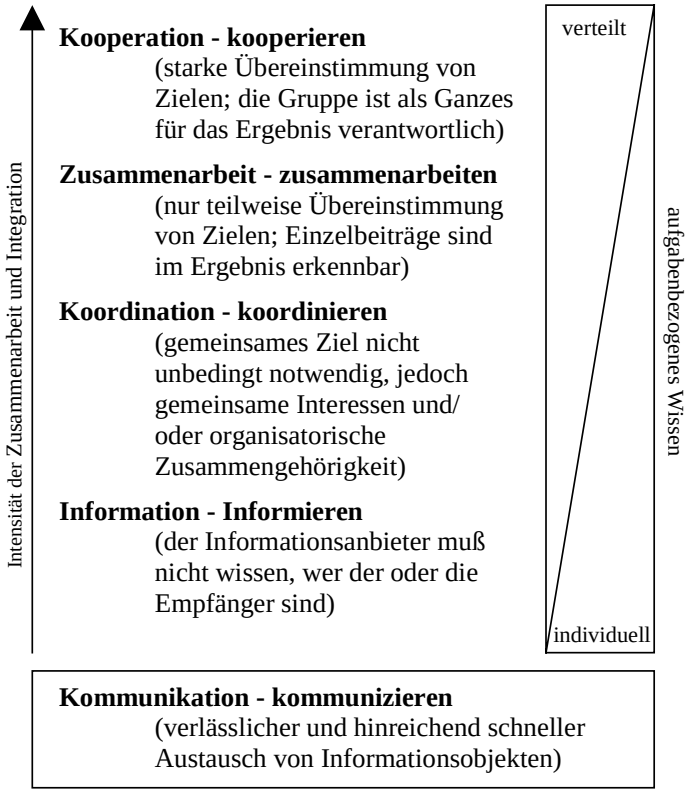


Abbildung 3: Ebenen der Kooperation in Anlehnung an [Lubich_95]

Koordination und Verständigung in kooperativer Arbeit beruhen demnach weitgehend auf Kommunikation.

[Oberquelle_91] beschränkt sich in seiner Definition kooperativer Arbeit nicht auf Kommunikation, sondern nennt zusätzliche wichtige Eigenschaften:

"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."

Nach diesen ersten Definitionen und Abgrenzungen kooperativer Arbeit wollen wir festhalten, daß wichtige Bedingungen erfüllt sein müssen, um kooperativ zusammenzuarbeiten. Einen Überblick über diese liefert [Piepenburg_91]:

Bedingungen von Kooperationen:
- **Zielidentität**: Es muß eine Übereinstimmung in Teilzielen der Beteiligten vorhanden sein.
- **Plan-Kompatibilität**: Handlungen, insbesondere bewußte Handlungen, laufen auf der Grundlage von Handlungsplänen ab. Kooperatives Arbeiten setzt voraus, daß neben der (evt. partiellen) Zielidentität die Handlungspläne der beteiligten Personen aufeinander abgestimmt und an individuellen Punkten miteinander gekoppelt werden.
- **Ressourcenaustausch**: Die notwendigen Randbedingungen der Handlungen müssen den beteiligten Handlungspartnern bekannt sein. Nur wenn ein Ressourcenaustausch gewährleistet ist, kann sich kooperatives Handeln entwickeln.
- **Regelbarkeit**: Das gegenseitige Beeinflussen der Handlungspläne und die flexible Angleichung von Teilplänen erfordern eine Regelbarkeit, die über die eigenen Handlunsanteile hinausgeht.
- **Kontrolle**: Die (interne) Kontrollierbarkeit der eigenen wie auch die (externe) Kontrollierbarkeit von Handlungsabfolgen der Kooperationspartner ist ein unabdingbarer Bestandteil der Regulierbarkeit des Handlungsablaufs.

3.1.2 Kooperationsformen

[Piepenburg_91] faßt **Formen von Kooperationen** in drei *Dimensionen:*

- *bilaterale vs. multiple Kooperation*
  Zur Aufwandsabschätzung kooperativer Handlungen differenziert er nach *bilateraler* Kooperation, d.h. Kooperation unter Beteiligung von zwei Partnern und *multipler* Kooperation, d.h. Kooperation unter Beteiligung von mehr als zwei Partnern.

- *konjunktive vs. disjunktive Kooperation*
  Diese Dimension bezieht sich auf die Art und Weise, in der Kooperationspartner das Arbeitsergebnis herbeiführen. Wirken die beteiligten Akteure mit eigenen, eindeutig definierten Handlungen auf den Handlungsablauf bis hin zur Zielerreichung, so spricht er von *konjunktiver* Kooperation. Verfolgt hingegen jeder Akteur vergleichbare individuelle Handlungspläne und ist es lediglich erforderlich, daß ein Akteur bis zum antizipierten Ziel vordringt, bezeichnet er dies als *disjunktive* Kooperation.

- *unmittelbare vs. mittelbare Kooperation:*
  Diese Dimension bezieht sich auf die räumliche und/oder zeitliche Distanz der Kooperationspartner. *Unmittelbare Kooperation* ist immer dann gewährleistet, wenn die Akteure die Möglichkeit einer direkten, nicht ausschließlich technisch vermittelten

Kommunikation haben (von Angesicht zu Angesicht). *Mittelbare Kooperation* bezieht sich auf alle übrigen Formen, also

- zeitversetzt, aber am gleichen Ort (z.B. Schichtarbeit);
- zeitgleich, an verschiedenen Orten; sowie
- zeitversetzt, an verschiedenen Orten.

[Teufel et al_95] untersuchen die Gruppenarbeit in der Unternehmensorganisation als *kooperatives System*:

"Ein **kooperatives System** besteht aus zwei oder mehreren *Akteuren*, die durch *aufgabenbezogene Handlungen* ein *gemeinsames Ziel* verfolgen. Für die dazu notwendige *Interaktion* und *Koordination* verwenden die Akteure bestimmte *Interaktionstypen, -modelle und -medien* sowie bestimmte *Koordinationstypen und -modelle*, die sie unter dem Blickwinkel der *Strukturiertheit adäquater Problemlösungsstrategien* und dem sich daraus ergebenden *Ausmaß an Handlungsinterdependenzen* auswählen." [Teufel et al_95]

Sie erweitern die oben genannten drei Dimensionen von Kooperation nach [Piepenburg_91] und benennen dabei **Merkmale eines kooperativen Systems**:

- ***Anzahl der Akteure***

Die Kooperation zwischen Akteuren wird in Abhängigkeit der Anzahl der beteiligten Akteure als *bilateral* oder *multipel* bezeichnet (vgl. [Piepenburg_91]).

- ***Ausmaß an Handlungsinterdependenzen***

Bezugnehmend auf das Ausmaß der Interdependenzen zwischen den Handlungen mehrerer Akteure wird nach *konjunktiver* und *disjunktiver* Kooperation unterschieden (vgl. [Piepenburg_91]). Die Unterscheidung in konjunktive und disjunktive Kooperation ist wichtig, wenn Handlungen in einem kooperativen System *koordiniert* werden müssen.

- ***Koordination***

Unter dem Begriff *Coordination Theory* schlagen [Malone et al_94] eine Differenzierung der Koordination in vier Komponenten vor (vgl. Abb. 4). Jeder dieser Komponenten (*Ziele, Akivitäten, Akteure und Interdependenzen*) werden entsprechende Koordinationsprozesse zugeordnet.

Komponenten der Koordination	**Zugehörige Koordinationsprozesse**
Ziele	Ziele identifizieren
Aktivitäten	Ziele dekomponieren, Aktivitäten planen
Akteure	Akteure bestimmen und Aktivitäten zuordnen
Interdependenzen	Interdependenzen handhaben

Abbildung 4: Komponenten der Koordination und zugehörige Koordinations-prozesse nach [Malone et al_94]

[Malone et al_94] bezeichnen dabei die *Handhabung von Interdependenzen* als das Hauptproblem der Koordination und differenzieren dieses in Probleme der Handhabung der *Reihenfolge von Aktivitäten*, der *Synchronisation* und der *Ressourcennutzung*.

Nach [Teufel et al_95] sind die Funktionen der Koordination grundsätzlich auf zwei Basisfunktionen reduzierbar, die *Allokation knapper Ressourcen* und die *Kommunikation relevanter Resultate*.

Ökonomische Betrachtungen soziotechnischer Systeme weisen auf folgende **Koordinationstypen** hin [Teufel et al_95]: *Marktmechanismus* (Koordination durch Preise), *hierarchische Leitungsbeziehung* (Pläne, Programme sowie hierarchische ad-hoc-Entscheidungen als Vorgaben für die Akteure) und *das Prinzip der verständigungsorientierten Koordination* (Koordination durch Selbstabstimmung der Akteure).

- ***Elementare Prozesse***

Die elementaren Prozesse eines kooperativen Systems sind:

1. *Interaktionen*, die im wesentlichen Kommunikationsprozesse zwischen beteiligten Akteuren sind,
2. *Handlungen*, die als aufgabenbezogene Arbeitsprozesse auftreten und
3. Prozesse der *Koordination* als Abstimmung aufgabenbezogener Tätigkeiten.

Abbildung 5 zeigt nach [Mintzberg_92] mögliche Gestaltungsvarianten der Abstimmungs- und Kommunikationsbeziehungen in Abhängigkeit des verwendeten Koordinationsmechanismus.

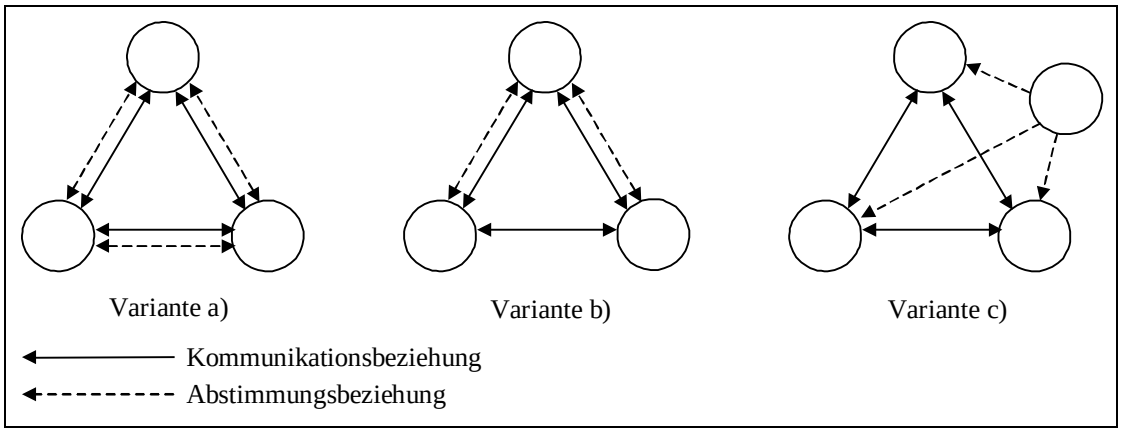


Abbildung 5: Gestaltungsvarianten zwischen Akteuren eines kooperativen Systems nach [Minzberg_92]

Bei allen Varianten wird vorausgesetzt, daß reziproke Kommunikationsbeziehungen zwischen allen Akteuren des kooperativen Systems bestehen. Variante a) liegt das Prinzip der Selbstabstimmung zugrunde. Dabei sind die Abstimmungsbeziehungen zwischen den einzelnen Akteuren reziprok. Variante b) zeigt die Beziehungen, die bei der Koordination durch hierarchische Leitungsbeziehungen notwendig sind. Die Abstimmungsbeziehungen gehen in diesem Fall von einem (oder mehreren) Akteuren aus. Variante c) zeigt einen wichtigen Spezialfall der Koordination durch hierarchische Leitungsbeziehungen, bei dem die Koordination durch die Vorgabe von Normen und Standards erfolgt. Die Abstimmungsbeziehung erfolgt ausgehend von einem oder mehreren Akteuren in Richtung der Akteure, welche die definierten Standards einhalten müssen. Die Standardisierung kann sich auf die Qualifikation der einzelnen Akteure, die Arbeitsprozesse oder die Arbeitsprodukte beziehen.

- ***Strukturiertheit der Problemlösungsstrategie***

In Abhängigkeit der Strukturiertheit einer angewandten Problemlösungsstrategie wird in Anlehnung an [Haugeneder_94] Kooperation als *rigide* oder *lose* bezeichnet. Rigide Kooperation liegt dann vor, wenn die organisatorischen Rahmenbedingungen die Zusammenarbeit der Akteure a priori vorgeben. In solchen Fällen weisen adäquate Problemlösungsstrategien einen hohen Grad an Strukturiertheit auf. Rigide Kooperation herrscht in kooperativen Systemen vor, die vorwiegend Koordinationsmechanismen gemäß Variante c) einsetzen (vgl. Abb. 5). Lose Kooperation liegt dann vor, wenn der Grad der Strukturiertheit gering ist und setzt reziproke Abstimmungsbeziehungen zwischen den Beteiligten voraus (Variante a).

- ***Interaktion***

Interaktionen zwischen Akteuren sind Kommunikationsprozesse. Solche Prozesse haben die Ausprägung *explizit*, wenn die Beteiligten aktiv Informationen austauschen wie in einem Gespräch von Angesicht zu Angesicht oder über elektronische Post-Systeme. Sie haben die Ausprägung *implizit*, wenn sie über das Medium gemeinsamer Informationsobjekte stattfinden, z.B. über gemeinsame Dokumente.
Außerdem werden noch *synchrone* und *asynchrone* Interaktion als **Interaktionstypen** genannt. Eine synchrone Interaktion liegt vor, wenn ein Akteur sie initiiert, und sie gleichzeitig bei allen Beteiligten wahrnehmbar wird. Dagegen ist eine Interaktion asynchron, wenn dies nicht zutrifft.
Weitere Merkmale zur Beschreibung der Interaktion sind die verwendeten **Interaktionsmodelle**, d.h. *1:1-, 1:n- und n:m-Kommunikationsbeziehungen* sowie die verwendeten **Interaktionsmedien**, d.h. *von Angesicht zu Angesicht, textuell, auditiv oder visuell*.

Abbildung 6 zeigt eine zusammenfassende Darstellung der Merkmale und möglichen Ausprägungen eines kooperativen Systems gemäß [Teufel et al_95].

Es sei noch darauf hingewiesen, daß sich in sehr **komplexen Kooperationsformen** wie z.B. in einem Krankenhaus eine Analyse für Außenstehende als sehr schwierig erweist, da die Kooperation in hohem Maß auf Konventionen und einem Vorverständnis zwischen den Beteiligten beruht. Das Besondere an diesen Kooperationsformen bezeichnen [Krabbel et al_96] mit dem Begriff *übergreifende Aufgaben*.

Die Merkmale von übergreifenden Aufgaben sind:

- Eine *Vielzahl von Einzelpersonen* unterschiedlicher Berufsgruppen mit z.T. sehr unterschiedlichen Tätigkeitsfeldern und Verantwortlichkeiten arbeiten zusammen. Dabei gehören die einzelnen Personen oft unterschiedlichen organisatorischen Einheiten an und arbeiten in der Regel räumlich voneinander getrennt.
- Die Kooperation zwischen den Beteiligten beruht auf *etablierte Konventionen*. Gleichzeitig erfordert sie ein hohes Maß an *Flexibilität*.
- Die Zusammenarbeit besteht oftmals nur aus *einfachen und kurz zu erledigenden Teiltätigkeiten*, die jedoch wiederholt und in *engen Zeitvorgaben* erledigt werden müssen.
- Die Kooperation erfordert eine Vielzahl von *Einzeltätigkeiten zur Koordination*, für die sich ebenfalls bestimmte Formen etabliert haben.
- Die Zusammenarbeit setzt an vielen Stellen ein *sicheres Verständnis* der Beteiligten über die jeweils flexibel von ihnen geforderten Teiltätigkeiten voraus.

- Nicht jeder Mitarbeiter hat einen festen Arbeitsplatz, sondern die Aufgaben müssen an *verschiedenen Orten* erledigt werden.
- Es herrscht z.T. eine *starke Arbeitsteilung* zwischen Personen eines Berufsfeldes vor.

Merkmale eines kooperativen Systems	*Ausprägung der Merkmale*
Anzahl Akteure	• zwei: bilaterale Kooperation • mehrere: multiple Kooperation
Strukturiertheit der Problemlösungsstrategie	• hoch: rigide Kooperation • gering: lose Kooperation
Ausmaß der Handlungsinterdependenzen	• hoch: konjunktive Kooperation • gering: disjunktive Kooperation
elementare Prozesse	• Interaktionen (Kommunikationsprozesse) • Arbeitsprozesse (aufgabenbezogene Handlungen) • Koordinationsprozesse (Abstimmung aufgabenbezogener Handlungen)
verwendete Koordinationstypen	• Selbstabstimmung • Markt • Hierarchie – hierarchische Leitungsbeziehungen – Standardisierung der – Qualifikation der Akteure – Arbeitsprozesse – Arbeitsprodukte
verwendete Interaktionstypen	• explizit/implizit • synchron/asynchron
verwendete Interaktionsmodelle	• 1:1-Kommunikationsbeziehung • 1:n-Kommunikationsbeziehung • n:m-Kommunikationsbeziehung
verwendete Interaktionsmedien	• von Angesicht zu Angesicht • textuell • auditiv • visuell

Abbildung 6: Merkmale eines kooperativen Systems nach [Teufel et al_95]

Mit den in diesem Abschnitt vorgestellten Merkmalen haben wir eine Möglichkeit gefunden, kooperative Arbeitssituationen zu analysieren, was wir im nächsten Abschnitt in Bezug auf unser Anwendungsbeispiel tun werden.

3.1.3 Diskussion der Kooperationsformen im Anwendungsbeispiel

Wir charakterisieren in diesem Abschnitt unser Anwendungsbeispiel, d.h. die Lehrplanung des Fachbereichs Informatik der Universität Hamburg aus Sicht des Arbeitsbereichs Softwaretechnik, in Bezug auf Kooperationsformen.

Zunächst jedoch wollen wir gemäß der Einteilung von Bair (vgl. Kapitel 3.1.1) die Intensität der Zusammenarbeit zwischen beteiligten Akteuren einstufen:

Die Beteiligten nehmen alle an demselben Arbeitsprozeß, nämlich der Lehrplanerstellung, teil. Sie haben also das gemeinsame übergeordnete Ziel der Lehrplanerstellung. Nicht alle Akteure sind gleichberechtigt, da z.B. AB-Leiter und Veranstalter unterschiedliche Positionen haben. Andererseits werden Entscheidungen z.B. in der AB-Lehrplansitzung gemeinsam

getroffen, wobei die Gruppe der Mitarbeiter am Arbeitsbereich durch die häufige Interaktion einen hohen Zusammenhalt hat. Dies wird dadurch verstärkt, daß die Erstellung des Lehrplans insgesamt bewertet wird, so daß die Konkurrenz minimal ist.

Die Zusammenarbeit enthält demnach mindestens Anteile der Ebene 3 bei Bair (vgl. Kapitel 3.1.1), nimmt aber auch intensivere Ausmaße wie auf Ebene 4 ein.

Nun untersuchen wir das Beispiel nach den Merkmalen von kooperativen Systemen, die im vorherigen Abschnitt vorgestellt wurden.

- ***Anzahl der Akteure***

Es handelt sich um eine *multiple Kooperation,* da es mehr als zwei Akteure gibt.

- ***Ausmaß der Handlungsinterdependenzen***

Die Beteiligten wirken mit eigenen Handlungsanteilen auf den Ablauf. Dabei gibt es ein hohes Maß an Interdependenzen. Erst wenn alle Beteiligten ihre Teilhandlungen ausgeführt haben und das jeweilige individuelle Teilziel erreicht wurde, kann das übergeordnete Kooperationsziel erreicht werden. Es handelt sich also um *konjunktive Kooperation*, bei der die Teilhandlungen *koordiniert* werden müssen.
Ein Beispiel dazu ist der Nachweis der Erfüllung der Lehrdeputate, wo die Veranstalter ein Formular von der FB-Verwaltung erhalten, dieses ausfüllen müssen, vom AB-Leiter gegenzeichnen lassen und an die FB-Verwaltung zurücksenden.

- ***Koordination***

Neben der eben erwähnten Koordination von Teilhandlungen müssen an verschiedenen Stellen sowohl die *Allokation knapper Ressourcen* als auch die *Kommunikation relevanter Resultate* koordiniert werden. Knappe Ressourcen sind z.B. die Räume, in denen die Lehrveranstaltungen stattfinden werden, die Kapazitäten des Fachbereichs, sowie die Zeit der Veranstalter. Als relevante Resultate, die kommuniziert werden müssen, kann man z.B. die bestätigten Lehrdeputate oder den Raum-/Zeitplan betrachten.
Es kommen zwei Koordinationstypen vor. Einerseits findet man an einigen Stellen eine *hierarchische Leitungsbeziehung*, insbesondere werden Zielvorgaben gemacht wie z.B. die zu erfüllenden Lehrdeputate durch die FB-Verwaltung. Andererseits findet während der Lehrplanungssitzung eine *verständigungsorientierte Koordination* statt, wo die Beteiligten sich im Prozeß der *Selbstabstimmung* koordinieren.

- ***Elementare Prozesse***

Es kommen alle Arten von elementaren Prozessen vor. Nähere Ausführungen dazu werden unter den entsprechenden Punkten Interaktionen, Handlungsinterdependenzen und Koordination gemacht.

- ***Strukturiertheit der Problemlösungsstrategie***

Die Strukturiertheit der Interaktions- und Handlungsbeziehungen zwischen den kooperierenden Akteuren ist bei den Prozessen der Selbstabstimmung naturgemäß gering, so daß an diesen Stellen eine *lose Kooperation* herrscht.
Überall dort, wo die Interaktion einzelner Akteure darin besteht, Dokumente auszutauschen, erkennt man einen strukturierten Ablauf. Die individuellen Teilhandlungen der Akteure werden dabei synchronisiert. Die Kooperation ist dann *rigide*.

- ***Interaktion***

Es kommen sowohl explizite als auch implizite Interaktionen im Lehrplanungsprozeß vor. Am Arbeitsbereich ist durch die Räumlichkeiten (alle Büros auf demselben Flur) die

Möglichkeit gegeben, jederzeit direkt von *Angesicht zu Angesicht* zu kommunizieren. Diese Möglichkeit wird häufig genutzt, wobei sozusagen Gespräche ”zwischen Tür und Angel” entstehen. In diesen Fällen findet also eine *explizite, synchrone* Interaktion statt. Ein weiteres Beispiel dafür ist die Lehrplanungssitzung, bei der alle Akteure sich direkt an der Diskussion beteiligen.
Als implizite Interaktionen kann man die oben beschriebenen Prozesse, bei denen gemeinsame Informationsobjekte (Dokumente) ausgetauscht werden, einordnen, d.h. die Interaktion findet über *textuelle* Medien statt. Die Kooperation ist dabei *asynchron* und *an verschiedenen* Orten.
Alle Arten von Kommunikationsbeziehungen kommen vor. So sind z.B. *1:1-Kommunikationsbeziehungen* alle Gespräche ”zwischen Tür und Angel” und *n:m-Beziehungen* die Kommunikation auf der Lehrplanungssitzung.
Ein interessanter Punkt ist die *1:n-Kommunikation*, die vom FB-Lehrplaner ausgeht und mit den AB-Lehrplanungsbeauftragten stattfindet. Hierbei initiiert der FB-Lehrplaner die Lehrplanungen an den Arbeitsbereichen und wartet dann auf deren erfolgreiche Beendigung, um dann seine eigenen Aufgaben weiterführen zu können. Er hat also die Rolle eines Koordinators.

Zusammenfassend läßt sich die im Lehrplanungsprozeß stattfindende Kooperation folgendermaßen charakterisieren:

Es handelt sich um eine multiple Kooperation, bei der im wesentlichen drei Kooperationsarten zu erkennen sind:

a) Aus Sicht des FB-Lehrplaners verläuft der Gesamtprozeß *sternförmig*: nachdem die jeweilige Lehrplanung an allen ABs parallel abgelaufen ist, werden ihm die Ergebnisse zugesendet, wodurch eine *Synchronisation* durch den FB-Lehrplaner erfolgen kann. Eine bildhafte Vorstellung dieses Ablaufs führt zum Begriff der sternförmigen Prozesse.

b) Die AB-Lehrplanung läßt sich als *verständigungsorientierten Prozeß* charakterisieren. Die beteiligten Akteure koordinieren sich untereinander durch *Selbstabstimmung*, wobei die Interaktion explizit durch Gespräche oder implizit durch Dokumentenaustausch erfolgt. Die Kooperation ist lose.

c) Es finden *fest strukturierte, standardisierte Prozesse* statt, die wiederholt jeweils dieselbe vorbestimmte Reihenfolge von Arbeitsschritten durchlaufen. Es handelt sich um eine rigide Kooperation. Bei diesen Prozessen erfolgt die Koordination dokumentenorientiert, d.h. die Teilhandlungen werden durch die Transaktion von Dokumenten koordiniert. Beispiele hierfür sind die Erstellung von Lehrberechtigungsanträgen sowie die Erstellung des Raum-/Zeitplans.

3.2 CSCW - Computer Supported Cooperative Work

Dieser Abschnitt führt die Begriffe Computer Supported Cooperative Work (CSCW) und Groupware ein (3.2.1) und stellt Klassifizierungsmöglichkeiten von CSCW-Applikationen vor (3.2.2).

3.2.1 Begriffsklärung

Das Forschungsgebiet Computerunterstützung für kooperative Arbeit (engl. *Computer Supported Cooperative Work - CSCW*) befaßt sich mit Fragestellungen der Entwicklung und des Einsatzes von spezialisierten Applikationen für Arbeitsgruppen, Teams und den dabei zu berücksichtigenden betriebswirtschaftlichen, psychologischen und soziologischen sowie technischen Aspekten. Aus Sicht der Informatik werden alle Methoden, Techniken und Werkzeuge, die versuchen, spezifische Unterstützungsbedürfnisse von Gruppen mit Informations- und Kommunikationstechnologie zu befriedigen, dem Begriff zugeordnet [Teufel et al_95].

Der Begriff CSCW wird uneinheitlich verwendet und häufig mit *Groupware* gleichgesetzt. [Grudin_94] spricht in dem Zusammenhang sogar vom ”Tower of Babel”-Problem: in verschiedenen Gebieten werden dieselben Begriffe auf verschiedene Art und Weise verwendet.
Es hat sich jedoch die Sichtweise durchgesetzt, unter dem Begriff CSCW allgemein das Forschungsgebiet als solches und mit Groupware die in diesem Gebiet entwickelte Software zu verstehen.
Wir schließen uns der Definition von [Teufel et al_95] an, die die Interdisziplinarität von CSCW hervorheben:

“*Computer Supported Cooperative Work (CSCW)* ist die Bezeichnung des Forschungsgebietes, welches auf interdisziplinärer Basis untersucht, wie Individuen in Arbeitsgruppen oder Teams zusammenarbeiten und wie sie dabei durch Informations- und Kommunikationstechnologie unterstützt werden können.” [Teufel et al_95]

Der Begriff Groupware wird abgrenzend zu CSCW von [Teufel et al_95] auch als *CSCW-Applikation* bezeichnet und folgendermaßen definiert:

“Groupware bzw. CSCW-Applikationen sind aus Software und eventuell spezifischer Hardware bestehende Systeme, durch die Gruppenarbeit unterstützt oder ermöglicht wird.” [Teufel et al_95]

Wir werden uns im Folgenden diesen Definitionen anschließen und mit CSCW die Disziplin bzw. das Forschungsgebiet als solches bezeichnen, während Groupware bzw. CSCW-Applikationen daraus hervorgehende Produkte bezeichnet.

3.2.2 Klassifikation von CSCW-Applikationen

Es existieren eine Vielzahl von CSCW-Applikationen. Diese Applikationen lassen sich nach ihrer Funktionalität bestimmten Applikationstypen zuordnen.
Die folgende Abbildung 7 zeigt eine funktionale Klassifikation der bekanntesten Applikationstypen nach [Böhm et al_96].

Klasse	**Beschreibung**
E-mail	Dient zum Austausch meist kurzer Mitteilungen zwischen zwei oder mehreren Personen.
Message conferencing	Bedeutet textbasiertes, asynchrones, aber auch synchrones elektronisches Konferieren.
File Transfer	Ein Austausch von Dateien, die z.B. Dokumente oder Grafiken enthalten können.
Document Management Systems	Zentrale Verwaltung von gemeinsam bearbeiteten Dokumenten. Das Spektrum der Systeme reicht von einfachen Dateiverwaltungssystemen bis hin zu komplexen Systemen.
Shared Blackboard	Bei Systemen wie Shared Blackboard können die Teilnehmer gleichzeitig in ein gemeinsames Fenster schreiben bzw. zeichnen.
Desktop Conferencing	Mehrere Teilnehmer können computerunterstützt an Audio- bzw. Videokonferenzen vom Schreibtisch aus teilnehmen.
Joint Viewing	Hierbei wird ein Dokument von einem Präsentator an die Teilnehmer einer Sitzung übertragen.
Joint Editing	Verschiedene Teilnehmer einer Sitzung können gleichzeitig ein gemeinsames Dokument auf ihren Bildschirmen bearbeiten. Alle Änderungen sind für jeden Teilnehmer sofort sichtbar und geschehen im Gegensatz zu Shared Blackboard Systemen unmittelbar im Dokument.
Scheduler	Mit Hilfe eines Schedulers können personelle und physikalische Ressourcen koordiniert und auf diese Weise Sitzungen vorbereitet werden.
Decision Support	Ein Group Decision Support System (GDSS) bietet die automatische Verarbeitung von Daten und Sitzungsresultaten. Dabei können die Teilnehmer ihre Gedanken und Ideen eingeben, und das System unterstützt die Organisation und Auswertung dieser Dateien.
Workflow Management System	In Workflow Management Systemen (WfMS) werden wohldefinierte Geschäftsprozesse repräsentiert. Diese Systeme dienen dazu, die nötigen Schritte eines solchen Prozesses schnell und effektiv in der richtigen Reihenfolge durch die betreffenden Personen durchführen lassen zu können.

Abbildung 7: Merkmale kooperativer Systeme und ihre Ausprägungen nach [Böhm et al_96]

In [Teufel et al_95] werden noch zwei Applikationstypen ergänzt:

- **Bulletin Board-Systeme**

Sind spezielle Datenbanken, die Meldungen verschiedener Autoren nach Themenschwerpunkten speichern und einer Vielzahl von Lesern zur Verfügung stellen. Meldungen eines Themas bilden eine sogenannte Interessensgruppe (engl. *News Group*). Die Meldungen besitzen ein semi-strukturiertes Format, ähnlich zu dem einer Meldung in elektronischen Post-Systemen.

- **Verteilte Hypertext-Systeme**

Stellen einer Vielzahl von Lesern Hypertext-Dokumente, d.h. nicht-linear strukturierte Informationen in Form von Texten und einfachen Graphiken, zur Verfügung.

Solche Anwendungstypen lassen sich in verschiedenen Schemata positionieren. Die Unterscheidung bezüglich der geographischen und zeitlichen Verteilung der Beteiligten bildet eine sehr bekannte Klassifikation, die sogenannte *Raum-Zeit-Matrix.* Eine weitere Möglichkeit ist die Gliederung nach *Unterstützungsfunktionen*, wobei zwischen Kommunikations-, Koordinations- und Kooperationsfunktionen unterschieden wird. Im Folgenden stellen wir diese beiden Schemata vor.

Raum-Zeit-Matrix

Nimmt man die geographische und die räumliche Verteilung der Beteiligten als Kriterium, so entsteht die Raum-Zeit-Matrix. [Grudin_94] unterscheidet jeweils drei Kategorien und bildet somit eine 3x3-Matrix. Abbildung 8 zeigt diese Matrix mit jeweils repräsentativen Anwendungstypen.

		TIME		
		Same	**Different but predictable**	**Different and unpredictable**
PLACE	**Same**	Meeting facilitation	Work shifts	Team rooms
	Different but predictable	Teleconferencing Videoconferencing Desktop conferencing	Electronic mail	Collaborative writing
	Different and unpredictable	Interactive multicast seminars	Computer bboards	Workflow

Abbildung 8: Raum-Zeit-Matrix nach [Grudin_94]

Die Beteiligten können sich
a) räumlich benachbart (*same place*),
b) an verschiedenen Orten, die bekannt sind (*different, but predictable place*) wie beim Austausch von E-mail, oder
c) an verschiedenen Orten, die nicht alle bekannt sind (*different, but unpredictable place*) wie beim Versenden einer Nachricht an eine Bulletin Board-News Group,

befinden.

Ihre Aktivitäten können dabei
a) synchron (*same time*) wie bei einem Meeting bzw. in einer Konferenz,
b) zu verschiedenen, aber vorhersehbaren Zeitpunkten (*different, but predictable time*) wie beim Austausch von E-mails unter Kollegen[1] oder
c) zu verschiedenen, unvorhersehbaren Zeitpunkten (*different, but unpredictable time*) wie beim gemeinsamen Schreiben eines Textes

stattfinden.

Die Raum-Zeit-Matrix liefert eine gute Möglichkeit, technische Anforderungen von CSCW-Applikationen zu identifizieren und wird deshalb besonders von Groupware-Entwicklern genutzt.
Sie birgt aber die Gefahr, organisatorische Zusammenhänge zu vernachlässigen. Da kooperative Arbeit aus unterschiedlichen Formen zusammenhängender Aktivitäten besteht (vgl. Kapitel 3.1.1), läßt sie sich nie nur einer einzigen der Kategorien zuordnen.

Kassifikation nach Unterstützungsfunktionen

Unabhängig von der Technologie können Applikationen nach ihren Unterstützungsfunktionen gegliedert werden. Unterstützbare Prozesse der Gruppenarbeit sind Kommunikations-, Koordinations- und Kooperationsprozesse (vgl. Kapitel 3.1.1). Je nach Schwergewicht der Betonung auf die eine oder die andere Unterstützungsfunktion lassen sich die verschiedenen Applikationstypen in dem in Abbildung 9 gezeigten Dreieck positionieren [Teufel et al_95].

Faßt man die so positionierten Applikationstypen zu Systemklassen zusammen, lassen sich folgende, sich überschneidende, Klassen bilden [Teufel et al_95]:

1. *Systemklasse Kommunikation*
Kommunikationssysteme ermöglichen den Informationsaustausch zwischen verschiedenen Kommunikationspartnern. Dabei werden in erster Linie Raum- und Zeitdifferenzen überbrückt. Solche Systeme zählen zu den ältesten CSCW-Entwicklungen und sind dementsprechend am weitesten verbreitet. Typische Beispiele sind elektronische Post-Systeme und Videokonferenzsysteme.

2. *Systemklasse gemeinsame Informationsräume*
Diese Klasse stellt gemeinsame Informationsräume für eine Gruppe zur Verfügung, in denen Informationen längere Zeit in geeigneter Form und mit Hilfe geeigneter Zugriffsmechanismen gespeichert werden. Der Informationsaustausch ist implizit (vgl. Kapitel 3.1.2). In diese Klasse fallen verteilte Hypertext-Systeme und spezielle Datenbanken, deren Informationen gleichzeitig von verschiedenen Benutzern abgefragt werden können. Bulletin Board-Systeme können ebenfalls zu dieser Klasse gezählt werden.

3. *Systemklasse Workflow Management*
Der Systemklasse Workflow Management sind Applikationstypen zugeordnet, welche für die Unterstützung betrieblicher Abläufe einsetzbar sind. Die Unterstützungsfunktion von Workflow Management-Systemen (WfMS) bezieht sich hauptsächlich auf die *Koordination* räumlich oder zeitlich verteilter Arbeitsgruppen. Dazu werden unter anderem Techniken aus dem Bereich elektronischer Post-Systeme und spezieller Datenbanksysteme eingesetzt.

[1] Grudin sieht den Austausch von E-mails unter Kollegen unter vorhersehbaren Zeitpunkten. Das ist unserer Meinung nach problematisch, da man nie genau vorhersehen kann, wann jemand eine E-mail letzendlich liest.

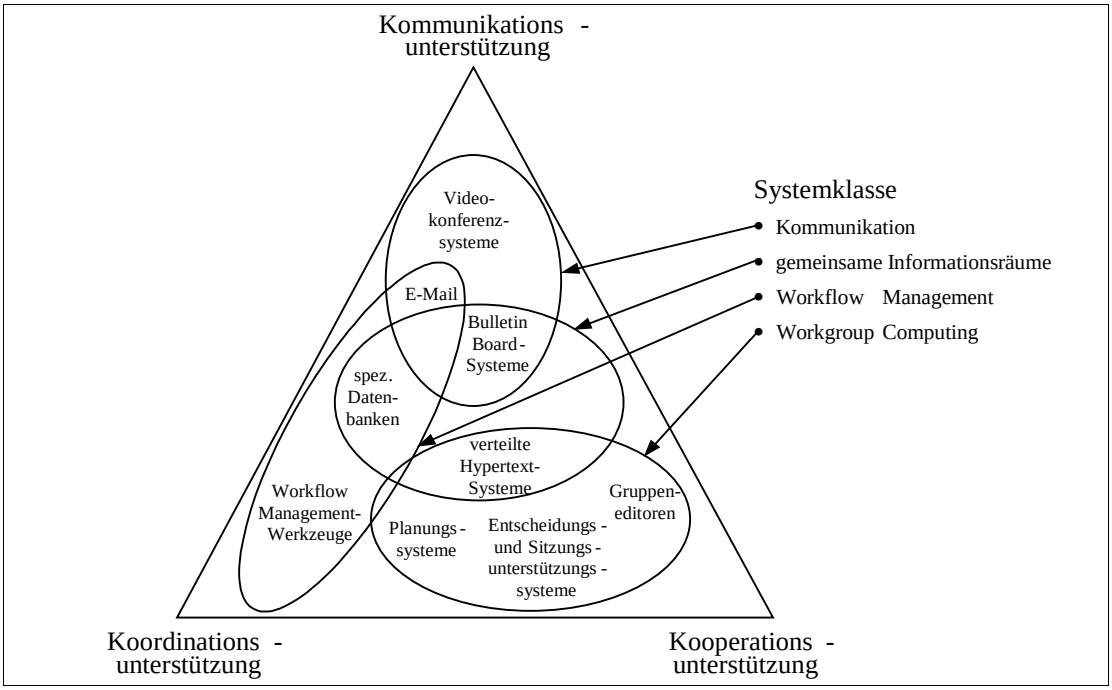


Abbildung 9: Klassifikationsschema nach Unterstützungsfunktionen

4. *Systemklasse Workgroup Computing*
Workgroup Computing-Systeme unterstützen die Kooperation von Personen, die in Gruppen oder Teams arbeiten und Aufgaben mit mittleren bis geringen Strukturierungsgraden und Wiederholungsfrequenzen zu lösen haben. Die Koordinationsfunktion bezieht sich hier auf die für die Problemstellung notwendigen Kooperationsbeziehungen innerhalb der Gruppe. In diese Klasse gehören Planungssysteme wie Terminverwaltungs- und -vereinbarungssysteme, Gruppeneditoren und Entscheidungs- und Sitzungsunterstützungssysteme.

Die Klassifikation nach Unterstützungsfunktionen bietet unserer Meinung nach eine gute Möglichkeit, für einen konkreten Anwendungsfall geeignete CSCW-Applikationen zu finden, indem man den Anwendungsfall auf mögliche Unterstützungsschwerpunkte untersucht. Dies wollen wir im folgenden Abschnitt mit unserem Anwendungsbeispiel tun.

3.2.3 Diskussion der Integration von CSCW-Applikationen im Anwendungsbeispiel

Wir wollen nun nach Unterstützungsmöglichkeiten des Anwendungsbeispiels durch CSCW-Applikationen suchen. Als Grundlage dient die Diskussion der Kooperationsformen im Anwendungsbeispiel (vgl. Kapitel 3.1.3).

Wir stellen zunächst die Frage, welche Unterstützungsschwerpunkte im Beispiel zu finden sind. Dazu gehen wir der Reihe nach die Kommunikations-, Kooperations- und Koordinationsunterstützungsfunktionen durch (vgl. Kap. 3.2.2):

- *Kommunikationsunterstützung*
  Die Mitarbeiter des Arbeitsbereichs befinden sich in umittelbarer räumlicher Nähe. Die explizite synchrone Kommunikation zwischen den Mitarbeitern ist also jederzeit möglich und muß nicht weiter unterstützt werden. Bei Abwesenheit eines Mitarbeiters findet die Kommunikation über ein vorhandenes E-mail-System statt.

Es findet im Lehrplanungsprozeß ein impliziter Informationsaustausch über gemeinsame Dokumente statt. Die Schaffung eines gemeinsamen Informationsraumes, z.B. über eine spezielle Datenbank, erscheint sinnvoll.

- *Kooperationsunterstützung*
  Die AB-Lehrplanungssitzung kann durch ein Sitzungsunterstützungssystem oder auch ein Planungssystem vorbereitet werden. Andererseits sind die Vorbereitung und die Resultate der Sitzung organisatorisch vorgegeben. Die Sitzung dient eher der Koordination der Akteure. Somit erscheint eine Koordinationsunterstützung sinnvoller (s.u.).
- *Koordinationsunterstützung*
  Die Teilhandlungen im Lehrplanungsprozeß erfordern ein hohes Maß an Koordination. Der Einsatz von Workflow Management-Werkzeugen wäre vor allem für den AB-Lehrplanungsbeauftragten, der den Lehrplanugsprozeß initiiert und koordiniert sehr hilfreich.

Zusammenfassend läßt sich feststellen, daß vorwiegend die Applikationstypen der Systemklasse Workflow Management-Systeme (WfMS), d.h. E-mail, spezielle Datenbanken und Workflow Management-Werkzeuge, Unterstützung bringen können. Aus diesem Grund vertiefen wir im nächsten Abschnitt die Konzepte von WfMS, um dann abschließend Aussagen über eine Systemrealisierung machen zu können.

3.3 Workflow Management Systeme (WfMS)

3.3.1 Begriffsklärung

Die zentralen Begriffe des Gebietes Workflow Management werden nicht einheitlich verwendet. Wir definieren deshalb hier die wichtigsten Begriffe nach [Teufel et al_95]:

- *Workflow*
  Ein Workflow ist eine endliche Folge von Aktivitäten, wobei die Folge durch Ereignisse ausgelöst und beendet wird (Begriffe wie Business Process, Vorgangskette, Prozeßkette, Geschäftsprozeß oder Geschäftsvorgang werden hier als Synonyme des Begriffes Workflow verstanden).
- *Workflow Management*
  Workflow Management umfaßt alle Aufgaben, die bei der Modellierung, der Simulation sowie bei der Ausführung und Steuerung von Workflows erfüllt werden müssen.
- *Workflow Management-System (WfMS)*
  Ein Workflow Management-System ist ein aus mehreren Werkzeugen bestehendes System, welches die Aufgaben des Workflow Managements durch die Ausführung von Software unterstützt.

Laut [Jablonski_97a, Glossar] versteht man unter einem WfMS

“ ... ein (re-)aktives Basissoftwaresystem zur Steuerung des Arbeitsflusses (Workflows) zwischen beteiligten Stellen nach den Vorgaben einer Ablaufspezifikation (Workflow-Schema). Zum Betrieb eines WfMS sind Workflow-Management-Anwendungen zu entwickeln. Ein WfMS unterstützt mit seinen Komponenten sowohl die Entwicklung (Modellierungskomponente) von Workflow-Management-Anwendungen als auch die Steuerung und Ausführung (Laufzeitkomponente) von Workflows.”

Aus dieser Definition wird als ein wesentliches Charakteristikum von WfMS deutlich, daß es sich um aktive, also ablaufsteuernde Softwaresysteme handelt, wobei der Ablauf durch ein sogenanntes *Workflow-Schema* (oder Workflow-Modell) festgelegt wird. Um ein Modell der Vorgänge einer Anwendungssituation erstellen zu können, welches als Grundlage zur Ablaufsteuerung verwendet werden soll, bedarf es detaillierter Kenntnisse der Struktur dieser Vorgänge.

Arbeitsvorgänge in der Realität (die ja Bestandteil von Geschäftsprozessen sind, die von WfMS ausgeführt werden sollen) zeichnen sich allerdings auch durch unstrukturierte Teile aus, die ebenfalls in den Anwendungen berücksichtigt werden müssen. Inwieweit dies von dem jeweiligen WfMS unterstützt wird (und ob eine Unterstützung überhaupt möglich ist), hängt von den individuellen Fähigkeiten des Systems ab. Laut [Jablonski_97a, S.6] werden die Fähigkeiten eines WfMS bzw. einer Workflow-Management-Anwendung hinsichtlich der Unterstützung von Arbeitsvorgängen nachhaltig dadurch bestimmt, welche Möglichkeiten zur Darstellung der Geschäftsprozesse angeboten werden. Diese Möglichkeiten werden wiederum durch die verwendete Workflow-Sprache des jeweiligen Produktes bestimmt, wodurch die Sprache eines WfMS als Beurteilungsgrundlage der Leistungsfähigkeit verwendet werden kann.

3.3.2 Theoretische Konzepte von WfMS

Workflow Management-Systemen (WfMS) sind Applikationstypen zugeordnet, welche für die Unterstützung *betrieblicher Abläufe* einsetzbar sind. Die Unterstützungsfunktion bezieht sich dabei hauptsächlich auf die *Koordination* räumlich oder zeitlich verteilter Arbeitsgruppen. Aus anwendungsorientierter Sicht stellt sich die Frage über die Art und Funktionsweise der Koordination. In diesem Kapitel wollen wir deshalb Modelle und Prinzipien zur Beschreibung von Koordinationsprozessen vorstellen.

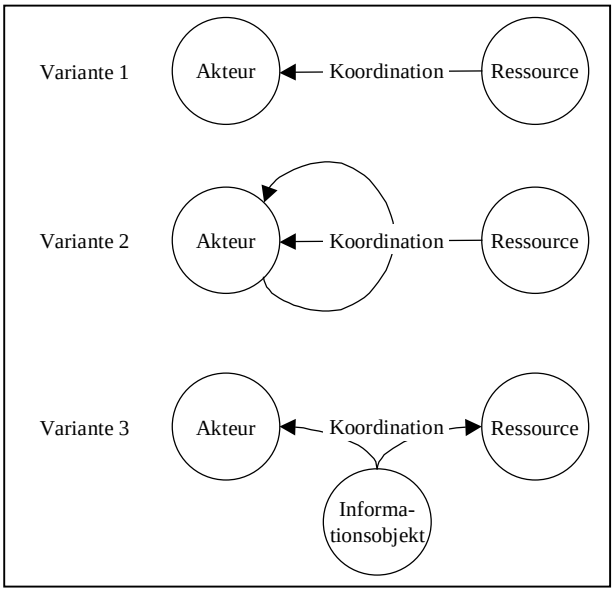


Abbildung 10: Varianten von Koordinationsstrukturen nach [Teufel et al_95]

Die von [Malone et al_94] präsentierte Coordination Theorie (vgl. Kapitel 3.2.2) nennt die *Handhabung von Interdependenzen* als das Hauptproblem der Koordination, welches in Probleme der Handhabung der *Reihenfolge von Aktivitäten*, der *Synchronisation* und der *Ressourcennutzung* differentiert werden kann.

Betrachtet man ausschließlich die an der Koordination beteiligten Objekte (Akteure, Ressourcen und gemeinsam wahrgenommene Informationsobjekte), so sind analog zu [Teufel et al_95] grundsätzlich drei Varianten einer Koordinationsstruktur denkbar (s. Abb. 10).

In der ersten Variante wird die Koordination von der verwendeten Ressource wahrgenommen, in der zweiten Variante übernehmen die beteiligten Akteure die Koordination selbst und in der dritten Variante erfolgt die Koordination durch gemeinsam wahrgenommene Informationsobjekte.

Um in der Realität eine dieser Varianten umsetzen zu können, sind Protokolle notwendig, mit deren Hilfe die beteiligten Objekte zu Koordinationszwecken miteinander kommunizieren können.

Grundsätzlich kann nach [Dittrich_91] zwischen *sozialen* und *technischen* Protokollen unterschieden werden. Die Vorteile von sozialen Protokollen liegen in der höheren Flexibilität in bezug auf wechselnde Situationen, der höheren Bandbreite der Kommunikation sowie der Herstellung sozialer Kontexte im Rahmen der Gruppenarbeit (*welcher* Akteur macht im Moment *was*). Allerdings versagen soziale Protokolle in Situationen, in denen durch die physischen Bedingungen sowie durch die Komplexität und den Umfang der Gruppenarbeit die Herstellung adäquater sozialer Kontexte nicht möglich ist. In solchen Fällen bieten technische Protokolle Vorteile, indem sie durch eindeutige und vordefinierte Regeln die Koordination sicherstellen und den fehlenden sozialen Kontext partiell ersetzen. Die Nachteile technischer Protokolle liegen in ihrer geringeren Flexibiltät und der Schwierigkeit, adäquate Ausnahmebehandlungsmechanismen zur Verfügung zu stellen.

CSCW-Applikationen benutzen zur Beschreibung und Definition von Koordinationsprozessen technische Protokolle. Diese unterscheiden sich darin, wie die wechselseitigen Abhängigkeiten zwischen den Einzelaktivitäten der Gruppenteilnehmer modelliert werden.

[Prinz_89] ordnet Modelle für die Koordination grob in vier Klassen ein:

- *Formularorientierte Modelle*

Besonders in Büroumgebungen hat die Benutzung von *Formularen* eine große Tradition. Eine gemeinsame Bearbeitung wird typischerweise als eine Art Umlauf eines gemeinsamen Informationsobjekts (Formulars) modelliert. Die Koordinationsstruktur entspricht der Variante 3 in Abbildung 10. Mit diesem Ansatz werden primär serielle Abläufe realisiert, wobei jedes Formular selbst die Ablaufkontrolle beinhaltet. Ein Beispiel für ein System, das diesen Ansatz verwendet ist Object Lens.

- *Vorgangsorientierte Modelle*

Bei den vorgangsorientierten Modellen steht die Koordination der jeweils beteiligten Einzelaktivitäten zu einem Vorgang im Vordergrund. Hinsichtlich ihrer Koordinationsstruktur sind vorgangsorientierte Modelle der Variante 1 in Abbildung 10 zuzuordnen. Essentielles Merkmal dieser Ansätze ist, daß das Wissen über den zu koordiniernden Vorgang logisch zentral in einer sogenannten Coordination entity gespeichert wird, welche die Aktivitäten der am Vorgang beteiligten Akteure kontrolliert und steuert. Mit Hilfe dieses Ansatzes können sowohl serielle als auch parallele Abläufe beschrieben werden.

- *Kommunikationsorientierte Modelle*

Bei diesem Ansatz, der die zugrunde liegenden Kommunikationsstukturen betrachtet, dienen rollenorientierte Beschreibungen dazu, Regeln für die Interaktionsmuster zwischen den Rollenträgern festzulegen. Dazu werden Konditionen spezifiziert, unter denen ein Informationsobjekt bearbeitet bzw. weitergeleitet wird.
Hinsichtlich ihrer Koordinationsstruktur können kommunikationsorientierte Modelle als Mischform aus den Varianten 1 und 2 in Abbildung 10 klassifiziert werden.

- *Konversationsorientierte Modelle*

Grundlage dieses Ansatzes ist die Sprechakttheorie (vgl. [Winograd et al_86]). Diese geht von der Annahme aus, daß Menschen durch sogenannte Sprechakte handeln. Die Idee, die hinter konversationsorientierten Ansätzen steckt, ist, daß Meldungen nach vordefinierten Klassen von Sprechakten klassifiziert werden können, um dann dem Benutzer mögliche Sprechakte für Antworten zur Verfügung zu stellen.

[Dittrich_91] kritisiert an diesen Koordinationsmodellen, daß sie i.d.R. keine Realzeitbedingungen betrachten, d.h. synchrone Gruppenaktivitäten wie Meetings werden, falls überhaupt, als Teilaktionen modelliert. Er fordert deshalb einen integrierten Ansatz, in dem sich alle zu koordinierenden Faktoren widerspiegeln.

Workflow Management-Systeme folgen meist dem vorgangsorientierten Ansatz (sie werden deshalb auch als *Vorgangsbearbeitungssysteme* bezeichnet). Es gibt aber auch Entwicklungen, andere Ansätze zu integrieren, wie z.B. der ActionWorkflow-Ansatz (vgl. [Medina-Mora et al_92]).

3.3.3 Modellierung von Workflows

[Jablonski_95] leitet aus den Konzepten der Koordinationstheorie eine für die Modellierung von Workflows essentielle Erkenntnis ab:

„Workflows (...) werden nicht ausschließlich durch Aktivitäten modelliert, sondern Aspekte wie Ziele, Aktoren und Abhängigkeiten müssen auch erfaßt werden. Aufgrund dieser Beobachtung untergliedert sich ein allgemeines Referenzmodell eines Workflows in einem aktivitätenbezogenen, einen aktorenbezogenen, einen abhängigkeitsbezogenen und einen kausalen Aspekt.“ [Jablonski_95]

Ausgehend von dieser Erkenntnis muß das modellierte Schema eines Workflows sowohl Aktivitäten (*Was*), Aktoren (*Wer*), Abhängigkeiten (*Wann und unter welchen Konditionen*) und Kausalitäten (*Warum*) sowie die Beziehungen dieser Elemente untereinander abbilden. Wir gehen nachfolgend auf die einzelnen Aspekte analog zu [Jablonski_95] und [Teufel et al_95] ein.

Modellierung von Aktivitäten

Workflows können auf mehreren Abstraktionsebenen modelliert werden. Auf der Ebene mit dem größten Detaillierungsgrad besteht ein Workflow aus Aktivitäten. Werden *aktivitätenbezogene* Aspekte modelliert, so entspricht das resultierende Schema dem Was des Workflows. Einzelnen Aktivitäten werden Applikationen zugeordnet, welche zur Ausführung der Aktivität notwendig sind. Einer Aktivität muß jedoch nicht zwingend eine Applikation zugeordnet sein. Sie kann auch als Platzhalter für beispielsweise Telefonate, Sitzungen oder dergleichen dienen.

Modellierung von Aktoren

Die Modellierung aktorenbezogener Aspekte eines Workflows beantwortet die Frage, wer Aktivitäten innerhalb eines Workflows ausführt. [Jablonski_95] differenziert die Diskussion dieser Frage in zwei Teilaspekte: (1) die *Zuordnung von Aktoren zu Aktivitäten* und (2) die *Notifikation, Synchronisation und Verwaltung von Arbeitslisten von Aktoren*. Bei der Zuordnung von Aktoren zu Aktivitäten muß definiert werden, ob ein ganz bestimmter Aktor zugeordnet werden soll, oder ob zunächst ein Aktoren-Typ zugewiesen wird. Ein Aktoren-Typ wird als Rolle bezeichnet. Eine Rolle ist definiert durch eine Menge von Fähigkeiten und Kompetenzen, welche alle Aktoren aufweisen, die dieser Rolle zugehörig sind. Wird nun ein Aktor einer Aktivität zugeordnet, muß ihm dies mitgeteilt werden. Diese Mitteilung wird *Notifikation* genannt. Wird eine Rolle einer Aktivität zugeordnet, so können prinzipiell mehrere Aktoren, die diese Rolle einnehmen, informiert werden und die Ausführung der Aktivität erfolgt dann durch einen oder mehrere Aktoren. Dazu muß die Ausführung der Aktivitäten mit geeigneten Mechanismen *synchronisiert* werden. In vielen Szenarien ist es denkbar, daß einem Aktor über die Notifikation eine ganze Reihe von Aufgaben zugewiesen wird. Diese Aufgaben können über individuelle Arbeitslisten von Aktoren verwaltet werden.

Modellierung von Abhängigkeiten

Im Zusammenhang mit der Modellierung von Workflows werden grundsätzlich drei Typen von Abhängigkeiten unterschieden:

- die Ablaufkontrolle,
- der Datenfluß und
- die Historie.

Die *Ablaufkontrolle* bestimmt den Ablauf eines Workflows und definiert, *wann* und *unter welchen Konditionen* Aktivitäten innerhalb eines Workflows ausgeführt werden. Die Spezifikation der Ablaufkontrolle erfolgt *präskriptiv* oder *deskriptiv*. Präskriptive Spezifikation liegt dann vor, wenn alternative Ausführungsreihenfolgen nur über die Auswertung vorgegebener Ablaufkonditionen zugelassen sind. Dabei werden drei fundamentale Formen der Ablaufkontrolle unterschieden: die *serielle, die alternative und die parallele* Ausführung. Deskriptive Spezifikation liegt dann vor, wenn eine Menge äquivalenter Ausführungsreihenfolgen unterstützt wird. Für die Definition der Konditionen, unter denen eine Reihenfolge ausgeführt wird werden zwei grundsätzliche Typen von Bedingungen verwendet: *Zeit- und Existenzbedingungen*.

Der *Datenfluß* in einem Workflow beschreibt, welche Daten (Nutzdaten oder Kontrolldaten) innerhalb oder zwischen Workflows weiter gereicht werden. Nutzdaten sind Daten, welche beispielsweise innerhalb einer Aktivität von einem Anwendungssystem zur Verfügung gestellt werden. Kontrolldaten sind ebenfalls Nutzdaten, jedoch dienen sie aus Sicht eines WfMS ausschließlich dazu, den korrekten Ablauf des Workflows sicherzustellen.

Im Rahmen einer sog. Historienverwaltung werden Protokolle ausgeführter Workflows verwaltet. Diese Protokolle dienen der Analyse von Workflows aus unterschiedlicher Sicht (beispielsweise technische Fehleranalysen, betriebswirtschaftliche Analysen, Revisionen).

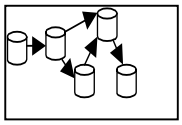
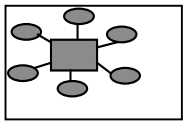
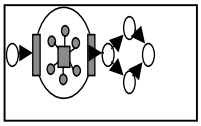
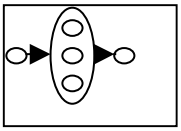
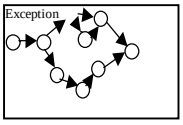
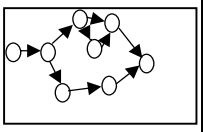
Modellierung von Kausalitäten

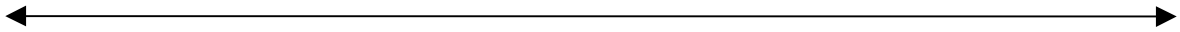
Die Modellierung kausaler Aspekte eines Workflows beantwortet die Frage nach dem Zweck – dem *Warum* – eines Workflows. Existierende WfMS erlauben die explizite Modellierung

kausaler Aspekte in der Regel nicht. Eine Folge davon ist, daß während des Ablaufes eines Workflows keine Prämissenüberprüfung durchgeführt werden kann. Das heißt, daß ein gestarteter Workflow immer zu Ende geführt wird (es sei denn, er wird manuell gestoppt oder der Stopp ist in irgendeinerweise vordefiniert). In vielen Fällen kann es jedoch erforderlich sein, nach bestimmten Abschnitten im Workflow eine Prämissenüberprüfung durchzuführen, welche unter Umständen zum Abbruch des Workflows führt. Diese Überprüfung kann jedoch nur dann erfolgen, wenn die Prämissen zuvor (explizit oder implizit) in Form von Kausalitäten definiert wurden.

3.3.4 Kategorien von Workflows

Welche Fähigkeiten ein WfMS aufweisen kann, bzw. welche Kategorien von Workflows es gibt, wird nachfolgend durch Darstellung des *Workflow Koninuums* (vgl. [Hilpert_94] / [Nastansky_96]) verdeutlicht (s. Abbildung 11).

Ad hoc Workflow	**Task Force**	**Semi – strukturierter Workflow**			**Standard Workflow**
E-mail basiert	Nicht vorherbe-stimmter Workflow	a) Vorherbestimmtes Team und nicht vorherbestimmte Person	b) Vorherbestimmtes Team und Anzahl von Person(en)	c) Ad hoc Ausnahmen	Fest strukturierter Workflow
"Store and forward"-System, kein gemeinsamer Datenzugriff 	Gemeinsamer Datenzugriff, Einzelschritte 	Kombination von definierter Aufgabe und nicht vorherbestimmter Person 	Kombination von definierter Aufgabe und vorherbestimmter Anzahl von Personen 	Strukturierter Workflow mit Ausnahmebehandlung 	Fest vorherbestimmte Workflowschritte 
Spontane (Re-)aktion	Diskussionsorientierte Bearbeitung im Team	Eine verantwortliche Person weist Aufgaben innerhalb des Teams zu und definiert den Abschluß der Bearbeitung	Die Anzahl der Teammitglieder für die Bearbeitung einer Aufgabe ist vorherbestimmt; die Bearbeitungsreihenfolge ist variabel	Einfach durchzuführende flexible Veränderung des Workflows	Fest vorherbestimmter und strukturierter Workflow
z.B. spontanes Sammeln von Informationen	z.B. Gemeinsames Verfassen von Publikationen	z.B. Gemeinsames Erstellen eines Jahresberichtes	z.B. Mitzeichnungsverfahren	z.B. Konsumentenkredit mit besonderen Kundenwünschen	z.B. Konsumentenkredit



Flexibel, veränderlich, einmalig — vorgegeben, strukturiert, wiederkehrend

Abbildung 11: Workflow-Kontinuum nach [Nastansky_96]

Das Workflow Kontinuum umfaßt einmalige, kurzlebige Prozesse (*Ad hoc Workflows*), selbständig agierende Arbeitsgruppen (*Task Forces*), teilstrukturierte Vorgänge (*Semi-strukturierte Workflows*) und fest strukturierte, vordefinierte Abläufe (*Standard Workflows*).

Ad hoc Workflows zeichnen sich dadurch aus, daß ihr Ablauf i.a. nicht vorherbestimmt werden kann und somit eine Strukturierung nicht möglich ist. Dies führt dazu, daß Ad hoc Workflows generell als nicht ‚automationswürdig' angesehen werden, da sich der Aufwand einer eingehenden Untersuchung des Prozeß-Istzustandes und die Erstellung und Implementierung einer Ablaufspezifikation nicht lohnen würde. Allerdings können in Ad hoc

Workflows auch wiederkehrende Teilabläufe identifiziert werden, die sich meist auf das automatische Weiterleiten von Informationen beziehen. Somit bietet es sich an, zur Unterstützung dieser Art von Workflows Systeme mit E-mail-Funktionalität zu verwenden.

Task Forces zeichnen sich durch ihre Eigenverantwortlichkeit aus. Sie bestehen aus einer bekannten Zahl von Teammitgliedern mit unterschiedlichen Qualifikationen / Zuständigkeiten und werden mit der Lösung von Aufgaben betreut, deren eindeutige Bearbeitungsreihenfolge allerdings nicht bekannt ist und dem Team überlassen wird. Somit ist eine strenge Vorgabe der Arbeitsabläufe durch ein Workflow-Modell ebenfalls nicht angebracht. Meist koordinieren sich die Teammitglieder mit Hilfe von Informationen über den momentanen Bearbeitungszustand der Aufgaben, welcher z.B. aus einem Shared Workplace oder einem Bulletin Board ersichtlich ist. Eine Workflow-Management-Anwendung könnte allerdings zur Koordination der gemeinsamen Datenhaltung eingesetzt werden.

Im Bereich der **semi-strukturierten Workflows** gibt es mehrere Ausprägungsmöglichkeiten. Als Gemeinsamkeit läßt sich die Kombination bzw. Integration von s*trukturiert*en und *unstrukturierten* Teilabläufen erkennen. Somit kommt diese Ausprägung von Workflows sehr nahe an den tatsächlichen Aufbau heutiger Geschäftsprozesse heran. Zu beachten ist, daß bei den Gruppenprozessen auf jeden Fall ein koordinierender Verantwortlicher bestimmt werden muß, damit ein gewisses Maß an Überwachung möglich ist.

Die semi-strukturierten Workflows werden in *offene Teambearbeitung, kontrollierte Teambearbeitung* und *ad hoc-Ausnahmen* vorgegebener Standard-Workflows unterteilt.

Eine offene Teambearbeitung (a), bei der das Team vorherbestimmt, die spezifischen Aufgabenträger jedoch nicht vorgegeben werden, kann in einen sonst vordefinierten Vorgang integriert werden. Dies enthebt den Organisator davon, Teilprozesse innerhalb eines Geamtvorgangs bis ins Detail planen zu müssen, für die sich ein solcher Aufwand nicht lohnt.

Demgegenüber gilt als eine Art etwas stärker vordefinierter Aufgabenzuweisung die kontrollierte Teambearbeitung bzw. Mitzeichnung (b). Dabei wird zwar festgelegt, daß eine bestimmte Anzahl von Gruppenmitgliedern eine Aufgabe innerhalb eines Vorgangsschritts bearbeiten muß. Welche Mitarbeiter dies jeweils im konkreten Einzelfall sind und in welcher Reihenfolge deren Bearbeitung erfolgt, bleibt der Absprache innerhalb der Arbeitsgruppe überlassen.

Ad hoc Ausnahmen (c) beziehen sich auf die Möglichkeit der Prozeßteilnehmer, zur Laufzeit mit anderen Teilnehmern in Kontakt treten zu können, um z.B. Rücksprachen zu führen oder Informationen zur Prozeßausführung zu erhalten, Rücksprünge oder Umleitungen im Prozeßablauf vornehmen zu können oder den Workflow-Typ zu wechseln. Es ist wichtig, daß diese Ausnahmebehandlungen zur Laufzeit möglich sind, denn nur so ist eine flexible Reaktion auf Ausnahmefälle in täglichen Arbeitsabläufen realisierbar.

Standard Workflows werden durch ihre starke Strukturierung geprägt. Gleichförmige Arbeitsvorgänge werden wiederholt durchlaufen, wobei es sich vorwiegend um Routinearbeit handelt. Somit ist eine eingehende Analyse dieser Prozesse und eine automatische Ausführung lohnenswert, da sie auch als Teilprozesse in anderen Abläufen Verwendung finden können.

Es ist wünschenswert, daß Systeme, die die Modellierung und Ausführung von Workflows unterstützen, die Berücksichtigung aller eben vorgestellten Kategorien gerecht werden.

3.3.5 Diskussion der Workflow-Kategorien im Anwendungsbeispiel

Wir untersuchen nun, welche Kategorien von Workflows im Anwendungsbeispiel vorkommen. Als Grundlage dient das Workflow-Kontinuum (vgl. Kap. 3.3.4).

Der Prozeß der FB-Lehrplanung kann als semi-strukturierter Workflow betrachtet werden, der folgende Workflow-Kategorien enthält.

a) Die *fest strukturierten, standardisierten Prozesse* wie z.B. die Erstellung von Lehrberechtigungsanträgen sind typische **Standard Workflows**. Die Koordination erfolgt im Anwendungsbeispiel vor allem durch Dokumentenaustausch.

b) Während der AB-Lehrplanung findet zwischen den Beteiligten eine *verständigungsorientierte Koordination* statt, wo die Beteiligten sich im Prozeß der *Selbstabstimmung* koordinieren. Das Ziel der AB-Lehrplanungssitzung ist die Aushandlung des Lehrplanes. Das Erreichen dieses Zieles erfolgt ohne Vorgaben in Eigenverantwortlichkeit der Sitzungsteilnehmer. Dies entspricht der Charakteristik einer **Task Force**, bzw. einer **offenen Teambearbeitung**, da dieser Prozeß in den sonst vordefinierten Geamtprozeß der FB-Lehrplanung einzuordnen ist. Im Anwendungsbeispiel sollte die Einrichtung eines gemeinsamen Informationsraumes ermöglicht werden, so daß z.B. eine gemeinsame Dokumentenablage realisiert werden kann.

c) Der FB-Lehrplaner muß zur Koordination der aus seiner Sicht sternförmig ablaufenden AB-Lehrplanungen die Möglichkeit haben, flexibel auf Ausnahmesituationen zu reagieren. Ein WfMS sollte dem FB-Lehrplaner also zunächst die Möglichkeit bieten, identische Prozesse sternförmig zu initiieren und zu koordinieren. Insbesondere müssen dabei **ad hoc-Ausnahmen** unterstützt werden. Er kann z.B. gezwungen sein, den Lehrplan eines Arbeitsbereichs zurückzusenden, wenn Unstimmigkeiten auftreten, oder einen Arbeitsbereich zum Einhalten von Zeitgrenzen zu ermahnen.

Für eine Realisierung des Anwendungsbeispiels muß also ein WfMS folgende Workflow-Kategorien unterstützen:

- Standard-Workflows,
- Task Force bzw. offene Teambearbeitung sowie
- Ad hoc-Ausnahmen

4 Modellierungskonzepte und Architektur von FlowMark

Dieses Kapitel beschäftigt sich mit dem WfMS FlowMark (Version 2.3) von IBM. Um einen Einblick in die FlowMark zugrundeliegenden Konzepte zu erhalten, werden in Kapitel 4.1 die Modellierungsmöglichkeiten des Systems vorgestellt. Hierbei wird dargestellt, wie die Modellierung der Aufbau- und Ablauforganisation aussieht und gesondert auf die Prozeßdefinitionssprache von FlowMark eingegangen. In Abschnitt 4.2 wird aufgezeigt, wie die Architektur eines WfMS aussehen sollte und welche Anforderungen an sie gestellt wird. Abschnitt 4.3 wird dann als Beispiel für eine konkrete Architektur die Architektur von FlowMark vorstellen, wobei auch die von FlowMark zur Verfügung gestellten Hilfsmittel Berücksichtigung finden.

4.1 Konzepte der Workflow-Modellierung in FlowMark

Unter einem Workflow-Modell[2] versteht man nach IBM FlowMark-Terminologie: „A complete representation of a process. It consists of the process diagram and settings and the definitions of staff, programs, and data structures associated with the activities of the process.“ [IBM_96a, S. 222]
Im folgenden werden zunächst die Möglichkeiten der Modellierung der Aufbauorganisation (vgl. Kapitel 1) und danach Bausteine und Konstruktionsprinzipien vorgestellt, die FlowMark zur Definition von Workflows anbietet. Die Ausführungen in 4.1.1 und 4.1.2 orientieren sich vorwiegend an [IBM_96a].

4.1.1 Modellierung der Aufbauorganisation

Die Modellierung der Aufbauorganisation wird als **Staff-Definition** bezeichnet. Staff wird definiert als: „The people and their roles, organizations, and levels as defined in a FlowMark database.“ [IBM_96a, S. 221] Es ist möglich, Personen, Rollen, Organisationen und Ebenen zu definieren, deren Bedeutung nachfolgend erklärt wird.

Die kleinsten Einheiten der Aufbauorganisation eines Unternehmens stellen in FlowMark die **Personen** dar. Für jeden Angestellten, der mit FlowMark arbeiten soll, muß ein Personen-Objekt angelegt werden, um so FlowMark über dessen Existenz in Kenntnis zu setzen. Jeder Person muß eine User ID (zur eindeutigen Identifikation) und eine Ebene (siehe weiter unten) zugeordnet werden. Optional kann ein Paßwort, Name und Telefonnummer angegeben werden. Falls die Person zur Zeit nicht arbeitet, gibt es die Möglichkeit sie als abwesend zu kennzeichnen. Des weiteren kann man Berechtigungen festlegen, die sich darauf beziehen, ob die Person Workflows modellieren darf, welche Prozesse sie starten darf, ob sie andere Personen neu in das System eintragen darf und welche anderen Personen auf die Arbeitsliste der gerade modellierten Person zugreifen dürfen. Für jede Person kann genau ein Stellvertreter definiert werden, dem die Aufgaben bei Abwesenheit zugeordnet werden. Eine Person kann Stellvertreter für beliebig viele weitere Personen sein. Bei Abwesenheit einer Person und ihres Stellvertreters wird die Aufgabe dessen Stellvertreter zugeordnet, d.h. die Stellvertreterregelung ist transitiv.

Personen können sowohl **Rollen** (vgl. Kapitel 3.3.3) als auch Organisationen zugeordnet werden wobei beachtet werden muß, daß eine Person nur genau einer Organisation (siehe weiter unten) zugeordnet werden kann.

---

[2] Der Begriff Workflow-Modell findet in FlowMark-Terminologie synonym zum Begriff Prozeß-Modell Verwendung.

In FlowMark kann eine Person Träger mehrerer Rollen sein, wobei verschiedene Personen die gleichen Rollen innehaben können. Einer Rolle können mehrere Mitglieder zugeordnet werden. In FlowMark gibt es drei vordefinierte Rollen: Administrator, Manager und Koordinator. Die Rolle des Administrators verschafft dem Träger alle Rechte innerhalb von FlowMark, wobei mindestens eine verwaltete Person Inhaber dieser Rolle sein muß. Einer Rolle kann ein Koordinator zugeordnet werden. Dies ist z.B. sinnvoll, wenn man autonome Arbeitsgruppen bilden möchte und dem Inhaber der Koordinatorrolle das Recht zuordnet, auf die Arbeitslisten der Gruppenmitglieder zuzugreifen. Dann kann dieser die Verteilung der Aufgaben an die einzelnen Gruppenmitglieder übernehmen. Die Rolle des Managers wird im nächsten Absatz geklärt.

Organisationen stellen die obersten Strukturierungseinheiten der Aufbauorganisation dar. Sie können z.B. die Abteilungen eines Unternehmens repräsentieren. Organisationen werden in FlowMark hierarchisch organisiert. Jede Organisation kann höchstens eine „Vaterorganisation“ haben, aber beliebig viele „Kindorganisationen“. Jede Organisation muß genau einen Manager haben, der diese Organisation leiten kann und auch Mitglied dieser Organisation ist. Wenn eine neue Organisation erzeugt wird, so ordnet FlowMark dieser Organisation automatisch einen Manager zu. Falls keine Person existiert, die noch keiner Organisation zugeordnet worden ist, erzeugt FlowMark einen „Default-User“ und ordnet ihm sowohl die Organisation als auch die Rolle des Managers dieser Organisation zu. Andernfalls wird die letzte neu erzeugte Person zugeordnet. Zu einer Organisation können beliebig viele Mitglieder mit unterschiedlichen Rollen gehören, mindestens jedoch ein Mitglied (Manager).

Ebenen stellen eine weitere Möglichkeit dar, Personen hinsichtlich ihrer Qualifikation voneinander zu unterscheiden. Es werden zehn Ebenen (0 – 9) unterschieden, deren Bezeichnungen frei festgelegt werden können. Die geringste Priorität hat Ebene null, die automatisch jeder Person zugeordnet wird. So könnte man den Personen mit der meisten Berufserfahrung eine höhere Ebene zuweisen als den Personen mit weniger Berufserfahrung und dies bei der Zuweisung von Aufgaben berücksichtigen. Zusätzliche Ebenen können nicht definiert werden.

4.1.2 Modellierung der Ablauforganisation

Unter dem Begriff der Ablauforganisation versteht man laut [Brockhaus_93a] die „Gestaltung der zur Aufgabenerfüllung notwendigen Arbeitsprozesse in Unternehmen, Behörden, Verwaltungen. Ziel der Ablauforganisation im Gegensatz zur Aufbauorganisation ist es, alle Arbeitselemente (Arbeitsinhalte, Arbeitsmittel, Personal, u.a.) so anzuordnen, daß die Arbeitsgänge zeitlich und räumlich aufeinander abgestimmt sind. Damit soll eine bestmögliche Auslastung aller Stellen erreicht werden.“

Bei der Festlegung der Ablauforganisation geht es also darum, das Zusammenspiel der Einheiten der Aufbauorganisation (vgl. Kapitel 1) zu bestimmen, zu optimieren und in einen kausalen Zusammenhang zu stellen. Die Teile der Arbeits- bzw. Geschäftsprozesse (vgl. Kapitel 1), deren Ausführung durch ein WfMS unterstützt werden soll, müssen dem WfMS durch ein Workflow-Modell bekanntgemacht werden. Welche Möglichkeiten der Modellierung der Ablauforganisation von FlowMark angeboten werden, wird im folgenden dargestellt.

Prozesse[3] können in FlowMark auf zwei Arten definiert werden. Zum einen unter Verwendung eines grafischen Editors und zum anderen unter Benutzung der FlowMark Definition Language (FDL). Auf die FDL wird später näher eingegangen.

Der grafische Editor ermöglicht es dem Benutzer, ohne Kenntnisse der FDL, Prozesse unter Verwendung von gewichteten, gefärbten, azyklischen Graphen zu definieren, welche in [Leymann et al_94] als PM-Graphen (Process Model Graphs) bezeichnet werden. Hierbei werden Aktivitäten durch Knoten repräsentiert, die durch Kontroll- und Datenfluß-Konnektoren miteinander verbunden sind. Die zur Modellierung der Prozesse verfügbaren Bausteine sind in Abbildung 12 dargestellt und ihre Semantik nachfolgend erklärt.

Wie ein Prozeß-Modell konkret aussehen kann, wird am Ende dieses Abschnitts anhand eines Beispielprozesses dargestellt. Außerdem finden sich in Kapitel 5, bei der Vorstellung der Umsetzung des Anwendungsbeispiels aus Kapitel 2, weitere Prozeß-Modelle.

Das sogenannte 'Top Level'-Element eines FlowMark Workflow-Modells ist der **Prozeß** (engl. Process). Ein Prozeß wird als eine Folge von **Aktivitäten** (engl. Activity) verstanden, die ausgeführt werden, um eine bestimmte Aufgabe zu erfüllen. Somit repräsentiert eine Aktivität einen Schritt innerhalb eines Prozesses. Um eine Einschränkung hinsichtlich des Personenkreises, der einen Prozeß starten darf, vornehmen zu können, ist es möglich, Prozesse sogenannten **Kategorien** zuzuordnen.

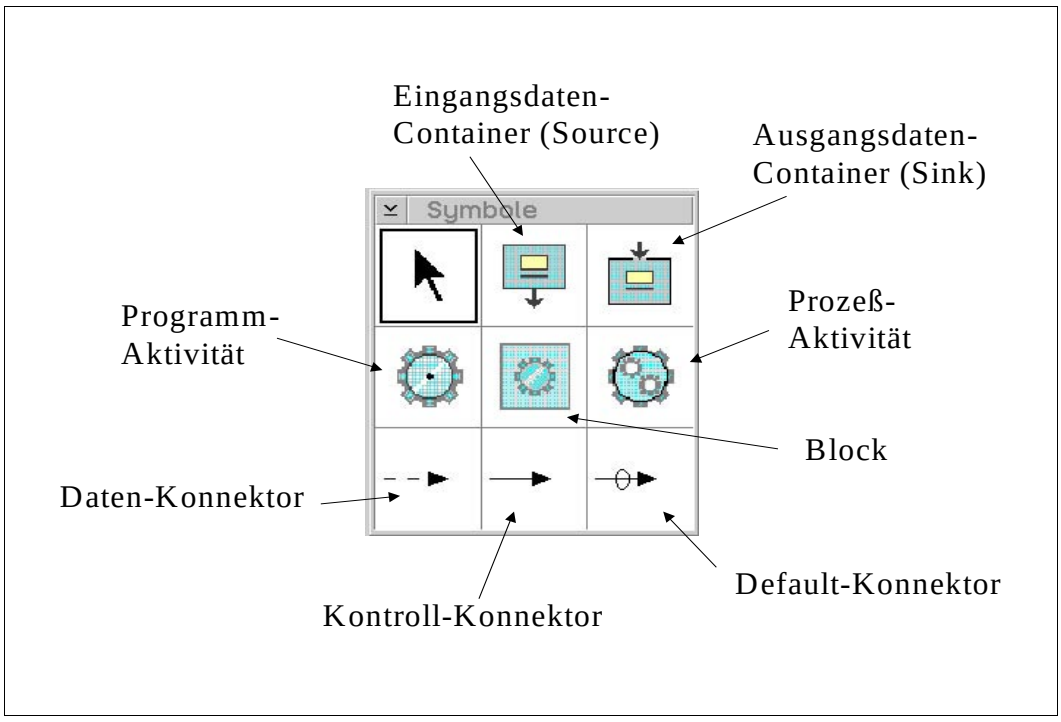


Abbildung 12: Modellierungsbausteine von FlowMark

FlowMark unterscheidet zwei Arten von Aktivitäten: **Programm-Aktivitäten** (engl. Program-Activity) und **Prozeß-Aktivitäten** (engl. Process-Activity). Einer Programm-Aktivität ist ein Programm zugeordnet, welches aufgerufen wird, wenn die Aktivität startet. Das Programm wird entweder automatisch ausgeführt oder muß explizit vom Benutzer gestartet werden (manuelle Ausführung). Programm-Aktivitäten werden auch als atomare Workflows

[3] Es sei angemerkt, daß die FlowMark-Terminologie nicht von Arbeits-, Geschäftsprozessen oder Ablauforganisation spricht, sondern nur von Prozessen. Dies wird im weiteren Verlauf der Arbeit übernommen.

bezeichnet (vgl. z.B. [Weske et al_98]). FlowMark müssen die Programmnamen bekannt sein, der Ort der Programme (Netzwerkadresse und Verzeichnis) und unter welchem Betriebssystem sie laufen. Der Unterschied zu Prozeß-Aktivitäten liegt darin, daß diesen keine Programme sondern Prozesse zugeordnet werden, die aktiviert werden, sobald die Aktivität startet. Prozeß-Aktivitäten können wiederum aus Programm-Aktivitäten und Prozeß-Aktivitäten bestehen. Man spricht in diesem Zusammenhang von **Sub-Prozessen** (engl. Sub-Processes) oder laut [Weske et al_98] auch von komplexen Workflows. Durch die Option, diese Sub-Prozesse auch in anderen Prozessen verwenden zu können, wird die Wiederverwendung und eine Hierarchisierung des Workflow-Modells ermöglicht. Sub-Prozesse werden bei der Modellierung nicht gesondert gekennzeichnet. Somit kann jeder beliebige Prozeß als Sub-Prozeß in anderen Prozessen Verwendung finden, wobei ein Sub-Prozeß innerhalb eines anderen Prozesses auch mehrmals vorkommen kann. Einer Aktivität werden diverse Informationen zugeordnet. Diese Informationen beziehen sich beispielsweise auf Bedingungen, die erfüllt sein müssen, damit die Aktivität aufgerufen (Startbedingung) bzw. beendet (Endebedingung) werden kann, oder in welchem Zeitraum die Aktivität abgearbeitet werden muß und wer für die Abarbeitung zuständig ist (weiteres in [IBM_96a, S. 4]).

Überall dort, wo eine Aktivität innerhalb einer Prozeßdefinition stehen kann, kann auch ein **Block** stehen. Ein Block stellt ein Modellierungskonstrukt dar, mit dessen Hilfe man mehrere zusammengehörige Aktivitäten gruppieren kann. Dadurch wird es möglich, die Komplexität einer Prozeßdefinition zu reduzieren, denn der Block ersetzt die Aktivitäten-Gruppe innerhalb der Definition. Am Prozeßablauf ändert die Verwendung dieses Konstruktes nichts, denn bei Aufruf des Blocks wird dieser an die Aktivitäten-Gruppe weitergeleitet. Man kann die Aktivitäten-Gruppe als Sub-Prozeß auffassen, mit dem Unterschied, daß er in diesem Fall nicht benannt und die Wiederverwendung in anderen Prozessen nicht möglich ist.

Durch die Konzepte des Blocks und der Prozeß-Aktivität wird das sogenannte **rekursive Konstruktionsprinzip** ermöglicht (vgl. [Jablonski et al_97a, S. 109]). Dies bedeutet, daß man die Prozesse in FlowMark nicht nur horizontal auf einer Modellierungsebene sondern auch vertikal über mehrere Modellierungsebenen hinweg entwerfen kann (siehe Abbildung 13). Horizontale Modellierung bezeichnet die Anordnung der Modellierungsbausteine auf einer Ebene, während die vertikale Modellierung eine schichtenartige (hierarchische) Anordnung meint. Da innerhalb eines Sub-Prozesses oder eines Blocks wiederum Prozeß-Aktivitäten oder Blöcke verwendet werden können, ist eine beliebig tiefe Schachtelung möglich.

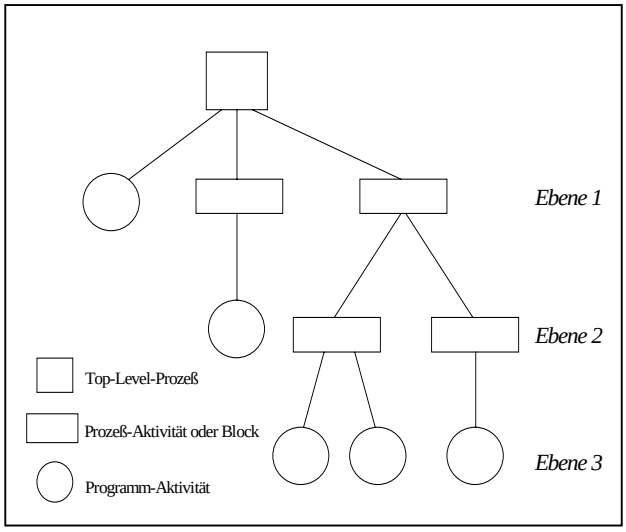


Abbildung 13: Hierarchische Modellierung (vgl. [Jablonski et al_97a, S. 110])

Eine weitere Strukturierungsmöglichkeit, die durch den Block zur Verfügung gestellt wird, ist die Modellierung von **Schleifen**. Hierzu gibt man zu einem Block eine Abbruchbedingung an, die erfüllt sein muß, damit der Block beendet werden kann. Ist sie nicht erfüllt, so wird der

Block bzw. die Aktivitäten-Gruppe innerhalb des Block solange wiederholt, bis die Abbruchbedingung wahr wird. Falls der Block wiederholt wird, können die in vorherigen Durchläufen produzierten Daten als Eingabedaten wiederverwendet werden. Falls die Wiederverwendung nicht erwünscht ist, werden bei erneutem Durchlauf die Eingangsdaten des ersten Durchlaufs verwendet. Das von FlowMark verwendete Schleifenkonzept entspricht dem Konzept der „repeat...until-Schleifen“, welches aus herkömmlichen Programmiersprachen bekannt ist.

Mit Hilfe eines Blocks ist auch die Definition von **Bündeln** (engl. Bundle) möglich. Unter einem Bündel wird eine spezielle Art eines Blocks verstanden, der die Erzeugung mehrerer Exemplare einer Programm- oder Prozeß-Aktivität zur Laufzeit ermöglicht. Die Aktivität, die mehrfach erzeugt wird, wird als **Muster-Aktivität** (engl. Pattern-Activity) bezeichnet. Die erzeugten Exemplare bezeichnet man als **Bündel-Aktivitäten** (engl. Bundle-Activity). Die Anzahl der zu erzeugenden Exemplare wird zur Laufzeit durch ein spezielles Programm, welches einer als **Planungs-Aktivität** (engl. Planning-Activity) bezeichneten Programm-Aktivität zugeordnet ist, bestimmt. Das Programm muß vor dem Bündel ausgeführt werden. Dem die Planungs-Aktivität implementierendem Programm muß bekannt gemacht werden, für welches Bündel es zuständig ist. Das an dieser Stelle zu verwendende Programm wird von FlowMark zur Verfügung gestellt[4]. Wieviele Exemplare erzeugt werden, kann auf zwei Arten bestimmt werden. Man unterscheidet zwischen der **personengetriebenen Erzeugung** (engl. person driven instanciation) und der **datengetriebenen Erzeugung** (engl. data driven instanciation). Bei der personengetriebenen Erzeugung ist es möglich, entweder direkt die Personen (mittels User ID) anzugeben, denen ein Exemplar zugewiesen werden soll (statische Zuweisung), oder aber eine Rolle anzugeben (dynamische Zuweisung). Die Planungs-Aktivität erzeugt dann so viele Exemplare wie Träger der Rolle vorhanden sind, falls keine weiteren Einschränkungen vorgenommen werden. Unter Einschränkungen fällt z.B. die Beschränkung auf eine bestimmte Organisation (Abteilung). Somit würden nur so viele Exemplare erzeugt werden, wie Angehörige der Abteilung bekannt sind, die den Auswahlkriterien genügen.

Bei der datengetriebenen Erzeugung spielt ein Array eine besondere Rolle. Eine Variable mit dem Namen **BundleInstanciatingArray** muß innerhalb des Eingangsdaten-Containers (siehe weiter unten) der Programm-Aktivität zu finden sein, da dies vom Planungs-Programm angenommen wird (falls ein anderer Name gewählt wird, muß dieser dem Programm beim Aufruf bekanntgemacht werden). Ja nachdem, wieviele Elemente dieses Arrays belegt sind, wird von der Programm-Aktivität eine entsprechende Anzahl von Exemplaren erzeugt und den der Aktivität zugeordneten Bearbeitern zugewiesen. Stehen weniger Bearbeiter zur Verfügung als Exemplare erzeugt werden, werden einigen Bearbeitern mehrere Exemplare zugeordnet. Hierfür gibt es verschiedene Verteilungsalgorithmen (siehe hierzu [IBM_96a, S. 161]).

Der **Kontrollfluß** bestimmt, in welcher Reihenfolge die Aktivitäten innerhalb eines Prozesses ausgeführt werden. Es ist möglich zu jeder Aktivität Eingangsbedingungen und Endebedingungen anzugeben, die erfüllt sein müssen, damit eine Aktivität begonnen oder beendet werden kann. Bei den Bedingungen handelt es sich um boolesche Funktionen, die auf Teilmengen der Ein- bzw. Ausgangsdaten-Containern operieren, die jeder Aktivität zugeordnet sind (siehe weiter unten).

[4] z.B. das Programm „exmbobcl.exe“ unter Windows NT

Aktivitäten werden mittels **Konnektoren** verbunden. FlowMark unterscheidet drei Arten von Konnektoren: **Kontroll-**, **Default-** und **Daten-Konnektoren**. Mit Hilfe von **Kontroll-Konnektoren** wird der Kontrollfluß festgelegt. Jeder Aktivität können mehrere Kontroll-Konnektoren zugeordnet werden. Kontroll-Konnektoren verbinden nur Aktivitäten einer Modellierungsebene. Es ist z.B. nicht möglich, eine Aktivität innerhalb eines Blocks mittels Kontroll-Konnektor mit einer Aktivität eines anderen Blocks zu verbinden. Dadurch wird die Modularität erhöht (vgl. [Alonso et al_97b, S. 7]). Zu jedem Kontroll-Konnektor kann eine Transitionsbedingung angegeben werden, die wahr sein muß, damit der jeweilige Pfad während der Prozeßausführung genommen wird. Bei Transitionsbedingungen handelt es sich ebenfalls um boolesche Funktionen, die auf einer Teilmenge der Daten des Ausgangsdaten-Containers operieren (siehe weiter unten). Falls eine Transitionsbedingung zu „falsch“ evaluiert wird, so wird mittels eines als **„dead path elimination“** bezeichneten Verfahrens dafür gesorgt, daß die mit dem Kontroll-Konnektor verbundene Aktivität und alle nachfolgend abhängigen als beendet betrachtet werden. Hierdurch wird verhindert, daß Folgeaktivitäten auf Ergebnisse vorhergehender Aktivitäten warten könnten, obwohl diese nie produziert werden (vgl. [Leymann et al_94]).

Der **Default-Konnektor** bestimmt den Kontrollfluß, wenn keine andere Transitionsbedingung nach Beendigung der entsprechenden Aktivität erfüllt werden konnte. Mittels Default-Konnektor können Ausnahmebehandlungen realisiert werden.

Daten-Konnektoren bestimmen den Datenfluß innerhalb eines Workflow-Modells. Jeder Aktivität können mehrere Eingangsdaten zugeordnet werden und die Ausgangsdaten können an verschiedene Nachfolgeaktivitäten übergeben werden, wobei dies nur möglich ist, wenn die Nachfolgeaktivität über einen Kontroll-Pfad (Folge von Kontroll-Konnektoren) erreichbar ist. Wenn dies nicht so wäre, würde das Verfahren der „dead path elimination“ nicht seinen Zweck erfüllen können, da es auf Kontroll-Konnektoren angewandt wird.

Der Speicherplatz, der für Eingangs- und Ausgangsdaten von Prozessen als auch Aktivitäten zur Verfügung gestellt wird, bezeichnet man als **Daten-Container.** Den Daten-Container für Eingangsdaten von Prozessen und Blöcken nennt man **Source**, den Daten-Container für Ausgangsdaten **Sink**. Die Daten-Container der Prozesse und Blöcke können von jeder Aktivität, die mittels eines Daten-Konnektors mit ihnen verbunden sind, gelesen (Source) und geschrieben (Sink) werden (globaler Speicherbereich). Die Daten-Container der Aktivitäten sind lokal zu den Aktivitäten. Ausgangsdaten können von Nachfolgeaktivitäten, zu denen ein Daten-Konnektor und ein Kontroll-Konnektor existiert, gelesen werden. Dadurch wird erreicht, daß Aktivitäten nicht implizit auf einem gemeinsamen Speicher operieren (außer Source und Sink). Dies muß durch die Daten-Konnektoren expliziert werden, wobei durch die Kontroll-Konnektoren eine kontrollierte Reihenfolge des Zugriffs sichergestellt wird. In Verbindung mit dem Mechanismus der „dead path elemination“ wird somit auch die Erfüllung des Bernstein-Kriteriums sichergestellt (vgl. [Leymann et al_94], [Baer_73]).

Die Eingangs- und Ausgangsdaten der Prozesse, Blöcke und Aktivitäten stellen **Datenstrukturen** dar. In FlowMark ist eine **Datenstruktur** eine geordnete Liste von Variablen (**Member** genannt), die einen Namen und einen Datentyp haben. Jede Datenstruktur kann maximal aus 256 sogenannten **Member Items** bestehen. Die von FlowMark angebotenen Datentypen sind **STRING, LONG, FLOAT** und **ARRAY**. FlowMark unterstützt eindimensionale Arrays, wobei die Größe auf 256 Elemente beschränkt ist. Unterschieden wird zwischen einer **default Datenstruktur** und **benutzerdefinierten Datenstrukturen**. Bei der default Datenstruktur handelt es sich um eine leere Datenstruktur,

die von FlowMark jedem Prozeß, jeder Aktivität und jedem Block zugeordnet wird und mit einer beliebigen benutzerdefinierten Datenstruktur überschrieben werden kann. Innerhalb jeder Datenstruktur finden sich die **vordefinierte Datenstruktur Member** _RC, _ACTIVITY, _PROCESS, _ACTIVITY_INFO und _PROCESS_INFO. Bei den Membern _RC, _ACTIVITY und _PROCESS, die als „fixed members“ bezeichnet werden, handelt es sich um Variablen, die von FlowMark gesetzt werden. _RC enthält den Return Code eines einer Programm-Aktivität zugeordneten Programms, _PROCESS und _ACTIVITY die Namen des Prozesses und der jeweiligen Aktivität. _PROCESS_INFO und _ACTIVITY_INFO enthalten die Einstellungen, die während der Modellierung vorgenommen wurden (z.B. Laufzeit des Prozesses/Aktivität, Rollenzuordnung, Organisationszuordnung, ...). Die Werte dieser vordefinierten Datenstrukturen können zur Laufzeit durch Nutzung von API-Funktionen (API = Application Programming Interface) überschrieben werden (vgl. hierzu Kapitel 4.3.4). Das überschreiben der „fixed members“ ist dagegen nicht möglich. Ihre Werte können lediglich gelesen werden. (vgl. [IBM_96c, S. 408 ff.])

Sollen Daten zwischen Aktivitäten ausgetauscht werden, so tritt hier das Verfahren des **Mapping** in kraft. Daten können nur von dem Ausgangsdaten-Container einer Aktivität zu dem Eingangsdaten-Container einer Nachfolgeaktivität übertragen werden. Hierbei muß darauf geachtet werden, daß die Datenstrukturen innerhalb der beteiligten Daten-Container den gleichen Typ aufweisen. Ist dies nicht der Fall, so muß explizit festgelegt werden, welcher Member der Datenstruktur des Ausgangsdaten-Containers welchem Member der Datenstruktur des Eingangsdaten-Containers zugeordnet wird, wobei die Member ebenfalls vom gleichen Typ sein müssen[5]. Handelt es sich um Datenstrukturen des gleichen Typs, so sorgt FlowMark durch automatisches Mapping für die richtige Zuordnung. Es ist zu beachten, daß Daten des Eingangsdaten-Containers einer Aktivität nicht automatisch in den Ausgangsdaten-Container der gleichen Aktivität übertragen werden, auch wenn die Datenstrukturen typmäßig übereinstimmen. Ausgangsdaten einer Aktivität müssen explizit gesetzt werden (durch Nutzung der APIs; vgl. Kapitel 4.3.2). Werden einer Aktivität Daten von mehr als einer Vorgängeraktivität zugeordnet, so ist ein explizites Mapping erforderlich, da in diesem Fall nicht automatisch entschieden werden kann, welches die korrekte Zuordnung der Daten ist. Dies gilt auch, wenn von mehreren Aktivitäten Ergebnisse auf den globalen Speicherbereich (Sink) geschrieben werden sollen.

Aktivitäten und Prozesse, die manuell (also durch tatsächliche Personen) auszuführen sind, müssen Rollen oder Personen zugeordnet werden. In der FlowMark-Terminologie wird diese Zuweisung als **Staff-Assignment** bezeichnet. Man unterscheidet in diesem Zusammenhang zwischen der dynamischen und der statischen Zuweisung von Personen zu Prozessen bzw. Aktivitäten. Wird erst zur Laufzeit bestimmt, welche Person eine bestimmte Aufgabe auszuführen hat, so spricht man von dynamischer Zuweisung. Hierbei wird jedem Inhaber der Rolle, die dem Prozeß bzw. der Aktivität zugeordnet ist, zur Laufzeit die Aufforderung zur Ausführung dynamisch durch Rollenauflösung zugewiesen. Falls Personen (mittels Angabe der User ID) statisch einer Aufgabe zugeordnet werden, so erhalten nur diese zur Laufzeit die Aufforderung zu deren Ausführung.

Zu guter Letzt sei an dieser Stelle noch ein (fiktiver) Beispielprozeß angegeben (Abbildung 14), um dem Leser eine Vorstellung darüber zu vermitteln, wie ein Prozeß-Modell in FlowMark aussehen könnte. Bei diesem Prozeß repräsentiert Block_A eine Schleife, die

[5] Eine explizite Festlegung der Daten-Zuordnung ist durch Doppelklick auf den entsprechenden Datenfluß-Konnektor zwischen den betroffenen Aktivitäten innerhalb des gerade modellierten Prozeß-Modells möglich.

abbricht, sobald der Member „Erledigt“ des Ausgabedaten-Containers des Blocks den Wert „ja“ enthält. Prog_C stellt die Planungs-Aktivität für Bundle_A dar. Proz_A dient der Ausnahmebehandlung, wenn die Transitionsbedingung „Summe > 20000“ zwischen Prog_A und Prog_C nicht erfüllt werden konnte. Der gesamte Prozeß gilt als beendet, sobald Proz_B und Block_A in den Zustand beendet gewechselt sind.

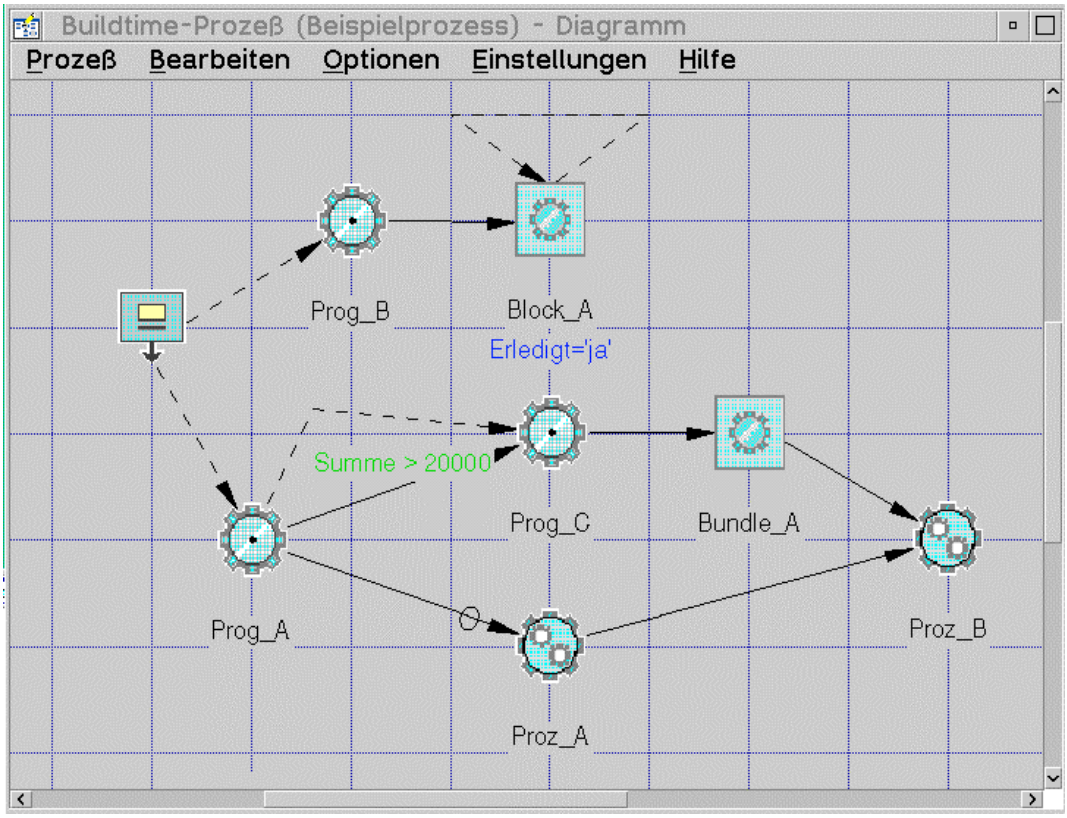


Abbildung 14: Beispiel eines FlowMark Prozeß-Modells

4.1.3 Diskussion der diagrammsprachlichen Modellierungsmöglichkeiten von FlowMark

Da die Workflows in FlowMark vorwiegend mit Hilfe eines grafischen Editors modelliert werden, ist an dieser Stelle eine genauere Untersuchung hinsichtlich der Erfüllung einer Reihe von Anforderungen, welche an diese Art der Modellierung in [Jablonski et al_97a, S. 162 ff.] gestellt werden, durchzuführen.

Als minimale Anforderung an eine Diagrammsprache wird die Möglichkeit der Darstellung von elementaren Aktivitäten genannt. Es sollte weiterhin möglich sein, den Kontroll- und Datenfluß zu spezifizieren. Der Kontrollfluß legt fest, in welcher Reihenfolge die einzelnen Aktivitäten des Workflows ausgeführt, der Datenfluß wie die Informationen innerhalb des Workflows verteilt werden. In der FlowMark-Diagrammsprache werden Bausteine zur Darstellung dieser Gesichtspunkte angeboten, wobei der Datenfluß getrennt von dem Kontrollfluß modelliert wird, was nicht unbedingt in allen im Workflow-Bereich verwendeten Diagrammsprachen der Fall ist [Jablonski et al_97a, S. 163]. Der Vorteil der expliziten Modellierung des Datenflusses ist darin zu sehen, daß dem Modellierer so die Möglichkeit der optimalen Verteilung der innerhalb des Workflows verwendeten Daten „in die Hand gegeben wird“ (vgl. [Alonso et al_97b, S. 7]).

Workflow-Modelle können sehr komplex werden. Daher ist es notwendig, Mechanismen zur Komplexitätsreduktion der Modelle anzubieten. Hierbei unterscheidet man zwischen der vertikalen (Dekomposition) und der horizontalen (Konkatenation) Unterteilung. Die Dekomposition ist in FlowMark, wie weiter oben ausführlich dargestellt, mittels Block- und Sub-Prozeß-Konzept möglich. Die Konkatenation erfordert zusätzliche Symbole für die Kennzeichnung von Schnittstellen, an denen Workflow-Modelle der gleichen Modellierungsebene miteinander verbunden werden können. Diese Möglichkeit wird von FlowMark nicht angeboten, wodurch eine horizontale Unterteilung der Modelle nicht möglich ist.

Ein Workflow kann sich aufteilen, Ablaufalternativen oder Wiederholungen enthalten. Dieses Verhalten wird mit Hilfe der von dem Modellierungstool angebotenen Kontrollflußkonstrukte beschrieben. FlowMark ermöglicht die Formulierung von Wiederholungen, alternativen, sequentiellen und parallelen Abläufen, wobei eine Synchronisation paralleler Abläufe möglich ist [Leymann et al_94, S. 331]. Diese den Kontrollkonstrukten höherer Programmiersprachen ähnelnden Formen der Ablaufkontrolle werden von [Jablonski_95] als fundamentale Formen der präskriptiven Ablaufkontrolle bezeichnet. Möglichkeiten zur deskriptiven Ablaufkontrolle werden nicht angeboten[6]. Man unterscheidet zwischen der expliziten und der impliziten Modellierung der Kontrollflußlogik. Bei der expliziten Modellierung wird das jeweilige Kontollflußkonstrukt durch einen eigenen Knoten repräsentiert. Bei der impliziten Modellierung wird der Kontrollfluß durch die Übergangsbedingungen beschrieben. Letztere Methode wird von FlowMark verwendet. Der Nachteil der impliziten Modellierung ist darin zu sehen, daß die Klarheit der Modelle dadurch verlorengeht, daß man nicht sofort erkennen kann, um welche Art von Verknüpfung (XOR, AND, OR) es sich handelt, was bei der expliziten Modellierung, durch die eigens dafür geschaffenen Konstrukte, keine Schwierigkeit darstellt. Allerdings entfällt bei der impliziten Modellierung die Redundanz, die sich bei der Verwendung von Knotentypen ergibt, da die durch sie wiedergegebene Logik nochmals durch Angabe einer eindeutig auswertbaren Spezifizierung der Ablaufbedingungen angegeben werden muß.

Eine weitere Anforderung, die an die Diagrammsprache im Zusammenhang mit der Programmiersprache (siehe nächstes Kapitel) gestellt wird, ist die Möglichkeit der konsistenten Weiterentwicklung der Sprache um zusätzliche Konstrukte. Damit neue Konstrukte der Diagrammsprache auch in der Programmiersprache (und umgekehrt) verfügbar gemacht werden können, ist eine bidirektionale Schnittstelle für eventuelle Ergänzungen nötig. So sollten z.B. neue Konstrukte der Diagrammsprache, die eine kompaktere Darstellung der Workflow-Modelle ermöglichen, in die Notation der Programmiersprache umgesetzt werden können. In FlowMark existiert eine bidirektionale Schnittstelle zwischen der Diagrammsprache und der Programmiersprache, wobei eine Erweiterung allerdings nicht möglich ist (vgl. [Jablonski et al_97a, S. 166]).

4.1.4 FlowMark Definition Language (FDL)

Nach [Jablonski et al_97a, S. 170] sind diagrammsprachliche Methoden alleine zur vollständigen Spezifikation von Workflows nicht ausreichend. Dies ist vor allem dann der Fall, wenn es um die Zuordnung von organisatorischen und technischen Ressourcen zu den Prozessen und deren Aktivitäten geht. Hier ist eine textuelle Spezifikation der Attribute wesentlich effizienter. [Alonso et al_97b, S. 9] führt in diesem Zusammenhang an, daß

[6] Zur präskriptiven und deskriptiven Ablaufkontrolle vgl. [Jablonski_95] und Kapitel 3.3.3.

besonders bei komplexen Workflows die Übersichtlichkeit bei der grafischen Modellierung schnell verloren gehen und die Verwendung einer Programmiersprache die Modellierung in diesem Fall handhabbarer machen kann.

Auch FlowMark bietet eine Möglichkeit an, Prozesse und deren Attribute vollständig auf textueller Basis zu definieren. Die hierzu verwendete Workflow-Sprache wird als FlowMark Definition Language (FDL) bezeichnet und im folgenden vorgestellt. Die FDL wird definiert als: „An external format for defining staff, programs, data structures and workflow models in a flat file. The definitions in the FDL file can then be imported into a FlowMark database.“[IBM_96c, S. 433]

Verwendungsmöglichkeiten der FlowMark Definition Language

Alle Prozeß-Definitionen, die diagrammsprachlich mittels des grafischen Editors durchgeführt werden können, können auch auf textueller Ebene in der FDL beschrieben werden. Es werden Konstrukte angeboten, welche die Definition von Prozessen, Aufbauorganisation, Servern, Programmen und Datenstrukturen ermöglichen (Details im nächsten Kapitel).

Laut [IBM_96a, S. 119 ff.] kommt der Benutzer (Programmierer oder Systemadministrator) in verschiedenen Situationen mit der FDL in Kontakt, welche hier kurz erwähnt werden sollen. Zunächst einmal hat der Programmierer die Wahl, ob er die Workflows lieber mit Hilfe der Diagrammsprache oder der FDL beschreiben möchte. Falls er sich für die FDL entscheidet, kann er hierzu einen beliebigen Editor verwenden der eine Ausgabe im ASCII-Format erzeugt. Diese Datei muß dann mittels des von FlowMark zur Verfügung gestellten Im/Export-Werkzeugs in die FlowMark-Datenbank importiert werden. Nach einer Konvertierung in das von FlowMark intern verwendete Format (wozu keine weiteren Informationen aus der Dokumentation hervorgehen) stehen die Definitionen zur weiteren Benutzung zur Verfügung und die definierten Prozesse können z.B. mittels des grafischen Editors weiterbearbeitet werden.

Das Im/Export-Werkzeug kann, wie der Name schon sagt, auch zum Export der in FlowMark verwalteten Workflow-Modelle verwendet werden. Hierbei werden die Definitionen vollständig in das FDL-Format umgewandelt und in eine Text-Datei geschrieben, die dann weiterbearbeitet werden kann. In einigen Situationen, wo z.B. große und unübersichtliche Workflows definiert werden sollen, kann es effizienter sein, ein Workflow-Modell zu exportieren, die Änderungen durchzuführen und dann wieder zu importieren, als die Änderungen mit dem grafischen Editor zu vollziehen. Der Benutzer kann beim Export der Workflow-Modelle entscheiden, welche Teile exportiert werden sollen (z.B. nur die Prozeß-Definitionen oder nur die definierten Rollen und Personen). Somit ist es auch möglich verschiedene FDL-Files anzulegen, die z.B. nur die Definitionen der Aufbauorganisation enthalten oder nur Prozesse einer bestimmten Kategorie

Das Im/Export-Werkzeug findet auch bei der Erzeugung von Backups seine Verwendung. Die generierten FDL-Files können dann an einem gesicherten Ort aufbewahrt werden.

Falls Workflow-Definitionen von einer FlowMark-Datenbank in eine andere kopiert werden sollen, muß ebenfalls auf das Im/Export-Werzeug und die FDL zurückgegriffen werden. Hier wird so verfahren, daß zunächst die entsprechenden Definitionen aus der einen Datenbank exportiert und dann in die andere Datenbank importiert werden. Auf diese Art und Weise können z.B. Definitionen aus verschiedenen Datenbanken auf einer neuen Datenbank

zusammengeführt werden (um z.B. schon definierte Prozesse und Datenstrukturen wiederzuverwenden oder eine Prozeß-Bibiliothek aufzubauen).

Falls schon ein FDL-File existiert, dem weitere Informationen hinzugefügt werden sollen, so können diese an das File angehängt werden.

Wenn man ein anderes Modellierungstool zur Definition der Workflow-Modelle verwenden möchte, diese allerdings durch FlowMark ausgeführt werden sollen, so könnte man einen Übersetzter entwickeln, der aus den Workflow-Modellen FDL-Files erzeugt, die dann in FlowMark importiert werden. Laut [Leymann et al_97b] existiert z.B. für das Geschäftsprozeßmodellierungs-Werkzeug ARIS der Firma IDS Prof. Scheer ein solcher Übersetzer.

FlowMark Definition Language

Im folgenden wird dargestellt, wie ein FDL-File aufgebaut ist und wie die Definition von Prozessen in FDL „mit allem was dazu gehört“ auszusehen hat. Ziel dieses Kapitels ist es nicht, alle Schlüsselwörter und Feinheiten der FDL zu erklären, sondern ein Überblick zu geben, um einen ersten Eindruck zu vermitteln. Für detailliertere Informationen sei auf [IBM_96a, S. 192 ff.] verwiesen.

Grundsätzlich teilt sich ein FDL-File in zwei Bereiche auf. Im ersten Bereich finden sich Deklarationen von Datenstrukturen, Programmen, Servern und Aufbauorganisation (Staff). Danach folgt die Definition der Prozesse. Mittels der „INCLUDE“-Anweisung können Definitionen, die sich in anderen FDL-Files befinden, importiert werden. Die Inhalte dieser Files werden dann bei der Übersetzung in die von FlowMark intern verwendete Sprache ebenfalls übersetzt. FDL ist „case-sensitive“. Alle Schlüsselworte sind in Großbuchstaben zu schreiben. Um Namensgleichheiten zu Schlüsselwörtern zu vermeiden, sollten alle Bezeichner in Hochkommata gesetzt werden.

Datenstrukturen, Programme, Server, Aufbauorganisation

In der FDL stehen drei Datentypen zur Verfügung: STRING, LONG und FLOAT. Außerdem ist es möglich Arrays dieser drei Basistypen zu erzeugen. Die Definition benutzerdefinierter Datenstrukturen wird mit dem Schlüsselwort STRUCTURE eingeleitet, gefolgt von dem Namen der Datenstruktur und den Membern (FlowMark bezeichnet Variablen als Member; siehe Kapitel 4.1.2). Die Definition ist mit dem Schlüsselwort END und Angabe des Namens abzuschließen. In Beispiel 1 ist die Definition einer Datenstruktur mit dem Namen „WordDatenstruktur“ dargestellt. Soll zur Laufzeit auf den Wert eines Members der Datenstruktur zugegriffen werden, so ist dies qualifiziert über Punktnotation möglich (z.B. WordDatenstruktur.DateiName).

```
STRUCTURE 'WordDatenstruktur'
DESCRIPTION    'Diese Datenstruktur wird bei der Verwendung von Microsoft Word zur Implementation einer Programm-
          Aktivität als Inhalt der Eingabe- und Ausgabedaten-Container verwendet'
  'DateiName':   STRING;  //Name der Datei, mit der Word zu starten ist
  'Erledigt':    STRING; //Wird zur Auswertung von Endebedingungen verwendet. (siehe Beispiel 5)
END 'WordDatenstruktur'
```

Beispiel 1: Benutzerdefinierte Datenstruktur

Die Definition von Programmen wird mit dem Schlüsselwort PROGRAM und Angabe des Namens eingeleitet (siehe Beispiel 2). Optional können die Namen der Datenstrukturen von

Ein- und Ausgabeparameter in Klammern angegeben werden. Wird kein Datentyp angegeben, so wird die Standarddatenstruktur angenommen. Innerhalb der Definition werden technische Angaben über Programmspezifika (z.B. DESCRIPTION), den Ort des Programms bei entferntem Programmaufruf (z.B. TCPIP ADDRSS) und plattformspezifische Angaben (z.B. PATH_AND_FILENAME, WORKING_DIRECTORY) gemacht. Sollen Parameter beim Programmaufruf mitgegeben werden, so wird dies durch das Schlüsselwort PARAMETER gekennzeichnet. Durch Einklammern des Bezeichners eines Members der Eingabedatenstruktur in Prozentzeichen kann auf dessen Wert zugegriffen werden um ihn z.B. wie in Beispiel 2 als Parameter beim Programmaufruf zu verwenden.

```
PROGRAM 'Word'   ( 'WordDatenstruktur', 'WordDatenstruktur' )
  DESCRIPTION ' Microsoft Word Version 8.0'
  WINNT PATH_AND_FILENAME 'Winword.exe'  // Word muß über die PATH-Umgebungsvariable verfügbar sein
  WINNT PARAMETER '%DateiName%'
END 'Word'
```

Beispiel 2: Programmdefinition

Server-Definitionen (Beispiel 3) werden mit dem Schlüsselwort SERVER gefolgt von dem Namen begonnen. Hier werden z.B. Angaben über zu verwendende Protokolle (PROTOCOL) oder Datenbanken gemacht (DATABASE).

```
SERVER 'EXMSRV'
  PROTOCOL TCPIP           // Als Kommunikationsprotokoll wird TCPIP verwendet
  ADDRESS  '134.100.9.131' // Die Datenbank befindet sich auf dem Rechner mit der angegebenen Adresse
  DATABASE 'EXMDB'      // Die Datenbank auf dem Rechner heißt EXMDB
END 'EXMSRV'
```

Beispiel 3: Serverdefinition

Unter dem Begriff Staff-Definition wird die Definition von Personen, Rollen, Organisationen und Ebenen zusammengefaßt. Die Definitionen werden mittels der entsprechenden Schlüsselwörter gefolgt vom Namen eingeleitet (PERSON, ROLE, ORGANIZATION, LEVEL). Bei der Definition einer Person (vgl. Beispiel 4) werden neben persönlichen Angaben z.B. auch Angaben zu Stellvertretern (SUBSTITUTE), Ebenen (LEVEL), Rollen (RELATED_ROLE), Organisation (RELATED_ORGANIZATION) und Authorisierungen (AUTHORIZED_FOR) gemacht.

```
// Personen
PERSON 'Hugo'
  PERSON_ID    'hugo'       // Person ID wird innerhalb von FlowMark zu eindeutigen identifikation verwendet
  FIRST_NAME   'Hugo'
  LAST_NAME    'Boss'
  PHONE        '4911'
  SUBSTITUTE   'karl'                    // Der Vertreter von Hugo ist die Person mit der PersonID 'karl'
  RELATED_ROLE   'Veranstalter'          // Hugo ist Träger der Rolle 'Veranstalter'
  RELATED_ORGANISATION 'AB SWT'          // und gehört der Organisation 'AB SWT' an
END 'Hugo'
```

Beispiel 4: Personendefiniton

Prozesse, Kontrollfluß und Datenfluß

Prozesse werden mittels des Schlüsselwortes PROCESS gefolgt vom Prozeßnamen definiert (vgl. Beispiel 5). Hier sind, wie bei der Definition von Programmen, ebenfalls optional Ein- und Ausgabeparameter durch Angabe der jeweiligen Datentypen festzulegen. Auch hier wird bei Verzicht auf Angabe die Standarddatenstruktur angenommen. Innerhalb der Definition sind z.B. Aussagen darüber zu machen, wer der Prozeßadministrator

(PROCESS_ADMINISTRATOR) ist, wie lange der Prozeß dauern darf (DURATION), ob beim Starten des Prozesses Daten angefordert werden sollen (PROMT_AT_PROCESS_START), usw. Außerdem sind Angaben über den Kontroll- und den Datenfluß innerhalb des Prozesses zu machen.

```
// Prozesse

/*Innerhalb des Prozesses AB-Lehrplanung existieren eine Programm-Aktivität (Lehrdeputate_Errechnen) und
*ein Block  (Lehrangebot_Deputate_Erstellen) zwischen denen Daten ausgetauscht werden  (WordDaten-
*struktur).  Ein Mapping der Member kann entfallen, da der Typ der Datenstrukturen der gleiche ist.
*(Mapping läuft automatisch).
*/

PROCESS 'AB-Lehrplanung' ( 'Standarddatenstruktur', 'Standarddatenstruktur' )
  CATEGORY 'AB-Lehrplanung'     // der Prozeß gehört der Kategorie AB-Lehrplanung an. Nur Berechtigte dürfen
                                // ihn starten (vgl. Kapitel 4.1.1)
 PROMT_AT_PROCESS_START

PROGRAM_ACTIVITY 'Lehrdeputate_Errechnen'   ( 'WordDatenstruktur', 'WordDatenstruktur' )
    DESCRIPTION 'Lehrdeputate errechnen.'
    PROGRAM     'Word'                                  // starten der Aktivität ruft das Programm Word auf
    EXIT AUTOMATIC      WHEN 'Erledigt="ja"'            // legt Endebedingung fest
    DONE_BY MEMBER OF ROLE  'AB-Lehrplanungsbeauftragter'
    INPUT_CONTAINER                                     // Verweis auf den Eingabedaten-Container
      'DateiName'   INITIAL_VALUE '/mStarterLehrdeputate'   // legt den Wert fest, mit dem der Member
                            // „DateiName“ des Eingabedaten-Struktur initialisiert wird. Hat „DateiName“ einen
                            // Wert, so wird dieser beim Aufruf von Word mitgegeben (vgl. Beispiel 2) und der
                            // Initial-Wert ignoriert.
  END 'Lehrdeputate_Errechnen'

BLOCK 'Lehrangebot_Deputate_Erstellen' ( 'WordDatenstruktur', 'Standarddatenstruktur' )
    START AUTOMATIC WHEN ALL CONNECTORS TRUE
          ...
END 'Lehrangebot_Deputate_Erstellen'

// nun den Kontrollfluß festlegen:
CONTROL FROM 'Lehrdeputate_Errechnen' TO 'Lehrangebot_Deputate_Erstellen'

//nun die Datenzuordnung festlegen
  DATA FROM 'Lehrdeputate_Errechnen'   TO 'Lehrangebot_Deputate_Erstellen'
    MAP '_STRUCT'    TO '_STRUCT'                       // kennzeichnet automatisches Mapping
    ...

END 'AB-Lehrplanung'
```

Beispiel 5: Prozeßdefinition mit Angabe des Kontroll- und Datenflusses

Zur Spezifikation des Kontrollflusses verwendet man das Schlüsselwort CONTROL entweder in Kombination mit „FROM a TO b“ oder „a -> b“ wobei a und b Aktivitäten repräsentieren. Falls es um die Festlegung des Kontrollflusses in Abhängigkeit von Alternativen geht, kann CONTROL auch in Kombination mit den Schlüsselwörtern WHEN und OTHERWISE verwendet werden. Hinter WHEN ist dann eine Bedingung in Anführungszeichen (als STRING) zu formulieren. Zur Formulierung des Datenflusses verwendet man die Schlüsselworte „DATA FROM a TO b“ (a und b sind Aktivitäten) wobei dann noch anzugeben ist, wie die einzelnen Member der beteiligten Datenstrukturen zuzuordnen sind. Hierzu verwendet man die Schlüsselworte „MAP c TO d“, wobei c und d Member repräsentieren (zum Thema Mapping siehe Kapitel 4.1.2) .

Unter Verwendung des Schlüsselwortes INPUT_CONTAINER kann auf den Inhalt des Eingabedaten-Containers zugegriffen werden. Ein Zugriff auf den Ausgabedaten-Container ist

unter Verwendung der FDL nicht möglich. Um Daten des Ausgabedaten-Containers zu setzen, muß auf die sogenannten APIs (Application Programming Interfaces) zurückgegriffen werden (siehe hierzu Kapitel 4.3.4).

Eigenschaften von Programmiersprachen

In [Pratt et al_98, S. 34 ff.] werden acht Attribute beschrieben, die gute Programmiersprachen aufweisen sollten. Diese werden hier vorgestellt, um eine Grundlage zur Beurteilung der FDL, welche sich im nächsten Abschnitt findet, zu liefern.

1. **Klarheit, Einfachheit und Einheitlichkeit:** Eine Programmiersprache sollte das Attribut der konzeptionellen Integrität aufweisen. Hierunter versteht man die Eigenschaft, eine gewisse Mindestanzahl an verschiedenen Konzepten aufzuweisen, deren Kombinationsregeln so einfach wie möglich sein sollten. Wichtig ist auch die Lesbarkeit der Programme. Dieses Attribut hängt von der verwendeten Syntax und insbesondere von der Semantik der Konstrukte ab. Die Bedeutung der Konstrukte sollte eindeutig sein und Unterschiede müssen sich in der Syntax der Sprache widerspiegeln.
2. **Orthogonalität:** Dieser Begriff bezeichnet die Eigenschaft einer Sprache, verschiedene Merkmale in allen möglichen Kombinationen zu vermischen, wobei jede Kombination eine sinnvolle Bedeutung hat. So sind z.B. ein Ausdruck, der einen Wert produziert und eine Bedingung, die einen Ausdruck bewertet, um einen booleschen Wert zu bekommen, orthogonal, wenn in der Bedingungsanweisung ein beliebiger Ausdruck benutzt werden kann.
3. **Natürliche Eignung für die Anwendung:** Eine Programmiersprache sollte die Eigenschaft der natürlichen Eignung für eine Anwendungsklasse aufweisen. So eignet sich C++ oder Smalltalk sehr gut für das objektorientierte Design, wohingegen Prolog eher zur Lösung logischer Probleme herangezogen wird. Somit sollten angemessene Datenstrukturen, Operationen, Kontrollstrukturen und eine natürliche Syntax für das zu lösende Problem der jeweiligen Anwendungsklasse angeboten werden.
4. **Unterstützung für Abstraktion:** Eine Programmiersprache sollte Datenstrukturen, Datentypen und Operationen zur Verfügung stellen, mit denen Abstraktionen formulierbar sind. Somit wird es möglich, Programmteile zu benutzen, ohne ihre konkrete Implementierung kennen zu müssen. Es müssen nur ihre abstrakten Eigenschaften bekannt sein. Neben der Erhöhung der Eigenschaften der Wiederverwendbarkeit und Änderbarkeit erhöht die Abstraktion auch den Zusammenhang der abstrakten Datenstrukturen und Operationen, die eine Problemlösung charakterisieren, und den einfachen Datenstrukturen und Operationen, die in einer Sprache vorhanden sind und auf die die Problemlösung heruntergebrochen werden muß.
5. **Leichte Programmverifikation:** Programme sollen zuverlässig sein. Zur Überprüfung dieser Eigenschaft gibt es verschiedene Methoden (formale Verifikation, Desk-Checking, Testen), die meist kombiniert angewandt werden. Eine Sprache sollte diese Methoden unterstützen, wobei die Einfachheit der semantischen und syntaktischen Struktur ein erster Ansatz ist.
6. **Programmierumgebung:** Der Umgang bzw. die Nutzbarkeit einer Programmiersprache wird nicht nur durch ihre technischen Eigenschaften, sondern auch durch ihre Umgebung bestimmt. Zu einer Programmierumgebung gehört z.B. die Verfügbarkeit einer Dokumentation, das Vorhandensein von auf die Sprache zugeschnittenen Editoren und Testpaketen, Einrichtungen zum Pflegen und Ändern eines Programmes. Falls diese

Unterstützungen angeboten werden, kann es dazu führen, daß eine technisch schwächere Sprache leichter benutzt und akzeptiert werden kann als eine stärkere.

7. **Portabilität von Programmen:** Aufgrund der Standardisierungen der Definitionen von Programmiersprachen, kann eine Portierbarkeit auf andere Computersysteme erreicht werden. Eine für viele Plattformen verfügbare Sprache ist eine gute Grundlage für portable Programme.
8. **Nutzungskosten:** Man unterscheidet folgende Kategorien:
   - *Kosten der Programmausführung:* Dieser Kostenfaktor betrifft die Schnelligkeit der Programmausführung. Dieser Faktor steht aufgrund der heutzutage vorhandenen schnellen Hardware bei den meisten Anwendungen nicht mehr im Vordergrund.
   - *Kosten der Programmübersetzung:* Dieser Faktor bezieht sich auf die Dauer der Programmcompilation. Besonders dort, wo viel übersetzt und getestet wird, stehen diese Kosten im Vordergrund und die Kosten der Programmausführung sind eher nebensächlich (z.B. in der Programmierausbildung).
   - *Kosten für Entwicklung, Prüfung und Nutzung des Programms:* Die Minimierung dieser Art von Kosten sind heutzutage als wesentlich wichtiger einzuschätzen als die traditionellen Kosten für die effiziente Programmausführung oder Übersetzung. Es geht darum, die Gesamtkosten für die Zyklen der Entwicklung und Umsetzung einer Problemlösung zu minimieren (entwickeln, kodieren, testen, nutzen).
   - *Kosten der Programmpflege:* Dieser Kostenfaktor kann als der größte betrachtet werden. Er bezieht sich auf die Behebung von Fehlern, die erst während der Benutzung zu Tage treten, auf Anpassungen an neue Hardware, auf Erweiterungen des Programms, usw. Eine Sprache sollte die wiederholte Änderung, Instandhaltung und Erweiterung der Programme durch verschiedene Programmierer unterstützen.

Diskussion der FlowMark Definition Language

In diesem Abschnitt soll die FDL hinsichtlich der Erfüllung der soeben beschriebenen Anforderungen an Programmiersprachen untersucht werden.
Die FDL beschreibt das Format, das verwendet werden muß, wenn FlowMark Prozeß-Modelle auf textueller Basis definiert werden sollen. Die im FDL-Format beschriebenen Prozeß-Modelle werden vom Import-Werkzeug in das von FlowMark intern verwendete Format übersetzt. Die FDL ist somit eine kompilierte Sprache (wobei das Zielformat der Kompilation allerdings in der Dokumentation nicht benannt wird).

Mit Hilfe der FDL kann genau das beschrieben werden, was auch diagrammsprachlich möglich ist. Es handelt sich praktisch um das textuelle Äquivalent zur Diagrammsprache von FlowMark. Dadurch ist der Sprachumfang sehr beschränkt, was dazu führt, daß die FDL sehr schnell erlernt werden kann. Die Semantik der Konstrukte ist eindeutig. Allerdings ist nicht ersichtlich, warum zwei Möglichkeiten zur Festlegung des Kontrollflusses angeboten werden („a -> b“ oder „FROM a TO b“), da sie die gleiche Semantik aufweisen.

Die FDL ist keine **orthogonale** Programmiersprache, da die elementaren Konstrukte nicht beliebig miteinander kombinierbar sind. So erfordert z.B. das elementare Konstrukt welches den Kontrollfluß festlegt, daß als Parameter nur Ausdrücke angegeben werden dürfen, die (Prozeß- oder Programm-) Aktivitäten bezeichnen. Diese können allerdings nicht z.B. mittels eines elementaren Operators wie „AND“ oder „NOT“ kombiniert werden um eine neue Aktivität zusammenzusetzen, die dann als Parameter übergeben wird.

Die FDL wurde extra für die Formulierung von Workflows und der Ausnutzung der von FlowMark angebotenen Möglichkeiten entwickelt. Die zur Verfügung gestellten Konstrukte ermöglichen eine dem Anwendungsgebiet angemessene Formulierung der Algorithmen (in diesem Fall Prozesse). Somit ist die **natürliche Eignung** der Sprache gegeben.

Hinsichtlich einer **Abstraktion** wird die Möglichkeit der Verwendung von Sub-Prozessen angeboten. Hier muß der Programmierer nur den Namen und die Ein- und Ausgabedatenstrukturen kennen um den Sub-Prozeß wiederzuverwenden. Um die konkrete Implementation braucht er sich nicht zu kümmern. Die Definition des Sub-Prozesses kann mittels INCLUDE-Anweisung importiert werden. Weitere Abstraktionsmöglichkeiten werden nicht angeboten. Somit bewegt sich das Abstraktionsniveau auf der Ebene von Unterprogrammen (vgl. [Pratt et al_98, S. 266]).

Eine **Überprüfung** der Syntax der FDL-Files erfolgt, wenn man sie von FlowMark importieren läßt. Logische Fehler sind erst zur Laufzeit erkennbar. Allerdings gibt es die Möglichkeit, die beschriebenen Prozesse vor der Übersetzung in die FRL (FlowMark Run-Time Language; das Format, in dem sich die Prozeß-Modelle zur Laufzeit befinden) mit Hilfe eines Animations-Werkzeugs zu testen, um so schon einige logische Fehler wie z.B. fehlende Zuordnungen von Personen zu Aktivitäten aufzudecken (das Animations-Werkzeug wird in Kapitel 4.3. beschrieben).

Es existiert keine **Programmierumgebung** für die FDL. Somit ist die Handhabung und Verwaltung vieler Prozesse außerhalb von FlowMark als schwierig einzuschätzen.

Eine **Portierung** der in FDL beschriebenen Prozesse auf ein anderes WfMS ist nicht möglich. Die FDL ist nicht konform zur Prozeßbeschreibungssprache der Workflow Management Coalition (WPDL = Workflow Process Definition Language) [WfMC_98].

Eine isolierte Betrachtung der Nutzungskosten der FDL ist nicht möglich. Der Grund hierfür liegt darin, daß sich Entwickler während der Entwicklungszyklen nicht nur auf die Nutzung der FDL beschränken werden, sondern sicherlich auch die diversen anderen Werkzeuge, die FlowMark anbietet, nutzen (die Werkzeuge werden in Kapitel 4.3 beschrieben). Somit wird an dieser Stelle auf eine Beurteilung verzichtet, da sie nicht aussagekräftig genug wäre.

4.1.5 Diskussion der Modellierungskonzepte von FlowMark

Wie in Kapitel 3.3.1 erwähnt, wird die Leistungsfähigkeit eines WfMS hinsichtlich der Unterstützung von Arbeitsvorgängen dadurch bestimmt, welche Möglichkeiten zur Modellierung von Geschäftsprozessen, die ja diese Art von Vorgängen beschreiben, angeboten werden (vgl. [Jablonski et al_97a, S. 6]). Die Modellierungsmöglichkeiten von FlowMark (sowohl grafisch als auch textuell) wurden im Verlauf des Kapitels 4.1. ausführlich dargestellt.

Die in Kapitel 3.3.3 geforderte Möglichkeit der getrennten Modellierung der in einem Workflow vorkommenden Aspekte (Aktivitäten, Aktoren, Abhängigkeiten, Kausalitäten) wird von FlowMark größtenteils angeboten. Lediglich die Formulierung deskriptiver Ablaufkontrolle und von Kausalitäten wird nicht angeboten. Somit kann die Frage nach dem „Warum“ eines Workflows in FlowMark nicht beantwortet werden. Da FlowMark als WfMS die Rolle eines Koordinators einnimmt (vgl. Kapitel 3.3.2), sollte man erwarten können, daß eine Überprüfung kausaler Aspekte möglich ist. Durch das Fehlen dieser Möglichkeit bleibt die Verantwortung der Prämissenprüfung bei den Akteuren des Systems, was einen Widerspruch im Hinblick auf die Systemphilosophie darstellt.

Als weiterer Nachteil der Modellierunsgkonzepte von FlowMark sei angeführt, daß Rücksprünge im Prozeßverlauf nicht formulierbar sind. Dies entspricht in keinerlei Hinsicht den täglich anzutreffenden Arbeitssituationen, da es immer wieder zu Fehlern kommen kann, die erst zu späteren Zeitpunkten aufgedeckt werden. Selbst wenn z.B. der nächste Bearbeiter eines Dokumentes einen Fehler aufdeckt, den sein Vorgänger gemacht hat, kann die Möglichkeit eines Rücksprung innerhalb des Prozeß-Modells nicht vorgesehen werden. Ebenso hat eine Änderung des Prozeß-Modells eines gerade laufenden Prozesses keine Auswirkung auf diesen. Somit ist es unmöglich, den Ablauf von Vorgängen spontan an sich ändernde Anforderungen anzupassen. Änderungen wirken sich erst nach Neuübersetzung und Neustart des modifizierten Prozeß-Modells aus. Allerdings besteht die Möglichkeit, Teile des Gesamtprozesses an sich verändernde Anforderungen anzupassen. Dies hängt mit dem Konzept zusammen, daß FlowMark Exemplare von Sub-Prozesse erst dann erzeugt, wenn der Prozeßverlauf an die entsprechende Stelle gelangt („late binding“). Somit kann durch Änderung des betreffenden Sub-Prozesses und rechtzeitiger Übersetzung eine Anpassung vollzogen werden. Allerdings werden Schwachstellen oft erst zur Laufzeit erkannt, wodurch eine Berücksichtigung nur beim nächsten Durchlauf möglich wird.

Als weitere Einschränkung ist die Vorgabe anzusehen, daß hinsichtlich der Modellierung der Aufbauorganisation ein vordefiniertes Organisationsmodell verwendet wird, welches die möglichen Beziehungen zwischen den Organisationseinheiten festlegt. Dies ist unter Berücksichtigung des ständigen Wandels der Aufbauorganisation heutiger Unternehmen aufgrund veränderter Marktsituationen (vgl. Kapitel 1) als zu inflexibel einzuschätzen.

Speziell zur FDL sei noch erwähnt, daß in Zukunft wohl mit keiner weiteren Unterstützung hinsichtlich der Entwicklung von Programmierumgebungen von Seiten IBMs gerechnet werden kann. Dies geht daraus hervor, daß in der Dokumentation der neuen Version von FlowMark (Version 3.1.2, vgl. [IBM_98b]) die FDL nur noch als externes Format für Prozeß-Modelle definiert wird, das den Austausch der Modelle zwischen FlowMark Datenbanken untereinander und Modellierungswerkzeugen anderer Hersteller und FlowMark ermöglicht. Die Verwendung als Programmiersprache ist nicht mehr vorgesehen. Bei der Umsetzung des in Kapitel 2 vorgestellten Anwendungsbeispiels haben wir ebenfalls darauf verzichtet, die FDL zu verwenden. Dies war nicht nötig, da sich die Größe der Prozeß-Modelle in überschaubaren Grenzen gehalten hat und das Modellierungswerkzeug einen komfortablen Umgang mit den Prozessen ermöglichte, der unter Verwendung der FDL so nicht möglich gewesen wäre.

4.2 Komponenten von Workflow-Management-Systemen

Innerhalb dieses Kapitels sollen die Komponenten, aus denen sich die Architektur von WfMS konstituiert, vorgestellt werden. Die folgenden Ausführungen beziehen sich vorwiegend auf die in [Jablonski et al_97a, S. 219 ff.] zu findenden. Es werden zunächst die Anforderungen, die an WfMS gestellt werden, aufgeführt, welche grundlegenden Einfluß auf den Aufbau der WfMS-Architektur haben (Kapitel 4.2.1). Danach werden Architekturmodelle vorgestellt (Kapitel 4.2.2). Dieses Kapitel stellt die Grundlage zur Beschreibung und Beurteilung der Architektur von FlowMark, als konkrete Implementierung eines WfMS, dar (Kapitel 4.3).

4.2.1 Anforderungen an Workflow-Management-Systeme

Die Anforderungen, die allgemein an Softwaresysteme gestellt werden, lassen sich grundsätzlich in funktionale und nichtfunktionale aufteilen. „Funktionale Anforderungen

beschreiben die angebotenen Dienste und die Reaktion des Systems auf Eingaben, während nichtfunktionale Anforderungen Bedingungen an die Funktionsausführung stellen, zum Beispiel geforderte Antwortzeiten oder Verfügbarkeit.“ [Jablonski et al_97a, S. 219] Diese Einteilung wird im Folgenden auch zur Klassifizierung der an WfMS gestellten Anforderungen verwendet.

Funktionale Anforderungen

Unter einem Aspekt versteht man laut [Jablonski et al_97a, Glossar] eine Sichtweise oder ein Gesichtspunkt unter dem etwas betrachtet wird. Wenn man Prozesse betrachtet, lassen sich vier Gesichtspunkte identifizieren, die relevant und in allen Prozessen zu finden sind (vgl. [Curtis et al_92]). Da Geschäftsprozesse ebenfalls Prozesse sind, deren als Workflows bezeichnete Implementationen von WfMS ausgeführt werden, weisen sie ebenfalls diese vier Aspekte auf. Es lassen sich Funktionsaspekt, Verhaltensaspekt, Informationsaspekt und Organisationsaspekt konkretisieren. Somit ergibt sich die Anforderung an die Architektur eines WfMS, diese im folgenden kurz dargestellten Aspekte auch zu erfüllen[7]. Die konkrete Implementation der Aspekte bleibt aufgrund ihres allgemeinen Charakters den Herstellern der Systeme überlassen. Für bestimmte Anwendungsgebiete können sich spezielle Anforderungen ergeben, die jedoch nicht unbedingt auf andere Gebiete übertragbar sind (z.B. Sicherheitsaspekte) und somit hier nicht näher betrachtet werden.

Der **Funktionsaspekt** definiert die einzelnen Aufgaben, die im Rahmen eines Workflows auszuführen sind. Ein WfMS muß somit in der Lage sein, Workflows und ihre Zusammensetzung aus Sub-Workflows und Workflow-Applikationen zu verwalten.

Der **Verhaltensaspekt** konstituiert den Kontrollfluß zwischen den Workflows. Er legt also die Reihenfolge die einzelnen Teilschritte eines Arbeitsvorgangs fest.

Der **Informationsaspekt** bestimmt den Datenfluß zwischen den Workflows und zwischen den Workflows und den Workflow-Applikationen.

Der **Organisationsaspekt** sorgt für die Zuweisung von Workflows und Aufgaben zu Organisationseinheiten.

Aufgrund der Tatsache, daß ein WfMS eine Vielzahl von Anwendungen zur Durchführung der in einem Workflow spezifizierten Aufgaben aufruft, ergibt sich ein weiterer Aspekt, der als **Operationsaspekt** bezeichnet wird und die Integration von Programmen (Applikationsintegration) meint, was charakteristisch für diese Art von Prozeß ist. Aufgrund der Relevanz der Integration von Applikationen in Workflow-Management-Anwendungen ist diesem Aspekt ein eigenes Kapitel gewidmet, in dem die Anforderungen detaillierter beschrieben werden (siehe Kapitel).

Bei WfMS handelt es sich um hochgradig interaktive Systeme. Dadurch ist es notwendig, entsprechende Benutzerschnittstellen zur Verfügung zu stellen, welches eine weitere funktionale Anforderung darstellt. Als Benutzer dieser Systeme lassen sich Endanwender, Entwickler und Administratoren identifizieren, wodurch sich die erforderlichen Schnittstellen ergeben.

[7] In Kapitel 3.3.3 wurde bereits darauf hingewiesen, daß Workflows die Betrachtung dieser unterschiedlichen Aspekte während ihrer Modellierung erfordern. Dies führt natürlich auch zur Notwendigkeit ihrer Berücksichtigung auf Architekturebene des unterstützenden Softwaresystems.

Über die **Anwenderschnittstelle** wird die Interaktion der Endanwender mit dem WfMS realisiert. Verwendung findet hier die sogenannte Arbeitsliste, in der die Aufgaben (mit weiteren Informationen wie Name des Workflow-Exemplars, Prioritäten, Eingangszeiten usw.) die von den Benutzern in ihrer täglichen Arbeit zu erledigen sind, aufgelistet werden. Es sollten Möglichkeiten zum Starten, Beenden, Abbrechen, Umleiten usw. von Workflow-Exemplaren geben.

Mittels **Entwicklerschnittstelle** interagiert der Entwickler von Workflow-Management-Anwendungen mit dem WfMS. Er benötigt Möglichkeiten zur Definition von Workflow-Modellen, zur Durchführung von Optimierungsanalysen und zur Überprüfung der Modelle auf formale Korrektheit und Gültigkeit.

Die **Administratorschnittstelle** stellt Möglichkeiten zur Benutzerverwaltung, zur Konfiguration des WfMS und zur Steuerung und Überwachung des Online-Betriebs des Systems zur Verfügung.

Nichtfunktionale Anforderungen

Zu den nichtfunktionalen Eigenschaften von WfMS werden Offenheit, Zuverlässigkeit und Analysierbarkeit gezählt, wobei die gewählten Begriffe im folgenden noch weiter aufgeschlüsselt werden.

Offenheit

Unter einem offenen System versteht man laut [POSIX_92] „ein System, das in ausreichendem Maße offengelegte Spezifikationen für Schnittstellen und dazugehörige Formate implementiert, damit entsprechend gestaltete Anwendungssoftware

- auf eine Vielzahl verschiedener Systeme portiert werden kann (mit Anpassungen)
- mit anderen Anwendungen lokal und entfernt interoperabel ist,
- mit Benutzern in einer Art interagiert, die das Wechseln der Benutzer zwischen den Systemen erleichtert.“

Die mit dieser Definition in Zusammenhang zu bringenden Forderungen nach Verteilbarkeit, Plattformunabhängigkeit und Skalier-und Erweiterbarkeit eines als offen zu bezeichnenden WfMS, führen zu einer offenen Client/Server-Architektur als Basis zur Erfüllung der Anforderungen. Um Plattformunabhängigkeit zu erreichen bietet sich die Verwendung von Middleware-Produkten an, welche die Kommunikation der verteilten Komponenten des WfMS ermöglichen. Middleware wird in [Jablonski et al_97a, Glossar] folgendermaßen definiert: „Middleware ist anwendungsneutrale Systemsoftware, die zwischen Anwendungsprogrammen und Betriebssystem, Datenbanksystemen und anderen Ressourcen-Managern läuft. Sie stellt Infrastrukturdienste zur Verfügung, auf deren Basis Applikationen ortstransparent entwickelt und betrieben werden können. Solche Dienste sind unter anderem Kommunikations-, Datenmanagement-, Transaktionsmanagement- sowie Netzwerk- und Systemmanagement-Dienste.“

Durch Offenlegen von Schnittstellen wird die Werkzeugintegration erleichtert und somit auch der Wechsel der Benutzer zwischen den Systemen, die auf für sie bekannte, auf das jeweilige WfMS angepaßte, Werkzeuge zurückgreifen können. Außerdem kann auf Basis offener Schnittstellen den Systemen neue Funktionalität durch zusätzliche Komponenten hinzugefügt werden, wodurch die funktionale Erweiterbarkeit der Systeme und somit die Anpaßbarkeit an veränderte Anforderungen gewährleistet wird.

„WfMS unterliegen prinzipiell keinen Beschränkungen hinsichtlich der Größe der Systeminstallation.“ [Schuster_97, S. 21] Durch die Forderungen, eine beliebig große Anzahl von Workflow-Exemplaren verwalten zu können und eine beliebig große Anzahl neuer Nutzer in das System zu integrieren, erwächst die Notwendigkeit der Skalierbarkeit und der Dezentralisierung (also Verteilung) der Funktionen der Systeme. Die Skalierbarkeit bezieht sich sowohl auf zu erwartenden durchschnittliche Antwortzeiten als auch auf die Anzahl der an einem WfMS partizipierende Knoten oder Benutzer, d. .h. daß sich das Laufzeitverhalten aufgrund einer höheren Nutzerzahl oder einer größeren Menge zu verwaltender Daten nicht verschlechtern sollte.

Auch zur Laufzeit sollte das System in der Lage sein, sich an veränderte Umgebungsbedingungen hinsichtlich Lastverteilung und Skalierung anzupassen, d.h. das System soll in der Lage sein, auch im laufenden Betrieb auf veränderte Umweltbedingungen (starkes Datenaufkommen, viele Benutzer, Ausfälle von Systemteilen, usw.) selbständig zu reagieren.

Zuverlässigkeit
Von WfMS wird eine hohe Systemverfügbarkeit und transaktionales Systemverhalten gefordert, wie es auch schon aus dem Datenbankbereich bekannt ist. Unter einer Transaktion wird generell eine Folge von Operationen verstanden, welche eine Datenbank von einem konsistenten Zustand in einen (nicht notwendiger Weise verschiedenen) konsistenten Zustand überführt [Jablonski et al_97a, S. 228]. Die von WfMS durchzuführenden Transaktionen unterscheiden sich allerdings von den Datenbanktransaktionen in den Punkten Transaktionsdauer, Transaktionsstruktur, Atomizität und Konsistenz und Isolation. Atomizität bedeutet, daß entweder alle Operationen einer Transaktion ausgeführt werden oder keine („Alles oder Nichts“). Konsistenz bedeutet, daß eine Transaktion eine Datenbank von einem konsistenten Zustand in einen anderen konsistenten Zustand überführt. Isolation heißt, daß nebenläufige Transaktionen sich nicht gegenseitig beeinflussen, also Zwischenzustände aktiver Transaktionen für andere Transaktionen nicht sichtbar sind. Der Unterschied zwischen Datenbanktransaktionen und Transaktionen in WfMS läßt sich dadurch begründen, daß Workflow-Aktivitäten meist wesentlich längere Ausführungszeiten aufweisen als traditionelle Transaktionen, komplexe Workflows hierarchisch aufgebaut sind (erfordert keine flachen, sondern geschachtelte Transaktionen), die „Alles oder Nichts“-Strategie aufgrund der Laufzeit der Workflows aus Kostengründen nicht angewandt werden kann und die isolierte Ausführung der Workflows aufgrund ihrer kooperativen Eigenschaft (z.B. gemeinsame Bearbeitung von Dokumenten) nicht akzeptabel ist. Aus dieser Tatsache entsteht die Forderung der Integration erweiterter Transaktionskonzepte (Transaktionskonzepte, die über das ACID-Paradigma hinausgehen) in Workflow-Sprachen um somit das Schachteln von Transaktionen und das Aufweichen einzelner ACID-Eigenschaften (ACID = Atomicity, Consistency, Isolation, Durability) zu ermöglichen (ACID-Eigenschaften siehe z.B. [Rechenberg et al_97, S. 765], erweiterte Transaktionskonzepte siehe z.B. [Leymann_97a]). Somit sollte z.B. das selektive Zurücksetzen von Teilschritten erlaubt sein und die Isolation teilweise aufgehoben werden. (näheres in [Jablonski et al_97a, S. 228 ff.]).

Analysierbarkeit
Dieser Bereich der nichtfunktionalen Anforderungen an ein WfMS bezieht sich auf die Überwachung und Analyse der Workflow-Ausführung, der Überwachung des WfMS und die Nachvollziehbarkeit.

Zur Überwachung eines WfMS müssen Mechanismen zur Protokollierung oder Führung eines Logs zur Verfügung gestellt werden, um das Verhalten des Systems jederzeit nachvollziehbar und beobachtbar zu machen (z.B. Visualisierung des Systemverhaltens durch Monitore zur Identifikation von Fehlverhalten, Engpässen oder Sicherheitsrisiken).

Damit sichergestellt ist, daß immer mit den neuesten Versionen der Workflows gearbeitet wird, bedarf es eines Versionierungsmechanismuß. Unter einem Versionierungsmechanismuß versteht man ein Konzept, mit dem man unterschiedliche Ausführungen (Versionen) eines Produktes (in diesem Fall eines Workflow-Modells) protokollieren kann. Dieses Konzept gestattet in Verbindung mit Überwachung und Aufzeichnung der Workflow-Ausführung den Aufbau einer Historie, wodurch es ermöglicht wird, den Ablauf abgearbeiteter Workflows zu analysieren, zu rekapitulieren und die Evolution des Modells zu beschrieben. Somit wird es möglich, verschiedene Versionen eines Workflows hinsichtlich ihrer Effizienz (durchschnittliche Gesamtlaufzeit, durchschnittliche Bearbeitungsdauer einzelner Aufgaben, Ausnutzung der vorhandenen Ressourcen, usw.) miteinander vergleichen zu können und eventuelle Verbesserungen oder Verschlechterungen zu identifizieren. Die gewonnenen Ergebnisse können dann in eine neu überarbeitete, verbesserte Version einfließen.

Anforderungen an die Applikationsintegration

Da die Applikationsintegration sowohl funktionale als auch nichtfunktionale Anforderungen an ein WfMS stellt (vgl. [Jablonski et al_97a, S. 225]), ist dieser Problematik hier ein eigener Abschnitt gewidmet. Die Ausführungen richten sich nach [Jablonski et al_97a, S. 367 ff.].

Unter Applikationsintegration versteht man die Einbindung von externen Applikationen in die Workflow-Ausführung. Man unterscheidet zwischen der unidirektionalen und der bidirektionalen Integration. Die unidirektionalen Integration beschränkt sich auf den Aufruf der Applikation durch das WfMS, wobei die Übergabe von Parametern und Rückgabewerten möglich sein sollte. Die bidirektionale Integration bezieht sich auf die Zusammenarbeit von Applikation und WfMS wobei die beiden Systeme als gleichberechtigte Kommunikationspartner gesehen werden. Die Kommunikation kann also von beiden Partnern ausgehen. Somit kann man in diesem Fall auch von Interoperabilität sprechen.

Applikationen dienen der Durchführung von Aufgaben, wobei darauf geachtet werden muß, daß bis dato verwendete Applikationen nach Einführung des WfMS ebenfalls noch zur Aufgabenerledigung zur Verfügung stehen. Dies kann zum einen mit der Investitionssicherung des Unternehmens und zum anderen mit der Benutzerakzeptanz begründet werden. Hinsichtlich der Wahl der zur Aufgabenerledigung angebotenen Programme ist zu sagen, daß diese verschiedene Kriterien der Software-Ergonomie erfüllen sollten. Stichworte sind hier Aufgabenangemessenheit der Programme und Wahlmöglichkeiten hinsichtlich der zu verwendenden Programme seitens der Benutzer, was als Handlungsspielraum bezeichnet wird und dem Benutzer ermöglichen soll, unterschiedliche Lösungsstrategien zur Erledigung seiner Aufgabe auswählen zu können (vgl. z.B. [Balzert et al_88]). Der Begriff der Aufgabenangemessenheit zielt darauf ab, daß die Funktionalität eines Programms den Benutzer in die Lage versetzen soll, eine ihm gestellte Aufgabe zu lösen. Dabei sollte darauf geachtet werden, daß die Funktionalität nicht zu mächtig ist, da sonst die Wahrscheinlichkeit der Fehlbedienung des Programms und Fehlbearbeitung der Aufgabe steigt. Die Funktionalität sollte allerdings auch nicht zu eingeschränkt sein, da es sonst zu einer zu strengen Führung und quasi Bevormundung der Benutzer kommen würde. „Damit würden die Benutzer an einer sinnvollen Nutzung ihrer Fachqualifikation und am Aufbau

einer zusätzlichen modalen Qualifikation (Fähigkeit das System zu handhaben) gehindert.“ [Maaß_94, S. 9]
Dem Benutzer muß ermöglicht werden, alle zur Durchführung seiner Aufgaben benötigten Programme von seinem Arbeitsplatz aus zugreifen zu können. Dies führt zu der Anforderung der Möglichkeit des entfernten Applikationsaufrufs.

Von den Applikationen wird der zuverlässige Programmaufruf als auch die zuverlässige Programmausführung gefordert, welches vor allem in verteilten Systemen den Einsatz zusätzlicher, die Kommunikation und Ausführung sichernder, Technologien verlangt.
Entfernter Aufruf und Zuverlässigkeit zählen zu den nichtfunktionalen Anforderungen, die anderen genannten zu den funktionalen.

4.2.2 Architekturmodelle von Workflow-Management-Systemen

Im folgenden soll keine konkrete Architektur von WfMS vorgestellt werden (siehe dazu Kapitel 4.3), sondern eine allgemeine Basisstruktur, welche die wichtigsten Komponenten identifiziert und deren Beziehungen untereinander verdeutlicht. Diese Basisstruktur beschreibt eine Architektur auf abstraktem Niveau und durch Verfeinerung kommt man zu sogenannten Implementierungsarchitekturen, die als Implementierungsgrundlage für konkrete Systeme verwendet werden können (vgl. [Jablonski_97b]). Die Basisstruktur (hier als funktionale Komponentenarchitektur bezeichnet) stellt das Ergebnis der Systemdefinition dar, wobei die Implementierungsarchitektur als Ergebnis des Entwurfs innerhalb der Aktivitäten bei der Softwareentwicklung aufgefaßt werden kann (vgl. [Rechenberg et al_97, S. 656]).

Funktionale Komponentenarchitektur

Die funktionale Komponentenarchitektur von WfMS beschreibt die prinzipiell benötigten Dienste, die von solchen Systemen angeboten werden sollten.

Aus den in Kapitel 4.1 dargestellten funktionalen Aspekten und den einzelnen Werkzeugen der Benutzerschnittstelle ergeben sich die funktionalen Komponenten der Architektur, wobei jedem Aspekt und jedem Werkzeug eine Komponente zugeordnet werden kann. Die Aspektkomponenten und die Werkzeuge lassen sich dann noch einmal gruppieren und man kommt zu der in Abbildung 15 zu sehenden Darstellung. Die Komponenten, welche die Aspekte realisieren, bilden das Kernsystem eines WfMS (auch als Workflow-Engine bezeichnet) und die Werkzeuge das Werkzeugsystem. Das Kernsystem kann noch einmal in die Bereiche Kern und Schale aufgeteilt werden, wobei die Koordinationskomponente den Kern und die anderen Komponenten die Schale repräsentieren (vgl. [Jablonski_97b]). Es bestehen Kommunikationsbeziehungen sowohl zwischen den Komponenten der Workflow-Engine als auch zwischen der Workflow-Engine und den Werkzeugen. Um die Abhängigkeiten der die Aspekte realisierenden Komponenten möglichst gering zu halten, kommunizieren diese Komponenten nicht direkt miteinander, sondern über die sogenannte Koordinationskomponente. Dadurch wird die Änderbarkeit erhöht und das Konstruktionsprinzip der wenigen und schmalen Schnittstellen eingehalten (vgl. [Meyer_90]).

Wie bereits in Kapitel 4.2.1 erwähnt, sind natürlich, je nach Anforderungen an die jeweiligen Systeme, weitere Aspekte zu berücksichtigen, was auch in Abbildung 15 angedeutet wird.
Im folgenden werden Möglichkeiten dargestellt, wie auf Grundlage dieser Komponentenarchitektur eine Implementierungsarchitektur aufgebaut werden kann (wobei es auch Mischformen geben kann).

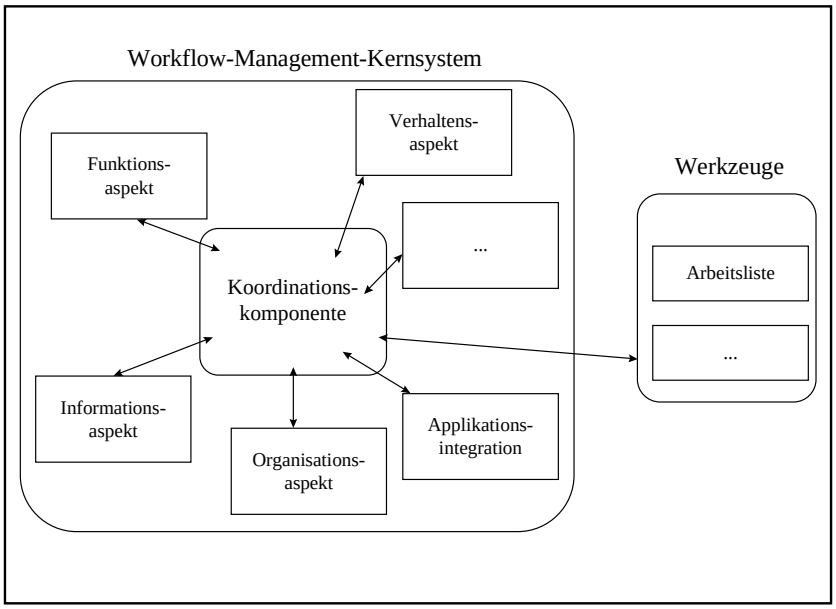


Abbildung 15: Funktionale Komponentenarchitektur eines WfMS [Jablonski et al_97a, S. 235]

Mögliche Implementierungsarchitekturen

In [Jablonski et al_97a, S. 236 ff.] werden drei mögliche Implementierungsarchitekturen vorgestellt.

Der erste Architekturvorschlag (Abbildung 16, A), der als **Implementierungsarchitektur auf Basis unabhängiger Aspektmodule** bezeichnet wird, orientiert sich sehr nah an der oben dargestellten funktionalen Komponentenarchitektur. Hierbei wird jede funktionale Komponente durch ein (oder mehrere) voneinander unabhängige Aspektmodule realisiert (keine gemeinsamen Daten, kein Zugriff auf andere Aspektmodule), welche die geforderten Leistungen erbringen. Die Kommunikation findet nicht zwischen den Aspektmodulen statt, sondern über eine ausgezeichnete Komponente, die als Koordinationsmodul bezeichnet wird. Das Koordinationsmodul stellt die aktive Komponente der Architektur dar und die Aspektmodule die passiven, die durch die Koordinationskomponente aufgerufen und koordiniert werden. Es existiert ein Benachrichtigungsmodul, welches die Verwaltung der Arbeitslisten übernimmt und ebenfalls mittels Koordinationskomponente koordiniert wird.

Der zweite Ansatz (Abbildung 16, B), als **Implementierungsarchitektur mit ausgezeichnetem Aspekt** bezeichnet, unterscheidet sich vom ersten dadurch, daß die einzelnen Aspekte nicht mehr gleichberechtigt betrachtet werden. Es existieren ein oder mehrere ausgezeichnete Aspekte, welche die Ausführung der Workflows hauptsächlich bestimmen. Somit entfällt ein ausgezeichnetes Koordinationsmodul, da dessen Aufgaben nun von dem (oder den) ausgezeichneten Aspektmodul übernommen werden und die untergeordneten Aspekte aufruft.

Der dritte Ansatz (Abbildung 16, C), als **Implementierungsarchitektur mit Workflow-Interpreter bezeichnet**, unterschiedet sich grundlegend von den beiden ersten. Hier wird eine sogenannte operationale Zwischensprache (formale Sprache, für die eine Ausführungssemantik existiert) zur Darstellung der Aspekte verwendet. Die tatsächliche Ausführung wird durch ein Modul realisiert, welches in der Lage ist, die operationale

Zwischensprache zu interpretieren. Somit werden Aspektmodule nicht benötigt und das Interpretationsmodul kann als Koordinationskomponente gesehen werden. Hilfsmodule werden nur zum Applikationsaufruf oder zur Abbildung eines Workflow-Modells in eine Zwischensprache benötigt.

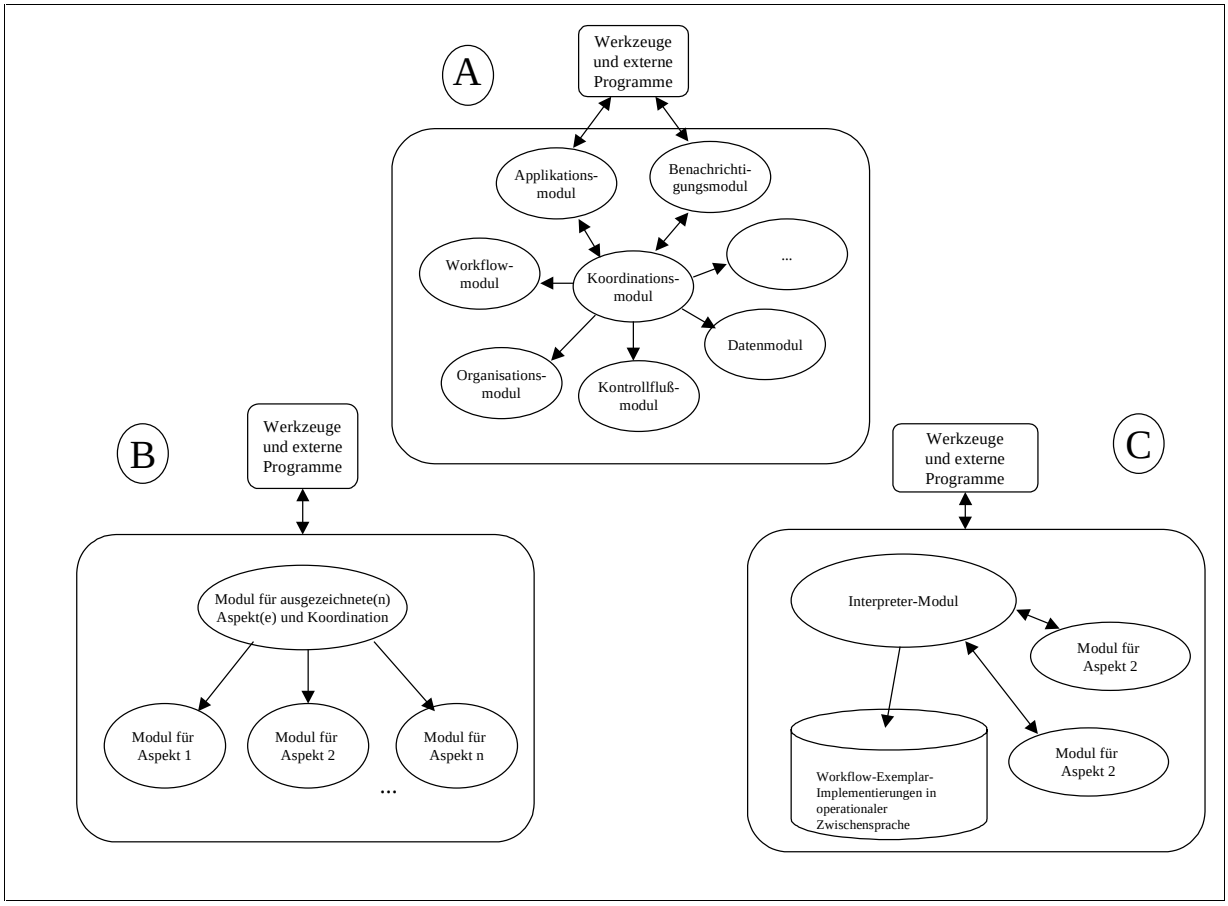


Abbildung 16: Implementierungsarchitekturen auf Basis unabhängiger Aspektmodule(A), mit ausgezeichneten Aspekt (B) und Workflow-Interpreter (C). [Jablonski et al_97a, S. 236 ff.]

Die Anbindung von Werkzeugen ist nur in Vorschlag eins konkreter dargestellt. Hier kann die Anbindung über weitere Module analog zum Benachrichtigungsmodul erfolgen. In Vorschlag zwei und drei werden hierzu keine weiteren Aussagen gemacht.

Hinsichtlich der Beurteilung der verschiedenen Vorschläge ist zu sagen, daß sich der erste und zweite Vorschlag jeweils durch Modularität auszeichnet und dadurch die Änder- und Erweiterbarkeit erhöht ist. Beim zweiten beschränkt sich diese Eigenschaft allerdings nur auf die Aspektmodule, welche die untergeordneten Aspekte repräsentieren. Das ausgezeichnete

Modul kann sehr komplex sein und somit die Änderbarkeit erschweren oder sogar unmöglich machen. Beim dritten Vorschlag hängt die Änder- und Erweiterbarkeit davon ab, ob die verwendete Zwischensprache hinreichend mächtig bezüglich der Formulierbarkeit der beiden betrachteten Gesichtspunkte ist.

4.3 Die Architektur von FlowMark

Nachdem im vorherigen Kapitel Vorschläge für Implementierungsarchitekturen von WfMS und die an sie gestellten Anforderungen vorgestellt wurden, soll in diesem Kapitel die Architektur von FlowMark als konkrete Implementation eines WfMS beschrieben werden (vgl. [Leymann et al_94]). Hierbei wird auch untersucht, in wieweit FlowMark sich konform zu den gestellten funktionalen und nichtfunktionalen Anforderungen verhält.

Die Architektur von FlowMark wird als 3-stufige Client/Server-Architektur bezeichnet (vgl. [Leymann_97a]). Auf der höchsten Ebene teilt sich die Architektur in zwei Bereiche auf, die als Build-Time und Run-Time bezeichnet werden. Innerhalb jedes Bereiches ist dann noch einmal zwischen einer Server- und eine Client-Komponente zu unterscheiden.

4.3.1 Die FlowMark Build-Time Architektur

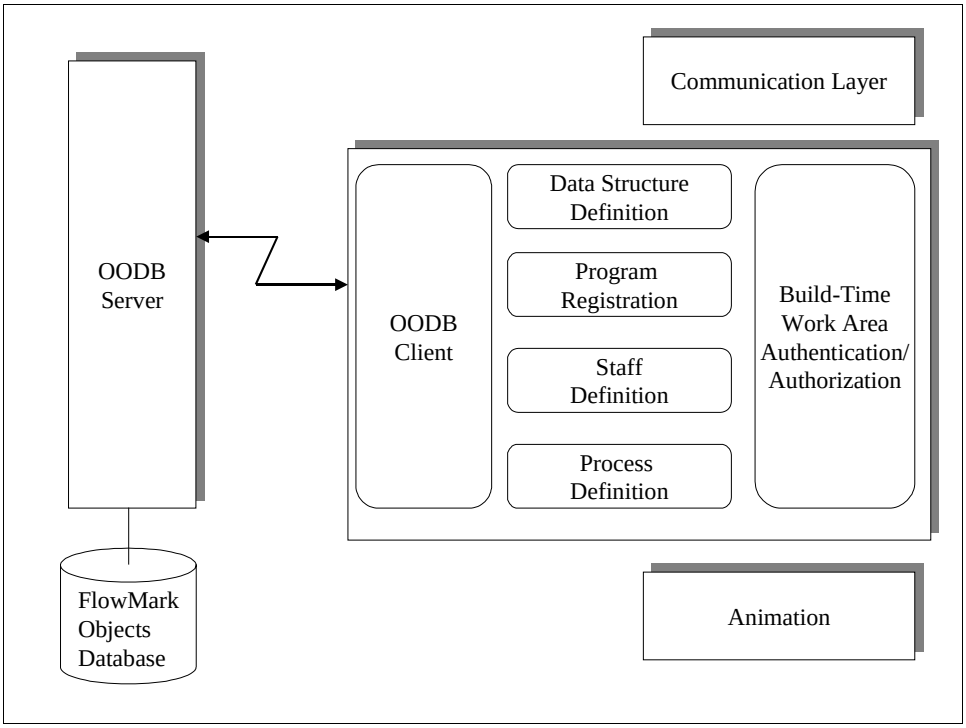


Abbildung 17: FlowMark Build-Time Architektur [Leymann et al_94, S. 333]

Die FlowMark **Build-Time** Architektur wird in Abbildung 17 dargestellt. Innerhalb der Build-Time findet die Modellierung der Workflows auf der Client-Seite statt. Alle hierzu benötigten Informationen werden auf der Server-Seite in einer Datenbank verwaltet, die sich potentiell auf einem anderen System befinden kann. Die Kommunikation von Client und Server wird über eine Kommunikationskomponente realisiert, die als **Communication Layer** bezeichnet wird. Sie abstrahiert von dem verwendeten Kommunikationsprotokoll (TCP/IP oder APPC) und ist als Messaging Service implementiert. Sowohl der synchrone als auch der asynchrone Austausch von Nachrichten ist möglich. Die Server-Seite besteht aus dem **OODB-Server** und einer objektorientierte Datenbank von „ObjectStore“. Der OODB-Server überwacht und synchronisiert die entfernten Zugriffe auf die Datenbank. In der Datenbank selbst werden alle innerhalb des Systems relevanten Informationen gehalten. Diese Informationen betreffen abgearbeitete Prozesse, aktive Prozeß-Exemplare und Umgebungsinformationen (Staff und Applikationen). Auf der Client-Seite übernimmt die als **OODB-Client** bezeichnete

Komponente die Kommunikationsaufgabe. Es existieren weiterhin ausgezeichnete Komponenten für die Definition von Datenstrukturen (**Data Structure Definition**), der Aufbauorganisation (**Staff Definition**), von Prozessen (**Process Definition**) und für die Registration von Programmen (**Program Registration**). Die einzelnen Komponenten kommunizieren ebenfalls über die Communication Layer um Informationen über veränderte Zustände auszutauschen. Die grafische Bedienoberfläche (**Build-Time Work Area**) zusammen mit der **Authentication/Authorization** bildet eine weitere Komponente der Build-Time-Architektur. Über die grafische Bedienoberfläche können die einzelnen Komponenten der Build-Time vom Benutzer angesprochen werden. Die Authentication/Authorization-Komponente stellt sicher, daß nur die Personen auf die Build-Time zugreifen können, die hierzu auch berechtigt sind. Die Animationskomponente wird in 4.3.3 detailliert beschrieben.

4.3.2 Die FlowMark Run-Time Architektur

Die FlowMark **Run-Time** Architektur wird in Abbildung 18 dargestellt. Die Server-Seite beinhaltet hier ebenfalls den **OODB-Server** und die „Object-Store“-Datenbank. Des weiteren existiert ein sogenannter **Run-Time-Server**. Dieser braucht nicht auf dem selben System installiert zu sein, wie der OODB-Server. Aus diesem Grund beinhaltet die Komponente auch einen **OODB-Client**, der die Verbindung zum OODB-Server herstellt. Die weiteren Komponenten des Run-Time-Servers sind der Programmausführungs-Server (**Program Execution Server)**, der Arbeitslisten-Server (**Work List Server)** und die Prozessausführung (**Process Execution)**. Die Aufgabe des Program Execution Servers besteht darin, die Ausführung von registrierten Programmen sicherzustellen. Er hat das System zu bestimmen, auf dem das entsprechende Programm auszuführen ist. Die dazu notwendigen Informationen erhält er aus den Daten der Programm-Registration, die zur Build-Time festgelegt werden und in der OODB verfügbar sind. Hierbei kann es sich um Programme handeln, die sich auf dem System befinden, mit dem der Benutzer gerade arbeitet, die Programme können sich allerdings auch auf einem anderen System befinden (**remote program execution**). Der Work List Server verwaltet die Arbeitslisten der Benutzer und versorgt sie mit auszuführenden Aktivitäten. Die Process Execution überwacht die Ausführung der Prozesse entsprechend der zugrundeliegenden Workflow-Modelle. Seine Aufgabe besteht vorwiegend in der Auswertung von Transitionsbedingungen, der Aktivierung und Deaktivierung von Kontroll-Konnektoren und des Anstoßens von Zustandsänderungen der Aktivitäten (vgl. auch [Alonso et al_97b]). Client und Server kommunizieren ebenfalls über die Communication Layer.

Der Client verfügt über die Komponenten Arbeitslisten-Verwaltung (**Work List Handler**), Prozeßliste (**Process List**), **Run-Time Work Area und Authetication/Authorization**, **Program Execution, Program API und Process API** (API = Application Programming Interface). Die Aufgabe des Work List Handler, der mit dem Work List Server kommuniziert, besteht darin, die Arbeitslisten des jeweiligen Benutzers zu verwalten (ein Benutzer kann mehrere besitzen, die er an seine Bedürfnisse anpassen kann). Hierzu gehört der Eintrag neuer auszuführender Aktivitäten, das Löschen beendeter Aktivitäten, das Transferieren von Aktivitäten zwischen Arbeitslisten usw. Die Process List verwaltet Schablonen (Templates) und Exemplare aller Prozesse, zu dessen Ausführung der jeweilige Benutzer berechtigt ist, bzw. die er sehen darf. Ein Exemplar darf ein Benutzer sehen, wenn er es erzeugt hat oder wenn er der Administrator des Prozesses ist. Die Run-Time Work Area realisiert die grafische Bedienoberfläche und somit den Zugang des Benutzers zu der Prozeß- und Arbeitsliste. Program Execution, Program API und Process API dienen der Applikationsintegration. Die

Program Execution Komponente kommuniziert mit dem Program Execution Server und ruft lokal Programme (evtl. mit Parametern) auf, die die Program-Aktivitäten implementieren. Die Program API ermöglicht es Programmen, auf Daten-Container von Prozessen, Blöcken und Aktivitäten innerhalb von Prozessen zuzugreifen. Über die Process API ist es Applikationen möglich, Prozesse von FlowMark starten, unterbrechen, abbrechen und beenden zu lassen. Die Authentication/Authorization stellt sicher, daß nur berechtigte Personen zur FlowMark Run-Time Zugang erhalten.

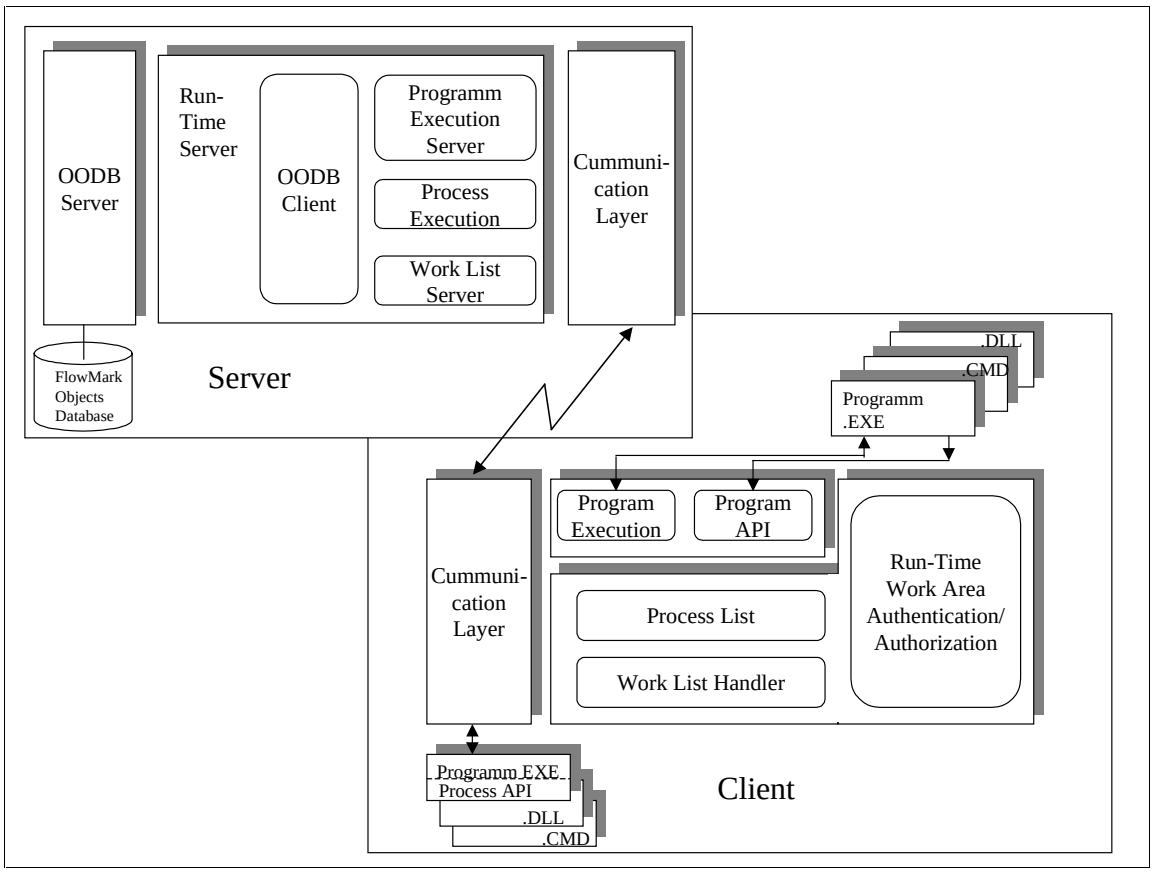


Abbildung 18: FlowMark Run-Time Architektur [Leymann et al_94, S. 334]

4.3.3 Hilfsmittel von FlowMark zur Build-Time und zur Run-Time

In diesem Kapitel sollen die Hilfsmittel vorgestellt werden, die FlowMark den Benutzern dieses Systems zur Verfügung stellt. Es handelt sich hierbei um Werkzeuge die es ermöglichen, Informationen über die von FlowMark verwalteten Prozesse zu erhalten, und zwar während ihrer Modellierung (Build-Time) und ihres Ablaufes (Run-Time). Diese Informationen beziehen sich z.B. auf die Korrektheit der Prozeßmodelle oder deren Ausführungszustand.

Animationstool

Um schon zur Build Time die modellierten Prozesse zu testen, stellt FlowMark ein Animationstool zur Verfügung. Dieses Tool ermöglicht es dem Entwickler, die Ausführung der Prozesse zu simulieren, um auf diesem Wege eventuelle Fehler und Schwachstellen des Prozeßmodells vor dessen Übersetzung zu identifizieren. Die getesteten Prozesse brauchen weder vollständig modelliert zu sein, noch müssen die später zu benutzenden Applikationen auf dem verwendeten System zur Verfügung stehen. Das Animationstool kann also auch als

Debugger aufgefaßt werden, wie man ihn aus herkömmlichen Programmierumgebungen kennt.

Während der Animation (als ‘Animation Session’ bezeichnet; [Leymann_97b] spricht auch von der Verwendung im ‘Debugging Mode‘) werden von dem Tool eventuell fehlende Informationen vom Benutzer angefordert (z.B. Informationen, die zur Auswertung von Abbruchbedingungen benötigt werden), die eingenommenen Prozeßzustände aufgezeichnet und das Fortschreiten des Prozesses unter Verwendung einer grafischen Repräsentation dargestellt. Zur Darstellung werden die von der Modellierung bekannten Bausteine verwendet, wobei der momentane Zustand des Prozesses mittels farblicher Kennzeichnung realisiert wird. So ändert sich z.B. die Farbe des verwendeten Kontrollfluß-Konnektors von schwarz auf grün, ausgelassene Pfade werden rot gekennzeichnet.

Die zur Ausführung des Workflows vorgesehenen Personen, Rollen und Organisationen werden jeweils mittels einer eigenen Arbeitsliste repräsentiert. Auf dieser Arbeitsliste werden während des Animationsablaufes die auszuführenden Aktivitäten plaziert. Der die Animation ausführende Entwickler kann dann die Aktivitäten selektieren, ausführen und Attribute abfragen. Falls innerhalb des Prozesses Aktivitäten auftauchen, die noch keinem Beteiligten zugeordnet sind, werden sie in die ebenfalls vorhandene Arbeitsliste des Administrators eingetragen. Somit kann diese Art von Modellierungsfehler schnell entdeckt werden.

Wenn der Entwickler eine Aktivität ausführt, öffnet sich zunächst ein Fenster, in dem die Eingabe- und Ausgabedatenstruktur der Aktivität in Form eines sogenannten ‘Data Container Animationsobjektes’ dargestellt wird. Es besteht nun die Möglichkeit, die Werte der einzelnen Member der Eingabedatenstruktur zu betrachten und die Member der Ausgabedatenstruktur zu setzen. Die eingegebenen Daten werden dann zur Auswertung von Abbruchbedingungen bzw. Transitionsbedingungen verwendet und der Animationsprozeß ergebnisabhängig fortgeführt (entlang der Kontrollfluß-Konnektoren, deren Transitionsbedingung zu wahr evaluiert wird).

Während einer ‘Animation Session’ ist es möglich, an schon ausgeführte Aktivitäten zurückzuspringen und andere Werte der Ausgabedatenstruktur festzulegen. Auf diese Art und Weise können eventuell unterschiedliche Ausführungswege nacheinander verfolgt werden.

Daten über den Verlauf der Animation werden vom FlowMark Workflow Manager in einem sogenannten ‘Message Log’ (auch als ‘Trace’ bezeichnet) festgehalten (vgl. [IBMa_96]). Die aufgezeichneten Daten können dann für einen erneuten, automatischen Animationslauf verwendet werden, z.B. zu Demonstrationszwecken (Verwendung des Animations-Tools im sogenannten ‘Regression Mode‘; vgl. [Leymann_97b]) .

Audit Trail

Dieses von FlowMark zur Verfügung gestellte Hilfsmittel ermöglicht die Aufzeichnung von Informationen zur Laufzeit von Prozeßexemplaren. Für jedes gestartete Exemplar eines Prozesses kann ein Audit Trail erzeugt werden. Die in diesem File im ASCII-Format festgehaltenen Informationen beziehen sich auf Zeitpunkte von Ereignissen, Ereignistypen (z.B. Prozeß gestartet, Prozeß beendet, Benachrichtigung versendet, usw.), Informationen zu Prozessen (Zeitstempel, Name, Top-Level-Prozeß, Sub-Prozeß, Name des Prozeßmodels, usw.), Informationen zu Aktivitäten (User ID des Starters, Aktivitätentyp, Aktivitätenstatus, Programmname, u.a.), usw. Diese als „audit trail records“ bezeichneten Informationen können z.B. in einer relationalen Datenbank gespeichert und mittels SQL-Abfragen ausgewertet werden. (vgl. [IBM_96c, S. 425 ff.])

Sinn und Zweck der Sammlung dieser Informationen kann darin gesehen werden, daß so logistische Probleme identifiziert werden können. Es kann festgestellt werden, ob z.B. eine Person mit unverhältnismäßig vielen oder zu wenigen Aufgaben, im direkten Vergleich zu anderen an dem Prozeß beteiligten, versorgt wird. (vgl. [IBMa_96, S. 148])

So ist es z.B. möglich, zu verschiedenen Versionen eines Workflows jeweils Audit Trails anzulegen und diese dann auszuwerten und zu vergleichen um festzustellen, ob z.B. eine neue Version die erhofften Verbesserungen gebracht hat oder nicht. FlowMark bietet allerdings keine Möglichkeit an, Audit Trails zu visualisieren. Außerdem ermöglicht die Archivierung von Audit Trails, daß Abläufe und Entscheidungsprozesse auch nach längerer Zeit noch nachvollzogen werden können. Dies ist in Behörden nach [Böhm et al_95] Zwang und wird auch in zunehmenden Maße von Unternehmen verlangt (ISO 900x-Zertifizierung).

Process Monitor
Mit Hilfe dieses Werkzeugs ist es möglich den Bearbeitungszustand eines Prozeßexemplars festzustellen. Der Bearbeitungszustand wird dem Benutzer in grafischer Form präsentiert, welche der Darstellungsform entspricht, die zur Prozeßmodellierung verwendet werden kann.

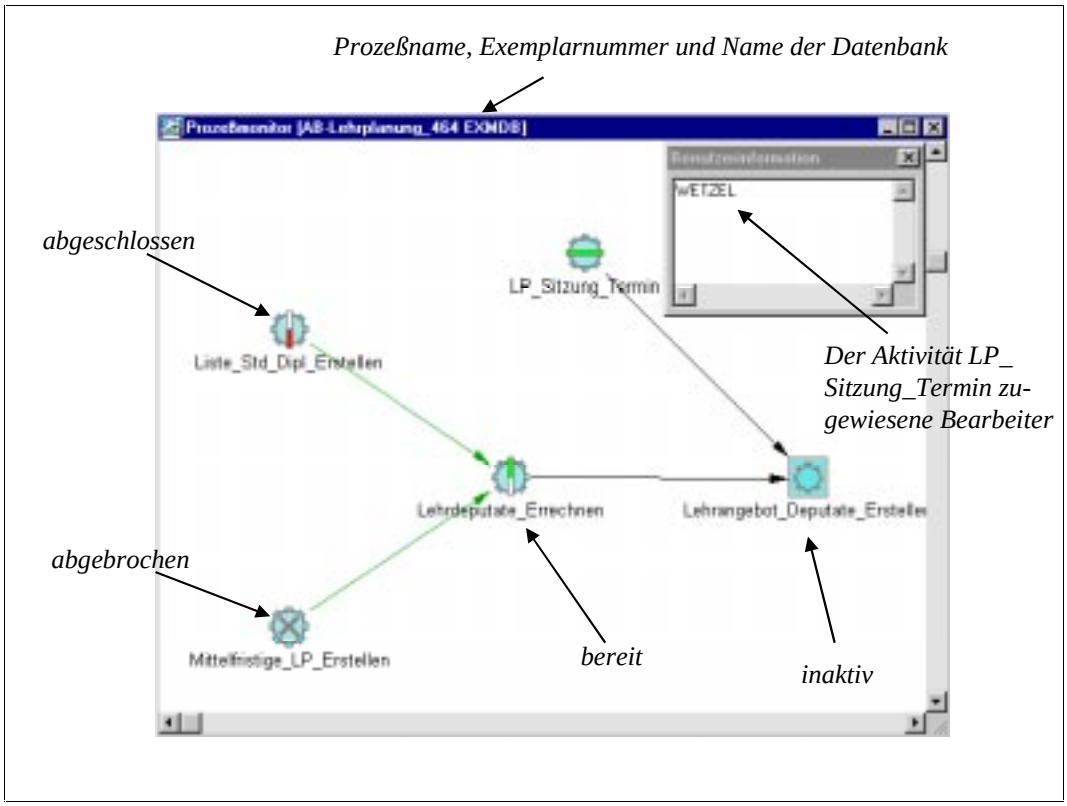


Abbildung 19: Process Monitor zum Sub-Prozeß „AB-Lehrplanung“

Abgearbeitete, gerade bearbeitete und noch zu bearbeitende Aktivitäten können durch unterschiedliche Symbole erkannt werden (vgl. Abbildung 19). Durch Doppelklick auf eine der schon bearbeiteten bzw. aktiven Aktivitäten sind die zugewiesenen Bearbeiter feststellbar (User ID wird angezeigt). Der Process Monitor kann von jedem der an der Prozeßausführung beteiligten Personen aufgerufen werden. (vgl. [IBMb_96, S. 59ff.]) Es ist zu beachten, daß Sub-Prozesse oder übergeordnete Prozesse nicht eingesehen werden können.

Mit Hilfe dieses Werkzeugs kann ein Benutzer z.B. feststellen, wer eine Aufgabe vor ihm im Prozeßverlauf bearbeitet hat. Falls er nun zur Bearbeitung seiner momentanen Aufgabe

Informationen von seinem Vorgänger benötigt, so kann er mit diesen in Kontakt treten (außerhalb von FlowMark, da innerhalb von FlowMark hierfür keine Möglichkeit vorgesehen ist). Man könnte den Process Monitor auch dazu verwenden, alle Bearbeiter ausfindig zu machen, denen eine Aktivität zugewiesen wurde. Falls Termindruck besteht, könnte gezielt mit den Bearbeitern in Kontakt getreten werden, um sie zur bevorzugten Bearbeitung der jeweiligen Aufgabe aufzufordern.

Notification Service

Es ist möglich, für den gesamten Prozeß und für jede der darin definierten Aktivitäten festzulegen, innerhalb welchen Zeitraums diese abgearbeitet werden müssen. Es ist zu beachten, daß keine Deadline angegeben werden kann, sondern nur eine Gesamtlaufzeit. Das Konzept der Deadline kann allerdings unter Verwendung der API-Funktionen (vgl. Kapitel 4.3.4) simuliert werden, indem man erst zur Laufzeit die Gesamtlaufzeit in Abhängigkeit der noch verbleibenden Zeit bis zur Deadline setzt. Wird der festgelegte Zeitraum überschritten, so sorgt FlowMark mittels des Notification Service dafür, daß eine ebenfalls zur Build Time festgelegte Person (Prozeßadministrator, Manager, Koordinator oder eine bestimmte Person) über die Überschreitung des Zeitlimits informiert wird. Hierzu trägt das System in der Arbeitsliste der benachrichtigten Person eine besondere Aktivität ein, welche die Überschreitung kennzeichnet. Das in diesem Fall verwendete Symbol ist eine Glocke mit einer Bimmel. Anhand des Aussehens der Glocke kann der Zustand der Aktivität festgestellt werden (mögliche Zustände sind: Ready, Running oder Suspended). Durch Doppelklick auf dieses Symbol kann sich die benachrichtigte Person weitere Informationen über die Aktivität verschaffen. Wenn die Aktivität noch nicht gestartet wurde (Zustand Ready), so wird dem Benachrichtigten eine Liste aller Personen zur Verfügung gestellt, denen die Aktivität zugewiesen wurde. Befindet sich die Aktivität in dem Zustand Running, so wird die Identität des Starters der Aktivität angezeigt. Ist die Aktivität im Zustand Suspended, so wird die Identität des Starters des Prozeßexemplares zu dem die Aktivität gehört angezeigt und die des Prozeßadministrators (sofern einer definiert ist). Falls der Benachrichtigte über die Berechtigung verfügt auf die Arbeitsliste des oder der Verursacher der Benachrichtigung zuzugreifen, so kann er die betroffene Aktivität auf seine Arbeitsliste übertragen und bearbeiten. Liegt die Berechtigung nicht vor, so muß eine Regelung außerhalb des Systems gefunden werden.

Es ist außerdem möglich, einen zweiten Zeitraum festzulegen. Dieser Zeitraum ist relevant, wenn trotz Versenden der ersten Benachrichtigung die Aktivität immer noch nicht bearbeitet worden ist. In diesem Fall erhält der Prozeßadministrator die Benachrichtigung. In dessen Arbeitsliste erschient nun ein Glockensymbol mit zwei Bimmeln. Gibt es keinen Prozeßadministrator, so geht die Benachrichtigung an den Prozeßstarter. (vgl. [IBM_96a, S. 101] und [IBM_96b, S. 47ff.])

Der Notification Service ist vergleichbar mit dem Konzept der Ausnahmebehandlung, welches in modernen Programmiersprachen Verwendung findet. „Ausnahmen signalisieren, daß eine Vorbedingung einer Operation verletzt wurde, und die Operation daher nicht ordnungsgemäß zu Ende geführt werden konnte.“ [Rechenberg et al_97, S. 393] Ausnahmebehandlung dient dazu, Ausnahmen abzufangen und geeignet auf ihr Auftreten zu reagieren[8]. Der Vergleich des Notification Service mit diesem Konzept bietet sich an, da ein überschreiten des Zeitraums innerhalb der eine Aktivität oder ein Prozeß bearbeitet sein muß,

[8] Mechanismen, wie auf Ausnahmen geeignet reagiert werden kann, werden z.B. in [Felten_98] zusammenfassend dargestellt.

der Verletzung einer Vorbedingung entspricht, auf die in geeigneter Art und Weise (versenden einer Benachrichtigung) reagiert wird.

Mittels des Notification Service kann z.B. verhindert werden, daß Aufgaben unerledigt bleiben und dadurch Aufträge nicht termingerecht erledigt werden können. Bei Überschreitungen der Zeitvorgaben können Verantwortliche benachrichtigt werden, die dann eingreifend handeln. Somit ist es z.B. auch möglich, überforderte Bearbeiter zu identifizieren, falls eine Auslösung der Benachrichtigungen regelmäßig von ihnen ausgeht. Hierdurch könnten Rückschlüsse auf die Qualifikation der Bearbeiter oder aber auf deren Überlastung und somit auf ein unpassendes Workflow-Modell gezogen werden, da das Aussenden (mit Verursacher) auch im Audit Trail (siehe weiter oben) festgehalten wird.

4.3.4 Applikationsintegrationsmöglichkeiten von FlowMark

Laut [IBM_96d] bietet FlowMark verschiedene Möglichkeiten der Applikationsintegration an. Es ist möglich, von FlowMark aus Fremdprogramme (Legacy Programme) aufzurufen. Diese müssen vorher mittels der **Programm-Registration** erfaßt worden sein. Dem aufgerufenen Programm können Parameter mitgegeben werden, wobei die Rückgabe von Informationen an FlowMark vom Programm aus nicht möglich ist. [Leymann et al_94] spricht in diesem Zusammenhang von Applikationen, die sich nicht über die Existenz von FlowMark bewußt sind („not FlowMark-aware“).

Es werden **Programmierschnittstellen** (API = Application Programming Interface) für eine Vielzahl von Programmiersprachen, in Abhängigkeit vom verwendeten Betriebssystem, zur Verfügung gestellt. Über diese APIs ist es u.a. möglich, die Ausführung von Prozessen zu kontrollieren und auf die Datencontainer der Prozesse und Aktivitäten zuzugreifen (vgl. Process- und Program-API in Abbildung 7, Kapitel 4.3.2). Für Windows 3.1 und Windows NT werden APIs für C, C++ und VisualBasic angeboten. Unter OS/2 kann man APIs für C, C++, REXX und COBOL nutzen und unter AIX APIs für C und REXX. Es wird zwischen zwei Arten von APIs unterschieden: FlowMark **language APIs** und FlowMark **workflow client APIs** (vgl. Tabelle 3).

	Language APIs	***Workflow client APIs***
Lesen/Schreiben von Daten-Containern	Ja	Ja
Prozeß-Exemplare von Schablonen erzeugen	Ja	Ja
Prozesse verwalten (starten, unterbrechen, beenden, usw.)	Ja	Ja
Auf einem Workflow-Server (und somit bei FlowMark) ein- und ausloggen	Nein	Ja
Arbeitslisten verwalten (erzeugen, löschen, aktualisieren, usw.)	Nein	Ja
Einträge (work items) von Arbeitslisten verwalten (Zustand abfragen, Endebedingung abfragen, löschen, auf andere Arbeitslisten transferieren, beenden, starten, usw.)	Nein	Ja
Attribute von Benutzern abfragen (User ID, Rollen, Organisation, Paßwort, Vertreter, usw.)	Nein (bis auf User ID)	Ja

Tabelle 3: Vergleich der Funktionalitäten der FlowMark language - und workflow client APIs

Die language APIs ermöglichen die eben genannten Aktionen, wobei die workflow client APIs weitergehende Funktionalitäten insbesondere hinsichtlich der Kontrolle von

Arbeitslisten und deren Inhalt (work items) anbieten. Man kann FlowMark durch Nutzung der workflow client APIs praktisch von außen steuern. Workflow client APIs gibt es allerdings nur für C, C++ und Lotus Notes, wobei die LotusNotes API auf die C++-API aufsetzt und in der Version 2.3 von FlowMark nur für OS/2 angeboten wird. Zur Verdeutlichung der unterschiedlichen Leistungsfähigkeit der beiden API-Gruppen sei auf Tabelle 3 verwiesen.

Eine weitere Integrationsmöglichkeit bezeichnet das **Service Broker Konzept**. Dieses Konzept wird aus Platzgründen hier nur ansatzweise dargestellt. Detaillierte Informationen finden sich in [IBM_96d, S. 2 ff.]. Bei diesem Konzept handelt es sich um ein Verfahren, bei dem eine Verbindung zu einer Server-Anwendung (z.B. LotusNotes) aufgebaut und über beliebige Zeit gehalten wird. Von dieser Server-Anwendung werden Dienste (Funktionen) angeboten, die von anderen Programmen, die von FlowMark verwaltet werden, genutzt werden können. Ein sogenannter Service Broker überwacht die Verbindung zum Dienstanbieter und leitet Anfragen an ihn weiter bzw. Ergebnisse an die Anfrager zurück. Sinn dieses Konzeptes ist es, zu verhindern, daß die den Dienstanbieter repräsentierenden Anwendungen bei jeder Anfrage neu gestartet werden müssen bzw. eine neue Verbindung etabliert werden muß. Die Verwendung dieses Konzeptes ist z.B. bei der Integration von Datenbanken oder Großrechneranwendungen nützlich. Somit wird auch verhindert, daß die Benutzer sich eventuell jedesmal neu als berechtigte Nutzer identifizieren müssen. Die Identifikation erfolgt einmal durch Anmeldung bei FlowMark und ist solange gültig, wie der Service Broker die Verbindung aufrechterhält[9]. Es ist möglich eigene Service Broker zu entwickeln. Falls Service Broker von den Herstellern der Legacy-Systeme zur Verfügung gestellt werden würden, wäre die bidirektionale Integration dieser Programme möglich. Zur Zeit stehen nur Service Broker für LotusNotes, VisualAge, VisualInfo und FlowMark selbst zur Verfügung (laut [IBM_96d]).

Zu guter Letzt sei noch die Möglichkeit der Erstellung von sogenannten Building Blocks erwähnt. Mit Hilfe dieses Konzeptes können Integrationsprogramme erstellt werden, die veröffentlichte Schnittstellen des zu integrierenden Produktes und von FlowMark verwenden. Von FlowMark wird ein solcher Building Block zur Verfügung gestellt, der die Kommunikation zwischen zwei FlowMark-Systemen via MQSeries ermöglicht (vgl. [IBM_96d, S. 178 ff.]). MQSeries ist eine nachrichtenorientierte Middleware von IBM, die Warteschlangen (Queues) zum Senden und Empfangen von Nachrichten verwendet (siehe [Felten_98]).

4.3.5 Diskussion der Architektur von FlowMark

Die Architektur von FlowMark stellt eine Implementierung des zweiten in Kapitel 4.2.2 vorgestellten Vorschlags einer **Implementierungsarchitektur mit ausgezeichnetem Aspekt** dar. Der ausgezeichnete Aspekt ist der Verhaltensaspekt. Dieser entscheidet bei FlowMark allein über die Ausführbarkeit von Aktivitäten. Er legt alle möglichen Ausführungsreihenfolgen innerhalb der Workflows fest und dominiert bzw. koordiniert die anderen Aspekte (vgl. [Jablonski et al_97a, S. 239]). Der Verhaltensaspekt wird mittels der Prozeßefinitions-Komponente zur Build-Time über die Kontroll-Konnektoren und die Transitionsbedingungen festgelegt, wobei die korrekte Ausführung zur Run-Time über die

[9] Der Service Broker meldet sich stellvertretend für die Dienstanfrager bei der Server-Anwendung an; die Identifikation des Benutzers gegenüber der Anwendung ist nicht mehr nötig, da die Berechtigung schon durch die Berechtigung zur Ausführung der Aktivität innerhalb von FlowMark gegeben ist.

Prozeßausführungs-Komponente durch Auflösung der Workflow-Definition sichergestellt wird.

Die anderen geforderten Aspekte sind ebenfalls implementiert. Der Informationsaspekt wird durch die Datendefinitions-Komponente, der Organisationsaspekt durch die Aufbauorganisationdefinitions-Komponente und der Operationsaspekt durch die Programm-Registrations-Komponente definiert. Alle Komponenten werden dann bei der Definition der Workflows durch die Prozeßdefinitons-Komponente benutzt. Der Funktionsaspekt wird ebenfalls durch die Prozeßdefinitions-Komponente festgelegt, die somit zwei Aspekte implementiert.

Die in Kapitel 4.2.1 geforderten **Schnittstellen** (Anwenderschnittstelle, Entwicklerschnittstelle und Administratorschnittstelle) werden über die Build- und Run-Time Work Areas realisiert (vgl. Abbildung 6 und Abbildung 7, Kapitel 4.3.1 und 4.3.2). Dem Benutzer werden alle erforderlichen Funktionalitäten zur Verfügung gestellt.

FlowMark kann als ein **eingeschränkt offenes System** bezeichnet werden, da es zwar auf verschiedenen Systemplattformen (Windows, Windows NT, OS/2, AIX, HP-UX) einsetzbar ist, aber nicht auf allen existierenden. Es basiert auf einer **Client/Server-Architektur**. Die Kommunikation der einzelnen Komponenten wird mittels einer Nachrichtenorientierten Middleware (Message Oriented Middleware = MOM) (vgl. [Felten_98]), die den Austausch sowohl synchroner als auch asynchroner Nachrichten ermöglicht, realisiert. Mittels der beiden APIs (Process und Program) wird eine Schnittstelle zur **Erweiterung des Systems** um neue Komponenten und zur **Werkzeugintegration** angeboten (Werkzeugintegration mittels FlowMark workflow clients).

Es besteht die Möglichkeit, FlowMark auf mehrere Domänen (z.B. Abteilungen) mit eigener Datenbank aufzuteilen (vgl. [IBM_96a, S. 108 ff.]). Die einzelnen Datenbanken bzw. Domänen können unter Verwendung des sogenannten Delivery-Servers miteinander kommunizieren. Dadurch wird es möglich, Teile eines Prozesses (Sub-Prozeß) innerhalb verschiedener Domänen ausführen zu lassen. Die **Skalierbarkeit** des Systems ist allerdings als eingeschränkt zu bezeichnen, da eine Änderung des physischen Aufbaus des Systems meist zu Änderungen des Prozeß-Modells führt. Welcher Sub-Prozeß innerhalb welcher Domäne ausgeführt werden soll, kann allerdings auch dynamisch zur Run-Time (über Eingabedaten-Container) festgelegt werden. Somit kann FlowMark zur Laufzeit an veränderte Umweltbedingungen hinsichtlich **Lastverteilung** angepaßt werden. Allerdings wird kein Werkzeug zur Lastermittlung von FlowMark zur Verfügung gestellt. Hier muß auf andere Hersteller zurückgegriffen werden.

Die Verteilung des Systems geschieht **nicht transparent**. Dies ist zum einen damit zu begründen, daß Benutzer sich explizit an einem bestimmten Server anmelden müssen. Zum anderen müssen die Programmierer, wenn sie zur Build-Time die Domäne bestimmen möchten, in welcher der jeweilige Sub-Prozeß ausgeführt werden soll, den Namen des Servers dieser Domäne kennen. Außerdem ist die manuelle Verteilung der Prozeß-Modelle auf die einzelnen Datenbanken der Domänen notwendig, damit dort die entsprechenden Sub-Prozesse instanziert und ausgeführt werden können.

Zur **zuverlässigen Ausführung** der Workflows wird der Mechanismus des Forward Recovery eingesetzt. Hierunter versteht man, daß die Ausführung im Falle eines Fehlers nach dessen Behebung dort fortgesetzt wird, wo der Fehler stattgefunden hat [Alonso et al_96]. Die aktuellen Zustände der jeweiligen Workflow-Ausführung werden persistent in der Datenbank

gespeichert. Eine Aktivität wird erst als beendet betrachtet, wenn sie erfolgreich ausgeführt wurde. Ist dies z.B. aufgrund eventueller Softwarefehler nicht der Fall, so erscheint sie erneut in der Worklist der Benutzer. Weiterführende Transaktionskonzepte wie sie z.B. in [Leymann_97a] vorgestellt werden, werden nicht unterstützt.

Zu bemängeln ist die Tatsache, daß die gesamte Ausführung der Prozesse von der Verfügbarkeit einer einzelnen Datenbank abhängig ist, da keine Backup-Mechanismen wie z.B. in [Alonso et al_97a] vorgeschlagen, angeboten werden. Auch die Tatsache, daß einzelne Teile eines Workflows auf eigene Domänen mit eigener Datenbank ausgelagert werden können, ändert nichts an der Tatsache, daß bei Ausfall einer Datenbank die weitere Prozeßausführung (zumindest des betroffenen Teilprozesses) nicht möglich ist.

Die **Analysierbarkeit** von FlowMark wird mittels der in Kapitel 4.3.3 dargestellten Werkzeuge ermöglicht. So dient das Audit Trail zur Protokollierung aller Ereignisse die während des Betriebes auftreten, wodurch eine Analyse des Systemverhaltens ermöglicht wird. Der Status der momentanen Ausführung der Workflows kann jederzeit mittels des Prozeß-Monitors erfahren werden. Falls eine neue Version eines Workflows eingeführt wird, dessen Name mit dem Namen der alten Version identisch ist, so sorgt FlowMark durch hinzufügen einer fortlaufenden Nummer für die Eindeutigkeit der Versionen. Die alte Version kann parallel weiter verwendet oder auf einer dafür ausgezeichneten Datenbank evtl. mit aufgezeichneten Audit Trails archiviert werden. Somit ist der Aufbau einer Historie und ein Vergleich der Effizienz der jeweiligen Workflow-Modelle zu Verbesserungszwecken möglich. Ein Werkzeug, das diesen Vergleich ermöglicht, wird von FlowMark nicht angeboten. Hier muß auf andere Anbieter verwiesen werden. Der Mechanismus der Versionierung kann allerdings umgangen werden, indem man die alte Version ersetzen läßt. Exemplare der alten Version werden dadurch nicht beeinflußt und bis zum Ende abgearbeitet. Es können aber keine neuen Exemplare der alten Version mehr erzeugt werden. Als weiterer Nachteil sei noch angeführt, daß Audit Trails nicht vom Animationswerkzeug importiert werden können, wodurch eine Analyse abgearbeiteter Workflow-Exemplare ermöglicht werden könnte.

FlowMark ermöglicht sowohl die unidirektionale als auch die bidirektionale **Applikationsintegration**. Die unidirektionale Integration wird über die Program-Execution Komponente und die bidirektionale Integration über die APIs und das Service Broker Konzept ermöglicht. Der entfernte Aufruf von Programmen ist ebenfalls möglich. Die zuverlässige Ausführung von Programmen kann in sofern sichergestellt werden, daß eine Programm-Aktivität erst dann als beendet gilt, wenn die zugeordnete Applikation erfolgreich beendet wurde. Dies kann z.B. durch Abfrage des Return-Codes (Member „_RC“ im Output-Container der Programm-Aktivität) Überprüft werden. Transaktionales Verhalten der Applikationen kann von FlowMark aus nicht sichergestellt werden, sondern wird den Programmen überlassen. Werden z.B. inkonsistente Datenzustände durch ein integriertes Programm verursacht und dies erst zu einem späteren Zeitpunkt bemerkt, ist ein Rückgängigmachen nur außerhalb von FlowMark möglich. Rücksprünge innerhalb der Abarbeitung von Prozessen sind bei FlowMark nicht möglich.

Es können sowohl Standard-Applikationen als auch selbstgeschriebene Programme integriert werden. Bei Standard-Applikationen ist normalerweise nur die unidirektionale Integration möglich. Mittels des Service Broker Konzeptes kann allerdings eine bidirektionale Kommunikation mit der entsprechenden Anwendung erreicht werden, wie es z.B. bei LotusNotes der Fall ist. Somit ist sichergestellt, daß Legacy-Programme integriert werden

können, womit **Investitionssicherheit** gewährleistet werden kann. Allerdings ist es nicht möglich, mehrere Programme gleichzeitig zur Aufgabenlösung anzubieten. Hier muß man sich auf ein Programm festlegen. Somit ist der **Handlungsspielraum** der Benutzer eingeschränkt. Die Benutzer können zur Aufgabenerledigung jedoch mit einer beliebigen Anzahl von Hilfsprogrammen unterstützt werden, die von FlowMark aus gestartet werden können. Sie sind allerdings unabhängig von der jeweiligen Workflow-Ausführung.

5 Umsetzung des Lehrplanungsprozesses mit FlowMark

In diesem Abschnitt werden die aus unserer Perspektive wichtigsten Aspekte der Umsetzung des im Kapitel 2 beschriebenen Anwendungsbeispiels vorgestellt. Hierbei wird auf die Netzarchitektur, die Programmintegration, Teile des entworfenen Prozeß-Modell und die zur Umsetzung genutzten Dienste des Systems FlowMark eingegangen. Es wird aufgezeigt, auf welche Probleme wir bei der Umsetzung des Anwendungsbeispiels gestoßen sind, und wo FlowMark gute Unterstützung geleistet hat.

Wir haben uns, wie bereits in Kapitel 2 erwähnt, nur auf die tatsächliche Umsetzung des Lehrplanungsprozesses innerhalb des Arbeitsbereichs Softwaretechnik des Fachbereichs Informatik beschränkt, da uns für diesen Teilprozeß des gesamten Lehrplanungsprozesses die notwendigen Detailinformationen zur Verfügung standen. Im folgenden soll allerdings trotzdem kurz dargestellt werden, wie die Netzarchitektur einer FlowMark Installation aussehen könnte, die den gesamten universitären Lehrplanungsprozeß unterstützt, um die Leistungsfähigkeit des WfMS FlowMark besser einschätzen zu können.

5.1 Netzarchitektur

In [IBM_96a, S. 108 ff.] werden drei Workflow Szenarios vorgestellt, die beschreiben, wie die Ausführung eines Workflow über mehrere Domänen (Abteilungen) aufgeteilt werden kann, und wie das System hierfür konfiguriert werden muß (Netzarchitektur).
Architekturvorschlag Nummer eins (vgl. Abbildung 20) beschreibt ein Szenario, in dem die Workflows in einem lokalen Bereich ausgeführt werden. In diesem Bereich existieren ein Run-Time Server mit Datenbank und OODB-Server, ein Build-Time Server, ein Build-Time Client und mehrere Run-Time Clients. Dieser Vorschlag entspricht der Testinstallation, die wir zur Umsetzung des Anwendungsbeispiels verwendet haben. Vorschlag Nummer zwei weitet die Workflow-Ausführung auf mehrere lokale Bereiche aus. Hier existiert in jedem Bereich ein Run-Time Server an dem die Run-Time Clients angeschlossen sind. Die Kommunikation der Server untereinander wird mit Hilfe eines sogenannten Delivery Servers realisiert. Es gibt nur eine zentrale Datenbank in einem der Bereiche. Architekturvorschlag Nummer drei (vgl. Abbildung 21) zeichnet sich dadurch aus, daß nun in jedem Bereich eine Datenbank, ein Run-Time Server und ein Delivery Server existiert.

Um den gesamten Lehrplanungsprozeß der Universität Hamburg zu unterstützen, wäre eine Netzarchitektur zu wählen, die dem dritten der eben dargestellten Architekturvorschläge entspricht. Diese Architektur ermöglicht es, Sub-Prozesse auf eigenen Servern und Datenbanken ausführen zu lassen. Somit kann erreicht werden, daß sich die Aufbauorganisation der Universität, hinsichtlich der Aufteilung in

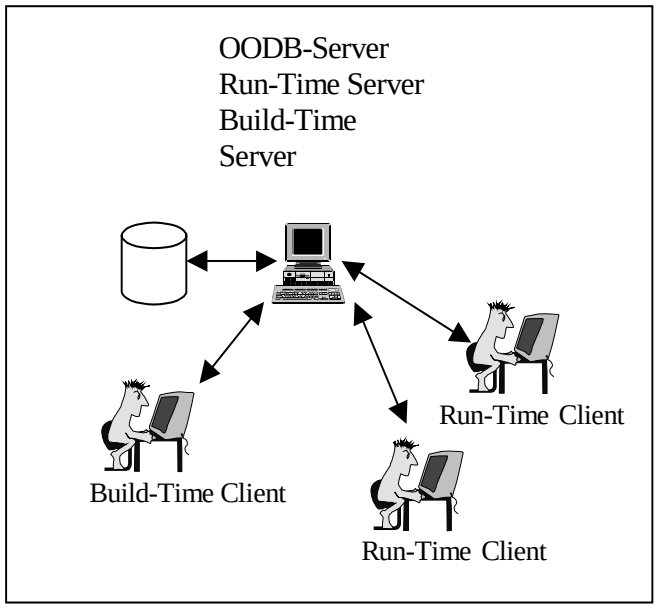


Abbildung 20: Testinstallation nach Architektur-vorschlag Nr. 1 in [IBM_96a]

verschiedene Fachbereiche, auch in der Netzarchitektur wiederfindet, wenn jedem Fachbereich eine Domäne zugeordnet wird. Laut [IBM_96a, S. 111] sollte man diese Architektur u. a. wählen, wenn bestimmte Teile eines Workflows bestimmten Abteilungen zugeordnet werden, die sich potentiell auch an verschiedenen Orten befinden können und eine Vielzahl von Personen an der Ausführung beteiligt sind. Dies würde auf die Anwendungssituation zutreffen, wenn der gesamte Lehrplanungsprozeß zu unterstützen wäre. Die Lehrplanungsprozesse laufen in jedem Fachbereich weitestgehend unabhängig von der Lehrplanung in anderen Fachbereichen der Universität ab. Die Fachbereiche befinden sich nicht alle am gleichen Ort (z.B. FB-Informatik in Stellingen und FB-BWL am Campus) und alle Mitarbeiter der Universität, die Lehrveranstaltungen anbieten, sind an dem Lehrplanungsprozeß beteiligt.

Wie auf Abbildung 21 zu sehen ist, könnte jedem Fachbereich und der Verwaltung der Universität ein eigenes Teilnetz zugeordnet werden. Die Administration des Systems und die Pflege des Workflow-Modells des Lehrplanungsprozesses würde im Bereich der Verwaltung stattfinden. Hier könnten neue Benutzer gegenüber dem System autorisiert und die Aufbauorganisation auf dem aktuellsten Stand gehalten werden. Aus diesem Grund findet sich hier auch die Build-Time (Server und Client). Der gesamte Lehrplanungsprozeß sollte von einer autorisierten Person der Uni-Verwaltung angestoßen werden, damit sichergestellt ist, daß alle Beteiligten die Aufforderung zur Lehrplanung rechtzeitig erhalten. Der Campus Raum/Zeitplaner wird ebenfalls der Verwaltung zugeordnet, welches die Verfügbarkeit von Run-Time Clients und eines Run-Time Servers in diesem Teilnetz erfordert. Mittels Delivery Server könnte die Kommunikation der sich in den einzelnen Fachbereichen befindlichen Run-Time Servern, die den Zugriff der Run-Time Clients auf die Datenbanken ermöglichen, realisiert werden.

Um die Ausführung von Sub-Prozessen auf dafür vorgesehenen FlowMark Servern zu ermöglichen, bietet FlowMark an, zur Build-Time für jeden Prozeß den entsprechenden Server zu spezifizieren (statische Zuweisung durch Angabe des Servernamens). Der jeweilige Server muß FlowMark bekannt, also im Vorwege registriert worden sein. Neben dieser statischen Zuweisung besteht auch die Möglichkeit der dynamischen Zuweisung des Servers, auf dem der Sub-Prozeß ausgeführt werden soll. Hierzu muß bei dessen Definition die Option eingestellt werden, daß der Name des zu verwendenden Servers zur Run-Time im Input-Container des Prozesses zu finden ist (wobei die Angabe des Member-Namens erforderlich ist; vgl. Kapitel 4.1).

Im weiteren Verlauf dieses Kapitels werden wir uns auf die Beschreibung der Umsetzung des Lehrplanungsprozesses innerhalb des FB-Informatik beschränken.

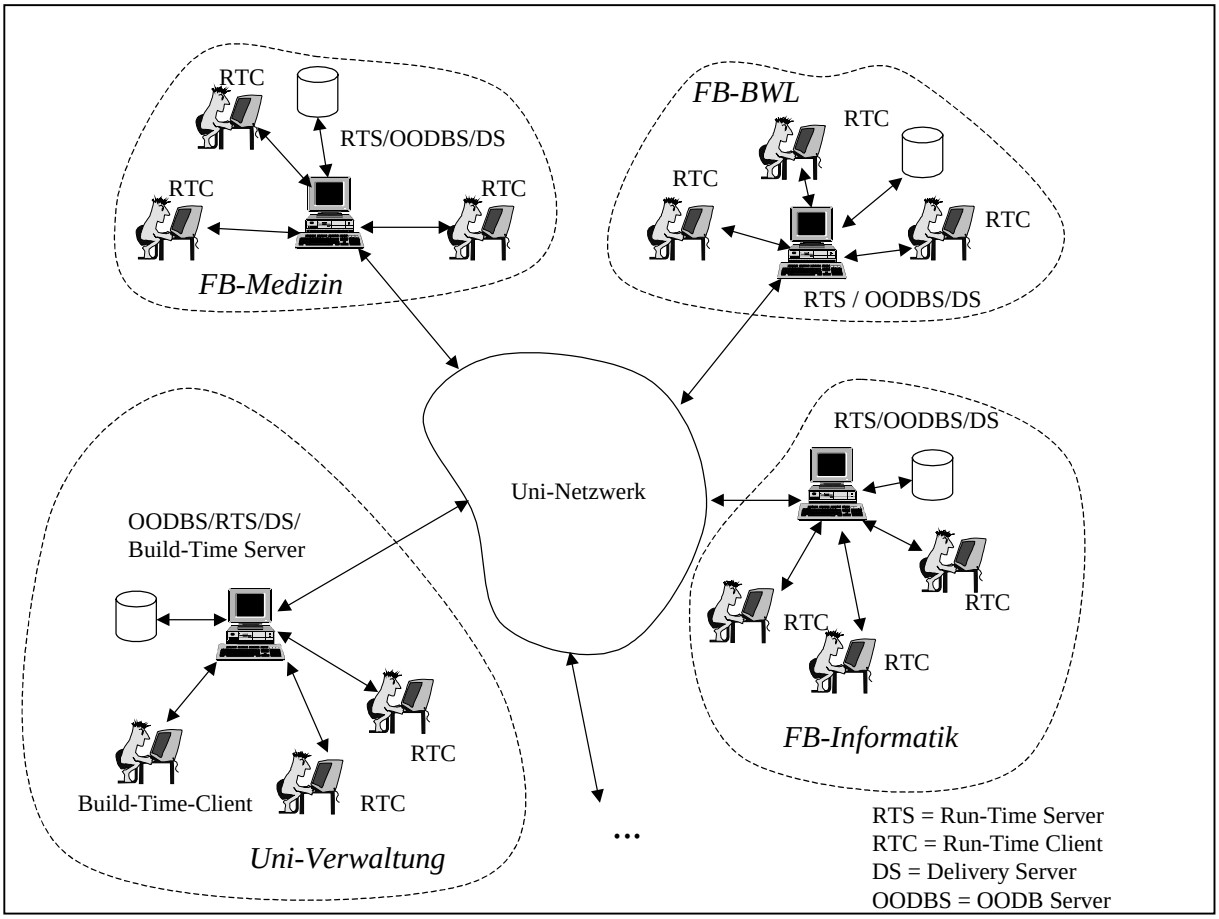


Abbildung 21: Mögliche Netzarchitektur der Implementierung (Architekturvorschlag Nr. 3 in [IBM_96a])

5.2 Sternförmige Prozesse

Die Unterstützung des Lehrplanungsprozesses erfordert die parallele Ausführung der Prozesse „Lehrplanung übernächstes Semester“ bzw. „Lehrplanung nächstes Semester“ in den einzelnen Arbeitsbereichen des Fachbereichs Informatik. Wenn man davon ausgeht, daß die Lehrplanung in den einzelnen ABs und auf die gleiche Art und Weise abläuft, so kann man zur Unterstützung auf das Konzept des Bündels (engl. Bundle) zurückgreifen (vgl. Kapitel 4.1.2). Die Nutzung dieses Konzeptes ermöglicht es, in den einzelnen Arbeitsbereichen die gleichen Prozesse (Exemplare, die von einer Schablone erzeugt wurden) parallel ausführen zu lassen.

Bei den parallel ausgeführten Prozessen („AB-Lehrplanung über- bzw. nächstes Semester“) handelt es sich um Sub-Prozesse, die Bestandteil der übergeordneten Prozesse „FB-Lehrplanung über- bzw. nächstes Semester“ sind. Die Sub-Prozesse werden hier als sternförmig bezeichnet, da zu einem bestimmten Zeitpunkt im Ablauf des übergeordneten Prozesses mehrere gleichartige Exemplare der untergeordneten Prozesse erzeugt werden, die parallel ausgeführt und zu einem späteren Zeitpunkt (nach ihrer Abarbeitung) synchronisiert werden. Eine Aktivität, die innerhalb des übergeordneten Prozesses dem Bündel nachgeordnet ist, kann erst dann ausgeführt werden, wenn alle Exemplare des Sub-Prozesses erfolgreich abgearbeitet wurden. Eine bildhafte Vorstellung dieses Ablaufs führt zum Begriff der

sternförmigen Prozesse, da sich die parallel ausgeführten Prozesse durch ihre Gleichartigkeit auszeichnen.

Um die Vorgänge, die bei der Erzeugung von Exemplaren einer Schablone mittels des Bündel-Konzeptes stattfinden, vertiefender darzustellen, soll an dieser Stelle kurz gezeigt werden, wie zur Laufzeit den einzelnen ABs Exemplare des Prozesses „AB-Lehrplanung übernächstes Semester“ zugewiesen werden.

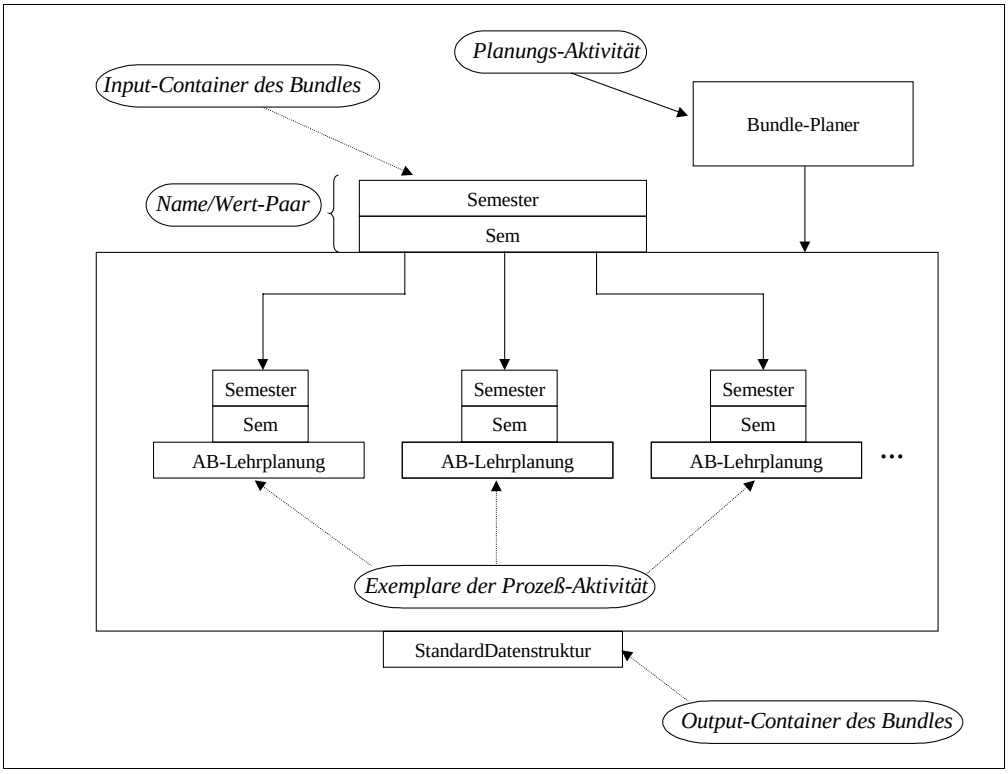


Abbildung 22: Erzeugung von Sub-Prozeß-Exemplaren zur Laufzeit

Das Konzept des Bundles kann in diesem Fall mit der Möglichkeit der dynamischen Zuweisung der Personen, die für die Ausführung der Sub-Prozesse zuständig sind, kombiniert werden. Das Bundle wird mit der Option der „personengetriebene Erzeugung“ gestartet (vgl. Kapitel 4.1.2). In diesem Fall wird kein ausgezeichnetes Array benötigt, da die Anzahl der zu erzeugenden Exemplare zur Laufzeit aus der Anzahl der Träger der Rolle AB-Lehrplanungsbeauftragter[10] des FB-Informatik ermittelt wird. Jedem Rolleninhaber wird dann ein Exemplar des Sub-Prozesses zugewiesen und muß von diesem gestartet werden. Die Bestimmung der Anzahl der Exemplare (als „AB-Lehrplanung“ bezeichnet) übernimmt ein spezielles von FlowMark mitgeliefertes Programm, welches der Planungs-Aktivität „Bundle-Planer“ zugeordnet wird (vgl. Abbildung 22). Als weiteren Parameter erwartet der Sub-Prozeß die Angabe des Semesters, für welches die Lehrplanung stattfindet. Dieser Parameter wird über den Eingabedaten-Container des Bundles an die Sub-Prozeß-Exemplare weitergeleitet.

[10] Wir verwenden zur Bestimmung der Anzahl der zu erzeugenden Exemplare des Sub-Prozesses nicht die ABs sondern die AB-Lehrplanungsbeauftragten der ABs, da FlowMark die Anwendung dieses Verfahrens nur auf Rollen bzw. Personen erlaubt. Da in jedem AB genau ein Träger dieser Rolle existiert, stellt die Verwendung kein Problem dar. Wenn die Erzeugung der Anzahl der Exemplare an den ABs festgemacht werden soll, so müßte die Variante der datengetriebenen Erzeugung verwendet werden, was komplizierter zu handhaben wäre und in diesem Fall keine Verbesserung bringen würde.

Hierzu muß ein Daten-Konnektor zwischen Source des Bundles und Eingabedaten-Container der Prozeß-Aktivität „AB-Lehrplanung“ innerhalb des Prozeß-Modells existieren. Das korrespondierende FlowMark Prozeß-Modell wird in Abbildung 23 dargestellt. Hierbei repräsentiert der Screenshot im oberen Bereich der Abbildung 23 den übergeordneten Prozeß (Ebene1, vgl. Kapitel 4.1.2), der die Planungsaktivität „Bundle-Planer“ und das Bundle „AB-Lehrplanung“ enthält. Das Prozeß-Modell des Bundles ist im unteren Screenshot dargestellt (Ebene 2), wobei die Prozeß-Aktivität „AB-Lehrplanung“ die Muster-Aktivität (in diesem Fall ein Sub-Prozeß) repräsentiert, von der die Exemplare erzeugt werden.

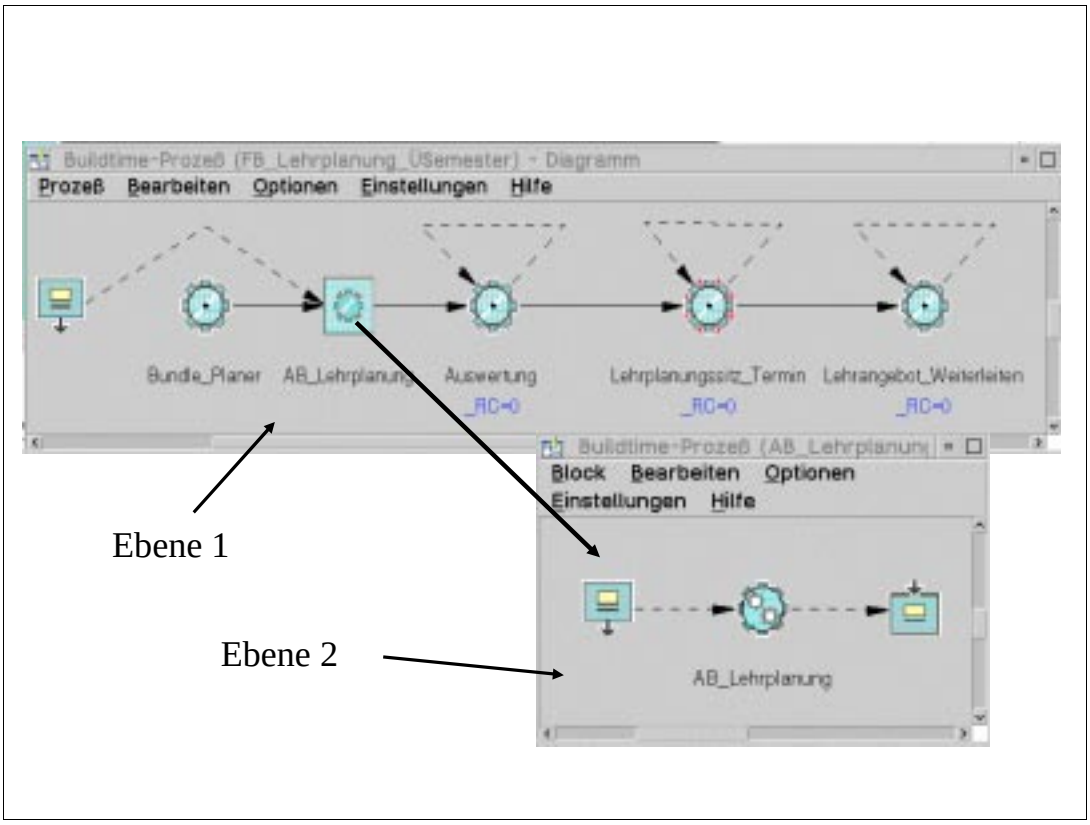


Abbildung 23: Ausschnitt aus dem Prozeß-Modell zur „FB-Lehrplanung übernächstes Semester“

Neben der Möglichkeit, die Anzahl der zu erzeugenden Exemplare über die Anzahl der Rollenträger zu bestimmen, gibt es noch die Möglichkeit der datengetriebenen Erzeugung (vgl. Kapitel 4.1.2). Von dieser Möglichkeit könnte z.B. Gebrauch gemacht werden, wenn Sub-Prozesse auf verschiedenen Servern ausgeführt werden sollen (falls z.B. der gesamte Lehrplanungsprozeß der Universität unterstützt werden sollte). In diesem Fall würde man als Elemente des erforderlichen Arrays (BundleInstantiatingArray) die Servernamen setzen. Dieses Array würde der Planungs-Aktivität als Parameter übergeben und könnte von dem der Planungs-Aktivität zugeordnetem Programm dann zur Bestimmung der Server verwendet werden, die jeweils Exemplare der Sub-Prozesse auszuführen haben.

5.3 Programmintegration

Wie in Kapitel 3 dargestellt, findet eine Interaktion der beteiligten Personen, neben der direkten Kommunikation in den Sitzungen (von Angesicht zu Angesicht), auch indirekt über

Dokumente statt. Um die letztgenannte Form der Interaktion zu unterstützen haben wir uns entschlossen, den Benutzern zur Erstellung und Auswertung von Dokumenten ein Werkzeug an die Hand zu geben, mit denen sie diese Aufgaben effizient erledigen können. Wir haben uns für die Integration von Microsoft Word entschieden, da uns diese Textverarbeitung zum einen zur Verfügung steht (Investitionssicherung) und zum anderen auch standardmäßig zur Erstellung von Dokumenten verwendet wird. Somit fällt seitens der Benutzer kaum Einarbeitungszeit an, was zu erhöhter Benutzerakzeptanz führt. (vgl. Kapitel 4.2.1)

Zur Integration von Microsoft Word nutzen wir sowohl die Möglichkeit von FlowMark aus Fremdprogramme mit Übergabeparametern zu starten, als auch die Möglichkeit auf Datencontainer von Aktivitäten und Prozessen mittels Programmierschnittstelle zuzugreifen (vgl. Kapitel 4.3.4). Da Microsoft Word 7.0 Visual Basic als Programmiersprache „von Haus aus mitbringt“ und FlowMark eine Visual Basic-API unter Windows NT zur Verfügung stellt, haben wir uns für die Verwendung dieser zu der Gruppe der language APIs (vgl. Kapitel 4.3.4) gehörenden Schnittstelle entschieden.

FlowMark bietet keine Dokumentenmanagement-Funktionalität an. Aus diesem Grund mußten wir uns ein eigenes Konzept überlegen, wie die Dokumente verwaltet werden sollen. Dies wird nachfolgend dargestellt.

Die Dokumente, die innerhalb des Prozeßablaufs erstellt werden, werden nach Freigabe durch den Benutzer auf einem allgemein zugänglichen Rechner in einem bestimmten Verzeichnis gespeichert (im folgenden als Dokumentenserver bezeichnet). Hier befinden sich auch die Dokumentvorlagen, die zur Erstellung der Dokumente genutzt werden. Startet FlowMark Microsoft Word, so wird Microsoft Word durch Parameterübergabe zur Ausführung eines Makros aufgefordert, daß die automatische Erstellung eines Dokumentes aus einer hierfür vorgesehenen Dokumentenvorlage übernimmt. Das Makro kann durch Nutzung der Visual Basic API die Position der Dokumentenvorlage (z.B. TCP/IP-Adresse des Dokumentenservers, Verzeichnis und Name der Dokumentenvorlage) aus dem Eingabedaten-Container der Microsoft Word zugeordneten Programmaktivität erfahren, da die entsprechenden Member des Eingabedaten-Containers im Vorwege gesetzt wurden. Nach Beendigung der Aktivität wird das erstellte Dokument zurück auf den Dokumentenserver gelegt und kann weiterverwendet werden. Damit das Dokument eindeutig identifizierbar bleibt, wird die Visual Basic-API genutzt, um von FlowMark die User ID des Bearbeiters der Programm-Aktivität zu erfahren und sie als Bestandteil des Namens des erstellten Dokumentes zu verwenden. Der vollständige Name wird dann FlowMark mittels Visual Basic-API, durch setzen eines Members des Ausgabedaten-Containers der Programm-Aktivität, bekannt gemacht. Die abgefragte User ID wird ebenfalls dazu verwendet innerhalb der Dokumente festzuhalten, wer sie erstellt hat und wer der letzte Bearbeiter gewesen ist.

Falls der Benutzer seine Arbeit an einem Dokument unterbrechen möchte, ohne daß das Dokument auf dem Dokumentenserver veröffentlicht wird und die Programm-Aktivität als beendet gilt, so hat er die Möglichkeit das Dokument lokal zwischenzuspeichern und Word zu beenden. Auch hier wird FlowMark durch setzen des Ausgabedaten-Containers mitgeteilt, wo und unter welchem Namen der Benutzer das Dokument gespeichert hat. Die Programm-Aktivität erscheint erneut im Zustand „ready“ auf der Arbeitsliste des Benutzers. Startet der Benutzer in diesem Fall die Programm-Aktivität, so wird Microsoft Word mit der lokalen Kopie statt mit der Dokumentvorlage aufgerufen.

Damit nach Freigabe eines Dokumentes zur weiteren Bearbeitung keine Änderungen mehr an der eventuell lokal existierenden Kopie gemacht werden können, wird diese automatisch

gesperrt. Es ist nur noch erlaubt Kommentare einzufügen. Somit steht z.B. dem AB-Lehrplanungsbeauftragten eine Kopie des Lehrtableaus für eventuelle Überprüfungen und Vergleiche zur Verfügung, mögliche inkonsistente Versionen werden aber verhindert.

Um dem Benutzer bei der Erstellung der Dokumente die Kommunikation mit anderen an dem Lehrplanungsprozeß beteiligten Personen zu ermöglichen, wird ein e-mail-Programm (z.B. Netscape) als Hilfsprogramm zur Verfügung gestellt, welches ebenfalls von FlowMark aus gestartet werden kann.

5.4 Diskussion der Umsetzung mit FlowMark

In diesem Abschnitt wird herausgearbeitet, in welchen Bereichen FlowMark gute Unterstützung bei der Umsetzung des Anwendungsbeispiels in eine Workflow-Anwendung geleistet hat (Möglichkeiten) und auf welche Schwierigkeiten wir gestoßen sind (Grenzen). Hierbei wird auch berücksichtigt, in wie weit die in Kapitel 3 herausgearbeiteten Kooperationsformen und Workflow-Kategorien unterstützt werden können. Abschließend wird eine Bewertung der Unterstützungsmöglichkeiten von FlowMark vorgenommen.

5.4.1 Möglichkeiten der Umsetzung

Wie in Kapitel 3 dargestellt, zeichnet sich das Anwendungsbeispiel u.a. dadurch aus, daß eine multiple Kooperationssituation vorliegt. FlowMark ist in der Lage solche Kooperationssituationen zu unterstützen, was durch die Formulierbarkeit der Aufbauorganisation und der Zuweisung ihrer Einheiten an Aktivitäten und Prozesse innerhalb eines Workflow-Modells und somit deren Berücksichtigung zur Laufzeit gegeben ist (vgl. Kapitel 4.1.1).

Die Berücksichtigung konjunktiver Kooperationsformen ist in FlowMark durch die Definition sequentieller und paralleler Abläufe mit Synchronisation (präskriptive Ablaufkontrolle, vgl. Kapitel 3.3.3) möglich. Ein Fortschreiten des jeweiligen Prozesses ist nur möglich, wenn die einzelnen Aktivitäten erfolgreich bearbeitet wurden, wobei der gesamte Prozeß erst dann als beendet gilt, wenn alle Aktivitäten und Sub-Prozesse beendet sind. Somit wird sichergestellt, daß alle Prozeßbeteiligten ihre Aufgabe erledigen, was zur Erreichung des Gesamtzieles notwendig ist.

FlowMark ermöglicht die Formulierung gleichartiger paralleler Abläufe auf bequeme Art und Weise mittels des Bündelkonzeptes. Dies hat sich bei der Umsetzung des Lehrplanungsprozesses als sehr nützlich erwiesen, da eines seiner weiteren wesentlichen Charakteristika gerade die parallele Ausführung gleichartiger Sub-Prozesse (sternförmige Prozesse; 1:n-Kommunikation) ist, wobei keine Interdependenzen zwischen den Sub-Prozessen bestehen und sie lediglich eine Synchronisation nach ihrer Abarbeitung erfordern. So kann z.B. der FB-Lehrplaner seiner Rolle als Koordinator der FB-Lehrplanung unter Nutzung dieses Konzeptes und des Monitoring-Tools (vgl. Kapitel 4.3.3) gerecht werden. Das Monitoring Tool ermöglicht die Feststellung des Bearbeitungszustandes des Prozesses. Falls es zu Verzögerungen bei der Erstellung der AB-Lehrpläne kommt, kann der FB-Lehrplaner herausfinden, wer die Verzögerungen verursacht hat und gezielt mit dem Verantwortlichen in Kontakt treten, um gemeinsam eine Lösung des Problems zu finden.

5.4.2 Grenzen der Umsetzung

Das größte Manko von FlowMark, welches uns bei der Umsetzung des Lehrplanungsprozesses aufgefallen ist, ist die fehlende Dokumentenmanagement-Funktionalität. Eine Unterstützung der asynchronen, rigiden Kommunikation über Dokumente, wie vom Anwendungsbeispiel gefordert, ist nur mit Zusatzaufwand möglich. Es gibt zwar die Möglichkeit, FlowMark über die angebotenen Programmierschnittstellen (APIs) mitzuteilen, wo sich die zu bearbeitenden Dokumente befinden und wie sie heißen, allerdings muß der Transport und die Verwaltung der Dokumente außerhalb des Systems stattfinden. Außerdem ist zu beachten, daß sich der intendierte Gebrauch der Datencontainer nicht auf den Transport von Nutzdaten bezieht, sondern auf den Transport von Kontrolldaten welche zur Auswertung von Transitionsbedingungen benötigt werden und den Kontrollfluß festlegen[11].

Lose Kooperation und somit die Koordination durch Selbstabstimmung der Akteure, wie sie z.B. in den Sitzungen stattfindet, ist mit FlowMark nicht unterstützbar. Das Vorkommen solcher Kooperationsformen kann lediglich durch Verwendung der sogenannten „Manual Checklist“ berücksichtigt werden. Bei der „Manual Checklist“ handelt es sich laut [IBM_96a, S. 173] um ein Programm, welches die Berücksichtigung von Vorgängen, die außerhalb des Systems stattfinden (wie z.B. Telefonate, Sitzungen, usw.), im Workflow-Modell ermöglicht. Das Programm stellt eine Liste von auszuführenden Aktivitäten bereit, die vom Benutzer abgehakt werden müssen („to do Liste“). Erst nachdem alle Eintragungen abgehakt wurden, wird die Prozeß-Ausführung fortgesetzt. Um eine Unterstützung loser Kooperation sicherzustellen, muß auf andere Systeme, wie z.B. Joint Editing- oder Decision Support-Systeme (vgl. Kapitel 3.2.2) zurückgegriffen werden, wobei in diesem Falle eine Applikationsintegration (vgl. Kapitel 4.2.1.3) durchzuführen wäre. Da FlowMark nur die Berücksichtigung loser Kooperation ermöglicht, sind somit auch Workflows, die der Kategorie der Task Forces aus dem Workflow-Kontinuum angehören, nicht unterstützbar. Da Task Forces wiederum als Bestandteile semi-strukturierter Workflows gesehen werden, folgt daraus, daß auch diese Art von Workflows nicht vollständig unterstützt werden kann.

Rücksprünge im Prozeßablauf sind nicht möglich. Falls bei der Dokumentenerstellung Fehler gemacht und diese erst zu einem späteren Zeitpunkt bemerkt werden (Dokument ist schon freigegeben), so ist es unmöglich die bis dato durchgeführten Aktionen rückgängig zu machen. Der Prozeß muß entweder abgebrochen und neu gestartet werden oder es muß eine Lösung außerhalb des Systems gefunden werden. Es wird zwar die Möglichkeit der Berücksichtigung von Fehlerfällen im Workflow-Modell angeboten (siehe Default-Konnektor, Kapitel 4.1.2), allerdings müßten alle möglichen Fehlerfälle im voraus durchdacht und eine Lösungsstrategie bekannt sein, was so gut wie unmöglich ist. Von FlowMark wird ebenso keine Möglichkeit angeboten, zur Laufzeit den Workflow-Typ zu wechseln (vgl. Kapitel 4.1.5). Somit liegt eine Unterstützung von ad-hoc Ausnahmen erfordernden Workflows (zählen zu den semi-strukturierten Workflows) außerhalb der Fähigkeiten von FlowMark.

Die von uns herausgearbieteten Möglichkeiten und Grenzen der Unterstützung kooperativer Arbeitssituationen durch FlowMark, welche sich aus den Anforderungen des

[11] Von [Alonso et al_97a] wird eine Lösung dieses Problems vorgestellt, die sich auf die Integration der beiden Systeme Exotica/FMQM (FlowMark on Message Queue Manager) und Lotus Notes stützt. Bei dem System Exotica/FMQM handelt es sich um eine Weiterentwicklung von FlowMark (Froschungsprojekt), welche sich vorwiegend durch verbesserte Fehlerbehandlung und erweiterte Skalierbarkeit auszeichnet. Eine Beschreibung dieses Systems findet sich u.a. in [Alonso et al_95] und [Alonso et al_97a].

Anwendungsbeispiels ergeben haben, werden in der nachfolgenden Tabelle zusammenfassend dargestellt.

Anforderung	***Direkt unterstützbar***
Multiple Kooperation	Ja
Konjunktive Kooperation	Ja
Rigide Kooperation über Dokumenten-austausch	Nein (erfordert Applikationsintegration)
Lose Kooperation und Selbstabstimmung (Sitzungen)	Nein (nur Berücksichtigung bzw. Applikationsintegration erforderlich)
Sternförmige Prozesse	Ja (Weiterarbeitung nur möglich, wenn alle Subprozesse abgearbeitet sind)

Tabelle 4: Möglichkeiten und Grenzen bei der Umsetzung des Anwendungsbeispiels mit FlowMark

5.4.3 Bewertung von FlowMark

Laut [Jablonski_97a, S.6] werden die Fähigkeiten eines WfMS bzw. einer Workflow-Management-Anwendung hinsichtlich der Unterstützung von Arbeitsvorgängen nachhaltig dadurch bestimmt, welche Möglichkeiten zur Darstellung der Geschäftsprozesse angeboten werden (vgl. Kap. 3.3.1). WfMS sollen sowohl fest strukturierte als auch unstrukturierte Arbeitsvorgänge unterstützen. Alle Kategorien des Workflow Kontinuums (vgl. Kap. 3.3.4) sollten modelliert werden können.

Standard Workflows lassen sich unserer Meinung nach in FlowMark hinsichtlich des Kontrollflusses gut modellieren. Insbesondere die Möglichkeit, sternförmige Prozesse mit Hilfe des Bündel-Konzepts zu modellieren erweist sich als hilfreich. Hinsichtlich der Unterstützung des Dokumentenaustausches zwischen Aktivitäten sind jedoch klare Nachteile zu erkennen. Der Fluß von Dokumenten wird in FlowMark nicht unterstützt. Man ist gezwungen, den Umweg über einen eigens eingerichteten Informationsraum zu gehen, wobei die Aktivitäten dann jeweils nur den Namen des Dokuments und die Verzeichnisstruktur, unter der dieses gespeichert ist, austauschen. Dieser Umstand stellt unserer Meinung nach einen erheblichen Aufwand dar, sobald eine größere Anzahl von Dokumenten ausgetauscht werden soll. Der in FlowMark realisierte Datenfluß eignet sich eher für Anwendungen, bei denen geringe Mengen von Informationen ausgetauscht werden wie z.B. in typischen Sachbearbeitungsvorgängen.

FlowMark stößt in seinen Modellierungsmöglichkeiten auf Grenzen, sobald lose Kooperationen, wie sie in **Task Forces** bzw. bei der **offenen Teambearbeitung** vorkommen**,** modelliert werden sollen. In der Aufbauorganisation werden Arbeitsgruppen in FlowMark nicht unterstützt. Es gibt nur die Möglichkeit, einer Rolle einen Koordinator zuzuweisen, der dann anfallende Aufgaben an die Inhaber der Rolle verteilt. Typische Kooperationsformen der Gruppenarbeit, wie z.B. synchrone Kommunikationsprozesse oder Verhandlungen werden nicht unterstützt. Lösungsmöglichkeiten bestehen durch die Integration von Workgroup Computing-Applikationen wie z.B. Group Decision Support Systems (GDSS). In diesem Fall liegt jedoch die Kontrolle nicht mehr beim WfMS, da solche Applikationen lange andauern

können. In Anwendungen, die lose Kooperationen beinhalten, ist also FlowMark aus unserer Sicht nicht geeignet.

Zu den in Kapitel 4.3.3 vorgestellten **Hilfswerkzeugen** von FlowMark ist anzumerken, daß diese ebenfalls einige Mängel aufweisen. So ermöglicht beispielsweise der Notification Service nur die Spezifizierung eines Zeitraumes, innerhalb dessen ein Sub-Prozeß bzw. eine Aktivität abgearbeitet werden muß. Es ist nicht möglich ein Datum festzulegen, bis zu dem die jeweiligen Aufgaben erledigt sein müssen. Somit muß auf die Nutzung von API-Funktionen zurückgegriffen werden, um eine Deadline zu simulieren, was die Handhabung umständlicher macht.

Die **Monitoring-Funktionalität** im laufenden Prozeß beschränkt sich auf die Möglichkeit der Feststellung des momentanen Zustandes und der Historie des Prozesses. Für an den Prozeß Partizipierende ist es nur möglich, die Personen festzustellen, die zu einem früheren Zeitpunkt eine Aufgabe bearbeitet haben. Es ist nicht feststellbar, wer (Personen oder Rolleninhaber) die nächste Aufgabe erledigen wird. Somit kann sich der die momentane Aufgabe erfüllende nicht auf die Arbeitsgewohnheiten des Nachfolgers einstellen bzw. eventuell mit ihm in Kontakt treten, um entsprechende Informationen auszutauschen.

Ein weiteres Manko stellt die Tatsache dar, daß die mittels **Audit Trail** aufgezeichneten Informationen nicht vom **Animationstool** zu Simulationszwecken importiert werden können. Um die Daten auszuwerten muß auf Werkzeuge anderer Hersteller zurückgegriffen werden, was bei einem System dieser Preiskategorie nicht einzusehen ist.

6 Zusammenfassung und Ausblick

In dieser Arbeit wird anhand eines Anwendungsbeispiels (der universitäre Lehrplanungsprozeß am Fachbereich Informatik) die Unterstützung verschiedener Kooperationsformen durch das Workflow Management-System (WfMS) FlowMark von IBM untersucht.

Im Anwendungsbeispiel finden sich drei charakteristische Kooperationsformen (vgl. Kap.3.1.3):

a) Ein Fachbereichs-Lehrplanungsbeauftragter kontrolliert und synchronisiert die Lehrplanung an den Arbeitsbereichen. Dieser Prozeß ist bildlich vorgestellt *sternförmig.*
b) Die Arbeitsbereichs-Lehrplanung verläuft als *verständigungsorientierter Prozeß*. Die Kooperation ist lose.
c) Es finden *fest strukturierte, standardisierte Prozesse* statt, die wiederholt jeweils dieselbe vorbestimmte Reihenfolge von Arbeitsschritten durchlaufen. Die Kooperation ist rigide.

Die Fähigkeiten von WfMS werden nach den verschiedenen Möglichkeiten zur Darstellung von Geschäftsprozessen in einem Workflow-Modell beurteilt (vgl. Kap. 3.3.3). Das Workflow-Modell sollte die in einem Workflow vorkommenden Aspekte (Aktivitäten, Aktoren, Abhängigkeiten, Kausalitäten) enthalten.

Insbesondere sollten sowohl stark strukturierte Standard Workflows als auch wenig strukturierte Workflows (wie z.B. bei Task Forces) sowie deren Kombinierung (offene Teamarbeit, ad-hoc Ausnahmen) dargestellt werden können (vgl. Kap. 3.3.4).

Betrachtet man die drei oben beschriebenen Kooperationsarten des Anwendungsbeispiels, so erkennt man Standard Workflows (c), Task Forces (b) sowie semi-strukturierte Workflows (a). Für eine Realisierung des Anwendungsbeispiels sollte ein WfMS folgende Workflow-Kategorien unterstützen (vgl. Kap. 3.3.5):

- Standard-Workflows,
- Task Force bzw. offene Teambearbeitung sowie
- Ad hoc-Ausnahmen

Das WfMS FlowMark von IBM ist darauf ausgerichtet, Standard Workflows umzusetzen. FlowMark erfüllt größtenteils die geforderte Möglichkeit der getrennten Modellierung der in einem Workflow vorkommenden Aspekte (Aktivitäten, Aktoren, Abhängigkeiten, Kausalitäten). Lediglich die Formulierung deskriptiver Ablaufkontrolle und von Kausalitäten wird nicht angeboten (vgl. Kap. 4.1.5).

Viele aktuelle WfMS beschränken sich auf die Modellierung sequentieller Abläufe. In der Realität weisen jedoch Geschäftsprozesse komplexere Strukturen auf, wie Querverbindungen zwischen einzelnen Prozessen, Vor- und Rückläufe etc. Es müssen also in Zukunft Ansätze für die Modellierung und Organisation von Workflows gefunden werden, die diesen Anforderungen entsprechen. Ein Beispiel ist der ActionWorkflow-Ansatz (vgl. [Action Technologies_93])

WfMS können nicht alle Formen kooperativer Arbeit unterstützen. Um unterschiedliche voneinander abhängige Kooperationen zu unterstützen müssen WfMS mit anderen Groupware-Applikationen zusammen eingesetzt werden. Aktuelle Forschungen beziehen sich deshalb auf die Integrationsmöglichkeiten von Groupware-Applikationen in WfMS. Das Problem dabei ist vor allem, einen Synchronisationsmechanismus zwischen dem WfMS und

anderen Groupware-Systemen zu finden. Erste Lösungsmöglichkeiten dazu und einen Überblick anderer Forschungsprojekte zu dem Thema bietet [Schwab et al_95].

Es gibt auch Ansätze, in vorhandene Groupware-Systeme wie z.B. Lotus Notes Koordinationsfunktionen zu integrieren, so daß auch Abläufe gesteuert werden können (vgl. z.B. [Nastansky_96]).

Das Nachfolgerprodukt von FlowMark heißt „MQSeries Workflow“. Der Name resultiert aus der Integration von FlowMark in die MQSeries-Produktpalette von IBM. Dies führt dazu, daß FlowMark nun auch von den Vorteilen, die MQSeries[12] bietet, profitieren kann. Das neue Produkt zeichnet sich insbesondere durch erweiterte Applikationsintegrationsmöglichkeiten (durch den „MQSeries Integrator“) und Skalierbarkeit aus. Laut [Chambers et al_99] ist nun sogar eine dynamische Lastverteilung zur Laufzeit möglich (vgl. Kapitel 4.3.5). Aus [IBM_98b] geht hervor, daß sich an den Modellierungsmöglichkeiten allerdings nicht sehr viel geändert hat. Somit scheinen die in dieser Arbeit in Bezug auf unterstützbare Arten kooperativer Arbeitssituationen gewonnen Ergebnisse auch auf die neue Version anwendbar.

[12] Zu Konzepten von MQSeries siehe z.B. [Felten_98]

Literatur

[Action Technologies_93] Action Technologies Inc.: *Action Workflow White Paper-Components and Architecture,* 1993

[Alonso et al_96] Alonso, G., Agrawal, D., El Abbadi, A., Mohan, C.: *Functionality and Limitations of Current Workflow Management Systems.* http://www.inf.ethz.ch/personal/alonso/reference_list.html

[Alonso et al_97a] Alonso, G., Reinwald, B., Mohan, C.: *Distributed Data Management In Workflow Environments.* http://www.inf.ethz.ch/personal/alonso/reference_list.html, 1997

[Alonso et al_97b] Alonso, G., Mohan, C.: *Workflow Management Systems: The Next Generation Of Distributed Processing Tools.* In: Sushil Jajodia und Larry Kerschber (editors), Advandced Transaction Models and Architectures. Kluwer Academic Publishers, 1997. Auch unter: http://www.inf.ethz.ch/personal/alonso/reference_list.html, 1997.

[Baer_73] Baer, J. L.: *A Survey of Some Theoretical Aspects of Multiprocessing.* In: Computing Surveys 5, No. 1, S. 31 – 80, 1973.

[Bair_89] J.H. Bair: *Supporting Cooperative Work with Computers: Addressing Meeting Mania*, in: Proc. 34th IEEE Computer Society International Conference (COMPCON), IEEE Computer Society, S. 208-217, 1989

[Balzert et al_88] Balzert, H., Ackermann, D.: *Einführung in die Softwareergonomie.* De Gruyter, Berlin, 1988.

[Bannon et al_89] L. Bannon, K. Schmidt: *Four characters in search of a context,* in: Proc. First European Conference on Computer Supported Cooperative Work, ECSCW'89

[Böhm et al_95] Böhm, M., Schulze, W.: *Grundlagen von Workflow Management Systemen.* In: Wissenschaftliche Beiträge zur Informatik, Heft 2, 1995.

[Böhm et al_96] A. Böhm, W. Oberndorfer, R. Schmitz, S. Uellner: *Werkzeuge für Forscher und Manager in europäischen Telekommuikationsprojekten*, in: H. Krcmar, H. Lewe, G. Schwabe Herausforderung Telekooperation, Berlin Heidelberg, 1996

[Brockhaus_93a] Brockhaus-Enzyklopädie, APU-BEC, 2. Band, 19. Auflage, F. A. Brockhaus Verlag, 1993.

[Brockhaus_93b] Brockhaus-Enzyklopädie, SR-TEO, 21. Band, 19. Auflage, F. A. Brockhaus Verlag, 1993.

[Chambers et al_99] B. Chambers, D. Chan, K. West: *Functional Assessment of IBM MQSeries Workflow 3.1*, Doculabs White Paper, 1999

[Curtis et al_92] Curtis, B., Kellner, M.I., Over, J.: *Process Modeling.* In: Communications of the ACM, 35:9, S. 75-90, September, 1992.

[Dittrich_91] J. Dittrich: *Koordinationsmodelle für Computerunterstütze Gruppen*, in: Friedrich J. Rödiger K.-H. (Hrsg.): Computergestützte Gruppenarbeit (CSCW), Teubner, Stuttgart, 1991

[Felten_98] Felten, A.: *Entwurf und Implementierung eines verteilten objektorientierten Fehlerbehandlungssystems in einem heterogenen Systemumfeld.* Diplomarbeit am Arbeitsbereich Softwaretechnik der Universität Hamburg, 1998.

[Grudin_94] J. Grudin: *Computer-Supported Cooperative Work: History and Focus*, IEEE Computer, Vol. 27,Nr. 5, S. 19-26, 1994

[Haugeneder et al_94] H. Haugeneder, D. Steiner: *Ein Mehragentenansatz zur Unterstützung kooperativer Arbeit*, in: Hasenkamp U., Kirn S., Syring M. (Hrsg.): CSCW-Computer Supported Cooperative Work, Addison-Wesley, Bonn, S. 203-229

[Hilpert et al_94] Hilpert, W., Nastansky, L.: *The GroupFlow System: A Scalable Approach to Workflow Management between Cooperation an Automation.* In: Innovation bei Rechen- und Kommunikationssystemen – Eine Herausforderung an die Informatik, Hrsg. Bernd Wolfinger, Proceedings, 24. GI Jahrestagung im Rahmen des 13th World Computer Congress, IFIP Congres 1994, Springer-Verlag, 1995, S. 473 – 479.

[IBM_96a] International Business Machines Corporation: *IBM FlowMark: Modeling Workflow.* Version 2 Release 3, 1996.

[IBM_96b] International Business Machines Corporation: *IBM FlowMark: Managing Your Workflow.* Version 2 Release 3, 1996.

[IBM_96c] International Business Machines Corporation: *IBM FlowMark: Programming Guide.* Version 2 Release 3, 1996.

[IBM_96d] International Business Machines Corporation: *IBM FlowMark: Application Integration Guide.* Version 2 Release 3, 1996.

[IBM_98] International Business Machines Corporation: *IBM FlowMark at a glance.*

[IBM_98b] International Business Machines Corporation: *IBM MQSeries Workflow: Getting Started with Buildtime.* Version 3.1.2, 1998

[IBM_98c] International Business Machines Corporation: *IBM MQSeries Workflow: Concepts and Architecture.* Version 3.1, 1998

[Jablonski_95] Jablonski, S.: *Workflow-Management-Systeme: Motivation, Modellierung, Architektur.* In: Informatik-Spektrum 18 (1995), S.13 - 34, Springer-Verlag 1995.

[Jablonski et al_97a] Jablonski, S., Böhm, M., Schulze, W. (Hrsg.): *WorkflowManagement: Entwicklung von Anwendungen und Systemen; Facetten einer neuen Technologie.* Dpunkt-Verlag, 1997

[Jablonski_97b] Jablonski, S.: *Architektur von Workflow-Management-Systemen.* In: Informatik: Forschung und Entwicklung, Band 12, S. 72-81, 1997.

[Keller et al_97] Keller, G., Teufel, T.: *SAP R/3 prozeßorientiert anwenden.* Addison-Wesley, 1997

[Krabbel et al_96] Krabbel, A., Wetzel, I., Ratuski, S.: *Objektorientierte Analysetechniken für übergreifende Aufgaben.* In: Beiträge der GI-Fachtagung Softwaretechnik 1996, Koblenz. S. 65 – 72, 12.-13. September 1996.

[Leymann et al_94] Leymann, F., Altenhuber, W.: *Managing business processes as an information resource.* In: IBM Systems Journal, Vol. 33, No.2, 1994.

[Leymann_97a] Leymann, F.: *Transaktionsunterstützung für Workflows.* In: Informatik Forschung und Entwicklung, Band 12, S. 82-90, 1997.

[Leymann et al_97b] Leymann, F., Roller, D.: *Workflow-based applications.* In: IBM Systems Journal, Vol. 36, No. 1, 1997.

[Lubich_95] Hannes P. Lubich: *Towards a CSCW framework for scientific cooperation in Europe* , Springer, 1995

[Maaß_94] Maaß, S.: *Strömungen, Leitbilder und Begrifflichkeit der Software-Ergonomie. Beiträge zur Entwicklung und Systematisierung eines Fachgebietes.* Habilitationsleistung im Fach Informatik am Fachbereich Informatik der Universität Hamburg, 1994.

[Malone et al_94] T.W. Malone, K. Crowstone: *The Interdisciplinary Study of Coordination*, ACM Computing Surveys, Vol. 26, Nr. 1,S.87-119,1994

[Meyer_90] Meyer, B.: *Objektorientierte Softwareentwicklung.* Carl Hanser Verlag, 1990.

[Medina-Mora et al_92] R. Medina-Mora, T. Winograd, R. Flores, F. Flores: *The Action Workflow Approach to Workflow Management Technology*, in: Proc. ACM Conference on Computer Supported Cooperative Work, Toronto, 1992

[Mintzberg_92] H. Mintzberg: *Die Mintzberg-Struktur: Organisationen effektiver gestalten*, Verlag Moderne Industrie, Landsberg/Lech,1992

[Nastansky et al_96] Nastansky, L., Ott,M.: *Teambasiertes Workflowmanagement und Analyse prozeßorientierter Teamarbeit im Bereich zwischen kooperativer und strukturierter Vorgangsbearbeitung.* http://fb5www.uni-paderborn.de/wi/wi2/ , 1996.

[Oberquelle_91] Horst Oberquelle (Hrsg.): *Kooperative Arbeit und Computerunterstützung: Stand und Perspektiven,* Verlag für Angew. Psychologie Hogrefe, 1991

[Piepenburg_91] U. Piepenburg: *Ein Konzept von Kooperation und die technische Unterstützung kooperativer Prozesse,* in: [Oberquelle_91]

[Prinz_89] W. Prinz: *Survey of Group Communication Models and Systems*, in: Pankoke-Babatz U. (Hrsg.): Computer Based Group Communication – The AMIGO Activity Model, Ellis Horwood, Chichester, England, S. 127-180, 1989

[POSIX_92] IEEE: *Portable Operating System Interface for Computer Environments (POSIX).* IEEE Standard 1003.0. In: Guide to POSIX open Systems Envoronment, 1992

[Pratt et al_98] Pratt, T., Zelkowitz, M.: *Programmiersprachen: Design und Implementierung.* Prentice Hall, 1998.

[Rechenberg et al_97] Rechenberg, P., Pomberger, G.(Hrsg.): *Informatik-Handbuch.* Carl Hanser Verlag, 1997.

[Riggert_98] Riggert, W.: *Betriebliche Informationssysteme*. 1998

[Schäl_96] Schäl, T.: *Workflow Management Systems For Process Organisations*. Springer-Verlag, 1996

[Schuster_97] Schuster, H.: *Architektur verteilter Workflow Management Systeme.* Arbeitsberichte des Instituts für mathematische Maschinen und Datenverarbeitung (Informatik), Dissertation, Band 30, Nummer 6, Erlangen, 1997.

[Schwab et al_95] K. Schwab, H. Ludwig: *Ein ereignisbasierter Ansatz zur Integration von Workflow-Management-Systemen mit Groupware-Werkzeugen*, in: W. Augsburger, H. Ludwig, K. Schwab (Hrsg.): Koordinationsmethoden und -werkzeuge bei der computergestützten

kooperativen Arbeit, Tagungsband, Bamberger Beiträge zur Wirtschaftsinformatik Nr. 30 / 1995

[Taylor_11] Taylor, F. W.: *The Principles of Scientific Management.* New York, London, 1911.

[Teufel et al_95] S. Teufel, C. Sauter, T. Mühlherr, K. Bauknecht: *Computerunterstützung für die Gruppenarbeit*, Addison-Wesley, 1995

[Weiß et al_96] Weiß, D., Krcmar, H.: *Workflow-Management: Herkunft und Klassifikation.* In: Wirtschaftsinformatik 38 (1996) 5, S. 503 – 513

[Weske et al_98] Weske, M., Hündling, J., Kuropka, D., Schuschel, H.: *Objektorientierter Entwurf eines flexiblen Workflow Management Systems.* In: Informatik Forschung Entwicklung, Band 13, S. 179 – 195, 1998.

[WfMC_94] Workflow Management Coalition: *The Workflow Reference Model.* Document number TC00-1003, Document status - Issue 1.1, Author: Hollingsworth, D., 1994. http://www.wfmc.org

[WfMC_96] Workflow Management Coalition: *Terminologie & Glossary.* DocumentNumber WFMC-TC-1011, Document Status - Issue 2.0. 1996. http://www.wfmc.org

[WfMC_98] Workflow Management Coalition: *Interface 1: Process Definition Interchange Process Model*. Document Number WfMC TC-1016-P, Document Status - 7.05 beta. 1998. http://www.wfmc.org

[WfMC_98b] Workflow Management Coalition: *Workflow and Internet: Catalysts for Radical Change.* A WfMC White Paper, 1998. http://www.wfmc.org

[Winograd et al_86] T. Winograd, F. Flores: *Understanding Computers and Cognition: A New Foundation for Design*, Ablex Publishing Corporation, Norwood, NJ:, 1986

[Züllighoven_98] Züllighoven, H.: *Das objektorientierte Konstruktionshandbuch nach dem Werkzeug & Material-Ansatz*. Dpunkt-Verlag 1998

Stand der Internet-Links: 19.5.1999.