# *Vergleich und Bewertung von Production Workflow-Management-Systemen*

Diplomarbeit an der Universität Ulm
Fakultät für Informatik

vorgelegt von:

**Udo Martschat**

1.Gutachter: Prof. Dr. P. Dadam
2.Gutachter: Dr. M. Reichert

2001

# Inhaltsverzeichnis


**1 Einführung** **1**

1.1 Aspekte der Modellierungs-Komponente (Entwicklungsaspekte) . . . 2

1.2 Aspekte der Laufzeitkomponente (Anwendungsaspekte) . . . 2

1.3 Ziele und Gliederung der Arbeit . . . 3

**2 Buildtime** **6**

2.1 MQSeries WF . . . 6

2.1.1 Editor . . . 6

2.1.2 Vorgehensweise . . . 9

2.1.3 Metamodelle . . . 13

2.1.4 WPDL . . . 26

2.2 Staffware . . . 27

2.2.1 Editoren . . . 27

2.2.2 Vorgehensweise . . . 31

2.2.3 Metamodelle . . . 34

2.2.4 WPDL . . . 40

2.3 Tabellarischer Vergleich der Systeme . . . 41

2.4 Kritik und Bewertung . . . 43

2.4.1 Editor . . . 43

2.4.2 Prozessmetamodell . . . 43

2.4.3 Organisationsmetamodell . . . 44

2.4.4 Datenmetamodell . . . 45

2.4.5 Test und Verifikation . . . 45

2.4.6 Tabellarische Bewertung . . . 45

**3 Runtime** **47**

3.1 MQSeries WF . . . 47

3.1.1 Architektur und Plattformen . . . 47

3.1.2 Server-Komponenten . . . 49

3.1.3 Standard-Client . . . 51

3.1.4 Kommunikation Client - Server . . . 60

3.1.5 Datenspeicherung . . . 62

3.2 Staffware . . . 62

3.2.1 Architektur und Plattformen . . . 62

3.2.2 Server-Komponente . . . 64

3.2.3 Standard-Client . . . 66

3.2.4 Kommunikation Client - Server . . . 70

3.2.5 Datenspeicherung . . . 72

**4 Korrektheit und Konsistenz** **73**
4.1 Konsistenzsicherung zur Entwicklungszeit . . . 73
4.1.1 MQSeries WF . . . 74
4.1.2 Staffware . . . 76
4.1.3 Betrachtung ausgewählter Szenarien . . . 78
4.2 Schemaevolution . . . 85
4.2.1 MQSeries WF . . . 85
4.2.2 Staffware . . . 85
4.2.3 Betrachtung ausgewählter Szenarien . . . 87
4.2.4 Versionierung . . . 95
4.3 Bewertung . . . 95

**5 Schnittstellen der Systeme** **96**
5.1 Das Referenzmodell der WfMC . . . 96
5.1.1 Import/Export von Prozessmodellen (Interface 1) . . . 98
5.1.2 Client- und Anwendungsprogrammierung (Interface 2 und 3) . . . 99
5.1.3 Interoperabilität zwischen WFMS (Interface 4) . . . 102
5.1.4 Administrations- und Monitoringwerkzeuge (Interface 5) . . . 105
5.2 Weitere Standards . . . 106
5.2.1 OMG Workflow Management Facility Standard (JointFlow) . . . 106
5.2.2 SWAP (Simple Workflow Access Protocol) . . . 106
5.3 MQSeries WF . . . 107
5.3.1 Import/Export von Prozessmodellen . . . 108
5.3.2 Client- und Anwendungsprogrammierung . . . 108
5.3.3 Interoperabilität zwischen WFMS . . . 113
5.3.4 Administrations- und Monitoringwerkzeuge . . . 114
5.4 Staffware . . . 116
5.4.1 Import/Export von Prozessmodellen . . . 117
5.4.2 Client- und Anwendungsprogrammierung . . . 118
5.4.3 Interoperabilität zwischen WFMS . . . 124
5.4.4 Administrations- und Monitoringwerkzeuge . . . 124
5.5 Vergleich der Systeme . . . 129

**6 Fortschrittliche Funktionalitäten** **130**
6.1 Ad-hoc-Änderungen . . . 130
6.2 Robustheit von WFMS . . . 132
6.2.1 Revovery von WMFS (*Phoenix Behavior*) . . . 132
6.2.2 Backup . . . 134
6.3 Animation und Simulation . . . 134
6.4 Zeitmanagement . . . 135
6.5 Sicherheit . . . 135
6.6 Performance und Skalierbarkeit . . . 137
6.7 Transaktionskonzepte . . . 138

**7 Zusammenfassung und Ausblick** **139**

**A Installationsvoraussetzungen** **141**

**B Dokumentation der Systeme** **143**

**C WPDL** **144**
C.1 WFMC . . . . . 144
C.2 MQSeries WF - FDL . . . . . 145
C.3 Staffware - XFR . . . . . 147

**D Datenbank-Tabellen-Struktur (Staffware)** **150**

**E Glossar** **151**

**Literaturverzeichnis** **152**

# Abbildungsverzeichnis


1.1 Trennung von Ablauflogik und Anwendungen in WFMS . . . . . 1
1.2 Zusammenhänge in einem WFMS (aus [Fis01]) . . . . . 4

2.1 Der Aufbau der Buildtime-Komponente . . . . . 7
2.2 Buildtime nach erfolgreichem Login . . . . . 8
2.3 Buildtime mit Arbeitsbereich . . . . . 8
2.4 Die Werkzeugleiste . . . . . 9
2.5 Die Organisationsstruktur . . . . . 14
2.6 Probleme mit der Vertretungsregelung bei MQSeries WF . . . . . 16
2.7 Prozess-Metamodell von MQSeris WF . . . . . 17
2.8 Deadlocks durch Schleifen . . . . . 19
2.9 Graphische Darstellung der Ausgangslage . . . . . 20
2.10 Problem bei der Deathpath-Elemination . . . . . 21
2.11 Ergebnis nach der Deathpath-Elemination . . . . . 21
2.12 Bearbeiterzuordnung mittels Container . . . . . 23
2.13 Einstellung eines 10sek. Zeitintervalles . . . . . 24
2.14 Gültigkeit von Prozessen . . . . . 25
2.15 Der Aufbau der Buildtime-Komponente . . . . . 27
2.16 Staffware Client . . . . . 28
2.17 Der Graphical-Workflow-Designer . . . . . 29
2.18 Der Graphical-Workflow-Designer mit Prozessmodelldiagramm . . . . . 29
2.19 Der Graphical-Formular-Designer mit Feldern . . . . . 30
2.20 Das Administrations-Tool zur Erstellung des Organisationsmodells . . . . . 31
2.21 Die Organisationsstruktur (Staffware) . . . . . 35
2.22 Möglichkeiten zur Verbindung von Aktivitäten . . . . . 37
2.23 Definition einer Bearbeiterzuordnung in Staffware . . . . . 38
2.24 Deadlines mittels Zeitraum bzw. Ausdruck . . . . . 39
2.25 nicht mögliche Bearbeiterzuordnung . . . . . 44

3.1 Architektur der MQSeries WF Runtime-Komponente . . . . . 47
3.2 Systemnetz von MQSeries WF (aus [IBM99] . . . . . 49
3.3 Server-Komponente von MQSeries WF . . . . . 50
3.4 Runtime-Client von MQSeries WF . . . . . 52
3.5 Runtime-Client von MQSeries WF (incl. Listen) . . . . . 53
3.6 Erstellen einer neuen Liste in MQSeries WF Runtime . . . . . 54
3.7 graphische Überwachung von Instanzen in MQSeries WF . . . . . 55
3.8 Ausführungszustände von Prozessinstanzen in MQSeries WF [IBM99e] . . . . . 56
3.9 Übertragen von *Workitems* im Zustand `bereit` oder `ausgesetzt` . . . . . 57
3.10 Ausführungszustände von *Workitems* in MQSeries WF - Prozessinstanz im Zustand `running` [IBM99e] . . . . . 58
3.11 Ausführungszustände von *Workitems* in MQSeries WF - Prozessinstanz im Zustand `suspended` [IBM99e] . . . . . 59

3.12 Ausführungszustände von *Workitems* in MQSeries WF - Prozessinstanz im Zustand `terminated` [IBM99e] . . . 59
3.13 Kommunikation zwischen Client und Server mittels MQSeries . . . 61
3.14 Verlauf einer Anfrage durch den ORB [OMG01] . . . 62
3.15 Architektur der Runtime-Komponente von Staffware 2000 . . . 63
3.16 Das Mulitnode-Konzept von Staffware . . . 64
3.17 Der Standard-Client von Staffware . . . 66
3.18 Der Standard-Client von Staffware mit graphischer Darstellung der Workitem-Liste 68
3.19 Kommunikationsaufbau zwischen Client und Server . . . 71
3.20 Kommunikation zwischen Client und WQS/WIS nach erfolgreichem Verbindungsaufbau . . . 72

4.1 Unzureichende Versorgung eines Eingangscontainer . . . 78
4.2 Lost Update durch unsynchronisiertes schreiben eines Eingangscontainer . . . 80
4.3 Zyklen ohne Abbruchbedingung . . . 80
4.4 Nicht ausreichende Transitionsbedingungen . . . 81
4.5 Verschachtelungen durch Subprozesse . . . 82
4.6 Fehlende Bearbeiterzuordnung . . . 83
4.7 Instanziieren unvollständiger Prozessmodelle in Staffware . . . 86
4.8 Dokumentation unvollständiger Prozessmodelle in Staffware . . . 86
4.9 Einfügen neuer Aktivitäten in das Prozessmodell . . . 88
4.10 Löschen einer Aktivität des Prozessmodells . . . 89
4.11 Starten einer gelöschten Aktivität . . . 90
4.12 Abänderung des Kontrollflusses . . . 90
4.13 Nicht zustellbare Workitems in Staffware . . . 92
4.14 Ausgabe des Export-/Import-Dienstprogramms . . . 93

5.1 Das Workflow-Referenzmodell - Komponenten und Schnittstellen (aus [WFMC95]) 97
5.2 Workflow-Standards und verwandte Dokumente . . . 98
5.3 Zusammenhang von Prozessmodellen, -instanzen und Arbeitslisten (aus [WFMC99b]) 100
5.4 WAPI-Funktionen der Schnittstelle 2 . . . 101
5.5 Die Funktionsweise der Schnittstelle 2 . . . 102
5.6 Kommunikation mittels WF-XML [WFMC00a] . . . 103
5.7 WF-Objektmodell der OMG (aus [OMG00]) . . . 107
5.8 Komponenten und Tools von MQSeries WF . . . 108
5.9 Der Aufbau der MQSeries WF Client-Komponente . . . 109
5.10 Die API-Struktur von MQSeries WF (aus [IBM99e]) . . . 109
5.11 Die Architektur des MQSeries WF WebClient (aus [IBM01]) . . . 111
5.12 Das Zusammenspiel von MQSeris WF und Domino WF (aus [IBM01c]) . . . 114
5.13 Auswahlmenü des Verwaltungsprogramms (aus [IBM99]) . . . 115
5.14 Instanz-Überwachung im Standard-Client . . . 116
5.15 Komponenten und Tools von Staffware (Schnittstellen 1,2,3 und 5) [SW01b] . . . 117
5.16 Staffware Client-Typen [SW99d] . . . 118
5.17 Login der SEO VB Sample Application . . . 120
5.18 Die Architektur des Staffware WebClient . . . 121
5.19 Die Kommunikation zwischen Staffware und einem externen Client [SW99a] . . . 123
5.20 graphische Darstellung ausgewählter Arbeitslisten . . . 128

6.1 Ad-hoc-Änderungen (aus [Dad99]) . . . 131
6.2 Recovery bei WFMS (aus [LeRo00]) . . . 133

C.1 Das Prozessmodelldiagramm (fmccred.fdl) . . . 146

C.2 Das Prozessmodelldiagramm (loan.xfr) . . . . . 148

D.1 Datenbank-Tabellen-Struktur in Oracle 8i (aus [SW00f]) . . . . . 150

# Tabellenverzeichnis


1.1 Die verwendeten WFMS . . . . . 3
2.1 Kontrollflusskonstrukte (MQSeries Workflow) . . . . . 19
2.2 Vergleich der Buildtime-Komponenten . . . . . 41
2.3 Bewertung der Buildtime-Komponenten . . . . . 46
3.1 Plattformen für die MQSeries WF Runtime-Komponenten . . . . . 48
3.2 Plattformen für die Staffware Runtime-Komponenten . . . . . 63
3.3 Visuelle Darstellung von Statusänderungen einer Arbeitsliste . . . . . 67
3.4 Visuelle Darstellung des Status eines Workitem . . . . . 68
3.5 Die Optionen der Menüleiste des Staffware-Clients . . . . . 69
4.1 Warnungen und Fehlermeldungen der Option verify . . . . . 74
4.2 Abweisungen von Änderungen am Prozessmodell . . . . . 75
4.3 Abweisungen von Änderungen oder unvollständigen Eingaben am Prozessmodell . 77
5.1 Die fünf Schnittstellen der WFMC . . . . . 97
5.2 Die Struktur von WF-XML-Nachrichten . . . . . 104
5.3 Konformität von MQSeries WF zu den Schnittstellen der WFMC . . . . . 107
5.4 Administrations- und Überwachungswerkzeuge von MQSeries WF . . . . . 114
5.5 Konformität von Staffware zu den Schnittstellen der WFMC . . . . . 116
5.6 SEO-Client Toolkits . . . . . 119
5.7 Entwicklung von Anwendungen . . . . . 124
5.8 Administrations- und Überwachungswerkzeuge . . . . . 125
A.1 Installationsvoraussetzung der verwendeten WFMS . . . . . 141
B.1 Die mitgelieferte Dokumentation der WFMS . . . . . 143
E.1 Abkürzungsverzeichnis . . . . . 151

# Kapitel 1

# Einführung

Workflow-Management-Systeme (WFMS) sind eine relativ neue Technologie zur Realisierung, Verwaltung und Wartung prozessorientierter Anwendungen. Konventionelle WFMS unterstützen die (teilweise) Ausführung von Geschäftsprozessen entsprechend vordefinierter Arbeitsabläufe. Eine für ein WFMS verständliche Darstellung eines Arbeitsablaufs wird als Workflow (WF) bezeichnet. Workflow-Management ist dem entsprechend die komplette oder teilweise Automatisierung von (Geschäfts-) Prozessen, bei der Dokumente, Informationen (Daten) und Aufgaben von einem Anwender zum anderen gemäß prozessorientierter Regeln weitergeleitet werden [WFMC95]. Typisch für WFMS ist die Aufteilung der Anwendungssysteme in einen Ablauf- und in einen Funktionsteil. Letzterer besteht aus einer Sammlung von Programmbausteinen (siehe Abbildung 1.1) besteht.

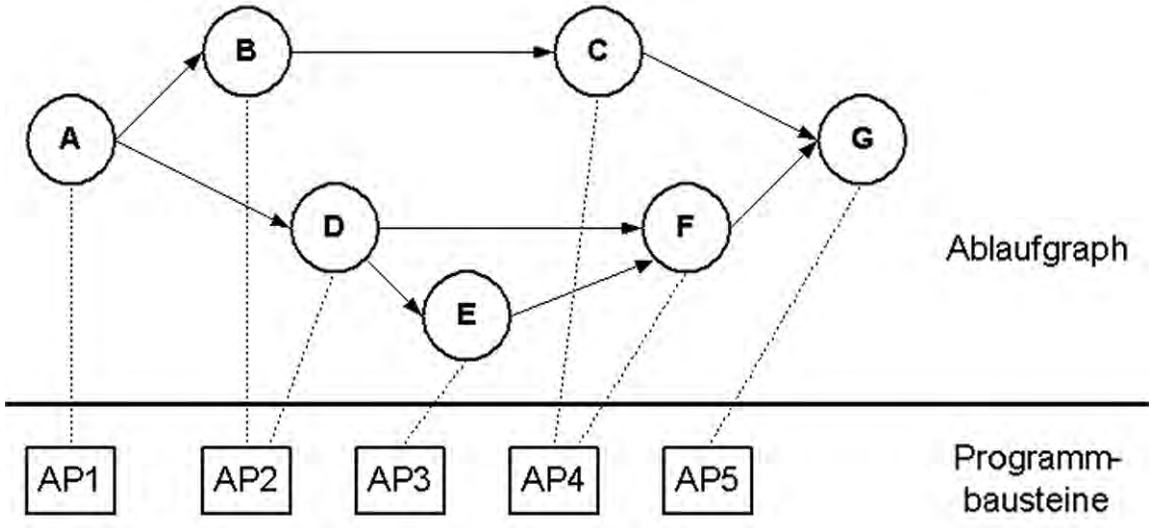



Abbildung 1.1: Trennung von Ablauflogik und Anwendungen in WFMS

Durch diese Aufteilung werden die verschiedenen Aspekte eines WF, wie Tätigkeiten, zuständige Bearbeiter, Ausführungsreihenfolgen und -bedingungen (Kontrollflüsse) sowie Informationsflüsse (Datenflüsse) von der Implementierung der Bearbeitungsschritte (in Form von Programmbausteinen bzw. Anwendungskomponenten) getrennt. Zur Erfüllung dieser Aufgaben enthält ein WFMS Komponenten für die Spezifikation, Verifikation, Steuerung und Überwachung von Workflows. Diese Systemkomponenten können grob in die Klassen Modellierungs- und Laufzeitkomponenten (*Buildtime- und Runtime-Komponenten*) eingeteilt werden.

## 1.1 Aspekte der Modellierungs-Komponente (Entwicklungsaspekte)

Ein Geschäftsprozess umfaßt Abschnitte, die computergestützt bearbeitet werden können und solche, deren Ausführung nicht durch den Computer unterstützt werden [LeRo00]. Die vom Computer unterstützten Abschnitte werden durch das `Prozessmodell` (*WF-Schema, WF-Modell*) beschrieben.
Zu dessen Entwicklung muß die Modellierungskomponente des WFMS eine getrennte Abbildung von Ablauf- und Funktionsteil ermöglichen (siehe Abbildung 1.1). Die Entwicklung des Ablaufteils beinhaltet die Definiton der einzelnen Prozessschritte (*Aktivitäten*, z.B. Tätigkeiten eines Geschäftsprozesses) und deren Eigenschaften, sowie die Modellierung des Kontroll- und Datenflusses zwischen ihnen. Dieser Teil steht für die Prozesslogik eines Anwendungssystems. Während der Erstellung des Funktionsteils wird für jeden einzelnen Prozessschritt ein Programmbaustein (*Invoked Applications*), entsprechend der geforderten Funktionalität, ausgewählt.

Die vier Hauptvorteile dieser Aufteilung in der Entwicklung von Prozessmodellen sind [LeRo97]:

- `Flexibilität` (*Flexibility*) (Die Ausführungsreihenfolge von Aktivitäten kann geändert werden, ohne Änderungen an den Programmbausteinen vornehmen zu müssen)
- `Integrierbarkeit` (*Integration*) (Programmbausteine können (theoretisch) in vielen WFMS bzw. WFMS-basierten Anwendungen eingesetzt werden)
- `Skalierbarkeit` (*Scalability*) (Die Realisierung einfacher bzw. kleiner wie auch sehr umfangreicher WF-Anwendungen)
- `Wiederverwendbarkeit` (*Reusability*) (Programmbausteine können in beliebig vielen Prozessmodellen wiederverwendet werden)

Ein WFMS kann verschiedene Prozesstypen, im Sinne von Geschäftsprozessen (z.B. Bearbeitung eines Kundenauftrages, Abwicklung eines Schadensfalls, Behandlung eines Patienten), unterstützen. Entsprechend der oben beschriebenen Entwicklung muß dafür mit einer Modellierungskomponente für jeden Prozesstyp ein entsprechendes Prozessmodell erstellt werden. Die Möglichkeiten zur Erstellung von Prozessmodellen sind durch die vom WFMS bereitgestellten WF-Metamodelle (*WF-Beschreibungssprache*) bestimmt. Deren Ausdrucksmächtigkeit legt fest, in wie weit mehr oder weniger komplexe Geschäftsprozesse abgebildet werden können. Viele WF-Metamodelle er-möglichen zudem die hierarchische Strukturierung von Prozessmodellen. Dazu werden spezielle Aktivitäten verwendet, welche eine Menge von Aktivitäten in Form eines separat modellierten Prozessmodells darstellen. Desweiteren sind automatische Aktivitäten möglich, welche ohne Benutzerinteraktion vom WFMS ausgeführt werden. "Normale" Aktivitäten werden von einer zur Ausführung berechtigten Person gestartet. Dazu verfügt eine Person über eine Arbeitsliste (*Worklist*), deren Einträge (*Workitems*) für die von ihr ausführbaren Aktivitäten stehen. Startet eine andere berechtigte Person eine solche Aktivität, werden die zugehörigen Workitems aus allen anderen Arbeitslisten entfernt.

## 1.2 Aspekte der Laufzeitkomponente (Anwendungsaspekte)

Eine berechtigte Person kann dem WFMS bekannte Prozessmodelle auswählen und starten. Dieser Vorgang erzeugt eine Instanz des ausgewählten Prozessmodells (*Prozess-Instanz, WF-Instanz*). Die Instanzen eines Prozessmodells können unabhängig voneinander, d.h. in einem unterschiedlichen

Kontext, ausgeführt werden. Die Aktivitäten einer Prozessinstanz werden als *Aktivitäteninstanzen* bezeichnet. Eine Aktivitätsinstanz kann in Form von Workitems einem oder mehrern Personen zur Bearbeitung angeboten werden.
Die Laufzeitkomponenten eines WFMS (*WF-Engine*) sind zuständig für:

- die Ausführung der WF-Instanzen
- die Verwaltung instanzspezifischer Daten (z.B. WF-Historiesn, Steuerungsrelevante Daten)
- die Verteilung anstehender Aktivitäten an die berechtigten Personen
- die Ausführung automatischer Aktivitäten
- den Aufruf der zu einer Aktitivität gehörenden Programmbausteine

Daraus entstehen für den WFMS-Anwender einige Vorteile:

- Die Ausführung eines Prozesses wird vom System aktiv überwacht. Dadurch können menschliche Fehler (z.B. Vergessen eines Arbeitsschrittes) reduziert werden.
- Die Ausführung eines Prozesses wird während der Ausführung dokumentiert und bietet so eine Vielzahl wichtiger Betriebsdaten (z.B. Dauer einer Tätigkeit, Kosten eines Prozesses).

In Abbildung 1.2 werden die Zusammenhänge zwischen Geschäftsprozess, Prozessmodell, Aktivitäten, WFMS, Prozessinstanz, Aktivitätsinstanz, Workitems und Programmbausteinen graphisch dargestellt. Der linke Zweig steht für die Aufgaben der Modellierungskomponente und der rechte Zweig entsprechend für die Aufgaben einer Laufzeitkomponente.

## 1.3 Ziele und Gliederung der Arbeit

Mit dieser Arbeit sollen die aktuellen Einsatzmöglichkeiten und Probleme von WFMS betrachtet werden. Daneben ist die Feststellung fehlender oder mangelnder Funktionalitäten in allen Bereichen eines WFMS ein wichtiger Bestandteil dieser Arbeit.
Viele WFMS weisen eine dokumentorientierten Ausrichtung auf. Bei dieser Variante wird in der Literatur oft die Metapher der Handakte gewählt, welche verschiedene Arbeitsstationen durchläuft [Dad99]. Aus heutiger Sicht ist die mangelnde Fähigkeit zur Integration fremder Anwendungen der größte Nachteil dieser Art von WFMS. Einen Lösungsansatz stellen die *Production Workflow Management Systeme* dar [LeRo00]. Durch sie wurde die bereits beschriebene Trennung von Prozesslogik und Anwendungsprogrammen eingeführt, wodurch eine verbesserte Integrierbarkeit für Anwendungen möglich wird.
Zwei der bedeutensten Vertreter von Production WFMS sind MQSeries WF und Staffware. Sie werden in dieser Arbeit einer systematischen und umfassenden Evaluation unterzogen.
Die Arbeit teilt sich in theoretische und praktische Anteile. Für die praktischen Bereiche wurden Test mit den beiden WFMS **MQSeries Workflow** von IBM und **Staffware 2000** von Staffware plc. durchgeführt. Die Tabelle 1.1 gibt einen Überblick über die verwendeten WFMS.

| Produkt | **MQSeries Workflow** | **Staffware 2000** |
|---|---|---|
| Version | `3.2.1` | `8.1` |
| Hersteller | `IBM` | `Staffware plc.` |
| Betriebssystem | `Microsoft Windows NT 4.0` | `Microsoft Windows NT 4.0` |
| gefordertes Service Pack (SP) | `SP3` | `SP4` |
| Zusätzliche Komponenten | `WebClient` | `WebClient` |

Tabelle 1.1: Die verwendeten WFMS

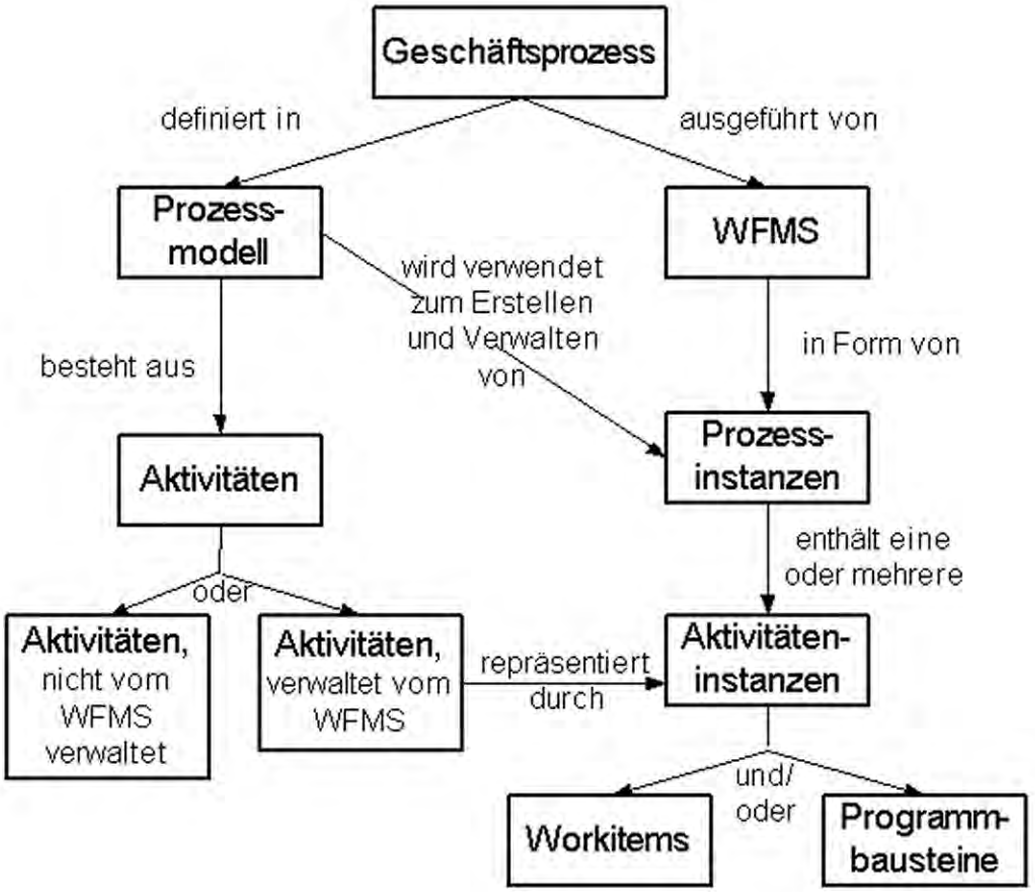



Abbildung 1.2: Zusammenhänge in einem WFMS (aus [Fis01])

Beide WFMS wurden auf einen Test-Computer installiert. Dieser verfügt über das Betriebssystem Windows NT4.0 (*SP 4*) von Micorsoft und einen 650 Mhz-Prozessor (AMD-Duron). Weitere Softwareinstallationen, mit Ausnahme der vom WFMS benötigten Software, wurden nicht getätigt. Eine vollständige Beschreibung der Installationsvoraussetzungen ist im Anhang A zu finden.

In Kapitel 2 werden die Buildtime-Komponenten der beiden WFMS besprochen. Neben einer Darstellung der Modellierungsmöglichkeiten erfolgt ein Vergleich, eine Bewertung und eine Kritik an diesen beiden Buildtime-Komponenten.

Kapitel 3 beschäftigt sich mit den Runtime-Komponenten der beiden WFMS. Wichtige Aspekte sind die grundlegende Architekur und die angebotenen Plattformen. Zu den Runtime-Komponenten zählen die Server-Komponenten, die Clients und die gebotenen Möglichkeiten zur Datenspeicherung. Im weiteren wird die Kommunikation der verschiedenen Komponenten untereinander betrachtet.

Kapitel 4 untersucht die Unterstützung des Entwicklers bei der Erstellung und Wartung von Prozessmodellen und -instanzen. Ein WFMS muß gewährleisten, daß Prozessmodelle und -instanzen stets korrekt und konsistent sind.

Kapitel 5 stellt aktuelle Standardisierungsbemühungen im Bereich der WFMS vor. Die vorhandenen Schnittstellen der beiden Test-WFMS werden auf Funktionalität und Konformität zu den Standards geprüft und (teilweise) beispielhaft implementiert.

Kapitel 6 widmet sich fortschrittlichen WF-Funktionalitäten. Diese sind größtenteils in heutigen (und in den beiden Test-WFMS) noch nicht implementiert. Es soll aufgezeigt werden, welche Funktionen den heutigen WFMS fehlen und warum sie benötigt werden.

Kapitel 7 fasst die Ergebnisse dieser Arbeit zusammen und gibt einen Ausblick über die weitere Entwicklung von WFMS. Es werden notwendige Erweiterungen heutiger WFMS besprochen und mit Ankündigungen der Hersteller der beiden Test-WFMS verglichen.

# Kapitel 2

# Buildtime

Die Buildtime-Komponente eines WFMS stellt die Funktionen zum Erstellen, Testen und Verwalten aller zu einer prozessorientierten Anwendung gehörenden Daten bereit. Diese Daten beinhalten die folgenden Dimensionen:

- Prozessmodell
- Organisationsmodell
- Datenmodell
- Anwendungskomponenten

Alle Daten werden in einer Datenbank des WFMS gespeichert. Zusätzlich bietet die Buildtime-Komponente Funktionen zum Festlegen und Verwalten administrativer und systembezogener Daten an. Dazu gehören z.B. Angaben darüber, wie oft sich ein bestimmter Benutzer gleichzeitig am WFMS anmelden darf. Sichtbar wird die Buildtime-Komponente in zweierlei Hinsicht. Zum einem beim Erstellen und Verwalten der genannten Dimensionen mittels einer graphischen Endbenutzerschnittstelle (Editor/GUI), zum anderen durch eine (standardisierte) "Workflow-Sprache" (WPDL="*workflow process definition language*" [WFMC00a]). Die WPDL dient zum Informationsaustausch zwischen verschiedenen WFMS sowie zwischen Geschäftsprozessmodellierungswerkzeugen.

## 2.1 MQSeries WF

In diesem Abschnitt werden die zugrundeliegenden Konzepte und Funktionen der Buildtime-Komponente von MQSeries WF V3.2.1 dargestellt. Dazu gehören verschiedene Metamodelle, deren Mächtigkeit und die Verständlichkeit bzw. Strukturiertheit der graphischen Oberfläche. Um die Arbeit mit der Buildtime-Komponente möglichst effizient zu gestalten, müssen zusätzlich die Abhängigkeiten bzw. das Zusammenspiel der verschiedenen Modellierungselemente betrachtet werden. Die Bestandteile der Buildtime-Komponente und deren Funktionen werden in Abbildung 2.1 dargestellt.

### 2.1.1 Editor

Nach der Installation existiert im Startmenü unter `Programme` ein neuer Eintrag namens `MQSeries Workflow`. Er beinhaltet den Punkt `MQSeries Workflow Buildtime`, bei dessen Auswahl die Buildtime-Komponente startet. Als Standardlogin nach der Installation gilt der USER:ADMIN und das PASSWORD:password [IBM99a]. Nach dem erfolgreichen Anmelden erscheint das Buildtime-Fenster. Am unteren Ende des Fensters wird die Statusleiste angezeigt. Diese enthält grundlegende Informationen, wie den Namen der benutzten Datenbank und den Login-Namen des aktuellen Benutzers.

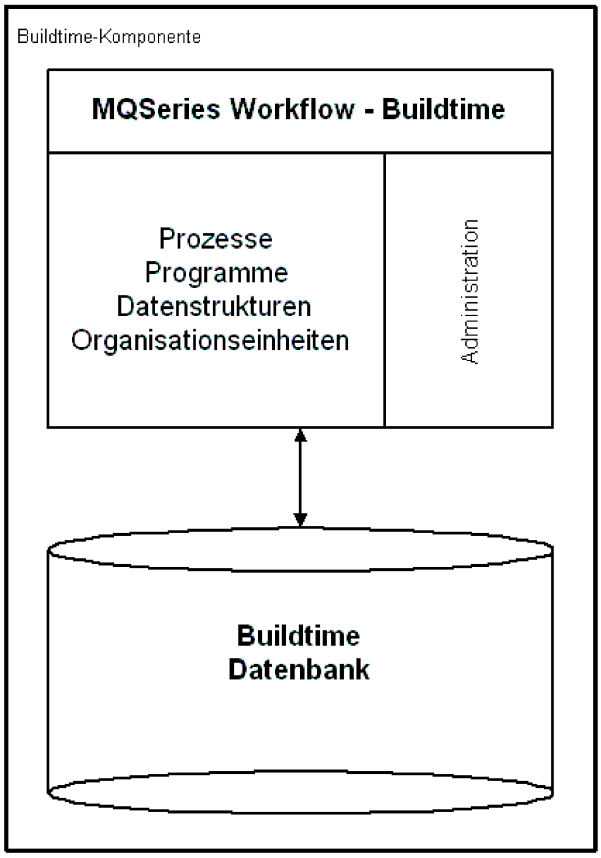



Abbildung 2.1: Der Aufbau der Buildtime-Komponente

Das Buildtime-Fenster unterteilt sich in zwei Bereiche (siehe Abbildung 2.2).

- Im linken Bereich erscheint in Form einer Baumstruktur die Darstellung aller beteiligten Objekte. Die Register oberhalb dieses Bereiches ermöglichen den schnellen Wechsel zwischen den verschiedenen Baumstrukturen für die Objekte Prozessmodelle (*Processes*), Mitarbeiter (*Staff*), Netzwerk (*Network*) und Anwendungen (*Implementations*). Bei der Auswahl des Registers `Prozesse` werden alle in der Buildtime-Datenbank gespeicherten Prozesskategorien, Prozessemodelle und Aktivitäten angezeigt. Kategorien dienen zur Gliederung der Prozessmodelle. In Abbildung 2.3 werden die Kategorien "orgtest" und "No category assigned" angezeigt. Unter dem Register `Mitarbeiter` werden die organisatorischen Elemente Personen (*Person*), Rollen (*Role*), Organisationseinheiten (*Organisation*) und Stufen (*Level*) dargestellt. Im Register `Netzwerk` werden die Bestandteile des Gesamtsystems hierarchisch geordnet zusammengefasst. Dies beinhaltet die Modellelemente Domain, Systemgruppen und Server. Im Bereich `Anwendungen` finden sich die für die Festlegung von Ein-/Ausgabecontainern definierten Datenstrukturen und die registrierten Programme. In der Abbildung 2.2 sind die Prozesskategorien "orgtest" und "No category assigned" sichtbar. Die Kategorie "orgtest" enthält drei Prozesse ( siehe Abbildung 2.3). Jedes Prozessmodell untergliedert sich in mehrere Aktivitäten.

- Der rechte Bereich wird als Arbeitsbereich bezeichnet. Er ist zu Beginn leer (Abbildung 2.2). Hier werden die Prozessmodelldiagramme erstellt und angezeigt. Dazu muß mit einem Rechtsklick auf ein Prozessmodell, im linken Bereich, dessen Befehlsmenü geöffnet und der Punkt `Diagram` ausgewählt werden (siehe Abbildung 2.3).

Abbildung 2.2: Buildtime nach erfolgreichem Login

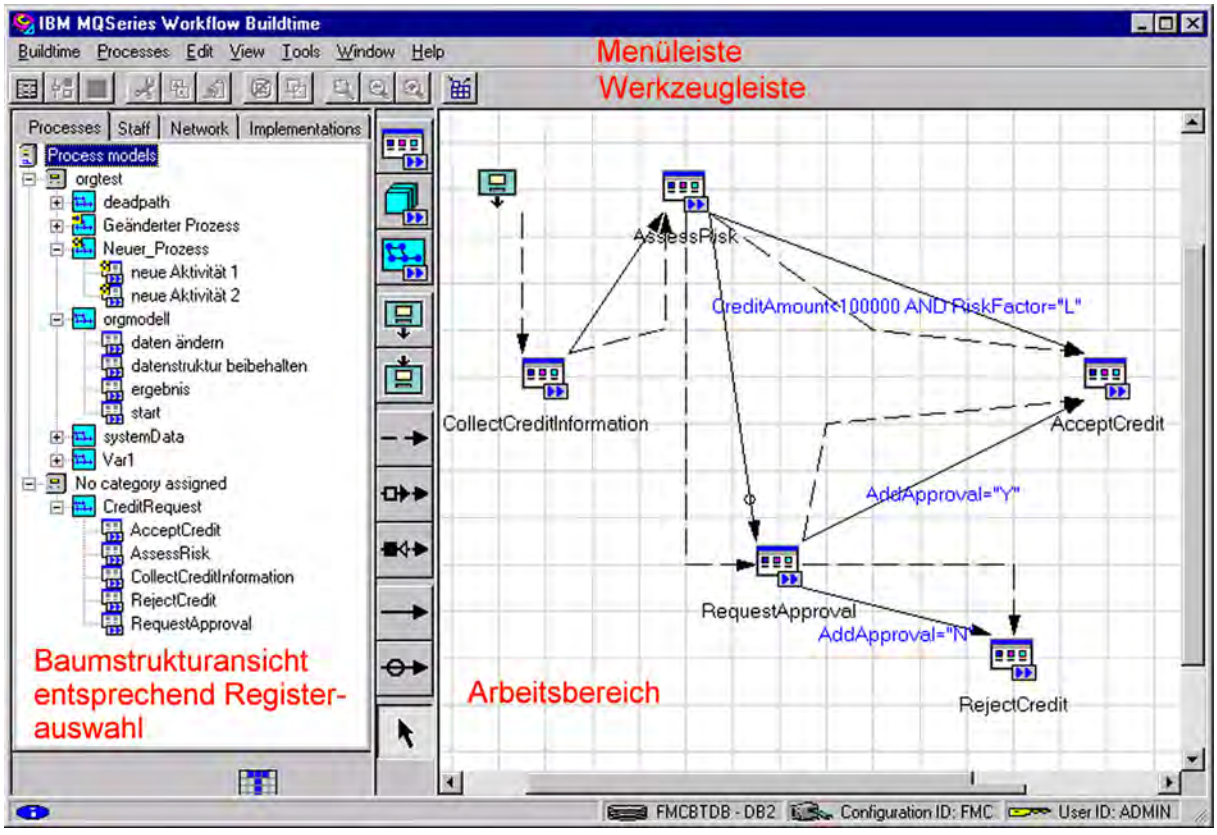



Abbildung 2.3: Buildtime mit Arbeitsbereich

Zusätzlich zu den oben genannten Bereichen enthält das Buildtime-Fenster noch eine Menü- und Werkzeugleiste. Es können Funktionen aus der Menüleiste ausgewählt oder die Werkzeugleiste (siehe Abbildung 2.4) verwendet werden. Die angebotenen Funktionen sind abhängig von der dargestellten Baumstruktur oder Fenstermarkierung. Wird beispielsweise im linken Bereich das Register `Staff` ausgewählt, erscheint entsprechend die zugehörige Baumstruktur. Die Menü- und Werkzeugleiste enthält dann nur Funktionen, die speziell für die Definition von Mitarbeiterstrukturen benötigt werden.

Wird die Diagrammansicht eines Prozessmodells aufgerufen, erscheint zwischen dem linken Bereich und dem Arbeitsbereich automatisch eine Leiste mit einer Auswahl an Elementen zur Gestaltung von Prozessmodelldiagrammen. Eine detaillierte Beschreibung dieser Elemente findet in Abschnitt 2.1.3.2 statt. Die graphische Darstellung dieser Leiste kann ausgeschaltet werden. Dazu muß zuerst das Diagramm-Fenster aktiv sein. Nun erscheint in der Menüleiste unter `View` der Unterpunkt `Drawing Tools` mit den Optionen `show`, `hide` und `customize`. Die jeweils aktuell geltende Einstellung ist mit einem Häckchen versehen. Wird `hide` ausgewählt, verschwindet die Leiste und der Arbeitsbereich vergrößert sich um diesen Bereich.

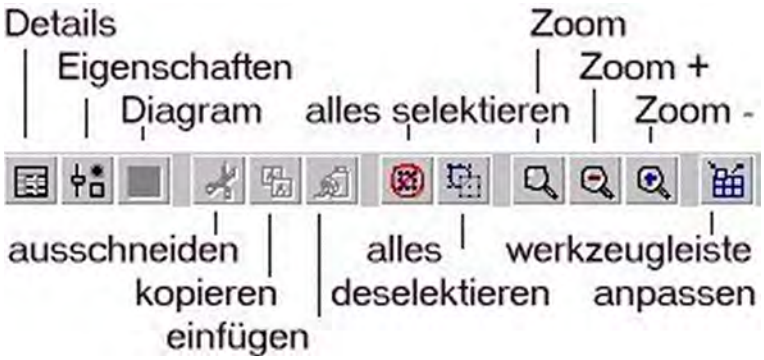



Abbildung 2.4: Die Werkzeugleiste

### 2.1.2 Vorgehensweise

Während der Erstellung eines Prozessmodelles müssen sukzessive alle prozessrelevanten Metadaten in die Buildtime eingegeben werden. Da bei MQSeries WF für einige Modellierschritte gewisse Vorbedingungen erfüllt sein müssen - z.B. können Programme erst korrekt registriert werden, wenn die nötigen Datenstrukturen existieren - sollte eine bestimmte Reihenfolge bei der Erstellung des Prozessmodells eingehalten werden. Die folgende Vorgehensweise stellt sicher, daß für jeden Modellierschritt alle Vorbedingungen erfüllt sind [IBM99c]. Dabei wird vereinfachend davon ausgegangen, daß ein komplettes Modell neu ( *"from scratch"*) erstellt werden soll.
Anlegen der

1. Mitarbeiterstruktur des Unternehmens, einschließlich der Rollen und Stufen die zur Beschreibung des Unternehmens benötigt werden.

2. Netzwerkeigenschaften für die Domain- und Serverkomponenten (z.B. Daten des Prüfprotokoll speichern).

3. Datenstrukturen, für die Beschreibung der Eingabe- und Ausgabecontainer von Prozessen, Aktivitäten und Programmen.

4. Registrierung der Anwendungsprogramme für die Aktivitäten

5. Prozessmodelldiagramme, welche alle Aktivitäten, Blöcke und Subprozesse zeigen (mit zugehörigen Daten- und Kontrollflussverbindungen (*Konnektoren*)). Für jedes Prozessmodell müssen zudem seine Eigenschaften (z.B. die Gültigkeit des Prozessmodells, siehe Abbildung 2.14 ) festgelegt werden.

6. Logik hinter dem Prozessmodelldiagramm:

   - Für jede Aktivität werden die Start- und Endbedingungen, die potentiellen Bearbeiter, die Ein- und Ausgabecontainer und die Anwendungsprogramme festgelegt.

- Für den Kontrollfluss kann jeder Kontrollflusskonnektor optional mit einer Transitionsbedingung versehen werden.
- Für jeden Datenflusskonnektor muß festgelegt werden, wie die Daten aus dem Outputcontainer einer Quell-Aktivität auf den Inputcontainer der Ziel-Aktivität abgebildet werden.

Das Anlegen neuer Elemente eines Objektes erfolgt immer durch Rechtsklick mit der Maus auf ein Objekt. Dadurch öffnet sich ein Befehlsmenü. In diesem Menü finden sich die möglichen Operationen für dieses Objekt. Mit der Auswahl von `new Object` öffnet sich ein Eigenschaftenfenster dieses neuen Objektes. Nach Eingabe und Bestätigung der Eigenschaften ist das neue Objekt erstellt.
In den folgenden Abschnitten wird kurz auf jede dieser Vorbedingungen eingegangen. Abschließend erfolgt eine Darstellung der Möglichkeiten zur Verifizierung und Speicherung der Daten.

#### 2.1.2.1 Definition von Organisations- und Personalstrukturen

In der ( Baumstruktur-)Ansicht `Staff` existieren die Objekte Personen, Rollen, Organisationen und Level. Das Objekt Level enthält die zehn Elemente 0-9. Es kann weder ein Element gelöscht noch hinzugefügt werden. Lediglich die Beschreibung und der Name eines konkreten Levels kann geändert werden. Bei den anderen drei Objekten können neue Elemente dazugefügt oder existierende Elemente entfernt werden. Beim Anlegen eines Elementes werden sowohl optionale als auch obligate Eigenschaften verlangt. Obligat ist beispielsweise der Name des neuen Elementes, während eine textuelle Beschreibung immer optional ist. Auf das Fehlen obligater Beschreibungsdaten wird mit einer Fehlermeldung hingewiesen und das Anlegen dieses Elementes verhindert. Zu den Personen-Eigenschaften, wie Name oder Passwort, gehören auch Zugehörigkeiten zu Rollen und/oder einer Organisation. Nach dem Eintragen und Bestätigen der Eigenschaften existiert diese neue Person in der Buildtime-Komponente. Aus den Abhängigkeiten der Objekte (siehe Abbildung 2.5) ergibt sich folgende Modellierungsreihenfolge:

- Level
- Personen
- Rollen und Organisationen
- Beziehungen zwischen Rollen und Personen
- Beziehungen zwischen Organisationen und Personen

Querbezüge zwischen Objekten werden von der Buildtime-Komponente überwacht. Werden neue Objekte hinzugefügt, werden damit verbundene Querbezüge zu existierenden Objekten eingefügt. Das bedeutet, dass wenn einer Person eine bestimmte Rolle zugewiesen wird, diese Person automatisch bei Ansicht dieser Rolle als Mitglied angezeigt wird. Werden existierende Objekte entfernt, so wird geprüft ob Querbezüge zu anderen Objekten bestehen. Ist es möglich, existierende Querbezüge zu entfernen, wird dies gemacht. Ansonsten wird das Löschen des Objektes mit aussagekräftiger Fehlermeldung verhindert (z.B. wenn eine Person gelöscht werden soll, die Manager einer Organisation ist).

#### 2.1.2.2 Erstellen von Netzwerkeigenschaften

Die Darstellung des Netzwerkes erfolgt hierarchisch [IBM99b]. An oberster Stelle ist die Domäne des MQSeries WF-Netzwerkes dargestellt. Darunter befinden sich Systemgruppen welche sich in Systeme unterteilen. Die auf einer physischen Maschine installierten Systemkomponenten werden als Knoten bezeichnet.

Die Domäne stellt die ganze Organisation oder Teile der Organisation dar. Eine darunterliegende Systemgruppe kann z.B. ein geographischer Bereich sein. Ein System enthält die Client/Server-Komponenten, die zur Ausführung der Prozessmodelle benötigt werden.
Alle Einstellungen (z.B. Daten des Prüfprotokolls speichern), die auf der obersten Ebene getroffen werden, gelten gleichzeitig für die darunterliegenden Ebenen. Sollen in einer unteren Ebene andere Einstellungen wie in der Domäne gelten, können diese explizit angegeben werden. Diese Einstellungen gelten dann nur für diese und die darunterliegenden Ebenen. Das bedeutet, daß Einstellungen auf Systemgruppen-Ebene auch für die darunterliegenden Systeme gelten.

#### 2.1.2.3 Erstellen von Datenstrukturen

Datenstrukturen bilden den Inhalt der Ein- und Ausgabecontainer von Prozessmodellen, Aktivitäten und Blöcken ab. Jede Art von Daten, die in Ein- oder Ausgabecontainern und evtl. in Ende- oder Transitionsbedingungen vorkommt (siehe Abschnitt 2.1.3.2.2), muß in einer Datenstruktur spezifiziert sein. In der Baumstrukturansicht `implementations` ist unter `data structures` standardmäßig das Objekt `Default Data Structures` angelegt. Dieses Objekt kann nicht gelöscht, aber erweitert werden. Auf die Möglichkeiten für das Erweitern oder Erstellen von Datenstrukturen wird in Abschnitt 2.1.3.3 eingegangen.

#### 2.1.2.4 Registrierung der Anwendungsprogramme

Damit Aktivitäten die ihnen zugewiesenen Programme zur Laufzeit starten können, müssen diese in der Buildtime registriert werden. Als Mindestspezifikation wird der Name des Programmes erwartet. Wird der Name des Programmes ohne Aufrufpfad angegeben, muß sich das Programm im `bin`-Verzeichnis des MQSeries WF-Server befinden. Zusätzlich können die Systemumgebung, die Datenstrukturen für Ein- und Ausgabecontainer sowie jegliche dem Programm übermittelten Parameter (z.B. Aufrufoptionen) festgelegt werden. Durch die Einstellung der Systemumgebung kann sichergestellt werden, daß Benutzer verschiedener Betriebssysteme ein für Ihr Betriebssystem lauffähiges Programm rufen.

#### 2.1.2.5 Erstellen von Prozessmodelldiagrammen

Um ein neues Prozessmodelldiagramm zu erstellen, muß zuerst die Diagrammansicht eines Prozessmodells aufgerufen werden (s.o.). Mit dem Mauszeiger kann dann ein Objekt (z.B. ein Aktivitätenknoten oder ein Konnektor) ausgewählt und im Arbeitsbereich platziert werden. Konnektoren können nicht beliebig im Arbeitsbereich erstellt werden. Um einen Konnektor hinzuzufügen, muß dieser zuerst ausgewählt und anschließend das Start- und danach das Zielobjekt angeklickt werden. Existierende Konnektoren können nicht auf ein neues Start- oder Endobjekt "umgebogen" werden. Nicht erlaubte Verbindungen mittels Konnektoren werden mittels akustischem Signal (z.B. Kontrollkonnektor zw. Startknoten (Quelle) und erster Aktivität) oder Fehlermeldung (z.B. bei Zyklen) dem Benutzer erkennbar gemacht und unterbunden. Um die Übersichtlichkeit bei großen Prozessmodellen zu erhalten, kann der Verlauf von Konnektoren durch den Prozessmodellierer beeinflußt werden. Falls ganze Aktivitäten im Arbeitsbereich verschoben werden, zieht die Buildtime automatisch alle beteiligten Konnektoren mit.
Negativ fällt auf, daß bei großen Prozessmodelldiagrammen dem Modellierer keine verkleinerte Darstellung zur Orientierung angeboten wird. Um einen Überblick zu gewinnen, muß der Benutzer mit den Möglichkeiten zur Verkleinerung und Vergrößerung des Diagrammes auskommen.

#### 2.1.2.6 Verifizieren der Daten

MQSeries WF bietet dem Benutzer die Möglichkeit eine Vielzahl von Abhängigkeiten und Einstellungen der Bestandteile eines Prozess-Metamodells verifizieren zu lassen. Dazu existiert im

Menüpunkt `Tools` die Option `verify`. Bei der Verifizierung eines Prozessmodells werden folgende Überprüfungen durchgeführt [IBM99c]:

- für ein Prozessmodell (z.B. darf das Prozessmodelldiagramm nicht leer sein)
- für ein Prozessmodell und seine Aktivitäten (z.B. müssen die input-/output-Datencontainer existieren)
- für alle Aktivitäten (z.B. dürfen pro Aktivität nicht mehr als 254 eingehende Kontrollkonnektoren existieren)
- für Programm- und Prozessaktivitäten (z.B. müssen die Eigenschaften eines Programms für mindestens eine Plattform (z.B. Windows NT) festgelegt werden)
- für Prozessaktivitäten (z.B. darf eine Startaktivität nicht auf das Prozessmodell verweisen, zu dem sie selbst gehört)
- für Blockaktivitäten (z.B. dürfen nicht mehr als 32766 Blockaktivitäten pro Prozessmodell existieren)
- für Programaktivitäten (z.B. müssen hinterlegte Programme existieren)
- für Kontrollkonnektoren (z.B. muß die Schaltbedingung ein gültiger Boolean-Ausdruck sein)
- für Datenkonnektoren (z.B. muß die Startaktivität eines Datenkonnektors mittels eines Kontrollpfades (Reihe von Kontrollkonnektoren) die Zielaktivität des Datenkonnektors erreichen können)
- für Datenstrukturen (z.B. dürfen Sie keine Schlingen (A enthält B und B enthält A) enthalten)

#### 2.1.2.7 Speicherung der Daten

Die Buildtime-Komponente von MQSeries WF verfügt über eine eigene, vom Laufzeitsystem getrennte, Datenbank. Diese Art der Datenspeicherung wird als "*disconnected mode*" [LeRo00] bezeichnet und besitzt folgende Vorteile:

- Die Buildtime-Datenbank kann eingesetzt werden, ohne daß die Leistung aktiver Prozesse beeinträchtigt wird.
- Die Datenbanktabellen können getrennt opimiert werden: Die Buildtime-Datenbank für die Modellerstellung und die Runtime-Datenbank für die Ausführung vieler Transaktionen.

Prozessmodelle in der Buildtime-Datenbank können noch unvollständig oder mit Fehlern behaftet sein. Ein fertiges Prozessmodell kann in ein FDL-File (FDL="Flowmark Definition Language") exportiert und in die Runtime-Datenbank importiert (siehe Kapitel 3) werden. Ebenso kann ein in einer FDL-Datei existierendes Prozess-, Organisations-, oder Datenmodell mittels Import in die Buildtime übernommen werden. Die Import- und Exportfunktion sind in der Menüleiste unter `Buildtime` zu finden.

**Unterstützte Datenbanken**
Zugriffe auf die Buildtime-Datenbank erfolgen mit ODBC-Unterstützung (*Open Database Connectivity*). Es können die folgenden Datenbanken verwendet werden:

- DB2 Universal Database

- Microsoft Jet Datenbankmaschine

**Unterstützung des Modellieres**
Wenn in der Buildtime-Datenbank Änderungen an einem Prozessmodell vorgenommen werden, müssen entsprechend nur diese in die Runtime-Datenbank eingespielt werden. Dabei wird der Modellierer von der Buildtime unterstützt. Ein Objekt in der Buildtime kann den Status `standard`, `neu`, `verändert` oder `wird gelöscht` haben. Der Status wird neben dem Objekt durch eine kleine Grafik angezeigt. Wurde ein Prozessmodell neu angelegt, haben alle Objekte den Status `neu`. Nach dem Einspielen in die Runtime-Datenbank kann der Status dieser Objekte auf `standard` gesetzt werden (`Buildtime- Object status-> reset all to default`). Alle zukünftigen Änderungen oder Erweiterungen des Prozessmodells werden sichtbar, indem die Buildtime diese Objekte mit dem entsprechenden Status versieht.

## 2.1.3 Metamodelle

Eine Buildtime-Datenbank umfaßt in MQSeries WF ein Organisationsmodell, ein Datenmodell sowie ein oder mehrere Prozessmodelle. Um die Möglichkeiten der Prozessmodellierung besser betrachten zu können, werden die diesen Modellen zugrundeliegenden Metamodelle nun genauer betrachtet.

### 2.1.3.1 Organisationsmetamodell

Ein Organisationsmodell dient der Darstellung von Personen und Organisationseinheiten (incl. des hierarchischen Aufbaus) eines (oder mehrerer) Unternehmen, die an der Ausführung von Prozessen beteiligt sein sollen. Dem Benutzer werden drei Modellierungselemente angeboten:

1. `Person`: Ein Mitarbeiter eines Unternehmens, der in der MQSeries WF-Datenbank angelegt wird. Die Gesamtheit der im Organisationsmodell definierten Personen stellt die an Prozessen teilnehmende Belegschaft dar. Eine Person kann für verschiedene Aufgaben in MQSeries WF autorisiert werden.
   Ein Modellelement, speziell für die Beschreibung von Personen, sind Level. Mit ihnen ist es möglich, Personen besser voneinander abzugrenzen. Die Level 0 bis 9 sind vordefiniert, und es können nur ihr Name und ihre Beschreibung verändert werden. So könnte 9 “hohes Fachwissen“ und 0 “kein Fachwissen“ bedeuten. Diese Levels können bei der Definition von Bearbeiterzuordnungen für Aktivitäten verwendet werden.

2. `Rolle`: Eine Rolle stellt eine Funktion oder Fähigkeit dar, die eine Person oder eine Gruppe von Personen gemeinsam erfüllt. Wird eine Aktivität einer Rolle zugewiesen, erhalten zur Laufzeit alle Mitglieder dieser Rolle diese Aktivität in ihrer Arbeitsliste. Jeder von ihnen ist berechtigt, diese Aktivität auszuführen. Für jede Rolle kann ein Koordinator festgelegt werden. Damit können Teams oder kleine Abteilungen abgebildet werden. Rollen sind ein Kriterium, anhand dessen dynamische Bearbeiterzuordnungen (siehe Abschnitt 2.1.3.2.4) erfolgen können. Es sind drei Rollen vordefiniert, die nicht geändert werden können. Dies sind die Rollen `System Administrator`, `Manager` und `Koordinator`. Die Rolle `System Administrator` verfügt über alle Rechte in MQSeries WF und muß immer exakt einer Person zugewiesen sein.

3. `Organisation`: Durch diese administrative Einheit wird die Struktur eines Unternehmens beschrieben. Mit Organisationen können Hierarchien abgebildet werden. Dabei kann eine Organisation nur eine übergeordnete Organisation, aber beliebig viele untergeordnete Organisationen haben. Eine Organisation muß eine Person als Manager zugewiesen haben. Eine Person kann immer nur einer Organisation angehören.

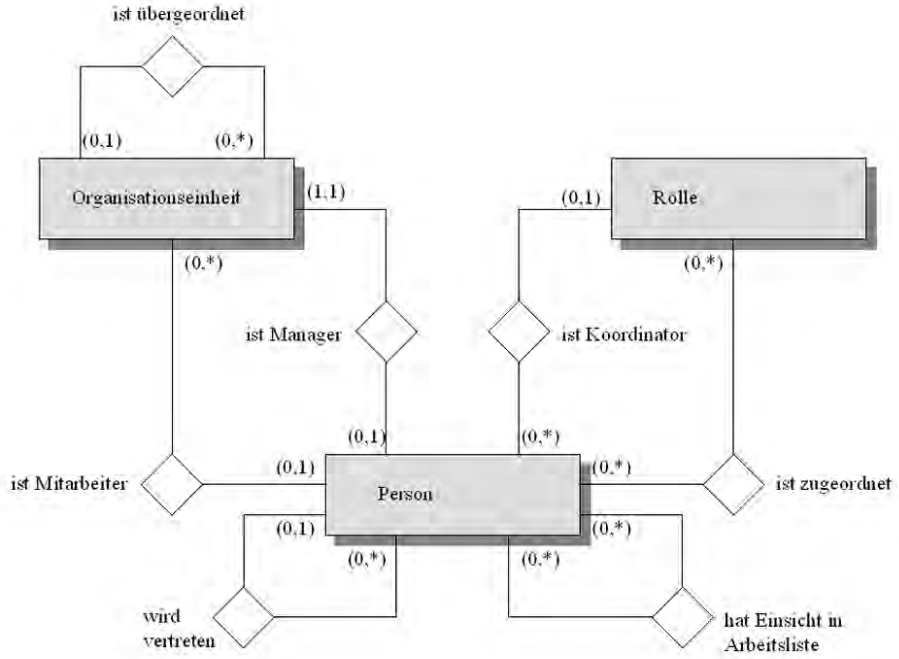



Abbildung 2.5: Die Organisationsstruktur

Das Zusammenspiel der drei Elemente wird in Abbildung 2.5 verdeutlicht.
Im folgenden werden einige wichtige Eigenschaften innerhalb von Organisationsmodellen beschrieben.

1. Vertreterregelungen
   Einer Person kann genau eine andere Person als Vertretung zugewiesen werden. Vertreterregelungen können direkt bei der Definition einer Person erfolgen oder zu jedem beliebigen späteren Zeitpunkt im MQSeries WF-Runtime-Client. Die als Vertreter gekennzeichnete Person erhält im Vertretungsfall alle neuen Arbeitslisteneinträge, ab dem Zeitpunkt, in dem die zu vertretende Person im System als abwesend gekennzeichnet wird. Arbeitslisteneinträge, die die zu vertretende Person davor erhalten hat, bekommt die Vertretung allerdings nicht zugeteilt. Sie verbleiben in der Arbeitsliste der zu vertretenden Person. Um diese Problematik zu lösen, kann beliebig vielen Personen die Einsicht in eine fremde Arbeitsliste gewährt werden (z.B. einem Abteilungsleiter die Einsicht in die Arbeitslisten seiner Mitarbeiter). Haben diese Personen die nötigen Rechte, können Sie Arbeitslisteneinträge aus der fremden Arbeitsliste an andere Personen übertragen (siehe Rechtevergabe). Die Vertreterregelung gilt bis (1) die abwesende Person die Eigenschaft `abwesend` im Runtime-Client storniert oder (2) zum erneuten Anmelden der ehemals abwesenden Person an den Runtime-Client.
   Welche der beiden Varianten gilt, wird in den Eigenschaften einer Person im Punkt `General` festgelegt. Ist das Kästchen für `Automatically reset absent indicator when person starts working with` nicht markiert gilt Variante (1), ansonsten Variante (2). Das manuelle Zurücksetzen der Eigenschaft `abwesend` erscheint gefährlich. Der Benutzer sieht nach dem Starten nicht, ob er an- oder abwesend ist. Es erscheint keine Meldung oder graphische Veränderung, die ihn darauf aufmerksam machen würde. Erhält ein Benutzer eher selten Einträge in seiner Arbeitsliste, kann es deshalb lange dauern bis er den Fehler bemerkt.

2. Rechtevergabe
   Die Vergabe von Rechten erfolgt auf Benutzerebene. Dabei erhalten Rollen oder Organisationen keine besonderen Rechte. Bei den Arten von Rechten kann unterschieden werden

zwischen Rechten für die Buildtime- und Runtime-Komponente. Für die Buildtimekomponente können für folgende Aufgabenbereiche Rechte vergeben werden:

- Prozessdefinition (*Process definition*)
- Mitarbeiterdefinition (*Staff definition*)
- Rechtevergabe für Mitarbeiter (*Staff authorization definition*)
- Definition von Netzwerkeigenschaften (*Topology authorization*)

Folgende Rechte betreffen die Runtime-Komponente:

- Zugriff auf Prozess-Kategorien
- Zugriff auf Arbeitslisten
- Administration von Kategorien
- Verwaltungsprogramm (Serveradministration, Verwaltung Fehler-/Systemprotokoll) (*Operation administration*)

Erhält ein Benutzer ein *Workitem* in seiner Arbeitsliste hat er automatisch die nötigen Rechte um damit arbeiten zu können (z.B. starten, suspendieren, übertragen).

**Grenzen des Organisationsmodells von MQSeries WF**
Anhand von zwei Beispielen sollen Probleme des Organisationsmodells von MQSeries WF erläutert werden. Das erste Beispiel zeigt die Schwierigkeiten auf, wenn Mitarbeiterstrukturen größerer Unternehmen auf das Modell abgebildet werden sollen. Auf die unzureichenden Möglichkeiten zur Abbildung von Vertreterregelung wird im zweiten Beispiel eingegangen.

**Beispiel 2.1**: (*Probleme bei großen Unternehmen*)
Für dieses Beispiel gehen wir von der Mitarbeiterstruktur eines Krankenhauses aus. Das Krankenhaus verfügt typischerweise über mehrere Abteilungen und über hunderte bis tausende von Mitarbeitern. Die Abteilungen werden mittels Organisationen und die Ärzte, Schwestern usw. mittels Rollen abgebildet. Vertiefende Informationen finden sich in [Kon96].
Auch wenn Mitarbeiter denselben Beruf ausüben, können Sie sich in Bezug auf einzelne Fähigkeiten und Kompetenzen dennoch unterscheiden. Beispielsweise können speziell geschulte Mitarbeiter oder langjährige Mitarbeiter über spezielle Fähigkeiten oder Rechte verfügen. Für eine Schwester eines Krankenhauses denkbare Fähigkeiten:

- Giftschrank verwalten
- Spritze geben
- Blutdruck messen
- Belegungsplan ändern

Diese Rechte müssten in die Rolle Schwester als Eigenschaften eingearbeitet werden. Damit würde sichergestellt, daß eine Schwester nur Aufgaben erhält, die Sie erfüllen kann.
MQSeries WF bietet in diesem Zusammenhang die `Level` an. Die zehn fest vorgegebenen `Level` 0-9 gelten aber für das gesamte Organisationsmodell und nicht nur für eine Abteilung oder eine bestimmte Rolle. Diese stark begrenzte Modllierungsmöglichkeit bietet deshalb keine adäquate Lösung. Die einzigste in MQSeries WF realisierte Alternative besteht darin, entsprechende Fähigkeiten und Kompetenzen einer Schwester über mehrere Rollen abzubilden. Das bedeutet, es werden Rollen wie Schwester_Giftschrank oder Schwester_Spritze definiert. Doch damit ist nur ein Teilproblem gelöst. Aufgrund der in MQSeries WF eingeschränkten Möglichkeit zur Definition von

Bearbeiterformeln für Aktivitäten (z.B. keine konjunktive oder disjunktive Verknüpfung von Rollen) müssten auch für alle möglichen Kombinationen von Eigenschaften entsprechende Rollen (z.B. Schwester_Giftschrank_Spritze) definiert werden. In Unternehmen wie Krankenhäusern führt dies zu extrem komplexen und nicht mehr überschaubaren Rollendefinitonen.

**Beispiel 2.2**: (*Unzureichende Vertreterregelung*)
Durch die Möglichkeit, die in der Realität üblichen hierarchischen Beziehungen von Organisationseinheiten abbilden zu können, ist das Organisationsmodell für viele Fälle ausreichend. Allerdings wird von MQSeries WF keine adäquate Vertreterregelung angeboten.
Anhand folgendem Szenario soll dieser Sachverhalt verdeutlicht werden. Ein Unternehmen A besteht aus dem Hauptsitz A1 und zwei ausgegliederten Fachbereichen B1 und B2. Die beiden Fachbereiche sind A1 fachlich und disziplinarisch unterstellt. Somit ist der Leiter von A1 verantwortlich für A1, B1 und B2. Dies wird im linken Bereich von Abbildung 2.6 dargestellt. Was aber wenn dieser Leiter ausfällt? In der Praxis würden seine Aufgaben oftmals auf kompetente Mitarbeiter aller Bereiche aufgeteilt (siehe rechter Bereich in Abbildung 2.6). Dies könnte so aussehen, dass die Leiter von B1 und B2 die fachliche Leitung Ihrer Bereiche selbst übernehmen und die disziplinarische Leitung des Unternehmens einer qualifizierten Person aus A1 zufällt.

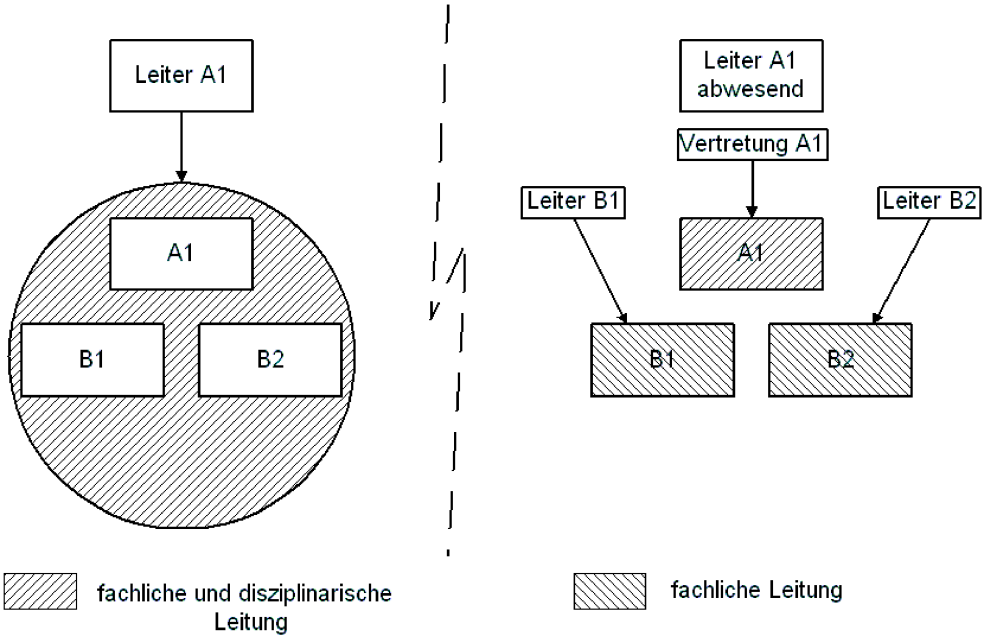



Abbildung 2.6: Probleme mit der Vertretungsregelung bei MQSeries WF

Dieses Szenario ist in MQSeries WF aufgrund der Beschränkung der Vertreterregelung auf exakt eine Person nicht möglich. Es besteht nur die Möglichkeit, dass eine Person die Vertretung übernimmt und fachfremde Arbeiten aus seiner Arbeitsliste an die anderen Bereichsleiter überträgt.

#### 2.1.3.2 Prozessmetamodell

Ein Prozessmodell ist eine graphische Darstellung eines Geschäftsprozesses, einschließlich der zugehörigen Geschäftsaktivitäten sowie ihrer Bearbeitungsreihenfolgen und -bedingungen. Dazu gehören auch die Festlegung der Bearbeiter- und Ressourcenzuordnungen für einzelne Prozessschritte (Aktivitäten). Ein Prozessmodell umfasst folgende Elemente:

- Aktivitäten
- Bearbeitungsreihenfolgen und -bedingungen von Aktivitäten (Kontrollfluss)

- Transitionsbedingungen von Kontrollkonnektoren (beschreiben mögliche Übergänge zwischen Aktivitäten und die dazu benötigten Bedingungen während der Workflow-Ausführung) (Kontrollfluss)
- Menge der potentiellen Bearbeiter einzelner Prozessaktivitäten (Bearbeiterzuordnung)
- Workflowrelevante Daten bzw. Ein-/Ausgabedatencontainer für Aktivitäten und die Verknüpfung von Ausgabecontainern einer Aktivität mit Eingabecontainern nachfolgender Aktivitäten (Datenfluss)
- Programme, die für die Ausführung von elementaren Aktivitäten erforderlich sind (Aktivitätenprogramme)
- Die maximale Dauer von Aktivitäten und Prozessen. (Zeit bzw. Deadlines)

Abbildung 2.7 verdeutlicht die Zusammenhänge dieser Elemente und deren Eigenschaften (in Anlehnung an [WFMC99]) :

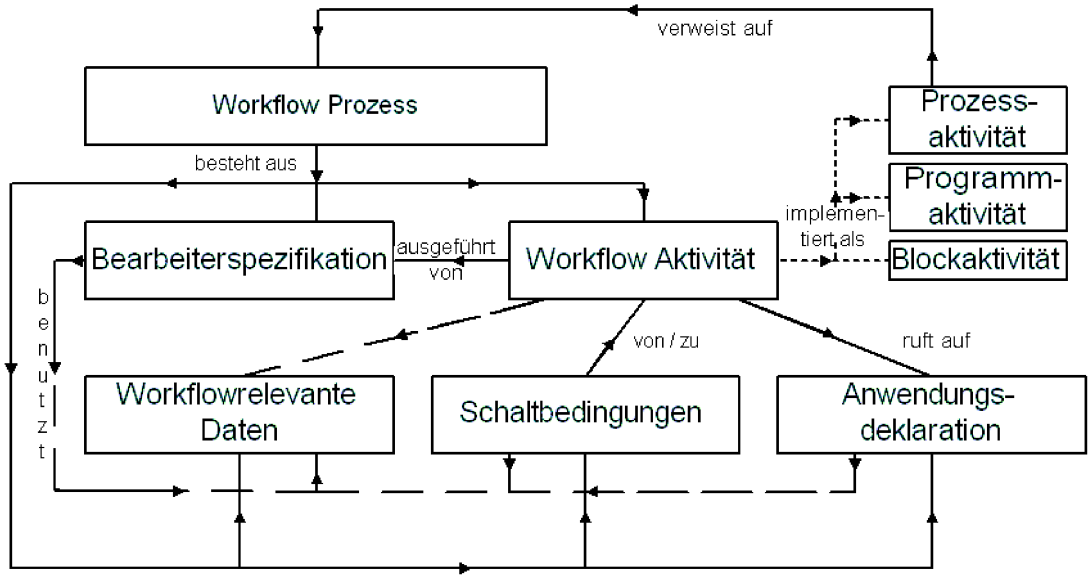



Abbildung 2.7: Prozess-Metamodell von MQSeris WF

#### 2.1.3.2.1 Aktivitäten

MQSeries WF unterscheidet zwischen Programm-, Block- und Prozess-Aktivitäten.

Eine `Programmaktivität` wird direkt mit einem in der Buildtime registrierten Programm verknüpft. Das Programm wird ausgeführt, wenn die Aktivität in der Runtime gestartet wird. Nach Beendigung des Programmes wird die Exit-Bedingung der Aktivität ausgewertet. Abhängig davon geht die Aktivität in den Zustand `fertig` oder `bereit` (d.h. erneutes Ausführen der Aktivität) über. Falls eine manuelle Beendigung der Aktivität gefordert wird, muß der Bearbeiter die Beendigung der Aktivität bestätigen.

Eine `Blockaktivität` klammert eine Menge von Aktivitäten. Diese werden zur Laufzeit solange wiederholt, bis die Exit-Bedingung des Blocks zutrifft. Die Modellierung der Aktivitäten findet in einem separaten Diagramm statt. Blockaktivitäten dienen zur Strukturierung von Prozessmodellen. Darüber hinaus ermöglichen sie die Abbildung von Do-Until-Schleifen (siehe [IBM99c]).

Eine `Prozessaktivität` steht für einen anderes Prozessmodell (Subprozess). Mit ihr wird in das aktuelle Prozessmodell ein bereits in der Buildtime existierendes Prozessmodell als Sub-Workflow eingebunden, d.h. Prozessmodelle können zur Definition von Subprozessen anderer Prozessmodelle wiederverwendet werden. Dieses Prozessmodell wird zur Laufzeit aufgerufen, wenn die Prozessaktivität startet. Nach ihrer Bearbeitung wird die Exit-Bedingung der Prozessaktivität ausgewertet. Abhängig davon wird die Prozessaktivität erneut gestartet oder aber sie ist beendet.
Mit Prozessaktivitäten sind Abbildungen rekursiver Aufrufbeziehungen möglich. Es existiert nur eine Einschränkung. Ist die Prozessaktvität gleichzeitig Startaktivität eines Prozessmodells, so darf sie nicht auf diesen Verweisen.

Durch die Angabe der Option `at least one incoming true` bzw. `all incoming true` kann für Aktivitäten festgelegt werden, ob für ihre Ausführung alle eingehenden Kontrollkonnektoren auf `True` geschaltet sein müssen oder nicht. Damit hier keine Deadlocks enstehen, verfügt MQSeries WF über eine `Death-Path-Elemination` (siehe Abschnitt 2.1.3.2.2).
Jede Aktivität kann mit einer Priorität versehen werden. Eine Priorität wird durch einen Wert vom Typ Long, im Bereich von 0..9, repräsentiert. Ist kein Wert oder ein falscher Wert gesetzt worden, so wird die Priorität standardmäßig auf 0 gesetzt. MQSeries WF deutet diesen Wert nicht während der Ausführung.

#### 2.1.3.2.2 Kontrollfluss

Der Kontrollfluss wird graphisch modelliert. Er stellt die Abfolge der Aktivitäten eines Geschäftsprozesses dar. Zur Modellierung stehen zwei gerichtete Pfeile zur Verfügung, die zusätzlich mit Tranistionsbedingungen verknüpft werden können.
`Kontrollflusskonnektor`: Erhält er keine Tranistionsbedingung, evaluiert er nach dem Ausführen der Aktivität, des Blockes oder des Subprozesses zu `True`. Ansonsten wird die assoziierte Transitionsbedingung ausgewertet und der Konnektor entsprechend dem Ergebniss zu `True` oder `False` evaluiert. Eine Transitionsbedingung kann aus einer Bedingung ($X \leq 0$) oder einer logischen Kombination mehrerer Bedingungen (z.B. $X \leq 0 \wedge Y \leq 5$) bestehen. Die Transitionsbedingung muß sich auf im Outputcontainer der Aktivität befindliche Datenfelder beziehen.
`Standard Kontrollflusskonnektor`: Er evaluiert zu `True` falls kein anderer, von der entsprechenden Aktivität ausgehender Kontrollflusskonnektor nach Auswertung seiner Transitionsbedingung zu `True` evaluiert. Es können mehrere Standardkontrollflusskonnektoren eine Aktivität verlassen. Diese evaluieren entweder alle zu `True` oder `False`.
Die Ablauflogik wird ausschließlich über die Bedingungen der Kontrollflusskonnektoren sowie über Exit-Bedingungen der Aktivitäten festgelegt. Dies wird als implizite Modellierung der Kontrollflusslogik bezeichnet [JaBöSc97]. Nachteilig ist bei dieser Modellierungsart die mangelhafte Anschaulichkeit des Modells. Daraus resultiert eine erschwerte Überprüfbarkeit der Kontrollflusslogik. Beide Konnektoren können nur Aktivitäten, Blöcke und Subprozesse untereinander verbinden. Die Möglichkeiten zur Modellierung von Kontrollflusskonstrukten wird in Tabelle 2.1 dargestellt.

Durch Verzweigungen (AND-Split) werden Parallelausführungen von Aktivitäten möglich. Da nicht sichergestellt werden kann, welche Aktivität als Erste (bzw. Zweite usw) startet, müssen zwischen solchen Aktivitäten Datenabhängigkeiten vermieden werden. Eine Datenabhängigkeit besteht, wenn beide Aktivitäten auf den selben Daten arbeiten. In diesen Fällen kann es zu Datenverlusten (*Lost Update*) kommen (siehe Kapitel 2.1.3.2.3). Zyklische Abhängigkeiten zwischen Aktivitäten sind nicht erlaubt, da andernfalls Deadlocks konstruierbar wären. Ein Deadlock ist eine Situation, in der der gesamte Prozess nicht beendet ist, keine Aktivität sich in Ausführung befindet und keine Aktivität mehr den Zustand `bereit` erreichen kann. Der Grund dafür liegt in der Art der Festlegung der Aktivierungsbedingungen für Aktivitäten.

| Sequenz | JA |
|---|---|
| Verzweigungen | |
| AND-Split | JA |
| OR-Split | JA |
| AND-Join | JA |
| OR-Join | JA |
| Schleife | |
| mittels Kontrollkonnektoren | Nein |
| mittels Exit-Bedingung | JA |
| Hierarchische Strukturierung | JA |

Tabelle 2.1: Kontrollflusskonstrukte (MQSeries WF)

Damit eine Aktivität aktiviert werden kann muß folgendes gelten:

- Notwendige Bedingung: Alle eingehenden Konnektoren der Aktivität sind mit `True` oder `False` bewertet
- Hinreichende Bedingung: Hier sind zwei Fälle zu unterscheiden:
  1. alle eingehenden Konnektoren müssen zu `True` bewertet worden sein (*all incoming true*)
  2. mindestens ein Konnektor muß zu `True` bewertet worden sein (*at least one incoming true*)

Würde eine Schleife über Kontrollkonnektoren modelliert (siehe Abbildung 2.8), könnte die notwendige Bedingung für die Aktiviterung einer Aktivität (im Beispiel die Aktivität D) niemals erfüllt werden.

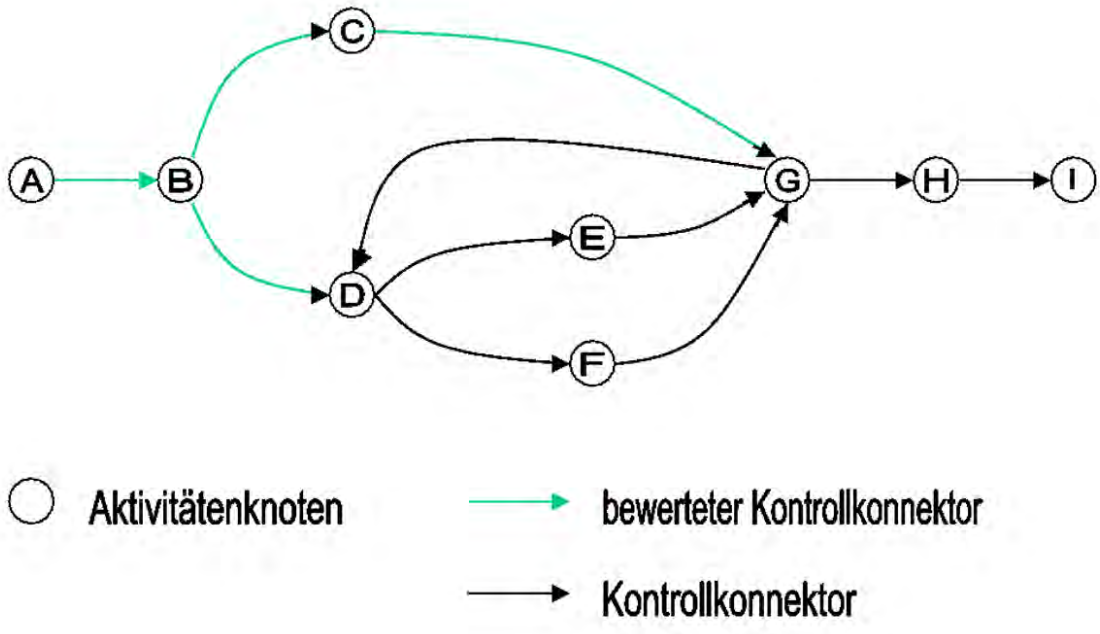



Abbildung 2.8: Deadlocks durch Schleifen

Durch das Verbot von Schleifen ist zugleich sichergestellt, daß der Kontrollfluss über mindestens eine Start- (Aktitität A) und mindestens eine Ende-Aktivität (Aktitität I) verfügt. Eine Aktivität wird als Start-Aktivität des Kontrollflussgraphen bezeichnet, wenn sie keine eingehenden Kontrollkonnektoren besitzt. Dementsprechend wird eine Aktivität als Ende-Aktivität des Kontrollflusses

bezeichnet, wenn Sie keine ausgehenden Kontrollkonnektoren besitzt. Die Ausführung einer Prozessinstanz beginnt mit der Aktivierung aller Start-Aktivitäten und endet nach der Verarbeitung aller Ende-Aktivitäten.
Eine Death-Path-Elimination findet bei beiden Kontrollflusskonnektoren statt. Evaluiert ein Kontrollflusskonnektor (Variante1) bzw. alle eingehenden Kontrollkonnektoren (Variante2) zu `False`, kann die Zielaktivität nicht mehr ausgeführt werden. Dadurch können andere Aktivitäten ebenfalls unerreichbar werden. Das System erkennt dieses Problem und markiert alle nicht mehr erreichbaren Konnektoren und Aktivitäten entsprechend. Durch die besondere Markierung weiß das System, daß es auf diesen Konnektor nicht mehr warten muß und wertet die Bedingung ohne diesen Konnektor aus. Dadurch werden Deadlocks vermieden. In Kapitel 3 wird die Reaktion der Runtime auf dieses Verhalten aufgezeigt.
Das folgende Szenario wird die Probleme mit der Death-Path-Elimination verdeutlichen. Verzweigt der Kontrollfluss nach einer Aktivität, werden die Bedingungen der Konnektoren ausgewertet. Für die Konnektoren deren Bedingung zu `False` evaluiert tritt die `Death-Path-Elimination` in Kraft. Sind die Bedingungen der Verzweigungen so gewählt, dass für alle möglichen Werte mindestens eine Bedingung zu true evaluiert, treten keine Probleme auf (siehe Abbildung 2.9). Gibt es Werte, für die alle Bedingugen zu `False` evaluieren (siehe Abbildung 2.10) kommt es zu einem unerwünschten Effekt. Die Aktivität ist beendet, alle Bedingungen der Kontrollflusskonnektoren evaluieren zu False. Die `Death-Path-Elimination` greift. Alle folgenden Aktivitäten werden als unerreichbar markiert und die ganze Prozessinstanz gilt als erfolgreich beendet (siehe Abbildung 2.11). Der Modellierer erhält keinen Hinweis auf solche (unerwünschten) Szenarien.

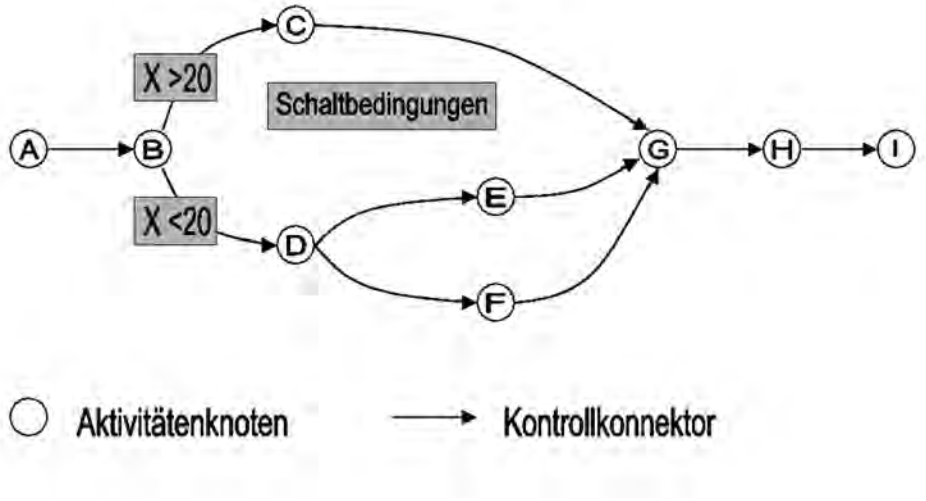



Abbildung 2.9: Graphische Darstellung der Ausgangslage

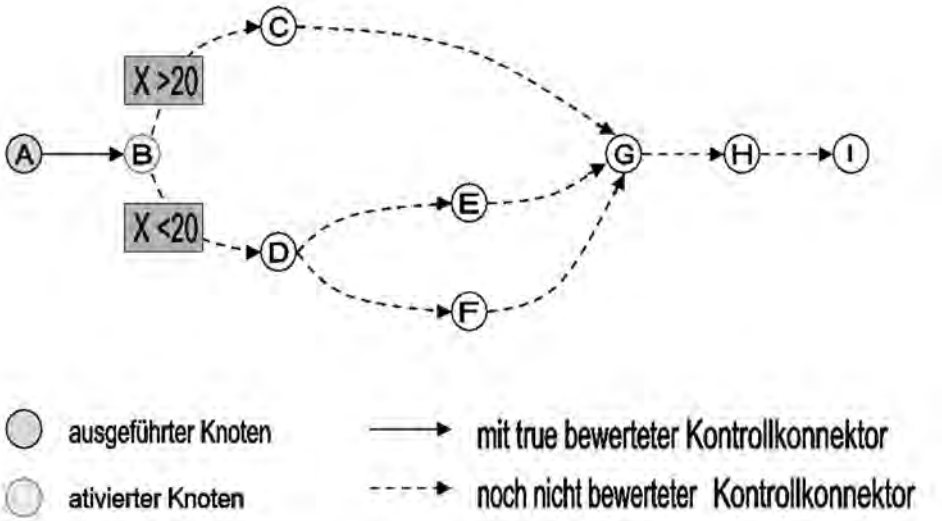



Abbildung 2.10: Problem bei der Deathpath-Elemination

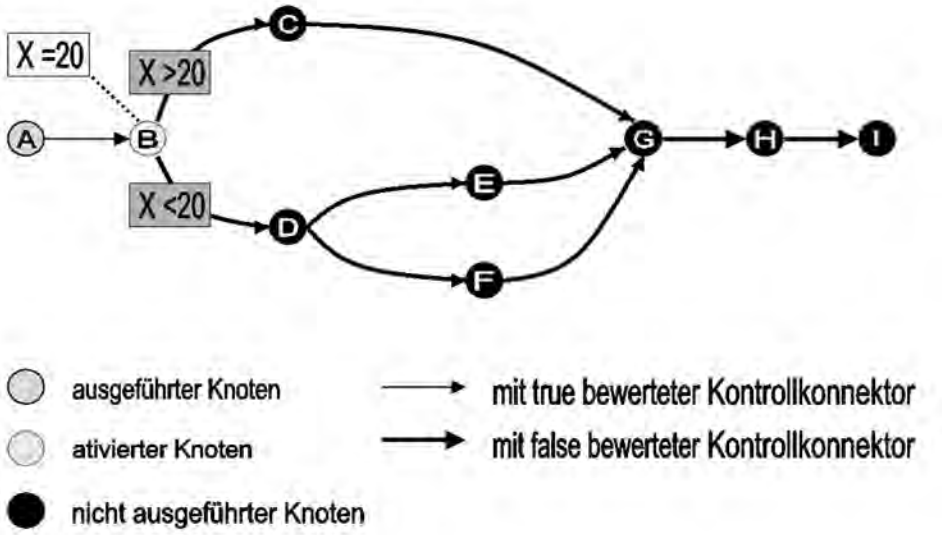



Abbildung 2.11: Ergebnis nach der Deathpath-Elemination

#### 2.1.3.2.3 Datenfluss

Datenflüsse werden in MQSeries WF graphisch mittels Datenkonnektoren, in Form von gestrichelten Pfeilen (Abbildung 2.3), dargestellt. Sie bilden den Verlauf der Daten zwischen den (Ein-/Ausgabe-) Datencontainern der Aktivitäten ab. Zur Datenübernahme aus bzw. Datenrückgabe an den Superprozess kann in einem Subprozess oder Block der Datenfluss zwischen Quelle und beliebig vielen Aktivitäten sowie zwischen beliebigen Aktivitäten des Subprozesses oder Blocks und Senke modelliert werden.
Die Verwendung von Datenflusskonnektoren zwischen zwei Aktivitäten ist nur möglich, wenn Aktivität B von Aktivität A aus über eine Folge von Kontrollkonnektoren erreicht werden kann.
Es werden drei verschiedene Arten von Datenkonnektoren angeboten [IBM99c]. Der Datenkonnektor, der Standard-Datenkonnektor und der Loop-Datenkonnektor. Standard- und Loop-Datenkonnektoren müssen an der selben Aktivität beginnen und enden. Dies ist in MQSeries WF die einzigste Form eines erlaubten Zyklus.

- `Datenkonnektoren`: Daten aus dem Output-Containern einer Aktivität können zu einem oder mehreren Input-Containern nachfolgender Aktivitäten gereicht werden. Haben Output- und Input-Container die selbe Datenstruktur und gibt es keinen weiteren Datenkonnektor zur Zielaktivität, so nimmt MQSeries WF das Mapping automatisch vor. Änderungen sind aber jederzeit möglich.

- `Standard-Datenkonnektoren`: Die Daten werden vom InputContainer einer Aktivität zu deren Output-Container geleitet.
  Input- und Outputcontainer einer Aktivität können über verschiedene Datenstrukturen verfügen. Dieser Datenkonnektor "mappt" die Datenstruktur des Inputcontainers oder Teile davon auf die Datenstruktur des Outputcontainers. Dieses Verfahren entlastet den Programmierer der hinterlegten Anwendung. Er muß sich nicht mehr um alle Daten kümmern.

- `Loop-Datenkonnektoren`: Die Daten werden vom Outputcontainer einer Aktivität zu deren Inputcontainer geleitet. Wird eine Aktivität nach Auswertung der Exit-Bedingung nochmals gestartet, kann mit diesem Datenkonnektor ihr Inputcontainer für den erneuten Start festgelegt werden. So können beispielsweise feste Startwerte für eine Anwendung sichergestellt werden (mittels *Default*-Werten).

Ein Mapping von Datencontainerfeldern ist nur bei identischen Datentypen möglich.
Auf einen Inputcontainer einer Aktivität können mehrere Outputcontainer verschiedener Aktivitäten abgebildet werden. Das Ergebnis ist nichtdeterministisch!
Mittels `Copy und Paste` können Teile eines Prozessmodelles ausgeschnitten und in ein anderes Prozessmodell eingefügt oder als neues Prozessmodell angelegt werden. Die Einstellungen der Konnektoren bleiben dabei erhalten. Einzelne Konnektoren oder Folgen von Konnektoren ohne Aktivitäten können mittels `Copy and Paste` nicht vervielfältigt werden. Ob die Einstellungen eines Datenkonnektors ausgeführt werden, richtet sich nach dem zugehörigen Kontrollkonnektor. Schaltet dieser auf `true` werden die Einstellungen des Datenkonnektors gültig. Die Datenkonnektoren zwischen Quelle und beliebigen Aktivitäten werden bei betreten des Subprozesses oder des Blocks gültig. Die Datenkonnektoren zwischen beliebigen Aktivitäten und Senke werden beim beenden der entsprechenden Aktivität gültig.

#### 2.1.3.2.4 Bearbeiterzuordnung

Einer Programmaktivität kann ein Bearbeiter statisch oder dynamisch zugewiesen werden. In der statischen Variante wird explizit eine existierende Person zugewiesen. Dazu muß ihr Name bekannt sein. In der dynamischen Bearbeiterzuordung wird erst zur Laufzeit entschieden, wem die Aktivität zur Bearbeitung angeboten werden soll. Dem liegen zwei Vorgehensweisen zugrunde:

- Die Bearbeiterzuordnung erfolgt anhand des bisherigen Verlaufs des Prozesses (*abhängige Bearbeiterzuordnung*).
  Als Bearbeiter kann
  - der Starter des Prozesses
  - der Manager des Starters des Prozesses
  - der Starter einer vorherigen Aktivität
  - der Manager des Starters einer vorherigen Aktivität
  - jede Person außer dem Starter einer vorherigen Aktivität (`Vier-Augen-Prinzip`)

  gewählt werden.

Während des Prozessverlaufes können auch Personen-, Rollen- und Organisationsnamen mittels Container übergeben werden. Dabei kann ein Wert aus der Datenstruktur des Eingabecontainers der Aktivität zur Bearbeiterzuordnung gewählt werden. Wird die Option `Staff from predefined members` gewählt, wird der Inhalt der Datenstruktur `_acitity_info` ausgewertet.
Bei der Arbeit mit Containern muß in jedem Fall das Mappen der Daten besonders beachtet werden. In Abbildung 2.12 wird das Arbeiten mit Containern graphisch veranschaulicht. In diesem Beispiel würde die Aktivität `D` jedem Mitglied der Rolle `Arzt` angeboten.

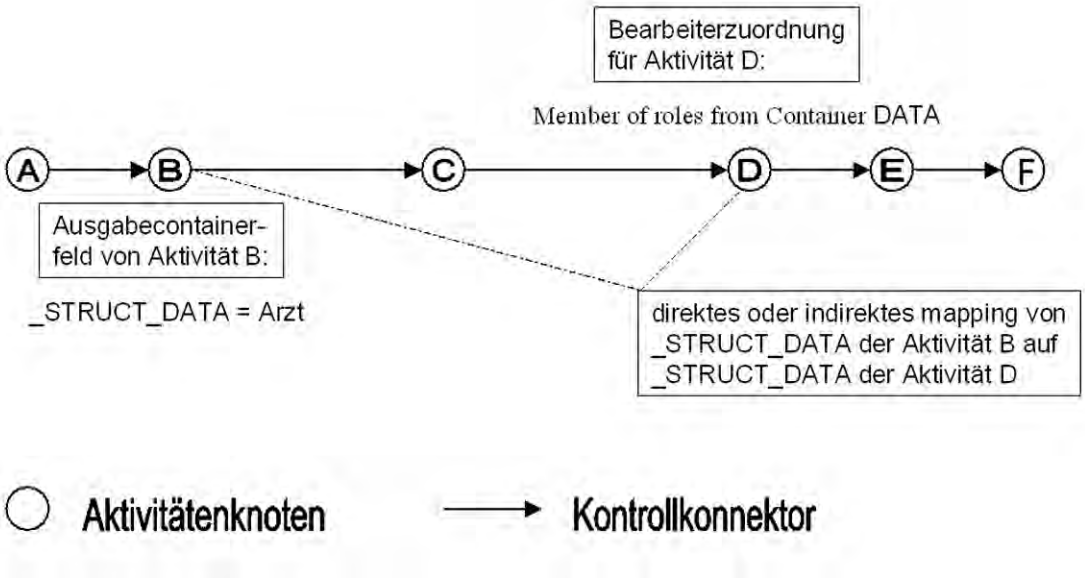



Abbildung 2.12: Bearbeiterzuordnung mittels Container

- Die Bearbeiterzuordnung erfolgt unter Bezugnahme auf Teile des Organisationsmodells (*unabhängige Bearbeiterzuordung*). Eine unabhängige Bearbeiterzuornung kann auf Organisationen oder Rollen Bezug nehmen. Kombinationen sind ebenso möglich, wie Einschränkungen auf Manager oder Koordinatoren. Für Organisationen kann festgelegt werden, ob Mitarbeiter untergeordneter Organisationen bei der dynamischen Bearbeiterzuordnung einbezogen werden sollen oder nicht.
  Zur Laufzeit bekommen alle Personen, die die angegebenen Bedingungen erfüllen, die Aktivität zur Ausführung angeboten.

Weitere Einschränkungen in Bezug auf die Menge potentieller Bearbeiter einer Aktivität sind durch die Festlegung von Levels möglich. Bei der dynamischen Bearbeiterzuordnung wird einmal, beim Instanzieren der Aktivität festgestellt, welche Personen die Bedingungen erfüllen. Daraus resultieren zwei Probleme [IBM00]:

- Zwischen einem Arbeitslisteneintrag (*Workitem*) und dessen Besitzer besteht eine 1-zu-1-Beziehung. Erfüllen nun 10 Personen die Kriterien zur Bearbeiterzuordnung, werden auch 10 Workitems angelegt. Dieser erhöhte Aufwand erfordert eine erhöhte Leistungsfähigkeit des Systemes.

- Werden im Organisationsmodell Personen zu einer Rolle oder zu einer Organisation nach Durchführung der Bearbeiterzuordnung für eine Aktivität hinzugefügt, werden für diese Personen keine entsprechenden Workitems mehr angelegt.

Dieses Problem wurde erkannt. In [IBM00] wird eine `Workaround-Lösung` angeboten. Dabei wird aber das zugrundeliegende Organisationsmodell aufgeweicht. Für einen Aufgabenbereich wird nicht mehr eine Rolle, sondern eine Person angelegt. Mitarbeiter dieses Aufgabenbereiches werden als Personen mit Zugriffsrechten auf diese spezielle Person modelliert. Dadurch wird in jedem Fall nur noch ein `Workitem` angelegt. Ebenfalls sind `Workitems` für Personen sichtbar, die erst nach der Durchführung der Bearbeiterzuordnung der zugehörigen Aktivität hinzugefügt wurden. Somit werden alle oben genannten Nachteile aufgehoben. Durch die Umgehung des eigentlichen Organisationsmodelles ergeben sich aber neue Probleme. So ist ohne das Organisationselement `Rolle` auch kein Koordinator eines Aufgabenbereiches zu benennen.
Allerdings sind mit diesen Möglichkeiten nicht alle denkbaren dynamischen Bearbeiterzuordnungen abbildbar. Das eine Aktivität von einer Person derselben Organisationseinheit wie der Bearbeiter einer Vorgänger-Aktivität ausgeführt wird, ist nicht modellierbar. Diese Problematik und ihre Konsequenz wird in Kapitel 2.4 genauer dargestellt.

#### 2.1.3.2.5 Zeit

Zur Überwachung von `Deadlines` können für Aktivitäten und Prozessmodelle einfache Zeitintervalle (maximale Bearbeitungsdauer) festgelegt werden. Diese Intervalle können durch Angabe von Jahr, Monat, Woche, Tag, Stunden, Minuten und Sekunden spezifiziert werden (Abbildung 2.13).

Abbildung 2.13: Einstellung eines 10sek. Zeitintervalles

Überschreitet die tatsächliche Ausführungsdauer einer Prozessinstanz die maximale Bearbeitungsdauer, erhält der Prozessadministrator eine Benachrichtigung in seiner Arbeitsliste. Weitere Einstellungen in Bezug auf Deadlines sind für Prozessmodelle nicht möglich.
Bei Aktivitäten sind die Einstellungsmöglichkeiten vielfältiger. Wer die erste Benachrichtigung nach Überschreitung der Aktivitätendauer erhält, kann entsprechend der normalen Bearbeiterzuordnung festgelegt werden. Falls diese Benachrichtigung nicht dem Prozessadministrator zugeht, kann ein weiteres Zeitintervall festgelegt werden. Wird die Aktivität innerhalb dieses zweiten Zeitintervalls ebenfalls nicht ausgeführt und beendet, erhält letzlich der Prozessadministrator eine Benachrichtigung [1].
Folgendes Beispiel soll die eingeschränkten Möglichkeiten, mit den durch die Angabe maximaler Bearbeitungsdauer beschriebenen Deadlines, verdeutlichen.

[1] Die Verwendung dieses Features konnte nicht nachvollzogen werden. Auf überschrittene Deadlines folgte bei mehreren MQSeries WF-Systemen keine der eingestellten Benachrichtigungen.

**Beispiel 2.3**: (*Eingeschränkte Möglichkeiten mit Deadlines*)
Eine Abteilung eines Unternehmen fertigt täglich sehr viele und relativ kleine Prozesse ab. Der Aufwand für eine Prozessbearbeitung ist absehbar und wird mit 15 bis 30 Minuten beziffert. Es wäre wünschenswert, wenn Mitarbeiter 30 Minuten vor Arbeitszeitende keine neuen Prozesse mehr starten könnten. Dies ist in MQSeries WF jedoch nicht möglich.
Die bekannte Prozessdauer kann als Richtlinie für die maximale Dauer des Prozesses, als Intervalldauer, verwendet werden. Eine fixierte Uhrzeit ist als Deadline nicht möglich.

**Gültigkeit von Prozessmodellen**
Für ein Prozessmodell kann festgelegt werden, ab welchem Datum und welcher Uhrzeit dieses gültig ist (siehe Abbildung 2.14). Diese Einstellung ermöglicht es dem Prozessmodellierer sein Prozessmodell zu einem beliebigen Zeitpunkt in die Runtime-Komponente von MQSeries WF exportieren zu können. Das neue Prozessmodell wird für alle Personen erst zum angegebenen Zeitpunkt gültig [2]

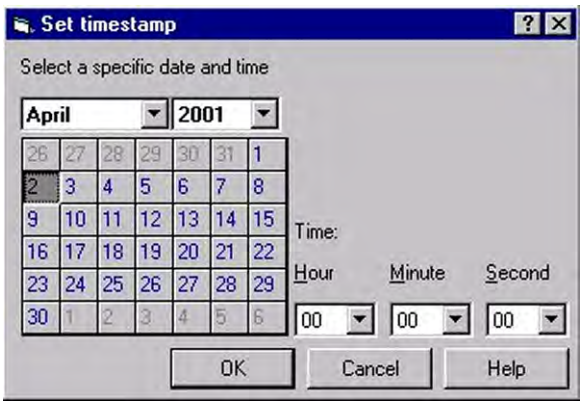



Abbildung 2.14: Gültigkeit von Prozessen

**Aufbewahrung beendeter Prozessinstanzen und Aktivitäten**
Für jedes Prozessmodell und die darin enthaltenen Aktivitäten kann festgelegt werden, wie lange (siehe Abb. 2.13) diese nach ihrer Ausführung physisch gespeichert bleiben.
Die Einstellungen für das Prozessmodell bzw. die Aktivitäten beziehen sich in der Runtime auf die Prozessinstanzen bzw. Arbeitslisteneinträge (*Workitems*).

#### 2.1.3.3 Datenmetamodell

Datenstrukturen beschreiben den Inhalt der Ein-/Ausgabe-Datencontainer der Prozesse, Aktivitäten und Blöcke. Alle Daten, die als Ein-/Ausgabe verwendet werden oder die in einer End-/Schaltbedingung referenziert werden, müssen in einer Datenstruktur festgelegt und definiert werden. Zum Aufbau dieser Datenstrukturen werden die elementaren Datentypen String (variabler Länge), Binary (variabler Länge), Gleitkommazahl (float) und long sowie bereits definierte Datenstrukturen verwendet.
Alle Datentypen und bestehende Datenstrukturen können als einzelnes Feld (*Member*) einer Datenstruktur oder aber in Form eines Array angelegt werden. Die Elemente eines Arrays sind immer vom selben Basistyp bzw. von der selben Datenstruktur. Die maximale Anzahl der Mitglieder einer Datenstruktur beträgt 512 [IBM99e]. Ein Feld, das ein Array repräsentiert, wird dabei mit der Anzahl seiner Elemente gerechnet. Die Buildtime-Komponente verhindert das Anlegen von

[2]Die Verwendung dieses Features konnte nicht nachvollzogen werden. Trotz mehrmaligen Versuchs auf zwei verschiedenen MQSeries WF-Systemen waren die eingespielten Prozessmodelle (mit Gültigkeitsdatum in der Zukunft) stets sofort verfügbar.

Arrays mit mehr als 512 Elemente. Das Anlegen einer Datenstruktur mit mehreren Arrays der Größe 511 wird nicht unterbunden.
Der Modellierer kann keine eigenen Datentypen kreieren.
Die neu geschaffenen Datenstrukturen sind wiederverwendbar, da sie ohne spezielle Bindung an eine Aktivität oder eines Prozessmodells in der Buildtime erstellt werden. Dadurch spart der Modellierer Zeit und es resultiert eine bessere Übersichtlichkeit der Buildtime. Häufig verwendete Datenstrukturen (wie z.B. Personendaten) können in beliebig vielen Prozessmodellen verwendet werden. Der Export und Import von Datenstrukturen ist möglich.
Ob auf Daten nur lesend oder schreibend zugegriffen werden darf, kann nicht festgelegt werden. Dies wird dem Programmierer der Anwendungsprogramme überlassen.

### 2.1.4 WPDL

Nachdem die zuvor beschriebenen Bestandteile eines Workflow-Modells graphisch und textuell erstellt wurden, müssen diese dem WFMS zugänglich gemacht werden. Dies ist nur durch eine dem WFMS verständlichen Sprache möglich. Die von MQSeries WF verwendete Sprache heißt `FDL` (*Flowmark Definition Language*).
Diese Sprache spiegelt die zugrunde liegende Struktur der Metamodelle (siehe 2.1.3) wieder. Jedes Objekt, wie z.B. eine Person oder ein Kontrollkonnektor, wird durch ein Schlüsselwort identifiziert und mit einem eindeutigen Namen versehen. Die verschiedenen Eigenschaften der Objekte werden durch Parameter, oder optional durch Schlüsselwörter, beschrieben. Objekte, Parameter und deren Werte werden nur durch Zeichenketten dargestellt. Dementsprechend können alle gesetzten Definitionen mittels eines einfachen Zeichen-Editors auf jeder beliebigen Plattform abgeändert werden. Theoretisch können Definitionen, z.B. von Organisationsstrukturen, direkt in einem Zeichen-Editor erstellt und in die Runtime eingespielt werden. Aufgrund der Komplexität eines Workflow-Modells ist diese Möglichkeit aber nur von theoretischer Natur.
Erzeugt wird eine FDL-Datei mittels Aufruf der Export-Funktion im Menüpunkt `Buildtime`.
Eine vollständige und dokumentierte FDL-Datei, welche einen Kreditantrag wiederspiegelt, findet sich im Anhang C.2.

## 2.2 Staffware

Die für MQSeries WF beschriebenen Funktionalitäten und Fähigkeiten einer Buildtime-Komponente werden in Staffware durch mehrere Komponenten zur Verfügung gestellt. Das Prozess- und Datenmodell wird im `Graphical-Workflow-Designer` (GWD) und `Graphical-Form-Designer` (GFD) erstellt. Das Organisationsmodell, administrative Daten und systeminterne Daten werden über ein spezielles Administrations-Tool erstellt und gepflegt. Das Importieren und Exportieren von Prozessmodellen mittels einer WPDL (XFR-Dateien) wird durch ein DOS-Programm (SWUTIL.exe) bewerkstelligt. Abbildung 2.15 verdeutlicht diese Gliederung.
Im folgenden werden die Konzepte und Fähigkeiten der verschiedenen Staffware-Komponenten beschrieben. Von Bedeutung sind insbesondere die verschiedenen Metamodelle, deren Mächtigkeit und die Verständlichkeit bzw. Strukturiertheit der graphischen Oberflächen.

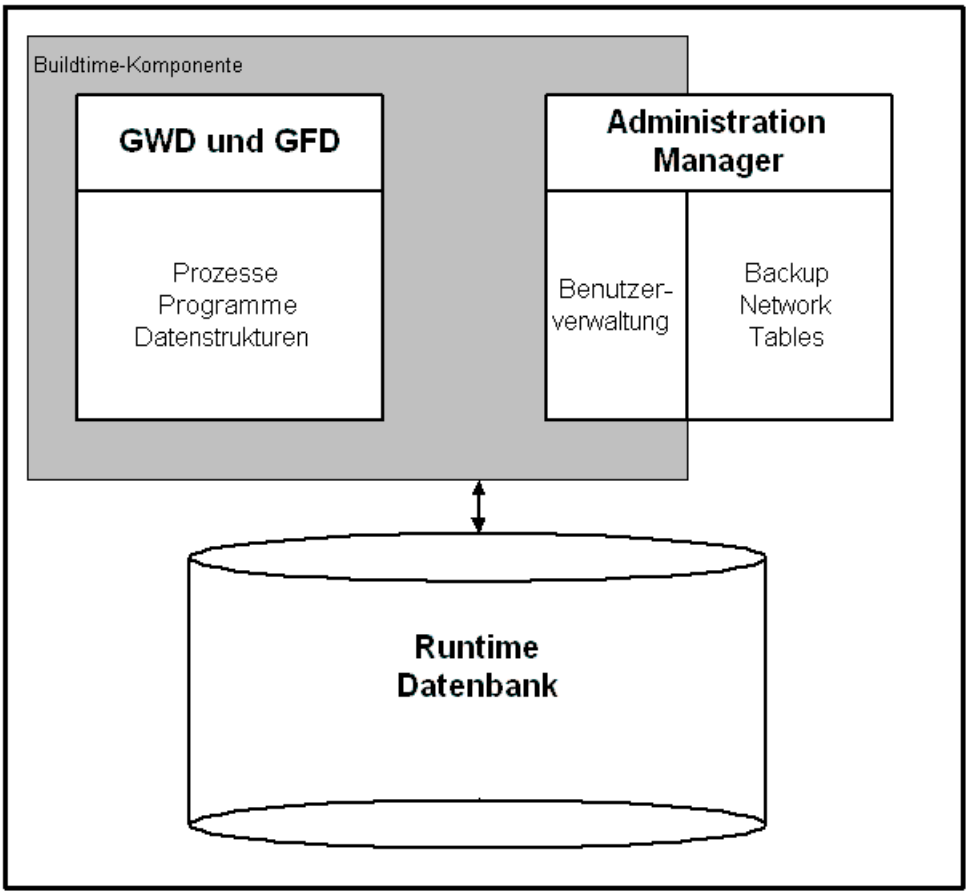



Abbildung 2.15: Der Aufbau der Buildtime-Komponente

### 2.2.1 Editoren

Nach der Installation existiert im Startmenü unter `Programme` ein neuer Eintrag namens `Staffware`. Er beinhaltet die Punkte `Staffware Client`, `Administration Tool` und `Staffware Utility Server`. Die Komponenten GWD und GFD sind nicht direkt zu starten, sondern nur aus dem `Staffware Client` heraus. Durch Auswahl von `Administration Tool` wird das Administrations-Tool gestartet. Dieses beinhaltet die Mitarbeiterverwaltung.
Für alle Staffware-Komponenten werden die Standard-Benutzer (swpro, swadmin, swuser) mit den während der Installation ausgewählten Passwörtern als Login-Kennung benötigt. Nach der Auswahl des `Staffware Client` im Window-Menü und dem folgenden Login erscheint die Oberfläche des Standard-Clienten (2.16).

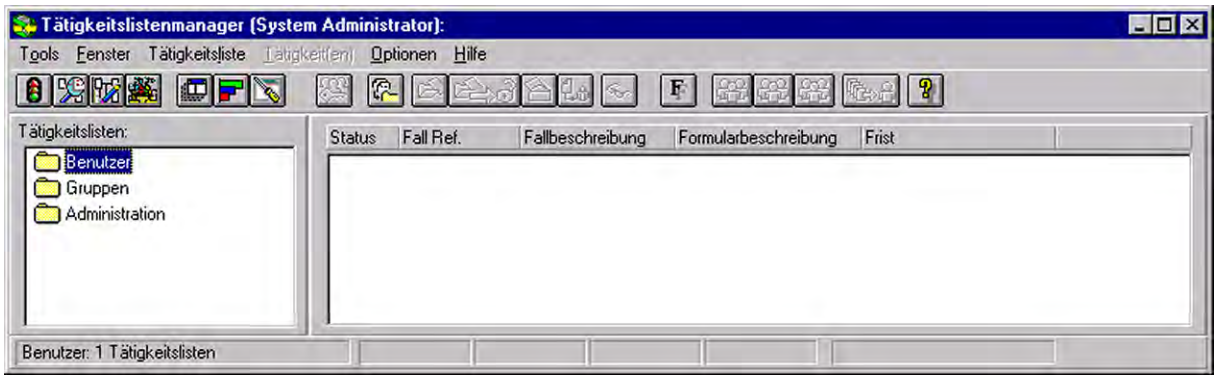



Abbildung 2.16: Staffware Client

Der Standard-Client wird in Kapitel 3 genauer besprochen. Durch Aufruf von `GWD`, in der Menüleiste unter `Tools`, wird der GWD gestartet. Allerdings können nur Benutzer mit entsprechenden Rechten (siehe Abschnitt 2.2.3.1) den GWD aufrufen. Ansonsten steht diese Option nicht zur Verfügung.

#### 2.2.1.1 GWD

Der GWD (Abbildung 2.17) besteht aus einer Menü-, Status- und Werkzeugleiste sowie dem Arbeitsbereich. In der Statusleiste am untern Rand des Fensters wird die aktuelle Auswahl eines Tools aus der Menü- oder Werkzeugleiste und die gültige Markierung eines Objektes im Arbeitsbereich angezeigt. Der Arbeitsbereich ist zu Beginn bis auf das Symbol für den Prozessstartpunkt (eine Ampel mit Fragezeichen) leer.
Die Elemente der Werkzeugleiste entsprechen den Auswahlmöglichkeiten des Punktes `Tool` der Menüleiste. Sie stellt damit einen direkten Zugriff auf die Modellierungselemente zur Verfügung.
In Abbildung 2.18 beinhaltet der Arbeitsbereich des GWD ein Prozessmodelldiagramm. Die graphische Darstellung des dargestellten Prozessmodelldiagramms kann durch den Benutzer angepasst werden. Dies beinhaltet die freie Wahl der Darstellung der Aktivitäten (*Icons*) und die farbliche Darstellung der Verbindungslinien zwischen ihnen.

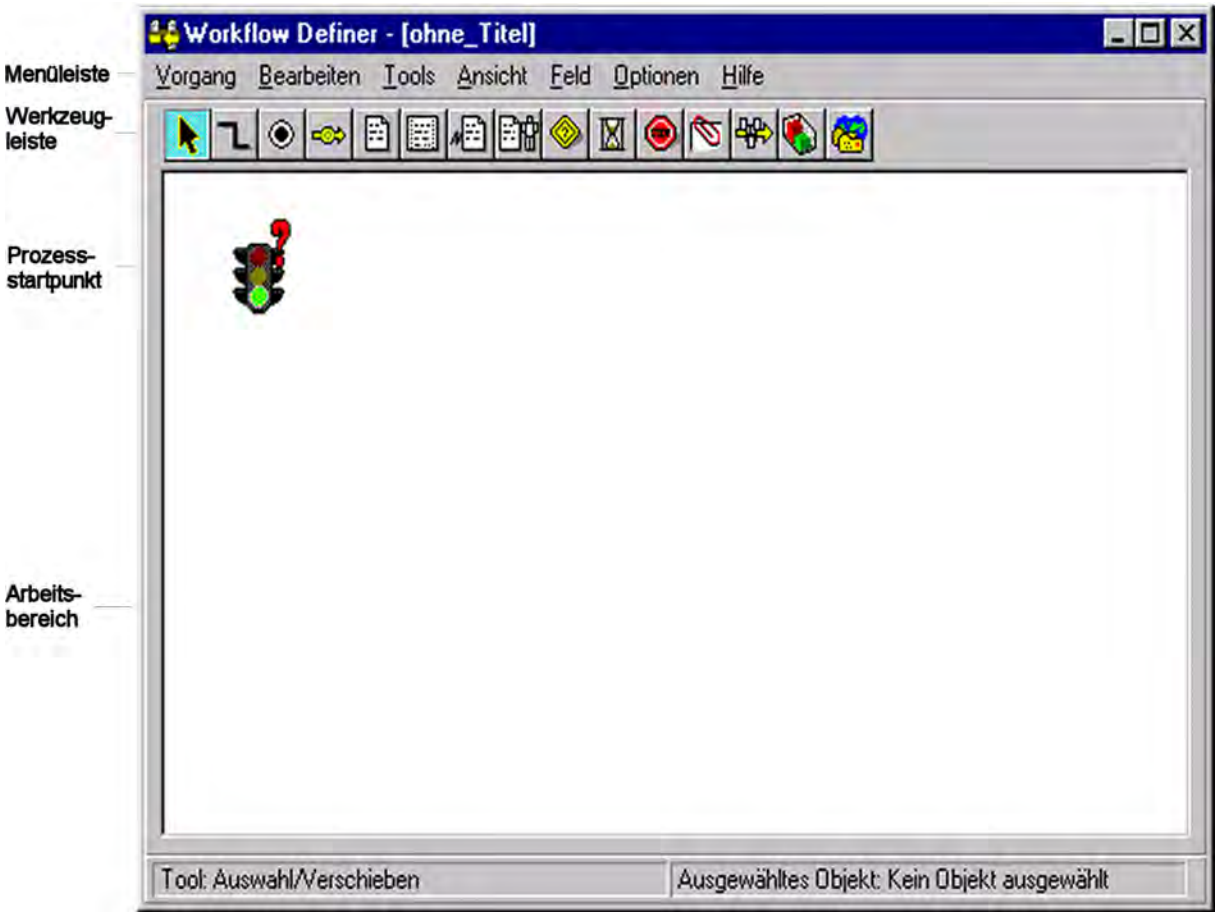



Abbildung 2.17: Der Graphical-Workflow-Designer

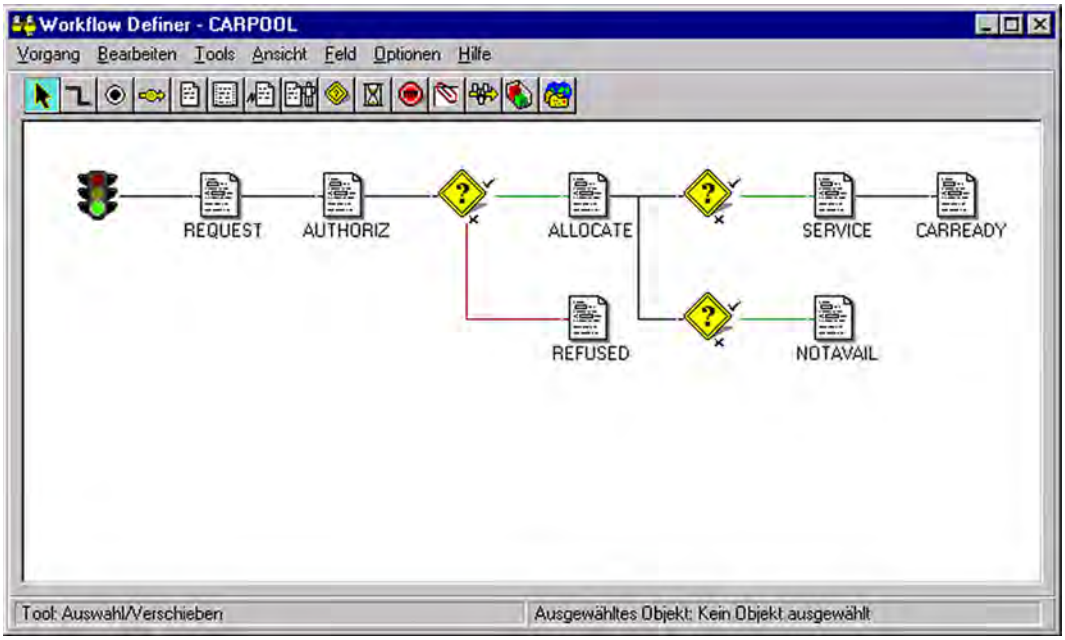



Abbildung 2.18: Der Graphical-Workflow-Designer mit Prozessmodelldiagramm

#### 2.2.1.2 GFD

Der GFD dient der Erstellung von elektronischen Formularen. Ein Formular ist eine Maske (graphische Oberfläche, siehe Abbildung 2.19) die dem Benutzer bei der Ausführung einer Aktivität angezeigt wird. Es stellt das Benutzerinterface einer Aktivität dar. Optional kann die Software `Staffware for VBA` (*Visual Basic for Applications*) geladen werden. Damit können Formulare in einem Visual Basic Projekt erstellt werden (siehe [SW99], [SW01a] und Kapitel 5).

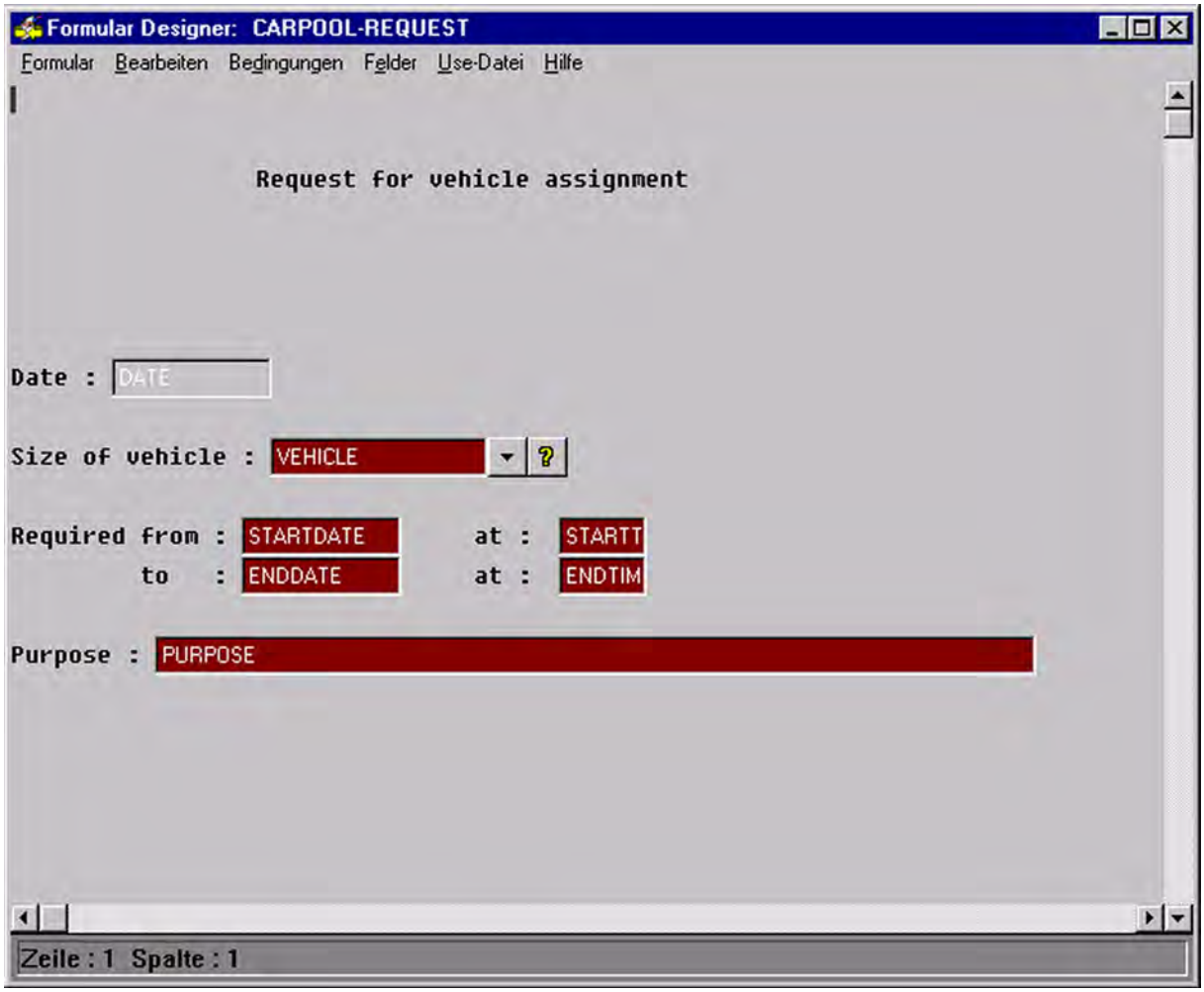



Abbildung 2.19: Der Graphical-Formular-Designer mit Feldern

Mit einem Formular können prozessrelevante Daten dargestellt und geändert werden (Abbildung 2.19). Dazu müssen

- Daten-Felder definiert und
- Felder in das Formular eingefügt werden

Ein Formular kann aber auch nur einen Text, z.B. als Nachricht, weitergeben.
Staffware-Felder sind nach ihrer Definition für alle Formulare des zugehörigen Prozessmodells verfügbar. Zusätzlich können Felder inklusive ihrer Defninierung, mittels `Copy and Paste`, von einem Prozessmodell zu einem anderen kopiert werden.

Jedes Staffware-Feld kann zusätzlich noch mit einem Staffware-Kommando (*Staffware Commands*) verknüpft werden. Ein Staffware-Kommando kann beispielsweise ein externes Programm aufrufen, ein Staffware-Skript aufrufen oder für ein Formularfeld einen neuen Wert berechnen. Am

häufigsten werden diese Verknüpfungen für Datenbankanfragen oder Datenübermittlugen genutzt [SW00b]. Alle Möglichkeiten mit Staffware-Kommandos, welche aus Staffware-Ausdrücken bestehen (*Staffware Expressions*), werden in [SW00e] dargestellt.

#### 2.2.1.3 Benutzerverwaltung

Für ein erfolgreiches Login in das `Administration Tool` sind für den Benutzer Administrationsrechte erforderlich. Die drei während der Installation angelegten Benutzer verfügen über diese Rechte. Im Administration-Tool, welches in Kapitel 5 genauer besprochen wird, kann der Benutzer den Punkt `User Manager` aufrufen. Dadurch öffnet sich ein neues Fenster (Abbildung 2.20).
Über Registerkarten können die Darstellungen für `Personen`, `Gruppen`, `Personen und Gruppen`, `Rollen` und `Eigenschaften` ausgewählt werden. Die Werkzeug- und Menüleiste bietet Funktionen entsprechend der ausgewählten Registerkarte.
Vorgenommene Veränderungen müssen dem System nach Beendigung des `User Manager` durch Aufruf des Punktes `Move SysInfo` im `Administration Tool` bekannt gemacht werden.

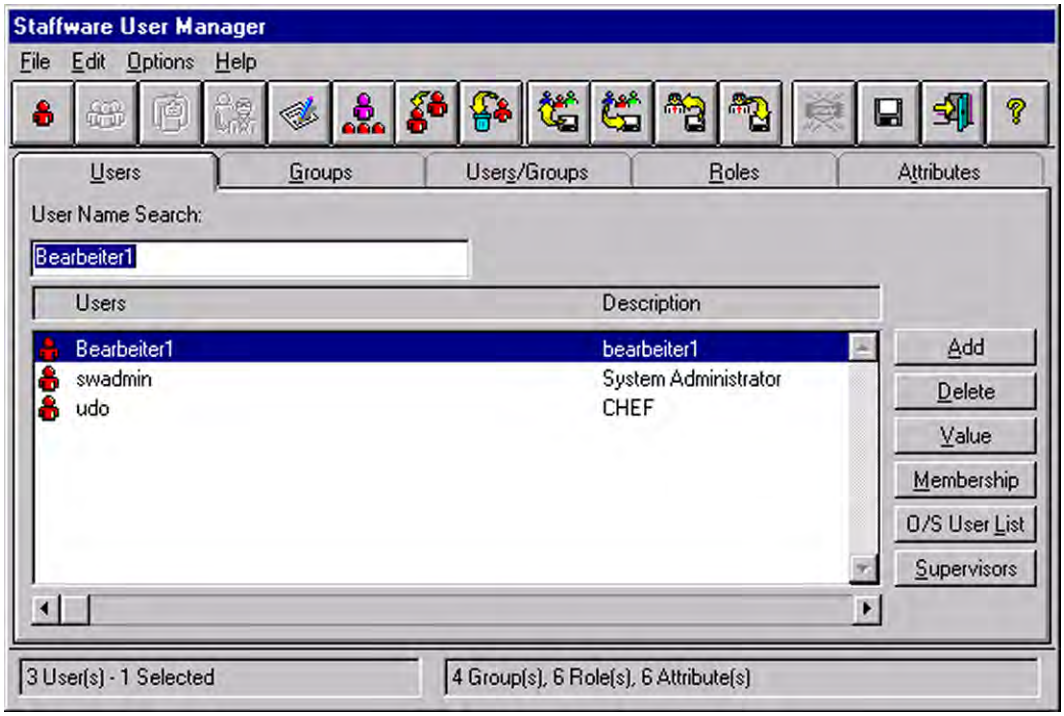



Abbildung 2.20: Das Administrations-Tool zur Erstellung des Organisationsmodells

### 2.2.2 Vorgehensweise

Die Modellierung eines Prozessmodells erfolgt schrittweise. Durch eine Modellierung in der nachfolgend beschriebenen Reihenfolge ist sichergestellt, daß eventuell existierende Vorbedingungen für jeden Schritt erfüllt sind und das Prozessmodell die gewünschte Funktionalität erbringt.

- Modellierung der Mitarbeiterstruktur
- Erstellen des Prozessmodelldiagrammes. Dazu müssen die Aktivitäten und die Kontrollflusslogik erstellt werden. Die Modellierung von Aktivitäten enthält:
  - Bearbeiterzuordnung
  - Zugeordnete Formulare

  - Zugehörigen Aktionen (z.B. Skripte)
  - Deadline

- Testen des Prozessmodells
- Freigabe des Prozessmodells

In den folgenden Abschnitten wird kurz auf jede dieser Vorbedingungen eingegangen. Abschließend erfolgt eine Darstellung der Speicherung der Daten.

#### 2.2.2.1 Erstellen von Mitarbeiterstrukturen

Mitarbeiterstrukturen können auf zwei verschiedene Arten erstellt werden.

1. Eigenständiges Modellieren der Mitarbeiterstruktur durch einen System-Administrator.
   In der Benutzerverwaltung des Administration-Tools können Personen, Rollen und Gruppen angelegt werden. Entsprechend der Auswahl der Registerkarte für ein Organisationsobjekt stellt die Werkzeugleiste Funktionen zum Erstellen, Editieren oder Löschen zur Verfügung. Ausgehend von den Abhängigkeiten der Objekte (siehe Abbildung 2.21) bietet sich folgende Modellierungsreihenfolge an:

   - Personen
   - Gruppen
   - Rollen
   - Beziehungen zwischen Personen und Gruppen

   Beim Erstellen eines Organisationsobjektes ist die Angabe des Namens obligat. Namen von Personen werden daraufhin überprüft, ob in der Benutzerverwaltung (von Windows NT) bereits eine Person mit diesem Namen existiert. Ist dies nicht der Fall, wird das Anlegen der Person nicht gestattet. Dies liegt daran, dass Staffware-Benutzern kein Passwort zugeordnet werden kann. Das Passwort für das Betriebssystem wird gleichzeitig Staffware-Passwort. Bei Personen und Gruppen kann optional eine Beschreibung des Objektes angegeben werden.
   Beim Anlegen einer Rolle wird obligat die Eingabe einer zugeordneten Person oder Gruppe erwartet. Daher ist bei korrekter Eingabe zu einem späterem Zeitpunkt keine Modellierung der Beziehung Rolle zu Person oder Gruppe nötig.
   Gruppen können theoretisch ohne eine Person als Mitglied angelegt werden. Normalerweise werden nach der Erstellung mittels `Membership` die Mitglieder festgelgt. Diese "leeren" Gruppen können in der weitern Modellierung verwendet werden, z.B. für eine Rollendefinition. `Rollen` hingegen können nur bei gleichzeitiger Angabe der zugehörigen `Person` oder `Gruppe` erstellt werden. Die Überwachung der Korrektheit der Mitarbeiterstruktur ist allein dem Modellierer überlassen.
   Für die Objekte werden mögliche Eigenschaften, wie z.B. `Language`, vorgegeben. Diese können beliebig erweitert werden. Eine Eigenschaft besteht aus einem beschreibenden Namen und der Festlegung des Typs (z.B. `Numeric`), von welchem ein zugehöriger Wert sein muß. Die Ernennung eines `Supervisor` für jede `Person` und jede `Rolle` ist nicht zwingend. Der `Supervisor` ist die einzigste Möglichkeit der Implementierung einer Vertreterregelung (siehe Kapitel 2.2.3.1) und daher von besonderer Bedeutung.

2. Durch Übernahme aus einem externen `directory service`.
   Durch die Unterstützung des `LDAP` (*Lightweight Directory Access Protocol*) ist es Staffware möglich, `directory services` wie X.500 und Microsoft Active Directory zu integrieren [SW01]. Komplette Mitarbeiterstukturen können somit importiert werden und ersparen eine

doppelte Administration von Staffware-Benutzern. Im Falle von X.500 [SW99b] kann im `User Manager` noch die Mitarbeiterstruktur betrachtet werden, Änderungen sind aber nur noch über den X.500 `DIT` (*Directory Information Tree*) [Web98] möglich.
Der wohl wichtigste Punkt für die Verwendung von `directory services` ist die Möglichkeit die fehlende hierarchische Strukturierung des Staffware-Organisationsmodells zu umgehen.

#### 2.2.2.2 Erstellen von Prozessmodelldiagrammen

Prozessmodelldiagramme werden im GWD erstellt. Die Werkzeugleiste stellt einen direkten Zugriff auf die Objekte (z.B. Aktivitäten, Konnektoren) dar. Mit einem Mausklick wird ein Objekt ausgewählt und kann anschließend auf dem Arbeitsberich positioniert werden. In [SW00a] wird folgende Reihenfolge zur Erstellung eines Prozessmodelldiagrammes vorgeschlagen:

- Planung des Prozessmodells: Dies beinhaltet das Untersuchen eines realen (Geschäfts-) Prozesses und die Unterteilung in seine Komponenten. Daraus resultieren die Aktivitäten des Prozessmodelldiagrammes. In diesem Schritt werden bereits die Formulare und deren potentielle Bearbeiter identifiziert.

- Erstellen der Aktivitäten und deren Abfolge: Dies beinhaltet die Bearbeiterzuordnung und das Erstellen der Formulare, Aktionen und Deadlines.

- Testen und Verbessern des Prozessmodells (siehe Abschnitt 2.2.2.3): Durch Testläufe kann festgestellt werden, was ein Benutzer beim späteren Ablauf des Prozessmodells sieht. Erforderlichenfalls kann das Prozessmodelldiagramm abgeändert werden.

- Freigabe des Prozessmodells (siehe Abschnitt 2.2.2.4): Falls keine Einschränkungen gemacht wurden, kann nach der Freigabe jeder Benutzer eine Instanz dieses Prozessmodells starten. Nach der Freigabe bleibt der Ersteller des Prozessmodelldiagrammes dessen Besitzer. Dadurch hat er die Möglichkeit, zu jedem späteren Zeitpunkt dieses Prozessmodelldiagramm abzuändern.

#### 2.2.2.3 Testen eines Prozessmodelldiagrammes

Während des Testvorgangs wird eine Instanz des Prozessmodells ausgeführt. Dadurch kann der Ersteller des Prozessmodells feststellen, was die späteren Benutzer bei Ansicht Ihrer Arbeitsliste sehen werden. Zur Veränderung des Prozessmodelldiagrammes stehen die selben Optionen wie bei dessen Erstellung zur Verfügung.

#### 2.2.2.4 Freigabe eines Prozessmodelldiagrammes

Erfüllt ein Prozessmodelldiagramm die Vorstellungen des Erstellers, kann es der Allgemeinheit, oder einem Teil der Benutzer, zugänglich gemacht werden. Dazu muß der Status des Prozessmodells auf `Freigegeben` gesetzt werden. Dies ist durch Aufruf von `Status` möglich. Die optionale Benutzerzuordnung erfolgt durch Aufruf von `Zugriffsberechtigung` im Menüpunkt `Vorgang`.
Vor der Speicherung eines Prozessmodells muß zusätzlich festgelegt werden, ob es sich um einen Haupt- oder Subprozess handelt. Dementsprechend kann ein Prozessmodell entweder direkt instanziiert werden oder nur als Subprozess eines Prozessmodells verwendet werden.

#### 2.2.2.5 Speicherung der Daten

Die Buildtime-Komponente von Staffware besitzt keine eigene Datenbank. Alle in der Buildtime vorgenommenen Einstellungen werden in der Runtime-Datenbank (siehe Kapitel 3) gespeichert.

### 2.2.3 Metamodelle

Die Fähigkeiten und Möglichkeiten von Staffware werden in den zugrundeliegenden Metamodellen deutlich. Im folgenden werden das Organisations-, das Prozess- und das Datenmodell untersucht. Dabei wird auf existierende Probleme und unzureichende Möglichkeiten hingewiesen.

#### 2.2.3.1 Organisationsmetamodell

Das Organisationsmetamodell stellt die Elemente `Person`, `Rolle` und `Gruppe` sowie deren Beziehungen dar (siehe Abbildung 2.21.
Dem Benutzer werden drei Modellierungselemente angeboten.

- `Person`: Mitarbeiter, die bereits im Betriebssytem als User bekannt sind, können mit ihrem User-Namen als Staffware-Benutzer angelegt werden. Diese Personen können an Prozessen teilnehmen und sie bilden die Grundlage der Mitarbeiterstruktur. Das Passwort der Person auf Betriebssystemebene ist auch für alle Staffware-Komponenten gültig und kann von diesen nicht geändert werden.

- `Rolle`: Sie stellt eine Funktion oder Fähigkeit dar, die durch genau eine Person oder Gruppe erfüllt wird. Der Rollenbegriff wird von Staffware anders verwendet als von MQSeries WF. Diese aus diesen Rollenbegriff entstehende Problematik wird in Kapitel 2.4 genauer besprochen.

- `Gruppe`: Eine Gruppe kann nur aus Personen bestehen. Mit Gruppen können Abteilungen bzw. logische Unterteilungen der Mitarbeiterstruktur abgebildet werden.

Für die Elemente `Person` und `Gruppe` können ein oder mehrere `Supervisor` ernannt werden. Ein `Supervisor` erhält zur Runtime besondere administrative Rechte in Bezug auf die ihm zugewiesenen Personen und/oder Gruppen. Ein Beispiel für solche Rechte wird im Abschnitt Vertreterregelungen beschrieben.
Im Folgenden werden einige wichtige Eigenschaften des Organisationsmodells beschrieben.

1. Vertreterregelungen:
   Für eine Person oder Gruppe können beliebig viele (auch Null) `Supervisor` festgelegt werden. Ein Supervisor kann im Staffware-Client Zugriffsberechtigungen und Weiterleitungen erstellen. Eine Zugriffsberechtigung kann an eine oder mehrere Personen vergeben werden. Eine Weiterleitung kann an eine Person oder eine Gruppe erfolgen. Vergibt ein Supervisor eine Zugriffsberechtigung auf Person A an Person B, so erhalten beide Personen die Tätigkeiten der Person A in ihrer Tätigkeitsliste. Ist eine Weiterleitung für Person A eingerichtet worden, so sieht Person A keine neuen Tätigkeiten mehr in ihrer Tätigkeitsliste. Die Tätigkeiten erscheinen nur noch bei den zur Weiterleitung angegebenen Personen oder Gruppen. Weiterleitungen und Zugiffsberechtigungen können zeitlich beschränkt sein. In diesem Fall verlieren Sie nach Ablauf eines Datums oder einer Frist ihre Gültigkeit. Sind Sie unbefristet vergeben worden bleiben Sie gültig bis der entsprechende `Supervisor` die vergebenen Rechte wieder zurücknimmt.

2. Rechtevergabe:
   Eine Person verfügt über fünf vordefinierte Eigenschaften. Dies sind die Eigenschaften `Description`, `Language`, `Menuname`, `Sortmail` und `Userflags`. Eine Möglichkeit zur Rechtevergabe steckt hinter der Eigenschaft `Menuname`. Hier wird festgelegt, auf welche Staffware-Komponenten eine Person zugreifen kann. Folgende Werte sind möglich [SW00d]:

   - `User` - normaler Benutzer: Diese Personen können Arbeitslisten einsehen und Prozesse starten. Dies ist der Standardwert.

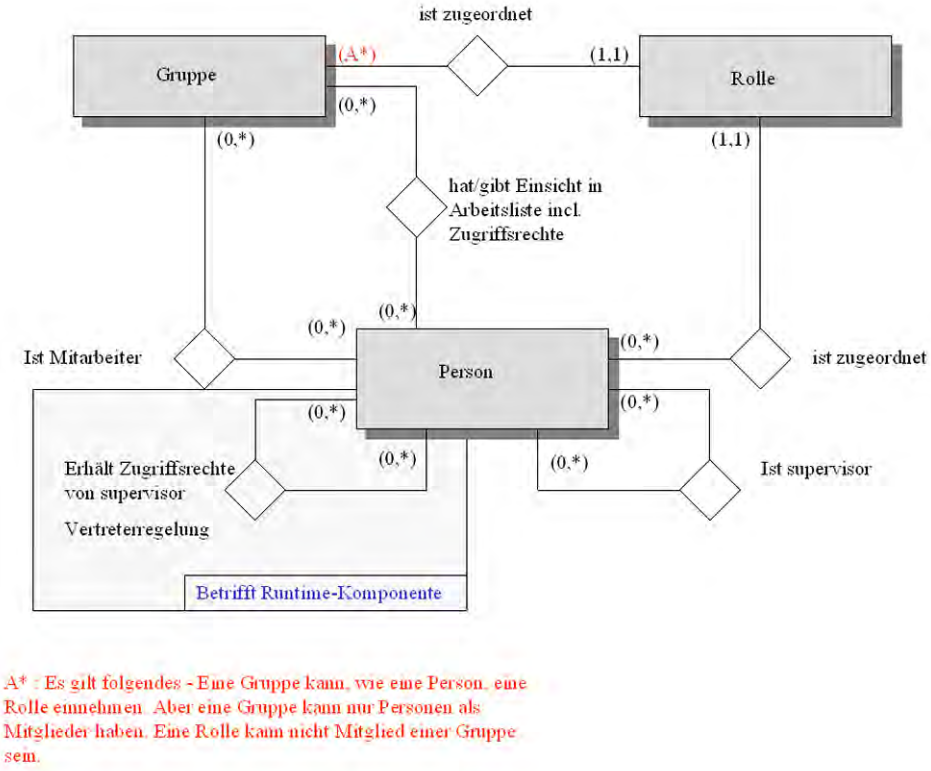



Abbildung 2.21: Die Organisationsstruktur (Staffware)

- `Manager` - Manager und Supervisor: Diese Person verfügt zusätzlich über Rechte zur Prozessadministration und zur Einsicht in den Prozessablauf (*Audit Trail*).
- `Prodef` - Prozessmodellierer: Diese Person verfügt zusätzlich über Rechte zur Erstellung von Prozessen.
- `Admin` - Systemadministratoren: Diese Personen haben uneingeschränkte Rechte in allen Staffware-Komponenten.

Der Wert für die Person `swadmin` (`Menuname = Admin`) kann nicht geändert werden. Damit ist sichergestellt, daß immer eine Person über alle Rechte verfügt.

In diesem Organisationsmodell fällt negativ auf, daß es keinerlei Möglichkeiten zur Abbildung hierarchischer Strukturen gibt. Dazu wäre es nötig, Gruppen in Beziehung zueinander stellen zu können. In der aktuellen Version können nur flache Strukturen abgebildet werden, was nicht der natürlichen Struktur eines Unternehmens entspricht.

#### 2.2.3.2 Prozessmetamodell

In Abschnitt 2.1.3.2 wurden die durch ein Prozessmodell definierten Elemente bereits aufgeführt. Im Folgenden werden die einzelnen Punkte des Prozessmodells von Staffware beleuchtet.

##### 2.2.3.2.1 Aktivitäten

Dem Modellierer stehen sieben Arten von Aktivitäten zur Verfügung. Es gibt Schritte, automatische Schritte, Skripte, Ereignisse, Subprozesse, EIS Reports und den Open Client Step.

- `Schritt`: Dieser Aktivitätentyp wird wohl am häufigsten benutzt. Mit ihm wird ein Formular erzeugt und festgelegt, welche Personen dieses zur Ausführung erhalten sollen. Zusätzlich kann eine zeitliche Einschränkung angegeben werden, bis wann der Schritt beendet sein muß.

- `Automatischer Schritt`: Der automatische Schritt startet ein externes Programm, das auf dem Staffware Server läuft. Bei Aktivierung der Aktivität bedarf es nicht dem Eingriff einer Person, der Schritt erscheint deshalb in keiner Arbeitsliste. Eine Datenübergabe kann sowohl von Staffware an das Programm, als auch umgekehrt durch Austausch von Dateien erfolgen [SW01c].

- `Skripte`: Ein Skript beinhaltet mehrere Befehle und kann an verschiedenen Stellen im Prozessablauf gerufen werden. Skripte sind besonders nützlich, wenn mehrere externe Programme gerufen werden müssen (z.B. beim Datenaustausch aktiver Windows-Anwendungen (DDE) [SW01c]). Skripte werden mit der `CALL`-Funktion in Formular- oder Feldbefehlen gerufen.

- `Ereigniss`: Mit einem Ereignis können der Ablauf eines Prozesses kontrolliert und die mit ihm verbundenen Daten von externen Programmen verändert werden. Ein Ereigniss wird nie an jemanden verschickt. Es wird vom Staffware-Server als ausstehend betrachtet, bis ein Ereignis es aufruft (mittels `swutil` mit Nennung von Instanznamen und Aktivitätenname) [SW00b].

- `Subprocedure`: Diese Aktivität ruft aus einem Prozessmodell einen bereits existierenden Sub-Prozess auf. Dazu muß ein Prozessmodell speziell als Sub-Prozess gekennzeichnet werden.

- `EIS Report`: Der EIS Report (*Executive Information System*) erstellt einen (graphisch aufbereiteten) Report aus Daten verschiedenener Prozessesinstanzen . Zum Einsatz des EIS Report wird eine separate Software benötigt.

- `Open Client Step`: Er stellt eine Aktivität dar, mit dem Tätigkeiten in ein anderes Programm, wie Micorsoft Exchange oder Lotus Notes (siehe Kapitel 5), gesendet werden können.

#### 2.2.3.2.2 Kontrollfluss

Zur Modellierung des Kontrollflusses stehen die Objekte `Linie` (*line*), `Strecke` (*router*), `Bedingung` (*condition*), `Synchronisation` (*wait*) und `Ende` (*stop*)[SW00a]. Linie, Bedingung und Synchronisation sind aktive Objekte, welche Einfluß auf den Kontrollfluss haben. Die restlichen Objekte dienen lediglich dem besseren Verständnis des Diagramms und haben erklärenden Charakter.

- Linie : Der eigentliche Kontrollfluss wird duch Linien dargestellt. Eine Linie kann nicht mit Bedingungen versehen werden. Die Art, wie eine Linie zwei Objekte verbindet, entscheidet über die zugehörige Aktion. Es gibt vier verschiedene Möglichkeiten Objekte miteinander zu verbinden (siehe Abbildung 2.22):

  1. `normal`: Der Standardkonnektor verläuft von der rechten Seite der ersten Aktivität zur linken Seite der zweiten Aktivität. Mit ihm sind keine weiteren Aktionen verbunden.
  2. `zurückgezogen` (*withdraw*): Der Konnektor verläuft von der rechten Seite der ersten Aktivität zur oberen Seite der zweiten Aktivität. Wird die erste Aktivität ausgeführt, wird die zweite zurückgezogen und aus den Arbeitslisten entfernt. Verwendung findet dieser Konnektor beispielsweise bei gleichzeitig anstehenden Aktivitäten, von denen nur eine ausgeführt werden soll.
  3. `Frist` (*deadline*): Dieser Konnektor verläuft von der unteren Seite der ersten Aktivität zur rechten Seite der zweiten Aktivität. Mit ihm ist eine Frist zur Ausführung der ersten Aktivität verbunden. Wird diese überschritten kommt alternativ die zweite Aktivität zur Ausführung.
  4. `zurückgezogen und Frist`: Dieser Konnektor verläuft von der unteren Seite der ersten Aktivität zur oberen Seite der zweiten Aktivität. Er erlaubt es, die zweite Aktivität bei Fristüberschreitung der ersten Aktivität zurückzuziehen.

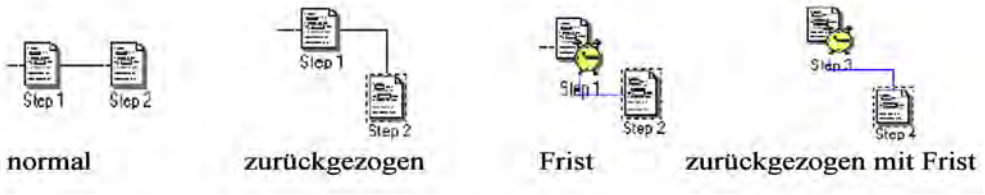



Abbildung 2.22: Möglichkeiten zur Verbindung von Aktivitäten

Konnektoren mit Fristen geben dem Modellierer die Möglichkeit, ein bestimmtes Datum oder eine gewisse Zeitspanne, bis zu dem eine Aktivität ausgeführt sein muß, festzulegen.

- `Strecke`: Dieses Objekt dient dem Modellierer, um das Prozessmodelldiagramm übersichtlicher gestalten zu können. So kann eine Linie von einer Aktivität zur nächsten über beliebig viele dieser Objekte gezielt geleitet werden. Aktionen sind nicht damit verbunden. Entscheidend ist, wie der Konnektor die erste Aktivität verläßt und wie er die zweite erreicht.

- `Bedingung`: Bedingungen werden mittels des Objektes `Bedingung` in den Kontrollfluss eingebunden. Eine Linie geht nun von einer Aktivität zur linken Seite einer Bedingung und verzweigt dort. In einer Bedingung kann ein logischer Ausdruck oder eine Verknüpfung logischer Ausdrücke stehen. Nach Freigabe der vorhergehenden Aktivität wird dieser Ausdruck ausgewertet. Im Falle von `True` folgt der Kontrollfluss der Linie, die die Bedingung an der rechten Seite verläßt. Im Falle von `False` wird die Linie nach unten verfolgt. Es können nur korrekte Ausdrücke eingefügt werden. Korrekt ist ein Ausdruck, wenn er sich auf existierende Felder des Prozessmodells bezieht und bei Vergleichen nur gleiche Typen miteinander vergleicht. Andere Ausdrücke werden mit der Fehlermeldung "Syntaxfehler" abgelehnt.

- `Synchronisation`: Dieses Objekt dient dazu, parallele Verzweigungen wieder zusammenzufassen. Das Objekt wird von der Aktivität aktiviert, deren Konnektor zu dessen linken Seite führt. Gewartet wird auf die Aktivitäten deren Konnektoren zur oberen Seite führen.

- `Ende`: Dieses Objekt verdeutlicht das Ende des Prozessmodells. Mit ihm sind keine Aktionen verbunden.

Um Teile eines Prozessmodells zu beschreiben, existiert das Objekt `Anmerkung` (*annotation*). Es kann Text aufnehmen und an gewünschter Stelle im Prozessmodell angezeigen.
Im Prozessmodelldiagramm können Zyklen modelliert werden. Dadurch ist es möglich, Prozessmodelle zu modellieren, welche niemals enden können. Der Modellierer ist selbst für den korrekten Ablauf bzw. für das korrekte Modellieren eines Prozessmodellss verantwortlich.
Häufige Abfolgen von Aktivitäten können als Sub-Prozess gespeichert und in andere (Haupt-)Prozesse eingefügt werden. `Paste and Copy` wird nicht unterstützt. Auch das "Umhängen" einmal gezogener Konnektoren ist nicht möglich. Änderungen sind nur duch Löschen und nochmaliges Einfügen möglich. Problematisch ist dies z.B. bei Bedingungen mit aufwendigen logischen Ausdrücken. Das Löschen eines Konnektors erfolgt in ungewöhnlicher Weise. Indem ein existierender Konnektor nochmals gezogen wird, wird dieser entfernt.

#### 2.2.3.2.3 Formularerstellung - "Datenfluss"

In Staffware wird der Datenfluss nicht explizit modelliert. Ein in einem Formular definiertes Feld steht allen Aktivitäten dieses Prozessmodells zur Verfügung. Die damit verbundenen Probleme muß der Modellierer selbst lösen.

- Bei Prozessmodellenen mit vielen Feldern sollte dokumentiert werden, wo ein Feld erstmalig auftritt und wo lesend oder schreibend darauf zugegriffen wird. Ansonsten könnte lesend auf ein noch nicht geschriebenes Feld zugegriffen werden!

- Es muß sichergestellt werden, daß jedes Feld, welches Daten fordert (z.B. Felder mit Formeln), auch mit entsprechenden Daten versorgt wird.

#### 2.2.3.2.4 Bearbeiterzuordnung

Soll einer Aktivität ein Bearbeiter zugeordnet werden, öffnet sich ein neues Fenster (siehe Abbildung 2.23).

Abbildung 2.23: Definition einer Bearbeiterzuordnung in Staffware

Als Bearbeiter kann festgelegt werden [SW01c]:

- `Benutzer`: Als Benutzer kann ein gültiger Personen- oder Gruppenname aus dem Staffware Benutzerverzeichnis oder ein bestimmter Name durch einen Staffware-Ausdruck (z.B. SW_STARTER für den Prozessstarter) angegeben werden. In einem System mit mehreren Staffware-Servern (*Mulitnode-System*, siehe Kapitel 3) werden Benutzer in der Form `user@node` angegeben.

- `Rolle`: Hier kann eine gültige Rolle eingetragen werden.

- `Feld`: Dies ist der Name eines Textfeldes. Bei Start einer Aktivität muß dieses Feld einen oder mehrere, durch Komma getrennte Adressaten beinhalten. Jeder Adressat muß ein gültiger Personennamen, eine Rolle oder ein Gruppenname sein.

Kombinationen von Benutzer-, Rollen- oder Feldangaben sind erlaubt. Alle Angaben werden auf Existenz im System (Benutzer und Rolle) bzw. Prozessmodell (Feld) überprüft. Bei negativem Ergebnis wird die entsprechende Eingabe mit einer aussagekräftigen Fehlermeldung zurückgewiesen. Von besonderer Bedeutung ist die Zuordnung mehrerer Bearbeiter zu einer Aktivität. So erhalten alle in der Spalte `Benutzer` angegebenen Personen und Gruppen die Aktivität in ihrer Arbeitsliste und müssen diese auch ausführen (siehe Beispiel 2.4).

**Beispiel 2.4**: (*Mehrfache Ausführung von Aktivitäten*)
Als Benutzer sind die Personen `Admin` und `Vorstand` sowie die Gruppe `Beirat` eingetragen. Das für diese Aktivität erstellte *Workitem* muß dreimal ausgeführt werden. Von den beiden Personen und einem Mitglied der angegebenen Gruppe. Erst dann wird die Aktivität freigegeben.

Eine Aktivität, die von mehreren Bearbeitern ausgeführt wird, sollte keine Datenfelder verändern. Wird dies nicht beachtet, erhält ein verwendetes Datenfeld nach der Beendigung der Aktivität dem vom letzten Bearbeiter eingegebenen Wert. Die Werte der vorherigen Bearbeiter gehen verloren (*Lost Update*).

#### 2.2.3.2.5 Zeit

Der maximale Zeitaufwand für eine Aktivität kann durch eine Frist (Deadline) angegeben werden. Eine Frist kann als Zeitraum oder als Ausdruck festgelegt werden (siehe Abbildung 2.24). Ein Zeitraum wird durch Angabe von Jahr, Monat, Wochen, Tag, Stunden und Minuten definiert. Ein Ausdruck wird durch ein Datum und eine Uhrzeit dargestellt. Beide Varianten können durch zusätzliche Eigenschaften ergänzt werden. So kann erzwungen werden, daß die Aktivität nach Ablauf der Frist zurückgezogen wird und somit nicht mehr ausgeführt werden kann.
Ob die Fristen überhaupt ausgewertet werden, kann von einer oder mehreren Bedingungen abhängig gemacht werden. Es muß mindestens eine Bedingung zutreffen, damit die Fristen ausgewertet werden.

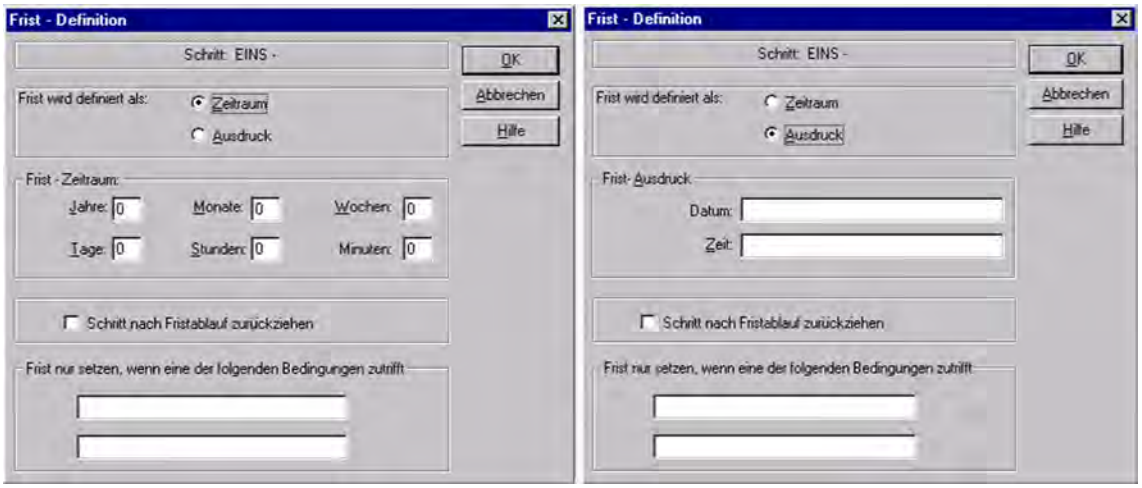



Abbildung 2.24: Deadlines mittels Zeitraum bzw. Ausdruck

### 2.2.3.3 Datenmetamodell

Als Grundlage des Datenmetamodells werden die Fähigkeiten und Möglichkeiten des `GFD` betrachtet.
Zur Erstellung eines Formulars können Felder definiert werden. Mögliche Feldtypen sind `Text`, `Numerisch`, `Numerisch durch Komma getrennt`, `Datum`, `Zeit`, `Memo`, `Anhang` und `Datenbank`.
Die acht Feldtypen sind wie folgt definiert ([SW01a]):

- `Text`: alpanumerisch; 1-255 Zeichen
- `Numerisch`: ausschließlich numerisch; 2-15 Zeichen; Dezimalpunkt möglich (Bsp.: 244,547.83)
- `Numerisch durch Komma getrennt`: numerisch mit Trennung der Tausenderstellung durch Komma (Bsp.: 1,113,762.95)

- `Datum`: numerisch; vorgegebene Länge von zehn Zeichen (MM/TT/JJ oder MM/TT/JJJJ abhängig von Einstellung in Konfigurationsdatei)
- `Zeit`: numerisch; vorgegebene Länge von fünf Zeichen (hh:mm)
- `Memo`: alphanumerisch; Text ohne Beschränkung auf Anzahl der Zeichen (wird in einem neuen Fenster erstellt und in separater Datei gespeichert).
- `Anhang`: alphanumerisch; 1-64 Zeichen; muss den kompleten Pfad des Dateinamens enthalten. Als Anhang wird eine externe Datei bezeichnet, die mit einer Aktivität verbunden ist und zur Arbeitsliste geleitet wird.
- `Datenbank`: Stellt Bezug auf eine Staffware-Tabelle her; diese wird im Auswahlmenü `Tabelle` angegeben.

Im `Administration Tool` können Staffware-spezifische Datenbanken (*Tables*) angelegt werden ([SW00d]). Diese dienen zur Speicherung häufig benutzter Werte. Die Übernahme solcher Werte in Formularfelder erfolgt anhand von Schlüsselwerten in der Datenbank.
Wird ein Feld in ein Formular eingefügt, so muß ein zuvor definiertes Feld und eine der folgenden Eigenschaften ausgewählt werden [SW01a]:

- `Muss`: Das Feld muß vom Benutzer ausgefüllt werden, bevor das Formular freigegeben werden kann.
- `Optional`: Ein Formular kann freigegeben werden, auch wenn dieses Feld nicht ausgefüllt ist.
- `Anzeige`: Der Wert des Feldes wird angezeigt und kann vom Benutzer nicht geändert werden.
- `Berechnet`: Der Wert des Feldes wird anhand einer spezifischen Kalkulation, die bei der Erstellung des Feldes angegeben werden muß, errechnet.
- `Verborgen`: Funktioniert wie `Berechnet`, der Wert wird aber nicht im Formular dargestellt.
- `Als Text anzeigen`: Es wird nur die Anzahl der Zeichen für den Wert des Feldes angezeigt.

### 2.2.4 WPDL

Da neu erstellte Prozessmodelle direkt in der Runtime-Datenbank gespeichert werden, wird die WPDL lediglich zum Austausch von Prozessmodellen zwischen verschiedenen Systemen oder zur Sicherung von Prozessmodellen benötigt. Die von Staffware verwendete WPDL wird mit `XFR` abgekürzt. Das Importieren und Exportieren von Prozessmodellen erfolgt über das Programm `SWUTIL.exe`. Dieses Programm liegt im Verzeichnisbaum des Staffware-Servers unter `X:/swserver/`*ComputerName*`/bin`. Um in dieses Verzeichnis zu gelangen muß der Benutzer über Staffware-Administrationsrechte verfügen. Nach dessen Aufruf erscheint eine DOS-Box mit einer Auswahl an Funktionen. Zum Importieren bzw. Exportieren ist es wichtig, den exakten Namen des Prozessmodells zu kennen, da dieses Tool den Benutzer keine Auswahl aus den vorhandenen Prozessmodellen anbietet. Zu importierende XFR-Dateien müssen im Verzeichnis *../swserver/util* abgelegt werden.
Ein partielles Exportieren von beispielsweise Personen ist wegen der Verknüpfung von Staffware und dem Betriebssystem nicht möglich.

## 2.3 Tabellarischer Vergleich der Systeme

| **Systeme** | **MQSeries WF** | **Staffware 2000** |
|---|---|---|
| Komponenten<br>*mit Buildtime-Fähigkeiten* | Buildtime | GWD(1) - GFD(2)<br>Admin-Tool(3)<br>SWUTIL.exe (4) |
| | | |
| **System-Ausstattung** | | |
| Microsoft Windows NT 4.0 Workstation SP4 | √ | √ |
| AMD Duron 650Mhz Prozessor; 256MB RAM | √ | √ |
| Alle Dienste der Systeme als NT-Services | √ | √ |
| **Editor** | | |
| Import und Export von WPDL-Files | √ | √ (4) |
| Erstellen des Prozessmodells | √ | √ (1) |
| Erstellen des Organisatonsmodells | √ | √ (3) |
| Erstellen des Datenmodells | √ | √ (2) |
| **Prozessmodell** | | |
| Explizite Kontrollflusslogik | X | √ |
| Implizite Kontrollflusslogik | √ | X |
| Death-Path-Elemination | √ | X |
| Modellierung von Schleifen | X | √ |
| Eigenständige Datenflussmodellierung | √ | X |
| Datenzugriff abhängig von Kontrollfluss | √ | X |
| Deadlines für Prozessisntanzen | √ | √ |
| Deadlines für Aktivitäten | | |
| Mittels Zeitraum | √ | √ |
| Mittels Ausdruck | X | √ |
| Benachrichtigung bei Überschreiten | √ | √ |
| Spezielle Folgeaktionen bei Überschreiten | X | √ |
| Einbinden von Block-Aktivitäten | √ | X |
| Einbinden von Sub-Prozesse | √ | √ |
| Differenzierung Haupt-/Sub-Prozess | X | √ |
| Bearbeiterzuordnung abhängig vom Prozessverlauf | √ | √ |
| Aktivität muß von mehreren Personen ausgeführt werden | X | √ |
| Verkleinerte Ansicht des Prozessmodells | X | X |
| Kategorisierung der Prozessmodelle | √ | X |
| **Datenmodell** | | |
| Erstellen neuer Datentypen | X | X |
| Wiederverwendbarkeit von Datenstrukturen (kein copy&paste) | √ | X |
| Begrenzte Größe von Datenstrukturen | √ | X |
| Datentyp Boolean | X | X |
| Datentyp Date | X | √ |
| Import und Export von Tables | X | √ (4) |

| | **MQSeries WF** | **Staffware 2000** |
|---|---|---|
| | Buildtime | GWD(1) - GFD(2)<br>Admin-Tool(3)<br>SWUTIL.exe (4) |
| | | |
| **Organisationsmodell** | | |
| Abbildung hierarchischer Strukturen | √ | X |
| Rechtevergabe bezüglich Build-/Runtime | √ | √ |
| Rollendefinitionen für mehrere Einzelpersonen | √ | X |
| Rollen durch Eigenschaften unterscheidbar | X | X |
| Einsicht in Arbeitsliste an Personen vergeben | √ | X |
| direkte Vertreterregelung | √ | X |
| indirekte Vertreterregelung (Supervisor) | X | √ |
| Vertretung durch mehrere Personen (Runtime) | X | √ |
| **Simulieren und verifizieren von Prozessen** | | |
| Simulation | X | √ (Runtime) |
| Verifikation einiger Merkmale | √ | X |
| **Allgemeines** | | |
| Freigabe eines Prozessmodells ab bestimmten Datum | √ | X |
| Freigabe eines Prozessmodellss an bestimmte Personen | X | √ |
| Datumsberechung auf Werktage beschränkbar [SW00d] | X | √ |
| Verwaltung aller Modelle in einem Tool | √ | X |
| eigene Buildtime-Datenbank | √ | X |

Tabelle 2.2: Vergleich der Buildtime-Komponenten

## 2.4 Kritik und Bewertung

Die Buildtime-Komponenten beider Systeme haben Stärken und Schwächen. Dieses Kapitel beschäftigt sich hauptsächlich mit den Schwächen der Systeme. Wird eine Thema nur auf ein System bezogen, so kann davon ausgegangen werden, daß angesprochene Probleme im anderen System gelöst sind oder die ganze Thematik im anderen System nicht anzutreffen ist.
Der erste auffällige Unterschied der beiden Buildtime-Komponenten liegt in ihrer Umsetzung. So findet man die Buildtime-Komponente in `MQSeries WF` integriert in einem Tool. `Staffware` hingegen hat die Fähigkeiten und Funktionen der Buldtime-Komponente auf mehrere Tools verteilt. Dies vermindert die Übersichtlichkeit für den Benutzer und erschwert dessen Arbeit.

### 2.4.1 Editor

Auf den ersten Blick erscheint der `GWD` von `Staffware` sehr übersichtlich. Die Objekte zur Modellierung eines Prozessmodells sind zum schnellen Zugriff in der Werkzeugleiste abgelegt. Der Arbeitsbereich ist sofort geöffnet und es kann mit der Modellierung begonnen werden. In `MQSeries WF` kann erst nach dem Erstellen eines Prozessmodells und dem Öffnen des zugehörigen Prozessmodelldiagramms im Arbeitsbereich mit dem Modellieren begonnen werden.
Im Arbeitsbereich beider WFMS werden Prozessmodelle graphisch dargestellt. In `MQSeries WF` können mehrere Prozessmodelle gleichzeitig angezeigt werden. Für jedes Prozessmodell öffnet sich im Arbeitsbereich der Buildtime-Komponente ein neues Fensters. Die Anordnung der Fenster ist dem Benutzer überlassen. Hilfreich ist diese Eigenschaft, falls Teile eines Prozessmodells in einen anderes Prozessmodell mittels *"Copy and Paste"* übernommen werden sollen. Mit `Staffware` kann immer nur ein Prozessmodell betrachtet werden. Das zusätzliche Öffnen und Schließen von Prozessmodellen erhöht den Arbeitsaufwand des Benutzers.
Bei beiden Systemen fällt auf, daß die graphische Gestaltung eines Prozessmodells sehr stark vom Benutzer abhängt. Dies kann als Vorteil und als Nachteil ausgelegt werden. So können der Verlauf der Konnektoren (sehr umständlich bei nachträglichen Änderungen) und deren Farbe vom Benutzer gewählt werden. `Staffware` unterstützt den Benutzer zusätzlich durch eine Reihe automatischer graphischer Markierungen. Beispielsweise werden die ein- und ausgehenden Konnektoren der markierten Aktivität durch eine gestichtelte Linie dargestellt. Positiv in beiden Systemen ist, daß das Anbringen von ungültigen Verbindungen nicht zugelassen wird und somit eine Fehlerquelle ausgeschlossen wird.
Die Unterstützung des Benutzers durch Bereitstellung von möglichen Daten fehlt oftmals bei `Staffware`. Bei dem Erstellen von logischen Ausdrücken für Bedingungen wird der Benutzer vom System nicht unterstützt. Die Festlegung der Empfänger von Aktivitäten wird nur begrenzt durch eine Anzeige von möglichen Benutzern und Gruppen unterstützt. Eine Anzeige von definierten Rollen oder Feldern findet nicht statt. Der Benutzer muß sich sehr genau auskennen (z.B. Schreibweise von Variblen) oder umständlich in einer Systemkomponente nachsehen. In `MQSeries WF` wiegt dieses Problem nicht so schwer, da alle Daten in einer Komponente vereinigt sind. So kann sich der Benutzer schneller benötigtes Wissen aneignen.

### 2.4.2 Prozessmetamodell

Die Möglichkeiten zur Bearbeiterzuordnung sind in beiden Systemen unzureichend. Folgendes Beispiel soll dies verdeutlichen.

**Beispiel 2.5**: (*Unzureichende Bearbeiterzurodnung*)
Ein Krankenhaus verfügt über die Abteilungen 1 (Kleinkinder), die Abteilung 2 (Notaufnahme) und die Abteilung 3 (Röntgen). Bringt ein Pfleger aus der Abteilung 1 bzw. 2 einen Patienten zum Röntgen, so sollen die Ergebnisse dem Arzt der Abteilung zugänglich gemacht werden.

In Abbildung 2.25 ist ein Ausschnitt eines denkbaren Prozessmodells dargestellt. Die Abteilungen wurden als Organisationseinheiten (OE) übernommen. Die Personen Pfleger und Arzt wurden als Rollendefinition übernommen.

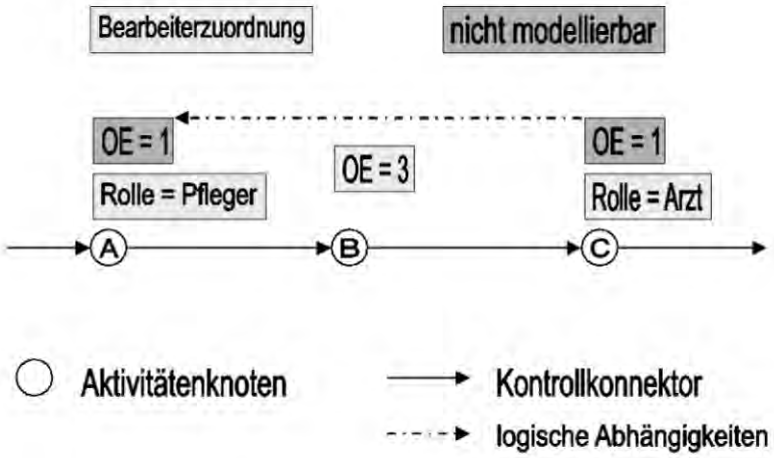



Abbildung 2.25: nicht mögliche Bearbeiterzuordnung

Die Aktivität C in Abbildung 2.25 sollen alle Ärzte aus der Abteilung, aus der der Pfleger bzw. Starter der Aktivität A stammt, erhalten. Eine logische Verknüpfung in Sinne von "die selbe Organisation wie Starter von Aktivität A" ist in beiden Systemen nicht möglich.
Als Zwischenlösung können in `MQSeries WF` Container und in `Staffware` Felder verwendet werden. In diesem Fall muß der Starter der Aktivität A seine Abteilung angeben. Durch Auswertung dieser Daten kann die Bearbeiterzuordnung in Aktivität C erfolgen. Offensichtlicher Nachteil dieser Lösung ist die Einbeziehung des Benutzers, welche gerade durch die Bearbeiterzuordnung verhindert werden soll.

`MQSeries WF` bietet die Möglichkeit große Prozessmodelle mit Hilfe von Blockaktivitäten logisch zu gliedern und übersichtlicher zu gestalten. `Staffware` bietet hierzu keine vergleichbare Lösung an. Im Ernstfall muß sich der Staffware-Benutzer mühsam, eventuell unter Zuhilfename von `zoom in/out`, zum gewünschten Ausschnitt des Prozessmodells navigieren. Eine verkleinerte Gesamtdarstellung eines Prozessmodells fehlt in beiden Systemen.

### 2.4.3 Organisationsmetamodell

Ein Organisationsmetamodell sollte fähig sein reale Unternehmensstrukturen angemessen abzubilden. Ein Unternehmen kann aus verschiedenen Bereichen (z.B. angekaufte Firmen) bestehen, welche sich wiederum aus Abteilungen und Gruppen zusammensetzen (hierarchischer Aufbau). Erfüllt ein Organisationsmetamodell diese Anforderungen nicht, so ist das ganze System für viele Unternehmen nicht zum flächendeckenden Einsatz geeignet.
`Staffware` bietet selbst keine Möglichkeiten zur Abbildung von hierarchischen Strukturen im Organisationsmetamodell. Hierarchische Strukturen können nur unter Verwendung vorhandener *directory services* geschaffen werden. Wieviele Unternehmen (besonders Unternehmen aus dem Mittelstand) verfügen aber beispielsweise über `X.500` oder sind bereit, `X.500` zur Verwendung in `Staffware` einzuführen? In `MQSeries WF` können hierarchische Sturkturen durch Organisationen und deren Beziehungen untereinander abgebildet werden.

Der von `Staffware` verwendete Begriff "Rolle" ist für den Benutzer verwirrend. Da eine Rolle nur mit einer Person besetzt werden kann handelt es sich hierbei eher um eine Stelle. Man kann einer Rolle aber auch eine Gruppe zuweisen und dadurch, wenn auch auf indirekten Weg, wieder eine Rollendefinition erhalten. Unklar ist, wozu diese Zuweisung gut sein soll, wenn bereits effektiv mit der Gruppe gearbeitet werden kann. Dies verkompliziert nur das Organisationsmodell unnötig und läßt es undurchsichtiger erscheinen.

### 2.4.4 Datenmetamodell

Das Datenmetamodell beider Systeme ist nicht vollständig. Beispielsweise fehlt in beiden Systemen der Datentyp `Boolean`, der beispielsweise für Bedingungen sehr gut verwendet werden könnte.
Neue Datentypen können in beiden Systemen nicht erstellt werden. Der Benutzer muß mit den angebotenem Datenmetamodell auskommen.
`Staffware` hebt sich positiv durch den Datentyp `Datum` und die mögliche Verwendung von `Tables` (kleinen Datenbanken) ab.
Negativ fällt bei `MQSeries WF` auf, daß das gesetzte Limit von 512 Mitgliedern pro Datenstruktur nicht konsequent überwacht wird. Die Buildtime-Komponente verhindert das Anlegen von Arrays mit mehr als 512 Elementen. Das Anlegen von Datenstrukturen mit mehr als einem Array der Größe 511 wird aber nicht unterbunden. Die Reaktionen darauf sind nicht überschaubar[3].

### 2.4.5 Test und Verifikation

Bevor ein neues Prozessmodell zur Benutzung für alle Personen freigegeben wird, sollte der Modellierer sicherstellen können, daß es die gewünschte Funktionalität erbringt.
In beiden Systemen werden viele mögliche Fehlerquellen während der Modellierung bereits verhindert (z.B. Schleifen in MQSeries WF und Bearbeiterzuordnung mit nicht existierender Person in Staffware). Dennoch verbleiben eine Vielzahl an mögichen Fehlerquellen.
`MQSeries WF` bietet in der Menüleiste der Buildtime-Komponente unter `Process` den Menüpunkt `Verify`. Dabei werden die in 2.1.2.6 erklärten Punkte überprüft.
`Staffware` ermöglicht dem Modellierer die Simulation seiner Prozessmodelle. Dabei erhält der Modellierer im Standard-Client (siehe Kapitel 3) einen extra Arbeitsliste. Die Variante von Staffware ist sehr vorteilhaft, da der Modellierer den Ablauf des Prozessmodells in seiner späteren Umgebung (Standard-Client) überprüfen kann. Bei MQSeries WF wird das Prozessmodell auf Korrektheit untersucht, wobei "falsche" (im Sinne von ungewollten) Bearbeiterzurdnungen nicht auffallen.

### 2.4.6 Tabellarische Bewertung

Zur Bewertung stehen drei Bewertungen zur Verfügung.

- \+ : gut bzw. ja
- o : neutral
- \- : schlecht bzw. nein

[3]Bei einem Test-Computer "stürzte" `MQSeries WF` regelmäßig ab, bei dem Versuch eine Datenstruktur mit mehr als 512 Mitgliedern anzusehen.

| | **MQSeries WF** | **Staffware 2000** |
|---|---|---|
| leichte und schnelle Bedienung aller Buildtime-Funktionen | + | - |
| schneller Modellierungsbeginn | o | + |
| Untergliederung von Prozessmodellen | + | o |
| Gleichzeitige Bearbeitung mehrerer Prozessmodelle | + | - |
| Unterstützung des Modellieres bei graph. Darstellung von Prozessmodellen | o | o |
| Möglichkeiten zur Bearbeiterzuordnung | o | o |
| Arbeiten mit Subprozessen | + | o |
| Hierarchischer Aufbau des Organisationsmodells | + | - |
| Mögliche Verwendung von `directory services` | - | + |
| Vollständiges Datenmodell | o | o |
| Erweiterbarkeit Datenmodell | - | - |
| Simulation von Prozessmodellen | - | + |
| Verifikation von Prozessmodellen | + | - |

Tabelle 2.3: Bewertung der Buildtime-Komponenten

# Kapitel 3

# Runtime

Die Runtime-Komponente eines WFMS stellt alle benötigten Funktionen zum Erzeugen, Ausführen, Verwalten, Überwachen und Speichern von Prozessinstanzen bereit. Spezielle Administrations-Programme bieten die Möglichkeit, Runtime-Komponenten zu konfigurieren und zu verwalten. Die Standard-Runtime-Komponenten von MQSeries WF und Staffware werden in diesem Kapitel behandelt. Die Vorstellung der Administrations-Programme erfolgt in Kapitel 5.

## 3.1 MQSeries WF

MQSeries WF besitzt eine `3-Tier-Architektur` [IBM99b]. In den folgenden Abschnitten wird diese Architektur näher erläutert und die Funktionalität der einzelnen *Tiers* (Stufen) beschrieben.

### 3.1.1 Architektur und Plattformen

In Abbildung 3.1 sind die drei *Tiers* der MQSeries WF Runtime-Komponente abgebildet. Es sind auch Installationsvarianten möglich, bei denen das System weniger als drei Bereiche aufweist, etwa wenn der WF-Server und die Datenbank auf derselben Maschine installiert werden [IBM99a].

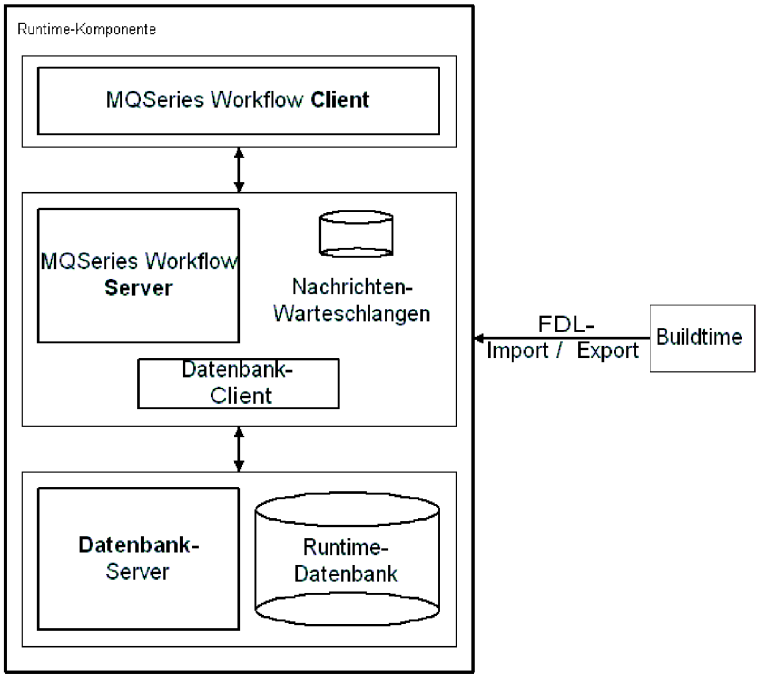



Abbildung 3.1: Architektur der MQSeries WF Runtime-Komponente

Bei den drei *Tiers* handelt es sich um die Client-Komponente, die Server-Komponenten und den Datenbank-Server.

- `Client-Komponente`: Sie steht für das Client-API von MQSeries WF (siehe Abschnitt 5.3.2) und die Client-Programme, die dieses API verwenden. Clients bieten Funktionen zur Manipulation und Darstellung von Arbeitslisten und führen die Programmaktivitäten zur Interaktion mit Benutzern mittels Programmausführungsagent aus. Die Kommunikation zwischen Client und Server (siehe Abschnitt 3.1.4) erfolgt über MQSeries, unter Verwendung der Client-Nachrichtenebene MQ Workflow oder über die CORBA [1] IIOP [2].

- `Server-Komponenten`: Die Server-Komponenten übernehmen zur Laufzeit die Ausführung und Überwachung von Prozessen. Die Komponenten dieser *Tier* können auf mehrere Maschinen verteilt werden, um die Arbeitsbelastung gleichmäßig zu verteilen. Die Kommunikation zwischen den Server-Komponenten basiert auf den Nachrichtenschlangen (*MessageQueues*) von MQSeries (siehe Abschnitt 3.1.2).

- `Datenbank-Server`: Für eine Systemgruppe von MQSeries WF verwendet die Datenbank für jede Prozessinstanz WF-Kontrolldaten (z.B. Statusinformationen) und WF-relevanten Daten (z.B. Werte von Ausgabedatencontainern). Für die Kommunikation zwischen Datenbank-Server und seinem Client werden die von DB2 unterstützen Transportprotokolle verwendet.

Nicht alle Komponenten eines MQSeries WF-Systems sind auf allen Plattformen verfügbar. Tabelle 3.1 gibt einen Überblick zur Plattformstrategie von MQSeries WF.
Dabei werden die Betriebssysteme AIX, HP-UX und Sun Solaris unter der Überschrift UNIX zusammengefaßt. Die Komponenten von MQ Workflow unter UNIX können auf allen diesen Plattformen installiert werden [IBM99a].

| Komponente | **UNIX** | **OS/2** | **OS/390** | **Windows NT** | **Windows 95/98** |
|---|---|---|---|---|---|
| Server | Ja | Ja | Ja | Ja | Nein |
| Verwaltungsprogramm | Ja | Ja | Nein [3] | Ja | Ja |
| Standard-Client | Nein | Nein | Nein [3] | Ja | Ja |
| Client für Lotus Notes | Nein | Ja | Nein [3] | Ja | Ja |
| Lotus Notes DB-Schablonen | Nein | Ja | Nein [3] | Ja | Ja |
| API Runtime-Bibliotheken | Ja | Ja | Nein [3] | Ja | Ja |
| Programmausführungsagent | Ja | Ja | Nein [3] | Ja | Ja |
| Java CORBA Agent | Ja [1] | Ja [2] | Nein [3] | Ja | Ja |
| Java Beans | Ja | Nein | Nein [3] | Ja | Ja |
| API Development Kit | Ja | Ja | Nein [3] | Ja | Ja |
| Runtime-DB-Programme | Ja | Ja | Nein [3] | Ja | Ja |

[1] Der Java CORBA Agent steht unter HP-UX nicht zur Verfügung.

[2] Der Java CORBA Agent unter OS/2 Warp unterstützt nur lokale Bindungen und schließt Java Beans mit ein.

[3] Es existieren in der verfügbaren Literatur keine Hinweise auf eine Verfügbarkeit dieser Komponente.

Tabelle 3.1: Plattformen für die MQSeries WF Runtime-Komponenten

[1] *Common Object Request Broker Architecture*
[2] *Internet InterORB Protocol*

**Systemnetze mit MQSeries WF**
Bei Bedarf können Installationen mit mehr als einem MQSeries WF-System durchgeführt werden [IBM99b]. MQSeries WF ist ein Client/Server-System mit einer hierarchisch aufgebauten Struktur (siehe Abbildung 3.2).

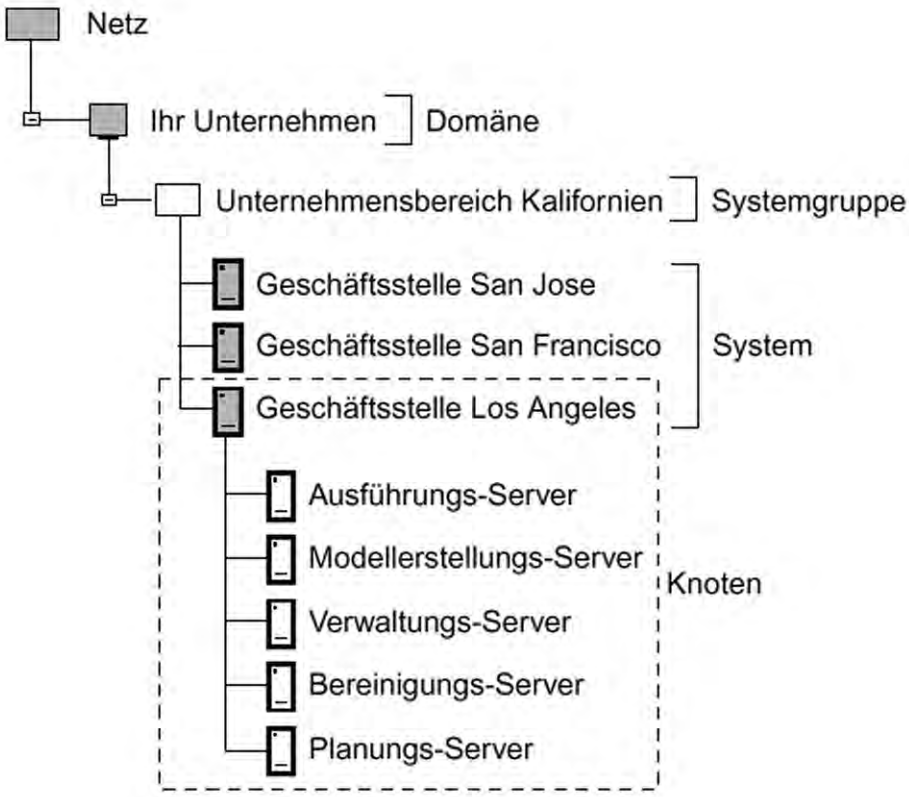



Abbildung 3.2: Systemnetz von MQSeries WF (aus [IBM99]

Die höchste Ebene in der Hierarchie ist die Domäne, die die gesamte Organisation oder Teile Ihrer Organisation darstellt. Die nächsttiefere Ebene in einer Domäne ist die Systemgruppe. Dies kann z. B. ein geografischer Bereich sein. Eine Systemgruppe kann mehrere Systeme enthalten, die auf dieselbe Datenbank zugreifen.

## 3.1.2 Server-Komponenten

Die Koordination eines MQSeries WF-Systems und seiner Clients erfolgt über die Server-Komponenten. Ein MQSeries WF-Server besteht aus einer oder mehreren Komponenten. Im Einzelnen sind dies der Ausführungs-Server (*Execution Server*), der Verwaltungs-Server (*Administration Server*), der Planungs-Server (*Scheduling Server*) und der Bereinigungs-Server (*Cleanup Server*). Zu einem MQSeries WF-Server gehört noch ein spezielles Import-/Export-Dienstprogramm. Es ist nötig, um die in der Buildtime erstellten Prozessmodelle (FDL-Dateien) der Runtime verfügbar zu machen. Abbildung 3.3 zeigt die Server-Komponenten, aus denen sich ein MQSeries WF-System zusammensetzt.

### 3.1.2.1 Aufgabenbereiche der Server-Komponenten

Die Aufgaben der einzelnen Komponenten werden nun dargestellt:

- `Ausführungs-Server`: Er stellt sicher das die richtige Aktivität zum richtigen Zeitpunkt an die richtige Person gelangt. Dazu muß er folgende Aufgaben bewältigen:

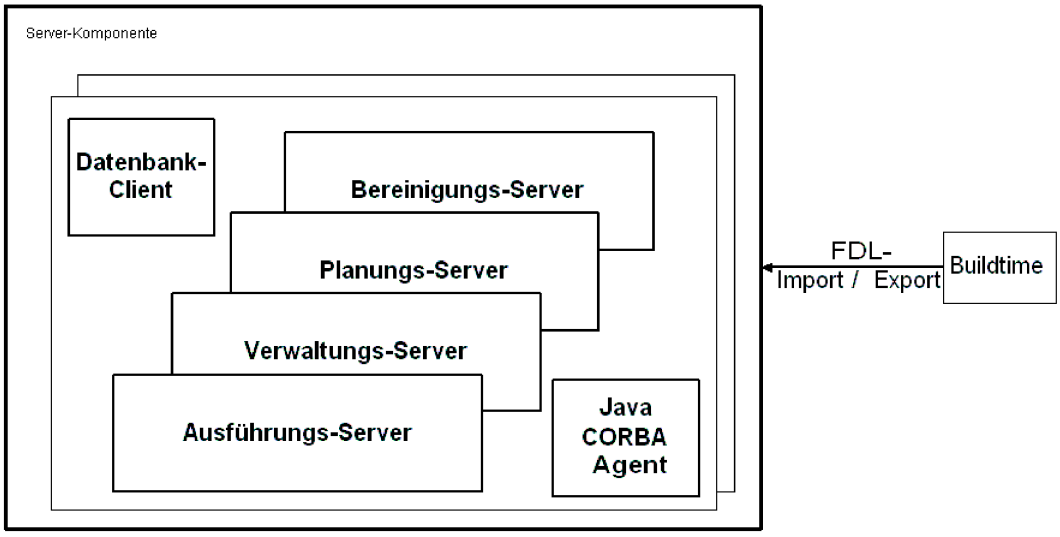



Abbildung 3.3: Server-Komponente von MQSeries WF

- Manipulation von Prozessinstanzen (Erzeugen, Start, Stopp, Aussetzen und Wiederaufnehmen) überwachen.
- Ausführung von Prozessmodellen, d.h. Interpretation der Aktivitätennetze, Auswertung von Bearbeiterzuordnungen, Ausführung von Anwendungsprogrammen sowie Übernahme bzw. Versorgung von Containerdaten.
- Navigation zwischen Aktivitäten
- Verwaltung von WF-Kontrolldaten (Statusinformationen), Protokollierungsvorgänge (*Audit Trails*)
- Verwaltung von Benutzerarbeitslisten

Der Ausführungs-Server tritt als `Datenbank-Client` auf und kommuniziert mit dem Datenbank-Server.

- `Verwaltung-Server`: Er verwaltet das MQSeries WF-System und kommuniziert mit allen anderen Komponenten in einem System oder einer Systemgruppe. Der Verwaltungsserver gewährleistet die Verfügbarkeit und den Betrieb aller Server-Komponenten. Außerdem führt er bei allen Server-Komponenten die Fehlerbehebung, z.B. nach Crash, durch.
  Der Zugriff auf den Verwaltungs-Server erfolgt über ein Verwaltungsprogramm (`fmcautil.exe`), welches im `bin`-Verzeichnis des Servers liegt. Zum Start des Programmes muß eine Person über besondere Rechte verfügen (siehe Rechtvergabe Abschnitt 2.1.3.1).

- `Planungs-Server`: Er überwacht die zur Buildtime festgelegten maximalen Bearbeitungszeiträume (*Deadlines*) der Aktivitäten. Werden gesetzte Werte überschritten, sendet er Mitteilungen an die Arbeitslisten der entsprechenden Personen.

- `Bereinigungs-Server`: Er ist für das physische Löschen abgeschlossener Prozessinstanzen verantwortlich. Abhängig von den zur Buildtime vorgenommenen Festlegungen werden abgeschlossene Prozessinstanzen entweder unverzüglich oder zu einem späteren Zeitpunkt, z.B. wenn das System inaktiv ist, gelöscht. Spezielle Archivierungsfunktionen für den Export von WF-Kontrolldaten und WF-relevanten Daten gibt es in MQSeries WF nicht.

- `Java CORBA Agent`: Er ist für die Weiterleitung der CORBA-IIOP Anforderungen von der Java-API zum Ausführungs-Server und für die Rücksendung der Antworten verantwortlich.

Für eine MQSeries WF-Systemgruppe sind nur ein Bereinigungs-Server und ein Planungs-Server erforderlich. Dagegen kann eine Systemgruppe mehrere Ausführungs-Server umfassen.
Werden mehrere Ausführungs-Server definiert, werden diese als sogenannte *Hot-Pool-Exemplare* bezeichnet. Jeder dieser Ausführungs-Server stellt eine eigene Verbindung zur Runtime-Datenbank her. Dies ist möglich, da MQSeries WF die Funktion für Mehrbenutzersysteme der DB2-Datenbank nutzen kann. Dadurch wird die Leistungsfähigkeit des gesamten Systems verbessert [IBM99b].

#### 3.1.2.2 Das Import-/Export-Dienstprogramm

Dieses Programm kann nur auf dem MQSeries WF-Server gestartet werden. Der Aufruf muß in einer `DOS`-Box erfolgen. Das Import-/Export-Dienstprogramm bietet folgende Möglichkeiten:

- Export eines Prozessmodells aus der Runtime in eine FDL-Datei
- Import einer FDL-Datei in die Runitme-Datenbank

Eine FDL-Datei, die importiert werden soll, muß nicht zwingend mit der Buildtime-Komponente von MQSeries WF erstellt worden sein. Es genügt wenn die FDL-Datei der festgelegten Syntax ([IBM99c]) entspricht.
Um sicherzustellen, daß nur "korrekte" Prozessmodelle in der Runtime-Datenbank enthalten sind, wird ein Prozessmodell während des Imports verifiziert (siehe Abschnitt 2.1.2.6). Entspricht das Prozessmodell den Anforderungen, wird es angenommen und in die Datenbank eingespielt. Im Fehlerfall wird es nicht importiert und eine aussagekräftige Fehlermeldung ausgegeben.
FDL-Dateien, die Änderungen bereits in der Runtime-Datenbank existierender Prozessmodelle beinhalten, werden gleich behandelt.

**Beispiel 3.1**: (*Import eines Prozessmodells*)
`fmcibie /i=in.fdl /u=admin /p=pwd`
Diese Befehlszeile startet das Dienstprogramm und importiert die Datei mit dem Namen *in.fdl*, angemeldet als Person *admin* mit dem Passwort *pwd*.
`fmcibie /i=in.fdl /u=admin /p=pwd /t`
Diese Befehlszeile startet das Dienstprogramm, importiert und übersetzt die FDL-Datei zur Verwendung in der Runtime-Datenbank.
ACHTUNG: Die Option `/t` (*Translation*) ist zwingend nötig wenn ein neues Prozessmodell oder Änderungen an einem existierenden Prozessmodell impotiert werden sollen. FDL-Dateien die inhaltlich nur das Organisation- oder Datenmetamodell betreffen können bzw. sollen ohne diese Option importiert werden.

**Beispiel 3.2**: (*Export eines Prozessmodells*)
`fmcibie /e=out.fdl /u=admin /p=pwd /c‘‘EXPORT PERSON*‘‘`
Diese Befehlszeile startet das Dienstprogramm und exportiert die Definitionen aller Personen des Workflow Modells in die Datei *out.fdl*.
`fmcibie /e=out.fdl /u=admin /p=pwd /c‘‘EXPORT PERSON*‘‘`
Mit der Option `/v` (*Verify*) können FDL-Dateien, die z.B. nicht mit der MQSeries WF Buildtime-Komponente erstellt worden sind, verifiziert werden.

### 3.1.3 Standard-Client

Für die Betriebssysteme Windows NT 4.0 und Windows 95/98 von Microsoft wird ein Standard-Client bereitgestellt. Mit dem MQSeries WF Standard-Client können Prozessmodelle verwaltet, Prozessinstanzen erzeugt, gestartet, überwacht und Arbeitslisten verwaltet werden. Mit Hilfe des Standard-Clients kann der Benutzer aktiv in den Arbeitsablauf einer Instanz bzw. in die Kontrolle eines Arbeitslisteneintrages (*Workitem*) eingreifen.

#### 3.1.3.1 Starten und Einrichten des Standard-Clients

Der Standard-Client kann unter Windows NT durch Auswahl von `MQSeries Workflow Client` im Programm-Menü `MQSeries Workflow` gestartet werden. Es erscheint das Fenster zur Anmeldung. Hier muß eine gültige Benutzer-ID und das dazugehörige Passwort angegeben werden. Der Standard-Login nach der Installation ist User=*ADMIN* und Passwort=*password* [IBM99f]. Danach öffnet sich das Fenster des Standard-Clients (Abbildung 3.4).

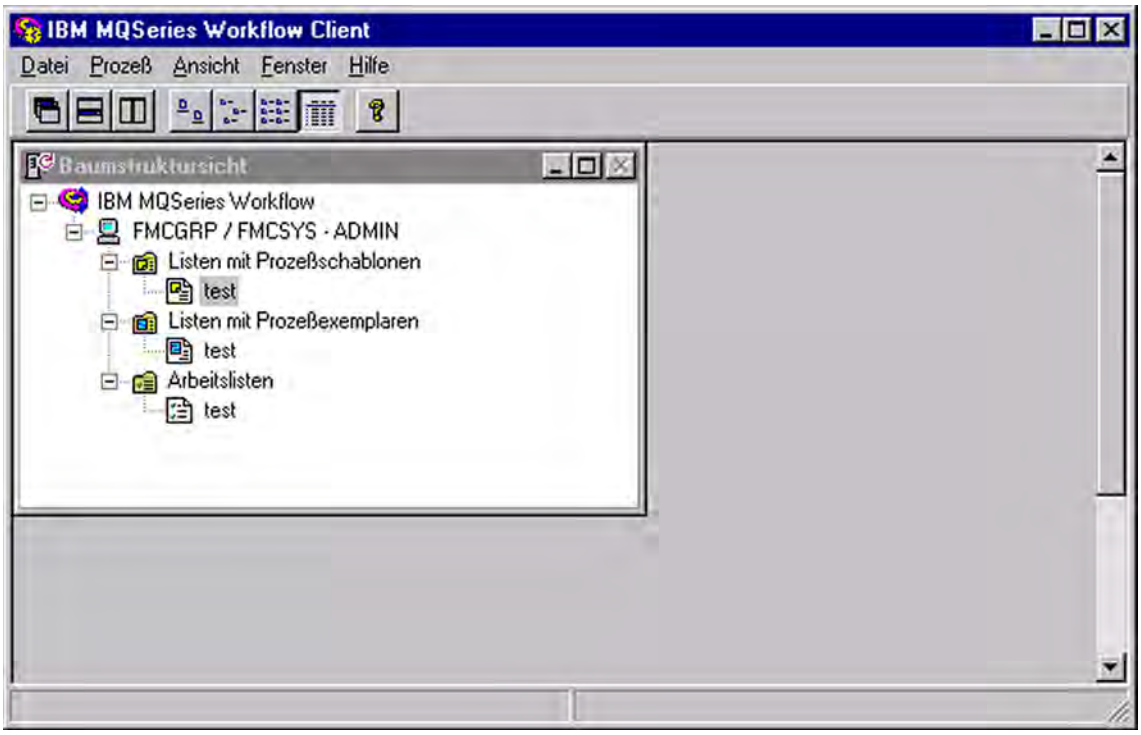



Abbildung 3.4: Runtime-Client von MQSeries WF

Im Standard-Client erscheint das Fenster `Baumstruktursicht` mit folgendem Inhalt:

- Systemgruppe / System / Benutzer-ID
- Listen mit Prozessschablonen (*Prozessmodellen*)
- Listen mit Prozessexemplaren (*Prozessinstanzen*)
- Arbeitslisten es angemeldeten Benutzers

Eine `Prozessschablone` entspricht einem in der Buildtime erstellten Prozessmodell. Ein `Prozessexemplar` stellt eine Instanz eines Prozessmodells dar. In einer Arbeitsliste werden die von einer Person auszuführenden Aufgaben (*Workitems*) angezeigt. In Abbildung 3.4 existiert eine Arbeitsliste sowie je eine Liste für Prozeßschablonen und Prozeßexemplare. Alle Listen haben den Namen `test`. Durch einen Doppelklick auf eine Liste wird das dazugehörige Fenster markiert. In Abbildung 3.5 sind die Personen V2 und ADMIN am Standard-Client angemeldet. Für beide Personen sind alle Listen-Fenster geöffnet.
Beim erstmaligen Aufruf des Standard-Client nach Installation des WF-Systems (normalerweise sollte dies durch den System-Administrator erfolgen) ist zu beachten, dass noch keine Listen angelegt sind. Die in Abbildung 3.4 sichtbaren Listen `test` wurden bereits durch den System-Administrator, beim erstmaligen Aufruf des Standard-Client, angelegt und erscheinen danach bei allen anderen Personen standardmäßig.

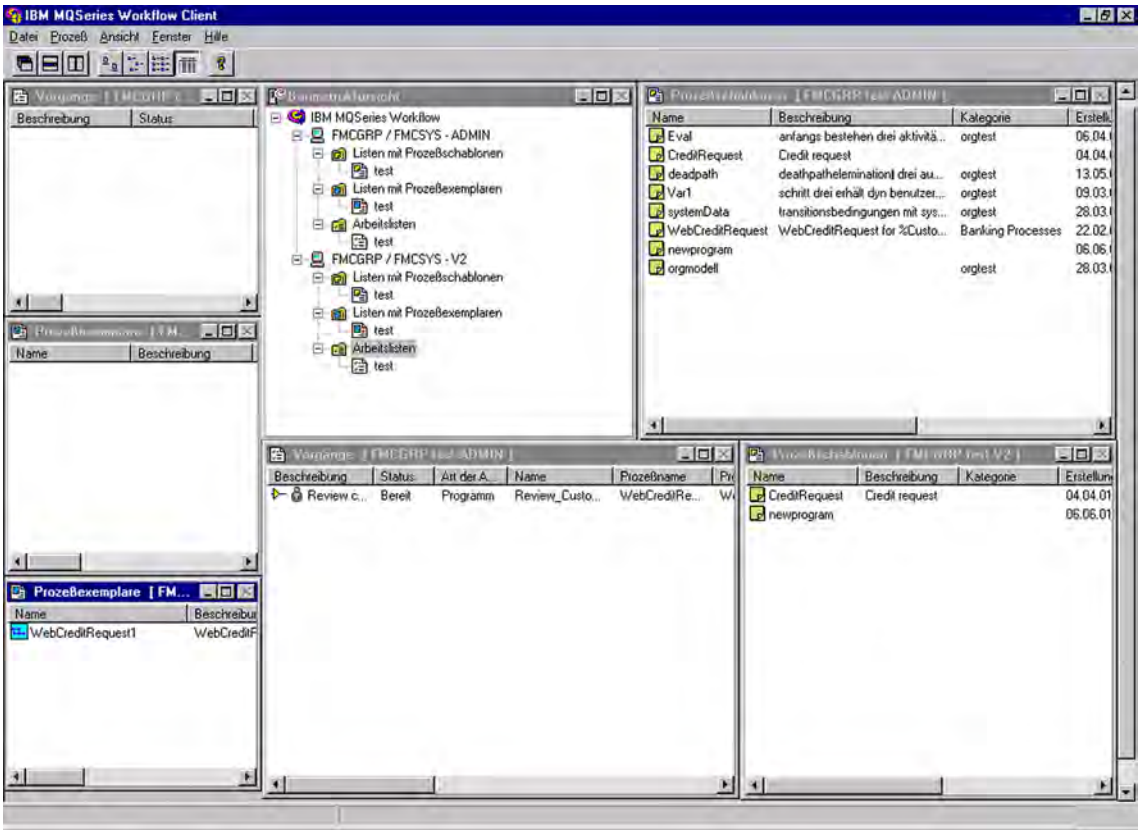



Abbildung 3.5: Runtime-Client von MQSeries WF (incl. Listen)

**Erstellen von Listen**
Zum Erstellen einer Liste sind folgende Schritte nötig:

- Rechtsklick in der Baumstrukuransicht auf `Listen mit Prozesßschablonen`, `Listen mit Prozeßexemplaren` oder `Arbeitslisten`. Es öffnet sich ein Menü.
- Auswahl des Menüpunktes `Erstellen einer neuen Liste`.
- Angabe der Eigenschaften der neuen Liste (siehe unten).

In Abbildung 3.6 ist das Fenster zum Anlegen einer neuen Liste abgebildet. Es existieren die Register `Allgemein`, `Filtern` und `Sortieren`. Unter `Allgemein` muß der Listenname zwingend eingegeben werden. Mögliche Einstellungen in den Registern `Filtern` und `Sortieren` sind optional und dienen lediglich einer benutzerangepassten Darstellung der Arbeitsliste.
Filtereigenschaften einer Arbeitsliste legen beispielsweise fest, welche *Workitems* angezeigt werden sollen. So kann eine Person, die viele *Workitems* erhält und zusätzlich noch Einblick in die Arbeitslisten anderer Personen hat (diese *Workitems* werden ebenfalls in ihrer Arbeitsliste angezeigt), mit Hilfe von Filtereigenschaften mehrere Arbeitslisten definieren. Mindestens zwei Arbeitslisten wären sinnvoll: Eine Arbeitsliste für die eigenen, direkt an die Person adressierten *Workitems* und eine weitere Arbeitsliste für die restelichen, nur durch ihre besonderen Rechte (z.B. Vertreterregelung) sichtbaren *Workitems*.
Es können beliebig viele Prozeßschablonen-, Prozeßexemplar- und Arbeitslisten pro Person angelegt werden. Wichtig ist nur, daß die Gesamtheit der Listen alle möglichen Fälle bzw. Eigenschaften abdeckt. Ansonsten könnten z.B. *Workitems* "verloren gehen", da Sie in keiner Arbeitsliste angezeigt werden. Überschneidungen von Listen führen dazu, daß ein *Workitem* in mehreren Listen vorkommen kann.

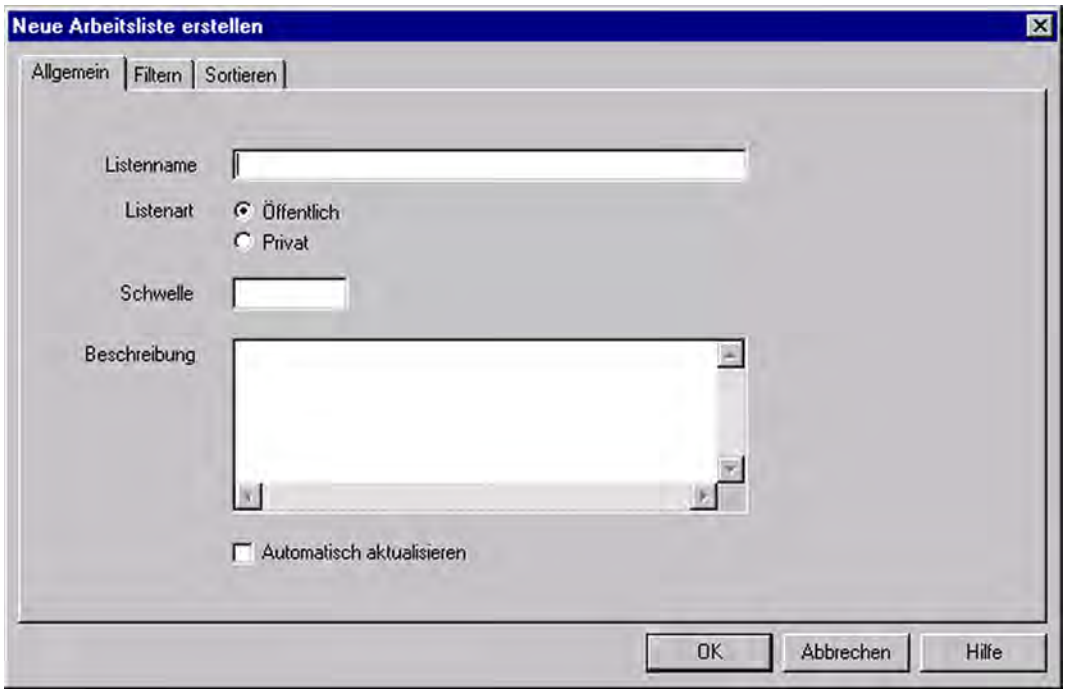



Abbildung 3.6: Erstellen einer neuen Liste in MQSeries WF Runtime

#### 3.1.3.2 Prozeßschablonenlisten

In einer Prozeßschablonenliste werden alle für die Person verfügbaren Prozessmodelle angezeigt. Wieviele Prozessmodelle eine Person in der Liste angezeigt bekommt, hängt von den Filtereinstellungen der Liste und den Rechten der Person ab.
Die Rechtevergabe erfolgte in der Buildtime anhand der Prozess- Kategorien. Eine Person sieht in ihrer Liste nur Prozessmodelle aus Kategorien, für die Sie entsprechende Rechte erhalten hat. In 3.5 hat die Person `ADMIN` Rechte für die Kategorien *orgtest*, *Banking Processes* und "*no category assigned*" (Prozesse ohne Zuordnung). Die Person `V2` hingegen hat nur die Rechte für die Kategorie "*no category assigned*".
Um einen Prozessmodell starten zu können, muß zuerst eine Instanz (Prozessexemplar) des Prozessmodells erzeugt werden. Mittels rechtem Mausklick auf das zu instanziierende Prozessmodell öffnet man dessen Menü . Durch Auswahl des Menüpunktes `Exemplar erstellen` wird eine neue Instanz erzeugt. Durch Auswahl des Menüpunktes `Exemplar erstellen und starten` wird die erzeugte Instanz sofort gestartet und die erzeugten `Workitems` in den entsprechenden Arbeitslisten angezeigt.

#### 3.1.3.3 Prozeßexemplarlisten

In dieser Liste werden alle Instanzen von Prozessmodellen, über die die Person entsprechende Rechte besitzt, angezeigt.
Folgende Aktionen sind mit Prozessexemplaren möglich [IBM99d]:

- Starten eines Prozessexemplars
- Stoppen eines bereits aktiven Prozessexemplars
- Wiederaufnahme eines ausgesetzten Prozessexemplars
- Erneutes Starten eines abgeschlossenen oder beendeten Prozessexemplars
- Bestimmen des Prozessexemplar-Status

- Überwachen des Fortschritts eines Prozessexemplars

#### 3.1.3.3.1 Arbeiten mit Instanzen

Durch einen Doppelklick auf eine Instanz kann diese, wenn Sie im Zustand `Bereit` ist, gestartet werden. Die dadurch neu erzeugten *Workitems* werden dann in die Arbeitslisten der Personen gestellt, denen sie zugeordnet wurden. Wenn ein *Workitem* für automatisches Starten definiert wurde (Aktion `erstellen und starten`), wird es unverzüglich bzw. sobald ein zugeordneter Benutzer sich angemeldet hat, gestartet. Die Aktionen `aussetzen` und `beenden` sind nur bei einer aktiven Instanz möglich. Ist eine Instanz `fertiggestellt` oder `beendet`, kann sie `gelöscht` werden. Eine graphische Darstellung der möglichen Zustände und Zustandsübergänge findet sich in Abbildung 3.8.

#### 3.1.3.3.2 Überwachen von Instanzen

Durch Monitoring kann der Fortschritt einer Instanz graphisch überwacht werden. Zur Verwendung dieser Komponente sind folgende Schritte nötig:

1. Klick mit der rechten Maustaste auf die Instanz, deren Verlauf beobachtet werden soll.
2. Auswahl der Aktion `Überwachen`.

In einem neuen Fenster wird das Prozessmodelldiagramm der Instanz so angezeigt, wie es in der Buildtime erstellt wurde. Die Überwachung zeigt die Aktivitäten und deren Status an (siehe Abbildung 3.7). Mögliche Zustände einer Aktivität werden in nachfolgenden Abschnitt dargestellt. Eine Liste der Symbole für den Status und deren Bedeutung können über die Online-Hilfe abgerufen werden.

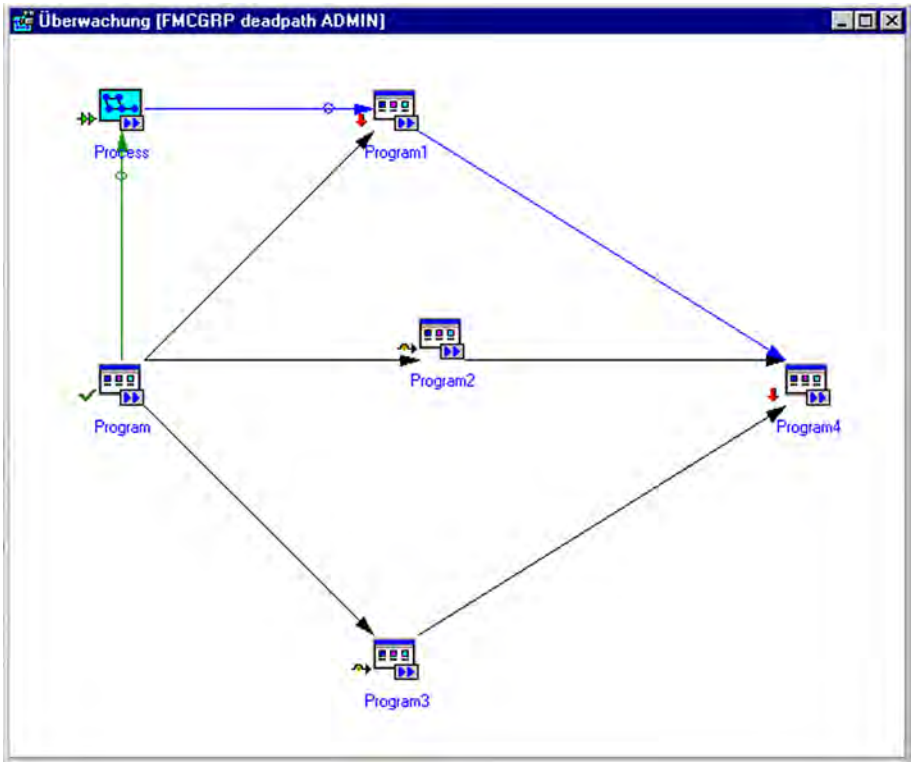



Abbildung 3.7: graphische Überwachung von Instanzen in MQSeries WF

#### 3.1.3.3.3 Ausführungssemantik von Instanzen

In Abbildung 3.8 sind alle möglichen Zustände von Prozessinstanzen und erlaubte Zustandsübergänge graphisch dargestellt.

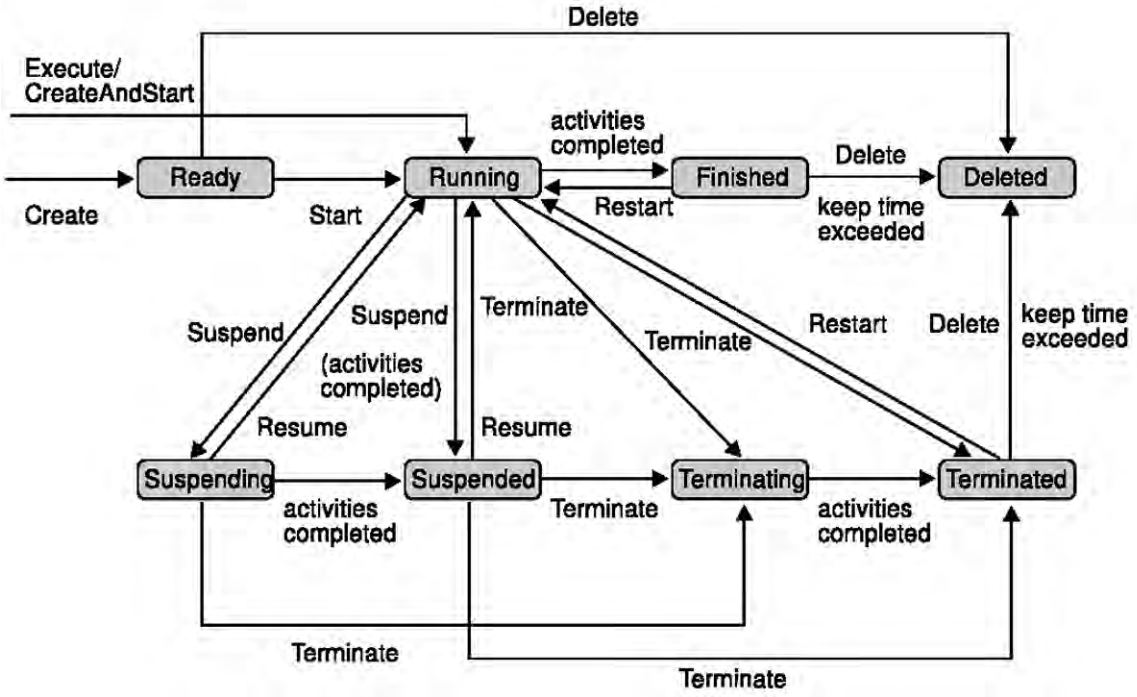



Abbildung 3.8: Ausführungszustände von Prozessinstanzen in MQSeries WF [IBM99e]

#### 3.1.3.4 Arbeitslisten

In MQSeries WF wird eine Aktivität, die von einer bestimmten Person ausgeführt werden muß, als `Vorgang` bezeichnet [IBM99b]. Dies entspricht einem *Workitem*. Im weiteren wird nur noch der Begriff *Workitem* verwendet.
Folgende Aktionen sind mit Arbeitslisten[3] ausführbar [IBM99d]:

- Filter- und Sortierkriterien für Arbeitslisten auswählen
- Arbeitslisten erstellen und löschen
- *Workitems* in einer Arbeitsliste starten
- Eine Statusänderung für ein *Workitem* erzwingen (z.B. Aussetzen eines *Workitems*)
- Mit *Workitems* und Prozessen arbeiten, zu denen Mitteilungen (wegen Zeitüberschreitung) gesendet wurden
- Abgeschlossene *Workitems* löschen
- Fortschritt von *Workitems* innerhalb einer Instanz überwachen

Wie Arbeitslisten und Filterkriterien erstellt werden, wurde bereits erörtert. Sortierkriterien (z.B. nach Name aufsteigend) werden beim Erstellen einer neuen Arbeitsliste angegeben.

##### 3.1.3.4.1 Arbeiten mit Workitems

Nach dem Start einer Prozessinstanz werden für alle Start-Aktivitäten, dies sind in MQSeries WF alle Aktivitäten ohne eingehende Kontrollkonnektoren, entsprechende Einträge (*Workitems*) in den Arbeitslisten der potentiellen Bearbeiter erzeugt. Ein solches *Workitem* befindet sich dann im

[3] In der Kopfzeile eines Arbeitslistenfensters steht "`Vorgänge [`*Systemgruppe Arbeitslistenname User-Id*`]`" (siehe Abbildung 3.5).

Zustand `bereit` (*ready*).
Ein *Workitem* kann auf zwei Arten gestartet werden:

1. Durch Doppelklick auf das *Workitem*
2. Durch Auswahl ein oder mehrerer *Workitems* und anschließender Auswahl von `Start` im Menüleistenpunkt `Aktivität`.

Für Aktivitäten mit Startmodus *automatic* geht das zugehörige *Workitem* direkt nach seiner Erstellung in den Zustand *running* über.
Die für ein *Workitem* möglichen Aktionen sind von seinem aktuellen Zustand abhängig. Durch Rechtsklick auf das *Workitem* etwa werden die aktuell zulässigen Aktionen bezogen auf das *Workitem* aufgelistet (siehe Abschnitt 3.1.3.4.2).

**Übertragen von Workitems**
Damit ein *Workitem* von einer Person, die die damit verknüpfte Anwendungsfunktion nicht ausführen darf, bearbeitet werden kann, muß es an diese Person übertragen werden. Ein *Workitem* kann nur von einer Person mit entsprechenden Rechten übertragen werden. Diese Rechte werden in der Buildtime-Komponente vergeben. Um die Berechtigungen einer Person zur Runtime einsehen zu können, muß sie im Fenster `Benutzermerkmale` das Register `Berechtigung` aufrufen. Dort sind unter "`Zugriff auf Aktivitäten von`" alle Personen angegeben, deren *Workitems* eingesehen und übertragen werden dürfen. Eigene *Workitems* dürfen diesen Personen übertragen werden.
Nach Auswahl von `übertragen` im Menü eines *Workitem* erscheint das Fenster aus Abbildung 3.9.

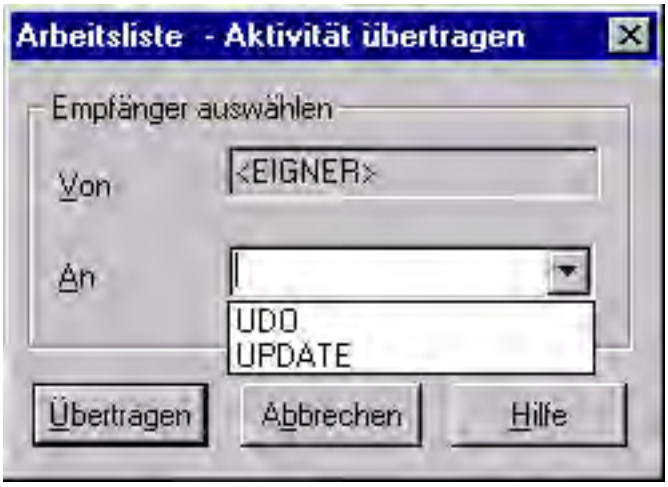



Abbildung 3.9: Übertragen von *Workitems* im Zustand `bereit` oder `ausgesetzt`

In der Auswahlliste `An` sind bereits alle Personen, für die entsprechende Berechtigungen existieren, eingetragen. Soll ein fremdes *Workitem* in die eigene Arbeitsliste übertragen werden, so muß der eigene Name angegeben werden.

#### 3.1.3.4.2 Statusänderung eines Workitems
Der Benutzer kann den Status eines *Workitems* ändern und dadurch in den Arbeitsablauf eingreifen. Folgende Statusänderungen sind möglich (im Auswahlmenü eines *Workitems* verfügbar):

- `Neustart` eines *Workitems*
- `Abschließen` eines *Workitems* erzwingen
- `Beenden` eines *Workitems*

Es sind nicht immer alle genannten Statusänderungen bei einem *Workitem* zulässig (z.B. kann `Abschließen erzwingen` eines *Workitems* im Status `InError` nicht ausgeführt werden). Alle in einem bestimmten Zustand eines *Workitems* zulässigen Statusänderungen sind aus den Abbildungen 3.10, 3.11 und 3.12 ersichtlich.

#### 3.1.3.4.3 Ausführungssemantik von Workitems

Die folgenden Diagramme geben Auskunft über die möglichen Zustände der *Workitems* und der im jeweiligen Zustand erlaubten Aktionen. Die Person muß über die entsprechenden Rechte verfügen. Die Aktionen und Zustände sind vom Zustand der zugehörigen Prozessinstanz abhängig.

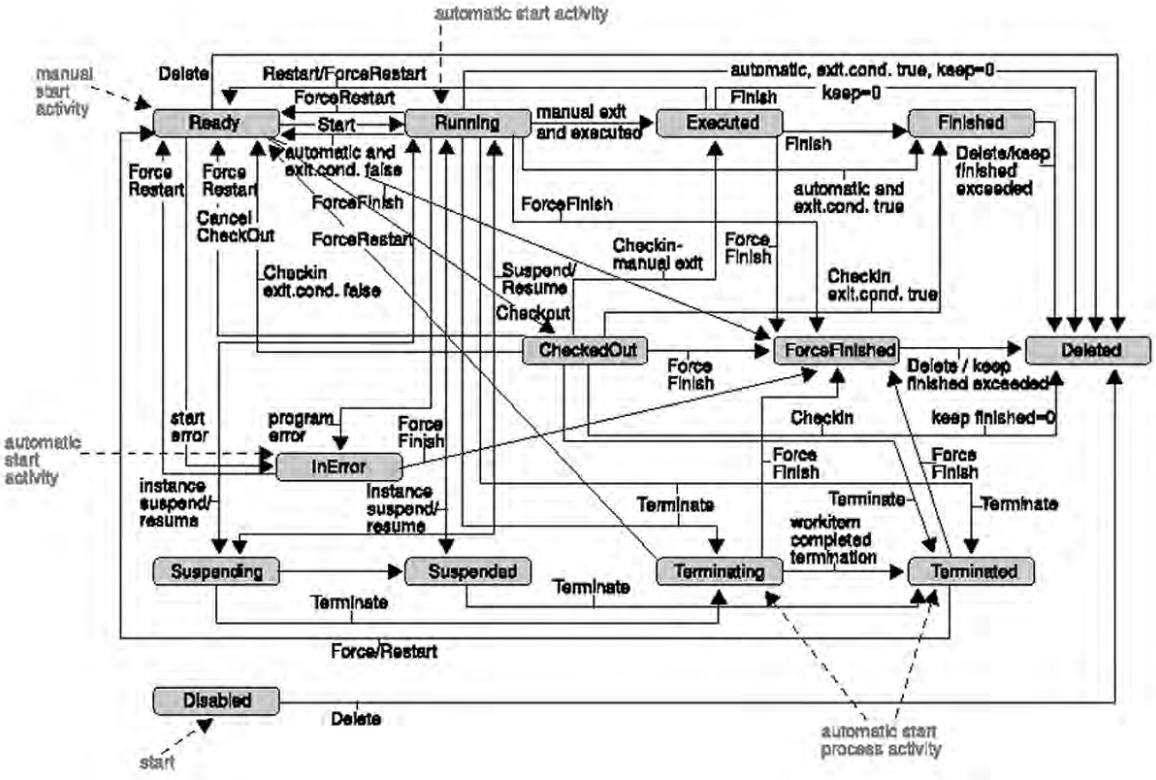



Abbildung 3.10: Ausführungszustände von *Workitems* in MQSeries WF - Prozessinstanz im Zustand `running` [IBM99e]

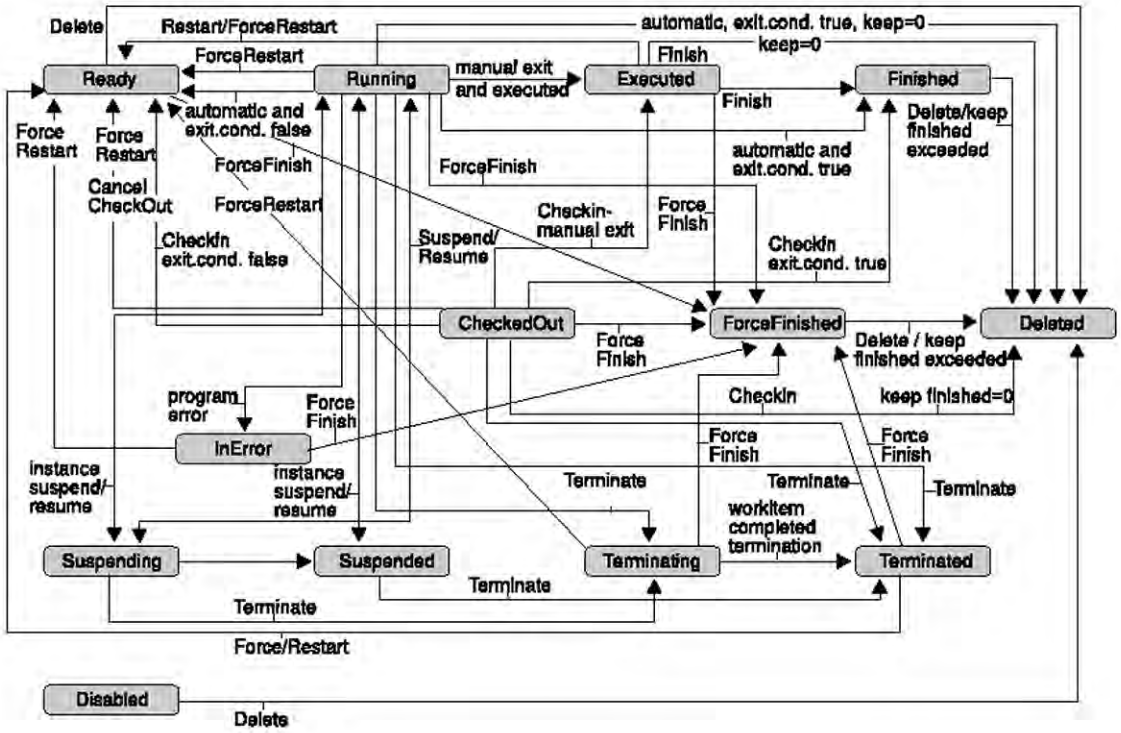



Abbildung 3.11: Ausführungszustände von *Workitems* in MQSeries WF - Prozessinstanz im Zustand `suspended` [IBM99e]

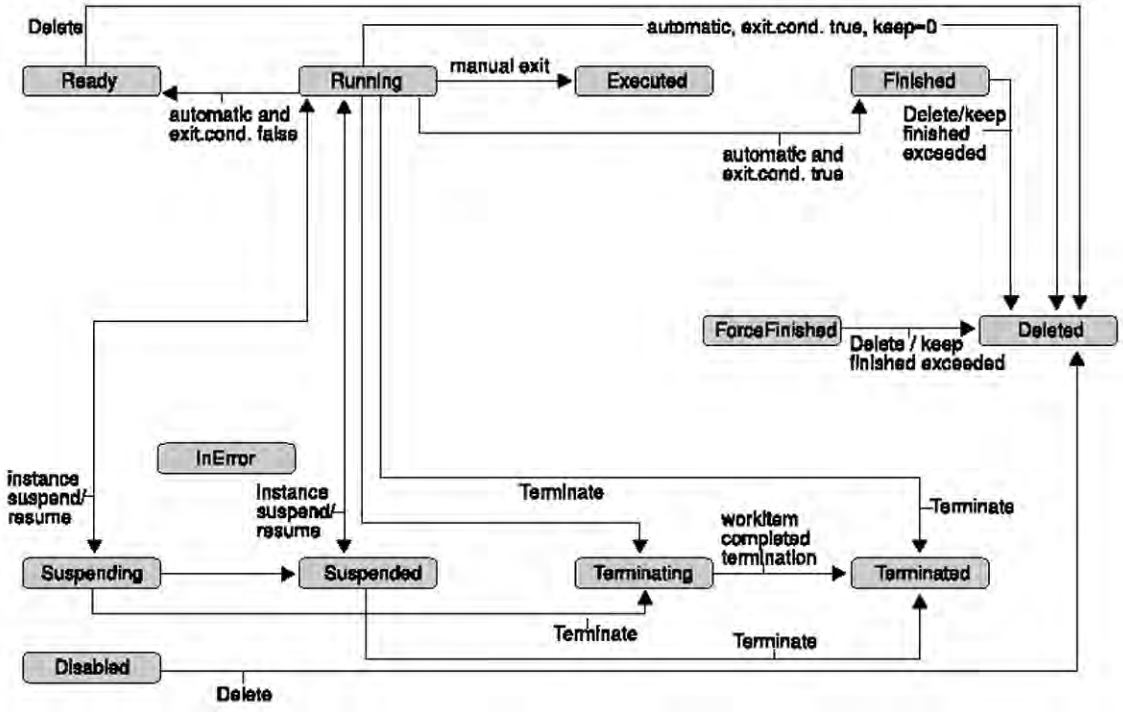



Abbildung 3.12: Ausführungszustände von *Workitems* in MQSeries WF - Prozessinstanz im Zustand `terminated` [IBM99e]

### 3.1.4 Kommunikation Client - Server

Zur Kommunikation zwischen einer Client-Komponente und den Server-Komponenten gibt es drei Varianten ([IBM99b]):

- MQSeries (unter Verwendung der Client-Nachrichtenebene MQ-Workflow)
- CORBA (*Common Object Request Broker Architecture*) IIOP (*Internet Inter-ORB Protocol*)
- RMI (*Remote Method Invocation*) IIOP (*Internet Inter-ORB Protocol*), für Java-Clienten.

In den folgenden drei Abschnitten werden diese Kommunikationsverfahren kurz vorgestellt.

#### 3.1.4.1 MQSeries Nachrichtenwarteschlangen

MQSeries ist das auf Nachrichtenschlangen (*Message Queues*) basierende Kommunikationssystem von IBM. Es wird hier verwendet, um die Kommunikation zwischen MQSeries WF-Servern und zwischen MQSeries WF-Server und Client zu verwalten. Dabei erhalten alle Komponenten des Systems ihre Nachrichten aus der von MQSeries verwalteten Eingangs-Warteschlange.
Auf der Server-Seite wird die Funktionalität vom `Queue Manager` und auf der Client-Seite vom `MQI-Client` (*Message Queue Interface*) erbracht:

- `Queue Manager`: Er ist für den Transport der Nachrichten hin bzw. weg von den Warteschlangen verantwortlich. Das Versenden und Empfangen der Nachrichten im Netzwerk wird serverseitig ebenfalls durch den Queue Manager durchgeführt.
- `MQI-Client`: Er empfängt Nachrichten aus dem Netzwerk und sendet sie an die Client-Komponente. Über das MQI ist es Programmen möglich, Dienste des `Queue Managers` abzufragen.

In Abbildung 3.13 ist beispielhaft die Kommunikation zwischen einer Client- und einer Serverkomponente dargestellt.
Die Client-Komponente (z.B. das Verwaltungsprogramm) verlangt einen Dienst von einer Server-Komponente (z.B. der Verwaltungs-Server). Dazu legt sie eine Nachricht in die Eingangs-Warteschlange (`Queue 1`) des Verwaltungs-Servers(1). Nachdem der Verwaltungsserver die Nachricht aus der Nachrichtenwarteschlange gelesen hat (2), verarbeitet er die Anfrage anhand der in der Nachricht mitgelieferten Information. Will der Verwaltungs-Server eine Antwort zurückschicken (z.B. eine Bestätigung), so legt er diese Nachricht in die Eingangs-Warteschlange der Client-Komponente (`Queue 2`) (3). Die Client-Komponente kann mittels MQI ihre *Client Input Queue* überprüfen (4).

#### 3.1.4.2 CORBA IIOP

CORBA ist die Standardarchitektur verteilter Objekte der OMG (*Object Management Group*) [OMG01a]. Es werden Schnittstellen zum Aufruf von Methoden entfernter Objekte in der plattformunabhängigen Schnittstellenbeschreibungssprache IDL (*Interface Definition Language*) spezifiziert. Abbildungen der IDL auf eine spezielle Programmiersprache werden durch eine Bindung dieser Sprache an CORBA/IIOP implementiert. Die CORBA/IIOP Implementierung von JDK ist z.B. unter dem Namen `Java IDL` bekannt. In [SUN01d] wird ein Überblick über die CORBA/IIOP-Programmierung mit Java gegeben.

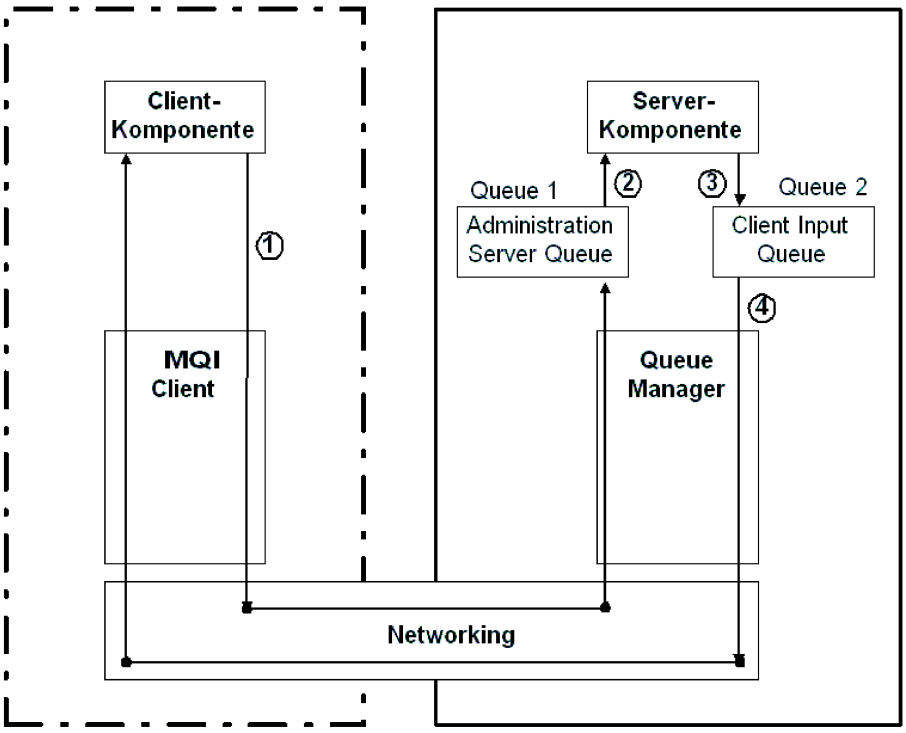



Abbildung 3.13: Kommunikation zwischen Client und Server mittels MQSeries

**ORB**
Eine wichtige Komponente von CORBA ist der `ORB` (*Object Request Broker*). Er übernimmt alle Aufgaben im Routing einer Anfrage vom Clienten zum Objekt und das Routing der Antwort zurück zum Client. In Abbildung 3.14 wird der Aufruf eines entfernten Objektes durch einen Client graphisch verdeutlicht.

**IIOP**
IIOP ist das von CORBA verwendete Kommunikationsprotokoll. Es basiert auf der CORBA-Spezifikation der OMG [IBM01b]. Durch IIOP wird festgelegt, auf welche Art und Weise die Datenbits zwischen einen CORBA-Client und einem CORBA-Server transportiert werden.

#### 3.1.4.3 RMI IIOP

Mit `RMI` (*Remote Method Invocation*) können verteilte Programme in der Programmiersprache Java geschrieben werden. Die Verwendung von RMI ist einfach, da keine separate IDL benutzt werden muß. Zur Kommunikation mit einem entfernten Java-Objekt wird JRMP (*Java Remote Method Protocol*) verwendet. Nachteile von RMI ([IBM01a]) sind

- die fehlende Unterstützung anderer Programmiersprachen
- die Unfähigkeit mit CORBA-Objekten zu kommunizieren, da ein nicht standardisiertes Kommunikationsprotokoll verwendet wird.

Das Kommunikationsprotokoll IIOP wird in Abschnitt 3.1.4.2 besprochen. Durch `RMI-IIOP` kann ein RMI-Objekt das Protokoll IIOP verwenden und mit CORBA-Objekten kommunizieren. Dadurch wird die einfache Handhabung von RMI mit CORBAs Vielfältigkeit an verwendbaren Programmiersprachen kombiniert.
RMI-IIOP wird von MQSeries WF ab der Version 3.2.2 und Servicepack 3 unterstützt.

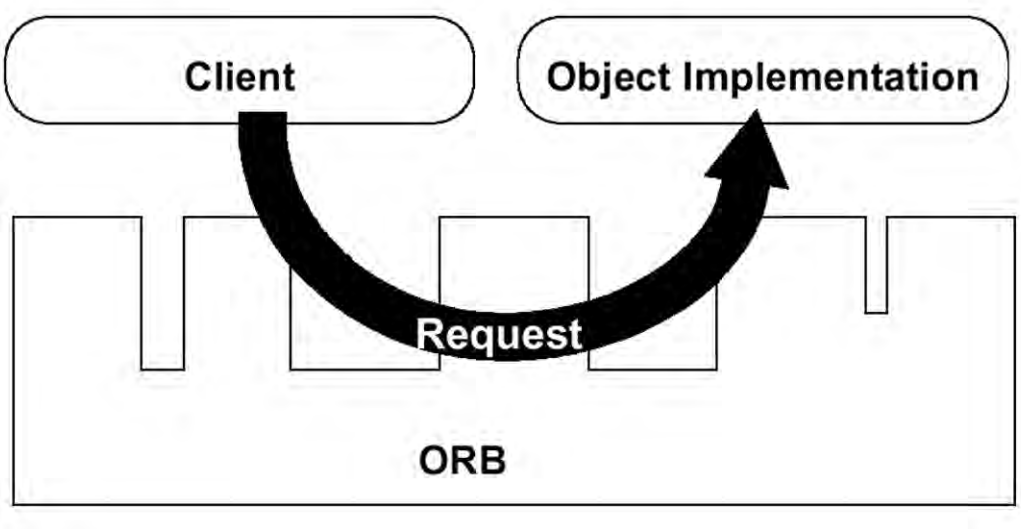



Abbildung 3.14: Verlauf einer Anfrage durch den ORB [OMG01]

### 3.1.5 Datenspeicherung

Als Runtime-Datenbank kann in MQSeries WF, anders als bei der Buildtime-Datenbank, nur die `DB2 (R) Universal Database` verwendet werden. Grund hierfür liegt vermutlich in der intensiven Nutzung von DB2-`Stored Procedures`, welche aus Effizienzgründen verwendet werden. Die Runtime-Datenbank dient zur Speicherung von:

- Prozessmodellen
- prozessrelevanten Daten (WF-Status, Werte von Datencontainern, usw.)

Zu den prozessrelevanten Daten, die zur Prozessausführung benötigt werden, gehört beispielsweise der Status von Prozessaktivitäten. Außerdem werden die in den Eingabe- und Ausgabecontainer übertragenen Daten während der Prozessausführung persistent gespeichert.
Darüberhinaus wird die Runtime-Datenbank in der navigationsbezogenen Logik zwischen den einzelnen Aktivitäten eines Prozessmodells verwendet. Dies erfolgt über SQL-Aufrufe [IBM99b].

## 3.2 Staffware

Die Runtime-Komponente von Staffware 2000 verfügt ebenfalls über eine dreistufige Architektur. Ein Überblick wird in den folgenden Abschnitten vermittelt.

### 3.2.1 Architektur und Plattformen

Abbildung 3.15 gibt einen Überblick über die Runtime-Komponente. Es existiert eine Client-Komponente, eine Server-Komponente und eine Datenspeicherungs-Komponente.

- `Client-Komponente`: Ein Client stellt die graphische Benutzerschnittstelle zum System dar. Neben dem Standard-Client (*Out of the Box Win 32 Client for Microsoft*) können für verschiedene Betriebssysteme angepaßte Clients mit dem `Staffware Enterprise Objects` (*SEO*) programmiert werden oder *Microsoft Exchange*, *Lotus Notes*, *Outlook*, *Genesis2* und *MEMO* als Externe-Clients verwendet werden (siehe Kapitel 5). Als Externe-Clients werden externe Systeme bezeichnte, die um Staffware-Funktionalität erweitert werden können.

- `Server-Komponente`: Der Server und seine Komponenten verwalten und überwachen die Ausführung von Prozessmodellen.
- `Datenspeicherungs-Komponente`: Sie enthält alle arbeitsablauf- und verwaltungsrelevanten Daten. Es stehen das Filesystem des Servers, der Microsoft SQL Server und Oracle 8i zur Verfügung.

Jegliche Kommunikation zwischen den Komponenten findet mittels `Remote Procedure Call` (*RPC*) statt.

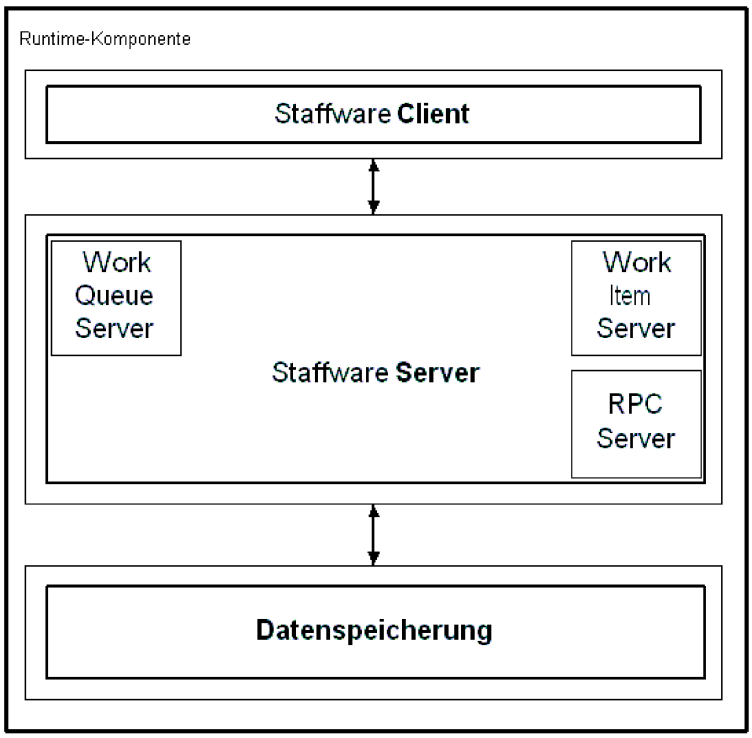



Abbildung 3.15: Architektur der Runtime-Komponente von Staffware 2000

Die oben beschriebenen Komponenten sind nicht auf allen Plattformen verfügbar. Unter den Oberbegriff `Unix` werden die Plattformen Sun Solaris, IBM AIX, HP-UX und Linux zusammengefaßt. Tabelle 3.2 stellt die Verfügbarkeit der Komponenten auf den verschiedenen Plattformen dar [SW01].

| Komponente | **UNIX** | **Microsoft 95/98** | **Windows NT** | **Windows 2000** |
|---|---|---|---|---|
| Client | Nein | Ja | Ja | Nein |
| Server | Ja | Nein | Ja | Ja |
| Datenspeicherung | | | | |
| Microsoft SQL Server | Nein | Ja | Ja | Ja |
| Oracle 8i | Ja | Ja | Ja | Ja |
| Filesystem | Ja | Ja | Ja | Ja |

Tabelle 3.2: Plattformen für die Staffware Runtime-Komponenten

**Multinode Konzept**
Das Kennzeichen eines Multinode-Konzeptes ist die Verknüpfung mehrerer Server in einem System. In Abbildung 3.16 wird eine Architekturvariante dieses Konzeptes dargestellt [SW01b].

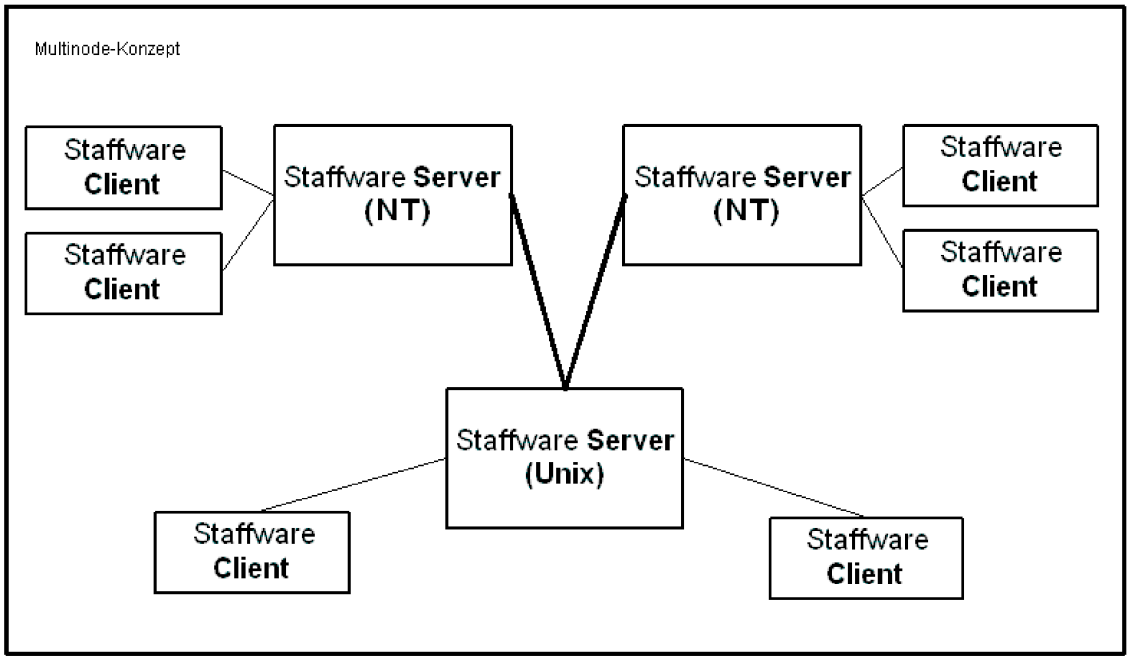



Abbildung 3.16: Das Mulitnode-Konzept von Staffware

Die Server stehen aus der Sicht einer Prozessinstanz in einer Master/Slave-Beziehung. Das bedeutet, daß immer nur ein Server die Verantwortung für die Ausführung einer Prozessinstanz trägt. Dieser Server wird als Master bezeichnet.
Die größten Vorteile des Mulitnode Konzeptes bestehen

- in der möglichen Verwendung mehrerer Server verschiedener Versionen und Plattformen.
- in der einfachen Erstellung standortübergreifender Architekturen.

Jeder Staffware-Client kann sich nur an einem Staffware-Server anmelden. Natürlich können auf einem Rechner mehrere Staffware-Clients laufen, so daß diese unterschiedliche Staffware-Server konnektieren.

### 3.2.2 Server-Komponente

Der Staffware-Server erfüllt folgende Aufgaben [SW00d]:

- Verarbeiten der freigegebenen *Workitems*
- Aktualisieren der System-Dateien
- Backup und Wiederherstellung
- Ausführen automatischer Schritte
- Überwachen von *Deadlines*
- Löschen abgeschlossener Instanzen

Die Serverfunktionalität wird in Windows NT durch zwei Dienste erbracht:

- **Staffware** *server* **Background** service
- **Staffware** *server* **Process Invocator** service

Dabei steht *server* für den Namen des Staffware-Servers.
Das `Starten` und `Stoppen` des Staffware-Servers ist durch diese Implementierung, wie bei normalen NT-Diensten, möglich. Somit kann eine bei der Installation gesetzte Startart des Staffware-Servers jederzeit geändert werden. Es stehen die für NT-Dienste üblichen Varianten `automatisch` und `manuell` zur Verfügung.

Weitere Funktionalitäten werden durch Staffware-Server-Prozesse ausgeführt. Dazu gehört die Steuerung der Kommunikation sowie die Verwaltung der Arbeitslisten und ihrer Workitems. Diese Aufgaben werden von verschiedenen Server-Komponenten ausgeführt:

- Remote Procedure Call Server (*RPC-Server*)
- Work Queue Server (*WQS*)
- Work Item Server (*WIS*)

Die Funktionsweise des `RPC-Server` wird im Abschnitt 3.2.4 genauer besprochen. Die Funktionalität des `WQS`, der durch den `RPC-Server` gestartet wird, und des `WIS` wird im folgenden erklärt.

- `RPC-Server`: Ein Client meldet sich am Staffware-Server an, indem er den RPC-Server konnektiert. Die dazu nötigen Informationen erhält der Client vom Portmapper.
- `WQS`: Er überwacht, welche Personen- und Gruppenarbeitslisten eine Person in einem Staffware Client angezeigt bekommt und welcher WIS die Liste verwaltet. Dieser Prozess wird durch das Programm *SWDIR*`/rpc/wqsrpc.exe` ausgeführt. Dabei steht *SWDIR* für das Verzeichnis des Staffware-Servers. Es existiert immer nur ein `wqsrpc`-Prozess.
- `WIS`: Ein WIS-Prozess ist eine Instanz eines WIS und beinhaltet Arbeitslisten, die ihm der WQS zugewiesen hat. Er verwaltet die Darstellung der *Workitems* in Arbeitslisten. Jede Arbeitsliste in Staffware wird von einem WIS-Prozess verwaltet und jeder WIS kann mehr als eine Arbeitsliste verarbeiten. Staffware beinhaltet standardmäßig fünf WIS. Die Anzahl der `wisrpc`-Prozesse wird über den Parameter *WQS_WIS_COUNT* in *SWDIR*`/rpc/staffcfg` konfiguriert. Dieser Prozess wird durch das Programm *SWDIR*`/rpc/wisrpc.exe` ausgeführt.

**Zuordnung der Arbeitslisten zu WIS-Prozessen**
Die Zuordung der Arbeitslisten erfolgt durch den WQS. Er durchläuft nach dem Alphabet die Liste der Personen und Gruppen und ordnet diese einem einzelnen WIS-Prozess zu. Die Zuordnung der Arbeitslisten erfolgt auf eine von zwei Arten:

- `Round Robin Queue Allocation`: Der WQS ordnet abwechselnd jedem WIS-Prozess eine Arbeisliste (zuvor alphabetisch geordnet) zu, bis alle Arbeitslisten zugeteilt worden sind. Bei 5 WIS und den Arbeitslisten A-O ergäbe sich folgende Zuordnung:
  Queues A,F,K to WIS 1
  Queues B,G,L to WIS 2
  Queues C,H,M to WIS 3
  Queues D,I,N to WIS 4
  Queues E,J,O to WIS 5
- `On-Demand Queue Allocation`: Bei dieser Art der Zuordnung werden die Arbeitslisten ebenfalls alphabetisch zugeordnet, allerdings dem ersten verfügbaren WIS. Wird einem WIS eine große Arbeitsliste zugeordnet, dauert es einige Zeit, bis er wieder bereit ist weitere Arbeitslisten zu akzeptieren. Dadurch können WIS-Prozesse mit kleineren Arbeitslisten eine größere Anzahl an Listen akzeptieren. Nachteilig ist, daß eine große Arbeitsliste nicht zwangsläufig viele Anfragen erhalten muß. Somit kann eine schlechte Lastverteilung entstehen. Mehr Details zur Lastverteilung findet sich in Kapitel 6.

### 3.2.3 Standard-Client

Der Staffware Standard-Client, welcher von Staffware als *Out of the Box Win 32 Client for Microsoft* bezeichnet wird, existiert für alle Microsoft-Plattformen. Mit dem Standard-Client können Prozessmodelle gestartet und überwacht, Arbeitslisten verwaltet und Zugriffsrechte vergeben werden.

#### 3.2.3.1 Starten und Einrichten des Standard-Clients

Der Standard-Client kann unter Windows NT durch Auswahl von `Staffware Client` im Programm-Menü `Staffware` gestartet werden. Es erscheint das Fenster zur Anmeldung. Hier muß eine gültige Benutzer-ID mit zugehörigem Passwort eingegeben werden. Während der Installation wurden bereits die drei Personen `swpro`,`swadmin` und `swuser` angelegt. Existieren mehrere Staffware-Server, so muß geprüft werden, ob der richtige Staffware-Server in der drop-down Liste `Server` ausgewählt ist. Im Anschluß an ein erfolgreiches Login öffnet sich das Fenster des Standard-Client (Abbildung 3.17).

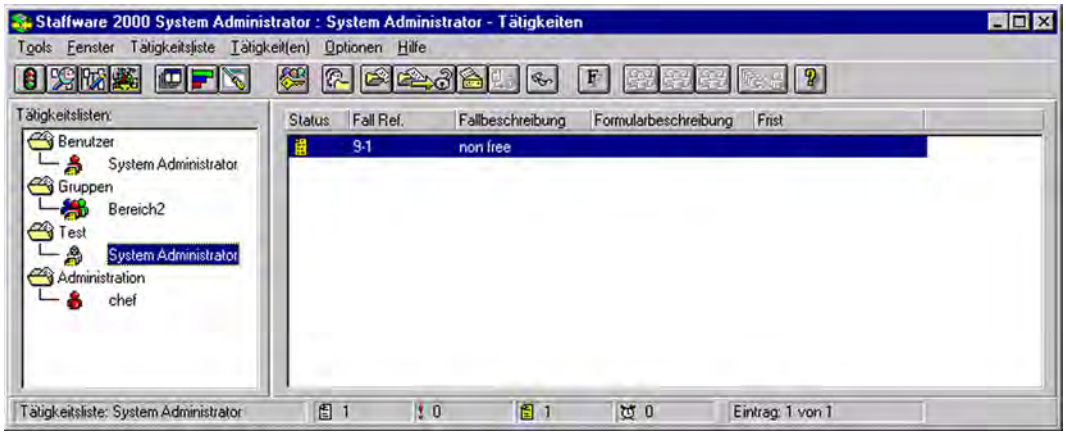



Abbildung 3.17: Der Standard-Client von Staffware

Es umfaßt zwei Bereiche, eine Übersicht zu allen *Tätigkeitslisten* (=Arbeitslisten) und die Darstellung der *Workitems* einer selektierten Liste (linker und rechter Bereich). Hinzu kommen je eine Status-,Menü- und Werkzeugleiste.

#### 3.2.3.2 Übersicht zu Arbeitslisten eines Benutzers

Im linken Bereich des Fensters aus Abbildung 3.17 werden alle Arbeitslisten, für die der angemeldete Benutzer über entsprechende Rechte verfügt, angezeigt. Um Arbeitslisten besser finden zu können, werden diese in folgende Kategorien aufgeteilt.

Die Kategorie `Benutzer-Arbeitslisten` enthält die einer Person direkt zugeordneten Arbeitslisten. Auf diese Arbeitslisten kann nur der angemeldete Benutzer zugreifen.

Die Kategorie `Gruppen-Arbeitslisten` enthält alle Arbeitslisten von Gruppen, der die Person angehört.

Die Kategorie `Test-Arbeitslisten` kann Arbeitslisten (wie links angedeutet) für Personen und Gruppen enthalten. Auf Arbeitslisten dieser Kategorie erhalten nur Personen Zugriff, die über Rechte zur Erstellung von Prozessmodellen verfügen und die ein noch nicht freigegebenes Prozessmodell zu Testzwecken gestartet haben.

Die Kategorie `Administration-Arbeitslisten` enthält alle Arbeitslisten von Personen und Gruppen, für die die Person über Supervisor-Rechte verfügt (siehe Kapitel 2).

Statusänderungen einzelner Arbeislisten werden dem Benutzer graphisch angezeigt, indem die oben dargestellten Symbole "erweitert" werden. In Tabelle 3.3 sind die existierenden Darstellungen für die `Benutzer-`, `Gruppen` und `Test`-Arbeitslisten aufgeführt.

| | |
|---|---|
| `gelbe Seite` | Die Arbeitsliste enthält ein noch nicht geöffnetes Workitem (siehe Benutzer-Arbeitsliste in 3.17). |
| `Uhr` | Die Arbeitsliste enthält mindestens ein Workitem mit festgelegter Deadline. |
| `rotes Ausrufezeichen` | Die Arbeitsliste enthält mindestens ein Workitem mit gesetzter Priorität. |
| `gelber Pfeil` | Diese Arbeitsliste (nur bei `Administration-Arbeitsliste` möglich) verfügt über eine aktive Weiterleitung. *Workitems* dieser Arbeitsliste werden direkt an eine vom Administrator festgelegte Arbeitsliste weitergeleitet. |

Tabelle 3.3: Visuelle Darstellung von Statusänderungen einer Arbeitsliste

Blinkt eines dieser Symbole, so hat sich seit der letzten Einsicht in die Arbeitsliste etwas geändert (z.B. Hinzunahme neuer Workitems oder Überschreiten einer Deadline).

#### 3.2.3.3 Darstellung einer einzelnen Arbeitsliste (Workitem-Liste)

Die `Workitem-Liste` ist nach dem Start des Standard-Clients leer. Erst nach Auswahl einer Arbeitsliste im linken Fensterbereich werden deren *Workitems* angezeigt. Standardmäßig werden für jedes *Workitems* die folgenden Daten dargestellt (siehe Abbildung 3.17):

- Status
- Fall-Referenz
- Fallbeschreibung
- Formularbeschreibung
- Frist

Der Status eines Workitems wird durch die in Tabelle 3.4 aufgeführten Symbole graphisch dargestellt.

| | |
|---|---|
| `gelbe Seite` | Das Workitem wurde seitdem es in der Arbeitsliste angezeigt wird noch nicht geöffnet. Zusätzlich wird der zugehörige Text in der Workitem-Liste blau angezeigt. |
| `weiße Seite` | Das Workitem wurde bereits geöffnet, aber wieder zurückgestellt. |
| `Briefumschlag` | Dieses Workitem kann direkt aus der Arbeitsliste freigegeben werden. Eine Bearbeitung durch den Benutzer ist nicht nötig. |
| `Schloss` | Das Workitem ist gesperrt und der Benutzer kann nicht mehr darauf zugreifen. Dies ist normalerweiser der Fall, wenn das Workitem bereits von einer anderen Person geöffnet wurde. Zusätzlich wird der zugehörige Text in der Workitem-Liste grau angezeigt. |
| `weiße Seite mit rotem Kreuz` | Das Workitem ist nicht mehr verfügbar. Dies ist der Fall wenn das Workitem bereits von einem anderen Benutzer freigegeben wurde. |

Tabelle 3.4: Visuelle Darstellung des Status eines Workitem

In Kombination mit diesen Symbolen können noch ein `rotes Ausrufezeichen` (Workitems mit Priorität) und ein `gelber Pfeil` (weitergeleitetes Workitem) auftreten.
Die Darstellung der Workitem-Liste kann auch graphisch erfolgen (siehe Abbildung 3.18).

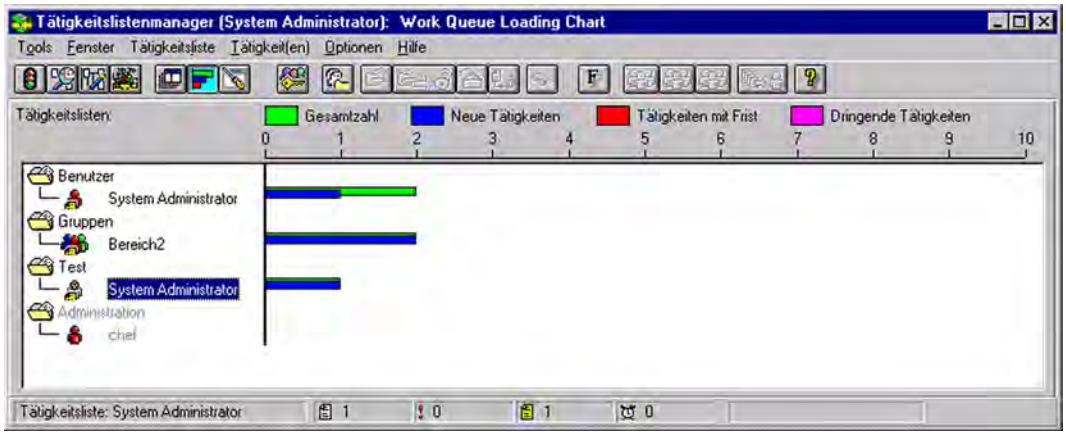



Abbildung 3.18: Der Standard-Client von Staffware mit graphischer Darstellung der Workitem-Liste

Es besteht die Möglichkeit, *Workitems* aus der eigenen Arbeitsliste an eine andere Arbeitsliste weiterzuleiten. Dazu müssen zwei Bedingungen erfüllt sein:

1. Der Prozessmodellierer muß bei Erstellung des Prozessmodells für eine Aktivität festgelegt haben, daß zugehörige *Workitems* weitergeleitet werden können.

2. Eine Person, die ein *Workitem* weiterleiten will, muß über besondere Rechte verfügen (siehe Rechtevergabe Kapitel 2).

Sind diese beiden Bedingungen erfüllt, kann eine Person ein in der Workitem-Liste markiertes *Workitem* (oder mehrere markierte *Workitems* ) weiterleiten. Nach der Auswahl von `weiterleiten` im Menüpunkt `Optionen` erscheint das Fenster `Workitem weiterleiten`. In diesem Fenster werden mögliche Arbeitslisten zur Weiterleitung angezeigt. Es kann eine Arbeitsliste ausgewählt werden, an die das *Workitem* weitergeleitet werden soll.

#### 3.2.3.4 Statusleiste

Die `Statusleiste` des Standard-Client unterteilt sich in sechs Bereiche (siehe Abbildung 3.18). In diesen Bereichen wird von links nach rechts folgendes angezeigt:

- Name der ausgewählten Arbeitsliste
- Anzahl der *Workitems* in der ausgewählten Arbeitsliste
- Anzahl der *Workitems* mit Priorität in der ausgewählten Arbeitsliste
- Anzahl der ungeöffneten *Workitems* in der ausgewählten Arbeitsliste
- Anzahl der *Workitems* mit Deadline in der ausgewählten Arbeitsliste
- Eintragsnummer des ausgeählten *Workitem* (siehe Abbildung 3.17)

#### 3.2.3.5 Menü- und Werkzeugleiste

Die Werkzeugleiste stellt einen direkten Zugriff auf einige wichtige Optionen des Menüs dar. Die Optionen, die einem Benutzer im Standard-Client angeboten werden, sind von seinen Rechten und vom aktuellen Kontext (Markierung von Arbeitslisten bzw. *Workitems*) abhängig. Beispielsweise kann der `GWD` nicht von jedem Benutzer aufgerufen werden (siehe Abschnitt 2.1.3.1).
Alle möglichen Optionen sind in Tabelle 3.5 aufgelistet.

| **Menüpunkt** | **Optionen** |
|---|---|
| `Tools` | Prozess-Start, Prozesss-Administration, Aufruf des GWD, Administration Manager, EIS Report, RunStep, Arbeitslisten Monitor, Passwort ändern. |
| `Window` | neues Standard-Client-Fenster, graphische Darstellung der Workitem-Liste |
| `Queue` | Aktualisieren, Zugriffsberechtigung, Administration, Listen aktualisieren |
| `Workitem` | Öffnen, Nächstes Öffnen, Freigeben, Weiterleiten, Suchen |
| `Options` | Schrift, Fensteroptionen, Autorepeat Prozess-Start, Autorepeat Workitem öffnen |

Tabelle 3.5: Die Optionen der Menüleiste des Staffware-Clients

**Autorepeat-Optionen**
Die Wirkungsweisen der Optionen *autorepeat Prozess-Start* und *autorepeat Workitem öffnen* sind nicht direkt ersichtlich und werden deshalb kurz erläutert.
Ist die Option *autorepeat Prozess-Start* aktiv, so erscheint direkt nach dem erfolgreichem Start eines Prozessmodells wieder das Fenster zum Starten eines weiteren Prozessmodells. Diese Option erspart dem Bearbeiter beim Starten mehrerer Prozessmodelle je einen Arbeitsschritt.
Ist die Option *autorepeat Workitem öffnen* aktiv, so wird direkt nach der Freigabe eines *Workitems* (wenn das der zugehörigen Aktivität hinterlegte Formular beendet wurde) das nächste anstehende Workitem gestartet. Will ein Bearbeiter seine Arbeit beenden, muß er diese Option explizit deaktivieren.

#### 3.2.3.6 Zugriffsrechte

Zugriffsrechte können für die Arbeitslisten von Personen und Gruppen vergeben werden. Dabei kann es sich um eine `Vertreterregelung (Einsicht)` oder um eine `Weiterleitung der Workitems` handeln. Alle Arbeitslisten, für die eine Person Zugriffsrechte erstellen kann, sind im Arbeitslisten-Fenster in der Kategorie `Administration` aufgeführt. Existiert die Kategorie für eine Person nicht, so verfügt sie über keine `Supervisor`-Rechte (siehe Rechtevergabe Kapitel 2).

### 3.2.4 Kommunikation Client - Server

Die Kommunikation zwischen Client und Server findet in Staffware mittels `Remote Procedure Calls` (*RPC*) `over TCP/IP` und unter Verwendung von `Ports` statt ([SW00d] und [SW00g]). Das bedeutet, daß Clients ihre benötigten Informationen zur Darstellung der Arbeitslisten und Workitems durch Anfragen an den Server erhalten. Mittels `RPC` können auch Dienste auf fremden Rechnern in Anspruch genommen werden bzw. Programme auf fremden Rechnern ausgeführt werden. Eine Kommunikation mittels `RPC` kann aus folgenden Gründen vorteilhaft sein:

- Ein zentraler Dienst bzw. ein zentrale Programm kann vielen Clients zugänglich gemacht werden.
- Der lokale Rechner kann die gewünschte Funktionalität nicht erbringen.
- Die Fähigkeiten eines besonders leistungsfähigen Rechners soll genutzt werden.
- Die Arbeitslast kann im Netzwerk verteilt werden.

Jedes Programm, auf das über einen entfernten Aufruf mittels `RPC` zugegriffen werden kann, muß eine eigene und eindeutige RPC-Nummer besitzen. Die Vergabe kann statisch oder dynamisch erfolgen [Web98].
Im folgenden sollen die Verbindungsaufnahme, die Kommunikation zwischen Client und Server und die benötigen Hilfsprogramme erläutert werden.

#### TCP/IP

TCP *(Transmission Control Protocol)* und IP *(Internet Protocol)* sind Teile der Protokollfamilie des amerikanischen Verteidigungsministeriums. Die Kombination TCP/IP stellt eine verbindungsorientierte Kommunikation zwischen einem Server und einem Client sicher [Tan98]. Dies bedeutet, daß Daten erst nach einem erfolgreichen Verbindungsaufbau ausgetauscht werden können.
Um einen Verbindungsaufbau zwischen Client und Server sicherstellen zu können, muß der Staffware-Server auf einem Rechner mit statischer (fest zugewiesener) IP-Adresse installiert worden sein. Ist der Staffware-Server auf einem Rechner installiert, der seine IP-Adresse dynamisch von einem DHCP[4]-Server bezieht, können sich Benutzer nicht mit dem Staffware-Client am Staffware-Server anmelden [SW00g].

#### Der Portmapper

Die Verbindung zwischen einem RPC-Client und einem RPC-Server wird unter Zuhilfenahme des Programmes `NobleNet Portmapper for TCP` erstellt. Der `Portmapper` stellt ein Inhaltsverzeichnis über alle RPC-Dienste, auf dem Rechner auf dem er läuft, bereit. So kann eine Anwendung auf dem RPC-Rechner eine Verbindung zu einem RPC-Dienst anfordern, indem die Anwendung dem Portmapper die zugehörige RPC-Nummer übermittelt. Die Antwort des Portmappers enthält die aktuelle Port-Nummer. Jetzt kann der Client den Dienst direkt über seinen Port ansprechen.
Beim Start des Staffware-Servers werden mehrere RPC-Server gestartet. Diese erhalten ihre RPC-Nummer dynamisch und werden an die vom Betriebssystem dynamisch erzeugten Ports gebunden. Danach erfolgt die Registrierung der Beziehung zwischen RPC-Nummer und Port beim Portmapper.

#### Der RPC-Server (Staffware Client RPC listener)

Alle am Staffware-Server eingehenden RPC-Aufrufe werden vom RPC-Server überwacht. Er hat

[4]DHCP ist ein TCP/IP-Dienstprotokoll, daß dynamisch geleaste Konfigurationen von Host-IP-Adressen anbietet. DHCP bietet sichere, zuverlässige und einfache TCP/IP-Netzwerkkonfiguration und verhindert Adressenkonflikte.

eine vordefinierte (statisch während der Installation festgelegte) RPC-Nummer, deren Default-Wert 391875 ist. Der RPC-Server wird beim Hochfahren des Staffware-Server gestartet und erfragt dann sofort einen Port vom Betriebssystem. Dieser Port wird an einen Socket gebunden (siehe dazu [Web98]). Danach registriert er sich mit seiner RPC-Nummer und dem vom Betriebssystem zugewiesenen Port beim Portmapper. Zur Verbindungsaufnahme sind die folgenden Schritte nötig.

1. Der Client konnektiert den `Portmapper` (mit einer RPC-Nummer)
2. Der `Portmapper` antwortet mit dem Port, an dem der RPC-Server gebunden ist
3. Der Client konnektiert den RPC-Server mittels des erfragten Ports und erfragt Details über die RPC-Nummern anderer benötigter RPC-Server (z.B. WQS)
4. Der Client erhält Daten zur direkten Kommunikation mit anderen RPC-Servern

Sind diese vier in Abbildung 3.19 graphisch veranschaulichten Schritte erfolgreich bearbeitet, erfolgt die weitere Kommunikation auf direkten Weg. In Abbildung 3.20 ist die Kommunikation zwischen Client und WQS/WIS dargestellt.
Da sich an einen Staffware-Server mehrere Clients gleichzeitig anmelden können, muß der RPC-Server die Kommunikation mit mehreren Clienten sicherstellen können. Wieviele Verbindungen ein RPC-Server gleichzeitig verwalten kann, wird in der Datei `staffcfg` mit dem Parameter `MAX_USERS_PER_PROCESS` festgelegt. Der Standardwert dieses Parameters ist 20 [SW00d]. Dadurch ist die Anzahl an RPC-Servern in einem System abhängig von der Anzahl der angemeldeten Clienten und von diesem Konfigurationsparameter.

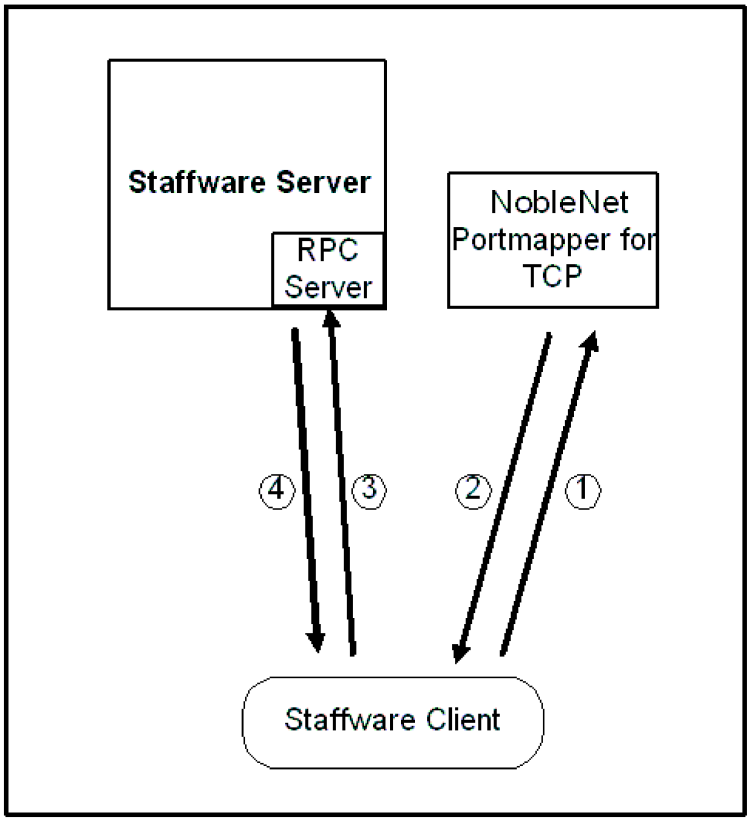



Abbildung 3.19: Kommunikationsaufbau zwischen Client und Server

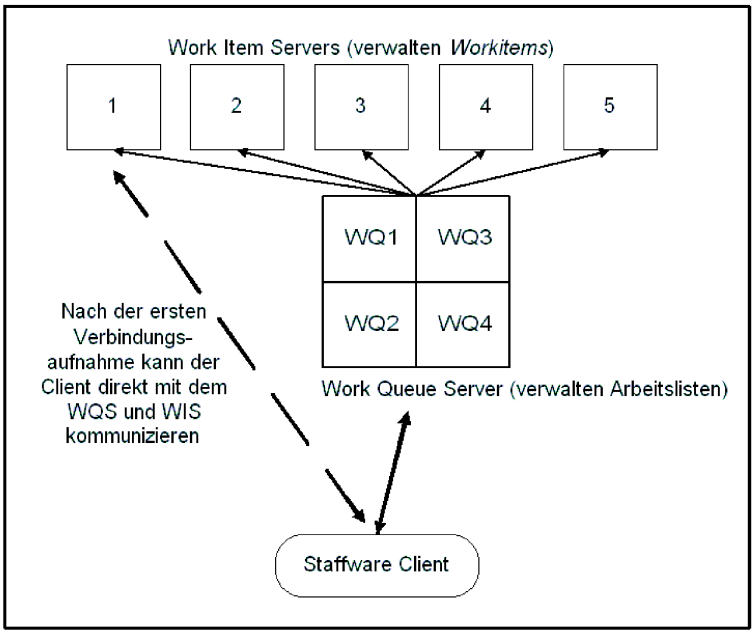



Abbildung 3.20: Kommunikation zwischen Client und WQS/WIS nach erfolgreichem Verbindungsaufbau

### 3.2.5 Datenspeicherung

Die Speicherung der Daten (Prozessmodelle, prozessrelevante Daten) kann auf zwei Arten erfolgen. Die Daten können

- im Filesystem (datei-basiert) des Servers oder
- in einer Datenbank.

abgelegt werden.
Als Datenbanken stehen der `Microsoft SQL Server` und `Oracle 8i` zur Verfügung. Die Verfügbarkeit der Datenspeicherungsmöglichkeiten auf verschiedenen Plattformen kann Abbildung 3.2.1 entnommen werden.
Die Daten können immer nur auf einer dieser Arten (entweder Filesystem oder Datenbank) gespeichert werden. Wie die Daten beispielsweise in einer Oracle 8i-Datenbanken gespeichert werden, kann anhand der Datenbank-Tabellen-Struktur im Anhang betrachtet werden (siehe Anhang D). In [MüSt99] wird auf leichte Performancenachteile der datei-basierten Lösung bei Workflow-Management-Anwendungen mit hohen Transaktionsraten und -volumina hingewiesen.

# Kapitel 4

# Korrektheit und Konsistenz von Prozessmodellen und -instanzen

Um auch unternehmenskritische Prozesse angemessen unterstützen zu können, muß ein WFMS robust und stabil laufen. Dazu muß zu jedem Zeitpunkt die Korrektheit und Konsistenz aller Prozessmodelle und -instanzen gewährleistet sein. So sollte es nicht möglich sein, daß aus inkorrekten Prozessmodellen neue Prozessinstanen abgeleitet werden können. Ebensowenig sollten bereits instanziierte Prozessmodelle von Benutzern derart abgeändert werden können, daß ihre Konsistenz oder Korrektheit verletzt wird.
Die Änderung von Prozessmodellen zur Modellierzeit (Entwicklungszeit) wird im Folgenden als `Modelländerungen` bezeichnet. Werden die Änderungen an Prozessmodellen mit laufenden Instanzen durchgeführt und die Auswirkungen auf die Prozessinstanzen überwacht, spricht man von `Schemaevolution`.
In den folgenden Abschnitten wird dargelegt, welche Unterstützung `MQSeries Workflow` und `Staffware` im Zusammenhang mit Modelländerungen und Schemaevolution bieten.

## 4.1 Konsistenzsicherung von Prozessmodellen zur Entwicklungszeit

In Bezug auf Erstellung und Änderung von Prozessmodellen muß sichergestellt werden, daß

1. nur korrekte Prozessmodelle zur Instanziierung freigegeben werden und

2. ein einmal freigegebenes (instanziierbares) Prozessmodell nach einer Änderung wieder korrekt und konsistent ist.

Damit ein Prozessmodell als korrekt betrachtet werden kann, müssen sein Ablauf- und Funktionsteil, sowie das Zusammenspiel (z.B. Datenübergabe) zwischen ihnen korrekt sein.
Besonders beachtet wird im Folgenden, wann mögliche Fehler in einem Prozessmodell vom WFMS angezeigt bzw. für den Modellierer sichtbar werden. Dies kann direkt beim Versuch der Änderung des Prozessmodells, beim Verifizieren (MQSeries WF) bzw. Testen (Staffware) des Prozessmodells oder erst zur Laufzeit (falls ein Fehler während der Modellierung nicht entdeckt wurde) geschehen. Die letzte Variante kann fatale Auswirkungen haben und sollte in einem WFMS eigentlich nicht vorkommen.
Die folgenden Abschnitte betrachten die von `MQSeries WF` und `Staffware` in diesem Zusammenhang gebotene Unterstützung. Im letzten Abschnitt werden besonders problematische Aspekte der Korrektheits- und Konsistenzsicherung zur Modellierzeit betrachtet.

### 4.1.1 MQSeries WF

Der Prozessmodellierer wird bei der Entwicklung von Prozessmodellen auf verschiedenste Art und Weise unterstützt.

#### 4.1.1.1 Fehler- und Warnmeldungen

Zur Überprüfung von Prozessmodellen bietet MQSeries WF die Option `verify` an [IBM99c]. Damit werden die Bestandteile eines Prozessmodells und deren Beziehungen untereinander überprüft (z.B. Überprüfung der maximalen Anzahl eingehender Kontrollkonnektoren für eine Aktivität). Eine genauere Beschreibung der Option `verify` findet sich in Abschnitt 2.1.2.6.
Tabelle 4.1 stellt die häufigsten Meldungen und ihre Einteilung in `Fehler` und `Warnung` dar. Ein erkannter Fehler im Prozessmodell verhindert z.B. einen erfolgreichen Import in die Runtime-Datenbank. Eine Warnung weist den Prozessmodellierer auf mögliche Probleme hin, der Import des Prozessmodells in die Runtime-Datenbank ist aber trotzdem möglich.

| **Meldung** | **Fehler oder Warnung** |
|---|---|
| Ein `Kontrollflusskonnektor` einer Aktivität kann nie aktiviert werden | Warnung |
| Für einen `Datenkonnektor` wurde kein Datenmapping vorgenommen | Warnung |
| Ein `Datenkonnektor` verbindet zwei Aktivitäten, die nicht über eine Sequenz von Kontrollflusskonnektoren erreichbar sind | Fehler |
| Die `Datenstruktur` des Inputcontainers einer Aktivität ist nicht identisch mit der geforderten Datenstruktur des zugeordneten `Anwendungsprogramms` | Fehler |
| Einer `Programmaktivität` wurde kein Anwendungsprogramm zugeordnet | Fehler |
| Einer `Blockaktivität` wurden keine Aktivitäten zugeordnet | Fehler |
| Einer `Prozessaktivität` wurde kein Prozessmodell zugeordnet | Fehler |
| Der `Startaktivität` (als Prozessaktivität) wurde das eigene Prozessmodell zugeordnet | Fehler |
| Die `Bearbeiterzuordnung` einer Aktivität ist nicht auflösbar (z.B. Eine Aktivität wird der Person zugeordnet, die eine nachfolgende Aktivität ausführen) | Fehler |
| Die `Bearbeiterzuordnung` einer Aktivität erfolgt über einen Container (Datenstruktur), welcher im Outputcontainer der Vorgängeraktivität nicht existiert | Fehler |

Tabelle 4.1: Warnungen und Fehlermeldungen der Option `verify`

Neben diesen Hinweisen auf mögliche Fehlerquellen verhindert MQSeries WF eine Reihe von fehlerhaften Angaben und Einstellungen im Prozessmodell durch Zurückweisen der gewünschten Änderung. So kann beispielsweise ein Kontrollflusskonnektor nicht zwischen zwei Aktivitäten eingefügt werden, wenn sich daraus ein Zyklus im Kontrollflussgraph ergibt. Tabelle 4.2 führt Einstellungen und Eigenschaften eines Prozessmodells auf, die von MQSeries WF während der Modellierung verhindert werden. Dazu erscheint eine Fehlermeldung, die den Prozessmodellierer auf seinen Fehler aufmerksam macht.

| **Verhindert wird ...** |
|---|
| `... im Kontrollfluss ...` |
| das Anlegen von Zyklen über Kontrollflusskonnektoren |
| das Anlegen von Kontrollflusskonnektoren zwischen der Quelle eines Prozessmodells und seinen Aktivitäten |
| das Anlegen von Kontrollflusskonnektoren zwischen Aktivitäten und Senke |
| das Erstellen von Transitionsbedingungen mit Datenelementen, die im Outputcontainer der Quellaktivität des Kontrollflusskonnektors nicht vorhanden sind |
| das mehrmalige Anlegen von Kontrollflusskonnektoren zwischen zwei Aktivitäten (jedoch werden transitive Abhängigkeiten nicht reduziert) |
| `... im Datenfluss ...` |
| das Anlegen von Zyklen im Datenfluss mittels Datenkonnektoren |
| das Anlegen von Standard- und Loop-Datenkonnektoren zwischen verschiedenen Aktivitäten |
| das mehrmalige Anlegen von Datenflusskonnektoren zwischen zwei Aktivitäten |
| das Mappen ungleicher Datentypen (z.B. Integer auf String) |
| `... bei Aktivitäten ...` |
| das Einrichten von Einstellungen einer Programmaktivität ohne Angabe eines zugeordneten Anwendungsprogramms |
| das Einrichten von Einstellungen einer Prozessaktivität ohne Angabe eines zugeordneten Prozessmodells |

Tabelle 4.2: Abweisungen von Änderungen am Prozessmodell

#### 4.1.1.2 Prüfung von Cross-Referenzen

Eine weitere Unterstützung erfährt der Prozessmodellierer durch die automatische Überwachung von `Cross-Referenzen`. Dabei werden Bezüge innerhalb der verschiedenen Modelle automatisch überwacht (siehe Beispiel 4.1). Der Prozessmodellierer wird von dieser Aufgabe entlastet, d.h. mögliche Fehlerquellen werden systemseitig ausgeschlossen.
Bezüge zwischen verschiedenen Modellen werden nicht aktualisiert (siehe Beispiel 4.2).

**Beispiel 4.1**: (*Cross-Referenzen im Organisationsmodell*)
Eine Beziehung zwischen einer Person und einer Organisation kann auf zwei Arten hergestellt werden:

1. Eine Person kann einer Organisation zugewiesen werden.

2. Einer Organisation können Personen als Mitglieder zugewiesen werden.

Der Prozessmodellierer muß nur eine der beiden Beziehungen erstellen. Die zweite Beziehungsart wird automatisch von MQSeries WF ergänzt. Beim Löschen solcher Bezüge greift dieses Verfahren ebenfalls.

**Beispiel 4.2**: (*Cross-Referenzen zwischen Daten- und Prozessmodell*)
Der Name einer Datenstruktur wird geändert. Durch die Cross-Referenzen wird sichergestellt, daß bei alle registrierten Programmen, die diese Datenstruktur verwendet haben, ebenfalls die Namensänderung durchgeführt wird.
Verwendet eine Aktivität diese Datenstruktur, greifen die Cross-Referenzen nicht. Der Name der Datenstruktur wird dort nicht aktualisiert, damit das Prozessmodell weiterhin korrekt wäre, sondern einfach entfernt. Der Prozessmodellierer wird vom Löschen dieser Einstellungen nicht informiert.

#### 4.1.1.3 Spezielle Probleme durch fehlende Anwendungsprogramme

`Anwendungsprogramme` sind ein wichtiger Bestandteil von Prozessmodellen und zwingend nötig für die Ausführung von Programmaktivitäten.
Ein Anwendungsprogramm muß in MQSeries WF nur einmal registriert werden und kann dann in beliebig vielen Prozessmodellen verwendet werden. Zu beachten ist, daß bei der Registrierung eines Anwendungsprogramms dessen physische Existenz nicht überprüft wird. Ein möglicher Schreibfehler oder das Kopieren des Anwendungsprogramms in einen falschen Ordner werden somit nicht frühzeitig erkannt, sondern erst während der Ausführung des Prozessmodells für den Benutzer sichtbar.
Die Reaktion des WFMS ist dann wie folgt: Ein gestartetes Workitem zeigt dem Benutzer das Fehlen des Anwendungsprogrammes nicht direkt an. Stattdessen wechselt es, für den Benutzer ohne ersichtlichen Grund, in einen Fehlerzustand. Die Diagnose bleibt dem Administrator des WFMS überlassen. Nach Behebung des Fehlers kann der Benutzer durch Neustart des Workitems seine Arbeit regulär fortsetzen.

#### 4.1.1.4 Spezielle Probleme beim Datenmapping

Das `Datenmapping` zwischen zwei Aktivitäten kann in Form ganzer Datenstrukturen oder einzelner Datenelemente vorgenommen werden. Komplette Datenstrukturen werden von MQSeries WF automatisch gemappt, wenn die Datenstruktur des Outputcontainers identisch mit der Datenstruktur des Inputcontainers ist. Wird zusätzlich noch ein einzelnes Datenelement gemappt, so wird dieses Mapping von MQSeries WF ignoriert, wobei der Prozessmodellierer nicht gewarnt wird.
Dieses Problem könnte beispielsweise entstehen, wenn die Datenstruktur die Daten einer Personalakte enthält. Für diese Datenstruktur besteht ein Anwendungsprogramm, welches dem Bearbeiter alle Daten darstellt. Der Bearbeiter einer bestimmten Aktivität soll aber das Gehalt eines Kollegen nicht angezeigt bekommen. Deshalb werden dieser Aktivität nur der Name und die Adresse des Kollegen übergeben.
Um dieses Problem zu lösen, muß das von MQSeries WF automatisch vorgenommene Mapping entfernt werden. Danach gelten nur noch die Mappings der einzelnen Datenelemente. Alternativ kann das Anwendungsprogramm für diese Aktivität so implementiert werden, daß die Gehaltsdaten ausgeblendet werden.

### 4.1.2 Staffware

In Staffware wird der Prozessmodellierer auf drei Arten bei der Erstellung konsistenter Prozessmodelle unterstützt:

1. Fehlerhafte Angaben und Einstellungen werden zurückgewiesen und unvollständige Eingaben nicht zugelassen.

2. Fehlende Einstellungen an Aktivitäten werden graphisch angezeigt.

3. Prozessmodelle können getestet werden.

Durch den ersten Punkt verhindert Staffware eine Reihe von Fehlerquellen. So kann beispielsweise als Bearbeiter eines Formulars nur eine dem System bekannte Person zugewiesen werden (*fehlerhafte Angabe oder Einstellung*). Aktivitäten dürfen nicht ohne Bezeichner erstellt werden (*unvollständige Eingabe*). Fehler werden mit einer aussagekräftigen Fehlermeldung dem Prozessmodellierer bekannt gemacht. Unvollständige Eingaben werden verhindert, indem Einstellungen erst übernommen werden, wenn alle obligaten Datenfelder korrekt ausgefüllt sind. Tabelle 4.3 gibt einen Überbilck:

| **Verhindert werden ...** |
|---|
| `... Fehler ...` |
| bei der Erstellung von Bedingungen (z.B. bei Bedingungen (*Conditions*) im Kontrollflussgraph (siehe Kapitel 2)) |
| bei der Erstellung von Ausdrücken (z.B. Frist-Ausdruck für Zeit und Datum) |
| bei der Bearbeiterzuordnung (Es werden nur dem System bekannt Personen, Rollen und Gruppen akzeptiert) |
| falls es mehrere Kontrollflusskonnektoren zwischen zwei Aktivitäten gibt |
| falls es mehrere Kontrollflusskonnektoren zwischen Start- bzw. Endsymbol und mehrern Aktivitäten gibt |
| Kontrollflusskonnektoren zu einem Warteschritt falls die Startaktivität bereits eine oder mehrere ausgehende Kontrollflusskonnektoren besitzt |
| `... unvollständige Eingaben ...` |
| bei Aktivitäten (z.B. fehlender Name) |
| bei Feldern in einem Formular (z.B. Feld wurde noch nicht definiert) |
| bei Prozessmodellen (Der Status kann erst dann auf `Freigegeben` gesetzt werden, wenn alle nötigen Eingaben und Einstellungen vorgenommen wurden) |

Tabelle 4.3: Abweisungen von Änderungen oder unvollständigen Eingaben am Prozessmodell

Sind noch nicht alle benötigten Daten und Einstellungen einer Aktivität festgelegt worden, wird diese dem Prozessmodellierer durch ein rotes Fragezeichen an der rechten oberen Ecke der betroffenen Aktivität symbolisch angezeigt. Welche Daten oder Einstellungen fehlen ist hieraus allerdings nicht ersichtlich.
Nicht vollständig spezifizierte Aktivitäten verhindern die Freigabe des Prozessmodells. Sein Status ist dann `unvollständig`. Es kann vom Prozessmodellierer nicht direkt manipuliert werden, sondern nur wenn die fehlenden Daten oder Einstellungen festgelegt bzw. vorgenommen werden. Der Status wechselt dann automatisch in `nicht freigegeben`.

Ein Prozessmodell kann getestet werden, sobald es im Status `nicht freigegeben` ist. Zu diesem Zweck wird dem Prozessmodellierer ein solches Prozessmodell im Menü `Prozess starten` (*Case Start*) angezeigt. Er kann das Prozessmodell auswählen und starten. Die von diesem Prozessmodell erzeugten Workitems werden alle in der Test-Arbeitsliste des Prozessmodellierers angezeigt (siehe Kapitel 2.2.2.3).
Dadurch kann der Prozessmodellierer die Ausführung eines Prozessmodells simulieren sowie alle noch existierenden Fehler im Prozessmodell (z.B. falsche Bearbeiterzuordnung oder ungewollte Zyklen) erkennen und beseitigen.

#### 4.1.2.1 Spezielle Problemfälle von Staffware

In der verwendeten Version von Staffware können für Aktivitäten nur `Formulare` zur Interaktion mit dem Bearbeiter hinterlegt werden. Es wird jedoch nicht erzwungen, daß ein Formular angelegt wird. Eine graphische Anzeige der fehlenden Einstellung erfolgt ebenfalls nicht.
Geht man davon aus, daß jedes Prozessmodell vor der Freigabe getestet wird, ist dieses Verhalten nicht gefährlich. Es erzeugt lediglich Mehrarbeit, da entsprechende Fehler erst relativ spät erkannt werden.

Staffware erlaubt es dem Prozessmodellierer, Aktivitäten beliebig im Prozessmodelldiagramm zu platzieren. Ob eine Aktivität im Ablauf (Kontrollflussgraph) eingebunden ist, wird von Staffware nicht überwacht. So können Aktivitäten ohne Verbindung mit anderen Aktivitäten oder auch nur mit ausgehenden Kontrollflusskonnektoren erstellt werden. Diese Aktivitäten werden zur Laufzeit

nie aktiviert und werden deshalb für Bearbeiter niemals sichtbar.
Im Gegensatz zu MQSeries WF lässt Staffware nur eine Startaktivität zu. Dies ist die Aktivität, die mit dem Startsymbol durch einen Kontrollkonnektor verbunden ist.

### 4.1.3 Betrachtung ausgewählter Szenarien

Dieser Abschnitt stellt allgemeine Problemstellungen bei Modelländerungen vor und zeigt wie `MQSeries WF` und `Staffware` darauf reagieren.

#### 4.1.3.1 Datenflussszenarien

**Szenario 1** (*Unsichere Datenversorgung*)
Die obligaten Werte eines Eingabecontainers müssen zur korrekten Ausführung eines Aktivitätenprogramms von einer vorangehenden Aktivität geschrieben werden. Abbildung 4.1 stellt ein Szenario dar, bei dem die Versorgung des Eingangscontainer der Aktivität D nicht sichergestellt ist. Grund dafür ist, daß die Aktivität B1 nicht immer zwingend ausgeführt wird.

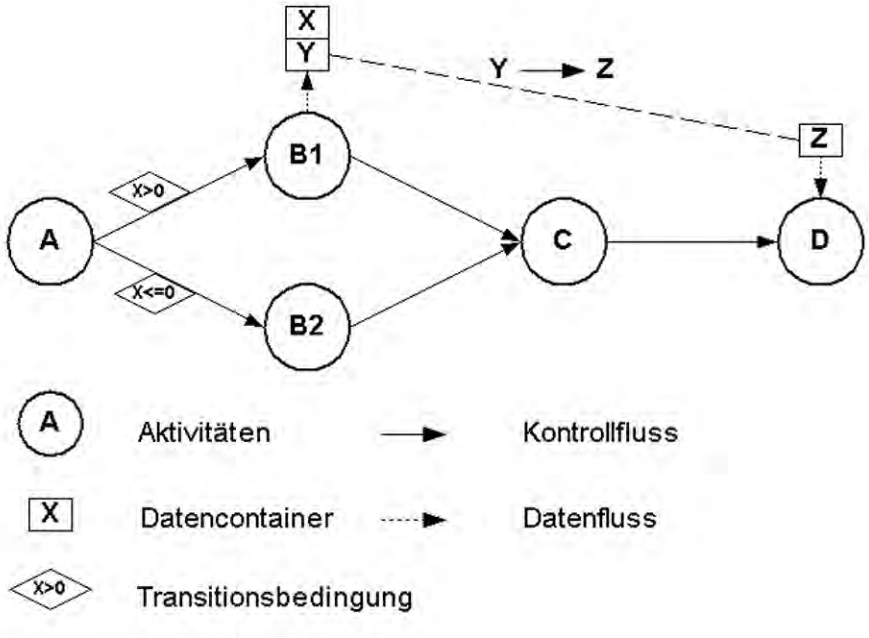



Abbildung 4.1: Unzureichende Versorgung eines Eingangscontainer

**MQSeries WF**:
Dieses Szenario kann mit der Buildtime-Komponente von MQSeries WF problemlos modelliert werden. Der Prozessmodellierer wird zu keinem Zeitpunkt auf diesen ggf. unerwünschten Sachverhalt hingewiesen. Bereits bei etwas komplexeren, unübersichtlicheren Prozessmodellen kann diese Art Fehler daher relativ einfach entstehen.
Sichtbar wird die fehlende Datenversorgung erst während der Ausführung des Anwendungsprogramms der Aktivität D. Die Reaktion ist dann von der Programmierung des Anwendungsprogramms (bzw. dessen Behandlung entsprechender Fehlerfälle) und nicht von MQSeries WF abhängig. Prinzipiell bietet die MQSeries WF-API die Möglichkeit, zu prüfen, ob ein Datenelement eines Containers geschrieben wurde oder nicht (siehe [IBM99e]).
Mögliche Reaktionen eines Anwendungsprogramms:

- Vorprogrammierte Fehlerausgaben, z.B. "*Dateninputcontainer konnte nicht gelesen werden*"

- Fehlerausgabe des Anwendungsprogramms, falls das fehlende Datenelement z.B. in einer mathematischen Rechung integriert ist (z.B. "*Division durch Null*").

- Keine Reaktion (falls das erfolgreiche Lesen des Datenelementes nicht überwacht wurde und das Datenelement im Anwendungsprogramm nicht verwendet wird).

Der Benutzer hat keine Möglichkeit, auf einen dieser für ihn sichtbaren Fehlerausgaben zu reagieren. Die Daten fehlen und können nicht einfach in die laufende Prozessinstanz eingefügt werden. Erhält der Benutzer keine Fehlerausgabe, ist es unsicher, ob die fehlende Datenversorgung schnell bemerkt wird. Möglicherweise führen gerufene Anwendungsprogramme sogar falsche Berechnungen durch. Bei Fehlerausgaben oder unerklärlichen leeren Feldern im Anwendungsprogramm (z.B. leere Bildschirmmaske) sollte der Prozessmodellierer oder der Administrator des WFMS informiert werden.

**Staffware**:
Oder-Verzweigungen können in Staffware durch `Bedingungen` modelliert werden. Somit ist das beschriebene Szenario modellierbar. Ein unversorgtes Formularfeld bleibt bei der Darstellung leer bzw. unvollständig ausgefüllt (z.B. wenn ein Feld sich aus den Werten mehrerer Felder zusammensetzt). Der Prozessmodellierer wird auf unversorgte Formularfelder nicht hingewiesen.
Eine besondere Problematik ergibt sich bei automatischen Schritten. Da hier Daten an ein externes Programm übergeben werden und keine Benutzerinteraktionen stattfinden, kann hier ein unversorgtes Formularfeld zu schwerwiegenden Fehlern (z.B. bei Berechnungen) führen. Eine Möglichkeit, diese Art Fehler auszuschließen besteht darin, sie in den externen Programmen gezielt abzufangen. Dies gilt auch falls für Aktivitäten keine Formulare, sondern VBA-Projekte (siehe Kapitel 5) hinterlegt werden.

**Szenario 2** (*Lost Updates*)
Enthält ein Prozessmodell parallele Verzweigungen, so kann der Eingangscontainer einer Aktivität von zwei oder mehreren im voraus paraller ablaufenden Aktivitäten geschrieben werden. In Abbildung 4.2 wird der Eingangscontainer $Z$ der Aktivität `D` von den beiden Aktivitäten `B1` und `B2` geschrieben. Allerdings ist nicht deterministisch, von welchem der beiden Knoten der Wert letztlich geliefert wird.

**MQSeries WF**:
Das Mappen mehrerer Outputcontainer auf einen Inputcontainer ist möglich. Eine Überprüfung, ob dieses Vorgehen sinnvoll ist (z.B. bei ODER-Verzweigungen im Kontrollfluss), wird von MQSeries WF nicht durchgeführt. Der Prozessmodellierer erhält auch keine Warnung auf einen eventuellen Datenverlust.
Für die Ausführung eines Prozessmodells bedeutet dies, das bis auf den letzten geschriebenen Wert eines mehrfach gemappten Datenelementes alle anderen Werte verloren gehen. Werden die entsprechenden Aktivitäten von verschiedenen Bearbeitern ausgeführt, ist der Fehler für die einzelnen Personen nicht direkt sichtbar.

**Staffware**:
Einmal definierte Felder können in Staffware-Formularen jederzeit verwendet werden. Dadurch können bei paralleler Auführung (siehe Abbildung 4.2) oder bei sequentieller Ausführung von Aktivitäten ungewollt bereits festgelegte Feldwerte überschrieben werden. Die Verwendung von Feldern wird von Staffware nicht überwacht. Der Prozessmodellierer erhält dementsprechend keinen Hinweis auf mögliche Gefahren, und er hat auch keine Möglichkeit sich einen Überblick über die Verwendung von Formularfeldern in einem Prozessmodell verschaffen.

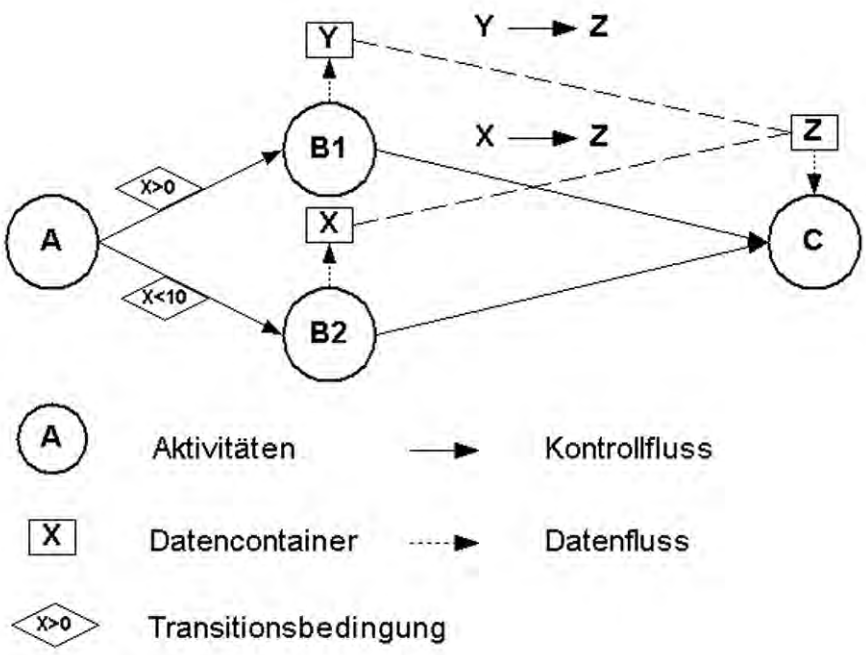



Abbildung 4.2: Lost Update durch unsynchronisiertes schreiben eines Eingangscontainer

#### 4.1.3.2 Kontrollflussszenarien

**Szenario 3** (*Deadlocks*)
Der Kontrollflussgraph beinhalte einen Zyklus, so dass keine eindeutige Endaktivität im Prozessmodell existiert. Das in Abbildung 4.3 dargestellte Prozessmodell etwa besitzt einen Zyklus zwischen den Aktivitäten A, B und C. Dadurch wird verhindert, daß eine Prozessinstanz regulär beendet werden kann.
Ein Zyklus in einem Kontrollflussgraphen führt ebenfalls dazu, daß in den meisten Fällen mindestens eine Aktivität mehr als einen eingehenden Kontrollkonnektor besitzt (Aktivität A in Abbildung 4.3). Falls eine Aktivität erst aktiviert werden kann, wenn alle eingehenden Kontrollflusskonnektoren bewertet sind, würde Aktivität A nie aktiviert werden können (*Deadlock*).

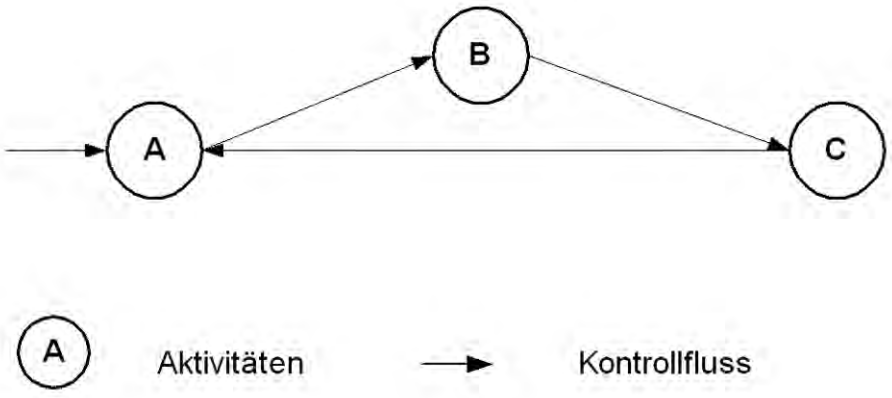



Abbildung 4.3: Zyklen ohne Abbruchbedingung

**MQSeries WF**:

Das Modellieren von Zyklen im Kontrollfluss ist in MQSeries WF nicht auf direktem Wege möglich. Wie in Tabelle 4.2 dargestellt, dürfen Kontrollflusskonnektoren nur angelegt werden, wenn sich daraus kein Zyklus ergibt. Der Grund dafür sind die von MQSeries WF verwendeten Bedingungen zur Aktivierung von Aktivitäten (siehe Kapitel 2.1.3.2.2). Der notwendigen Bedingung folgend, könnte die Aktivität A nie aktiviert werden, da der von Aktivität C eingehende Kontrollflusskonnektor nie bewertet wird. Dies wäre eine typische Deadlock-Situation.
Auf indirektem Weg können jedoch Zyklen modelliert werden. Alle Aktivitäten verfügen über eine sogenannte Exit-Bedingung. Eine Exit-Bedingung ist ein logischer Ausdruck oder eine Verknüpfung logischer Ausdrücke mit Bezug auf den Outputcontainer der Aktivität. Ist die Exit-Bedingung nach der Ausführung der Aktivität nicht erfüllt, wird sie nochmals aktiviert. Somit sind speziell mit Block- und Prozessaktivitäten einfache Schleifen modellierbar. Der Modellierer wird nicht gewarnt bzw. auf die Risiken (z.B. Aktivität wird endlos ausgeführt) hingewiesen.

**Staffware**:
Staffware erlaubt eine uneingeschränkte Modellierung von Zyklen. Der Prozessmodellierer wird bei der Erstellung eines Zyklus nicht auf die möglichen Gefahren (z.B. das Prozessmodell besitzt keine reguläre End-Aktivität) hingewiesen.
*Deadlocks* können in Staffware-Prozessmodellen nicht auftreten. Dies ist darin begründet, daß eine Aktivität aktiviert wird sobald eine im Kontrollfluss direkt vorgeschaltete Aktivität beendet wird und nicht wie bei MQSeries WF alle eingehenden Kontrollflusskonnektoren bewertet sein müssen.

**Szenario 4** (*Unzureichende Transitionsbedingungen*)
Transitionsbedingungen definieren Übergangsbedingungen zwischen Aktivitäten. Sie dienen der Modellierung bedingter Ablaufzweige. In Abbildung 4.4 decken die beiden Transitionsbedingungen nicht alle möglichen Werte der Transitionsvariablen ab. Dadurch können Teile eines Prozessmodells in gewissen Fällen unerreichbar bzw. niemals durchlaufen werden.

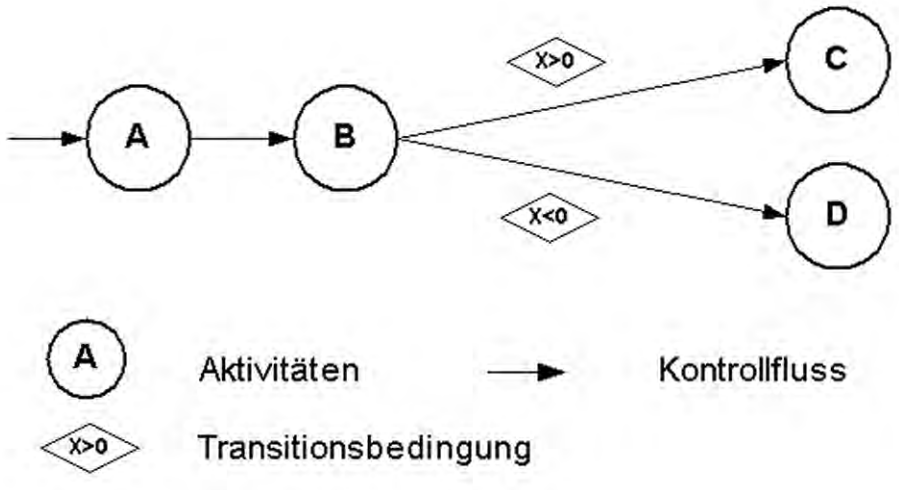



Abbildung 4.4: Nicht ausreichende Transitionsbedingungen

**MQSeries WF**:
Dieses Szenario kann in MQSeries WF modelliert werden. Der Prozessmodellierer wird nicht darauf hingewiesen, daß das Prozessmodell während der Ausführung eventuell zu früh beendet wird.

Natürlich gibt es Anwendungsbeispiele, in denen ein solches Verhalten erwünscht sein kann. Dennoch sollte der Prozessmodellierer zumindest auf mögliche Probleme aufmerksam gemacht werden. Ein solches Prozessmodell kann in die Runtime-Datenbank eingespielt und instanziiert werden. Wird für die Transitionsvariable in Abbildung 4.4 beispielsweise `x=0` gewählt, so werden beide Kontrollflusskonnektoren mit `FALSE` bewertet. Es greift die Death-Path-Elimination (siehe Abschnitt 2.1.3.2.2) und die Prozessinstanz gilt als regulär beendet.

**Staffware**:
Transitionsbedingungen werden in Staffware durch prädikative Bedingungen abgebildet. Bei mehr als einer Transitionsbedingung müssen dementsprechend mehrere Bedingungen verwendet werden. Für die Abbildung des Szenario 4 sind zwei Bedingungen ($X > 0$ und $X < 0$) zu definieren.
Bedingungen haben standardmäßig zwei Ausgänge - `TRUE` und `FALSE`. Für unser Szenario werden bei beiden Bedingungen nur die Ausgänge für `TRUE` mit einer folgenden Aktivität (die Aktivitäten C und D) verbunden. Evaluieren beide Bedingungen zu `FALSE`, gilt die Prozessinstanz als regulär beendet. Der Prozessmodellierer wird nicht gewarnt bzw. die Bedingung als unvollständig betrachtet.

**Szenario 5**: (*Endlos-Verschachtelungen durch Subprozesse*)
Ein Prozessmodell kann einen oder mehrere Subprozesse enthalten, die weiter verschachtelt sein können. Ruft eine Aktivität das Prozessmodell das sie beinhaltet (*Rekursion*), so kann es zu ungewollten Verschachtelungen (Zyklen) kommen.
Besonders problematisch ist die Verwendung eines Subprozesses als ersten Schritt bzw. Aktivität innerhalb eines Prozessmodells. In diesem Fall würden bei Instanziierung eines solchen Prozessmodells theoretisch unendlich neue Instanzen erzeugt, ohne das je eine der dem Subprozess folgenden Aktivitäten ausgeführt werden könnte.

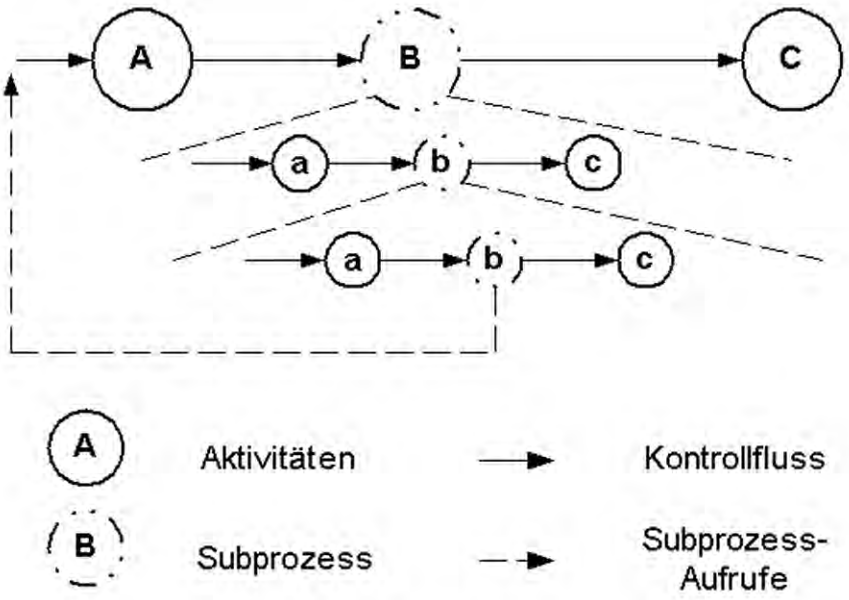



Abbildung 4.5: Verschachtelungen durch Subprozesse

**MQSeries WF**:
Prozessaktivitäten können an jeder beliebigen Stelle im Prozessmodell eingebaut werden. Dabei kann jedes in der Buildtime-Komponente hinterlegte Prozessmodell gewählt werden. Es existiert nur eine Einschränkung: Ist eine Prozessaktivität gleichzeitig Startaktivität eines Prozessmodells,

so darf ihr nicht das Prozessmodell zugeordnet werden, daß sie selbst beinhaltet. MQSeries WF verhindert diese Einstellung durch eine Fehlermeldung.
Dennoch sind Prozessmodelle modellierbar, die nie ein reguläres Ende finden. Das in Abbildung 4.5 dargestellte Prozessmodell etwa kann nie regulär beendet werden, da die Aktivität `C` niemals aktiviert werden kann. Es werden theoretisch beliebig viele Aufrufe des Subprozesses erfolgen, welcher natürlich nie beendet wird.
MQSeries WF warnt den Prozessmodellierer nicht vor solchen Konstruktionen. Der Prozessmodellierer wird nur in der Dokumentation der Buildtime-Komponenten [IBM99c] auf einen sorgsamen Gebrauch von Prozessaktivitäten hingewiesen.

**Staffware**:
Prozessmodelle müssen vor ihrer Speicherung als Haupt- bzw. Sub-Prozess gekennzeichnet werden. Dadurch sind rekursive Prozessaufrufe von Haupt-Prozessen nicht möglich.
Jedoch kann ein Subprozess sich selbst wieder als Subprozess aufrufen. Dadurch sind beliebige Rekursionen und Verschachtelungen von Sub-Prozessen möglich. Es bleibt dem Prozessmodellierer überlassen, wie er den Überblick über alle Prozessmodelle und deren Verwendung als Haupt- bzw. Subprozess bewahrt. Staffware bietet dafür keine Unterstützung.

#### 4.1.3.3 Bearbeiterzuordnung

**Szenario 6** (*Fehlende oder falsche Bearbeiterzuordnungen*)
Jeder Aktivität ist mittels statischer oder dynamischer Bearbeiterzuordnung eine Menge potentieller Bearbeiter zugewiesen. Wird einer Aktivität eine einzelne Person als Bearbeiter zugeordnet, so erhält nur diese das entsprechende Workitem in ihrer Arbeitsliste.
Wird diese Person (in Abbildung 4.6 ist dies Hr. Müller) während der Laufzeit aus dem WFMS entfernt, ist das Workitem nicht mehr erreichbar und die Prozessinstanz kann nicht mehr weiter ausgeführt werden.
Diese Problematik kann sich auch durch Schreibfehler oder fehlende Angaben in der Bearbeiterzuordnung ergeben.

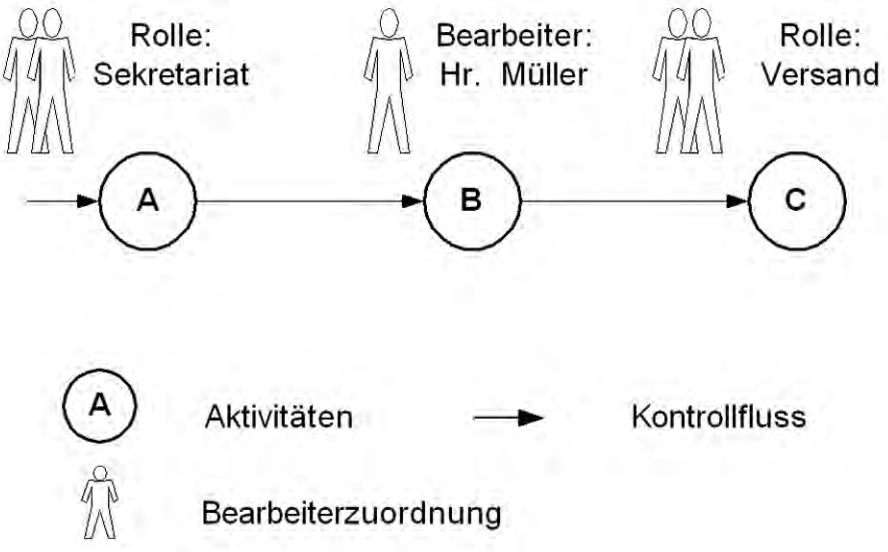



Abbildung 4.6: Fehlende Bearbeiterzuordnung

**MQSeries WF**:
Die Bearbeiterzuordnung von MQSeries WF ist sehr restriktiv. Es können nur Personen, Rollen und Organisationen verwendet werden, die zum Zeitpunkt der Erstellung in der Buildtime-Komponente existieren. Denkbare Schreibfehler bei der Bearbeiterzuordnung sind daher, im Gegensatz zum Vorgängerprodukt `Flowmark` [Ro96], ausgeschlossen.
Abbildung 4.6 zeigt ein Prozessmodell mit bereits vorgenommener Bearbeiterzuordnung. Wenn nun die Person `Hr. Müller` aus dem Organisationsmodell gelöscht wird, ist die Aktivität `B` ohne Bearbeiterzuordnung. Der Prozessmodellierer wird weder beim Löschen der Person noch beim Verifizieren des Prozessmodells auf diesen Feher hingewiesen. Das Prozessmodell kann ohne Warnung exportiert und in die Runtime-Datenbank importiert werden.
Tritt eine entsprechende Situation zur Laufzeit einer Prozessinstanz auf, so zeigt MQSeries ein sonderbares Verhalten (siehe Szenario 12): Bei Aktivierung der Aktivität `B` wird jeder in der Runtime-Komponente bekannten Person (vom Azubi bis zum Manager) ein entsprechendes Workitem zugewiesen. Eine solch gravierende "Sicherheitslücke" ist in vielen Umgebungen jedoch nicht akzeptabel.

**Staffware**:
Die Verwaltung von Personen, Gruppen und Rollen erfolgt bei Staffware zentral über den Administration-Manager. Dort kann zu jedem Zeitpunkt eine Person aus dem System gelöscht werden. Der Prozessmodellierer merkt davon nichts, auch wenn die gelöschte Person in einen oder mehreren Prozessmodellen referenziert wird. Neue Bearbeiterzuordnungen können mit der gelöschten Person nicht mehr erstellt werden, da diese sofort nach dem Publizieren ihrer Löschung nicht mehr verfügbar ist. In Szenario 12 wird die Reaktion von Staffware zur Laufzeit eines solchen Prozessmodells beschrieben.

**Szenario 7** (*Nicht erfüllbare Bearbeiterzuordnungen*)
Durch die dynamische Bearbeiterzuordnung (z.B. `Rolle = Arzt`) wird einer Aktivität eine Menge von möglichen Bearbeitern zugewiesen. Die Festlegung, welche Personen in die Bearbeitermenge fallen, findet zur Laufzeit bei der Aktivierung der Aktivität statt.
Probleme können entstehen, wenn die in der Bearbeiterzuordnung festgelegte dynamische Bearbeiterzuordnung sich aus mehreren einschränkenden Einstellungen zusammensetzt. Für eine Berarbeiterzuordnung der Art "`Rolle = Arzt und Organisation = Verwaltung`" sollte bereits zur Modellierzeit sichergestellt werden, ob es überhaupt Personen geben kann, die diese Kombination erfüllen.

**MQSeries WF**:
MQSeries ermöglicht eine dynamische Bearbeiterzuordnung anhand von `Rollen`, `Organisationen` und `Stufen`. Wird je eine Einstellung für die Rolle und für die Organisation vorgenommen, so werden sie mit einem logischen UND verknüpft. Das bedeutet, daß nur Personen die beide Einstellungen erfüllen das Workitem der entsprechenden Aktivität zur Laufzeit angezeigt bekommen. Die Buildtime-Komponente von MQSeries WF prüft nicht, ob es überhaupt Personen gibt, die die Einstellungen der dynamischen Bearbeiterzuordnung erfüllen. Eine Warnung wäre hier angebracht, da dieses Problem bei komplexeren Organisationsmodellen leicht entstehen kann.
Wird dennoch ein Prozessmodell mit einer nicht erfüllbaren Bearbeiterzuordnung instanziiert, so erhält zur Laufzeit der Administrator von MQSeries WF die Workitems der entsprechenden Aktivitäten. Damit hat er, wenn auch sehr spät, die Möglichkeit den Fehler zu erkennen und zu beheben.

**Staffware**:
Im Gegensatz zu MQSeries WF verknüpft Staffware die gemachten Angaben bei der Bearbeiterzuordnung nicht. So werden Angaben bzgl. Person, Gruppe und Rolle getrennt ausgewertet (im Sinne einer ODER-Verknüpfung der Angaben) und dementsprechend mehrere Workitems erstellt

(für alle Mitglieder der Gruppe=Verwaltung und für die Rolle=Arzt). Die in Szenario 7 dargestellte Problematik gibt es somit in Staffware nicht.
Diese Art der dynamischen Bearbeiterzuordnung von Staffware ist nur für "informative" Workitems (nur Weitergabe von vorhandenen Daten) anwendbar, da bei mehrfacher Ausführung eines Workitems kein schreibender Zugriff sinnvoll erfolgen kann (siehe Szenario 2).

## 4.2 Schemaevolution

Im Folgenden werden `Schemaänderungen` und die daraus resultierenden Auswirkungen auf sich in Ausführung befindliche Prozessinstanzen betrachtet. Ein WFMS kann auf verschiedene Arten auf Änderungen an Prozessmodellen reagieren:

1. Die Änderungen werden auf laufende Prozessinstanzen propagiert (`Schema-Evolution`, siehe [Dad99]),
2. die Änderungen werden nur auf eine Teilmenge der laufenden Prozessinstanzen angewendet (z.B. Prozesse mit noch nicht zu weit fortgeschrittenem Status) oder
3. die Änderungen gelten nur für neue Prozessinstanzen.

Unabhängig von der Art der Reaktion des WFMS können sich Probleme ergeben. Im folgenden Abschnitt werden allgemeine Problemstellungen aufgezeigt und am Beispiel von MQSeries WF und Staffware diskutiert.

### 4.2.1 MQSeries WF

Ein existierendes Prozessmodell kann in der Buildtime-Komponente von MQSeries WF abgeändert werden. Ist das abgeänderte Prozessmodell im Sinne der Anforderungen von MQSeries WF korrekt (siehe Abschnitt 4.1.1), kann es in die Runtime-Datenbank eingespielt werden. Ab diesem Zeitpunkt kann nur noch das neue (abgeänderte) Prozessmodell instanziiert werden, d.h. neue Prozessinstanzen werden immer von der aktuellsten Version eines Prozessmodells abgeleitet. Die aus diesem Verhalten resultierenden Nachteile werden in Abschnitt 4.2.4 besprochen.
Auf laufende Prozessinstanzen hat das Einspielen von Änderungen am Prozessmodell dagegen keine Auswirkungen. Dies liegt daran, daß für jede Instanz eine eigenständige Kopie des Prozessmodells angelegt wird.

**Indirekte Einflussmöglichkeit**
Die einzige Möglichkeit, die Änderungen an einem Prozessmodell für bereits laufende Prozessinstanzen zu berücksichtigen, resultiert in Verbindung mit Subprozessen. Der Grund dafür ist, daß ein Subprozess erst bei der Aktivierung der zugehörigen Prozessaktivität instanziiert wird. Entsprechend können Änderungen am Prozessmodell eines Subprozesses bis zu dessen Aktivierung berücksichtigt werden. Eine Besprechung dieses Verhaltens erfolgt in Abschnitt 4.2.3.4.

### 4.2.2 Staffware

Ein bereits freigegebenes (instanziierbares) Prozessmodell kann im GWD betrachtet und geändert werden. Nach einer Änderung kann sich ein Prozessmodell im Status

- `unvollständig` (die Änderungen entspechen nicht den in Abschnitt 4.1 besprochenen Anforderungen) oder
- `freigegeben` (es können Instanzen des Prozessmodells erstellt werden)

befinden. Da Staffware im Gegensatz zu MQSeries WF nur über **eine** Datenbank zur Speicherung sämtlicher Prozessmodelle verfügt, wirken sich Änderungen immer sofort auf die Laufzeitumgebung aus. Änderungen im Admin-Manager (z.B. Entfernen einer Person) werden erst durch die Propagierung auch für laufende Instanzen gültig.
Prozessmodelle können nach ihrer erneuten Speicherung, falls sie nach einer Änderung den Status `unvollständig` aufweisen, nicht mehr instanziiert werden. Der Versuch eine Instanz anzulegen wird mit dem in Abbildung 4.7 gezeigten Fehler verhindert.

Abbildung 4.7: Instanziieren unvollständiger Prozessmodelle in Staffware

In Ausführung befindliche Instanzen werden frühzeitig beendet, ohne das der Bearbeiter eine Benachrichtigung erhält. Das Staffware-Protokoll eines solchen Instanz-Abbruchs zeigt Abbildung 4.8. Dabei handelt es sich um die Ausgabe der Instanzverwaltung (*Case Manager*, siehe Kapitel 5) von Staffware.

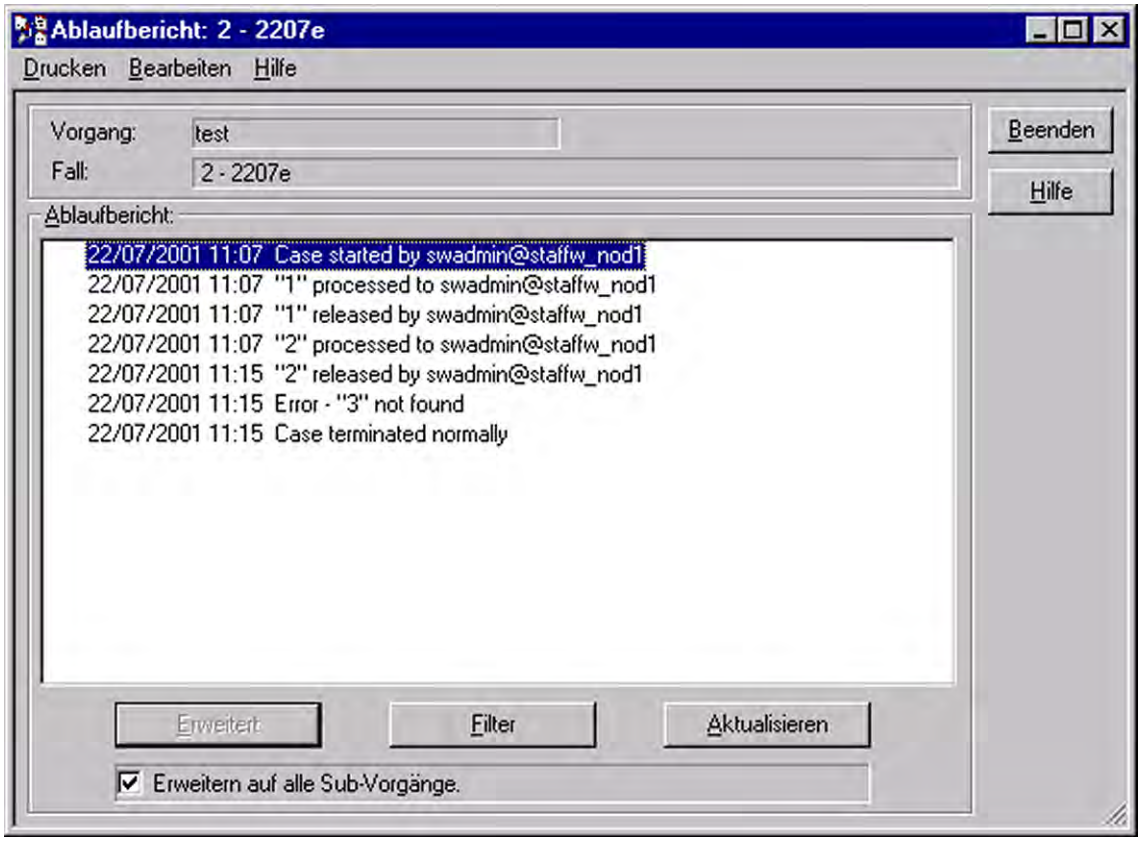



Abbildung 4.8: Dokumentation unvollständiger Prozessmodelle in Staffware

Nach einer Änderung weiterhin korrekte (*freigegebene*) Prozessmodelle erlangen nach dem Speichern sofort Gültigkeit. Die eingebrachten Änderungen sind nicht nur für neue Instanzen, sondern auch für laufende Instanzen gültig. So kann beispielsweise der Kontrollflussgraph einer laufenden

Instanz beeinflusst werden.
Der folgende Abschnitt verdeutlicht, welche Probleme und Gefahren dieses Verhalten für die verantwortlichen Personen (z.B. Prozessmodellierer, Systemadministratoren) verursacht.

Ein eigenartiges **(Fehl-)Verhalten** weist das Zusammenspiel der Bearbeiterverwaltung und die Einstellungen von Personen mit Supvervisor-Rechten im Standard-Client auf.
Person A habe Supervisor-Rechte in Bezug auf Person B. Des weiteren richte A (z.B. wegen dauerhafter Erkrankung von B) eine unbegrenzte Weiterleitung der Workitems von B in die eigene Arbeitsliste ein. B kann die Arbeit nicht mehr antreten und verläßt das Unternehmen, woraufhin der Staffware-Administrator diese Person aus dem WFMS entfernt. Dadurch verschwindet die Administrations-Arbeitsliste für B im Standard-Client von A. Eine neue Person wird zu Erfüllung der Arbeit von B eingestellt. Nun erhält diese Person dieselbe Bezeichnung wie sie B hatte (z.B. weil die Raumnummer als Bezeichnung gewählt wurde oder wirklich beide Müller heißen). Die alte Weiterleitung an A existiert wieder, obwohl sie für diese Person nie eingerichtet wurde. Auch besitzt A noch nicht einmal die Rechte diese Weiterleitung wieder rückgängig zu machen. Als Resultat bleibt eine Weiterleitung, die in diesem Zustand von niemanden entfernt werden kann. Dazu muß im `Administration Manager` allerdings erst ein Supervisor für B eingerichtet werden. Weitere Probleme ergeben sich, falls B über Supervisor-Rechte verfügte. Der Staffware-Administrator wird beim Löschen einer Person nicht auf solche Einstellungen hingewiesen. Daraus resultiert, daß die neue Person B die alten Supervisor-Rechte "erbt".
Das Potential zur Erzeugung unnötiger Fehler ist durch dieses Verhalten sehr hoch.

### 4.2.3 Betrachtung ausgewählter Szenarien

Durch die Schema-Evolution sollte auch (soweit gewünscht und möglich) eine Propagierung der Schemaänderungen auf laufende Prozessinstanzen sichergestellt werden. Erfolgt diese Propagierung nicht, werden die Änderungen erst für neue Prozessinstanzen gültig. Die korrekte Ausführung eines Prozessmodells ist in beiden Fällen nicht selbstverständlich. Die auftretenden Probleme werden in den folgenden Abschnitten aufgezeigt.
Ein Problem ensteht in jeden Fall wenn ein Prozessmodell nach einer Änderung das ursprüngliche Prozessmodell nicht ersetzen, sondern als eigenständiges Prozessmodell den Bearbeitern zur Verfügung stehen soll. Die Problematik der Versionierung wird in Abschnitt 4.2.4 besprochen.

#### 4.2.3.1 Prozessmodell

**Szenario 8** (*Einfügen neuer Aktivitäten*)
In ein Prozessmodell werden, wie in Abbildung 4.9 dargestellt, zwei Aktivitäten in den Kontrollfluss eingefügt. Zusätzlich gebe es zwischen den neuen Aktivitäten eine Datenabhängigkeit.
Laufende Prozessinstanzen können sich u.a. in folgenden Zuständen befinden:

1. Die neu eingefügten Aktivitäten liegen im noch nicht durchlaufenen Bereich des Prozessmodells.
2. Eine Aktivität liegt im bereits durchlaufenen und eine im noch nicht durchlaufenen Bereich des Prozessmodells.
3. Beide Aktivitäten liegen im bereits durchlaufenen Bereich des Prozessmodells.

**MQSeries WF**:
Auf laufende Prozessinstanzen werden diese Änderungen nicht propagiert.

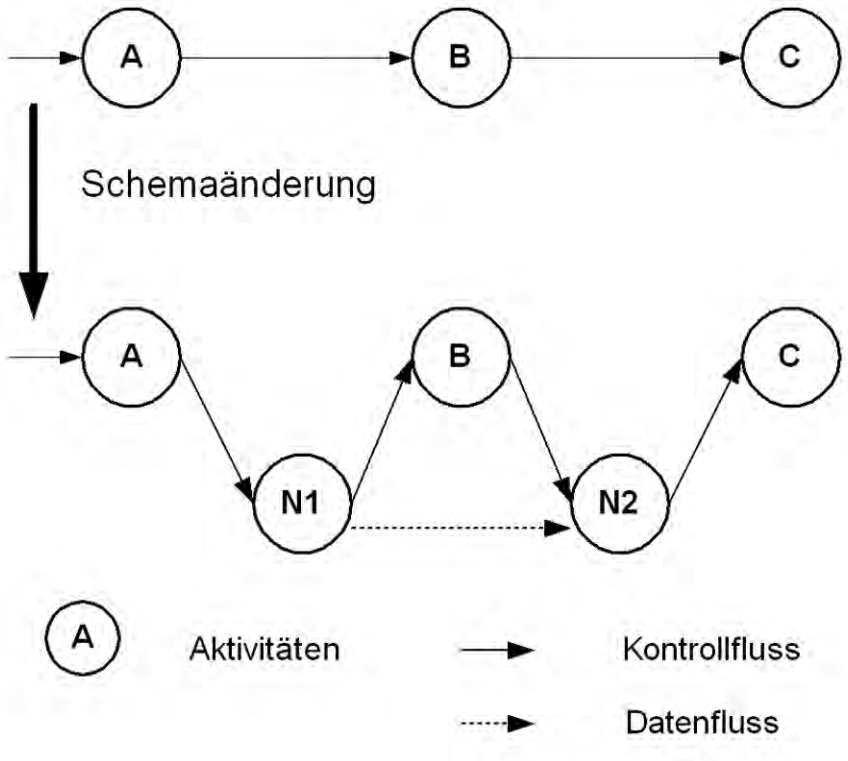



Abbildung 4.9: Einfügen neuer Aktivitäten in das Prozessmodell

**Staffware**:
Im folgenden werden die Reaktionen von Prozessinstanzen beschrieben, bei der jeweils eine der Aktivitäten A, B oder C zum Zeitpunkt der Schemaänderung aktiviert war. Es wird davon ausgegangen, daß die eingefügte Aktivität N1 die Eingabe einer neuen Variable a (Typ:*long*) fordert. Mit dieser und einer bereits bekannten Variable b (Typ:*long*) soll in N2 dem Bearbeiter ein neuer Wert ($a$=$a$*$b$) dargestellt werden.

- Fall 1 : Aktivität A ist aktiviert
  Alle eingebrachten Änderungen liegen im Kontrollflussgraph hinter der Aktivität A, also im noch nicht durchlaufenen Bereich. Somit können alle eingebrachten Änderungen im weiteren Verlauf der Prozessinstanz berücksichtigt werden. Nach der Bearbeitung von Aktivität A wird Aktivität N1 aktiviert. Das neue Prozessmodell wird fehlerfrei durchlaufen.

- Fall 2 : Aktivität B ist bereits aktiv
  Die neu erstellte Aktivität N1 liegt im Kontrollflussgraph vor der bereits aktivierten Aktivität B und wird nicht mehr ausgeführt. Der Datenfluss von Aktivität N1 nach N2 kann dadurch nicht stattfinden. Dadurch bleibt das Formularfeld, welches mit dem Ausgabe-Wert a von Aktivität N1 berechnet werden sollte, leer. Die Auswirkungen dieses Verhaltens sind unabsehbar, wenn Felder die im weiteren Prozessverlauf verarbeitet werden auf diese Weise keine Daten erhalten.
  Eine Warnung oder Anmerkung, warum das Feld leer ist, erhält der Bearbeiter nicht.

- Fall 3 : Aktivität C ist bereits aktiv
  Alle Änderungen liegen hinter der aktiven Aktivität. Demzufolge ist der weitere Verlauf des alten und neuen Prozessmodells identisch und die Prozessinstanz kann ungestört weiterlaufen.

Die Gefahr, Daten zu verlieren, ist sehr groß. Änderungen an Prozessmodellen sollten deshalb nur nach eingehender Betrachtung ihrer Auswirkungen gespeichert werden. Dies gilt vor allem für den Fall, daß aktive Instanzen des Prozessmodells existieren.

**Szenario 9** (*Löschen von Aktivitäten*)
Eine Aktivität wird aus dem Prozessmodell entfernt (siehe Abbildung 4.10). Von Bedeutung ist hierbei die Reaktion von Prozessinstanzen, in denen die gelöschte Aktivität aktiviert oder laufend war. Zu überprüfen ist, wie bzw. ob der Bearbeiter einer solchen Aktivität über die Änderungen informiert wird.

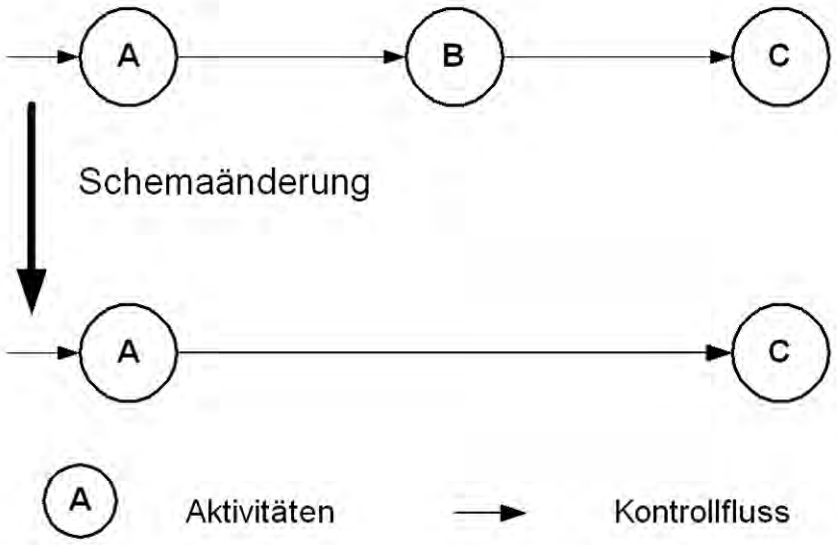



Abbildung 4.10: Löschen einer Aktivität des Prozessmodells

**MQSeries WF**:
Auf laufende Prozessinstanzen werden diese Änderungen nicht propagiert.

**Staffware**:
Vorausgesetzt wird, daß keine Datenabhängigkeiten zwischen der zu löschenden Aktivität B und einer Nachfolge-Aktivität existieren. Für diesen Fall bemerkt der Bearbeiter einer Instanz, dessen Aktivität B noch nicht aktiviert ist, nichts von der Schemaänderung.
Eine Aktivität kann aber auch aus einem Prozessmodell gelöscht werden, wenn sie in einer oder mehreren Instanzen aktiviert ist. Beim Start einer bereits gelöschten Aktivität erhält der Bearbeiter die in Abbildung 4.11 dargestellte Fehlermeldung.

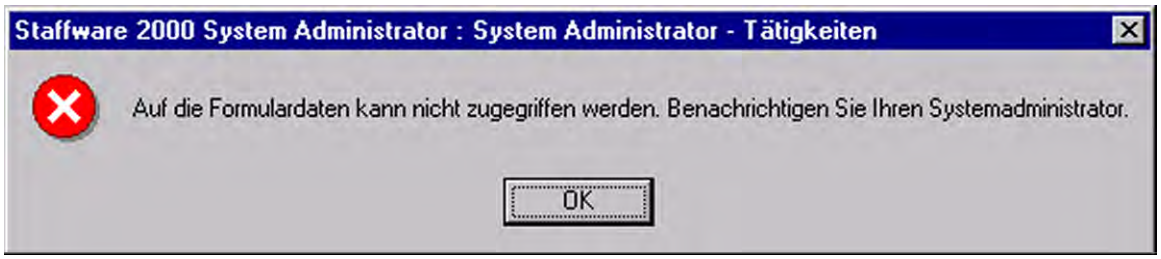



Abbildung 4.11: Starten einer gelöschten Aktivität

Das zugehörige Workitem bleibt in der Arbeitsliste des Bearbeiters. Ein Systemadministrator kann die Prozessinstanz entweder löschen oder er kann das Prozessmodell wieder in seinen alten Zustand (erneutes Einfügen der Aktivität B) versetzen.
Die weitere Ausführung solcher Instanzen ist somit nicht ohne Eingriff in das Prozessmodell möglich. Dieses Verhalten erzeugt einen unverhältnissmäßig hohen Arbeitsaufwand, der in der Praxis intolerabel ist.

**Szenario 10** (*Abänderung des Kontrollflusses*)
In Abbildung 4.12 wurde die Reihenfolge der Aktivitäten im Prozessmodell, abgeändert.
Ist für eine Prozessinstanz zum Zeitpunkt dieser Änderung eine der Aktivitäten B, C oder D aktiv, kann es zu einem unerwünschten Verhalten kommen. Die bereits ausgeführte Aktivität B könnte nochmals aktiviert werden oder die Aktivität D wird nie aktiviert.

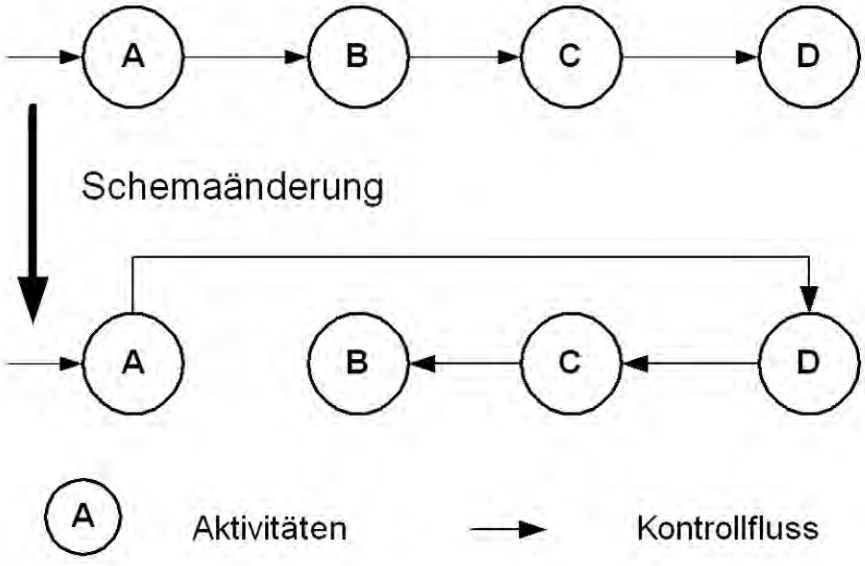



Abbildung 4.12: Abänderung des Kontrollflusses

**MQSeries WF**:
Auf laufende Prozessinstanzen werden diese Änderungen nicht propagiert.

**Staffware**:
Eine Änderung der Reihenfolge von Aktivitäten ist jederzeit möglich. Hier ist ebenfalls das neue Prozessmodell sofort nach der Speicherung gültig, und die Ausführung der Aktivitäten entspricht

dem Schema des neuen Prozessmodells. Ist beispielsweise zum Zeitpunkt der Schemaänderung die Aktivität C aktiviert, so wird nach deren Ausführung die Aktvität B erneut aktiviert.
Probleme treten auch auf, wenn Datenabhängigkeiten zwischen den Aktivitäten bestehen. Dies führt dann zu dem in Szenario 8 bereits beschriebenen Verhalten.

#### 4.2.3.2 Organisationsmodell

**Szenario 11** (*Ändern der Bearbeiterzuordnung*)
Die Bearbeiterzuordung für eine oder mehrere Aktivitäten wird geändert. Ebenso wie in Szenario 9, sind hier die Prozessinstanzen von Bedeutung, deren aktivierte Aktivitäten von der Schemaänderung betroffen sind. Bei diesen Aktvitäten kann eine Nachevaluierung der Bearbeiterzuordnung stattfinden oder die alte Bearbeiterzuordnung gültig bleiben.

**MQSeries WF**:
Auf laufende Prozessinstanzen werden diese Änderungen nicht propagiert.

**Staffware**:
Die Auswertung der Bearbeiterzuordnung erfolgt in Staffware bei der Aktivierung einer Aktivität. Dementsprechend bleiben Workitems in den einmal zugeordneten Arbeitslisten, auch wenn die Bearbeiterzuordnung für die zugehörige Aktivität geändert wird. Die Bearbeiterzuordnung für noch nicht aktivierte Aktivitäten kann aber jederzeit abgeändert werden. Daraus können Probleme entstehen, falls beispielsweise die erste und letzte Aktivität vom selben Bearbeiter ausgeführt werden sollen. Werden die Bearbeiterzuordnungen für diese beiden Aktivitäten geändert, kommt es zu Fehlern bei laufenden Instanzen. Bei diesen Instanzen wird dann die erste und die letzte Aktivität von verschiedenen Personen ausgeführt. In der Praxis kann dies zu einer ungewollten Unterbrechung in der Ausführung eines Geschäftsprozesses führen (z.B. weil dem Bearbeiter der letzten Aktivität das Wissen zum Ausführen dieser Aktivität fehlt). Eine Lösung kann das Übertragen oder Weiterleiten von Workitems darstellen.

**Szenario 12** (*Entfernen von verwendeten Personen*)
Eine Person soll aus dem WFMS entfernt werden. Diese Person kann Workitems aus aktiven Prozessinstanzen in ihrer Arbeitsliste haben.

**MQSeries WF**:
Wie bereits besprochen, ist das Entfernen von Personen aus der Buildtime-Datenbank zu jeder Zeit möglich (siehe Szenario 6). Um jedoch eine solche Änderung in die Runtime-Datenbank importieren zu können, muß die entsprechende Person vor dem Löschen aus der Buildtime-Datenbank entsprechend markiert (*mark for deletion*) und die Änderung in eine FDL-Datei exportiert werden. Das Importieren dieser Änderung in die Runtime-Datenbank wird allerdings mit Fehlermeldung abgewiesen, wenn die zu löschende Person:

- noch Workitems zugewiesen hat oder
- am System angemeldet ist.

Außerhalb dieser Fälle kann eine Person gelöscht werden.

**Staffware**:
Eine Person kann im Administration-Manager gelöscht werden obwohl:

- ihr Workitems zugewiesen sind.

- sie am System angemeldet ist.

Ist eine Person am System angemeldet während sie gelöscht wird, hat dies keine direkten Auswirkungen. Sie kann ihre Workitems bearbeiten und neue Instanzen starten. Jedoch erhält sie keine neuen Workitems mehr zugewiesen. Bei der Auflösung von Bearbeiterzuordnungen wird diese Person nicht mehr berücksichtigt. *Workitems* die diese Person noch als Bearbeiter zugewiesen haben, werden dem System-Administrator in einer gesonderten Arbeitsliste angezeigt (siehe Abbildung 4.13).
Meldet sich diese Person am System ab, kann sie sich nicht mehr anmelden.

Abbildung 4.13: Nicht zustellbare Workitems in Staffware

**Szenario 13** (*Nicht erfüllbare Bearbeiterzuordnungen*)
Für eine oder mehrere Aktivitäten wurden zur Modellierzeit dynamische Bearbeiterzuordnungen getroffen. Die Auflösung kann zur Laufzeit problematisch sein, falls keine Person den getroffenen Anforderungen entspricht. Dieser Fall ist möglich, wie Szenario 7 bereits gezeigt hat.

**MQSeries WF**:
Kann einer Aktivität bei ihrer Aktivierung keine Person, entsprechend der zur Modellierzeit vorgenommenen Bearbeiterzuordnung, zugewiesen werden, erhält der Systemadministrator das Workitem dieser Aktivität zugeteilt.

**Staffware**:
Wie in Szenario 7 bereits dargestellt, verknüpft Staffware die in der Bearbeiterzuordnung gemachten Angaben nicht, sondern wertet sie nach Person, Gruppe und Rolle getrennt aus (ODER-Verknüpfung der einzelnen Bearbeitermengen). Die Workitems werden dementsprechend zugeordnet.
Somit ist das obige Szenario in Staffware nicht modellierbar.

#### 4.2.3.3 Löschen von Prozessmodellen

**Szenario 14** (*Entfernen eines Prozessmodells* )
Ein Prozessmodell soll aus der Laufzeitumgebung entfernt werden. Da hierzu laufende Prozessinstanzen oder in Ausführung befindliche Workitems existieren können sollte der Benutzer entsprechend gewarnt werden. Das Löschen von Prozessmodellen mit laufenden Instanzen hat ebenso Nachteile, wie eine Begrenzung des Löschens auf Prozessmodelle ohne aktive Instanzen. Im ersten Fall kann ein Fehler schwerwiegende Folgen haben, im zweiten Fall kann das Löschen eines Prozessmodells sehr aufwendig werden, wenn sehr viele Prozessinstanzen existieren.

**MQSeries WF**:
Es exitieren "scheinbar" zwei Möglichkeiten zum löschen eines Prozessmodells:

- `Import-/Export-Dienstprogramm`
  Ein Prozessmodell kann in der Builtime-Komponente mit einem entsprechenden Status versehen werden (*mark for deletion*). Nach Export dieser Änderung in eine FDL-Datei, kann sie in die Runtime importiert werden.
  In Abbildung 4.14 wird das Prozessmodell `subprodIN` durch importieren einer FDL-Datei gelöscht. Erneute Versuche die FDL-Datei nochmals zu importieren wurden mit der Fehlermeldung "PROCESS sobprocIN ist nicht vorhanden" verhindert.

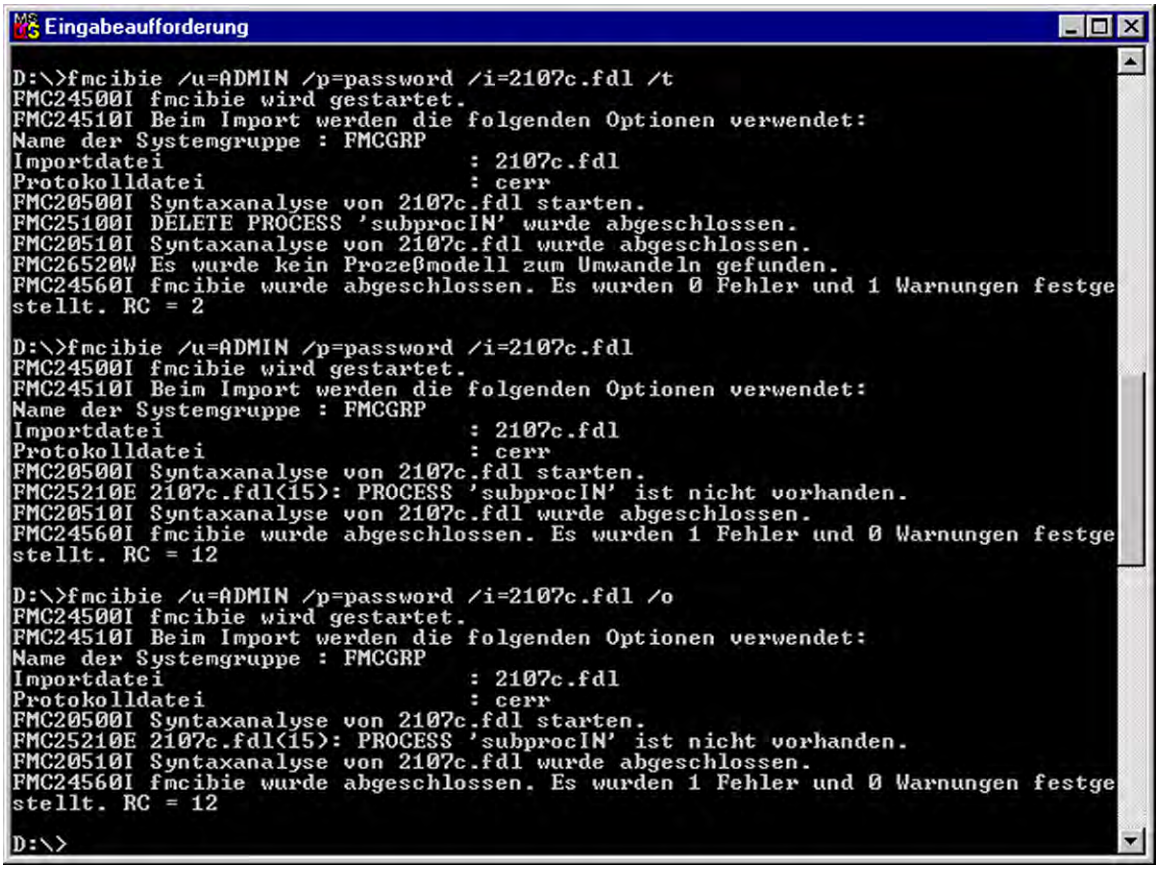



Abbildung 4.14: Ausgabe des Export-/Import-Dienstprogramms

Diese Ausgabe hat jedoch keinen Einfluss auf das weitere Verhalten von MQSeries WF. Die bereits gestarteten Instanzen des gelöschten Prozessmodells laufen ohne Fehlermeldung auf der Grundlage des bisherigen Prozessmodells weiter und das zugehörige Prozessmodell bleibt weiterhin verfügbar. Es können sogar neue Instanzen angelegt und ausgeführt werden.

- `Löschen des Prozessmodells im Standard-Client`
  Ein Prozessmodell kann im Standard-Client zu jeder Zeit gelöscht werden. Aktive Instanzen laufen ohne eine Beeinflussung zu Ende. Das gelöschte Prozessmodell verschwindet aber sofort aus der Liste der Prozessmodelle, und es können keine neuen Prozessinstanzen mehr erstellt werden.
  Bei dieser Variante ist der Prozessmodellierer verantwortlich dafür, daß das Prozessmodell ebenfalls aus der Buildtime-Datenbank gelöscht oder entsprechend markiert wird.

**Staffware**:
Damit ein Prozessmodell im GWD gelöscht werden kann und nicht mehr zur Instanziierung zur

Verfügung steht, muß eine Vorbedingung erfüllt sein: In der Instanzverwaltung dürfen weder aktive noch beendete (komplett ausgeführte) Instanzen dieses Prozessmodells vermerkt sein.
Da eine Instanz bei ihrem Start automatisch einen Eintrag in der Instanzverwaltung erzeugt, können in Staffware keine Prozessmodelle gelöscht werden, von denen sich noch Instanzen in Ausführung befinden. Dadurch wird ein geplantes Vorgehen auch beim Löschen von Prozessmodellen erzwungen. Allerdings kann es sehr aufwendig werden, wenn dennoch ein Prozessmodell mit vielen laufenden Instanzen gelöscht werden soll.

#### 4.2.3.4 Subprozesse

Eine besondere Problematik ergibt sich, wenn ein Prozessmodell als Subprozess in anderen Prozessmodellen verwendet werden kann. Änderungen an solchen (Sub-)Prozessmodellen können das Zusammenspiel (z.B. Datenübergabe) mit dem Hauptprozess "stören".

**Szenario 15** (*Änderungen an Subprozessen*)
Ein Subprozess empfängt bei seiner Instanziierung Daten seines Superprozesses und gibt nach seiner Bearbeitung Daten an diesen zurück. Nach einer Änderung kann der Superprozess zu viele, zu wenige oder falsche Daten an den Subprozess liefern.

**MQSeries WF**: Zur Übernahme von Daten aus seinem Superprozess verfügt ein Subprozess über eine Quelle - sie gibt Daten des Superprozesses an beliebige Aktivitäten des Subprozesses weiter - und eine Senke, die Daten an den Superprozess zurückgibt (siehe Kapitel 2). Wird die Quelle oder Senke im Subprozess entfernt, kann keine korrekte Datenübergabe zwischen den Prozessmodellen erfolgen.
Der Prozessmodellierer wird nicht gewarnt und es erfolgt zur Laufzeit kein Hinweis auf einen Fehler. Fehlermeldungen, die aus einer falschen Datenübergabe resultieren können, sind von der Programmierung der am Subprozess beteiligten Anwendungsprogramme abhängig.

**Staffware**:
Bei der Erstellung eins Subprozesses können zwischen Super- und Sub-Prozess die Übergabe- und Rückgabefelder definiert werden. Dabei können nur Felder vom selben Typ (z.B. long) aufeinander abgebildet werden.
Werden während der Ausführung eines Subprozesses dessen Formular-Felder vollständig gelöscht und durch andere ersetzt, kann keine korrekte Rückgabe an den Superprozess erfolgen. In diesem Fall behält der Superprozess die alten Werte der Felder auch nach Ausführung des Subprozesses bei. Grund dafür ist, daß Staffware das Löschen der Felder im Subprozess bemerkt und daher das Datenmapping zwischen Super- und Subprozess ebenfalls löscht. Dabei erfolgt keine Meldung an den Prozessmodellierer. Dieses Verhalten ist gefährlich und suggeriert dem Bearbeiter ein fehlerfreie Ausführung des Prozessmodells. Der Bearbeiter hat keine Möglichkeit zu erkennen, daß ein Fehler vorliegt. Es erfolgt keine Warnung, und der Superprozess kann weiterhin instanziiert werden. Lediglich wenn der Prozessmodellierer das Datenmapping zwischen Super- und Subprozess erneut betrachtet kann er merken, daß dieses automatisch beim Löschen der Formular-Felder des Subprozesses entfernt wurde.

**Szenario 16** (*Entfernen des Subprozsses* )
Ein Prozessmodell, welches als Subprozess verwendet wird, soll gelöscht werden. Falls dies möglich ist, sollten weitere Instanziierungen des Superprozesses, der das gelöschte Prozessmodell als Subprozess eingebunden hat, verhindert werden.

**MQSeries WF**:
Ein Prozessmodell das als Subprozess verwendet wird und aktivierte Aktivitäten aufweist, kann gelöscht werden. Das weitere Verhalten entspricht dem beim Löschen einfacher Prozesse (siehe Szenario 14).

**Staffware**:
Ein Subprozess kann gelöscht werden, sobald sein Superprozess gelöscht werden kann (siehe Szenario 14). Werden nach dem Löschen des Subprozesses noch weitere Instanzen des Superprozesses erstellt, werden diese an der Stelle des Subprozess-Aufrufs beendet. Der Superprozess kann nicht weiter bearbeitet werden. Dem Bearbeiter verschließt sich dieses Verhalten komplett. Der Systemadministrator kann in der Instanzverwaltung einen Fehler (siehe Abbildung 4.8) beim Aufruf des Subprozesses erkennen.

### 4.2.4 Versionierung

Sobald Änderungen an einem Prozessmodell durchgeführt werden, stellt sich die Frage nach einer Versionierung. So kann es nötig sein, daß verschiedene Versionen eines Prozessmodells gleichzeitig gültig sein sollen (siehe Beispiel 4.1) oder ein älteres Prozessmodell aus Sicherheitsgründen weiterhin verfügbar sein soll.

**Beispiel 4.1**: *(mehrere Prozessmodell-Versionen gleichzeitig verfügbar)*
Das komplette Prozessmodell einer Einkaufsabteilung einer Firma ist auf DM eingestellt. Zum 01.01.2002 muß dieser Prozess auch in Euro abgearbeitet werden können. Allerdings muß bis zum 01.03.2002 noch mit dem alten, auf DM basierten Prozessmodell gearbeitet werden können. Das Prozessmodell kann dementsprechend nicht einfach abgeändert werden.

MQSeries WF und Staffware bieten keine Unterstützung zur gezielten Versionierung von Prozessmodellen.

## 4.3 Bewertung

Sollen unternehmenskritische Prozesse durch ein WFMS unterstützt werden, bedingt dies ein besonders robustes und konsistentes Laufzeit-Verhalten. Dies bedeutet, daß in der Ausführung einer Prozessinstanz keine unvorhersehbaren oder ungewollten Effekte auftreten, wie z.B.

- Fehlverhalten (z.B. Absturz) der Anwendungsprogramme durch fehlende oder falsche Daten (siehe Szenario 1)
- Blockaden in der Ausführung einer Prozessinstanz (siehe Szeario 9)
- Zuordnung von Workitems an nicht autorisierte Personen (siehe Szenario 6)

MQSeries WF und Staffware sind auf dem richtigen Weg, dieses Ziel zu erreichen. Die Trennung von Prozesslogik und Anwendungsprogramm bringt viele Vorteile (siehe Kapitel 1), erhöht aber auch die Anzahl potentieller Fehlerquellen (Datenübergabe). Wie in den verschiedenen Szenarien gezeigt, existieren noch viele weitere potentielle Fehlerquellen. Hier liegen auch die Probleme von MQSeries WF und Staffware. Bei beiden WFMS sind die Überprüfungen der Einstellungen von Prozessmodellen bei weitem noch nicht komplett. Es können, wie in einigen Szenarien beschrieben, noch immer Fehler auftreten.
Daher sollte sowohl MQSeries WF wie auch Staffware noch nicht zur Unterstützung unternehmenskritischer Prozesse eingestzt werden.

# Kapitel 5

# Schnittstellen der Systeme

Durch die Einführung standardisierter Schnittstellen (*Interfaces*) soll es Anwendern eines WFMS ermöglicht werden, bestimmte Module eines Herstellers (= Teilsysteme eines WFMS, z.B. Buildtime-Komponente) durch die Module anderer Hersteller zu ersetzen bzw. zu ergänzen.
Diese Bemühungen wurden in den letzten Jahren am stärksten von der Workflow Management Coalition (`WFMC`) geprägt (siehe [WFMC95] und [Fis01]). Die WFMC wurde als nicht kommerzieller (*Non-Profit*) Interessensverband von WFMS-Anbietern und -Anwendern im Jahr 1993 gegründet. Mit mehr als 200 Mitgliedern in über 25 Ländern entwickelte sich die WFMC rasch zum führenden Organ in diesm schnell expandierenden Markt-Segment. Ziel ist es, einen umfassenden Standard im Bereich der WFMS zu etablieren, der

- die Interoperabilität zwischen Produkten verschiedener Hersteller ermöglicht,
- die Integration von WF-Anwendungen in andere IT-Dienste (z.B. Dokumentenmanagement) verbessert und
- die Verbreitung der Workflow Technologie fördern soll

(siehe [WFMC95]). Auf die interne Architektur eines WFMS wird in diesen Standards nicht eingegangen.
Die folgenden Abschnitte geben einen Überblick über die Standardisierungsbemühungen der WFMC und die bestehende Konformität bzw. Abweichung von MQSeries WF und Staffware.
In Abschnitt 5.1 wird das Referenzmodell der WFMC und in Abschnitt 5.2 weitere Workflow-Standards vorgestellt. Die Konformität von MQSeries WF und Staffware zu diesen Standards wird in den Abschnitten 5.3 und 5.4 beschrieben. Ein abschließender Vergleich der beiden WFMS erfolgt in Kapitel 5.5.

## 5.1 Das Referenzmodell der WfMC

Das Referenzmodell der WfMC (vgl. Abbildung 5.1) definiert verschiedene Schnittstellen zwischen dem Workflow Enactment Service und anderen Komponenten eines WFMS. Dabei werden fünf Schnittstellen zum `Workflow Enactment Service` unterschieden. Als `Workflow Enactment Service` bezeichnet die WFMC die zentrale Komponente eines WFMS, bestehend aus einer oder mehrerer WF-Engines [WFMC95]. Tabelle 5.1 gibt einen Überblick über die verschiedenen Schnittstellen:

| **Schnittstelle** | **WFMS-Komponente** | **Funktionalität** |
|---|---|---|
| `Schnittstelle 1` | Buildtime-Komponente | Import/Export von Prozessmodellen |
| `Schnittstelle 2` | Clients | Bereitstellung eines API zur Client-Programmierung |
| `Schnittstelle 3` | Anwendungs-programme | Bereitstellung eines API zur Einbindung von Anwendungsprogrammen |
| `Schnittstelle 4` | Andere WFMS | Interoperabilität zwischen WFMS |
| `Schnittstelle 5` | Administrations und Überwachungs Tools | Bereitstellung eines API zur Programmierung von Administrations- und Überwachungswerkzeugen |

Tabelle 5.1: Die fünf Schnittstellen der WFMC

Die Schnittstellen werden als API (*Application Programming Interface*) oder in Form von Austauschformaten implementiert. Dadurch werden die Dienste eines WFMS nutzbar und die Interaktion zwischen `Workflow Enactment Service` und den WFMS-Komponenten geregelt. In diesem Kontext wird normalerweise eine `API` als `WAPI` (*Workflow API*) bezeichnet.

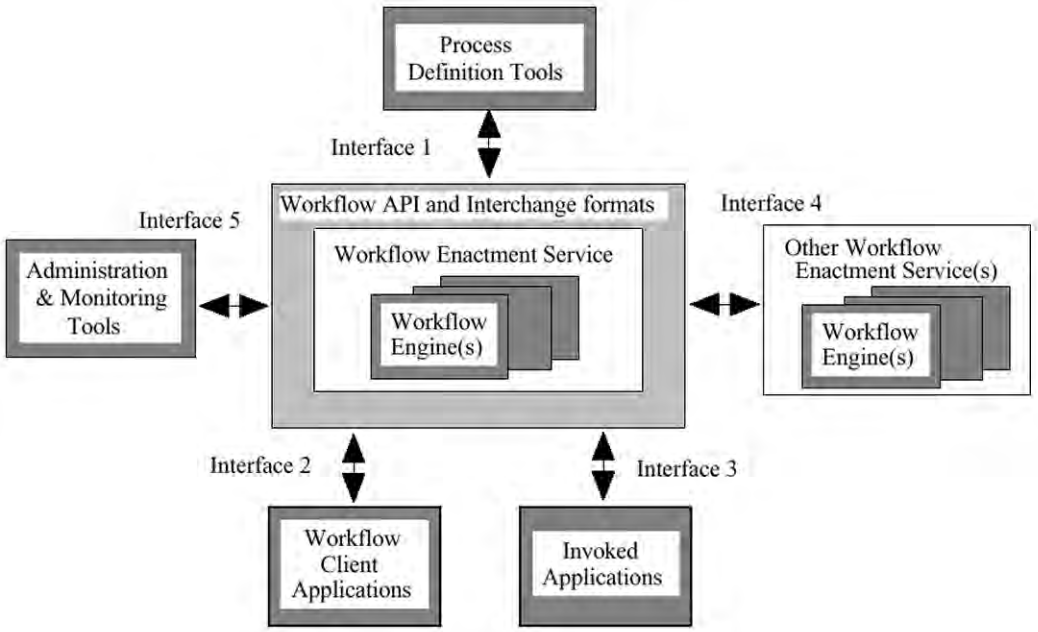



Abbildung 5.1: Das Workflow-Referenzmodell - Komponenten und Schnittstellen (aus [WFMC95])

Ein Überblick über die von der WFMC veröffentlichten Standard-Dokumente wird in Abbildung 5.2 in Anlehnung an [WFMC01a] vermittelt. Der aktuelle Stand der WFMC-Veröffentlichungen kann unter [WFMC01c] betrachtet werden. Für die Schnittstellen eins, vier und fünf existieren jeweils selbständige Dokumente. Die Schnittstellen zwei und drei sind in einem Dokument ([WFMC98b]) zusammengefaßt. In den folgenden Abschnitten werden die Anforderungen dieser Schnittstellen jeweils getrennt betrachtet. Das Dokument *Workflow Management Facility Spezifikation* wurde in Zusammenarbeit der OMG und der WFMC aufgestellt. Es wird in Abschnitt 5.2 vorgestellt.

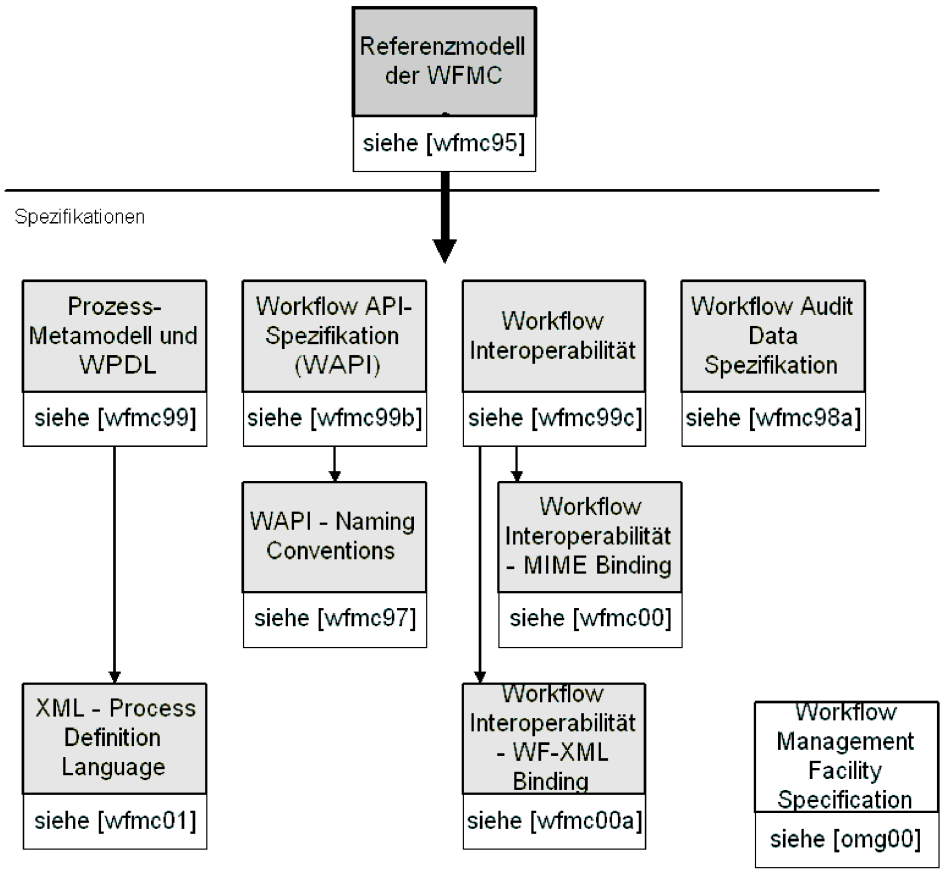



Abbildung 5.2: Workflow-Standards und verwandte Dokumente

### 5.1.1 Import/Export von Prozessmodellen (Interface 1)

Modellier- und Laufzeit-Komponenten der WFMS werden über diese Schnittstelle miteinander verbunden. Dazu wurde ein allgemeines Austauschformat spezifiziert, welches den Tranport von Prozessmodellen zwischen Buildtime- und Runtime-Komponenten ermöglicht. Diese Schnittstelle wird als `Schnittstelle für den Import/Export von Prozessmodellen` bezeichnet.
Der Wunsch Prozessmodelle austauschen zu können, als Ganzes oder in Teilen, hat eine Vielzahl von Gründen [WFMC99]. Die wichtigsten sind:

- Es können verschiedene Werkzeuge zur Modellierung und Ausführung von Prozessmodellen verwendet werden. Damit kann für jeden Einsatzbereich das beste Werkzeug genutzt werden.
- Alle Prozessmodelle können in einer Komponente verwaltet werden und sind für verschiedenste Programme und für die Runtime-Komponenten erreichbar.
- Der Transfer von Prozessmodellen zwischen WFMS kann nötig sein, um die Interoperabilität eines Prozesses (d.h. mehrere WFMS sind an der Ausführung beteiligt) zu ermöglichen.

Komponenten für die Verwendung dieser Schnittstelle sind die Buildtime- und Runtime-Komponenten von WFMS sowie (Geschäfts-) Prozessmodellierungs-, (Geschäfts-) Prozesssimulations- und WF-Analyse-Tools. Diese Komponenten können von verschiedenen Herstellern stammen. Der primäre Gedanke der Schnittstelle 1 gilt der Trennung von Buildtime- und Runtime-Komponente. Ein Austausch von Prozessmodellen zwischen Buildtime-Komponenten oder Runtime-Komponenten ist möglich und sinnvoll [WFMC99].

Die Spezifikation der Schnittstelle 1 untergliedert sich in zwei Teile [WFMC99]:

1. Ein `Prozess-Metamodell`, das die Hauptbestandteile eines Prozessmodells, ihre Beziehungen und ihre Merkmale beschreibt.

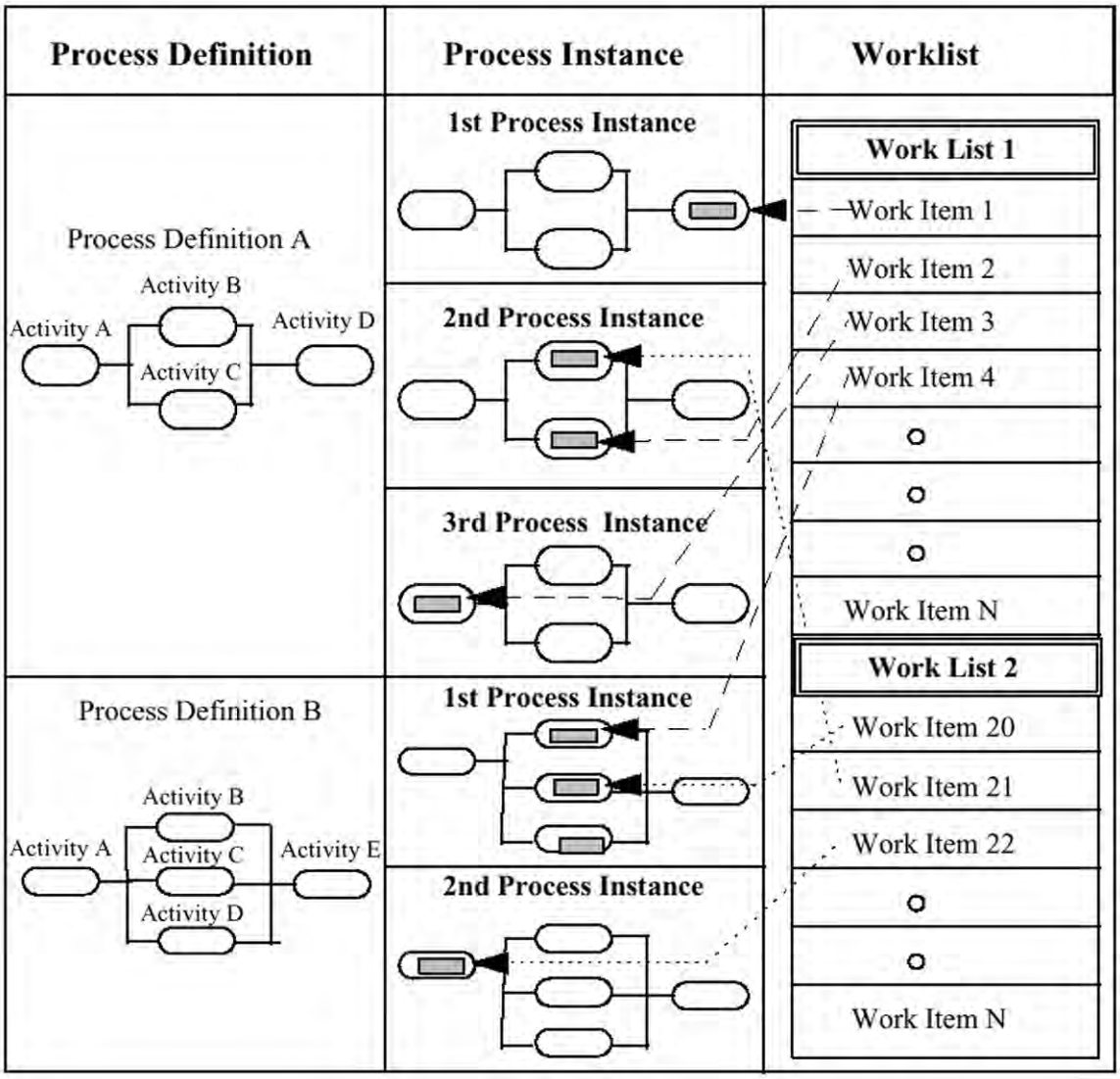
2. Verfahren zum Austausch von Prozessmodellen:

   - Eine `WPDL` (*Workflow Process Definition Language*) als formale Sprache für die Definition und den Austausch von Prozessmodellen mittels Dateien oder gemeinsamen Speicher.
   - Ein `API` innerhalb der WAPI zur Manipulation von Prozessmodellen. Die Funktionen der WAPI für diese Schnittstelle werden unter dem Begriff *Workflow Definition Functions* zusammengefasst. Dazu gehören beispielsweise Funktionen zum Erstellen einer Aktivität oder Festlegung einer Transitionsbedingung. Alle Funktionen der WAPI sind in [WFMC98b] beschrieben.

Durch die Unterteilung in Prozess-Metamodell, WPDL und WAPI soll eine größtmögliche Flexibilität in Bezug auf neue Technologien zum Austausch von Prozessmodellen erhalten werden.
Das in Abschnitt 2.1.3.2 beschriebene Prozess-Metamodell von MQSeries WF entspricht im wesentlichen dem der WFMC. Es wird deshalb an dieser Stelle nicht nochmals dargestellt. Die grundlegende Funktionalität einer WPDL wurde bereits in Kapitel 2 beschrieben. Eine beispielhafte Notation eines Prozessmodells mittels WPDL und weitere Informationen sind im Anhang C.1 einzusehen.
Die Implementierierung der Schnittstelle 1 kann, abhängig von den verwendeten Komponenten, Client- oder Serverseitig erfolgen [WFMC99]. Ein WFMS-Hersteller könnte die Import/Export-Funktion von Prozessmodellen nur durch die Runtime-Komponente anbieten. Dazu müssten Buildtime- und Runtime-Komponente auf dem selben Speichermedium arbeiten. Allgemein kann es als sinnvoll erachtet werden, wenn jede Komponente über die Möglichkeit zum Import/Export von Prozessmodellen verfügt. Dadurch wird eine zwingende Kopplung der Komponenten verhindert.
Der Austausch von Prozessmodellen mittels `XML-Process Definition Language` (siehe Abbildung 5.2 und [WFMC01]) befindet sich noch in der Entwicklungsphase. Er wird in *Kapitel 7* kurz betrachtet.

### 5.1.2 Client- und Anwendungsprogrammierung (Interface 2 und 3)

Ein Client-Programm ist die graphische Benutzer-Schnittstelle eines WFMS zu den Personen, die dieses WFMS verwenden. Es bietet eine Bedieneroberfläche, welche das Erzeugen und Starten von Prozessinstanzen ermöglicht und über die zur Ausführung anstehende Arbeiten (Workitems) gestartet, beendet, wiederholt oder unterbrochen werden können. Zu diesem Zweck stellt der Client eine Übersicht über die Prozessmodelle, Prozessinstanzen und Workitems des WFMS entsprechend den Benutzer-Rechten bereit. In Abbildung 5.3 werden die verschiedenen Sichten, die ein Client einem Benutzer anbieten sollte, dargestellt.
Wird ein Workitem im Client gestartet, so wird automatisch das für die zugehörigen Aktivität zur Modellierzeit hinterlegte Anwendungsprogramm aufgerufen. Ein Anwendungsprogramm ist eine vom WFMS getrennte Software-Komponente, durch die Daten dargestellt, eingelesen und manipuliert werden können.
Ursprünglich betrachtete die WFMC die Client- und Anwendungsprogrammierung getrennt voneinander. Während der Entwicklung beider Schnittstellen wurde ihre enge Verbindung deutlich. Inzwischen existiert für beide Schnittstellen nur noch eine Schnittstellenspezifikation (siehe Abbildung 5.2 und [WFMC98b]). Die folgenden zwei Abschnitte betrachten die Schnittstellen dennoch getrennt, um die verschiedenen Anforderungen und die Unterteilung der WAPI deutlich zu machen.




Prozessmodell = *Process Defninition*
Prozessinstanz = *Process Instance*
Arbeitsliste = *Worklist*

Abbildung 5.3: Zusammenhang von Prozessmodellen, -instanzen und Arbeitslisten (aus [WFMC99b])

#### 5.1.2.1 Client-Programmierung (Interface 2)

Die von der WFMC festgelegte Programmier-Schnittstelle zu Client-Anwendungen eines WFMS (*Workflow Client Applications*) ermöglicht die Realisierung angepasster Benutzerschnittstellen. Außerdem ermöglicht sie die Programmierung von WF-Client Anwendungen, welche die Funktionalität von WFMS (dem Workflow Enactment Service) in Anspruch nehmen. Dabei kann eine WF-Client Anwendung Teil eines ganzen WFMS sein, oder von einem Programmierer speziell an besondere Umstände (z.B. Vorgegebene Oberflächen, Kommunikation mit mehreren WfMSen verschiedener Hersteller) angepasst werden.

Die genannten Anforderungen werden durch die WAPI bereitgestellt. Die Funktionalitäten der WAPI für diese Schnittstelle sind (siehe auch Abbildung 5.4) [WFMC98b]:

- Verbindungsaufbau (*Connection Functions*)
- Prozess-Kontrollfunktionen (*Process Control Functions*), z.B. Start eines Prozessmodells.
- Prozess-Statusfunktionen (*Process Status Functions*), z.B. Erstellen einer Liste von Prozessinstanzen mit bestimmten Eigenschaften (*Suspended, Completed, etc.*).
- Steuerungsfunktionen für Workitems (*Activity Control Functions*), z.B. Statusänderung eines Workitems.
- Workitem-Statusfunktionen (*Activity Status Functions*), z.B. Erstellen einer Liste von Workitems mit bestimmten Eigenschaften (*Suspended, Completed, etc.*).
- Steuerungsfunktionen für Arbeitslisten (*Worklist Functions*), z.B. Erstellen einer Liste von Workitems für eine bestimmte Person.
- Administrationsfunktionen (*Administration Functions*), z.B. Ändern des Status aller Prozessinstanzen eines Prozessmodells.

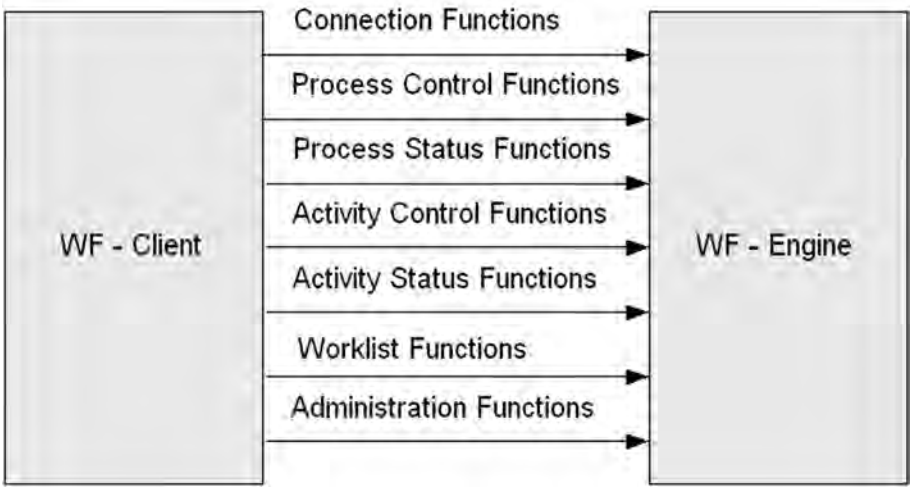



Abbildung 5.4: WAPI-Funktionen der Schnittstelle 2

In [WFMC97] werden WAPI-Namenskonventionen vorgestellt. Dieses Dokument enthält allgemeine *Header files* für die Programmierung in C. Ein Ziel der Namenskonvention für die WAPI war die erhöhte Lesbarkeit von WFMS-Implementationen.

#### 5.1.2.2 Anwendungsprogrammierung (Interface 3)

Die Schnittstelle 3 (*Invoked Applications*) definiert den Standard für den Aufruf von Anwendungskomponenten. Im Zusammenhang mit der Ausführung von Workitems. Dabei wird über diese Schnittstelle das Anwendungsprogramm nicht direkt aufgerufen, sondern lediglich ein `Tool Agent` (ein spezieller Anwendungs-Treiber). Der `Tool Agent` ist für den Start, den Stop, den Austausch von Worflow- bzw. Anwendungsspezifischen Daten und für die Kontrolle des Status (*started, stopped, suspended, terminated etc.*) eines Anwendungsprogrammes zuständig. Zwischen `Tool Agent` und Schnittstelle 3 wird eine bidirektionale Verbindung zum Empfangen und Senden von Anfragen erstellt (siehe Abbildung 5.5).

Ein Anwendungsprogramm ist eine Software-Komponente, die in eine oder mehrere Aktivitäten der selben oder verschiedener Prozessmodelle eingebunden wird. Zur Laufzeit wird diese Komponente beim Start eines zur Aktivität gehörigen Workitems aufgerufen, um dem Anwender die Bearbeitung seiner Aufgabe zu ermöglichen. Dabei können zwischen dem WFMS und den gerufenen Anwendungsprogrammen mittels `Tool Agent` Parameterdaten ausgetauscht werden.
Die Funktionen der WAPI für die Schnittstelle 3 ermöglichen[WFMC98b]:

Anwendungseinbindung (*Application Invocation Functions*), z.B. Start eines Anwendungsprogramms über einen `Tool Agent`.

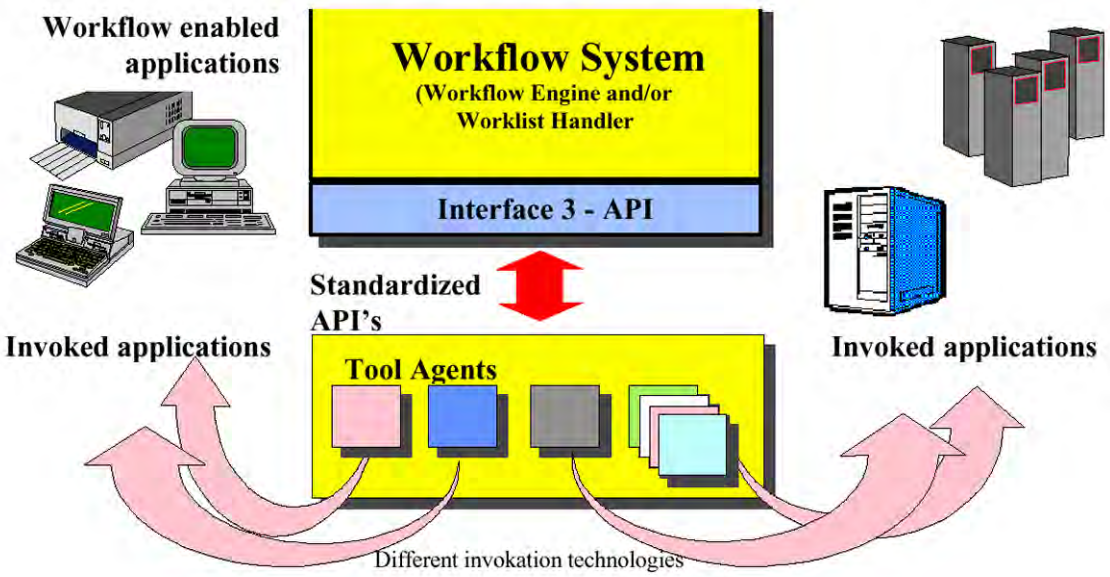



Abbildung 5.5: Die Funktionsweise der Schnittstelle 2

### 5.1.3 Interoperabilität zwischen WFMS (Interface 4)

Die Schnittstelle 4 der WFMC regelt die Interoperabilität zwischen WFMS verschiedener Hersteller. Die zu dieser Schnittstelle konformen WFMS sind in der Lage, Prozessdefinitionen untereinander auszutauschen und zu interpretieren [WFMC98].
Unabhängig davon unterscheidet die WFMC drei verschiedene Arten von WFMS-Interoperabilität:

- `Verkettete Prozesse` (*Chained processes*): Eine von einem WFMS `A` kontrollierte Prozessinstanz startet während ihrer Ausführung die Instanz eines Subprozesses auf einem WFMS `B`. Nach dem Start des Subprozesses verlaufen beide Prozessinstanzen unabhängig voneinander.

- `Geschachtelte Sub-Prozesse` (*Nested sub-processes*): Eine Prozessinstanz startet die Instanz eines Subprozesses auf einem anderen WFMS und wartet mit der weiteren Ausführung, bis der Subprozess seine Beendigung meldet.

- `Synchronisation` (*Parallel synchronized*): Zwei verschiedene WFMS führen je eine Prozessinstanz aus. Zwischen diesen Prozessinstanzen wurde zur Modellierzeit ein Treffpunkt spezifiziert. Erreicht eine Prozessinstanz diesen Punkt während der Ausführung, wartet sie bis die zweite Prozessinstanz diesen Punkt ebenfalls erreicht. An diesem Punkt können Daten ausgetauscht werden. Anschließend laufen die Prozessinstanzen wieder unabhängig voneinander weiter.

Die Variante `Synchonisation` wurde von der WFMC als mögliche Art der Interoperabilität aufgenommen. Sie ist aber nicht Teil der aktuellen Standardisierungsbemühungen.

Die erwähnten Interoperabilitätstrategien machen keine Aussagen über die Art und Weise der Kommunikation zwischen zwei WFMS. Daher wurden von der WFMC zwei Dokumente veröffentlicht, welche Mechanismen zum Nachrichtenaustausch auf WF-XML Basis [WFMC00a] und MIME3 Basis [WFMC00] definieren. In den folgenden Abschnitten werden kurz beide Spezifikationen vorgestellt.

#### 5.1.3.1 WF-XML

Der XML-basierte Nachrichtenaustausch zwischen WFMS ermöglicht speziell die Interoperabilitätsarten `verkettete und geschachtelte Prozesse`. Er soll sowohl synchron als auch asynchron erfolgen können. Die Spezifikation der WF-XML ist vom Transportmechanismus, von der Plattform und von der Hardware unabhängig. Um jedoch der Verbreitung von HTTP gerecht zu werden, wird der WF-XML Nachrichtenaustauch mit diesem Protokoll spezifiziert [WFMC00b].
In der aktuellen Version der WF-XML Interoperabilität-Spezifikation [WFMC00a] werden allerdings lediglich die folgenden Operationen definiert:

- Prozess instanziieren (*CreateProcessInstance*) (siehe Beispiel 5.2)
- Daten einer Instanz erfragen (*GetProcessInstanceData*)
- Status einer Instanz ändern (*ChangeProcessInstanceState*)
- Statusänderungen an den Initiator (Starter) einer Instanz zurückgeben (*ProcessInstanceStateChanged*)

Die Kommunikation zwischen zwei WFMS-Komponenten ist in Abbildung 5.6 dargestellt. Bei mindestens einer der beiden WFMS-Komponenten handelt es sich um einen `Workflow Enactment Service`. Dieser stellt Prozessmodelle (*Process Definition*) bereit und verwaltet die in Ausführung befindliche Prozessinstanz (*Process Instance*). Die zweite WFMS-Komponente agiert als Initiator (Starter) einer Prozessinstanz (*Observer*) (z.B. ein WFMS).

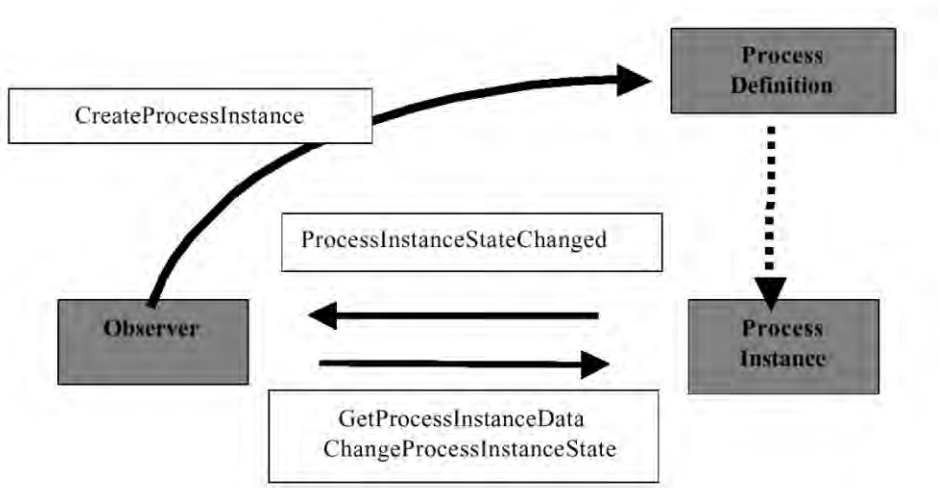



Abbildung 5.6: Kommunikation mittels WF-XML [WFMC00a]

**Beispiel 5.1**: (*Struktur von WF-XML-Nachrichten*)
Es existieren nur zwei Arten von WF-XML Nachrichten: Anfragen (*Request*) und Antworten (*Response*). Für beide Arten wird deren Struktur beispielhaft dargestellt (siehe Tabelle 5.2).

**WF-XML-Anfrage**

```
<?xml version=1.0?>
<WfMessage Version=1.0>
  <WfTransport/>
  <WfMessageHeader>
      <Request
         ResponseRequired="Yes"/>
      <Key>http://www.
XYZcompany.com/wfprocess/foo</Key>
  </WfMessageHeader>
  <WfMessageBody>
  ...
  </WfMessageBody>
</WfMessage>
```

**WF-XML-Antwort**

```
<?xml version=1.0?>
<WfMessage Version=1.0>
  <WfTransport/>
  <WfMessageHeader>
      <Response/>
      <Key>http://www.
XYZcompany.com/wfprocess/foo</Key>
  </WfMessageHeader>
  <WfMessageBody>
  ...
  </WfMessageBody>
</WfMessage>
```

Tabelle 5.2: Die Struktur von WF-XML-Nachrichten

**Beispiel 5.2**: (*Prozessinstanziierung mittels WF-XML-Nachrichten*)
Im folgenden wird beispielhaft die Instanziierung eines Prozessmodells gezeigt. Der Parameter `<Key>` enthält bei einer Anfrage den Empfänger und bei einer Antwort den Absender einer Nachricht [Fis01].

```
<?xml version="1.0"?> <WfMessage Version=1.0>
   <WfTransport/>
   <WfMessageHeader>
      <Request ResponseRequired=Yes>
      </Request>
      <Key>http://www.XYZcompany.com/wfprocess/foo
      </Key>
   </WfMessageHeader>
   <WfMessageBody>
      <CreateProcessInstance.Request StartImmediately=true>
         <ObserverKey>http://www.ABCcompany.com/wfprocessor
         </ObserverKey>
         <ContextData>
            <Computer>
               <Type>notebook</Type>
               <Series>Inspiron<Series>
               <Number>7500</Number>
               <Optioin>DVD</Option>
            </Computer>
         </ContextData>
      </CreateProcessInstance.Request>
   </WfMessageBody>
</WfMessage>
```

#### 5.1.3.2 MIME

In diesem Standard-Dokument [WFMC00] wird eine Möglichkeit zur Kommunikation zwischen WFMS beschrieben. Als Transportmechanismus werden Emails für die zwischen den WFMS ausgetauschten Nachrichten verwendet. Dabei werden diese Nachrichten MIME-codiert (*Multipurpose Internet Mail Extension*).
Folgende Operationen werden in der aktuellen Version der MIME Interoperabilitätsspezifikation [WFMC00] definiert:

- Prozess instanziieren (*CreateProcessInstance*)
- Status einer Instanz ändern (*ChangeProcessInstanceState*)
- Daten einer Instanz erfragen (*GetProcessInstanceAttributes*)
- Status einer Instanz abfragen (*GetProcessInstanceState*)
- Änderungen an den Daten eine Instanz an Initiator übermitteln (*ProcessInstanceAttributeChanged*)
- Status-Änderungen einer Instanz an Initiator übermitteln (*ProcessInstanceStateChanged*)
- Festlegen der Daten einer Instanz (*SetProcessInstanceAttributes*)
- Verbindungsaufbau und Verbindungsabbau zu fremdem WFMS (*StartConversation* und *StopConversation*)

Eine Nachricht kann als Anfrage (*request-message*), als Antwort (*response-message*) oder als Fehler (*error-message*) gekennzeichnet sein. Wird ein Fehler gemeldet, sollte diese Nachricht keine Operationen beinhalten.

**Beispiel 5.3**: (*Eine Nachricht auf MIME-Basis*)
Im folgenden wird beispielhaft die Instanziierung eines Prozessmodells gezeigt.

```
From: Workflow Engine X <x@wfmc1.org> To: Workflow Engine
Y<y@wfmc2.org> Subject: wfmc-if4-response[1]K1234+ABC&&
MIME-Version: 1.0 Content-type: text/plain

wfmc-if4-request[0]K1234+&1998-04-25T04:25:39Z&&
StartConversation?ContractID=Nice Group&Version=1.1&
SourceNodeID=xyz@wfmc.org&RootPID=24&
ProductID=MagicWorkflow/5.0&OpID=1&
SourcePID=24&SourceConversationID=K1234&&
CreateProcessInstance?ProcessDefinitionID=Open Account& Profile=chain&OpID=2&&
SetProcessInstanceAttributes?OpID=3&Number=2&
Name=copies&Type=WMTINT8&Value=3&
NAME=description&TYPE=WMTTEXT&VALUE=Two lines of text, this is
the first line\%OD\%OAThis is the second line.&&
ChangeProcessInstanceState?OpID=4&State=open.running&&
StopConversation?OpID=5&& end(38341)
```

### 5.1.4 Administrations- und Monitoringwerkzeuge (Interface 5)

Die Schnittstelle 5 des WFMC-Referenzmodells standardisiert den Zugriff auf Ausführungs-/Audit-Daten von Workflows. Damit können Administrations- und Monitoring-Tools WFMS-übergreifend eingesetzt werden. Zu diesem Zweck wird spezifiziert, welche Informationen während der Workflowausführung erfasst und aufgezeichnet werden [WFMC98a]:

- *Process Instance Audit Information* (z.B. Erstellen einer Prozessinstanz)
- *Activity Instance Audit Information* (z.B. Abbruch einer Prozessaktivität)
- *Workitem Audit Information* (z.B. Start eines Workitems)
- *Remote Operation Information* (z.B. Veränderungen am Status einer Prozessinstanz)
- *Process Definition Audit Information* (z.B. Veränderungen am Status des Prozessmodells)

- *Discretionary Audit Information* (Erweiterung geplant, z.B. speziell vereinbarte Daten zwischen zwei oder mehreren WFMS Hersteller)
- *Private Audit Information* (Erweiterung geplant, z.B. spezielle Daten, welche nicht jedes WFMS bereitstellt)

Diese Daten werden auch CWAD (*Common Workflow Audit Data*) genannt. Mit den CWAD-Informationen kann ein Unternehmen untersuchen bzw. bestimmen, was während der Ausführung eines Geschäftsprozessmodells durch ein WFMS geschah. Typische Fragestellungen für die Verwendung der CWAD sind:

- Wie lange dauerte die Ausführung von Prozessinstanz XY?
- Welche Aktivitäten wurden während der Ausführung einer Prozessinstanz aktiviert?

Die Art der Speicherung dieser Daten wird nicht festgelegt. Ein Zugriff bzw. eine Auswertung ist daher auf mehrere Arten (z.B. mittels SQL, C oder C++) denkbar.

## 5.2 Weitere Standards

Neben den Standardisierungsbemühungen der WFMC existieren andere Gruppen, die ebenfalls Vorschläge für Workflow-Standards entwickeln. Diese Gruppen beschränken ihre Standardisierungsbemühungen auf Auschnitte des gesamten Workflow-Management-Systems. So haben die `OMG` (*Object Management Group*) und die `SWAP Working Group` (*Standard Workflow Access Protocoll*) [SWAP01] Standards entwickelt, die die `Schnittstelle 4` der WFMC erweitern.

### 5.2.1 OMG Workflow Management Facility Standard (JointFlow)

Die OMG hat sich zum Ziel gesetzt, die Verbreitung der Objekttechnologie zu fördern und hat hierzu mit CORBA eine Referenzarchitektur [OMG01] vorgestellt. CORBA stellt eine plattform- und sprachunabhängige Infrastruktur für verteilte Objekte bereit (siehe Abschnitt 3.1.4.2). Die OMG hat sich entschlossen, ihre Standardisierungsbemühungen auf weitere Gebiete auszuweiten. Ein Vorschlag ist die `Workflow Management Facility` [OMG98], für die Anfang 1997 eine Arbeitsgruppe eingerichtet wurde. Ihr Ziel ist die Definition einer Schnittstelle (incl. Semantik) für die Ausführung und Manipulation von verteilten Workflow-Objekten und deren Metadaten. Dieser Vorschlag basiert auf den Standards der WFMC und stellt eine Basis für die Integration der Workflow Technologie in das OMG Architekturmodell (OMA = *Object Management Architecture*) dar. Weitere Informationen zu Anforderungen, zur Architektur und zu möglichen Implementierungen (speziell in OMA-konformen Architekturen) verteilter WFMS werden in [JaBöSc97] bereitgestellt. Abbildung 5.7 stellt das von der OMG vorgeschlagene Objektmodell (*Joint Workflow Management Facility Model*) dar.

### 5.2.2 SWAP (Simple Workflow Access Protocol)

Die SWAP-Spezifikation definiert ein internetbasiertes WF-Protokoll um Prozesse zu instanziieren, zu kontrollieren und zu überwachen [Fis01]. Diese Spezifikation kann als Implementierung der Schnittstelle 4 der WFMC auf HTTP-Basis angesehen werden.
Das Objektmodell der SWAP definiert die vier Typen Observers, ProcessInstances, ProcessDefinitions und Activities. Diese koorespondieren mit den Schnittstellen WfRequestor, WfProcess, WfProcessMgr und WfActivity des OMG-Standards (vergleiche Abbildung 5.7).
Nachrichten werden mit XML codiert und mittels HTTP-Anfragen/-Antworten zwischen den beteiligten WFMS verschickt.

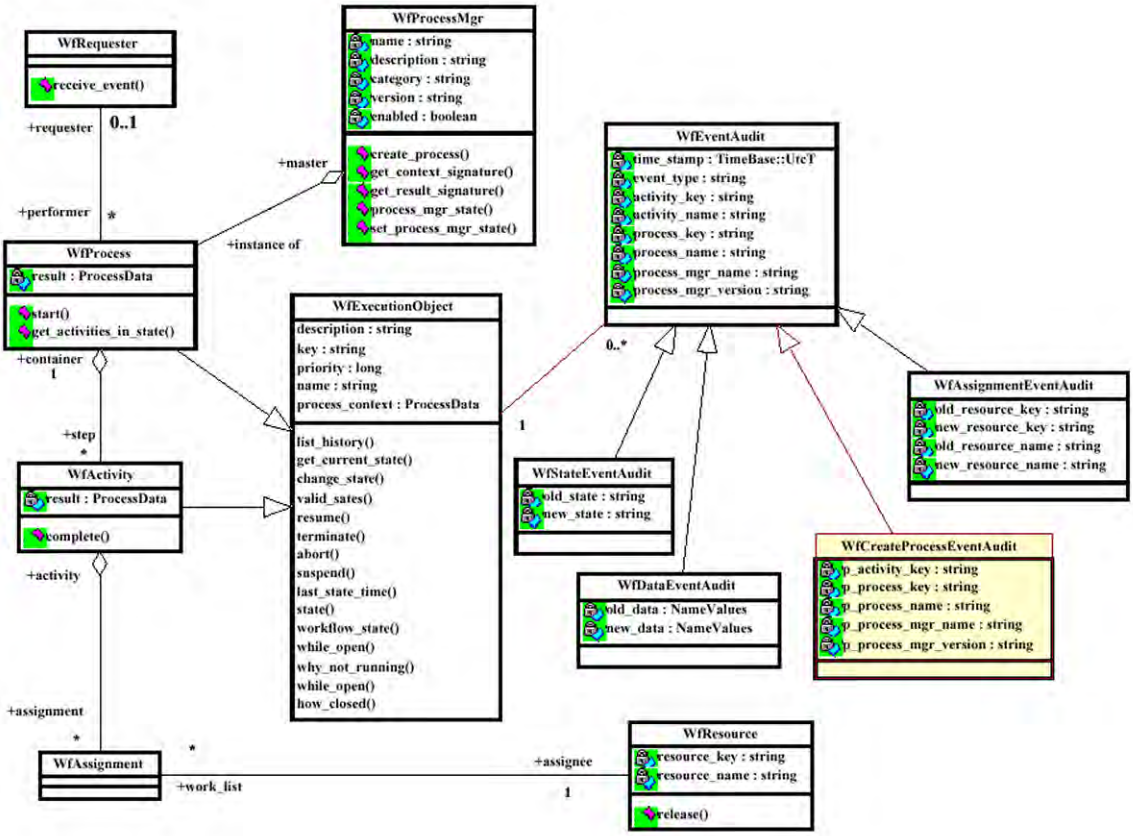



Abbildung 5.7: WF-Objektmodell der OMG (aus [OMG00])

## 5.3 MQSeries WF

Das Vorgängerprodukt von MQSeries WF (`Flowmark 2.3`) erfüllte die Schnittstellen 2, 3 und 4 der WFMC. Daher überrascht die Tatsache, daß MQSeries WF zu keiner der fünf Schnittstellen[1] der WFMC konform ist [WFMC01b]. In Tabelle 5.3 ist der aktuelle Stand[2] der Planungs- und Entwicklungsstufen der Schnittstellen dargestellt.

| | |
|---|---|
| Schnittstelle 1 | nicht implementiert (*Trial developments done*) |
| Schnittstelle 2 | geplant |
| Schnittstelle 4 | geplant |
| Schnittstelle 5 | geplant |
| OMG (Jointflow) | geplant |

Tabelle 5.3: Konformität von MQSeries WF zu den Schnittstellen der WFMC

In den folgenden Abschnitten wird gezeigt, in wie weit die Funktionen der fünf Schnittstellen der WFMC in MQSeries WF implementiert sind. Einen Gesamtüberblick der vorhandenen Tools ist in Abbildung 5.8 dargestellt. Der in der Abbildung verwendete Begriff *API Development Kit* bezieht sich auf die in Abbildung 5.10 dargestellte Struktur.

[1] Für die Standards der Schnittstelle 3 und WF-XML wurden bisher keine Aussagen seitens der WFMC über den Entwicklungs- oder Planungsstand getroffen.

[2] 27.05.2001

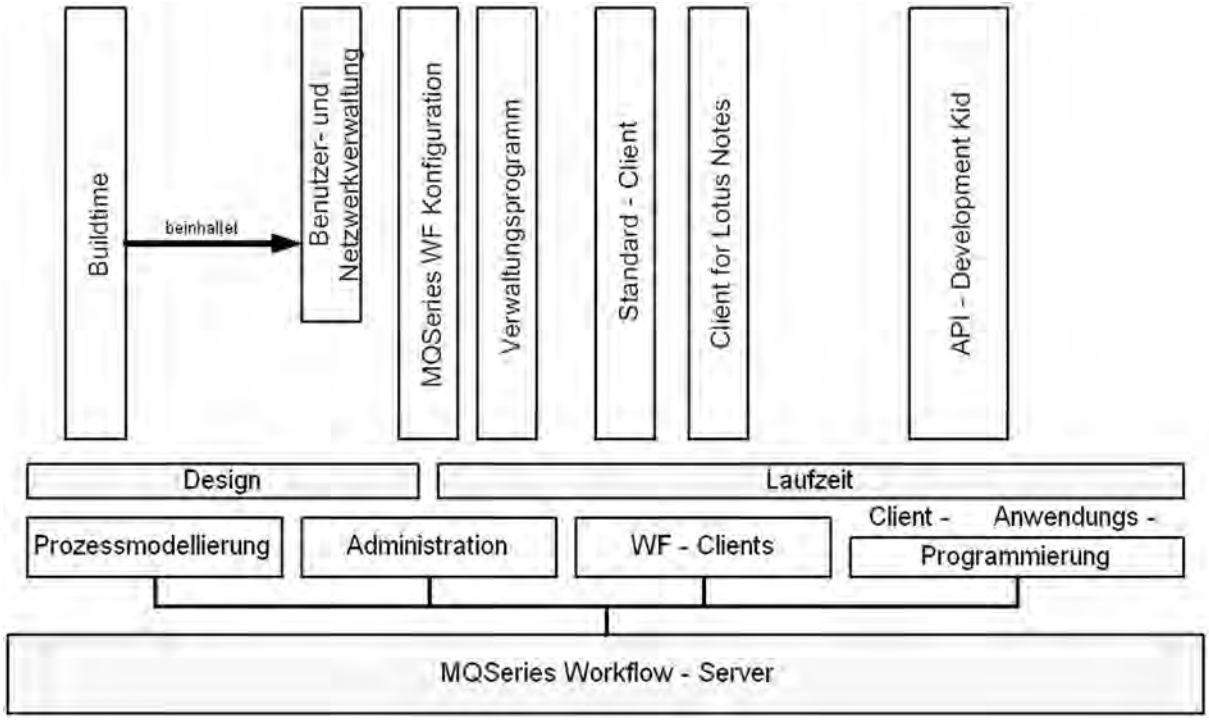



Abbildung 5.8: Komponenten und Tools von MQSeries WF

### 5.3.1 Import/Export von Prozessmodellen

MQSeries WF verfügt Client- und Serverseitig über eine Import-/Export-Schnittstelle. Die Buildtime-Komponente bietet dem Benutzer die Möglichkeit, einzelne, mehrere oder auch unvollständige Prozessmodelle zu importieren bzw. zu exportieren. Auf dem MQSeries WF-Server existiert dazu ein Import-/Export-Dienstprogramm (siehe Abschnitt 3.1.2.2), um Prozessmodelle in die Runtime-Datenbank zu importieren bzw. aus der Runtime-Datenbank zu exportieren.
Die hierfür verwendete WPDL wird als FDL bezeichnet (siehe Kapitel 2). Da die Struktur von FDL-Dateien bekannt ist [IBM99c], kann ein Prozessmodell theoretisch mit jedem beliebigen Text-Editor erstellt oder manipuliert werden.
Der Import/Export von FDL-Dateien wird von einigen Anwendungen fremder Hersteller unterstützt (z.B. das ARIS-Toolset von IDS Scheer [MüSt99] und [IBM01d]).

### 5.3.2 Client- und Anwendungsprogrammierung

Die Architektur eines MQSeries WF Clients ist in Abbildung 5.9 dargestellt [IBM99b]. Ein MQSeries WF Client startet Prozessmodelle und überwacht ihre Ausführung. Die Client-Komponente `Programm- ausführungsagent` ruft Anwendungsprogramme auf, die während der Prozessmodellierung für die Programmaktivitäten hinterlegt worden sind.
Die grundlegende Zugriffe auf Dienste des MQSeries WF-Server werden durch eine C-API und eine nachrichtenbasierte XML-Schnittstelle bereitgestellt. Durch die C-API kann der Zugriff auf Dienste des MQSeries WF-Servers von allen Programmiersprachen, welche C-Aufrufe unterstützen, erfolgen (siehe Abbildung 5.10).
Wie die folgenden Abschnitte zeigen, sind nicht alle Schnittstellen auf gleiche Weise zur Programmierung von Client- bzw. Anwendungsprogrammen geeignet.
Die in Abbildung 5.10 weiß unterlegten Felder bieten nur eine Unterstützen der Funktionalität des Vorgängerproduktes Flowmark (Version 2) [IBM99e]. Bei REXX handelt es sich um eine Skriptsprache. Eine beispielhafte Clientimplementierung kann in [Dad99] eingesehen werden. Im weiteren werden diese APIs nicht besprochen.

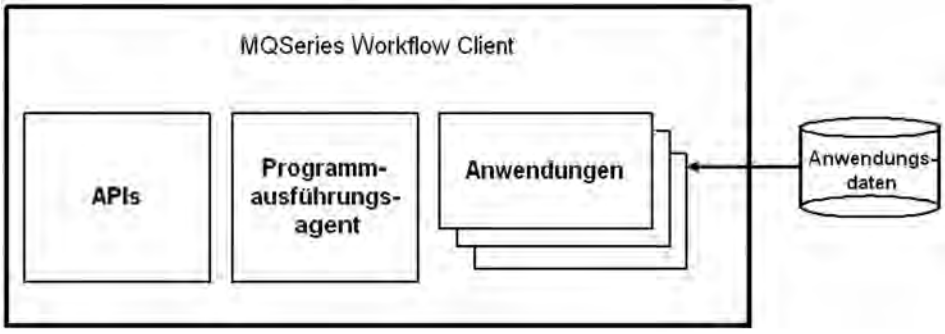



Abbildung 5.9: Der Aufbau der MQSeries WF Client-Komponente

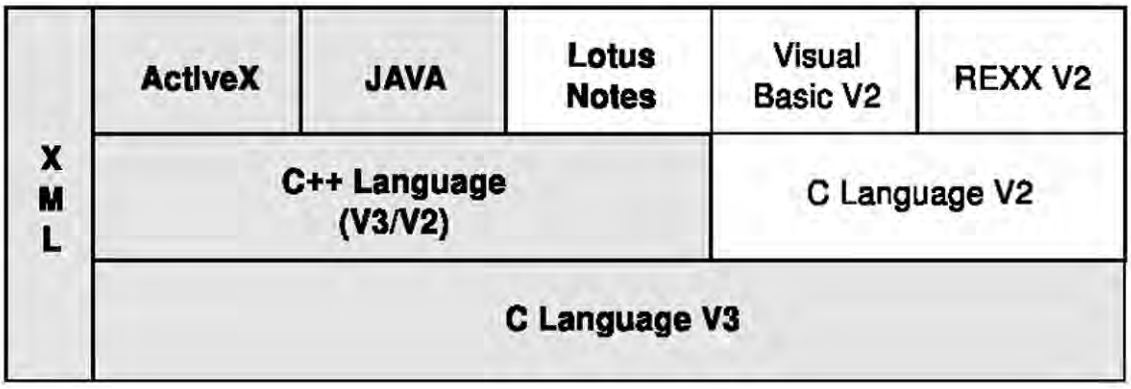



Abbildung 5.10: Die API-Struktur von MQSeries WF (aus [IBM99e])

#### 5.3.2.1 Clientprogrammierung

Für Windows-Umgebungen (NT und Windows 95/98) beinhaltet MQSeries WF einen Standard-Client (siehe Abschnitt 3.1.3). In diesem Client sind die Standardfunktionen (z.B. Arbeitslistenverwaltung) eines WF-Clients, wie in Abschnitt 5.1.2 beschrieben, enthalten.
Für Benutzer von Lotus Notes ist der `MQSeries WF Client for Lotus Notes` auf Windows-Plattformen und OS/2 erhältlich. Dieser Client besitzt keine eigene Oberfläche. Er integriert sich vollständig in Lotus Notes und bietet dem Benutzer die Funktionalität des Standard-Clients.

##### 5.3.2.1.1 Client-API

Neben der Verwendung von Standard-Clients, ist es durch eine Vielzahl von APIs (siehe Abbildung 5.10) möglich, angepasste Clients zu programmieren. Die MQSeries WF APIs stellen folgende Funktionen bereit [IBM99e]:

- Ausführen von Prozessmodellen (dazu gehört z.B. die Verwaltung von Prozessinstanzen und die Manipulation von Arbeitslisten und Workitems)
- Monitoringfunktionen (z.B. Überwachung der Ausführung von Prozessinstanzen)
- Administrationsfunktionen
- Empfang von Informationen des MQSeries WF-Server.

Die Programm-Struktur eines Clients ist im wesentlichen für alle APIs gleich. Im Beispiel 5.4 wird die Struktur eines Clients unter Verwendung der `C-/C++-API` dargestellt [IBM99e].

Die Implementierung eines Clients auf Basis der nachrichtenbasierten Schnittstelle (XML) ist nicht möglich.
Jede Anwendung die das XML-Nachrichtenformat[3] von MQSeries WF unterstützt, kann Anfragen an die Eingangs-Warteschlange des MQSeries-Server schicken und eine bestimmte Aktion anfordern. Allerdings werden nur zwei Aktionen durch diese nachrichtenbasierte Schnittstelle unterstützt:

- Erstellen und Starten einer Prozessinstanz (*CreateAndStartInstance()*)
- Ausführen einer Prozessinstanz (*ExecuteProcessInstance()*)

Zwingende Funktionalitäten, wie z.B. eine Arbeitslistenverwaltung, sind daher nicht möglich.

**Beispiel 5.4**: *(Die Struktur eines Clients [C/C++-API])*
Die Programm-Struktur lässt sich in drei Teile untergliedern:

- Setup-Phase (Deklaration der Programmvariablen und -objekte, Einbinden der benötigten MQSeries WF Header-Dateien[4], Initalisierung der MQSeries WF API, Anlegen eines Service Objektes (als Repräsentation des Servers) und Anmelden am Service Objekt)
- Aufruf von Aktionen entsprechend den Rechten des angemeldenten Benutzers.
- Verbindung trennen (Beenden der Session mit dem Service Objekt und Trennung der Verbindung zur MQSeries WF API)

```
// SETUP
#include <MQ Workflow API-prerequisites (C++)>
#include <MQ Workflow API>
int main()
{
     Declare Objects
     ...
     Connect
     Allocate Service Object
       Logon()
// Aktionen ausführen

       MQ Workflow API calls

// Verbindung trennen
       Logoff()
     Deallocate Service Object
     Disconnect
     return 0;
}
```

[3]MQSeries WF verwendet den Standard XML 1.0 (siehe http://www.w3.org/TR/REC-xml).
[4]Wird die C++-API verwendet müssen vor den MQSeries WF Header-Dateien noch die Header-Dateien für `bool, string` und `vector` eingebunden werden.

#### 5.3.2.1.2 WebClient

Der MQSeries WF WebClient ermöglicht den Zugriff auf MQSeries WF über das Internet. Dadurch sind die Funktionen eines WF-Clients über einen Internet-Browser ausführbar.
Eine Besonderheit des WebClient ist, daß er auf Computern einsatzfähig ist, die über keine MQSeries WF-API verfügen. Dies bedeutet, daß auf einem solchen Computer weder MQSeries noch ein PEA verfügbar sind.
Der WebClient verwendet die Java Servlet-Technologie (siehe Abb. 5.11). Daher muß ein Web-Server verwendet werden, der Java Servlets ausführen kann.

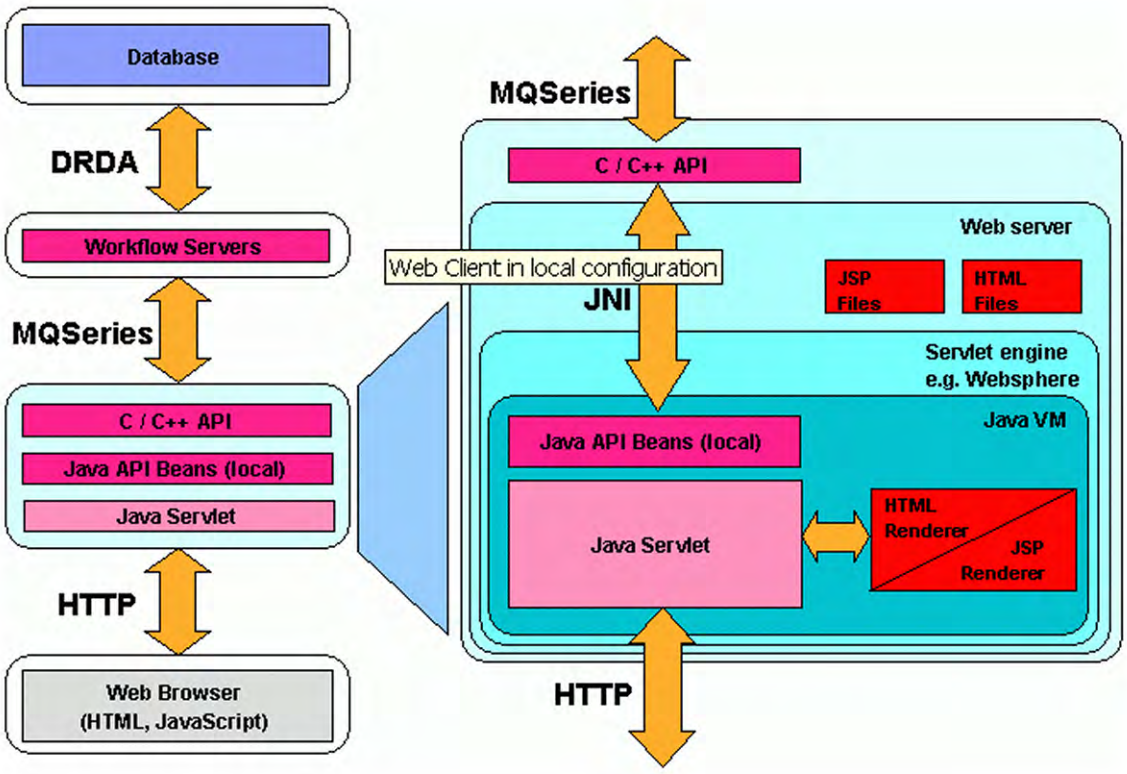



Abbildung 5.11: Die Architektur des MQSeries WF WebClient (aus [IBM01])

Die Ausführung eines Workitems kann dem Benutzer auf zwei Arten dargestellt werden:

- `Standard`: Ein Standard-Viewer generiert auf dem Web-Server aus den vom WFMS empfangenen Daten eine HTML-Seite. Dabei wird für jedes Element eines Input-Containers ein Feld in der HTML-Seite erzeugt.

- `Angepasst`: Für jedes Workitem können jsp-Dateien (*Java Server Pages*) hinterlegt werden (Workitem und jsp-Datei müssen den selben Namen haben). Der WebClient prüft ob für ein Workitem eine jsp-Datei existiert. Falls dies zutrifft, wird sie anstatt des Standard-Viewers gerufen. In Beispiel 5.5 wird der Aufbau einer solchen jsp-Datei gezeigt. Durch diese Variante ist es Unternehmen möglich, auf die Darstellung (der angezeigten HTML-Seiten) Einfluss zu nehmen (z.B. Anpassung der Darstellung im WebClient an die Darstellung der Anwendungsprogramme).

**Beispiel 5.5**: (*Struktur einer jsp-Datei*)
Der Zugriff auf Workitems und Daten-Container kann in einer jsp-Datei durch folgenden Code erreicht werden (siehe [IBM01]).

```
<%@ page language="java" info="JSP activity implementation" %>
    <%@ page import="com.ibm.workflow.api.*" %>
    <%@ page import="com.ibm.workflow.servlet.client.*" %>
    <jsp:useBean id="context" scope="request"
                 type="com.ibm.workflow.servlet.client.RequestContext"/>
    <html>
    <body>
    <% ReadOnlyContainer inData = context.getContainer();
       WorkItem workItem = context.getWorkItem(); %>
    :
    <%=context.openForm("checkInWorkItem", workItem.persistentOid())%>
    :
      <input type="submit" name="submitData" value="Complete work item">
      <input type="button" name="cancel" value="Cancel"
             onClick="javascript:{location=
      '<%=context.getCommand("cancelWorkItem", workItem.persistentOid())%>';}">
    </form>
    :
    </body></html>
```

#### 5.3.2.2 Anwendungsprogrammierung

Die Erstellung von Anwendungsprogrammen ist mit allen von MQSeries WF bereitgestellten APIs möglich. Mit Ausnahme der XML-Schnittstelle erfolgt der Aufruf eines Anwendungsprogramms durch eine Anforderung an den Programmausführungsagent oder den Ausführungsserver (PES = *Program Execution Server*). Diese führen dann das zur Modellierzeit einer Aktivität hinterlegte Anwendungsprogramm aus. Das Verfahren der Programmausführungsagenten orientiert sich am Vorschlag der WFMC (siehe Abschnitt 5.1.2.2).
Wird die nachrichtenbasierte Schnittstelle verwendet, ist es möglich, daß MQSeries WF die Anfrage zur Ausführung eines Anwendungsprogramms (im XML-Format) an eine vom Benutzer spezifizierte MQSeries Warteschlange schickt. Aus der Sicht von MQSeries WF verhält sich das Anwendungsprogramm wie der Ausführungsserver. Es erhält alle benötigten Daten und muß nach deren Bearbeitung MQSeries WF eine Antwort bereitstellen. Diese Art von Anwendungsprogrammen werden UPES[5] (*user-defined program execution server*) genannt. Ein UPES kann jedes selbstgeschriebene Anwendungsprogramm sein, daß das XML-Format von MQSeries WF richtig verarbeiten kann.

**Beispiel 5.6**: *(Die Struktur eines Anwendungsprogramms [C/C++-API])*
Die Programm-Struktur lässt sich in drei Teile untergliedern:

- Setup-Phase (Deklaration der Programmvariablen und -objekte, Einbinden der benötigten

[5]UPES werden in der Buildtime-Komponente von MQSeries WF unter Netzwerk registriert und mit einer MQSeries-Warteschlange verbunden. Anwendungen die den UPES verwenden sollen muß dieser explizit als PES zugewiesen werden.

MQSeries WF Header-Dateien[6], Initalisierung der MQSeries WF API, Initialisierung des Input- und Output-Container)

- Aufruf von Aktionen (lesender und schreibender Zugriff auf die Daten-Container).
- Verbindung trennen (Übergabe des Output-Container an den Programmausführungsagenten, Freigabe der gehaltenen Ressourcen)

```
// SETUP
#include <MQ Workflow API-prerequisites (C++)>
#include <MQ Workflow API>
int main()
{
     Declare Objects
     ...
     Connect
     InContainer()
       OutContainer()        falls benötigt
// Aktionen ausführen

       read values
       set values       falls benötigt

// Verbindung trennen
       SetOutContainer()       falls benötigt
     Deallocate Object
     Disconnect
     return rc;
}
```

In einem Anwendungsprogramm muß nicht zwingend ein Output-Container berücksichtigt werden. Dies ist für Anwendungsprogramme gedacht, die dem Benutzer Informationen darstellen und keine Änderungen an den Daten zulassen (sogenannte *Support Tools*).

Ein Anwendungsprogramm kann zusätzlich Client-Funktionalitäten implementieren, wie z.B. Monitoringfunktionen. Erreicht wird dies durch die spezielle Funktion *Passthrough()* [IBM99e], welche den Starter der zum Anwendungsprogramm gehörigen Workitem am Server anmeldet. Diese Funktion ersetzt das *Login* und ermöglicht die Verwendung von Client-Funktionalitäten.

### 5.3.3 Interoperabilität zwischen WFMS

MQSeries WF verfügt nicht über eine Implementierung der Schnittstelle 4 gemäß dem Standard-Vorschlag der WFMC. Dies geht zumindest aus den aktuellsten Veröffentlichungen der WFMC hervor (siehe [WFMC01d]).
In der Internetpräsenz von IBM steht ein Dokument zum Download [IBM01c], welches sich mit dem Zusammenspiel von `Domino Workflow`[7] und `MQSeries Workflow` beschäftigt. Abbildung 5.12 ist diesem Dokument entnommen. Es suggeriert dem Betrachter, daß MQSeries WF die Schnittstelle

[6]Wird die C++-API verwendet müssen vor den MQSeries WF Header-Dateien noch die Header-Dateien für `bool, string` und `vector` eingebunden werden.

[7]Domino Workflow ist ein Notes-basiertes WFMS der IBM-Tochter Lotus [Lot01]

4 der WFMC auf WF-XML-Basis erfüllt. Wie erwähnt wird dieser Sachverhalt von der WFMC in frei zugänglichen Dokumenten nicht bestätigt.
Weitere Informationen in Bezug auf die Funktionalität der von IBM entwickelten Schnittstelle 4 wurden in der verfügbaren Literatur nicht gefunden.

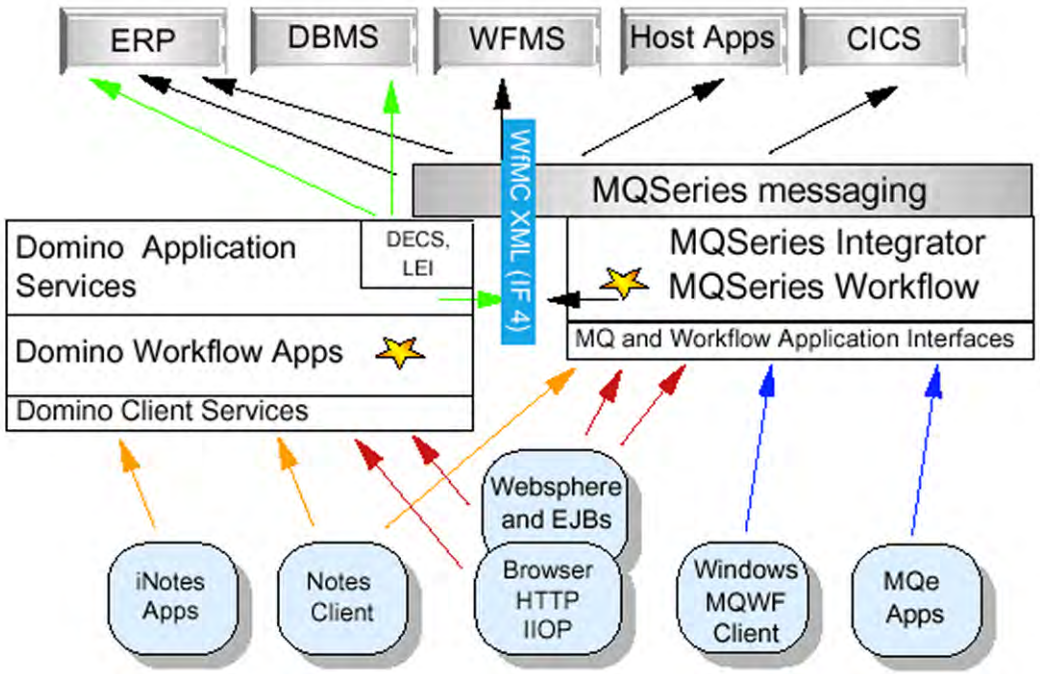



Abbildung 5.12: Das Zusammenspiel von MQSeris WF und Domino WF (aus [IBM01c])

### 5.3.4 Administrations- und Monitoringwerkzeuge

MQSeries WF verfügt über mehrere Werkzeuge zur Administration und zum Monitoring des WFMS. Tabelle 5.4 gibt hierüber einen Überblick:

| **Anwendungen** | **Aufruf als ...** |
|---|---|
| `Konfigurationsprogramm` | eigenständiges Tool |
| `Verwaltungsprogramm` (*Administration Utility*) | Kommandozeilenprogramm |
| `Instanz-Überwachung` | im Standard-Client |
| `Datenbankbereinigung` (*cleanup audit trail utility*) | Kommandozeilenprogramm |
| `Buildtime-Komponente` | eigenständiges Tool |

Tabelle 5.4: Administrations- und Überwachungswerkzeuge von MQSeries WF

#### 5.3.4.1 Konfigurationsprogramm

Das Konfigurationsprogramm *fmczacfg.exe* von MQSeries WF wird erstmals direkt nach der Installation gestartet. Es bietet eine graphische Benutzerschnittstelle und Konfigurationsmöglichkeiten für folgende Komponenten [IBM99a] :

- Buildtime-Datenbank
- Runtime-Datenbank
- Server

- Runtime-Datenbank-Dienstprogramme
- Cient
- Buildtime
- Java CORBA Agent

Die Konfiguration von MQSeries WF mithilfe dieses Tools ist nach der Installation zwingend nötig. Das Konfigurationsprogramm kann zu jeder beliebigen späteren Zeit erneut aufgerufen werden, um eine Konfiguration zu ändern oder um eine neue Konfiguration hinzuzufügen (siehe [IBM99a]).

#### 5.3.4.2 Verwaltungsprogramm

Durch das Verwaltungsprogramm *fmcautil.exe* wird das komplette MQSeries WF-System verwaltet [IBM99]. Es wird auf Kommandozeilenebene gestartet und bietet dem Benutzer, nach erfolgreichem Login (z.B. `fmcautil.exe /u=ADMIN /p=password`), ein Auswahlmenü an (siehe Abbildung 5.13).

```
FMC15010I Main Menu:
     s ... System Commands Menu
     m ... Select Server Menu
     e ... Errorlog Commands Menu
     l ... Systemlog Commands Menu
     u ... User Commands Menu
     x ... Exit Main Menu
```

Abbildung 5.13: Auswahlmenü des Verwaltungsprogramms (aus [IBM99])

#### 5.3.4.3 Instanz-Überwachung

Im Standard-Client von MQSeries WF kann sich ein Benutzer über den Ausführungszustand einer Instanz informieren. Durch Auswahl von `Überwachen` im Menü einer Prozessinstanz bzw. von `Prozessexemplar überwachen` im Menü eines Workitems wird eine graphische Darstellung der zugehörigen Prozessinstanz aufgerufen. Abbildung 5.14 zeigt beispielhaft die Darstellung einer in Ausfühung befindlichen Prozessinstanz. Aus der graphischen Darstellung ist erkennbar, ob eine Aktivität bereits beendet, aktiv oder noch nicht aktiviert ist. Weitere Informationen (z.B. Ausführungszeit einer Aktivität) können der Grafik nicht entnommen werden.

#### 5.3.4.4 Datenbankbereinigung

Durch das Kommandozeilenprogramm `fmcsclad.exe` können Einträge des in der Datenbank gespeicherten Audit Trails gelöscht werden. Gelöscht werden können folgende Daten:

- alle Einträge vor einem bestimmten Datum oder
- alle Einträge die zu einem Prozessmodell gehören, das vor einem bestimmten Datum beendet worden ist.

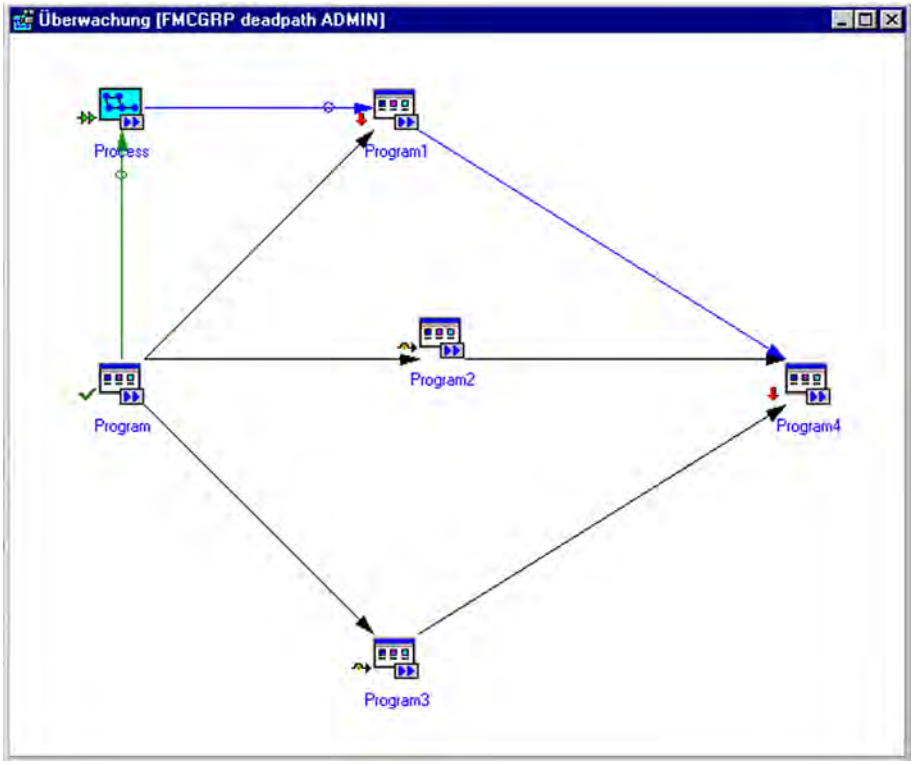



Abbildung 5.14: Instanz-Überwachung im Standard-Client

#### 5.3.4.5 Bestandteile der Buildtime

Viele Einstellungen werden in der Buildtime festgelegt. Durch Importieren in die Runtime werden diese Einstellungen für die Runtime-Komponenten gültig.
Folgende Einstellungen können vorgenommen werden:

- Benutzerverwaltung
- Netzwerkverwaltung
- Registrierung der (U)PES
- Aktivierung des Audit Trail

## 5.4 Staffware

Staffware bietet zahlreiche Schnittstellen an. Einige davon sind konform zu den Standards der WFMC. Staffware ist selbst Mitglied der WFMC und hat an der Entwicklung der Standards mitgewirkt. In Tabelle 5.5 ist der aktuelle Stand[8] der Konformität zu den Schnittstellen der WFMC aufgeführt. Informationen zu jeweils aktuellen Entwicklungen können unter [WFMC01d] abgerufen werden.

| | |
|---|---|
| Schnittstelle 1 | nicht implementiert (*Trial developments done*) |
| Schnittstelle 2/3 | nicht implementiert (*Prototype demonstrated*) |
| Schnittstelle 4<br>WF-XML 1.0 | implementiert (*Optional Module*)<br>implementiert (*Implemented*) |
| Schnittstelle 5 | nicht implementiert (*Trial Developments*) |

Tabelle 5.5: Konformität von Staffware zu den Schnittstellen der WFMC

[8] 27.05.2001

Die folgenden Abschnitte geben einen Überblick über existierende Staffware-Tools, nach funktionaler Zuordnung zu den WFMC-Schnittstellen. Abbildung 5.15 stellt einen Gesamtüberblick über den Einsatz möglicher Tools dar.

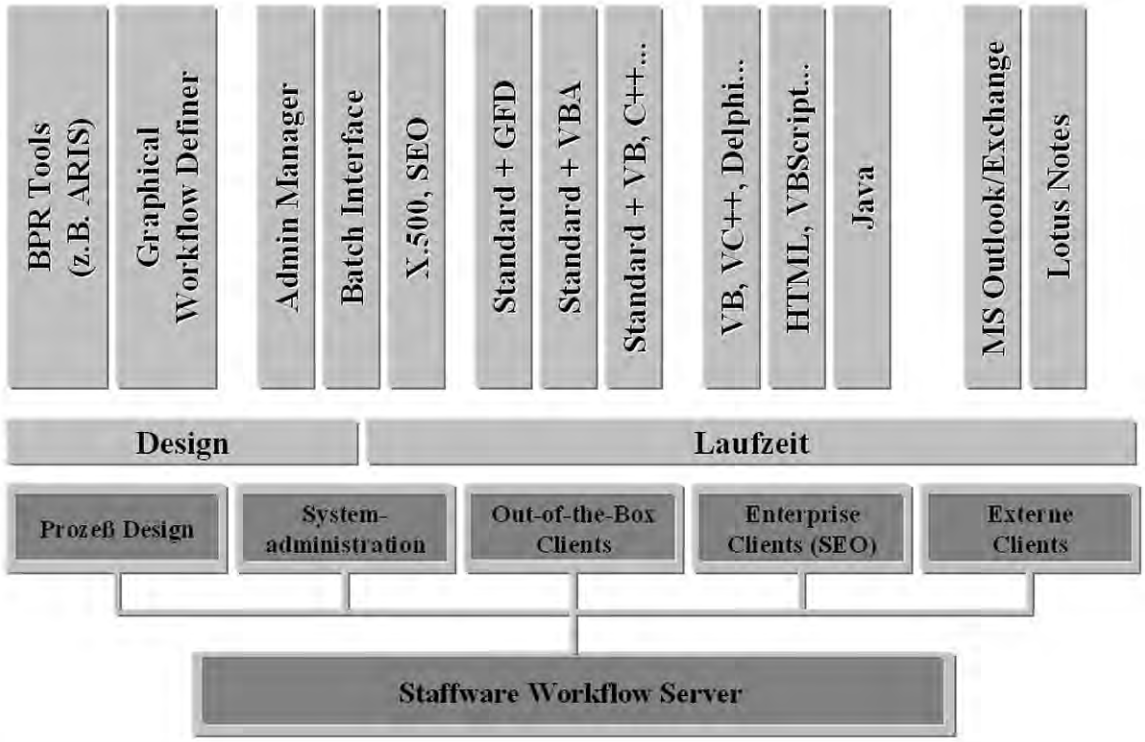



Abbildung 5.15: Komponenten und Tools von Staffware (Schnittstellen 1,2,3 und 5) [SW01b]

### 5.4.1 Import/Export von Prozessmodellen

Zur Prozessmodellierung bietet Staffware den `GWD` (*Graphical Workflow Definer*) an. Mit ihm können Prozessmodelle und die zugehörige Bearbeiter- und Anwendungszuordnungen erstellt werden (siehe Abschnitt 2.2).
Für Benutzer ist es nicht möglich, mit dem GWD Prozessmodelle zu importieren bzw. exportieren. Erstellte bzw. geänderte Prozessmodelle werden immer direkt in die Runtime-Datenbank gespeichert. Die von Staffware implementierte WPDL heißt XFR. Der Import und Export von XFR-Dateien ist nur über das Admin-Tool *SWUTIL* (siehe Abschnitt 5.4.4) möglich.

Im weiteren existieren unidirektionale Schnittstellen seitens einiger Hersteller von Geschäftsprozessmodellierungs- und Simulationstools: `Corporate Modeler` (*Hersteller: CASEwise*), `Structware` (*Hersteller: Ivyteam*) und `ARIS` (*Hersteller: IDS Scheer*) [MüSt99]. Damit ist es möglich, Modelle die mit diesen Tools erstellt wurden, in Staffware-Modelle zu überführen.
Eine solche Überführung generiert in der Regel keine fertigen Prozessmodelle. Typischerweise unterscheiden sich BPR- (*Business Process Reengineering-*) und Workflow-Projekte derart, daß überführte Modelle vor einer Weiterverwendung überarbeitet werden müssen. Dennoch kann dieses Verfahren zu einer erheblichen Zeitersparnis führen.
Das BPR-Tool `ARIS`-Toolset unterstützt zusätzlich die Übergabe der definierten Benutzer, Rollen und Gruppen an die Staffware Benutzerverwaltung. Zur Auswertung bereits abgearbeiteter Prozessinstanzen bietet der Hersteller von `ARIS` das Tool PPM *(Process Performance Monitor)* an.
Durch die Schnittstellen von `Corporate Modeler` und `Structware` ist eine Simulation (z.B. Ressourcenauslastung oder Prozesskosten) während der Prozessmodell-Entwicklung möglich. Dies ist

vorteilhaft, da Staffware selbst über keine Simulationskomponente verfügt.

### 5.4.2 Client- und Anwendungsprogrammierung

Die folgenden zwei Abschnitte zeigen die Möglichkeiten zur Entwicklung und Anpassung existierender Client- und Anwendungsprogramme.

#### 5.4.2.1 Clientprogrammierung

Staffware unterscheidet drei verschiedene Client-Typen (siehe Abbildung 5.16):

1. `Out-of-the-Box Clients`
2. `Enterprise Clients`
3. `Externe Clients`

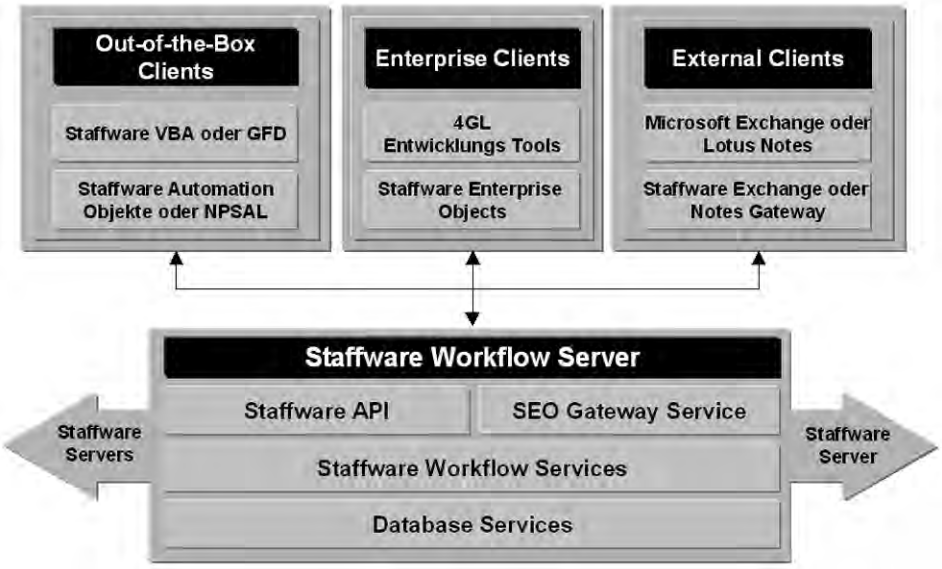



Abbildung 5.16: Staffware Client-Typen [SW99d]

##### 5.4.2.1.1 Out-of-the-Box Clients

Ein Beispiel für diesen Client-Typ ist der in Kapitel 3.2.3 besprochene Standard-Client. Er wurde von Staffware vollständig implementiert und steht dem Anwender zum sofortigen Einsatz zur Verfügung. Dazu enthält der Client den GFD und die VBA-Umgebung (als Extra-Tool, siehe Abschnitt 5.4.2.2) zur Erstellung von Anwendungsprogrammen.
Nachteilig an diesem Client-Typ ist die begrenzte Möglichkeit zur Integration in andere bzw. von anderen Applikationen. Zu diesem Zweck sind die Client-Typen in den folgenden Abschnitten besser geeignet.

##### 5.4.2.1.2 Enterprise Clients

Hierbei handelt es sich um Clienten, die von Staffware-Anwendern (Kunden oder Partner) selbst implementiert werden. Die Clients werden in Entwicklungsumgebungen generiert, welche die Staffware Funktionalität über die `Staffware Enterprise Objects` (*SEO*) einbinden [SW99d].
Dazu werden folgende Softwarepakete benötigt:

- Staffware SEO-Server (NT oder UNIX)

- Staffware SEO Client Entwicklungs Toolkit

- Entwicklungsumgebungen (z.B. Visual Basic, PowerBuilder, u.a.)

Die SEO bieten eine umfassende objektbasierte Werkzeugsammlung zur Erstellung von WF-Client-Anwendungen an. Ihnen wird durch ein Objektmodell der Zugriff auf die Funktionalität von Staffware ermöglicht.
In Tabelle 5.6 ist die aktuelle[9] Verfügbarkeit von SEO-Client Toolkits zusammengefasst. Die SEO machen die Leistungsfähigkeit eines Staffware-Servers auf nahezu jedem Client-Umfeld verfügbar, darunter z.B. Visual Basic, Oracle Forms, Java, ActiveX und HTML ([SW99e]).

| | | |
|---|---|---|
| SEO32/COM | Jeder Win32 COM Container kann durch die Benutzung der SEO32/COM auf den Staffware Server zugreifen. | verfügbar |
| SEO32/Java | Aus jeder Java Umgebung kann auf diese Objekte zugegriffen werden. | verfügbar |
| SEO/Java | SEO/Java wird auf Win32 und anderen Plattformen mit einer kompatiblen Java Virtual Machine laufen. | verfügbar |
| SEO/CORBA | SEO/CORBA wird auf CORBA Klassen beruhen. | geplant |

Tabelle 5.6: SEO-Client Toolkits [SW99d]

Für jede von Staffware unterstützte Server-Plattform (siehe Kapitel 3) ist der SEO-Server verfügbar. Die Kommunikation zwischen SEO-Client und SEO-Server erfolgt über TCP/IP Sockets.
Staffware bietet den Entwicklern von Enterprise Clients einen beispielhaft implementierten Enterprise Client mit allen Funktionalitäten und zugehörigem Quellcode. Dadurch werden dem Entwickler die Einarbeitung in SEO erleichtert und evtl. das Programmieren einzelner Funktionalitäten abgenommen. Folgendes Beispiel zeigt und kommentiert Ausschnitte aus dem vom Staffware entwickelten Enterprise Client.

**Beispiel 5.7**: (*Implementierung eines Enterprise Clients [Visual Basic]*)
Ein Enterprise-Client (`SEO VB Sample Application` - SEOVBSample.vbp), inklusive des in Microsoft Visual Basic verfassten Codes, und ein dazu passendes Prozessmodell (`LOAN`)[10] wird mit der SEO-Software ausgeliefert. Dieser Enterprise-Client bietet neben der Standard-Funktionalität eines Clients, noch integrierte Administrations- und Monitoringwerkzeuge (z.B. Benutzer- und Instanzverwaltung). Als Toolkit wurde SEO32/COM verwendet. Das SEO/COM Objektmodell ist [sw00c] zu entnehmen. Es soll an dieser Stelle nur die Verbindungsaufnahme dargestellt werden. Dies beinhaltet die Suche der verfügbaren Staffware-Server und den Login des Benutzers. Die dem Benutzer angezeigte Login-Oberfläche ist in Abbildung 5.17 abgebildet.

[9] Stand: Juni 2001
[10] Dieses Prozessmodell beinhaltet VBA-Formulare und kann nur auf einem Computer mit installierten VBA-Komponenten verwendet werden (siehe Abschnitt 5.4.2.2 und [SW99]).

Abbildung 5.17: Login der SEO VB Sample Application

### Anmelden eines Benutzers nach Serverauswahl

```
Private Sub cmdLogin_Click()
    Dim NodeKeys(0) As String
    On Error GoTo ErrorHandler
    If cboServers.ListIndex < 0 Then
        RetVal = MsgBox("Select Staffware Server.", vbExclamation)
        cboServers.SetFocus
    ElseIf txtUserName.Text = "" Then
        RetVal = MsgBox("User Name is required.", vbExclamation)
        txtUserName.SetFocus
    Else
        Screen.MousePointer = vbHourglass
        'index of item in combobox matches index of node in nodeinfos list
        NodeKeys(0) = oEnterprise.NodeInfos.Item(cboServers.ListIndex).Key    'insert key into array
```

### Anmelden eines Benutzers an den Staffware-Server und Empfang des SWNode-Objektes

```
        Set oNode = oEntUser.Login(NodeKeys, txtPassword.Text, txtUserName.Text)
        cmdLogoff.Enabled = True
        cmdLogin.Enabled = False
        frmMain.Show
        Me.Hide
        Screen.MousePointer = 0
    End If
    Exit Sub
ErrorHandler:
    Screen.MousePointer = 0
    RetVal = MsgBox("Error: " & Err.Number & " - " & Err.Description, vbExclamation)
End Sub
```

### Initialisierungsphase - Suche der verfügbaren Server

```
Private Sub Form_Load()
    Dim oNodeInfo As SWNodeInfo
    On Error GoTo ErrorHandler
    cboServers.Clear              'clear ListBox
    oEnterprise.PollCnt = 3     'set polling value
```

### Initialisieren der Staffware-Server Auswahlbox

```
    For Each oNodeInfo In oEnterprise.NodeInfos
        cboServers.AddItem (oNodeInfo.SWEOSrvDesc)
    Next
    If cboServers.ListCount > 0 Then
        cboServers.ListIndex = 0   'select first node in list
    End If
    Call SetWindowPosition(Me)

    Exit Sub
ErrorHandler:
    RetVal = MsgBox("Error: " & Err.Number & " - " & Err.Description, vbExclamation, "Application error")
End Sub
```

Eine neue und nur mit SEO implementierbare Möglichkeit ist der Support von "anonymen" Benutzern. Anonyme Benutzer benötigen kein vordefiniertes Benutzerkonto innerhalb von Staffware, um etwa Prozessmodelle zu starten oder nach Informationen zu recherchieren. Diese Funktionalität ist von entscheidender Bedeutung für jede E-commerce Applikation im Internet. Anonyme Benutzer werden im `SEO - Server Configuration Utility` auf eine in Staffware existierende Person abgebildet. Die Rechte eines anonymen Benutzers sind somit identisch mit den Rechten dieser Person. Allerdings kann keine Arbeit (*Workitems*) zu anonymen Benutzern weitergeleitet werden.

**Staffware WebClient**
Der von Staffware angebotene WebClient ist eine ASP-basierte (*Active Server Pages*) Webserver-Anwendung. Dadurch erhält der Benutzer mittels Web Browser Zugriff auf Staffware.
Die Vorteile eines WebClients sind:

- Organisationen können ihr Staffware-System über ein Internet oder Intranet ausweiten und ihren Mitarbeitern zu jedem Zeitpunkt an jedem Ort die Prozess-Teilnahme ermöglichen.
- Organisationen können ihre Geschäftspartner (Kunden und Partner) im Prozessablauf integrieren und so die Dauer von Prozessabläufen verkürzen.
- Unterstützung einheitlicher Darstellung von Desktop und webbasiertern Anwendungen.

Zur Kommunikation mit Staffware nutzt der Staffware Web Client `SEO` [SW00i]. Durch VB-Skripte in den ASP-Seiten ruft der Staffware Web Client den `SEO COM Client`. Die Architektur des Staffware Web Client ist in Abblidung 5.18 dargestellt.

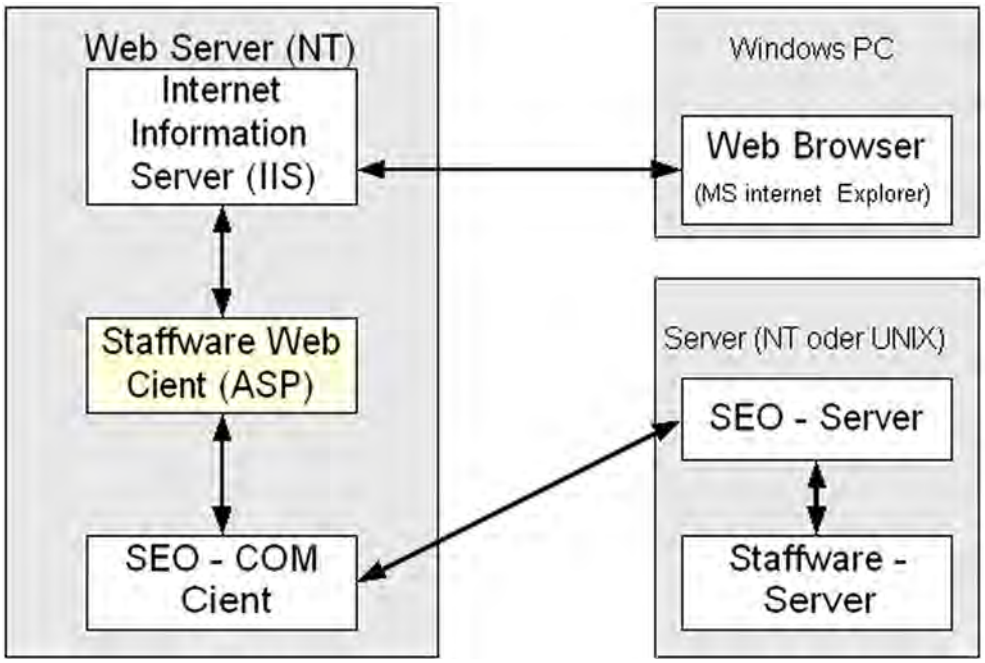



Abbildung 5.18: Die Architektur des Staffware WebClient

Die Hauptbestandteile des Staffware WebClient sind eine Sammlung von ASP-Seiten und ein Form-Konverter.
Jede ASP-Seite besteht aus HTML, VB-Skript und Java-Skript. Sie stellen die notwendige Logik und die Schnittstelle zum Bearbeiter bereit. Die wichtigsten Funktionen (z.B. Login am Staffware-Server) sind in den beiden Dateien `standard.asp` und `Webclient.asp` zusammengefasst [SW00i].

Der Form-Konverter liest im GFD erzeugte Formulare und wandelt diese in ASP-Seiten um. Diese können dann dem Bearbeiter im Web Browser angezeigt werden.
Der Staffware WebClient bietet, mit einigen wenigen Ausnahmen, die Funktionalitäten des Out-of-the-Box Clients. Im Folgenden sind die Funktionalitäten entsprechend der Verfügbarkeit im Staffware WebClient gegliedert.
Mittels GWD und GFD erstellte Prozessmodelle können unter Beachtung dieser Einschränkungen (siehe weiter unten) ohne Änderung webfähig gemacht werden [SW00h].

**Verfügbare Funktionalität [SW00h]:**

- Anmelden an Staffware Server
- Arbeiten mit Arbeitslisten
- Arbeiten mit Workitems
- Prozessinstanzüberwachung
- Starten neuer Prozessinstanzen

**Nicht verfügbare Funktionalität [SW01d]:**

- Listen können nicht verwendet werden
- Datenbanken (*Tables*) können nicht verwendet werden
- Felder können nicht validiert werden
- Skripte werden nicht ausgefürht
- Anmerkungen werden nicht angezeigt
- Das Feld `Anhang` kann nicht verwendet werden
- Anwendungs-Felder können nicht verwendet werden

Zu beachten sind die strengen Installationsvoraussetzungen für den Staffware WebClient. Als Web-Server kann nur der Microsoft Internet Information Server (IIS) und als Web Browser nur der Microsoft Internet Explorer verwendet werden. Grund dafür ist die Verwendung der ASP-Seiten von Microsoft, welche nur durch den IIS interpretiert werden können.

#### 5.4.2.1.3 Externe Clients

Dieser Client-Typ setzt sich aus einer Gruppe von externen Systemen zusammen, welche um die Staffware Funktionalität erweitert wurden. Hierzu zählen Systeme wie

- Microsoft Exchange,
- Outlook,
- Genesis2,
- MEMO und
- Lotus Notes.

Benutzer eines externen Clients erhalten alle ihre im WFMS anstehenden Aufgaben über den Posteingang des externen Systems. Prozessmodelle können aus den externen Clients heraus instanziiert werden.
Eine Besonderheit der externen Clients ist, daß sie nicht unbedingt direkt mit dem Staffware Server verbunden sein müssen, da dieser über die Möglichkeit verfügt, Arbeitseinheiten zu bündeln und diese dann per Mail an den verbundenen Client zu senden. Diese Methode ist am geeignetsten für Remote-User oder User, welche nur selten in Workflow gestützte Prozesse involviert sind.
Eine weitere Methode zur Integration von externen Systemen ist die Nutzung der SEO, um in die jeweilige Anwendung die Workflow-Funktionalitäten einzubinden [SW99d].
In Abbildung 5.19 wird anhand von *Microsoft Exchange* die Kommunikation zwischen Staffware und einem externen Client vorgestellt. Während der Prozesserstellung werden solche Aktivitäten als *Open Client Step* (siehe Kapitel 2) modelliert.

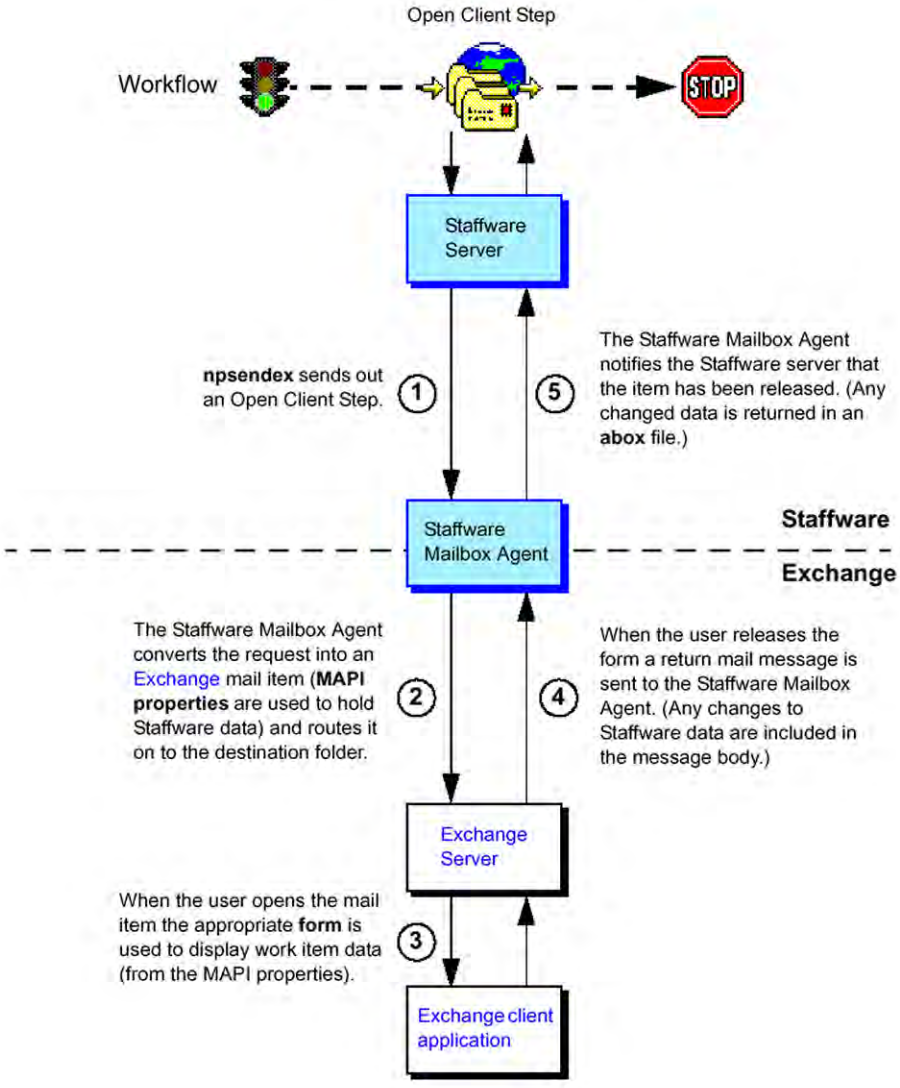



Abbildung 5.19: Die Kommunikation zwischen Staffware und einem externen Client [SW99a]

Der `Staffware Mailbox Agent` stellt ein `Gateway Programm` dar, welches der Verbindungspunkt zwischen Staffware und dem externen Client ist [SW00b]. Weitere Beschreibungen des `Staffware Mailbox Agent` sind in Abbildung 5.19 enthalten.

#### 5.4.2.2 Anwendungsprogrammierung

Staffware bietet mit dem integrierten `GFD` eine einfache und schnelle Möglichkeit, Formulare (Anwendungsprogramme) zu erstellen. Ein einfaches Formular besteht aus Text und Feldern (zur Ein- und Ausgabe von Daten). In Kapitel 2.2.3.3 wurden bereits die zur Verfügung stehenden Feldtypen dargestellt.
Allgemein können Staffware-Formulare zusätzlich folgendes beinhalten (z.B. in Form von Staffware-Kommandos) [SW01b]:

- Programmaufrufe (auf Client und Server)
- Datenbankaufrufe (SQL-Calls)
- Scriptausführung

Tabelle 5.7 fasst die Voraussetzungen zur Anwendungsentwicklung zusammen.

| **Anwendungen** | **Voraussetzungen** | **Standard oder Extra-Tool** |
|---|---|---|
| Staffware Formulare | GFD | Standard-Tool |
| VBA-Projekte | Staffware for VBA (*Visual Basic for Applications*) | Extra-Tool |

Tabelle 5.7: Entwicklung von Anwendungen

Durch das Extra-Tool `Staffware for VBA` kann ein Bearbeiter ein VBA-Projekt erstellen und dieses, wie ein Staffware-Formular, als Anwendungsprogramm einer Aktivität hinterlegen. `Staffware for VBA` integriert sich in den GWD und ermöglicht ein direktes Hinterlegen von VBA-Projekten für Aktivitäten. Die Bereitstellung der Funktionalität der Staffware Formulare für `Staffware for VBA` wird durch ein vordefiniertes Objekt (`OpenFormOCX`) erbracht. Zu beachten ist, daß `Staffware for VBA` die Funktionalitäten von `VB` nicht komplett[11] zur Verfügung stellt [SW99].

### 5.4.3 Interoperabilität zwischen WFMS

Staffware verfügt über vollständig implementierte Schnittstellen zu anderen WFMS. Vorausgesetzt wird die Unterstützung der Einbindung durch das fremde WFMS (z.B. SAP-WF) [SW01b].
Zuerst wurde der Nachrichtenaustausch zwischen WFMS auf MIME-Basis implementiert. Dies wurde im `Staffware SDK` (*Staffware Developper Toolkit*) implementiert.
Nachdem die WFMC ihren XML-basierten Interoperabilitäts-Standard veröffentlicht hatte, wurde die Kommunikation vollständig auf Basis von `WF-XML 1.0` [WFMC00a] implementiert.
Weitere Informationen sind weder aus frei zugänglichen Medien noch von Staffware zu bekommen.

### 5.4.4 Administrations- und Monitoringwerkzeuge

Zur Administration und Überwachung des Staffware WFMS existieren mehrere Tools. Tabelle 5.8 gibt einen Überblick.

[11] Die genaue Bedeutung dieser Aussage konnte in der zugänglichen Literatur nicht festgestellt werden.

| **Anwendungen** | **Aufruf als ...** | **Standard oder Extra-Tool** |
|---|---|---|
| `Administration Manager` | eigenständiges Tool und Tool im Standard-Client | Standard-Tool |
| `Case Manager` (Überwachung von Instanzen) | Tool im Standard-Client | Standard-Tool |
| `SWUTIL` (auf dem Staffware-Client) | Kommandozeilenprogramm | Standard-Tool |
| `SWUTIL` (auf dem Staffware-Server) | Kommandozeilenprogramm | Standard-Tool |
| `SWBATCH` (auf dem Staffware-Server) | Kommandozeilenprogramm | Standard-Tool |
| `EIS-Report` *Executive Information System* | Tool im Standard-Client | Standard-Komponenten und Extra-Komponenten |
| `SWIP` *Staffware Work In Progress* | Tool im Standard-Client | Extra-Tool |
| `SEO - Server Configuration Utility` | eigenständiges Tool | Extra-Tool |

Tabelle 5.8: Administrations- und Überwachungswerkzeuge

Die Tools SWUTIL (auf dem Staffware-Server) und SWBATCH (auf dem Staffware-Server) sind im Verzeichnis `../util/` des Staffware-Servers abgelegt. Das Tool SWUTIL (auf dem Staffware-Client) liegt im Installationsverzeichnis des Staffware-Clients.

#### 5.4.4.1 Administration Manager

Mit dem `Staffware Administration Manager` kann ein Administrator von jedem Staffware Client mit installiertem `Administration Manager` alle Staffware Server im Netzwerk (Multi-Node Konzept siehe Kapitel 3) verwalten. Dem Administrator stehen folgende Optionen zur Verfügung [SW00d]:

- `Benutzerverwaltung` (*User Manager*)
- `Backupverwaltung` (*Backup Manager*)
- `Datenbankverwaltung` (*Table Manager*)
- `Listenverwaltung` (*List Manager*)
- `Instanzverwaltung` (*Case Manager*)
- `Netzwerkverwaltung` (*Network Manager*)
- `Verbreitung von Veränderungen` (*Moving System Information*)

Die Option `Instanzverwaltung` entspricht dem Tool `Case Manager`. Bei Verwendung eines Staffware Servers und Staffware Client ab Version 8.1 kann dieses Tool nur noch aus dem Staffware Clienten aufgerufen werden. Wird dennoch diese Option gewählt, wird der Administrator mit einer aussagekräftigen Fehlermeldung auf diesen Sachverhalt hingewiesen.

#### 5.4.4.2 Überwachung von Instanzen (*Case Manager*)

Der *Case Manager* wird nur Personen mit Administrator-Rechten im Standard-Client angeboten. Dieses Tool bietet dem Benutzer einen Überblick über den Ablauf laufender oder bereits abgeschlossener Prozessinstanzen, z.B.

- Wie weit ist die Ausführung einer bestimmten Prozessinstanz fortgeschritten?
- Welcher Bearbeiter hat welche Aktivitäten ausgeführt?
- Wann begann die Ausführung einer Prozessinstanz?
- etc.

Mit diesem Tool können vier Aktionen ausgeführt werden:

- `Vorzeitiges Beenden einer Instanz`: Dabei werden alle zugehörigen Workitems aus den Arbeitslisten der Benutzer gelöscht. Diese Aktion wird im *Audit Trail* vermerkt.
- `Löschen einer Instanz`: Dabei wird die Instanz und alle zu der Instanz gehörigen Daten gelöscht.
- `Löschen aller Instanzen und Reset eines Prozessmodells`: Dabei werden alle Instanzen eines Prozessmodells gelöscht und der Instanzzähler für dieses Prozessmodell zurückgesetzt.
- `Ansicht des Ablaufprotokolls (`*Audit Trails*`) einer Instanz`: Dabei werden alle Ereignisse, die während einer Instanzausführung angefallen sind, in chronologischer Reihenfolge dargestellt.

Die Einsicht in die CWAD ist nur über dieses Tool zu erreichen. Abfragen können nicht an das Tool gestellt werden. Der Benutzer muß Daten und Zusammenhänge selbst aus den angebotenen Daten gewinnen.

#### 5.4.4.3 Staffware Dienstprogramm - SWUTIL (auf dem Staffware-Client)

Dieses Tool hat keine graphische Oberfläche. Es wird mit entsprechenden Aufrufparametern (siehe Beispiel 5.8) gerufen. Folgende Aufgaben können durchgeführt werden:

- Starten einer Prozessinstanz
- Auslösen eines Ereignisses

**Beispiel 5.8**: (*Syntaxbeispiel eines Prozessstarts*)
Der folgende Befehl startet eine Instanz des Prozessmodells P005 und gibt dieser Instanz die Beschreibung Huber mit.

```
swutil.exe START P005 Huber
```

Um die Aufrufe dieses Tools zum Staffware-Server zu leiten, wird ein gesondertes Programm benötigt. Der `Staffware Utility Server` (*staffsrv.exe*) wird automatisch bei einem Aufruf von SWUTIL gestartet und gibt diesen Aufruf an den Staffware Server zur Bearbeitung weiter [SW00]. Er besitzt keine Oberfläche und tritt für den Benutzer nach seinem Start nicht in Erscheinung. Der `Staffware Utility Server` bleibt aktiv bis der Computer ausgeschaltet oder das Programm explizit geschlossen wird. Dadurch erfolgt die Ausführung weiterer Aufrufe schneller als die Ausführung des ersten Aufrufs.

#### 5.4.4.4 Staffware Dienstprogramm - SWUTIL (auf dem Staffware-Server)

Dieses Tool ist ein Kommandozeilenprogramm, welches beim Aufruf (ohne Parameter) dem Benutzer ein Menü zur Auswahl der möglichen Arbeitsschritte anbietet [SW00d]. Folgende Aufgaben werden im Menü des Tools dem Benutzer angeboten:

- Ausgabe einer Datenbank (*Staffware-Table*)
- Löschen einer Datenbank (*Staffware-Table*)
- Ausgabe aller zu einer Instanz gehörenden Daten
- Exportieren eines Prozessmodells
- Exportieren einer Datenbank (*Staffware-Table*)
- Exportieren einer Liste
- Importieren einer XFR-Datei
- Start einer Prozessinstanz
- Start einer Prozessinstanz mit Daten zur Initialisierung der Staffware-Felder
- Verlassen des Menüs

Soll die Ausgabe der Daten einer Instanz formatiert werden, muß der Aufruf in der Kommandozeile erfolgen.
Am häufigsten wird dieses Tool zum Starten von Prozessinstanzen aus einer belieben Anwendung heraus verwendet. Dabei kann die Prozessinstanz mit und ohne Daten per Kommandozeilenaufruf gestartet werden [SW01e].

#### 5.4.4.5 Ausführung vieler Aktionen - SWBATCH (auf dem Staffware-Server)

Dieses Tool ermöglicht es dem Benutzer, eine große Anzahl an

- Instanzen zu starten (mit und ohne zugehörigen Daten)
- Ereignissen auszulösen (mit und ohne zugehörigen Daten)
- Instanzen zu beenden
- Instanzen zu löschen

Dieses Tool ist speziell für die Verarbeitung einer großen Anzahl an Operationen (z.B. starten oder löschen von Instanzen) optimiert [SW00d]. Sollen individuelle Operationen ausgeführt werden, so eignet sich das Tool `SWUTIL` besser.

#### 5.4.4.6 Auswertung von Daten einer Prozessinstanz - EIS

Das EIS (*Executive Information System*) ermöglichst es einem Prozessmodellierer, aus den Daten einer Instanz (welche in einer csv-Datei gespeichert werden) einen Report, im Sinne einer aufbereiteten Darstellung ausgewählter Daten, zu erstellen [SW00b]. Dieser Report kann mit Programmen, die `csv`-Dateien interpretieren können, betrachtet werden. Staffware `EIS` beinhaltet Standard- und Extra-Komponenten. Die Extraktion der Daten aus den Staffware-Server wird von dem Standard-Tool `SWCDATA.exe` übernommen. Eine Darstellung der erzeugten csv-Datei erfolgt in einem externen Betrachtungsprogramm. Dieses wird nicht von Staffware angeboten.

#### 5.4.4.7 Graphischer Arbeitsmonitor - SWIP (*Staffware Work In Progress*)

Das SWIP-Tool ermöglicht es dem System-Adminstrator, sich einen Überblick über die am Staffware-Server anstehenden Aufgaben zu erhalten. Es kann zwischen folgenden Darstellungen gewählt werden [SW99c]:

- Arbeitslisteneinträge ausgewählter Personen oder Gruppen. Dabei werden Arbeitslisteneinträge mittels `neu`, `deadline` und `gesamt` unterteilt (siehe Abbildung 5.20).
- Arbeitslisteneinträge einer Person oder Gruppe. Diese werden mittels zugehörigen Prozess und Status geordnet.
- "Alter" von Arbeitslisteneinträgen. Als Option stehen `Stunden`, `Tage` oder `Wochen` zur Verfügung.
- Arbeitslisteneinträge innerhalb eines bestimmten Zeitraums.

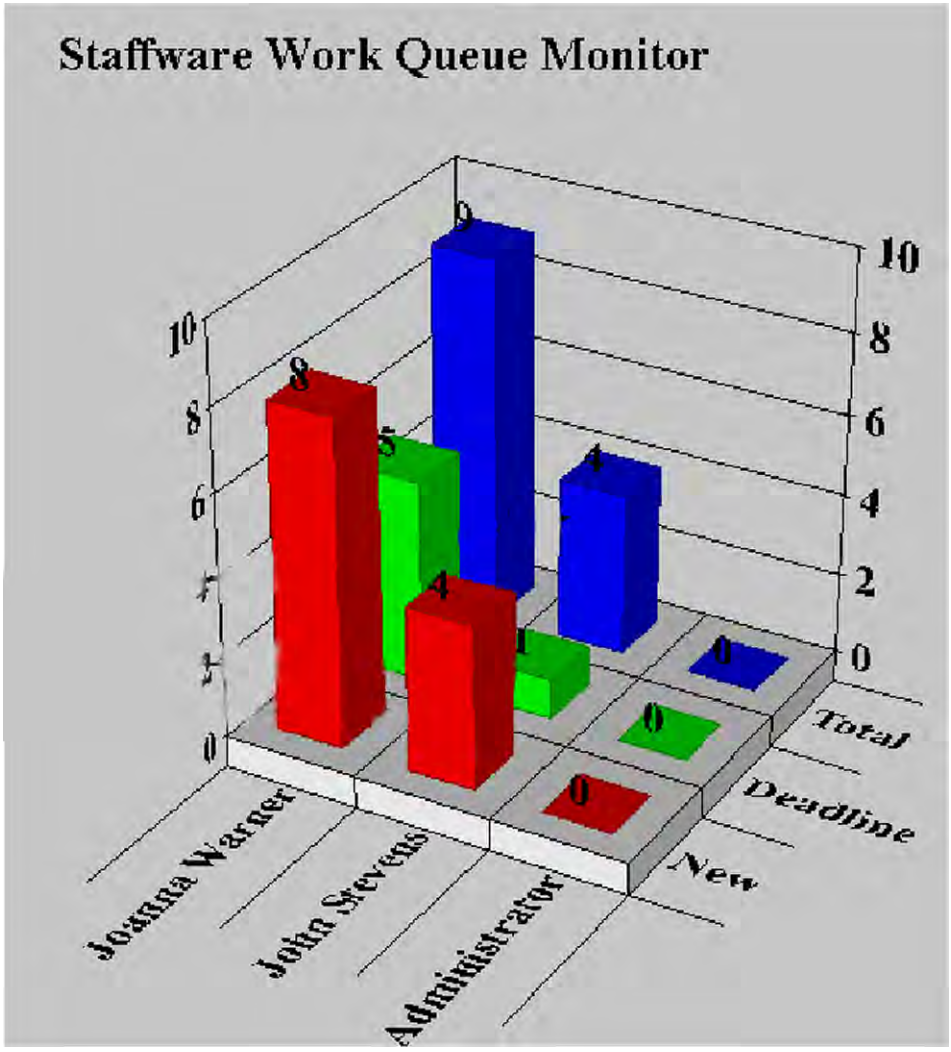



Abbildung 5.20: graphische Darstellung ausgewählter Arbeitslisten

#### 5.4.4.8 SEO - Server Configuration Utility

Dieses Konfigurationstool ermöglicht die Adminstration des SEO-Servers. Dabei werden Zugriffmöglichkeiten (im Sinne von Benutzerverwaltung), Verbindungsmöglichkeiten zu einem oder mehreren Staffware-Server, Cache- und Protokollverwaltung konfiguriert.

## 5.5 Vergleich der Systeme

Beide WFMS sind noch weit von einer vollständigen Implementierung der Standards der WFMC entfernt. Dieser Abschnitt soll Unterschiede in der Implementierung der Schnittstellen und daraus resultierende Vor- bzw. Nachteile aufzeigen.

**Schnittstelle 1**: `Staffware` ermöglicht nicht das gleichzeitige Exportieren mehrerer Prozessmodelle aus der Laufzeitumgebung mittels `swutil.exe`. Da beispielsweise Subprozesse nur auf diese Weise gesichert werden können (siehe Kapitel 6) ist dieses Verhalten unverständlich und sehr hinderlich. `MQSeries WF` dagegen ermöglicht problemlos das Exportieren mehrerer Prozessmodelle aus der Buildtime-Komponente.
Beide WFMS verfügen über eine eigene WPDL, die vom jeweils anderen WFMS nicht verwendet werden kann. Allerdings existieren Programme anderer Hersteller, welche diese WPDL unterstützen.

**Schnittstellen 2 & 3**: `MQSeries WF` bietet eine Vielzahl von APIs zur Programmierung von eigenen Clients und Anwendungsprogrammen. Durch die gute Dokumentation und viele Beispielprogramme ist eine schnelle Einarbeitung und Entwicklung eigener Komponenten möglich.
`Staffware` benötigt eine Server- und Clientseitige Erweiterung, um selbstprogrammierte Clients mit dem Server verbinden zu können. Zu diesem Zweck werden die SEO benötigt. Es entsteht ein nicht zu unterschätzender Mehraufwand. Eine Anwendungsprogrammierung ist nur mit *Staffware for VBA* möglich und stellt eine weitere Einschränkung dar.

**Schnittstelle 4**: `Staffware` hat diese Schnittstelle auf WF-XML-Basis implementiert. Bei `MQSeries WF` fehlt noch eine offizielle Stellungnahme seitens der WFMC, um die in diesem Kapitel vorgestellten Entwicklungen zu bestätigen. Bei beiden WFMS sind die zugänglichen Informationen sehr rar und teilweise widersprüchlich, so daß eine abschließende Urteilsbildung nicht erfolgen kann.

**Schnittstelle 5**: Das graphische Monitoring-Tool von `MQSeries WF` liefert fast keine Daten. Es kann nur der Zustand der Aktivitäten einer Prozessinstanz überwacht werden. Die Instanzüberwachung in `Staffware` stellt dem Benutzer mehr Daten zur Verfügung. So kann dort z.B. der Start-Termin einer Aktivität oder deren Bearbeiter eingesehen werden. Für weitere Auswertungen bietet Staffware gesonderte Tools, während MQSeries lediglich auf die Möglichkeit der Auswertung des Audit Trails in der Runtime-Datenbank verweist. Ein spezielles Tool wird seitens MQSeries WF nicht empfohlen. Ebensowenig werden Möglichkeiten zur gezielten Abfrage dieser Datenbankstruktur vorgestellt.

# Kapitel 6

# Fortschrittliche Funktionalitäten

Neben den besprochenen und meist auch implementierten Funktionalitäten von WFMS existieren weitere Anforderungen, welche eine sinnvolle Erweiterung von WFMS darstellen. Diese sind bei den beiden Test-WFMS und den meisten anderen WFMS noch nicht implementiert, da deren Einführung sich nicht immer einfach in die existierenden Architekturen eingliedern lassen. In den folgenden Abschnitten werden fortschrittliche Funktionalitäten vorgestellt und der Versuch unternommen, Bezug auf die beiden WFMS MQSeries WF und Staffware herzustellen.

## 6.1 Ad-hoc-Änderungen

Bisher wurde davon ausgegangen, daß Geschäftsprozesse einen starren Ablauf von Tätigkeiten darstellen und in ein starres Prozessmodell abgebildet werden können. Es existieren jedoch Anwendungsgebiete (z.B. Krankenhausprozesse) deren Arbeitsabläufe eine hohe Variabilität und Dynamik aufweisen [DaReKu00]. Dadurch wird eine Abbildung aller Varianten der Prozessausführung (inkl. aller Ausnahmen) in ein Prozessmodell und somit eine starre rechnergestützte Ausführung häufig unmöglich. In solchen Arbeitsabläufen kommt es öfters vor, daß vom geplanten Ablauf spontan abgewichen werden muß (z.B. plötzliche Verschlechterung des Zustandes eines Patienten). Deshalb muß es berechtigten Personen möglich sein, spontan vom vormodellierten Ablauf abzuweichen. Jede Form einer solchen Abweichung wird als Ad-hoc Abweichung [ReDa98] bezeichnet. Beispiele für Ad-hoc-Änderungen sind:

- Überspringen von Aktivitäten
- Verschieben von Aktivitäten
- Einfügen von Aktivitäten
- Zurückspringen von Aktivitäten
- Zurücksetzen von Aktivitäten

In Abbildung 6.1 werden die Ad-hoc-Änderungen graphisch veranschaulicht.

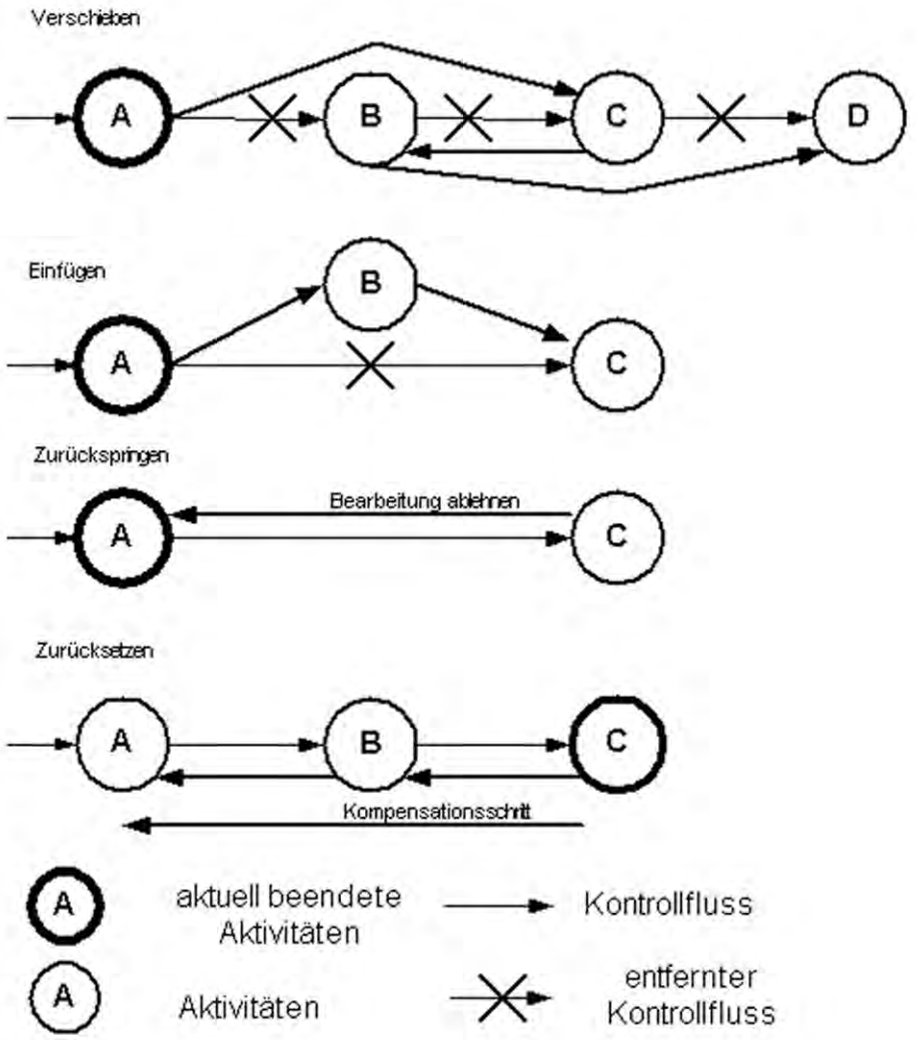



Abbildung 6.1: Ad-hoc-Änderungen (aus [Dad99])

Die dargestellten Ad-hoc-Änderungen werden auf Instanzebene vollzogen. Das bedeutet, daß das zugrundeliegende Prozessmodell oder andere Prozessinstanzen von den Änderungen nicht beeinflusst werden.

Wie bereits in Kapitel 4 angedeutet, bieten heutige WFMS kaum Unterstützung im Hinblick auf Ad-hoc-Änderungen [Re00]. Auch die beiden Test-WFMS bilden hier keine Ausnahme.

**MQSeries WF** verhindert jeden Eingriff in die Kontrolle laufender Instanzen. Bei der Instanziierung eines Prozessmodells wird eine eigenständige Kopie des Prozessmodells angelegt, auf welche aber keinerlei Änderungen mehr propagiert werden. Dieser Ansatz erscheint aber sehr günstig um Ad-hoc-Änderungen einführen zu können. Da jede Instanz eine eigene Kopie des Prozessmodells besitzt, könnte diese auch abgeändert werden, ohne andere Instanzen zu beeinflussen.
Änderungen an einem Prozessmodell werden in **Staffware** auch für laufende Instanzen gültig (siehe Kapitel 4). Dadurch kann in den Kontrollfluss eingegriffen werden und beschriebene Szenarien wie Überspringen, Einfügen oder Verschieben erreicht werden. Allerdings kann dies nicht als Implementierung von Ad-hoc-Änderungen gesehen werden. Diese Form der dynamischen Änderungen können in Staffware nicht für einzelne Instanzen eingebracht werden, sondern verändern unwiederruflich das Prozessmodell (*Schemaänderung*). Diese Änderungen gelten für neue Instanzen und für laufende Instanzen gleichermaßen. Dabei kann es allerdings, wie in Kapitel 4 gezeigt,

zu schweren Fehlern oder Inkonsistenzen bei der Ausführung einzelner Instanzen kommen.

## 6.2 Robustheit von WFMS

Ein robustes WFMS ist unempfindlich und wiederstandsfähig gegen äußere, evtl. auch unvorhersehbare, Einflüsse. Ziel ist es, daß ein WFMS nach einem Ausfall möglichst schnell wieder einsatzfähig ist. Es existieren zwei Verfahren um dieses Ziel zu erreichen. Je nach Art des äußeren Einflusses der zum Ausfall führte greift:

- Recovery (z.B. Wiederanlauf der WFMS nach Stromausfall)
- Backup (z.B. Wiederherstellen eines WFMS nach Totalverlust (z.B. Brand))

### 6.2.1 Revovery von WMFS (*Phoenix Behavior*)

Nach einem unplanmäßigen Stop (*System Crash*) eines WFMS (z.B. durch Stromausfall oder Defekt eine Hardware-Komponente) muß ein korrekter Wiederanlauf gewährleistet sein. Die zum Zeitpunkt des *system crash* aktiven Instanzen müssen ihre Ausführung fortsetzen können, unabhängig vom Zustand in dem sie sich befunden haben. Die dazu benötigten Funktionalitäten müssen vom WFMS bereitgestellt werden. Dieses Verhalten von WFMS wird als *Phoenix Behavior* bezeichnet (*"Wie Phoenix aus der Asche"*) [LeRo00].

Aus den beschriebenen Anforderungen wird deutlich, das WFMS anders auf einen *System Crash* reagieren müssen als beipsielsweise Datenbank Management Systeme (DBMS). Diese setzen beim Wiederanlauf alle zum Zeitpunkt des *system crash* aktiven Arbeiten komplett zurück. Dies ist möglich, da es sich hier im allgemeinen um kurze (Laufzeit) Arbeiten handelt. Bei WFMS ist dieses Verhalten nicht einsetzbar. Dafür stehen mehrere Gründe:

- Arbeitsabläufe eines WFMS können einen langen zeitlichen Raum in Anspruch nehmen, wodurch hohe Kosten entstehen können.
- Die Wiederholung eines Arbeitsschrittes würde zu einer doppelten Ausführung führen. Dies ist nicht tolerabel, da Arbeitsschritte Bezüge zur realen Welt haben können und nicht wiederholbar sein müssen (z.B. Überweisung von Geld).

Aus diesen Gründen muß ein WFMS für jede aktive Prozessinstanz einen persistenten Kontext bereitstellen und pflegen. Dieser kann beispielsweise in der Datenbank des WFMS gespeichert werden. In diesem persistenten Kontext werden alle WF-relvante Daten und Ereignisse gespeichert (z.B. Erstellen oder Start eines Workitems).
Im Falle eines *System Crash* können nun nach dem Wiederanlauf des WFMS alle Instanzen entsprechend ihres gespeicherten Kontextes eingerichtet werden. Abbildung 6.2 stellt das Speichern der WF-relevanten Daten und deren Verwendung beim einem Wiederanlauf eines WFMS nach einem *system crash* dar.
Noch ungeklärt ist das Verhalten des WFMS, wenn zum Zeitpunkt des Absturzes Workitems aktiv waren (z.B. laufende Anwendungen). Hier kann nur eine Analyse feststellen ob das zugehörige Anwendungsprogramm bereits Aktionen ausgeführt hat, welche Auswirkungen außerhalb des Anwendungsprogramms haben. Abhängig davon kann ein erneuter Start nötig oder ein weiteres Ausführen möglich sein.

Im folgenden werden durch Unterbrechung der Stromzufuhr und Neustart des Test-Computers ein *system crash* für beide Test-WFMS simuliert.

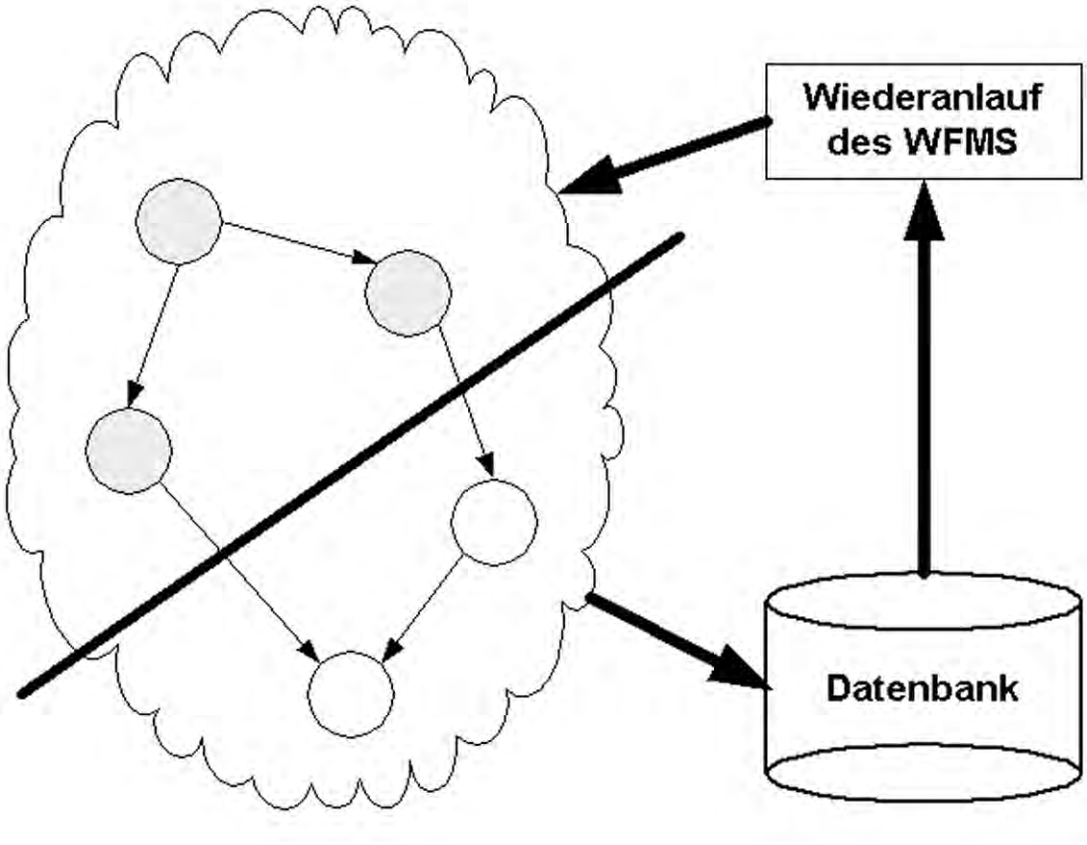



Abbildung 6.2: Recovery bei WFMS (aus [LeRo00])

Beide WFMS nutzen ihre Speichermedien (Datenbanken oder File-Systeme) zur Speicherung WF-relevanter Daten (siehe Kapitel 3).
**MQSeries WF**:
Aus der Dokumentation von MQSeries WF lassen sich keine Rückschlüsse auf eine besondere Recovery-Art ziehen. Es besteht nur der Hinweis, daß der Verwaltungsserver für die Fehlerbehung aller Komponenten zuständig ist [IBM99b].
Ein Wiederanlauf des WFMS ohne kompletten Verlust der in Ausführung befindlichen Instanzen ist jedoch möglich[1].

**Staffware**:
Die Dokumentation von Staffware greift das Thema Recovery ebenfalls nicht auf.
Die Tests ergaben folgende Erbegnisse:

Ein zum Zeitpunkt des Absturzes des Rechners im GWD geöffnetes Prozessmodell ist nach dem Wiederanlauf des WFMS gesperrt. Es kann von niemanden, auch nicht vom Administrator, geöffnet werden. Grund dafür ist, daß Staffware das Öffnen und Schließen eines Prozessmodells in einer Datei mitprotokolliert. Da bei einem Absturz das Schließen nicht stattfand gilt das Prozessmodell als geöffnet und kann nicht nochmals geöffnet werden. Lösung dieses Problems ist das Löschen der Protokoll-Datei (*swserver/computername/tsys/*sph*).

[1]Unter OS/390 kann der integrierte Startmechanismus (*automatic restart manager*) verwendet werden. Diese Einstellung wird in den Systemeigenschaften in der Buildtime-Komponente vorgenommen.

### 6.2.2 Backup

Ein Backup ist eine gesonderte Speicherung benutzerspezifischer Daten. Durch diese Daten soll es möglich sein, das vorhandene System (hier WFMS), beispielsweise nach einem Datenverlust, wieder in den Zustand des Zeitpunktes des Backups zurücksetzen zu können. Alle nach dem Backup eingebrachten Änderungen können später auch nicht wiederhergestellt werden. Deshalb ist eine überdachte Backup-Strategie für viele Unternehem zwingend erforderlich.
Um ein WFMS wiederherstellen zu können, müssen alle Prozessmodelle und Anwendungprogramme in das Backup einbezogen werden.
**MQSeries WF** verfügt über kein integriertes Backup-Verfahren und weist in der Dokumentation auch nicht auf die Notwendigkeit eines Backups hin. Nur für den Fall, daß die Buildtime-Datenbank reorganisiert werden soll, wird ein Backup empfohlen. Dazu ist die Sicherung der Datei `fmcbtdb.mdb` nötig. Dies beinhaltet nicht die aktuell verwendeten Anwendungsprogramme. Auf diese Art und Weise ist nach einem Verlust von Daten keine Wiederherstellung des WFMS möglich.
**Staffware** bietet im Administration Manager das Erstellen und auch das Einspielen bereits Erstellter Backups an. Erstellt werden können [SW00d]:

- `System-Backups`: Es werden alle Prozessmodelle, Instanzen und System-Daten in das Backup einbezogen. Dieses Backup ermöglicht ein vollständiges Wiederherstellen des Zustandes zum Zeitpunkt des Backups. Solch ein Backup sollte regelmäßig durchgeführt werden.

- `Prozessmodell-Backup`: Es werden ein Prozessmodell und deren Instanzen in das Backup einbezogen. Dies wird vor Änderungen am Prozessmodell empfohlen. Zu beachten ist, daß ein Backup von Prozessmodellen die Subprozesse beinhalten und Subprozessen selbst nicht möglich ist. Dies müssen mittels `swutil.exe` als XFR-Datei gesichert werden (siehe Kapitel 5).

Diese Backup-Verfahren legen auf der Festplatte im Verzeichnis des Staffware-Server einen neuen Ordner an (*../swbkp*).
Ein Backup das auf dem selben Rechner des WFMS selbst erfolgt, ist aus Sicherheitsgründen nicht zu empfehlen, da z.B. ist bei Defekt der Festplatte auch gleich das Backup verloren geht. Empfehlenswert ist, daß Backup-Verzeichnis auf externe Datenträger zu kopieren. Dies ist ebenfall nötig, fall ein Backup längerfristig gesichert werden soll. Ein erneutes Backup würde die Daten des vorhergegangenen Backups überschreiben.

## 6.3 Animation und Simulation

Durch eine graphische Animation von Prozessmodellen soll der Prozessmodellierer Einsicht

- in den Ablauf von Kontroll- und Datenfluss,

- in den Austausch von Daten zwischen Aktivitäten und Anwendungsprogrammen und

- in die Zuteilung der Workitems

erhalten [LeRo97]. Dazu verhält sich ein Animations-Tool ähnlich einem Debugger. Der Prozessmodellierer kann Schritt für Schritt (vorwärts und rückwärts) durch das Prozessmodell gehen und die Reaktionen bzw. das Verhalten austesten.
Die beiden WFMS MQSeries WF und Staffware bieten keine graphische Animation von Prozessmodellen. Flowmark, das Vorgängerprodukt von MQSeries WF, verfügte über eine graphische Animations-Komponente. Eine genauere Beschreibung der Möglichkeiten kann in [LeRo00] eingesehen werden.

Eine wesentliche Erweiterung gegenüber der Animation stellt die Simulation dar. Die Simulation unterteilt sich in die [LeRo97]

1. `analyitsche Simulation` und die
2. `diskrete Simulation`.

Bei der analytischen Simulation wird ein Prozessmodell betrachtet und dessen Bedarf an Personen (Anzahl und Zeit) und Ressourcen bestimmt. Durch monetäre Bewertung der Arbeitszeit kann beispielsweise eine Kostenbewertung des Prozessmodells erfolgen.
Während der diskreten Simulation wird betrachtet, wie mehrere Prozessinstanzen mit den am Ablauf beteiligten Ressourcen (Prozessoren, Datenbanken) auskommen.

Aus der Kombination der Ergebnisse beider Simulations-Arten kann ein Prozessmodell optimiert und/oder den äußeren Umständen angepasst werden.
So kann eine gute Simulation ein frühzeitiges Erkennen von Fehlern (z.B. eine Person bekommt zu einem Zeitpunkt zu viel Arbeit zugewiesen) ermöglichen. Dadurch können Geschäftsprozesse, deren Ausführung zu Problemen (z.B. finanziell, Überschreiten eines Budgets) führen ,rechtzeitig abgebrochen werden.

**MQSeries WF** und **Staffware** bieten keine direkten Möglichkeiten zur Simulation von Prozessmodellen. Jedoch bietet Staffware die Möglichkeit im GWD erstellte Prozessmodelle in Simulationstools zu importieren. Durch diesen Schritt ist eine indirekte Simulation von Staffware-Prozessmodellen möglich (siehe Kapitel 5.4.1).

## 6.4 Zeitmanagement

Ein Zeitmanagement ist in heutigen WFMS nur in sehr geringem Umfang implementiert. So bieten MQSeries WF und Staffware nur die Möglichkeit, die Ausführungszeit von Aktivitäten zu überwachen. Es kann noch nicht von einem "echten" Zeitmanagement gesprochen werden. Dazu müßten noch einige andere Einstellungen möglich sein, wie z.B.:

- Angabe von Zeitabständen zwischen dem Ausführungsende einer Aktivität und dem Start der Nachfolgeaktivität.
- Angabe eines Datum oder Uhrzeit, ab der das zu einer Aktivität gehörende Workitem in Arbeitslisten bereitgestellt werden darf.
- Eine festgelegte Zeitspanne nach Beendigung einer Prozessinstanz startet erneut eine Prozessinstanz desselben Prozessmodells (evtl. mit Datenbezug auf die jeweils vorhergehende Prozessinstanz).

Ein entscheidender Unterschied zu den heutigen Möglichkeiten ist, daß bei obigen Beispielen ein Workitem, abhängig von den Einstellungen, erst zu einem bestimmten Zeitpunkt in einer Arbeitsliste erscheinen darf. Die in MQSeries WF und Staffware implementierter Überwachung der Ausführungszeit bezieht sich nur auf Workitems, die bereits in Arbeitslisten enthalten sind.

## 6.5 Sicherheit

Ein WFMS sollte zu jedem Zeitpunkt nur für autorisierte Personen zugänglich sein. Diese Personen können, entsprechend ihren Rechten, Teile oder sogar die ganze Funktionalität des WFMS nutzen. Diese Vorgehensweise ist nötig, da beispielsweise nicht jeder Benutzer eines WFMS die

Rechte zum Löschen von Prozessmodellen besitzen sollte. Üblicherweise werden autorisierte Personen mit Attributen versehen, welche ihre Rechte verdeutlichen (z.B. Admin, Manager, User, etc.).
Ein spezielles Problem von WMFS ist, daß an der Ausführung eines Prozessmodells eine Vielzahl von Systemen (Betriebssysteme, TP Monitore, ORB, etc.) beteiligt sein können. Daher müßte jede Person einzeln allen Systemen bekannt gemacht werden. Ein Lösungsansatz ist die Übertragung dieser Aufgabe an ein `Security Management System` (SMS) [LeRo00]. Das SMS verwaltet Personen (Gruppen und Rollen) und informiert alle Systeme, falls eine neue Person angelegt wird. Diese Möglichkeit spielt bei heutigen WFMS noch keine Rolle, stellt aber eine attraktive Möglichkeit dar, den Administrationsaufwand zu reduzieren.

Seitens der **WFMC** gibt es bis heute nur die Festlegung, daß bei der Erstellung einer Verbindung zur WF-Engine ein Benutzername und ein Passwort benötigt wird [WFMC98b]. Alle weiteren Sicherheitsmechanismen sind bisher den Herstellern von WFMS überlassen.

Alle Komponenten von **MQSeries WF** verlangen einen Benutzernamen und ein Passwort zur Authentisierung[2]. Dabei kann zwischen der Buildtime und der Runtime unterschieden werden. Eine in der Buildtime neu hinzugefügte Person besitzt sofort die Rechte zum Starten der Buildtime-Komponente. Innerhalb der Buildtime-Komponente kann eine Person nur Aktionen abhängig von der ihr bei der Erstellung gewährten Rechten vornehmen. Alle möglichen Einstellungen sind in Kapitel 2.1.3.1 bereits diskutiert worden. Damit eine in der Buildtime registrierte Person eine der Runtime-Komponenten starten kann, muß sie mittels Import-/Export-Dienstprogramm (siehe Kapitel 3.1.2) importiert werden.

**Staffware** arbeitet bei Sicherheitsfragen sehr eng mit dem Betriebssystem zusammen. So können nur Personen im Administration Manager hinzugfügt werden, die dem Betriebssystem bekannt sind. Die existierenden Passwörter gelten automatisch für alle Staffware-Komponenten. Die Möglichkeiten zur Rechtevergabe an eine Person wurden bereits in Kapitel 2.2.3.1 ausführlich besprochen. Problematisch ist, daß die während der Installation erstellten Staffware-Administratoren im `Benutzer- Manager` des Betriebssystems gelöscht werden können. Wurden keine weiteren Personen mit Administratoren-Rechten angelegt, kann der Zugriff auf bestimmte Komponenten (z.B. Admin Manager) unmöglich werden und somit das WFMS unbrauchbar.

Vorteile von MQSeries WF:

- Die Buildtime-Komponente ist unabhängig von der Runtime-Komonente.
- Es können beliebige Personen verwendet werden (kein Zusammenhang mit Betriebssystem)

Vorteile von Staffware:

- Eine Bearbeiterzuordnung kann nur mit Personen erfolgen, die der WF-Engine zum Zeitpunkt der Modellierung bekannt sind und somit später auch wirklich zur Verfügung stehen.
- Es müssen keine Personendefinitionen importiert bzw exportiert werden.

Ein prinzipieller Unterschied zwischen MQSeries WF und Staffware liegt in der Art und Weise, wie Personen der Zugriff auf Prozessmodelle gestattet wird. MQSeries ordnet die Prozessmodelle in Kategorien und gibt Personen Zugriffsrechte auf Basis dieser Kategorien. Staffware hingegen erlaubt primär jeder registrierten Person den Zugriff auf alle Prozessmodelle. Soll dies nicht geschehen, muß bei der Erstellung des Prozessmodell angegeben werden, wer dieses starten darf.

[2]Nach der Installation gilt für alle Komponenten der Login=ADMIN und das Passwort=password

Beide WFMS stellen auf verschiedene Art sicher, daß nur authorisierte Personen mit dem WFMS arbeiten können. Eine gezielte Beschränkung der für eine Person nutzbaren Funktionalitäten der WFMS ist ebenfalls möglich.

## 6.6 Performance und Skalierbarkeit

Ein WFMS benötigt bei der Ausführung Ressourcen des Computers (z.B. den Prozessor), genauso wie jede andere Anwendung auf diesem Computer. Daher findet eine Art "Verteilungs-Kampf" um die begrenzten Ressourcen statt, wodurch die Performance einzelner Anwendungen leiden kann. Werden Performance-Ziele (z.B. Ausführungsdauer unter 5 Sekunden) für (Teil-) Anwendungsprogramme gesetzt, so wird ein Tool benötigt, daß die Ressourcenzuweisung übernimmt. Diese Aufgabe kann von einem `Workload Management System` übernommen werden [LeRo00]. MQSeries WF und Staffware bieten kein integriertes Workload Management System an. Ein Beispiel für ein solches System ist WLM/MVS, daß in OS/390-Betriebssystem integriert ist.

**MQSeries WF** gibt dem Administrator keine Werkzeuge zur Performance-Verbesserung. Wird die Arbeitslast (*engl. Workload*) für einen MQSeries WF-Server zu hoch, kann jederzeit ein oder mehrere weitere Server der bestehenden Installation hinzugefügt werden [IBM99a]. Alle Server arbeiten auf der selben Runtime-Datenbank. Die Arbeitslast verteilt sich somit gleichmäßig auf alle vorhandenen Server.
Im Falle einer Serverinstallation auf dem Betriebssystem OS/390 kann das obern erwähnte Workload Management System verwendet werden.

**Staffware** bietet die Möglichkeit, durch verschiedene Einstellungen in einigen Bereichen, Einfluss auf die Performance zu nehmen.
Einstellungen können vorgenommen werden in Bezug auf WQS und WIS (siehe Kapitel 3) [SW00d]:

- Anzahl WIS-Prozesse: Ein WIS-Prozess verwaltet die Darstellung der ihm zugewiesenen Arbeitslisten. Die maximal sinnvolle Anzahl an WIS-Prozessen entspricht der Anzahl der Arbeitslisten.

- WQS-Update-Intervall: Das Zeitintervall, nachdem nach neuen Arbeitslisten von neuen Benutzern oder Gruppen gesucht wird. Nach der Installation ist dieser Wert relativ klein. In großen Systemen, mit üblicherweise geringen Wechsel an Benutzern und Gruppen, sollte das Intervall drastisch erhöht werden (ca. 1 Stunde).

- WIS-Update-Intervall: Das Zeitintervall, indem sich die Arbeitslisten aktualisieren. Nach der Installation ist dieser Wert für große Systeme zu gering und sollte dringen erhöht werden (ca. 1 Minute).

Diese Werte dienen ausschließlich dazu, daß System von zu großer Arbeitslast zu befreien. Dies hat direkten Einfluss auf die Performance des gesamten WFMS. Performance-Einstellungen für einzelne Bereiche (Ausführungsdauer einer Aktivität) können nicht getroffen werden.

Bezüglich der Skalierbarkeit verfügt MQSeries WF über eindeutige Vorteile gegenüber Staffware. Dies liegt an der zusätzlichen Unterstützung des Betriebssystems OS/390 als Server-Plattform. Staffware hingegen bietet nur die Unterstützung von UNIX und Windows NT als Server-Plattform, welche MQSeries WF ebenfalls unterstützt.

## 6.7 Transaktionskonzepte

Ein Prozessmodell setzt sich bekanntermaßen aus Aktivitäten zusammen. All diese Aktivitäten stellen einen Geschäftsprozess aus der realen Welt dar und können als eine Arbeitseinheit betrachtet werden. Damit ist gemeint, daß Fehler in der Ausführung einzelner Aktivitäten starken Einfluss auf andere Aktivitäten bzw. auf den gesamten Erfolg des Geschäftsprozesses haben können. Beispiele dafür sind Geschäftsprozesse, welche beispielsweise Reservierungen von Flügen oder Hotelzimmern oder Geldtransfers beinhalten.
Bei dieser Problemstellung liegt der Gedanke an Transaktionsmechanismen sehr nahe. Die Eigenschaften von Transaktionen mit ACID-Semantik werden in [Dad96] und [LeRo00] genauer besprochen. Es wurde relativ schnell der Ausdruck *transactional workflows* gefunden. Eigenschaften von *transactional workflows* sind [LeRo00]:

- *run and gun*: Teile eines Prozessmodells werden zwingend nacheinander ausgeführt.
- *all-or-nothing*: Jede einzelne Aktivität stellt eine Transaktion dar. Eine Folge von Aktivitäten soll vom WFMS als einzelne verteilte Transaktion ausgeführt werden.
- *business-oriented*: Es können transaktionsbasierte und nicht transaktionsbasierte Aktivitäten zu einer Arbeitseinheit zusammengefasst werden (*Compensation spheres*).

Bis Heute erfüllen nur sehr wenige WFMS eine oder mehrere der oben genannten Eigenschaften. Ein Beispiele für eine Implementierungen ist `SAGAS`.
Die Idee von **SAGAS** ist, daß ein Prozess durch eine Verkettung von atomaren Transaktionen dargestellt werden kann. Für jede Aktivität werden Kompensationsinformationen gespeichert, um diese bei Abbruch des Prozesses rückgängig machen zu können. Wird eine Transaktion in dieser Kette von Transaktionen abgebrochen, so wird diese Transaktion durch ein klassisches Rollback rückgängig gemacht. Vorhergehende erfolgreich abgeschlossene Transaktionen werden durch die Kompensationsinformationen rückgängig gemacht. Ist die letzte Transaktion in der Kette erfolgreich abgeschlossen, ist der Prozess entsprechend den geforderten Transaktions-Eigenschaften erfolgreich abgeschlossen (siehe [Dad99] und [LeRo00]).

**MQSeries WF** bietet teilweise Unterstützung von Transaktionen. Für die nicht überwachte Ausführung von Backend-Programmen (als Transaktion) wird auf dem Server ein PES (siehe Kapitel 5.3.2.2) verwendet. Der PES steht nur für OS/390 (R) zur Verfügung und unterstützt den Aufruf von IMS- und CICS-Transaktionen [IBM99b].
In [IBM99a] wird bei der Einrichtung des Verwaltungs-Server als NT-Dienst das `Two-Phasen-Commit`-Protokoll [Dad96] erwähnt. Weitere Informationen über dessen Anwendung sind nicht verfügbar.

**Staffware** verfügt über keine integrierten Transaktionsmechanismen.

# Kapitel 7

# Zusammenfassung und Ausblick

WFMS und im speziellen Production WFMS bieten bereits heute eine gute Unterstützung für prozessorientierte Anwendungen. Wie diese Arbeit gezeigt hat, sollten jedoch in vielen Anwendungs-Bereichen (z.B. Krankenhaus) unternehmenskritische Geschäftsprozesse noch nicht mit MQSeries WF oder Staffware durchgeführt werden. Der Grund dafür ist die ungenügende Sicherstellung der Korrektheit und Konsistenz der Prozessmodelle und -instanzen (siehe Kapitel 4) sowie die fehlende bzw. mangelhaften Funktionalitäten in vielen wichtigen Bereichen (siehe Kapitel 6).

Die **WFMC** hat einige fehlende Funktionalitäten und Eigenschaften (z.B. Ad-Hoc-Änderungen, Archivierung, Backup, Prüfbarkeit auf Vollständigkeit der Prozessmodelle, etc) bereits identifiziert [WFMC98b]. Da dies allerdings schon vor mehreren Jahren[1] passierte, ist fraglich, wann bzw. ob die WFMC entsprechende Standard-Dokumente veröffentlicht werden.
Aktuell existiert nur ein Entwurf eines Standard-Dokumentes. Dieser spezifiziert eine weitere WPDL [WFMC01], die `XML Process Definition Language`.

Während der Ausarbeitung dieser Arbeit wurde von IBM eine neue Version von **MQSeries WF** veröffentlicht. Die nun aktuelle Version ist MQSeries WF Version 3.3 [IBM01d]. Sie verfügt u.a. über folgende neue Features:

- Unterstützung von Oracle 8
- Dynamische Subprozesseinbindung
- Erweiterte Größe der Daten-Container

Die von **Staffware** besprochene Version 8.1 ist aktuell. Es ist bisher kein Update angekündigt. Staffware bietet inzwischen aber ein eigenes Analyse-Tool an. Der `Staffware Process Monitor` [SW01f] wurde zusammen mit der IDS Scheer auf Basis des *Process Performance Monitor* entwickelt. Die Funktionalität dieses Produktes beinhaltet u.a.:

- Erzeugen von Prozessmodellen aus Audit-Daten
- Performance-Tests
- Dokumentation von Änderungen (*Modification histories*)

Im Prinzip ist die Entwicklung von Production WFMS, wie es MQSeries WF und Staffware sind, ein Schritt in die richtige Richtung. Durch die prozessorientierte Vorgehensweise, die durch eine konsequente Trennung von Ablauflogik und Anwendungsprogrammen erzielt wird, entstehen die bereits in Kapiel 1 vorgestellten Vorteile und Möglichkeiten.

[1] Die Vorschläge wurden 1998 veröffentlicht.

Wie in dieser Arbeit deutlich wurde, ist MQSeries WF in einigen Bereichen (z.B. Skalierbarkeit, Integrierbarkeit) Staffware durchaus überlegen. Staffware dagegen verfügt beispielsweise über Funktionalitäten (z.B. Änderungen an laufenden Prozessinstanzen, Eingebauter Formulareditor), die von MQSeries WF nicht angeboten werden.

Beide WFMS weisen im Prinzip die selben Schwachstellen auf. Sie nutzen das große Potential im Hinblick auf systemseitige Selbstüberwachung und Fehlerbehandlung zu wenig. Ebenfalls findet keine ausreichende Überwachung während der Prozessausführung statt, welche mittels Monitoring-Tools dem Bearbeiter wichtige Informationen bieten kann.
In Kapitel 4 wurde deutlich, wie einfach schwerwiegende Fehler in der Laufzeitumgebung beider WFMS entstehen können. In der Praxis sollen deshalb unternehmenskritische Geschäftsprozesse nicht mit ihnen durchgeführt werden.
Ein weiteres großes Manko der vorgestellten WFMS ist ihre mangelnde Flexibilität. Abweichungen vom vormodellierten Workflow, wie sie in der Praxis situationsbedingt immer wieder notwendig werden, sind nicht möglich. Dadurch bleibt das Einsatzspektrum dieser WFMS auf einige wenige starre Routineabläufe beschränkt.
Auch die mangelnde Funktionalität der WFMS (z.B. keine ausreichende Unterstützung von Zeitbeschränkungen, kein semantisches Rollback) läßt ihren breiten Einsatz nicht zu. Letztlich werden von ihnen derzeit Wartungsaspekte sträflich vernachlässigt (z.B. keine adäquate Unterstützung von WF-Schemaevolution).

Wie schnell diese Mängel behoben werden, ist abhängig von den Anstrengungen der Hersteller. Seitens der WFMC stagniert seit längerem die Entwicklung weiterer Standard-Dokumente. Momentan existieren keine eindeutigen Hinweise in der zugänglichen Literatur, wie schnell und in wie weit die Entwicklung von Production WFMS vorangetrieben wird.

# Anhang A

# Installationsvoraussetzungen

| Produkt | **MQSeries Workflow** | **Staffware 2000** |
|---|---|---|
| Version | 3.2 | 8.1 |
| Hersteller | IBM | Staffware plc. |
| Verwendetes Betriebssystem | Microsoft Windows NT 4.0 | Microsoft Windows NT 4.0 |
| gefordertes Service Pack (SP) | SP3 | SP4 |
| Zusätzliche Komponenten | WebClient | WebClient |
| **Softwarevoraussetzungen** | | |
| Obligate Software | MQSeries | RPC over TCP/IP |
| | DB2 | NobleNet Portmapper for TCP |
| Optionale Software | Microsoft Jetengine (statt DB2 für die Buildtime-Komponente) | |
| Sonstige Einschränkungen | | Es wird nur das NTFS-Dateisystem unterstützt. |
| | | Der Computer muß eine IP-Adresse besitzen. |
| | | Die Buildtime-Komponente kann nur auf einen Computer mit installiertem Client instaliert werden. |
| **Hardwarevoraussetzungen (Server)** | | |
| Prozessorgeschwindigkeit | 200 Mhz | keine Angaben |
| Hauptspeicher | 128 MB | keine Angaben |
| Festplattenkapazität | 2 GB | 2 GB |
| **Hardwarevoraussetzungen (Client)** | | |
| Prozessorgeschwindigkeit | 200 Mhz | keine Angaben |
| Hauptspeicher | 64 MB | keine Angaben |
| Festplattenkapazität | 60 MB | 5 MB |

| **Hardwarevoraussetzungen (Buildtime)** | | |
|---|---|---|
| Prozessorgeschwindigkeit | 200 Mhz | keine Angaben |
| Hauptspeicher | 95 MB | keine Angaben |
| Festplattenkapazität | 1 GB (incl. DB2) | 1,5 MB |
| **Voraussetzungen (WebClient)** | | |
| Betriebssytem | Alle Betriebssysteme mit Unterstützung für den MQSeries WF Java Agent | Microsoft Windows NT 4.0 |
| Web-Server | Webserver mit Servlet-Unterstützung (z.B. IBM WebSphere Application Server) | Microsoft Internet Information Server (IIS) Version 4.0 |
| Web-Browser | Microsoft Internet Explorer Version 4.0 oder Netscape Version 4.0 | Microsoft Internet Explorer Version 4.0 |
| Festplattenkapazität | keine Angaben | 2,5MB |
| Sonstiges | Java Developer Kid (JDK) | Staffware Enterprise Objects COM Client Version 1.1 |

Tabelle A.1: Die verwendeten WFMS

# Anhang B

# Dokumentation der Systeme

Mit der Software von MQSeries WF bzw. Staffware wurde folgende Dokumentation geliefert.

| **MQSeries Workflow** | **Staffware 2000** |
|---|---|
| Installation | User Documentation Library - Installation Guide |
| Konzepte und Architektur | New Features Guide (Staffware V8.1) |
| Einführung in die Runtime | Windows Client Users Guide |
| Getting Started with Runtime | Web Client Users Guide (for NT and Windows) |
| Administration Guide | Web Client Users Guide (for UNIX) |
| Programming Guide | SWIP Monitor Users Guide |
| | Administration Managers Guide |
| | Server Administrators Guide |
| | NT SQL Server Administrators Supplement |
| | Oracle Server Administrators Supplement |
| | Staffware for X.500 Administrators Supplement (Windows NT) |
| | GWD Procedure Definers Guide |
| | Expressions and Functions Reference Guide |
| | Forms Converter Users Guide |
| | Integrators Guide |
| | Staffware for Microsoft Exchange Integrators Guide |
| | Staffware for VBA Integrators Guide |
| | Staffware Enterprise Adapters for eLink Integrators Guide |
| | Web Client Technical Guide |
| | UNIX Web Client Technical Guide |
| | Staffware mDeveloper Toolkit Integrators Guide |
| | Staffware mDeveloper Toolkit Nokia Integrators Supplement |

Tabelle B.1: Die mitgelieferte Dokumentation der WFMS

# Anhang C

# WPDL

Die WPDL ist eine Sprache zur Beschreibung von Prozessmodellen mit ASCII-Zeichenfolgen. Dabei werden Schlüsselwörter zur Identifizierung von z.B. Objekten, Attributen oder Beschreibungen verwendet. Die WPDL der WFMC entspricht der EBNF (*Extended Backus Naur Form*) [WFMC99].

## C.1 WFMC

In [WFMC99a] ist die beispielhafte Notation eines Geschäftsmodells mittels WPDL dargestellt. Die WFMC unterteilt eine WPDL-Datei inhaltlich in zwei Bereiche:

- Organisatorischer Bereich (z.B. Personendaten, Anwendungen und Prozessrelevante Daten)
- Prozess-Bereich (z.B. Aktivitäten, Schaltbedingungen und Daten)

Im folgenden Beispiel werden beide Bereiche (gekürzt) in einer WPDL-Datei abgebildet. Die kompletet WPDL-Datei kann in [WFMC99a] eingesehen werden.
Eine WPDL-Datei kann mehrere Prozessmodelle beinhalten.

```
MODEL 'Representative_Business_Example'
  WPDL_VERSION "7.0 Beta"
  VENDOR "Vendor:Product:Release"
  CREATED 1998-07-15
  NAME "Representative Business Example"
  DESCRIPTION "WPDL-Notation of Representative Business Example"
  AUTHOR "WG/1B"
  STATUS UNDER_REVISION
  EXTENDED_ATTRIBUTE 'VENDOR_A_SPECIFIC' STRING
  "This is vendor A's attribute!"
```

**Definition der am Prozess beteiligten Personen**

```
PARTICIPANT 'France_Baroque'
    NAME "France Baroque"
    DESCRIPTION "President's Secretary"
    TYPE HUMAN
  END_PARTICIPANT
   ...
```

**Definition der am Prozess beteiligten Anwendungsprogramme**

```
  APPLICATION 'scan_document'
    NAME "scan document"
    TOOLNAME "winscan.exe"
    OUT_PARAMETERS 'scanned_document'
  END_APPLICATION
  ...
```

**Definition der am Prozess beteiligten (Daten-)Strukturen.**

```
  DATA 'document_type'
    TYPE STRING
    NAME "document type"
    DEFAULT_VALUE "Sales Order"
  END_DATA
  ...
```

**Definition des Prozessmodells**

```
WORKFLOW 'In_the_Mail_Room' CREATED 1998-07-15 NAME "In the
MailRoom"
```

**Definition der am Prozess beteiligten Aktivitäten**

```
  ACTIVITY 'MailRoom'
    NAME "Mail Room"
    IMPLEMENTATION APPLICATIONS
                     TOOL_LIST
                       'scan_document'
                       'identify_document'
                       'send_document'
                       DESCRIPTION "sequential execution"
                     END_TOOL_LIST
    PERFORMER 'Mail_Room_Clerk'
    SPLIT      AND
  END_ACTIVITY
  ...
```

**Definition der am Prozess beteiligten Transitionen**

```
TRANSITION 't_1'
    FROM 'MailRoom'
    TO 'Sales_Order_Handling'
    CONDITION 'document_type'="Sales Order"
  END_TRANSITION
```

**Ende des Prozessmodells**

```
END_WORKFLOW END_MODEL
```

## C.2 MQSeries WF - FDL

In diesem Abschnitt wird ein mit MQSeries WF V3.2.1. ausgeliefertes Prozessmodell (*fmccred.fdl*) mit erklärender Dokumentation gekürzt dargestellt. Das zugehörige Prozessmodell ist in Abbildung C.1 abgebildet. Weitergehende Erklärungen zu FDL-Dateien können [IBM99c] oder [JaBöSc97]

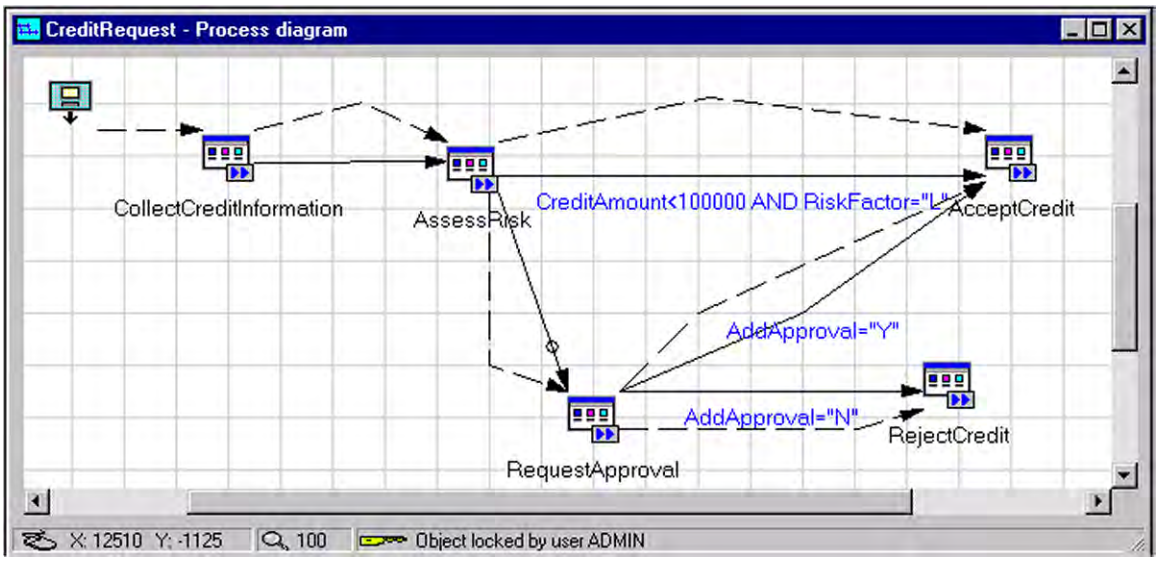



Abbildung C.1: Das Prozessmodelldiagramm (fmccred.fdl)

entnommen werden.
FDL-Dateien können wie von der WFMC gefordert mehrere Prozessmodelle beinhalten.

**fmccred.fdl**

```
/*
 *  FDL definitions generated by MQSeries Workflow at 1998-11-06, 19:11:32.
 */
CODEPAGE 437 FM_RELEASE V3R1
```

**Definition der am Prozess beteiligten (Daten-)Strukturen.**

```
STRUCTURE 'CreditInfo'
  'CreditRequestor': 'PersonInfo';
  'Address': STRING;
  'RiskFactor': STRING;
  'AddApproval': STRING;
  'CreditAmount': LONG;
END 'CreditInfo' ...
```

**Definition der am Prozess beteiligten (Anwendungs-)Programme.**

```
PROGRAM 'NAcceptCredit' ( 'CreditInfo', 'Default Data Structure' )
  NO STRUCTURES_FROM_ACTIVITY
  NOT UNATTENDED
  EXECUTION_MODE NORMAL
  INPUT_CONTAINER_ACCESS
  OUTPUT_CONTAINER_ACCESS
  WINNT EXE PATH_AND_FILENAME 'FMCN6BNA.EXE'
    STYLE VISIBLE
  WIN95 EXE PATH_AND_FILENAME 'FMCN6BNA.EXE'
    STYLE VISIBLE
END 'NAcceptCredit' ...
```

**Definition des Prozesses (incl. der beteiligten Aktivitäten).**

```
PROCESS 'CreditRequest' ( 'PersonInfo', 'Default Data Structure' )
  DESCRIPTION 'Credit request'
  PROMPT_AT_PROCESS_START
  NO AUTONOMY
  WINDOW ZOOM_FACTOR 100
  WINDOW PAPERSIZE WIDTH 2970 HEIGHT 2100
  WINDOW SHOW ALL CONNECTORS
  WINDOW SHOW TRANSITION_CONDITIONS
  SOURCE 1 XPOS -918 YPOS 433
  PROGRAM_ACTIVITY 'AcceptCredit' ( 'CreditInfo', 'Default Data Structure' )
    DESCRIPTION 'Credit request for %CreditRequestor.FirstName%
        %CreditRequestor.LastName%'
    START MANUAL WHEN AT_LEAST_ONE CONNECTOR TRUE
    EXIT AUTOMATIC
    LAYOUT XPOS 608 YPOS 166
    NAME_POSITION XPOS 608 YPOS 91
    DONE_BY STARTER_OF_ACTIVITY 'CollectCreditInformation'
    PROGRAM 'NAcceptCredit'
  END 'AcceptCredit'
 ...
```

**Definition des Kontrollfluss**

```
  CONTROL
    FROM 'AssessRisk' TO 'AcceptCredit'
    WHEN 'CreditAmount<100000 AND RiskFactor="L"'
      XPOS 188 YPOS 146
  ...
```

**Definition des Datenfluss**

```
  DATA
    FROM SOURCE 1 TO 'CollectCreditInformation'
    MAP '_STRUCT' TO '_STRUCT'
  ...
END 'CreditRequest'
```

## C.3 Staffware - XFR

In diesem Abschnitt wird ein mit Staffware 2000 ausgeliefertes Prozessmodell (*loan.xfr*) mit erklärender Dokumentation dargestellt. Weitergehende Erklärungen werden von Staffware in der mitgelieferten Dokumentation nicht bereitgestellt. Das Prozessmodell ist in Abbildung C.2 abgebildet.

**loan.xfr**

```
SW_ ^PROCEDURES LOAN,48!0,Loan Process
^PROCEDURE^LOAN,48,APPLY,Loan Process
^DIRACTIVE^$PANIC1,0,1,0,4096,23,Exception notification
^ADDRESSEES SW_OWNER ...
```

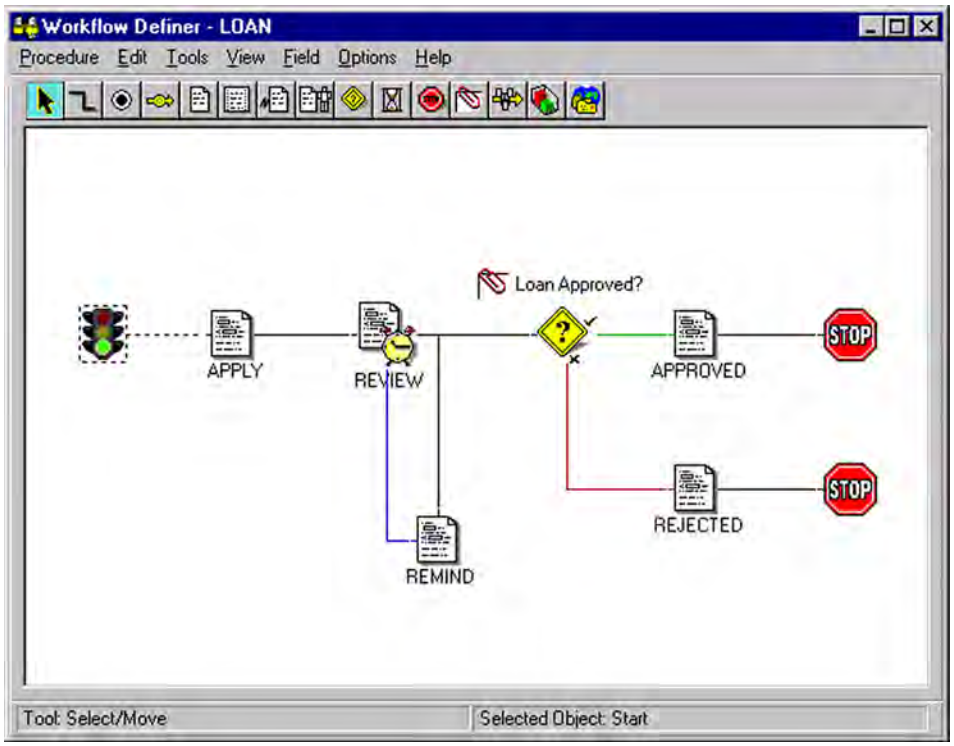



Abbildung C.2: Das Prozessmodelldiagramm (loan.xfr)

### Bedingung

```
PREMIND, ^ACTIONS DREMIND, ILoanOK = "yes" PAPPROVED, E PREJECTED,
F
```

### Formularbeschreibungen

```
... ^ACTIONS P ^PROJECTS ^DOCUMENT +0,0,LOAN APPROVED +0,0,
+0,0,Street: STREET +8,-1,^STREET/D,~~ +0,0,  City: CITY
+8,-1,^CITY/D,~~ +0,0, State: STATE  Zip: ZIP +8,-1,^STATE/D,~~
+20,-1,^ZIP/D,~~ +0,0, +0,0,FIRST           MIDDLE LAST
, +0,-1,^FIRST/D,~~ +16,-1,^MIDDLE/D,~~ +32,-1,^LAST/D,~~ +0,0,
+0,0,We are happy to inform you that your loan request has been
approved +0,0,under the following terms: +0,0, +0,0, Amount:
$AMOUNT +20,-1,^AMOUNT/D,~~
+0,0,             Term:  TERM (months)   Interest Rate:  RATE%
+20,-1,^TERM/D,~~ +52,-1,^RATE/D,~~ +0,0, +0,0,We will contact
you when the loan agreement has been prepared and +0,0,is ready
for your signature. +0,0, +0,0,Sincerely, +0,0, +0,0,Loan Officer
... ^FIELD^ZIP,65,5,0,366,- ^GWD 3,7 37 STEP 1 0 224 128 REVIEW
        IFELSE 2 7 336 128 LoanOK = "yes"
            STEP 3 7 416 128 APPROVED
            END
            STEP 4 3 416 224 REJECTED
            END
        END
        STEP 5 5 256 256 REMIND
        END
        STEP 5 3 256 256 REMIND
```

```
        END
...
```

### Definition von Bemerkungen

```
    ANNOTATIONS
 Loan Approved?

    END
END START 10 0 48 128
        STEP 6 7 128 128 APPLY
        END
END
```

# Anhang D

# Datenbank-Tabellen-Struktur (Staffware)

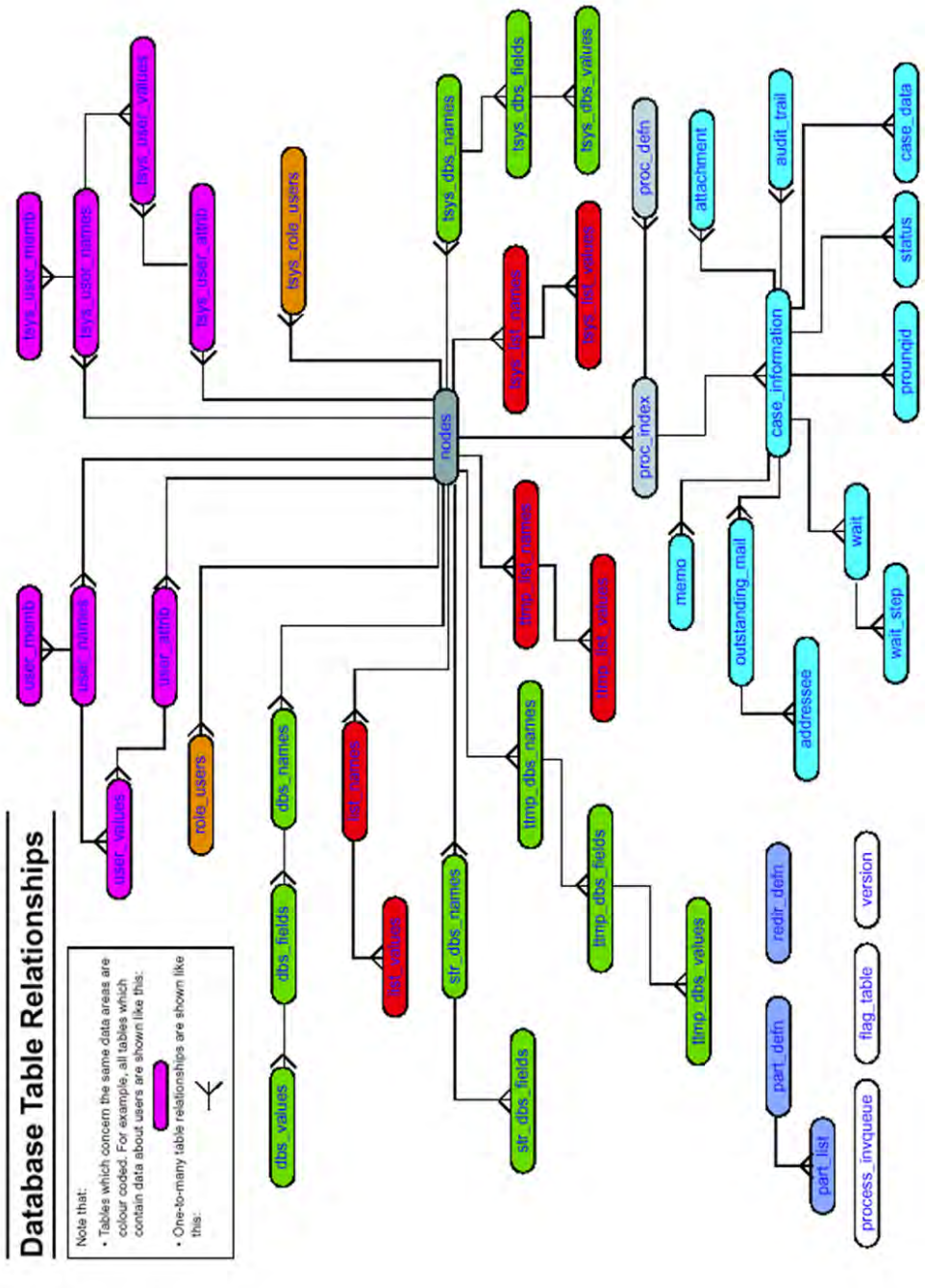



Abbildung D.1: Datenbank-Tabellen-Struktur in Oracle 8i (aus [SW00f])

# Anhang E

# Glossar

| | |
|---|---|
| API | Application Programming Interface |
| BPR | Business Process Reengineering |
| CORBA | Common Object Request Broker Architecture |
| DBMS | Database Management System |
| FDL | Flowmark Definition Language |
| GFD | Graphical Form Designer |
| GWD | Graphical Workflow Designer |
| GUI | Graphical User Interface |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext Transport Protocol |
| IF | Interface |
| IIOP | Internet Inter-ORB Protocol |
| IIS | Internet Information Server |
| IDL | Interface Definition Language |
| JRMP | Java Remote Method Protocol |
| MIME | Multipurpose Internet Mail Extension |
| OMA | Object Management Architecture |
| OMG | Object Management Group |
| ORB | Object Request Broker |
| PES | Program Execution Server |
| PPM | Process Performance Monitor |
| RMI | Remote Method Invocation |
| RPC | Remote Procedure Call |
| SDK | Staffware Developer Kid |
| SEO | Staffware Enterprise Object |
| SWAP | Simple Workflow Access Protocol |
| SWIP | Staffware work in progress |
| UPES | User-defined Program Execution Server |
| VB(A) | Visual Basic (for Application) |
| WAPI | Workflow Application Programming Inteface |
| WF | Workflow |
| WFMC | Workflow Management Coalition |
| WFMS | Workflow Management System |
| WPDL | Workflow Process Definition Language |
| XPDL | XML Process Definition Language |
| XML | Extensible Markup Language |

Tabelle E.1: Abkürzungsverzeichnis

# Literaturverzeichnis


[COR01] **CORBA**: *CORBA FAQ.htm*, http://www.corba.com, Mai 2001

[COR01a] **CORBA**: *ORB Basics.htm*, http://www.corba.com, Mai 2001

[Dad96] **P. Dadam**, *Verteilte Datenbanken und Client/Server-Systeme*, Grundlagen Konzepte Realisierungsformen, Springer Verlag, 1996

[Dad99] **P. Dadam**: *Worklfow-Management-Systeme*, Vorlesungsskript WS99/00, Universität Ulm

[DaReKu00] **P. Dadam, M. Reichert, K. Kuhn**, *Clinical Workflows - The Killer Application for Process-oriented Information Systems?*, Proc. 4th Int´l Conf. on Business Information Systems, April 2000

[Fis01] **L. Fischer**, *Workflow handbook 2001*, Future Strategies Inc., Book Division, 2001

[IBM99] **IBM**: *IBM MQSeries Workflow Administration Guide*, Juni 1999

[IBM99a] **IBM**: *IBM MQSeries Workflow Installation*, September 1999

[IBM99b] **IBM**: *IBM MQSeries Workflow Konzepte und Architektur*, September 1999

[IBM99c] **IBM**: *IBM MQSeries Workflow Getting Started with Buildtime*, September 1999

[IBM99d] **IBM**: *IBM MQSeries Workflow Einführung in die Runtime*, September 1999

[IBM99e] **IBM**: *IBM MQSeries Programming Guide*, September 1999

[IBM99f] **IBM**: *IBM MQSeries Wokflow Installation Guide*, September 1999

[IBM00] **IBM**: *IBM MQSeries Workflow Staff assignment Techniques*, Juli 2000

[IBM01] **IBM**: *WA81: MQSeries Workflow - Web Client*, : http://www-4.ibm.com/ software/ts/mqseries/txppacs/wa81.html, Mai 2001

[IBM01a] **IBM**: *MQSeries Workflow - Java Agent locator policy RMI-IIOP*, http://www6.software.ibm.com/MQSWF/RMI_IIOP_inst.html, Juni 2001

[IBM01b] **IBM**: *IIOP-Usage*, http://www-4.ibm.com/software/webservers/dgw/pub/overview.html, Juni 2001

[IBM01c] **IBM**: *Model-driven Process Automation*, Lotus Domino Workflow and IBM MQSeries Workflow, Juli 2001

[IBM01d] **IBM**: *MQSeries Workflow 3.3*, Business Process Management with IBM MQSeries Workflow, Juli 2001

[JaBöSc97] **S. Jablonski, M. Böhm, W. Schulze**, *Workflow-Management Entwicklung von Anwendungen und Systemen*, dpunkt Verlag, 1997

[Kon96] **I. Konyen**, *Organisations- und Ablaufstrukturen im Krankenhaus - Anforderungen, Werkzeuge und deren Anwendung*, Diplomarbeit, Universität Ulm, 1996

[LeRo97] **IBM System Journal**, *Vol 36, Application Development*, Kapitel: Workflow based Applications (Authoren: F. Leymann, D. Roller), 1997

[LeRo00] **F. Leymann, D. Roller**: *Production Workflow Concepts and Techniques*, Prentice Hall PTR, 2000

[Lot01] **Lotus**: *Workflow products*, www.lotus.com/home.nsf/welcome/domworkflow, 2001

[MüSt99] **B. F. Müller, P. Stolp**, *Workflow-Management in der industriellen Praxis*, Springer Verlag, 1999

[OMG98] **OMG**: *Workflow Management Facility*, Document Number: bom/98-06-07, Juli 1998

[OMG00] **OMG**: *Workflow Management Facility Specification*, 00-05-02.pdf, April 2000

[OMG01] **OMG**: *The Common Object Request Broker: Architecture and Specification*, 01-02-33.pdf, Februar 2001

[OMG01a] **OMG**: http://www.omg.org, Juni 2001

[Re00] **M. Reichert**, *Dynamische Ablaufänderungen in Workflow-Management-Systemen*, Dissertation, Universität Ulm, Mai 2000

[ReDa98] **M. Reichert, P. Dadam**: *ADEPTflex - Supporting Dynamic Changes of Workflows Wihtout Losing Control*, Journal of Intelligent Information Systems, Special Issue on Workflow Management Systems, Vol. 10 No. 2, März 1998

[Ro96] **D. Roller**, *Verifikation von Workflows in IBM Flowmark.* In: G. Vossen; J. Becker: Geschäftsprozessmodellierung und Workflow-Management. Thomson Publ., S.353-368, 1996

[SUN01d] **SUN**: *CORBA-IIOP-Programming*, http://java.sun.com/products/jdk/1.2/docs/guide/idl/index.html, Juni 2001

[SW98] **Staffware**: *Staffware for Lotus Notes Integrators Guide (for Windows NT)*, Mai 1998

[SW99] **Staffware**: *Staffware for VBA Integrators Guide*, März 1999

[SW99a] **Staffware**: *Staffware for Microsoft Exchange Integrators Guide*, März 1999

[SW99b] **Staffware**: *Staffware for X.500 Administrator´s Supplement*, März 1999

[SW99c] **Staffware**: *SWIP Monitor Users Guide*, Dezember 1999

[SW99d] **Staffware**: *White Paper Staffware Enterprise Objects "The next step in enterprise workflow"*, Version1, 1999

[SW99e] **Staffware**: *White Paper Staffware 2000*, Version1, 1999

[SW00] **Staffware**: *Staffware Windows Client Users Guide*, Oktober 2000

[SW00a] **Staffware**: *Staffware GWD Procedure Definer´s Guide*, Oktober 2000

[SW00b] **Staffware**: *Staffware Integrator's Guide*, Oktober 2000

[sw00c] **Staffware**: *SEO COM Object Model*, Oktober 2000

[SW00d] **Staffware**: *Staffware Server Administrators Guide*, Oktober 2000

[SW00e] **Staffware**: *Staffware Expressions and Functions Reference Manual*, Oktober 2000

[SW00f] **Staffware**: *Staffware Oracle Server Administrators Supplement*, Oktober 2000

[SW00g] **Staffware**: *Staffware NT Classic Server Version 8 Installation Guide*, Dezember 2000

[SW00h] **Staffware**: *Staffware V8.1 New Features Guide*, Dezember 2000

[SW00i] **Staffware**: *Staffware Web Client Technical Guide*, Dezember 2000

[SW01] **Staffware**: *Server Platforms*, www.staffware.de/products/sw2000.asp?sub=techspec, April 2001

[SW01a] **Staffware**: *Staffware Training Vorgangsentwicklung E1*, April 2001

[SW01b] **Staffware**: *Technischer Foliensatz für Uni Ulm*, Juni 2001

[SW01c] **Staffware**: *Staffware 2000 for Windows NT*, Online-Hilfe, Juni 2001

[SW01d] **Staffware**: *README for Staffware NT Web Client Version 8.1(0.0)*, Technische Onlinedokumentation unter www.staffware.de, Juni 2001

[SW01e] **Staffware**: *The SWUTIL Utility*, http://tech.staffware.com/Support_Notes/DIN000020.html, Juni 2001

[SW01f] **Staffware**: *Staffware launches Staffware Process Monitor for analyzing and optimizing Business Processes*, http://www.staffware.de/latestnews/prdetails.asp?relid=184 , August 2001

[SWAP01] **SWAP Working Group**: *What is Workflow*, http://www.ics.uci.edu/ ietfswap/, Juni 2001

[SWAP01a] **SWAP Working Group**: *What is Workflow*, http://www.ics.uci.edu/~ietfswap/ charter.html, Juni 2001

[Tan98] **A. S. Tanenbaum**: *Verteilte Betriebssysteme*, Prentice Hall, 1998

[Web98] **M. Weber**: *Verteilte Systeme*, Spektrum Akademischer Verlag, 1998

[WFMC95] **Workflow Management Coalition**: *The Workflow Reference Model*, Document Number WFMC TC00-1003, Januar 1995

[WFMC96] **Workflow Management Coalition**: *Workflow Standard - Interoperability Abstract Specification*, Document Number WFMC-TC-1012, October 1996

[WFMC97] **Workflow Management Coalition**: *Workflow Client Application (Interface 2) Application Programming Interface (WAPI) Naming Conventions*, Document Number WFMC-TC-1013, November 1997

[WFMC98] **Workflow Management Coalition**: *Reference Model Workflow Interoperabity, the Key to E-Commerce and to Process Scalability*, http://www.aiim.org/wfmc/if4article.pdf, Oktober 1998

[WFMC98a] **Workflow Management Coalition**: *Workflow Management Coalition Audit Data Specification*, Document Number WFMC-TC-1015, September 1998

[WFMC98b] **Workflow Management Coalition**: *Workflow Client Application Programming Interface (Interface 2&3) Specification*, Document Number WFMC-TC-1009, Juli 1998

[WFMC99] **Workflow Management Coalition**: *Interface1: Process Definition Interchange Process Model*, Document Number WfMC TC-1016-P, Oktober 1999

[WFMC99a] **Workflow Management Coalition**: *Interface1: Process Definition Interchange Q&A and Examples*, Document Number WFMC TC-1016-X, Januar 1999

[WFMC99b] **Workflow Management Coalition**: *Terminology & Glossary*, Document Number WFMC-TC-1011, Februar 1999

[WFMC99c] **Workflow Management Coalition**: *Workflow Standard - Interoperability Abstract Specification*, Document Number WFMC-TC-1012, October 1996

[WFMC00] **Workflow Management Coalition**: *Workflow Standard- Interoperability Internet e-mail MIME Binding* , Document Number WFMC-TC-1018 7-January-2000

[WFMC00a] **Workflow Management Coalition**: *Workflow Standard- Interoperability Wf-XML Binding* , Document Number WFMC-TC-1023, Mai 2000

[WFMC00b] **Workflow Management Coalition**: *Workflow Standard - Interoperability XML-HTTP binding*, Februar 2000

[WFMC01] **Workflow Management Coalition**: *Workflow Process Definition Interface – XML Process Definition Language*, Document Number WFMC-TC-1025 (Alpha Status), Mai 2001

[WFMC01a] **Workflow Management Coalition**: *Workflow Standards and Associated Documents*, www.wfmc.org/standards/docs/Stds_diagram.pdf, Juni 2001

[WFMC01b] **Workflow Management Coalition**: *WfMC - Conformance to Interface Standards*, http://www.wfmc.org/standards/conformance.htm, Juni 2001

[WFMC01c] **Workflow Management Coalition**: *Published Documents*, http://www.wfmc.org/standards/docs.htm, Juni 2001

[WFMC01d] **Workflow Management Coalition**: *Conformance Statement*, http://www.wfmc.org/standards/conformance.htm, Juni 2001

[Zah00] **R. Zahavi**, *Enterprise Application Integration with CORBA*, OMG Press, 2000

# Danksagung

Diese Arbeit wurde im Zeitraum vom 14. Februar 2001 bis zum 14. August 2001 in der Abteilung Datenbanken und Informationssysteme (DBIS) der Universität Ulm durchgeführt.

An dieser Stelle möchte ich mich bedanken bei:

- Herrn Prof. Dr. P. Dadam für die interessante Themenstellung
- Herrn Dr. M. Reichert für die intensive Betreuung und für die ständige Gesprächsbereitschaft in allen fachlichen Belangen
- Herrn Dipl. Inf. J. Enderle für die ständige Gesprächsbereitschaft in allen fachlichen Belangen

Ein besonderer Dank gilt meinen Eltern, die mir dieses Studium ermöglicht haben.

Name: Udo Martschat Matr.Nr. 363 481

**Erklärung**

Ich erkläre, daß ich die Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

(Unterschrift)