Interoperabilitätsstandards für Workflow-Management-Systeme zur unternehmensübergreifenden Prozesskopplung

Diplomarbeit an der Universität Ulm

Fakultät für Informatik

Abteilung Datenbanken und Informationssysteme

vorgelegt von

Christian Ergenschäfter

1. Gutachter: Prof. Dr. Peter Dadam

2. Gutachter: Dr. Manfred Reichert

2001

Inhaltsverzeichnis

1 Einleitung 5
1.1 Motivation 5
1.2 Aufgabenstellung 6
1.3 Aufbau der Arbeit 6
2 Workflow-Management – Überblick 8
2.1 Begriffe 8
2.2 Business Process Reengineering 8
2.3 Workflow-Management-Systeme 9
2.3.1 Grundlegende Funktionen eines WFMS 9
2.3.2 Struktur eines WFMS 10
2.4 Kategorisierung von rechnergestützten Prozessen 12
2.5 Workflow-Standardisierungs-Gruppen 13
2.5.1 Workflow Management Coalition 13
2.5.1.1 Workflow Reference Model 13
2.5.1.2 Import/Export von Prozessmodellen (Interface 1) 16
2.5.1.3 Client- und Anwendungsprogrammierung (Interface 2 & 3) 18
2.5.1.4 Interoperabilität zwischen WFMS (Interface 4) 21
2.5.1.5 Administrations- und Monitoringwerkzeuge (Interface 5) 21
2.5.2 Object Management Group (OMG) 22
2.5.2.1 Object Management Architecture (OMA) 23
2.5.2.2 Workflow Facility 24
2.5.3 Weitere Arbeitsgruppen 25
2.5.3.1 Process Specification Language (PSL) 26
2.5.3.2 ODMA Extension for Workflow 26
2.5.3.3 Simple Workflow Access Protocol (SWAP) 26
2.6 Zusammenfassung 27

3 Unternehmensübergreifende Prozesskopplung 29
3.1 Ausgangssituation 29
3.2 Fallbeispiele 30
3.2.1 Verfügbarkeitsprüfung durch Kunden 30
3.2.2 Kundenbestellung 31
3.2.3 Statusabfrage einer Bestellung 34
3.2.4 Stornierung einer Bestellung 35
3.3 Zusammenfassung 36
4 Die WfMC – Interoperability Abstract Specification 37
4.1 Definition Workflow-Interoperabilität 37
4.2 Arten der Kommunikation 37
4.3 Interoperabilitätsgrade 38
4.4 Interoperabilitätsmodelle 40
4.4.1 Verkettete Prozesse 40
4.4.2 Verschachtelte Prozesse 40
4.4.3 Parallel synchronisierte Prozesse 41
4.5 Prozessdatenaustausch 41
4.6 Operationen der abstrakten Spezifikation 42
4.7 Zusammenfassung 44
5 Interoperabilitätsstandards 46
5.1 Vor- und Nachteile von Interoperabilitätsstandards 46
5.1.1 Vorteile 47
5.1.2 Nachteile 48
5.2 Entwicklungsgeschichte der Interoperabilitätsstandards 48
5.3 WfMC E-Mail Binding 49
5.3.1 Beispiel einer Konversation 49
5.3.2 Aufbau einer MIME-Nachricht 50
5.3.3 Sicherheitsaspekte beim Nachrichtenaustausch 53
5.4 Simple Workflow Access Protocol (SWAP) 54

5.5 WfMC WF-XML Binding ... 54
5.5.1 Grundlagen von XML ... 55
5.5.2 WF-XML ... 57
5.5.2.1 Basismodell ... 57
5.5.2.2 Aufbau einer XML-Nachricht ... 57
5.5.2.3 Unterstützte Operationen ... 58
5.5.2.4 Übertragungsprotokoll ... 62
5.6 OMG jFlow Workflow Facility ... 62
5.7 Zusammenfassung ... 63
6 Interoperabilitätsschnittstelle in einem bestehenden WFMS ... 65
6.1 IBM MQ Series Workflow ... 65
6.1.1 Software-Architektur von MQ Series Workflow ... 65
6.1.2 Interoperabilitäts-Schnittstellen in MQ Series Workflow ... 67
6.1.2.1 Request Messages ... 69
6.1.2.2 Invoke Messages ... 70
6.1.3 Bewertung ... 72
6.2 Andere Vertreter ... 73
7 Fortschrittliche Anforderungen und Aspekte ... 74
8 Diskussion und Zusammenfassung ... 78
Literaturverzeichnis ... 80
Abkürzungsverzeichnis ... 85
Abbildungsverzeichnis ... 87
Tabellenverzeichnis ... 89
Anhang ... 90

1 Einleitung

Die heutige Marktsituation der Unternehmen ist durch zunehmende Verschärfung des Wettbewerbs und steigenden Konkurrenzdruck geprägt. Als Schlagwörter seien hier "Just-in-Time", "Time-to-Customer" und "Time-to-Market" genannt. Die daraus resultierenden Herausforderungen wie stärkere Kundenorientierung, gehobene Produktqualität, rasches Reagieren auf Marktveränderungen und kürzere Lieferzeiten, können von vielen Unternehmen nicht erfüllt werden. Die Ursache für das Scheitern von Firmen ist vielfach auf ihre starren, hierarchischen und funktionsorientierten Strukturen zurückzuführen. Gefordert sind dagegen flexible und prozessorientierte Organisationen, die rasch und dynamisch auf das Marktgeschehen reagieren können.

1.1 Motivation

Für Unternehmen in nahezu allen Branchen gewinnt die flexible Gestaltung von Geschäftsprozessen weg von der funktionalen, hin zur prozessorientierten Betrachtungsweise zunehmend an Bedeutung. Das Ziel sind schlanke, kundenorientierte Prozesse besonders in den Bereichen, die für die Wettbewerbsfähigkeit ausschlaggebend sind. Um dies umzusetzen, müssen vorhandene Abläufe und Strukturen erfasst, analysiert und optimiert werden (Geschäftsprozessreengineering) [LeRo00].

In jüngster Vergangenheit sind mehrere Ansätze entstanden, um Unternehmen ein flexibleres und prozessorientiertes Handeln zu ermöglichen. Exemplarisch seien an dieser Stelle das *Supply-Chain-Management* (SCM) und das *Customer-Relationship-Management* (CRM) genannt.

Das *Supply-Chain-Management* [KnMeZe00] hat seinen Schwerpunkt in der Optimierung der Prozesse entlang der Wertschöpfungskette. Die Ziele der Einführung eines SCM-Systems sind die Verkürzung von Durchlauf- und Lieferzeiten, die Minimierung von Lagerbeständen sowie die Steigerung der Kundennähe durch Verbesserung der Lieferbereitschaft und Erhöhung der Liefertreue. Schließlich soll eine Kostenersparnis durch optimale Ressourcenplanung und höhere Planungseffizienz erreicht werden.

Beim *Customer-Relationship-Management* [Rapp00] dagegen steht der Kunde im Mittelpunkt. Wichtigste Eckpunkte sind die Kostenreduktion durch gezielte Funktionsverlagerung von Unternehmen zum Kunden und die somit vereinfachte Datenpflege. Darunter versteht man z.B., dass eine Adressänderung direkt vom Kunden Online durchgeführt werden kann, ohne Kapazitäten des Unternehmens zu binden. Weitere Ziele sind die Erhöhung des Umsatzes durch konsequente Kundengewinnung, die verbesserte Kundenbetreuung (z.B. unter Nutzung von Data-Mining, Cross-Selling und Up-Selling) sowie der Aufbau langfristiger Mechanismen zur Kundenbindung. Dies erfordert insbesondere, dass der Kunde immer auf dem aktuellsten Informationsstand ist. Ebenso

wichtig ist die Erhöhung der Wertschöpfung durch Nutzung der Kundeninformation in internen Prozessen.

SCM- und CRM-Systeme bieten in ihrer ursprünglichen Ausrichtung jedoch keine bereichs- oder unternehmensübergreifende Planung und Optimierung der Prozess- bzw. Logistikkette. All diese Ansätze sind nur auf bestimmte Bereiche eines Unternehmens ausgerichtet; keiner betrachtet die Abläufe im Unternehmen in einer ganzheitlichen Sicht. Ein neuer Ansatz ist der Einsatz von Workflow-Management-Systemen (WFMS) zur globalen Unterstützung von Arbeitsabläufen im ganzen Unternehmen [ReDa00]. WFMS zeichnen sind durch die Trennung von Ablauflogik und Anwendungscode aus. WFMS-basierte Anwendungen lassen sich somit einfacher und schneller an geänderte Geschäftsprozesse anpassen. Mittels globalen übergeordneten Arbeitsabläufen lassen sich darüber hinaus verschiedene Bereiche in einem Unternehmen schnell miteinander verbinden.

Inzwischen sind verschiedene Workflow-Management-Systeme am Markt vertreten. Deshalb ist die Zusammenarbeit prozessorientierter Anwendungen firmenintern, konzernweit und auch unternehmensübergreifend recht schwierig. Um hier zu einer Verbesserung zu gelangen, das heißt um Prozesse bereichs- oder unternehmensübergreifend unterstützen zu können, werden Standardschnittstellen zwischen WFMS benötigt, damit diese einfach und effektiv untereinander Daten (z.B. Prozessdefinitionen, Statusinformationen zu laufenden Prozessen) austauschen können.

1.2 Aufgabenstellung

In dieser Arbeit soll der aktuelle Stand der Standardisierungsbemühungen im Bereich Interoperabilität von Workflow-Management-Systemen aufgezeigt werden. Dabei soll auf folgende Fragestellungen näher eingegangen werden:

1. Welche Standards gibt es im Workflow-Bereich und von welchen Gruppen werden diese veröffentlicht?
2. Werden diese Standards den allgemeinen Anforderungen an Workflow-basierte Anwendungen gerecht und was leisten sie konkret?
3. Werden derzeitige Standards von den Herstellern von Workflow-Management-Systemen umgesetzt?
4. Welche Gesichtspunkte sind bisher im Rahmen der bereits erfolgten Standardisierungen noch nicht beachtet worden bzw. welche Anforderungen werden zukünftig noch an WFMS-Standards zu stellen sein?

1.3 Aufbau der Arbeit

Diese Arbeit gliedert sich wie folgt: In Kapitel 2 wird zuerst eine kurze Einführung in das Thema Workflow-Management gegeben. Dabei werden generelle Workflow-Konzepte und der

schematische Aufbau eines Workflow-Management-Systems skizziert. Sie sind für das weitere Verständnis der Arbeit von Nöten. Anschließend werden die beiden wichtigsten Standardisierungsgruppen im Workflow-Bereich vorgestellt. Dies sind zum einen die *Workflow-Management Coalition* (WfMC) mit ihrem *Workflow Reference Model* und dessen Schnittstellen, zum anderen die *Object Management Group* (OMG) mit ihrer *Object Management Architecture* und dem *Workflow-Facility*.

Anhand eines organisationsübergreifenden Anwendungsszenarios wird in Kapitel 3 die Notwendigkeit standardisierter Schnittstellen zur unternehmensübergreifenden Kopplung von WFMS-unterstützten Prozessen aufgezeigt.

Kapitel 4 stellt die sogenannte *Abstract Specification* der WfMC vor. Sie bildet das grundlegende Konzept aller bisher veröffentlichten Interoperabilitäts-Standards.

Die auf Grundlage der *Abstract Specification* veröffentlichten Interoperabilitäts-Standards werden in Kapitel 5 in der zeitlichen Reihenfolge ihrer Entwicklung vorgestellt. Dazu zählen z.B. das MIME-Binding, das WF-XML-Binding der WfMC und das jFlow Facility der OMG.

Inwieweit gängige Standards derzeit auch in der Praxis umgesetzt werden, wird in Kapitel 6 diskutiert. Hier wird anhand des WFMS IBM MQ Series Workflow geprüft, ob die in Kapitel 5 vorgestellten Interoperabilitäts-Standards eingesetzt werden.

In Kapitel 7 wird ein Ausblick gegeben welche fortschrittlichen Anforderungen zukünftig an Workflow-Systeme zu stellen sind und welche Erweiterungen heutiger Standards dadurch notwendig werden.

Abschließend wird in Kapitel 8 die Arbeit zusammengefasst und die wichtigsten Erkenntnisse werden diskutiert.

2 Workflow-Management – Überblick

Dieses Kapitel gibt einen kurzen Überblick über grundlegende Konzepte des Workflow-Management und stellt wichtige Standardisierungsbestrebungen in diesem Bereich vor.

2.1 Begriffe

In der Terminologie der Workflow-Management-Coalition (siehe 2.5.1) wird Workflow bzw. Workflow-Management folgendermaßen definiert [WFMC99]:

> *"The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules."*[1]

Ein Geschäftsprozess besteht aus einer Menge von Tätigkeiten, den sogenannten Aktivitäten, die ganz bzw. teilweise von verschiedenen Personen in einer definierten Reihenfolge ausgeführt werden. Es handelt sich um die logische Ablaufstruktur zur Abwicklung eines Vorgangs (Kontrollfluss) und die Koordination der dafür benötigten Ressourcen. Dabei sind verschiedene Ablaufformen möglich. Aktivitäten können sequentiell, parallel oder alternativ bearbeitet werden. Die Gesamtheit der Beschreibung der Aktivitäten wird als Prozessmodell bezeichnet. Dazu gehört neben der Kontrollflussdefinition, die Modellierung des Datenflusses zwischen den Aktivitäten, die Beschreibung der Aufbauorganisation des Unternehmens, die Benennung von Verantwortlichen (bzw. potentieller Bearbeiter) für Aktivitäten sowie die Beschreibung der informationstechnischen Hilfsmittel zur Durchführung dieser Aktivitäten.

2.2 Business Process Reengineering

Workflow-Management steht oft im Zusammenhang mit dem Schlagwort *Business Process Reengineering* [HaCh95]. Um Workflow-Management-Systeme erfolgreich einsetzen zu können, müssen zuerst alle vorhandenen Strukturen und Abläufe im Unternehmen erfasst, analysiert und optimiert werden. Diesen Vorgang nennt man auch Geschäftsprozess-Optimierung *oder Business Process Reengineering* (kurz: BPR). Hier werden die Aufbau- und Ablauforganisation des Unternehmens hinterfragt sowie Funktionen der Mitarbeiter und Beziehungen untereinander analysiert. Anschließend können diese innerhalb eines WFMS abgebildet werden.

[1] "Die Automatisierung eines Geschäftsprozesses, in Teilen oder als Ganzes, in dem Dokumente, Informationen oder Aufgaben von einem Teilnehmer zum Nächsten weitergereicht werden und dabei festgelegten Regeln genügen müssen."

2.3 Workflow-Management-Systeme

WFMS werden in der Terminologie der WfMC folgendermaßen definiert [WFMC99]:

> *"A system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications."*[2]

Derzeitige WFMS eignen sich insbesondere zur Unterstützung von stark strukturierten, repetitiven Prozessen, die immer wieder in der gleichen oder ähnlichen Form auftreten. Dabei sind mehrere Personen involviert, die einem starken Koordinierungsbedarf unterliegen.

Man spricht in diesem Zusammenhang oftmals auch von prozessorientierten Systemen und *Production Workflow* [LeRo00]. Production-WFMS [Mar01] sind Systeme, bei denen Geschäftsvorfälle bzw. Prozesse nach einem vorher definierten Modell gesteuert werden. Sie eignen sich besonders gut für stark strukturierte und arbeitsteilige Organisationen.

2.3.1 Grundlegende Funktionen eines WFMS

WFMS trennen zwischen Modellierkomponenten (*Buildtime*) und Laufzeitkomponenten (*Runtime*). Mit Hilfe der Buildtime-Komponenten werden Prozesse analysiert und modelliert. Die Runtime-Komponenten sind zuständig für die Ausführung und Kontrolle der hieraus abgeleiteten Prozess-Instanzen sowie für die Interaktion mit den Benutzern und den zur Ausführung einzelner Prozessschritte gerufenen Workflow-Anwendungen. Der schematische Aufbau der beiden Funktionsbereiche ist in Abbildung 1 skizziert.

[2] "Ein System, das mit Hilfe von Software die Ausführung von Arbeitsabläufen sowohl definiert als auch selbst vornimmt und steuert, dabei auf einer oder mehreren Maschinen läuft und in der Lage ist, die Definitionen der Geschäftsprozesse zu interpretieren, mit den Mitarbeitern zusammenzuarbeiten und dort, wo es erforderlich ist, die Nutzung der IT-Werkzeuge und –Programme vorzuschlagen."

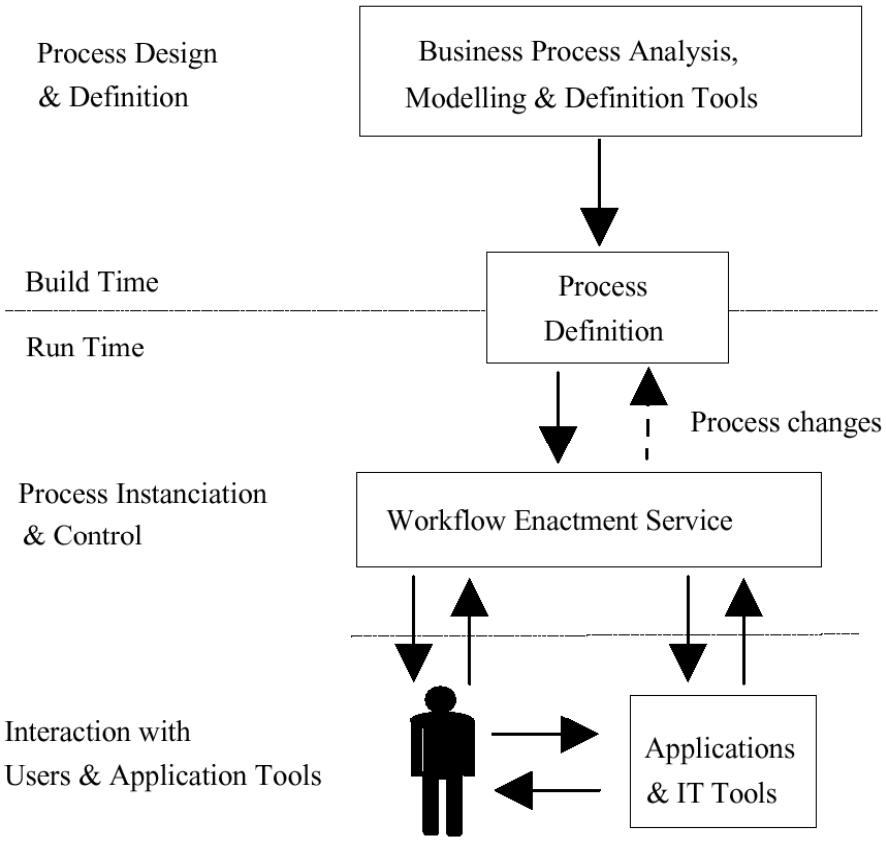


Abbildung 1: Schematischer Aufbau eines WFMS[3]

2.3.2 Struktur eines WFMS

Die Struktur eines WFMS ist nach [WFMC95] folgendermaßen definiert:

Zentrale Komponente eines WFMS ist die WF-Engine, die Komponente die Prozess-Definitionen (Prozess-Vorlagen) interpretiert und ausführt. Diese Prozess-Definitionen werden mit Modellierungs-Tools erstellt und in einer Datenbank verwaltet. Neben den Prozess-Definitionen wird auch das Organisationsmodell (mit Rollen, Stellen und Funktionsbeschreibungen) dort hinterlegt. Das System wird mittels Administrations- und Monitoringsystemen verwaltet. Dem Workflow-Nutzer stellt es die anstehenden Tätigkeiten bzw. Aufträge über Arbeitslisten zur Verfügung, die er unter Mithilfe von verknüpften Applikationen abarbeitet (siehe Abbildung 2).

[3] Aus [WFMC99], Seite 44

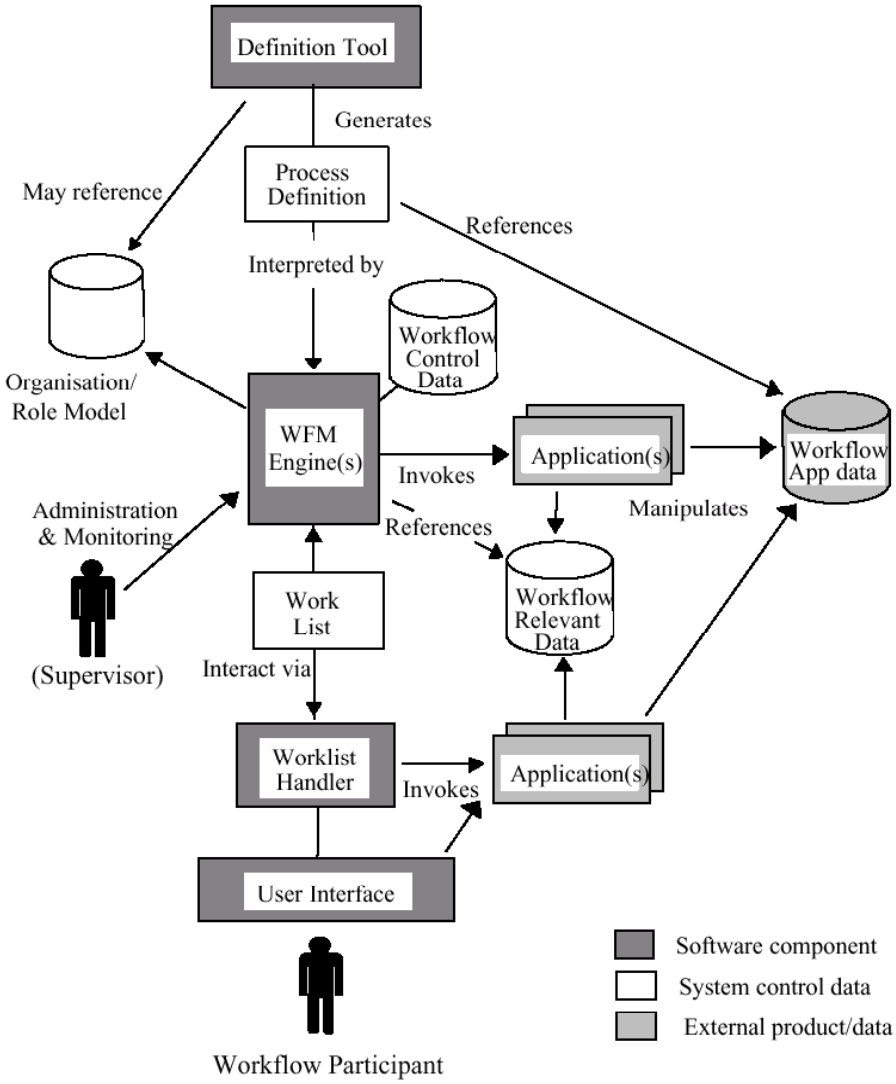


Abbildung 2: Struktur eines Workflow-Management-System[4]

Die wichtigste Eigenschaft eines „richtigen" WFMS ist die Trennung von Ablauflogik und Anwendungscode. Das heißt die Ablauflogik ist nicht in den Anwendungsprogrammen fest codiert, sondern sie wird durch sogenannte Ablaufgraphen innerhalb des WFMS repräsentiert. Diese können auf hohem Abstraktionsniveau geändert werden, um auf neue Anforderungen schnell reagieren zu können. Des weiteren werden durch das WFMS die im Ablaufgraphen definierten Anwendungsprogramme bzw. -funktionen aufgerufen. Dies hat den Vorteil, dass Softwarebausteine von unterschiedlichen Herstellern leicht in eine WFMS-basierte Anwendung eingebaut werden können. Das Konzept der Trennung von Ablauflogik und Anwendungscode ist im folgenden Schema verdeutlicht:

[4] Aus [WFMC99], Seite 39

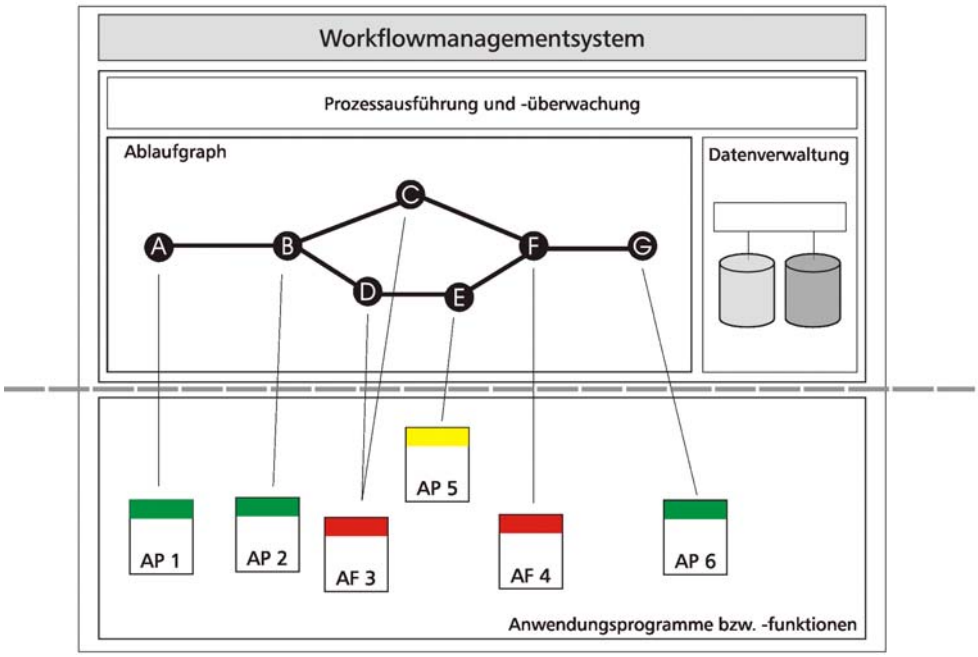


Abbildung 3: Trennung von Ablauflogik und Anwendungscode[5]

Neben diesen, durch die skizzierte Trennung geprägten, prozessorientierten WF-Systemen gibt es auch Groupware-Systeme und Dokumentenmanagementsysteme (DMS) mit Workflow-ähnlichem Charakter [DaRe00]. Groupware-Systeme sind im Gegensatz zu WFMS eher für schwach strukturierte Abläufe geeignet, wogegen DMS die Archivierung und Recherche von Dokumenten zum Zweck haben.

Im Folgenden liegt der Fokus auf prozessorientierten WFMS, die sich, je nach Struktur ihrer Prozesse, wieder unterschiedlich klassifizieren lassen. Hierauf wird im folgenden Abschnitt eingegangen.

2.4 Kategorisierung von rechnergestützten Prozessen

Bei der Kategorisierung nach der Struktur der modellierten Prozesse unterscheidet man zwischen Production-Workflow, Ad-hoc-Workflow und administrativem Workflow [Hast99]:

- ***Production-Workflow***

  Bei dem Production-Workflow [LeRo00] handelt es sich meist um stark strukturierte, repetitive Prozesse mit fester Ausführungsreihenfolge der Teilaufgaben. Es gibt wenige Ausnahmebehandlungen. Ein Production-Workflow hat einfache Kontrollstrukturen, wie Sequenz, Alternative und Parallelität.

[5] In Anlehnung an [ReDa00], Seite 25

- ***Ad-hoc-Workflow***

  Der Ad-hoc-Workflow besitzt eine flexible, zum Teil a priori unbekannte Prozessstruktur. Die Abfolge der einzelnen Aktivitäten ist stark situationsabhängig. Im Vorfeld können ein möglicher Kontrollfluss, sowie Vorbedingungen, die im Fall der Erfüllung eine bestimmte Aktivitätenausführung zur Folge haben, definiert werden. Während der Ausführung des Workflows können aber auch neue, a priori unbekannte und somit undefinierte Ausnahmesituationen auftreten, die wiederum zu einer Änderung des Kontrollflusses führen können (z.B. durch Hinzunehmen von Prozessschritten).

- ***Administrativer Workflow***

  Der administrative Workflow ist eine Mischung aus Produktions- und Ad-hoc-Workflow. Er besitzt eine wohldefinierte, jedoch nicht statische Prozessstruktur. Sie besteht aus flexiblen Arbeitsvorgängen mit bekannten Teilaufgaben, jedoch mit Freiheiten in ihrer Abfolge. Des weiteren existieren mehrere zulässige Ausführungsreihenfolgen.

2.5 Workflow-Standardisierungs-Gruppen

Um eine vereinfachte Zusammenarbeit verschiedener Softwaresysteme zu erreichen, werden Standards benötigt. Im Bereich Workflow-Management gibt es verschiedene Gruppen, die sich mit der Standardisierung von WFMS und ihren Schnittstellen beschäftigen. Die wichtigsten Gruppen sowie ihre Vorschläge und Standardisierungsbemühungen werden in diesem Abschnitt vorgestellt.

2.5.1 Workflow Management Coalition

Die Workflow Management Coalition (WfMC) wurde im August 1993 als eine „Non-profit" Organisation gegründet [WFMC01e]. In ihr sind Hersteller, Benutzer, Universitäten und Forschungsgruppen organisiert, um einen einheitlichen Standard im Bereich des Workflow-Management zu schaffen. Dieser soll zum einen die Interoperabilität zwischen Produkten verschiedener Hersteller ermöglichen, zum anderen soll die Integration von anderen IT-Diensten wie z.B. Dokumentenmanagement, verbessert werden. Des weiteren soll die Verbreitung von Workflow-Technologie gefördert werden. Mittlerweile umfasst die WfMC über 130 Mitglieder aus der ganzen Welt und hat sich als die führende Standardisierungsinstanz im Bereich des Workflow-Managements etabliert.

2.5.1.1 Workflow Reference Model

Die WfMC hat 1995 ihr *Workflow-Reference-Model* [WFMC95] (Abbildung 4) veröffentlicht, in welchem die Funktionsweise der Ablaufkomponente (*WF Enactment Service*), also der Workflow Engine(s) sowie die verschiedenen Schnittstellen zu den anderen Komponenten eines WFMS, definiert werden. Dabei macht die WfMC keine Angaben über die innere Struktur bzw.

Implementation eines WFMS und ihren Komponenten, sondern definiert nur deren Schnittstellen. Diese werden in Tabelle 1 in Kurzform dargestellt und in den folgenden Abschnitten näher erläutert.

Schnittstelle	Komponente	Funktionalität
Schnittstelle 1	Buildtime-Komponente	Import und Export von Prozessmodellen
Schnittstelle 2	Clients (z.B. für Arbeitslistenverwaltung)	Bereitstellung einer API zur Client-Programmierung
Schnittstelle 3	Anwendungsprogramme	Bereitstellung einer API zur Einbindung von Anwendungsprogrammen
Schnittstelle 4	Andere WFMS	Interoperabilität zwischen WFMS
Schnittstelle 5	Administrations- und Überwachungs-Tools	Bereitstellung einer API zur Programmierung von Administrations- und Überwachungswerkzeugen

Tabelle 1: Übersicht über die fünf Schnittstellen des WfMC Referenz Modells

Die Schnittstellen enthalten das Datenaustauschformat bzw. die Aufrufformate der verschiedenen Programmierinterfaces (API). Dadurch werden die Dienste eines WFMS nutzbar und die Interaktion zwischen dem Workflow Enactment Service und den WFMS-Komponenten geregelt. Im Kontext von WFMS wird eine API häufig als WAPI (Workflow API) bezeichnet.

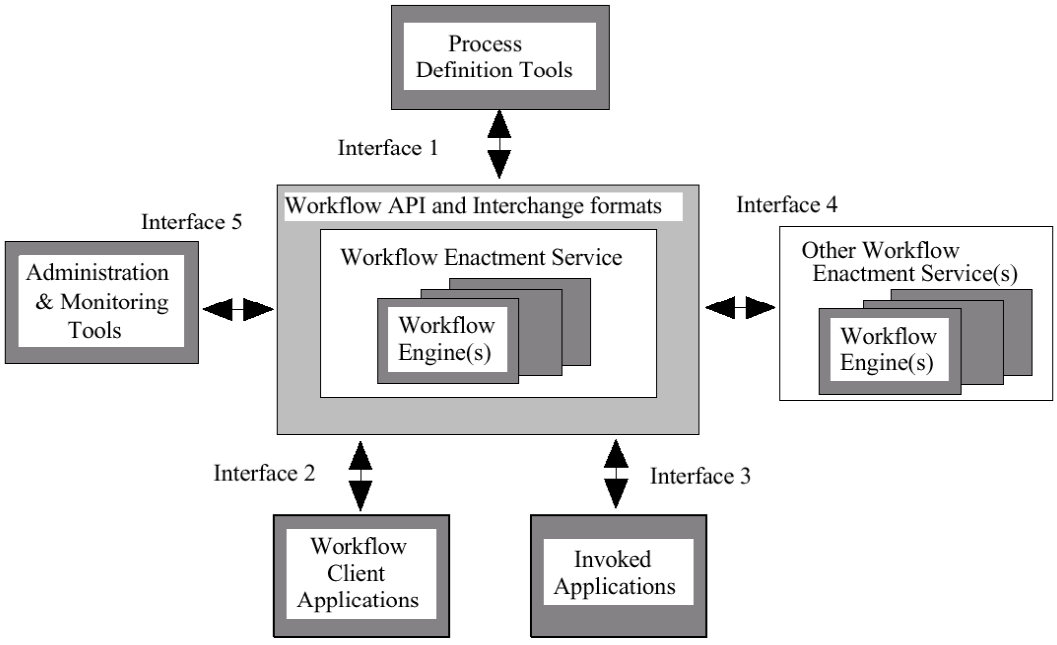


Abbildung 4: Das Workflow-Referenzmodell der WfMC [6]

[6] Aus [WFMC99], Seite 23

Abbildung 5 gibt einen Überblick über die momentan von der WfMC zu den verschiedenen Schnittstellen veröffentlichten Standard-Dokumente (Stand vom Mai 2001). Der jeweils aktuelle Stand kann unter [WFMC01d] eingesehen werden.

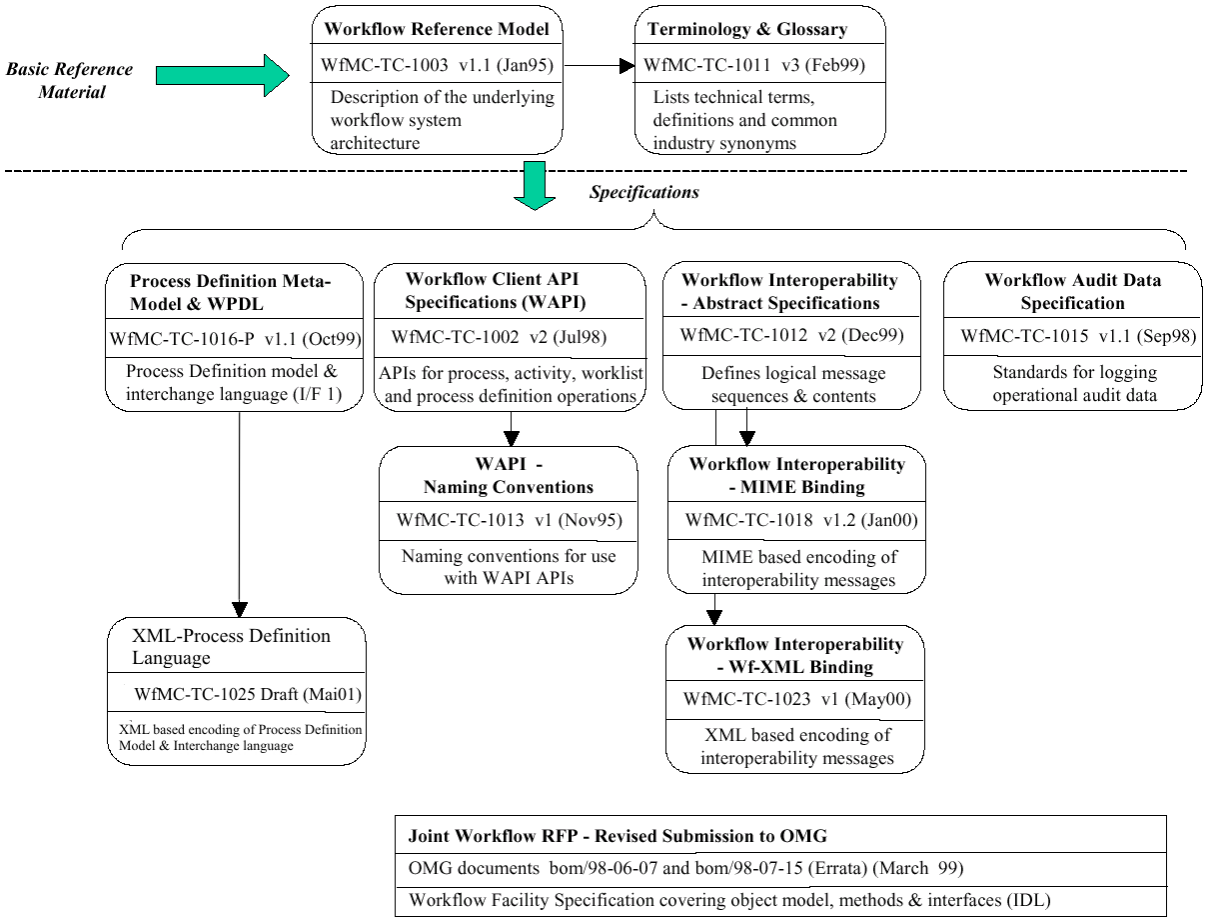


Abbildung 5: Workflow Standards und verwandte Dokumente der WfMC[7]

Für die Schnittstellen eins, vier und fünf existieren eigenständige Dokumente. Die Schnittstellen zwei und drei sind inzwischen zu einem Dokument zusammengefasst worden. In den folgenden Abschnitten werden die Anforderungen dieser Schnittstellen jeweils getrennt betrachtet. Das Dokument „Joint Workflow RFP" wurde in Zusammenarbeit von der WfMC und der OMG verfasst. Es wird in Abschnitt 2.5.2 noch näher vorgestellt.

Die einzelnen Schnittstellen aus Abbildung 4 sowie die Dokumente aus Abbildung 5 werden im Folgenden kurz dargestellt.

[7] In Anlehnung an [WFMC01b]

2.5.1.2 Import/Export von Prozessmodellen (Interface 1)

Der Austausch von Prozessmodellen zwischen Modellierwerkzeugen und Workflow-Engine steht im Mittelpunkt der Spezifikation der Schnittstelle 1 der WfMC. Ein standardisiertes Austauschformat für Prozessdefinitionen ist nötig, damit Werkzeuge (verschiedener Hersteller) zur Workflow-Modellierung (Bulidtime) verwendet werden können. Umgekehrt kann dadurch die Verwaltung der Prozessmodelle verschiedener WFMS mit dem selben Modellierwerkzeug geschehen. Schließlich wird im Hinblick auf die Interoperabilität verschiedener WFMS (siehe Interface 4) ein standardisiertes Austauschformat benötigt, damit Workflow-Definitionen zwischen verschiedenen WFMS übergeben und ausgeführt werden können.

Die Spezifikation [WFMC99a], die momentan in der Version 1.1 vorliegt (Stand: 29. Oktober 1999), untergliedert sich in mehrere Teile. Zuerst wurde ein Prozess-Metamodell definiert, welches den Aufbau einer Workflow-Definition festlegt:

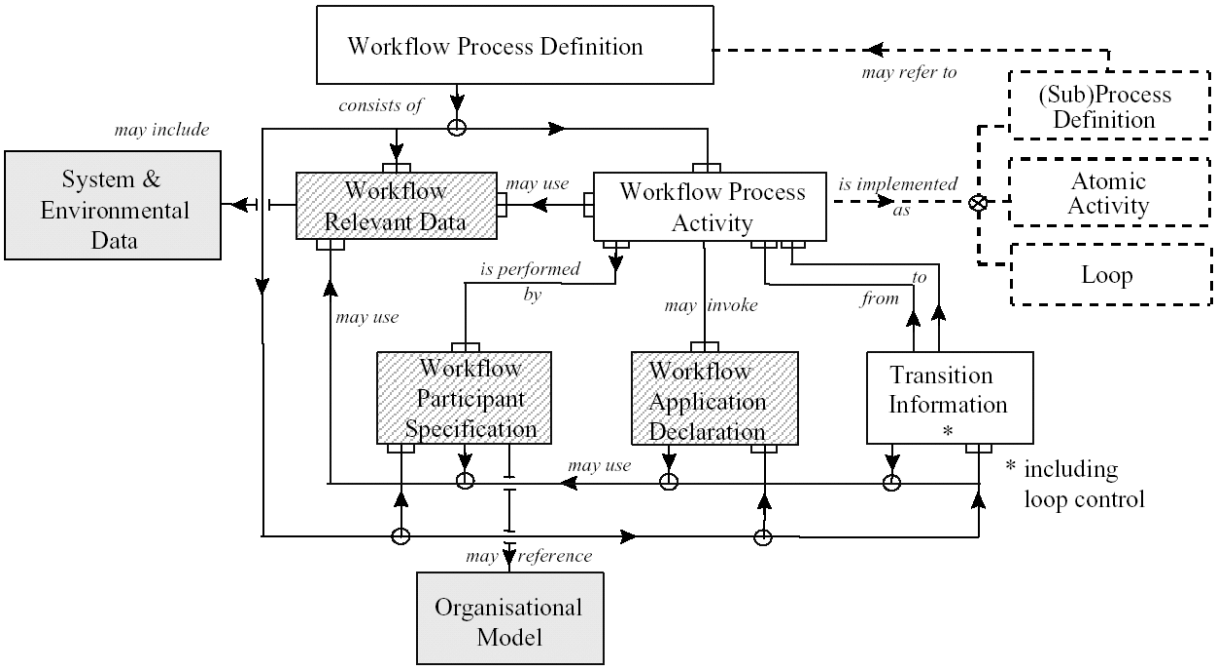


Abbildung 6: Prozess Metamodell[8]

Ein Workflow-Modell besteht aus einer oder mehreren Prozessaktivitäten, die ihrerseits elementaren Aktivitäten, Schleifen oder Sub-Workflows entsprechen. Die Ausführung einzelner Aktivitäten oder Sub-Workflows kann abhängig von bestimmten Ausführungs- bzw. Verzweigungsbedingungen (sogenannte Transitionsbedingungen) erfolgen. Dabei wird eine Aktivität jeweils bestimmten Teilnehmern bzw. Teilnehmergruppen unter Bezugnahme auf ein Organisationsmodell zugewiesen.

[8] Aus [WFMC99a], Seite 12

Workflow-Aktivitäten können externe Applikationen einbinden und Workflow-relevante Daten nutzen.

Ein weiterer Teil der Spezifikation ist die *Workflow Process Definition Language* (WPDL), die als formale Sprache die Definition und den Austausch von Prozessmodellen mittels Dateien oder eines gemeinsamen Speichers ermöglicht. In dieser Sprache werden Prozessabläufe, Transitionen, Transitionsbedingungen und Workflow-relevante Daten beschrieben. Dabei ist es unerheblich wie die interne Repräsentation einer Workflow-Definition implementiert ist - WFMS müssen eine Import/Export-Schnittstelle für das WPDL-Format unterstützen, damit z.B. Simulations-, Prozess-modellierungs- und Analysetools von verschiedenen Herstellern eingesetzt werden können.

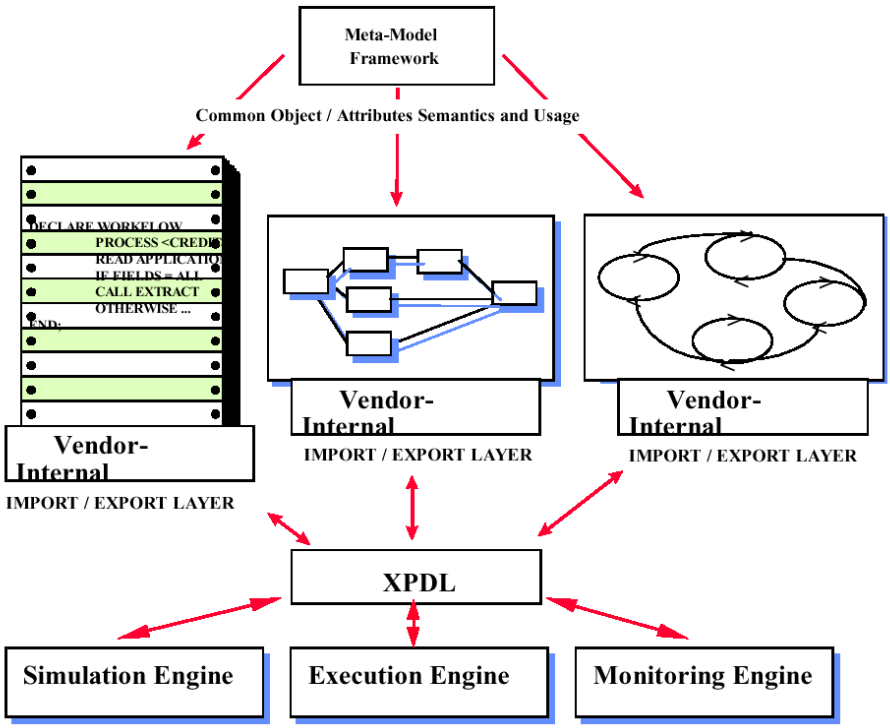


Abbildung 7: Das Konzept des Prozess-Definitions-Austausches[9]

Zusätzlich zu WPDL existiert noch eine API zur Manipulation von Prozessmodellen. Sie ist Teil des WAPI. Ihre Funktionen sind in [WFMC98] im Abschnitt *Workflow Definition Functions* zusammengefasst.

Inzwischen wurde von der WfMC unter der Bezeichnung *Workflow Process Definition Interface – XML Process Definition Language* [WFMC01] ein Draft vorgelegt, der auf der obigen Spezifikation aufbaut. Er nimmt allerdings die Repräsentation von Workflow-Modellen mit Hilfe einer XML-Definition [WFMC01a] vor, der sogenannten *XML Process Definition Language* (kurz XPDL).

[9] Aus [WFMC01a], Seite 6

2.5.1.3 Client- und Anwendungsprogrammierung (Interface 2 & 3)

Der Workflow-Client ist die (grafische) Schnittstelle zu den Nutzern eines Workflow-Management-Systems. Er bietet die Bedieneroberfläche, die das Erzeugen und Starten von Prozess-Instanzen ermöglicht. Die zur Ausführung anstehenden Tätigkeiten (*Workitems*) werden in Arbeitslisten eingetragen. Sie können mit Hilfe des Clients gestartet, beendet, unterbrochen oder angehalten werden. Dazu wird jeweils automatisch das zum Modellierzeitpunkt (Buildtime) verknüpfte Anwendungsprogramm aufgerufen. Ein Anwendungsprogramm ist dabei eine vom WFMS komplett getrennte Komponente (z.B. eine Bildschirmmaske), mit der Daten ausgetauscht werden können.

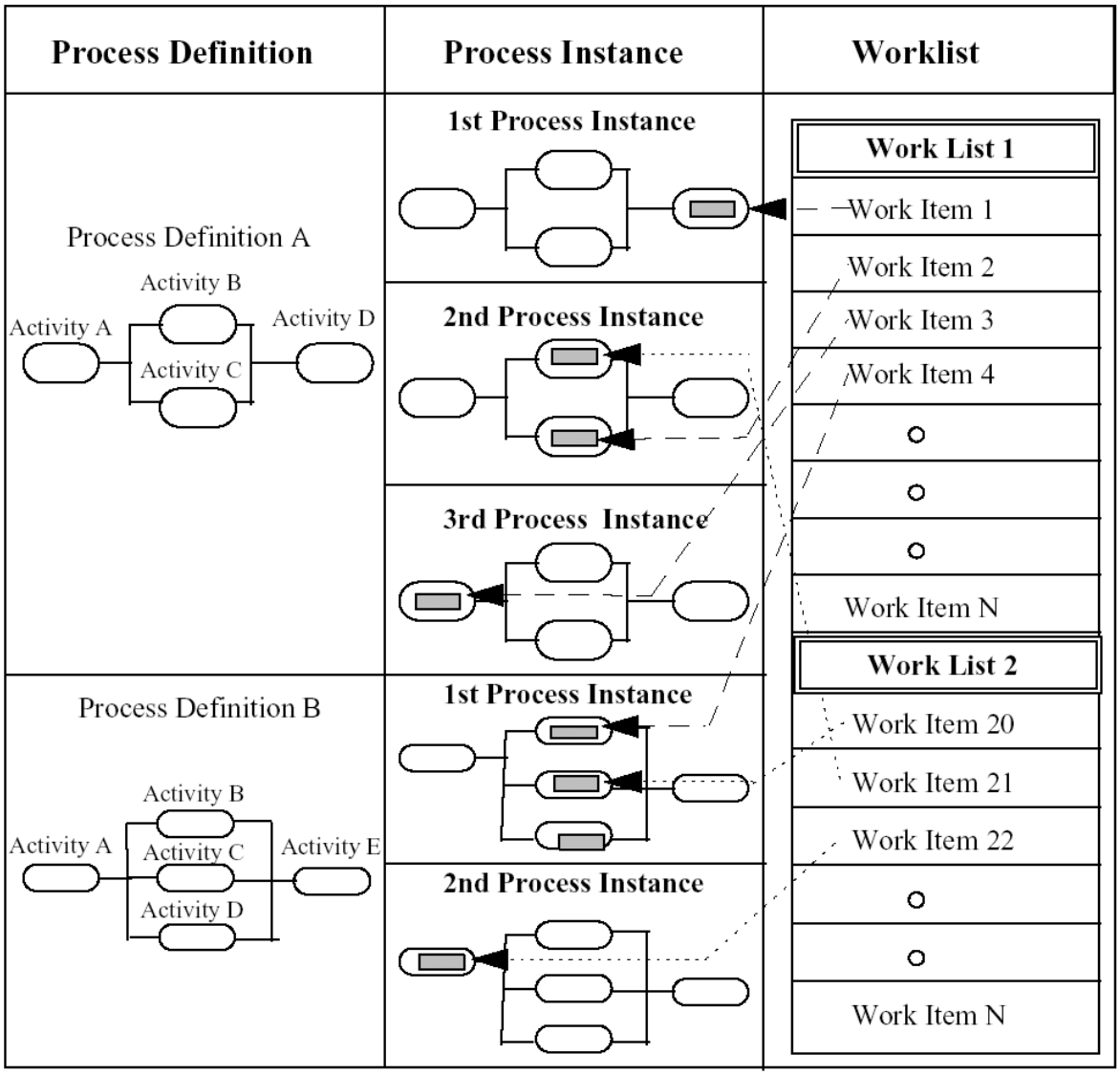


Abbildung 8: Zusammenhang von Prozessdefinitionen, -instanzen und Arbeitslisten[10]

[10] Aus [WFMC99], Seite 22

Je nach Rechten des Benutzers, stellt der Client eine Übersicht der Prozessmodelle (Process Definition), der laufenden Prozess-Instanzen (Process Instance) sowie der Arbeitsaufträge (Work-Items) in den entsprechenden Arbeitslisten (Worklist) dar. Den Zusammenhang zwischen diesen Begriffen zeigt Abbildung 8.

Ursprünglich wurde die Client- und die Anwendungsprogrammierung von der WfMC getrennt betrachtet (siehe [WFMC96a]). Später wurde die enge Verbindung dieser beiden Schnittstellen deutlich, weshalb sie inzwischen zu einer Schnittstelle zusammengefasst worden sind. Somit existiert nur noch eine Schnittstellendefinition, die als *Application Programming Interface (Interface 2&3)* veröffentlicht worden ist [WFMC98].

Die beiden folgenden Abschnitte betrachten die beiden Schnittstellen nochmals getrennt, um die verschiedenen Anforderungen und die Unterteilung der WAPI deutlich zu machen.

a) Client-Programmierung (Interface 2)

Die von der WfMC für Client-Anwendungen (*Workflow Client Applications*) festgelegte Programmier-Schnittstelle wird zur Erstellung angepasster Benutzerschnittstellen benötigt. Sie ermöglicht die Programmierung von Workflow-Clients, die auf die Funktionalitäten des WFMS zurückgreifen. Falls nicht der Standard-Client eines WFMS genutzt werden soll, kann ein Programmierer mit Hilfe dieser API entsprechend seiner Anforderungen einen eigenen Workflow-Client erstellen. Dieser zeichnet sich z.B. durch eine angepasste Benutzeroberfläche aus. Auch die Kommunikation mit mehreren WFMS ist vom Prinzip her möglich.

Das WAPI gliedert sich in folgende Funktionsgruppen:

- ***Verbindungskontrolle (Connection Functions)***
  z.B. Aufbau einer Verbindung zum WFMS
- ***Prozess-Kontrollfunktionen*** *(Process Control Functions)*
  z.B. Starten eines Prozesses, Unterbrechung einer Prozessausführung
- ***Prozess-Statusfunktionen*** *(Process Status Functions)*
  z.B. Erstellen einer Liste von Prozess-Instanzen anhand spezieller Kriterien (z.B. alle Prozess-Instanzen im Zustand `suspended`)
- ***Aktivitätenkontrollfunktionen*** *(Activity Control Functions)*
  z.B. Starten einer Aktivität, Beenden einer Aktivitätenausführung
- ***Workitem-Statusfunktionen*** *(Activity Status Functions)*
  z.B. Erstellen einer Liste von Arbeitsaufträgen anhand ausgewählter Kriterien bzw. Filter (z.B. alle Arbeitsaufträge des angemeldeten Users mit hoher Priorität)
- ***Steuerungsfunktionen für Arbeitslisten*** *(Worklist Functions)*
  z.B. Erstellen einer Liste mit Arbeitsaufträgen für eine bestimmte Person

- ***Administrationsfunktionen*** *(Administration Functions)*
  z.B. Ändern des Status aller Prozess-Instanzen eines Prozessmodells

In [WFMC97] werden noch WAPI Namenskonventionen vorgestellt, um eine verbesserte Lesbarkeit, eine leichte Portierbarkeit und eine hohe Wiederverwendbarkeit einer WAPI Implementation zu erreichen. Dieses Dokument enthält auch allgemeine Header-Files für die Programmierung in der Sprache C.

b) Anwendungsprogrammierung (Interface 3)

Die Schnittstelle 3 (*Invoked Applications*) definiert den Standard für den Aufruf von Anwendungskomponenten bei der Ausführung einzelner Aktivitäten. Über diese Schnittstelle wird das Anwendungsprogramm nicht direkt angesprochen, sondern indirekt über sogenannte *Tool Agenten*. Dabei handelt es sich um spezielle Anwendungstreiber (Wrapper) für bestimmte Aufruftechniken, wie z.B. OLE, DDE oder CORBA. Der Tool Agent ist für den Start bzw. die Beendigung von Anwendungsprogrammen sowie den Datenaustausch zuständig. Die Kommunikation läuft somit zwischen dem WFMS und dem Tool Agenten standardisiert ab (siehe Abbildung 9), unabhängig von der Art der Aufruftechnik der dahinter liegenden Anwendung. Dies lässt sich am besten mit dem ODBC-Aufruf im Bereich von Datenbanken vergleichen - hier ist der Aufruf auch unabhängig von der zugrundeliegenden Datenbank.

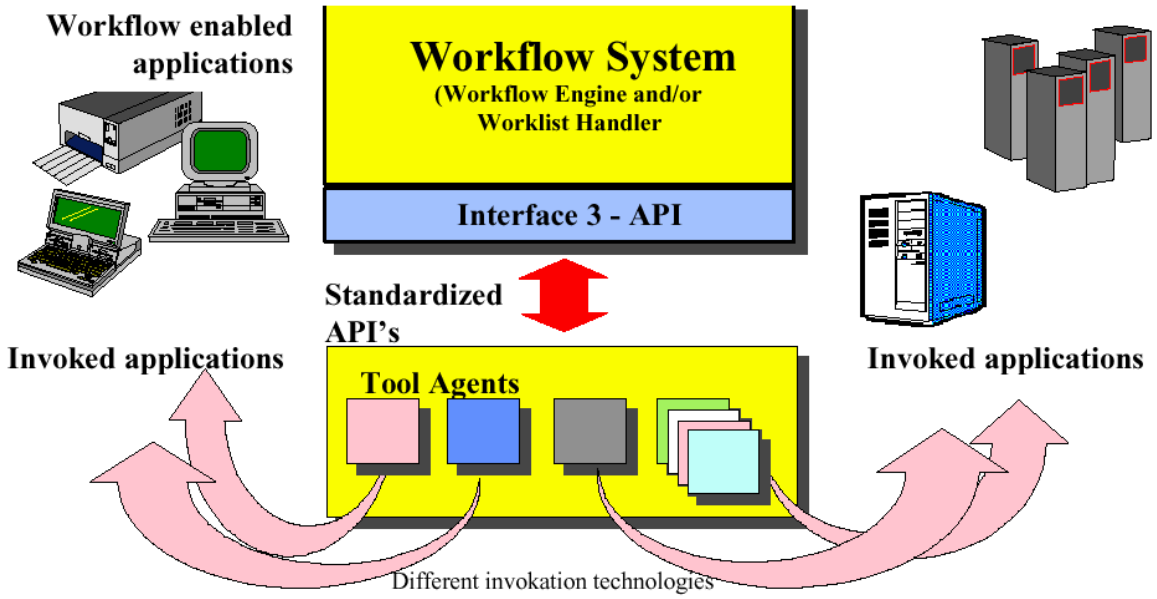


Abbildung 9: Funktionsweise der Tool Agenten[11]

[11] Aus [WFMC98], Seite 87

Die Funktionen des WAPI für diese Schnittstelle sind im Abschnitt *Application Invocation Functions* in [WFMC98] beschrieben. Sie lassen sich wie folgt gruppieren:

- ***Verbindung auf- bzw. abbauen (WMTAConnect und WMTADisconnect)***
  Erstellt und beendet eine Verbindung zum Tool Agenten.
- ***Anwendung aufrufen (WMTAInvokeApplication)***
  Weist einen Tool Agenten an eine Anwendung zu starten bzw. zu aktivieren. Dabei werden auch Workflow-relevante Daten (via Eingabedatencontainer) übergeben.
- ***Anwendungsstatus abfragen*** *(WMTARequestAppStatus)*
  Ermöglicht es dem WFMS den Status einer Anwendung (`running`, `waiting`) abzufragen.
- ***Anwendung beenden (WMTATerminateApp)***
  Weist einen Tool Agenten an eine Anwendung zu beenden. Die Rückgabe der Parameter-Daten an das WFMS erfolgt über sogenannte Ausgabedatencontainer.

2.5.1.4 Interoperabilität zwischen WFMS (Interface 4)

Um unternehmensübergreifend zwischen verschiedenen WFMS Prozessdaten austauschen und auf entfernten WFMS (Sub-)Prozesse ausführen zu können, wird eine entsprechende Interoperabilitäts-schnittstelle benötigt.

Die Schnittstelle 4 des WFMS-Referenzmodells regelt das Zusammenwirken verschiedener WFMS untereinander. Dazu wurde von der WfMC eine Spezifikation [WFMC96] veröffentlicht, die festlegt, wie die Zusammenarbeit von WFMS verschiedener Hersteller auszusehen hat. Diese Spezifikation wird in Kapitel 4 im Detail beschrieben.

Inzwischen sind mehrere Vorschläge für Interoperabilitätsstandards auf Basis dieser Spezifikation veröffentlicht worden. Sie verwenden z.B. E-Mail MIME- oder XML-Nachrichten zum Datenaus-tausch. Auf diese Vorschläge bzw. Standards wird in Kapitel 5 näher eingegangen.

2.5.1.5 Administrations- und Monitoringwerkzeuge (Interface 5)

Die Schnittstelle 5 des WfMC-Referenzmodells soll den Datenaustausch zwischen der WF-Engine und Werkzeugen für die Administration und das Monitoring von WF-Instanzen standardisieren. Dazu wurde von der WfMC eine *Audit Data Specification* [WFMC98a] veröffentlicht. Sie spezifiziert, welche Informationen während der Workflow-Ausführung von der WF-Engine erfasst und auf-gezeichnet werden. Diese Informationen lassen sich grob in die folgenden Gruppen untergliedern:

- ***Process Instance Audit Information***
  Audit-Daten zum Start, zu Statusänderungen oder zu Attributänderungen von WF-Instanzen.

- ***Activity Instance Audit Information***
  Audit-Daten zum Start, zu Statusänderungen oder zu Attributänderungen einzelner Aktivitäteninstanzen.
- ***Workitem Audit Information***
  Audit-Daten zur Zuweisung von Workitems an Nutzer sowie zu Status- und Attributänderungen von Workitems.
- ***Remote Operations Information***
  Audit-Daten zur Ausführung eines (Sub-) Workflows auf einem entfernten WFMS (z.B. Kommunikationsaufbau und -abbau, Start des Sub-Workflow bzw. Statuswechsel des Sub-Workflow auf dem entfernten WFMS.
- ***Process Definition Audit Information***
  Audit-Daten zu Statusänderungen von Prozess-Definitionen

Darüber hinaus wurden die folgenden (optionalen) Erweiterungen spezifiziert:

- ***Discretionary Audit Information***
  Audit-Daten, die zwischen zwei bzw. mehreren WFMS-Herstellern vereinbart werden. Sie sind jedoch nicht notwendig, um zum Standard der WfMC konform zu sein.
- ***Private Audit Information***
  Audit-Daten, die spezifisch vom jeweiligen Hersteller protokolliert werden. Sie dienen u.a. der Optimierung des Systems (z.B. zur Performance-Steigerung).

Diese Daten werden auch *Common Workflow Audit Data* (CWAD) genannt. Anhand dieser Daten kann ein Unternehmen nachvollziehen, was während der Ausführung eines Workflows abgelaufen ist. Auch lassen sich Erhebungen machen wie lange die durchschnittliche Ausführungsdauer eines bestimmten Workflows ist.

Die Art der Speicherung dieser CWAD-Daten ist nicht festgelegt. Ein Zugriff bzw. eine Auswertung ist daher auf mehrere Arten denkbar.

2.5.2 Object Management Group (OMG)

Neben der WfMC gibt es noch ein zweites Konsortium, das sich mit der Standardisierung von Workflows auseinandersetzt. Die Object Management Group (OMG) [OMG01] hat sich zum Ziel gesetzt, die Verbreitung der Objekttechnologie zu fördern und hat mit CORBA [OMG95b] hierzu eine Referenzarchitektur vorgestellt. CORBA stellt eine plattform- und sprachunabhängige Infrastruktur für verteilte Objekte bereit. Die OMG hat sich entschlossen ihre Standardisierungsbemühungen auf weitere Gebiete auszuweiten und mit den *CORBA Facilities* Standardisierungsbestrebungen für eine Reihe von weiteren Teilgebieten vorgeschlagen, die als Dienste bezeichnet

werden. All diesen Facilities ist gemeinsam, dass sie auf der von CORBA bereitgestellten Objekttechnologie aufsetzen.

2.5.2.1 Object Management Architecture (OMA)

Das Referenzmodell der *Object Management Architecture* [OMG95] besteht aus vier Komponenten (siehe Abbildung 10): dem *Object Request Broker*, den *Object Services*, den *Common Facilities* sowie den *Application Objects*. Diese werden im Folgenden näher erläutert.

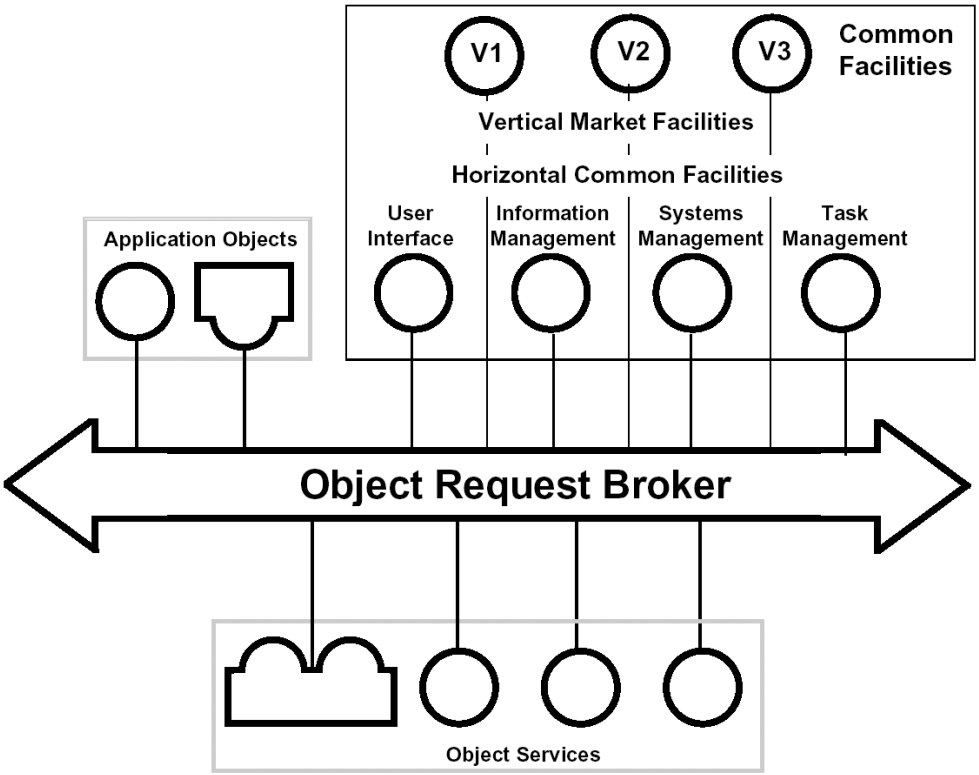


Abbildung 10: Referenzmodell der Object Management Architecture (OMA)[12]

- ***Object Request Broker (ORB)***
  Der ORB ist die zentrale, vermittelnde Basiskomponente in der CORBA-Architektur. Er ermöglicht es Objekten in einem verteilten System, transparent Anfragen zu stellen bzw. zu erhalten. Er bildet die Grundlage für die Kommunikation von verteilten Objekten in homogenen und heterogenen Systemen. Die Architektur sowie die Spezifikationen des ORB sind in [OMG95b] nachzulesen.

[12] Aus [OMG95a], Seite 1-2

- ***Object Service***

  Die Object Services [OMG98] sind eine Sammlung von systemnahen Diensten. Sie bieten Basisfunktionen, die von höheren Diensten bzw. von Anwendungen genutzt werden können. Die Dienste kommunizieren über den ORB miteinander und sind in jeder CORBA-Implementation vorhanden.

- ***Common Facilities***

  Die Common Facilities [OMG95a] sind eine Sammlung von höherwertigen Diensten für unterschiedliche Anwendungsarten, die nicht so elementar wie die Object Services sind. Hier unterscheidet man zwischen vertikalen Facilities, die in allen Bereichen benutzt werden (z.B. E-Mail), und horizontalen, also branchenspezifischen Facilities. Im Gegensatz zu den Object Services müssen nicht alle Common Facilities in jedem CORBA-konformen Produkt enthalten sein.

- ***Application Objects***

  Die Application Objects sind die eigentlichen Anwendungen, die speziell für bestimmte Domänen erstellt werden. Sie greifen wiederum auf Teile der Common Facilities bzw. auf die Object Services zurück. Sie unterliegen keinem OMG Standards und haben somit keine standardisierten Schnittstellen.

Die Eigenschaften dieser Objekte werden durch die sogenannte *Interface Definition Language* (IDL) beschrieben, deren Syntax an C++ angelehnt ist. Sie ermöglicht es, auf eine standardisierte und implementationsunabhängige Weise, die Eigenschaften solcher Objekte zu beschreiben. Für die Objekte ist es irrelevant, in welcher Programmiersprache sie implementiert sind; dies kann z.B. C++, Smalltalk oder Java sein. Sie verfügen über standardisierte Schnittstellen (Eigenschaften/Methoden) und können so mit Hilfe des ORB untereinander kommunizieren.

2.5.2.2 Workflow Facility

Einer der von der OMG vorgeschlagenen anwendungsabhängigen Dienste ist das *Workflow Facility* [OMG99], für das Anfang 1997 eine Arbeitsgruppe eingerichtet wurde. Die Gruppe veröffentlichte im Mai 1997 ein entsprechendes *Request for Proposal* (RFP) als dessen Ziel "die Definition einer Schnittstelle und deren Semantik für die Ausführung und Manipulation von verteilten Workflow-Objekten und deren Metadaten" angegeben wird. Die Workflow Facility soll hierbei als Plattform zum Aufbau flexibler Workflow-Anwendungen dienen, welche die Verbindung von Objekten und existierenden Applikationen realisiert. Im Juli 1998 wurde daraufhin als Reaktion auf das RFP ein überarbeiteter Vorschlag, basierend auf Standards der WfMC, veröffentlicht. Dieser Vorschlag wird von einer Reihe führender Unternehmen im Bereich Workflow-Technologie getragen und stellt eine Basis für die Integration der Workflow Technologie in das OMG-Architekturmodell dar. Es ist auch unter dem Namen „Joint Workflow Facility“ oder kurz „jFlow“ bekannt. Die Spezifikation liegt inzwischen in der Version 1.2 vom April 2000 vor (siehe [OMG00]). Wie die Objekthierarchie in jFlow definiert wurde, ist in Abbildung 11 dargestellt.

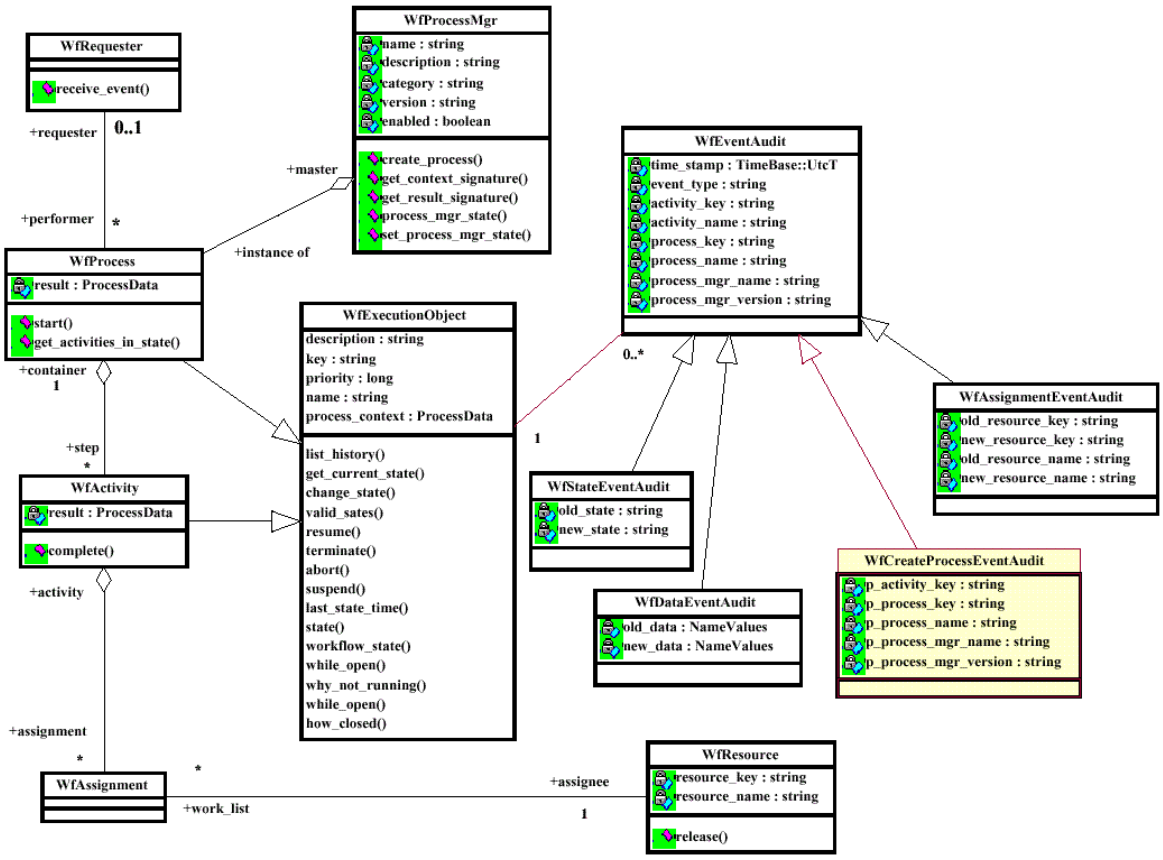


Abbildung 11: Joint Workflow Management Facility Model[13]

Dabei entspricht das Objekt *WfProcess* einer Prozess-Instanz. Diese wird durch das *WfProcessMgr*-Objekt, ausgehend von einer bekannten Prozess-Definition, instanziert. Die Prozess-Instanz besteht aus einer oder mehreren Aktivitäten, den *WfActivity*-Objekten. Diese werden durch sogenannte *WfAssignment*-Objekte Ressourcen, die *WfResource*-Objekte, zugewiesen. Daneben definiert das jFlow-Modell noch weitere Objekte, die unter anderem der Ausführung und der Protokollierung dienen.

2.5.3 Weitere Arbeitsgruppen

Neben der WfMC und der OMG gibt es noch andere Gruppen, die sich mit Teilaspekten einer Standardisierung im Workflow-Bereich beschäftigen. Hier seien beispielhaft das *Process Specification Language* (PSL) Projekt [PSL01] am *National Institute of Standards and Technology* (NIST), die *ODMA Extension for Workflow* [Hast99] der ODMA-Gruppe (Open Document Management API), sowie das *Simple Workflow Access Protocol* (SWAP) [SWEN98] der *Internet Engineering Task Force* (IETF) genannt.

[13] Aus [OMG00], Seite 2-3

2.5.3.1 Process Specification Language (PSL)

PSL dient dem Austausch von Prozessinformationen auf der Ebene des Interface 1 der WfMC. Dabei werden die auszutauschenden Prozessmodelle auf der Grundlage einer standardisierten Beschreibungssprache definiert. Nähere Informationen finden sich auf der PSL-Homepage [PSL01] sowie der PIF Spezifikation [PSLoJ]. In diesem Zusammenhang sei auch das *Process Interchange Format* (PIF) der PIF-Working-Group [PIF01] erwähnt. Es ist inzwischen im PSL-Projekt aufgegangen, da die Konzepte der beiden Ansätze sehr ähnlich sind. Zuvor hatte PSL seinen Fokus auf Fertigungsprozesse, wohingegen PIF seine Schwerpunkte im Bereich der Geschäftsprozesse legte.

2.5.3.2 ODMA Extension for Workflow

Die ODMA-Gruppe (*Open Document Management API*) ist eine offene Vereinigung von Herstellern von Dokumenten-Management-Systemen (DMS), elektronischen Archivsystemen (EAS) und Branchen-Applikationen. Die ODMA wurde unter dem Fachverband AIIM *(Association for Information and Image Management*) [AIIM01] Anfang 1994 gegründet. Sie besteht aus etwa 100 Mitgliedern, denen z.B. Corel, Novell, Oracle und Xerox angehören.

Der ODMA Fokus liegt auf der Integration von Dokumenten-Management-Systemen und Anwendungssoftware. Besonderes Augenmerk erhalten Office-Anwendungen wie Textverarbeitung und Tabellenkalkulation. Unterstützt eine Anwendung das ODMA-API, so kann die Applikation auf vorhandene Archiv- und Dokumenten-Management-Funktionen zurückgreifen und dort Dokumente ablegen und abrufen.

Mit der *ODMA Extension for Workflow*, die im April 1995 spezifiziert wurde, versucht die ODMA-Gruppe eine Standard-Schnittstelle zur Applikations-WFMS-Integration zu schaffen. Dabei folgt sie stark dem Ansatz der ODMA-Spezifikation zur Applikations-DMS-Integration. Drei Anwendungs-schwerpunkte sah die ODMA-Gruppe:

- Starten und Verknüpfen eines Workflows mit dem/den aktuellen Dokument(en).
- Bearbeiten des Dokuments eines auszuführenden Workitems.
- Signalisierung des Endes der Bearbeitung dieses Dokumentes an die Workflow-Engine.

Der Ansatz der ODMA-Gruppe hat sich jedoch nicht durchgesetzt. Zum einen ist er sehr dokumentenzentrisch, zum anderen befindet er sich in direkter Konkurrenz zu den Schnittstellen der Workflow-Management-Coalition [Hast99].

2.5.3.3 Simple Workflow Access Protocol (SWAP)

Das *Simple Workflow Access Protocol* [SWEN98] soll auf einfache Weise Interoperabilität im WF-Bereich ermöglichen. Es standardisiert die Kommunikation zwischen WFMS auf Basis des *Hyper Text Transfer Protocol* (HTTP) mit Hilfe von XML-Nachrichten. Auf diesen Vorschlag der IETF wird unter 5.4 noch näher eingegangen.

2.6 Zusammenfassung

Dieses Kapitel hat die grundlegenden Konzepte von Workflow-Management erörtert und einen Überblick zu den wichtigsten Standardisierungsbemühungen aus diesem im Bereich gegeben. Als wichtigste Standardisierungsgruppe ist die Workflow Management Coalition mit ihrem Workflow Referenz Modell zu nennen. Dieses Modell bildet heute die Grundlage sämtlicher ernstzunehmender Standards im WF-Bereich.

Mit dem Interface 1 wurde ein Austauschformat für den Import und Export von Prozessmodellen definiert. Allerdings geht dieser Ansatz nicht darauf ein, dass Geschäftsprozess-Modelle und Workflow-Modelle unterschiedliche Zielgruppen und damit – zumindest teilweise – auch unterschiedliche Inhalte haben [ReDa00]. Bei der Geschäftsprozess-Modellierung sind beispielsweise Verzweigungswahrscheinlichkeiten zur Durchführung von Simulationen relevant, die in dem Prozess-Metamodell des Interface 1 gar nicht bedacht wurden. Um zu sinnvollen Simulationsergebnissen zu gelangen werden bei der Geschäftsprozess-Modellierung des weiteren oftmals Aspekte abgebildet, die für eine elektronische Unterstützung nicht relevant bzw. nicht gewünscht sind und die deshalb nicht durch das Workflow-Modell berücksichtigt werden sollen [RSD97]. Das bedeutet, dass eine 1:1 Abbildung von einem semantischen Prozessmodell auf ein WF-Modell in der Praxis oftmals nicht möglich ist. Genau auf diese Annahme stützt sich aber das Interface 1 der WfMC. Schließlich bleiben bei der Definition des Interface 1 viele für die Modellierung und Steuerung von Workflows relevante Aspekte ausgeklammert. Beispielsweise wird nicht auf Zeitaspekte (z.B. Termine, Zeitabstände, Deadlines) oder Interworkflow-Abhängigkeiten eingegangen. Entsprechende Modell-Informationen sind in der Praxis aber für viele prozessorientierte Anwendungen (z.B. für Abläufe aus dem Krankenhaus- oder dem Engineering-Bereich [RSD97,DRK00]) essentiell.

Das Interface 2 und 3 des Referenz-Modells definiert ein standardisiertes API für die Client- und Anwendungsprogrammierung (WAPI). Doch auch dieser Standard geht nicht sehr weit. Es gibt beispielsweise keine Funktionen zur dynamischen Änderung von Workflow-Prozessen [ReDa98] oder zum Setzen von Terminen (vgl. oben). So sind nur die allerwichtigsten Grundfunktionen in diesem Interface definiert. Bietet ein WFMS hingegen solche fortschrittlichen Funktionen an, können diese nicht über das Standard-API benutzt werden. Des weiteren bleibt festzustellen, dass die WAPI-Spezifikation bisher ausschließlich in der Programmiersprache C erfolgt ist. Eine Portierung auf objektorientierte Sprachen (C++, Java) ist bis heute noch nicht erfolgt.

Auf Interface 4 wurde in diesem Kapitel nicht näher eingegangen, da die hier getroffenen Standardisierungen in den folgenden Kapiteln noch genauer diskutiert werden.

Das Interface 5, die Audit-Data-Spezifikation, definiert welche Informationen im Verlauf der Ausführung einer Workflow-Instanz protokolliert werden sollen. Allerdings wird nicht spezifiziert, wie diese Daten gespeichert und wie sie zu Auswertungszwecken wieder abgerufen werden können. Des weiteren muss die Aussagekraft dieser Daten bezweifelt werden, da sie auf starren WF-Aus-

führungen basieren; das heißt die Protokollierung weitergehender Informationen, etwa zu Laufzeitabweichungen vom geplanten Ablauf, ist nicht vorgesehen.

Zusammenfassend lässt sich feststellen, dass das Referenz-Modell der WfMC eine gute Basis für Standardisierungen im Bereich des Workflow-Management darstellt. Allerdings muss der heutige Funktionsumfang für die Zukunft erweitert werden, um auch für anspruchsvolle Anwendungen geeignet zu sein. Des weiteren müssen existierende Standards in einzelnen Bereichen noch genauer spezifiziert werden. Ob dies tatsächlich gelingen wird, darf bezweifelt werden. So hat es in den Spezifikationen der Interfaces 1, 2 3, und 5 in den letzten Jahren wenig „Bewegung“ gegeben.

3 Unternehmensübergreifende Prozesskopplung

Zunehmende Wettbewerbsintensität und -dynamik führen zu einem Umbruch in der Unternehmensführung. Erforderlich sind neue Strategien um das Erfolgspotential der Unternehmung nachhaltig zu sichern. Als oberstes Ziel gilt es Qualität, Produktivität, Kosten und Kundenzufriedenheit zu optimieren.

Ein Ansatz dieser Optimierungsbemühungen ist das *Supply-Chain-Management* (kurz: SCM) [KnMeZe00], mittels dem komplette Lieferketten elektronisch unterstützt werden sollen. Wichtigste Ziele dabei sind stark verkürzte Durchlaufzeiten, maximale Termintreue, minimale Lagerbestände, flexibler Einsatz von Kapazitäten und Ausrichtung der Prozesse am Kundennutzen. Zu diesem Zweck sollen die Prozessketten und Informationsflüsse im Unternehmen bzw. zwischen Unternehmen optimiert und durchgängig durch EDV-Systeme elektronisch unterstützt werden.

In diesem Kapitel wird anhand von einfachen Beispielen gezeigt wie Unternehmen durch die EDV-gestützte Kopplung von Prozessen den oben genannten Zielen ein Stück näher kommen können und welche Anforderungen es dabei zu bewältigen gilt.

3.1 Ausgangssituation

Wir betrachten den Herstellungsprozess und den Vertrieb eines fiktiven Produkts. Dieses Produkt setzt sich aus verschiedenen Komponenten zusammen, welche von einem bestimmten Lieferanten bereitgestellt werden. Diese Komponenten könnten auch von mehreren Lieferanten ausgeliefert werden. Aus Gründen der Vereinfachung wird im Folgenden aber von genau einem Lieferanten ausgegangen. Des weiteren nehmen wir an, dass der Hersteller nur ein Produkt mit verschiedenen Ausprägungen (Konfigurationen) erzeugt, das dann von einem Händler vertrieben wird. Schematisch sieht diese Lieferkette wie folgt aus:

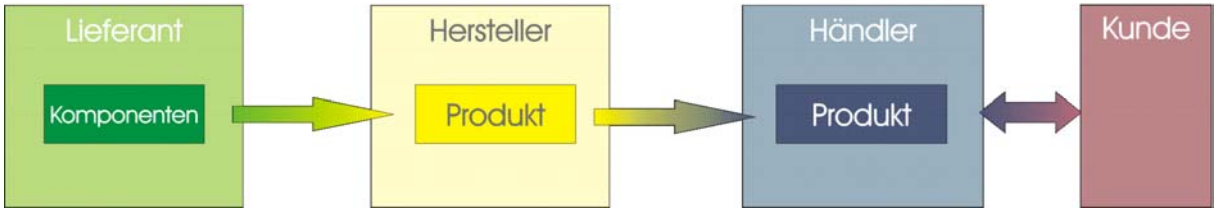


Abbildung 12: Schematische Darstellung der Ausgangsituation

Die nachfolgenden Beispiele erheben keinen Anspruch auf Vollständigkeit, sondern sollen lediglich die Vorteile einer unternehmensübergreifenden Prozesskopplung aufzeigen.

3.2 Fallbeispiele

Dieser Abschnitt diskutiert mehrere mögliche Szenarien, die beim Vertrieb und der Herstellung des obigen Produktes auftreten können. Dazu zählt die Verfügbarkeitsprüfung eines Produkts durch den Kunden, also die Prüfung, wie schnell das Produkt lieferbar ist. Des weiteren werden der eigentliche Bestellvorgang sowie eine Statusabfrage, die ermittelt wie weit der Herstellungs- bzw. Lieferprozess fortgeschritten ist, dargestellt. Schließlich wird noch die Möglichkeit der Stornierung einer Bestellung aufgezeigt.

3.2.1 Verfügbarkeitsprüfung durch Kunden

In der Praxis interessiert sich ein Endkunde oftmals dafür, ob das Produkt aktuell verfügbar ist bzw. wie lange die Lieferzeit voraussichtlich sein wird. Entsprechende Anfragen müssen vom Händler rasch beantwortet werden können.

In herkömmlichen Systemen läuft dieser Prozess etwa wie folgt ab: Der Kunde stellt eine Anfrage an den Händler z.B. per Telefon, per Fax oder über das Internet, falls eine Online-Abfragemöglichkeit besteht. Häufig erhält er dann nur die Antwort, ob das Produkt aktuell verfügbar ist oder nicht, ohne Nennung des möglichen Liefertermins. Da der Händler bestrebt sein wird, seine Lagerbestände niedrig zu halten, kommt es häufig vor, dass das Produkt nicht ab Lager lieferbar ist. Falls dem so ist, kann der Händler ohne Nachfrage beim Hersteller meist keinen Liefertermin angeben. Die hierzu notwendige Anfrage erfolgt dann wieder telefonisch, per Fax oder Online durch den Händler (beim Hersteller des Produkts). Muss nun der Hersteller seinerseits bei seinen Zulieferern nachfragen, ist weiterer Aufwand nötig (siehe Abbildung 13). Alles in allem kann die Bearbeitung einer Kundenanfrage recht lange dauern und sich sehr aufwendig gestalten, da mehrere Personen aus verschiedenen Firmen involviert sind. Hinzu kommt, dass die Systeme der Beteiligten in der Regel nicht miteinander gekoppelt sind.

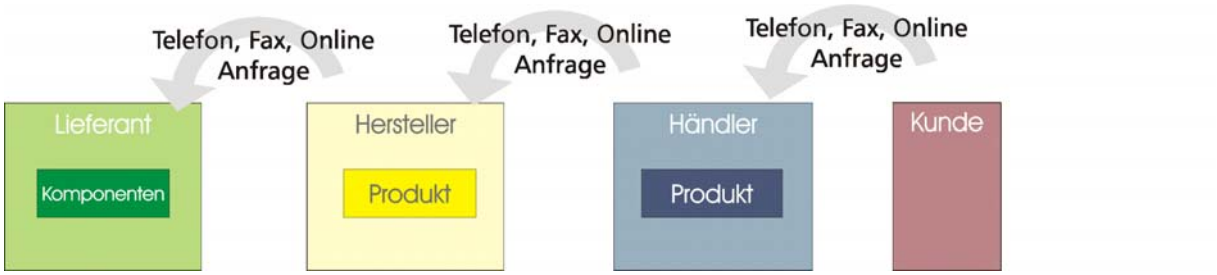


Abbildung 13: Darstellung einer herkömmlichen Verfügbarkeitsanfrage

Um solche Anfragen kosteneffizient und schnell beantworten zu können, bietet sich der Einsatz unternehmensübergreifender WFMS an. Dieses Szenario könnte auch mittels eines anderen speziellen Anwendungssystems implementiert werden, jedoch sind WFMS durch die größere

Flexibilität sowie die leichte Anpassbarkeit WFMS-basierter Anwendungen an geänderte Situationen für diese Art der Unternehmenskopplung gut geeignet.

Durch geeignete Modellierung der Prozesse ließe sich obige Anfrage wesentlich vereinfachen:

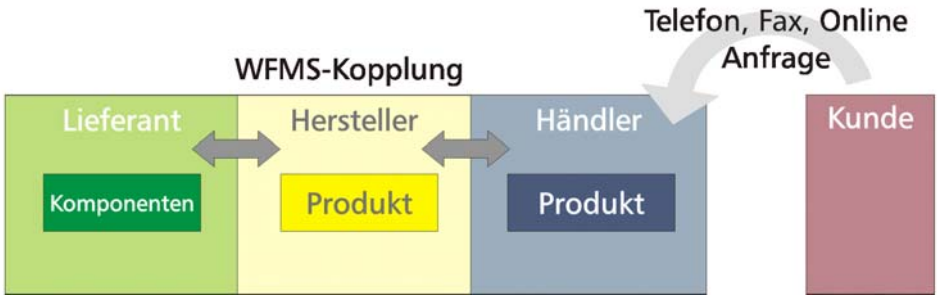


Abbildung 14: Darstellung einer verknüpften Verfügbarkeitsanfrage

Dabei kann der Kunde sich z.B. über das Internet auf der Webseite des Händlers einloggen, um seine Verfügbarkeitsanfrage zu stellen. Nun wird im WFMS des Händlers der Prozess gestartet, der die Verfügbarkeit prüft. Falls das gewünschte Produkt nicht auf Lager ist, wird das WFMS des Herstellers befragt, wie lange die Lieferzeit für das Produkt ist. Diese berechnet sich dann aus der Verfügbarkeit der Komponenten, die sich ebenfalls durch eine Anfrage im WFMS des Lieferanten bestimmen lässt.

Den Ablaufgraph der verknüpften Verfügbarkeitsanfrage zeigt Abbildung 15. Man erkennt an der Färbung der einzelnen Prozess-Schritte, auf welchen WFMS diese jeweils ausgeführt werden.

3.2.2 Kundenbestellung

Bei der Bestellung eines Produkts beim Händler durch den Kunden laufen die folgenden Prozesse ab:

Der Kunde bestellt das Produkt telefonisch, schriftlich oder Online beim Händler. Hat der Händler es auf Lager, wird es per Paketdienst bzw. Spedition dem Kunden ausgeliefert. Falls es nicht auf Lager ist, wird es beim Hersteller bestellt. Sind wiederum beim Hersteller nicht alle Komponenten verfügbar, müssen diese zuerst beim Lieferanten bestellt werden. Nach der Produktion beim Hersteller wird das Produkt an den Händler ausgeliefert, und dieser liefert es dann an den Kunden aus. Diese Weiterleitung der Informationen zwischen den beteiligten Unternehmen dauert eine gewisse Zeit und beansprucht entsprechende Ressourcen (z.B. Personal der beteiligten Firmen).

Durch den Einsatz von WFMS und durch eine unternehmensübergreifende Prozesskopplung kann sofort nach dem Bestelleingang beim Händler eine Verfügbarkeitsprüfung erfolgen. Falls die bestellte Ware beim Händler verfügbar ist, wird im WFMS des Paketdienstes automatisch ein Abholauftrag gestartet. Dieser veranlasst die Abholung der Lieferung, so dass die Ware dem Kunden anschließend durch den Paketdienst zugestellt wird. Ist das Produkt nicht auf Lager, wird dagegen die Bestellung sofort an den Hersteller bzw. an das WFMS des Herstellers weitergeleitet. Dieses WFMS wiederum überprüft, ob das Produkt ab Lager verfügbar ist.

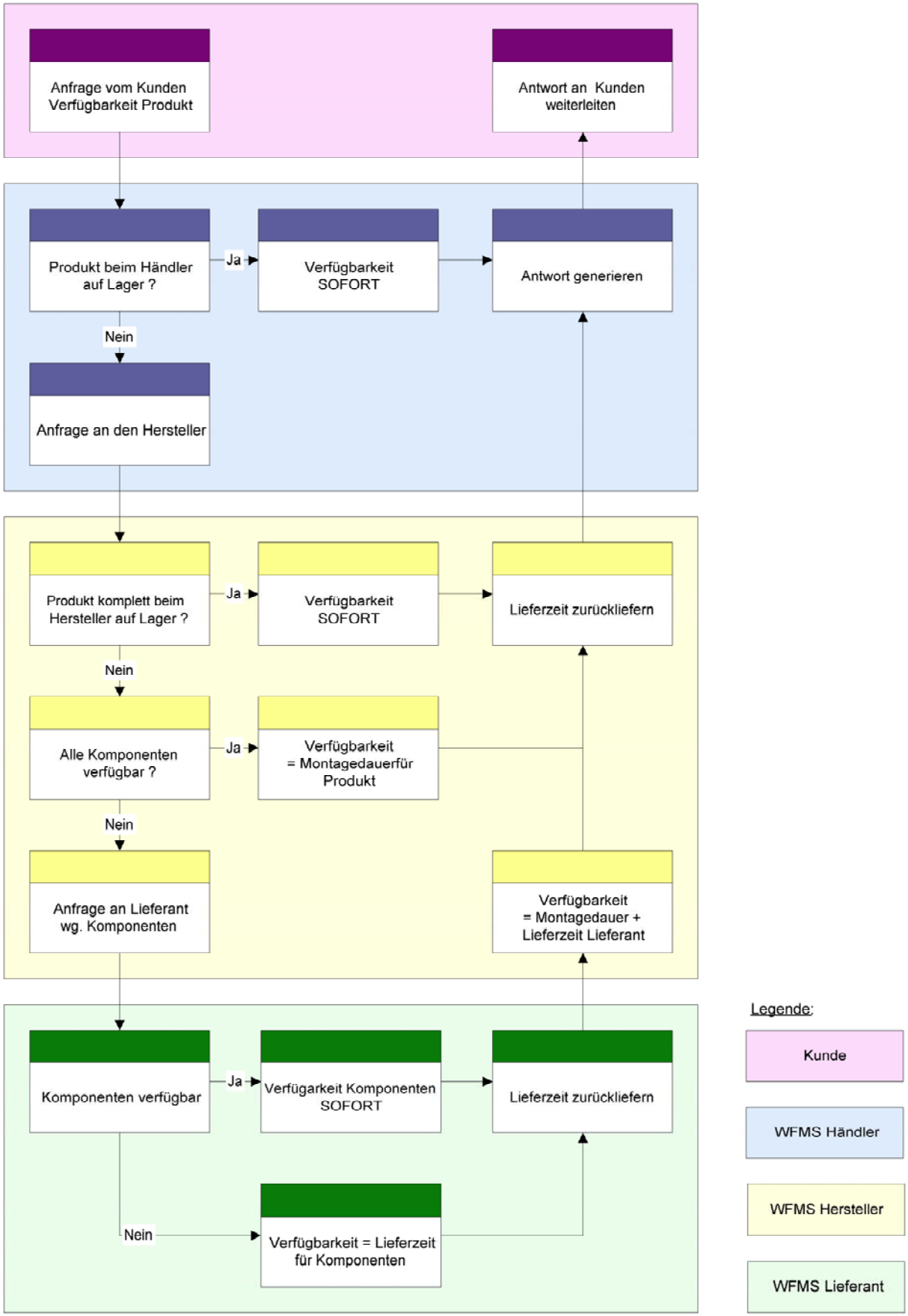


Abbildung 15: Ablaufgraph einer Verfügbarkeitsanfrage

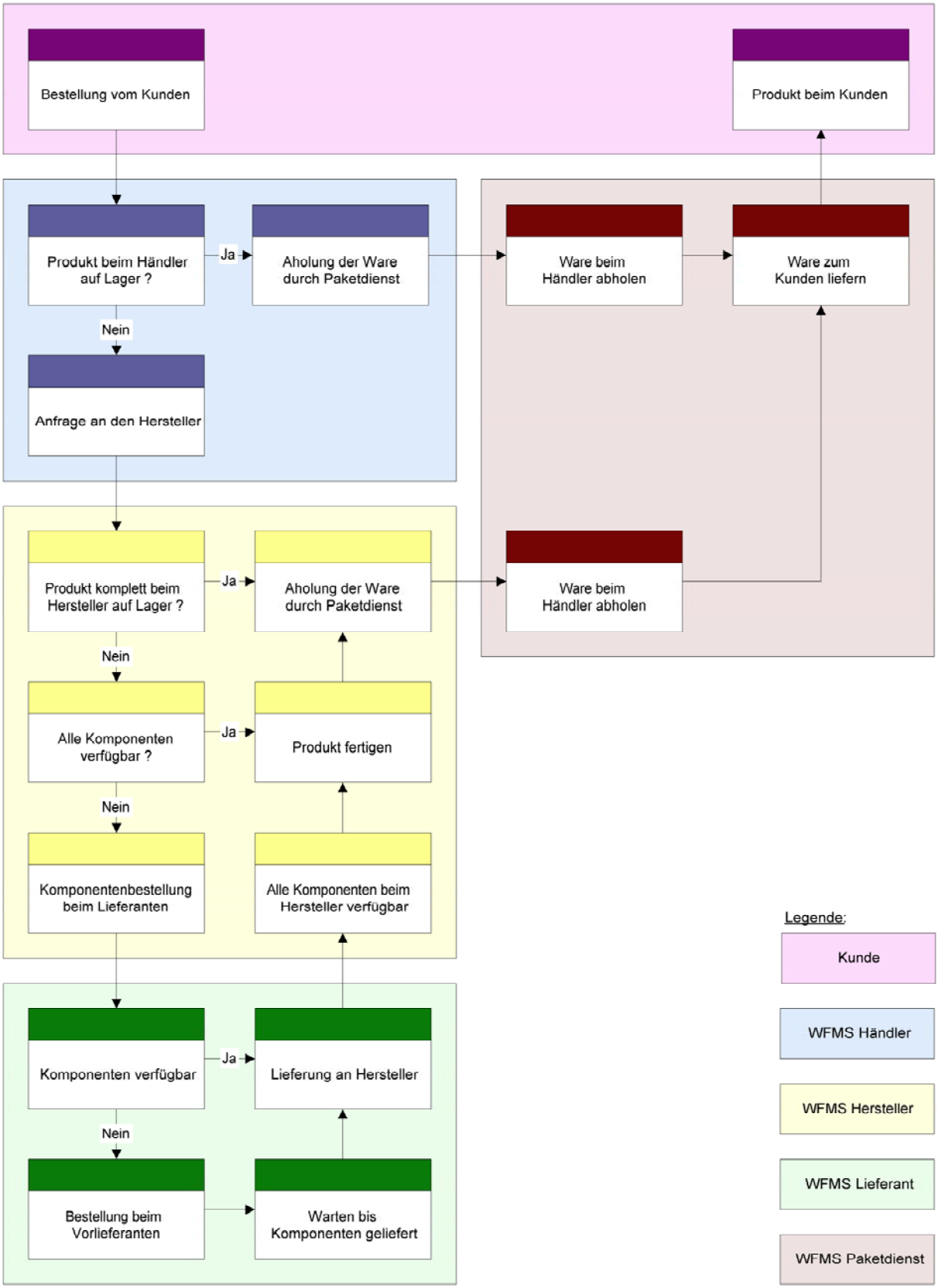


Abbildung 16: Ablaufgraph eines Bestellvorgangs

Falls dem so ist, wird wie oben verfahren, das heißt im WFMS des Paketdienstes wird automatisch ein Abholauftrag gestartet und die Zustellung der Ware zum Kunden veranlasst. Sollte das Produkt beim Hersteller noch nicht montiert worden sein, wird automatisch ein entsprechender Fertigungsauftrag generiert, welcher die Montage der einzelnen Komponenten zum fertigen Produkt veranlasst. Hier wird wieder die Verfügbarkeit der einzelnen Komponenten geprüft, die ggf. beim Lieferanten nachbestellt werden, usw.

Der Ablaufgraph dieses Bestellvorgangs eines verknüpften Verfügbarkeitsanfrage ist in Abbildung 16 dargestellt.

Somit lassen sich gerade an den Schnittstellen zwischen den Unternehmen die Reibungsverluste auf ein Minimum reduzieren, da die Prozesse automatisch gestartet werden und somit nur noch wenige menschliche Fehler (z.B. Unterlassungen, Bearbeitungsfehler) die Prozessausführungen beeinträchtigen können.

3.2.3 Statusabfrage einer Bestellung

Ein wichtiges Mittel, die Kundenzufriedenheit zu erhöhen, ist die vollkommene Transparenz zum Fortschritt eines Bestellvorgangs. Beim Einsatz von unternehmensübergreifend gekoppelten WFMS könnte sich der Kunde immer Online einen Überblick zum Stadium des Bestellvorgangs bzw. des Herstellungsprozesses der bestellten Ware verschaffen.

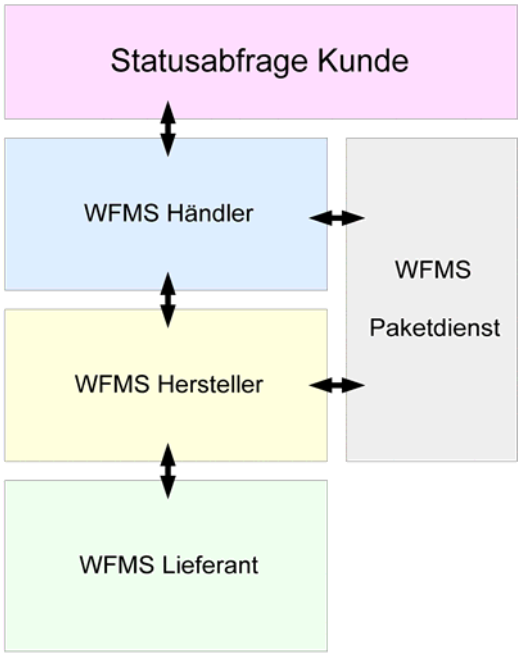


Abbildung 17: Schema der Statusabfrage einer Bestellung

In herkömmlichen Systemen hat der Kunde keine Möglichkeit, festzustellen, wie der aktuelle Status seiner Bestellung ist. Dies kann in der Regel nur „manuell“ durch einen Anruf beim Händler

geschehen, der wiederum beim Hersteller den Status der Bestellung nachfragen kann. Durch die Kopplung von verschiedenen WFMS wird es möglich, dass der Kunde den Status seiner Bestellung Online erfragen kann.

Das Schema einer solchen Status-Abfrage zeigt Abbildung 17. Hier werden, je nachdem in welchem Stadium sich die Prozessausführung befindet, zusätzlich die entsprechenden WFMS des Herstellers, des Lieferanten oder des Paketdienstes befragt und der entsprechende Ausführungs-Zustand an den Kunden zurückgeliefert.

3.2.4 Stornierung einer Bestellung

Ein weiterer Geschäftsvorfall ist die Stornierung einer Bestellung. Normalerweise storniert der Kunde seine Bestellung beim Händler, dieser muss seine Bestellung, falls noch möglich, beim Hersteller stornieren usw. Diese Stornierungen dauern recht lange, da sie jedes Mal manuell erfolgen müssen. Eine Stornierung muss ebenfalls an die entsprechenden Stellen, wie Buchhaltung und Versandabteilung, weitergeleitet werden, auch wenn die Sendung, obwohl sie unter Umständen verschickt wurde, nicht berechnet wird. Durch die Kopplung von WFMS kann die Stornierung weitgehend automatisiert werden. Storniert der Kunde die Lieferung im WFMS des Händlers, wird der entsprechende Herstellungsprozess abgebrochen, ebenso die zugehörigen Subprozesse die auf den gekoppelten WFMS ablaufen.

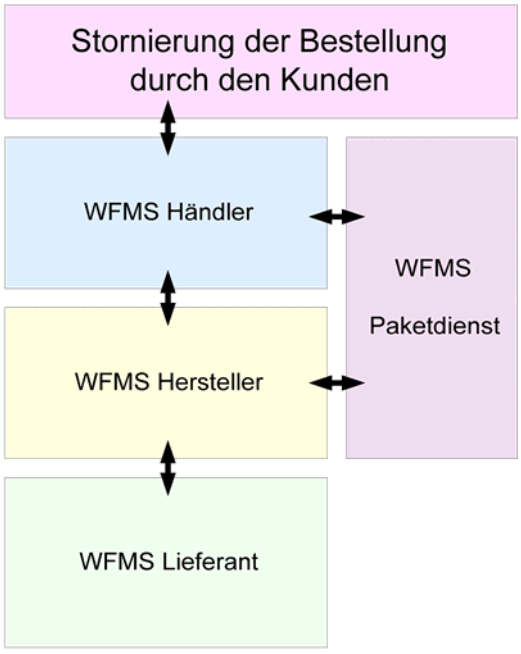


Abbildung 18: Schema der Stornierung einer Bestellung

Der Ablauf einer Stornierung ist in Abbildung 18 schematisch dargestellt. Hier werden, ähnlich wie bei der Statusabfrage, abhängig vom Ausführungszustand des Bestellvorgangs bzw. Herstellungs-

prozesses die entsprechenden verteilten Subprozesse abgebrochen und die Bestellung somit storniert. Dies bedeutet natürlich auch, dass entsprechende Mechanismen vorhanden sein müssen, um z.B. Lagerentnahmen wieder rückgängig zu machen oder bereits in Rechnung gestellte Bestellungen wieder gut schreiben zu können.

3.3 Zusammenfassung

Dieses Kapitel hat einen Überblick vermittelt wie die unternehmensübergreifende Prozesskopplung in der Praxis aussehen kann. Dies wurde am Beispiel eine Lieferkette dargestellt. Anhand der Fallbeispiele erkennt man, welche Möglichkeiten eine entsprechende Kopplung mindestens bieten muss. Dazu zählt an erster Stelle das Starten von Workflow-Instanzen auf entfernten WFMS. Zusätzlich sollten diese die Möglichkeit bieten, genaue Zusagen über die Ausführungsdauer von Arbeitsschritten zu machen. Dies wird notwendig, um die Gesamtausführungsdauer eines Workflows a priori abschätzen zu können. Des weiteren muss die Möglichkeit bestehen, immer den aktuellen Status der Ausführung einer Workflow-Instanz, auch über mehrere WFMS hinweg, abfragen zu können. Schließlich ist es auch wichtig, Prozess-Instanzen WFMS übergreifend abbrechen und die beteiligten WFMS wieder in einen konsistenten Zustand überführen zu können.

4 Die WfMC – Interoperability Abstract Specification

Um die im vorangehenden Kapitel dargestellten Fallbeispiele durchgängig elektronisch unterstützen zu können, müssen Informationen zu laufenden Prozessen unternehmensübergreifend gekoppelt werden. Die WfMC hat in einer abstrakten Spezifikation [WFMC96] aus dem Jahre 1996 Funktionalitäten beschrieben, die eine Interoperabilität zwischen verschiedenen WFMS ermöglichen sollen. Diese Spezifikation wird im Folgenden kurz vorgestellt. Auf darauf basierende Interoperabilitäts-Standards wird in Kapitel 5 eingegangen.

4.1 Definition Workflow-Interoperabilität

Bevor genauer auf die Spezifikation der WfMC eingegangen wird, erfolgt zuerst eine Definition des Begriffs Workflow-Interoperabilität. Die WfMC definiert Interoperabilität (Interoperability) folgendermaßen [WFMC99]:

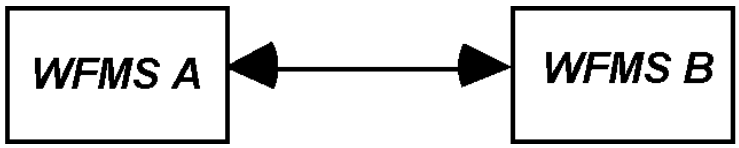
> *"The ability for two or more Workflow Engines to communicate and work together to co-ordinate work."*

WFMS sollen Prozessdefinitionen untereinander austauschen können, das heißt Prozessdefinitionen sollen auf verschiedenen Systemen ausführbar sein. Die Ergebnisse bzw. die Zustände bestimmter Prozesse werden wieder untereinander zur Verfügung gestellt, damit die beteiligten Systeme geeignet reagieren können.

4.2 Arten der Kommunikation

Interoperabilität zwischen verschiedenen Systemen kann prinzipiell auf mehrere Arten ermöglicht werden:

- ***Direkte Interaktion zwischen Workflow-Management-Systemen***
  Dabei ruft ein WFMS A ein WFMS B direkt auf (und umgekehrt).



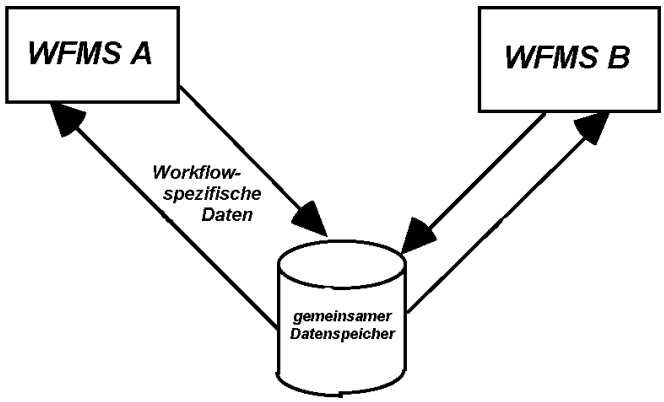
- ***Nachrichten-Austausch***
  Beim Nachrichtentausch versenden die beteiligten WFMS Nachrichten über ein Transportmedium, z.B. ein Netzwerk. Sie enthalten bestimmte Befehle bzw. Aufrufe, die dann vom Empfänger der Nachricht ausgeführt werden.

- ***Bridging***

  Beim Bridging wird ein Übersetzungsmechanismus, z.B. ein Gateway-Rechner, eingesetzt um die Befehle vom Datenaustausch-Protokoll eines WFMS A in das Protokoll eines WFMS B zu übersetzen.

- ***Gemeinsamer Datenspeicher***

  Hier wird von beiden WFMS ein gemeinsamer Datenspeicher genutzt, um Prozessdaten auszutauschen. Dieser Ansatz unterscheidet sich von den anderen dadurch, dass keine explizite Interaktion erfolgt.



4.3 Interoperabilitätsgrade

Die WfMC hat acht verschiedene Interoperabilitätsgrade (*Levels of Interoperability*) definiert, abhängig von der Architektur und der Implementierung der beteiligten WFMS [WFMC96]. Die Level bauen aufeinander auf, das heißt ein System eines bestimmten Levels muss auch die Anforderungen der niedrigeren Levels erfüllen.[14] Die Levels lassen sich folgendermaßen klassifizieren:

[14] Diese Bedingung lässt sich erst ab Level 3 erfüllen, da erst ab diesem Level Interoperabilität ermöglicht wird.

- ***Level 1: Keine Interoperabilität***
  Dieser Level charakterisiert WFMS, die keine Interaktion untereinander unterstützen und die infolgedessen auch keine Möglichkeit zur Interoperabilität bieten.

- ***Level 2: Koexistenz***
  Koexistente WFMS sind Systeme, die ebenfalls keine Möglichkeit der Interoperabilität beinhalten. Wegen der immer größeren Zahl an verschiedenen WFMS sollte aber zumindest die Möglichkeit bestehen, dass sie gemeinsam eingesetzt werden können und dabei die selbe Umgebung, wie Hardware, Netzwerktechnik oder Betriebssystem nutzen. Direkte Interaktion zwischen den WFMS ist jedoch bei dieser Interoperabilitätsstufe nicht möglich. Eine Zusammenarbeit zwischen den WFMS kann nur manuell realisiert werden, z.B. wenn ein Benutzer einen Prozess in dem WFMS B startet, nachdem der Vorgängerprozess auf einem anderen WFMS A beendet wurde.

- ***Level 3: Gateways***
  Mittels Gateways ist es verschiedenen Systemen möglich Daten untereinander auszutauschen (*Unique Gateways*). Sind mehrere Systeme beteiligt, übernimmt das Gateway auch Routing-Funktionen, um die Daten dem entsprechenden WFMS zur Verfügung zu stellen. Erfolgt der Austausch über standardisierte Gateways, wird der sogenannte Level 3a (*Common Gateway API*) erreicht.

- ***Level 4: Beschränkte Teilmenge von gemeinsamen API Aufrufen***
  Die Systeme haben eine standardisierte Schnittstelle, die eine direkte Interaktion zwischen ihnen ermöglicht. Die von diesem abstrakten Standart noch nicht festgelegte Schnittstelle soll grundlegende Funktionen beinhalten, die von möglichst allen Workflow-Produkten unterstützt werden sollen.

- ***Level 5: Komplettes Workflow API***
  Die Systeme haben eine einzige standardisierte Schnittstelle, die einen vollen Zugriff auf alle Operationen ermöglicht.

- ***Level 6: Gemeinsame Formatdefinitionen***
  Auf diesem Level benutzen die Systeme ein gemeinsames Format für Prozessdefinitionen, welches auch Routing-Aufgaben, Zugriffsrechte für Nutzer und die Verwaltung von Ressourcen beinhaltet. Die WfMC definiert dazu eine Prozessdefinitionssprache (WPDL), die für alle Systeme eine gemeinsame Grundfunktionalität beinhaltet. Des weiteren sollen die WFMS in verschiedene Klassen eingeteilt werden, abhängig von ihrem Anwendungsgebiet und den von ihnen spezifisch angebotenen Funktionen.

- ***Level 7: Protokoll-Kompatibilität***
  Bei Protokoll-kompatiblen Systemen sind alle Möglichkeiten des Datentauschs standardisiert. So ist es mit beliebigen Systemen möglich, über API-Schnittstellen zwischen Client und Server zu

kommunizieren und WF-Definitionen untereinander auszutauschen. Des weiteren sind Transaktions-Management und das Wiederherstellen eines korrekten Systemzustands nach Systemausfällen (Recovery-Mechanismen) standardisiert.

- ***Level 8: Gemeinsames Look & Feel***
  Dieser höchste Level der Interoperabilität beinhaltet alle vorhergehenden Level und die Tatsache, dass der Benutzer über die gleiche standardisierte Benutzerschnittstelle, bei allen WFMS kommuniziert. Dieser höchste Level wird sich in der Praxis aber wohl nicht durchsetzten lassen.

4.4 Interoperabilitätsmodelle

In [WFMC96] werden drei verschiedene Möglichkeiten genannt wie Interaktionen zwischen Prozess-Instanzen, die von verschiedenen WFMS kontrolliert werden, aussehen können.

4.4.1 Verkettete Prozesse

Ein Subprozess wird von einem übergeordneten Prozess einer Workflow-Engine A auf der Workflow-Engine B gestartet (siehe Abbildung 19). Sobald der Subprozess gestartet ist, spielt es keine Rolle, ob der übergeordnete Prozess terminiert oder weitergeführt wird. Der Subprozess läuft selbstständig weiter.

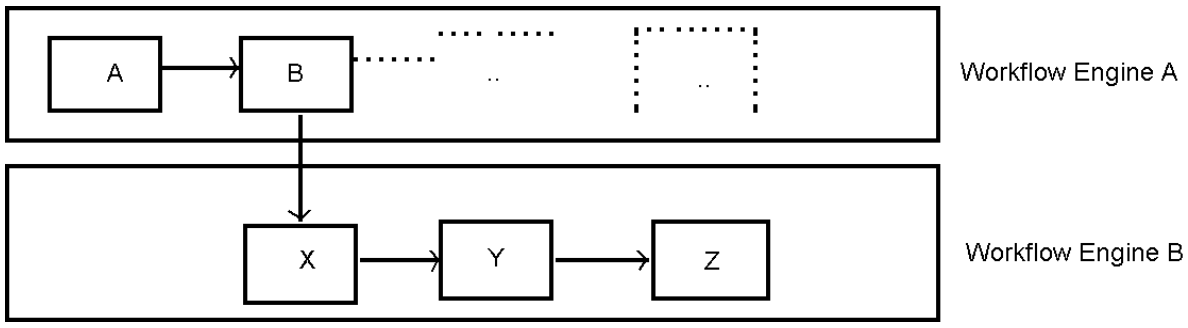


Abbildung 19: Verkettete Prozess-Ausführung[15]

4.4.2 Verschachtelte Prozesse

Der Subprozess wird von einem übergeordneten Prozess der Workflow-Engine A auf der Workflow-Engine B gestartet (siehe Abbildung 20). Anschließend werden beide Prozesse parallel ausgeführt. Der übergeordnete Prozess wartet an einer definierten Stelle mit der weiteren Ausführung, und zwar so lange bis der Subprozess beendet wird.

[15] Aus [WFMC96], Seite 13

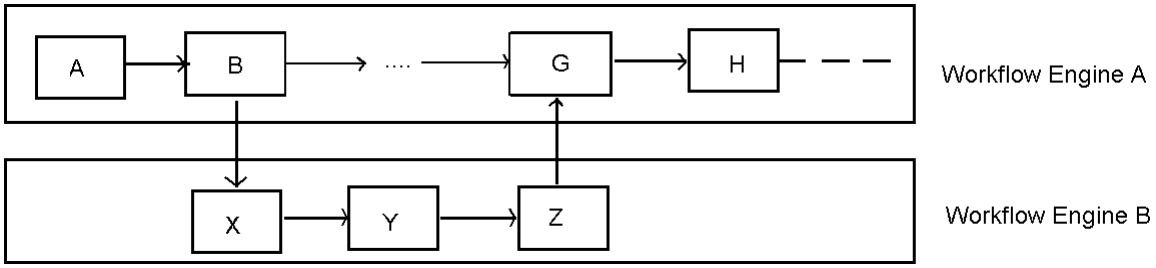


Abbildung 20: Verschachtelte Prozess-Ausführung[16]

4.4.3 Parallel synchronisierte Prozesse

Eine weitere Möglichkeit der Prozess-Ausführung ist die synchronisierte Prozess-Kopplung.

Dabei führen zwei Workflow-Engines nebenläufig zwei gleichberechtigte Prozesse aus. An einem Punkt der Ausführung wartet Workflow-Engine A mit der Ausführung auf Workflow-Engine B (oder umgekehrt). Wenn beide Workflow-Engines an diesem Punkt mit der Ausführung ihrer Prozess-Instanzen angekommen sind, werden prozess-relevante Daten zwischen ihnen ausgetauscht (siehe Abbildung 21). Anschließend fahren beide Engines mit der Ausführung der Prozesse selbständig fort.

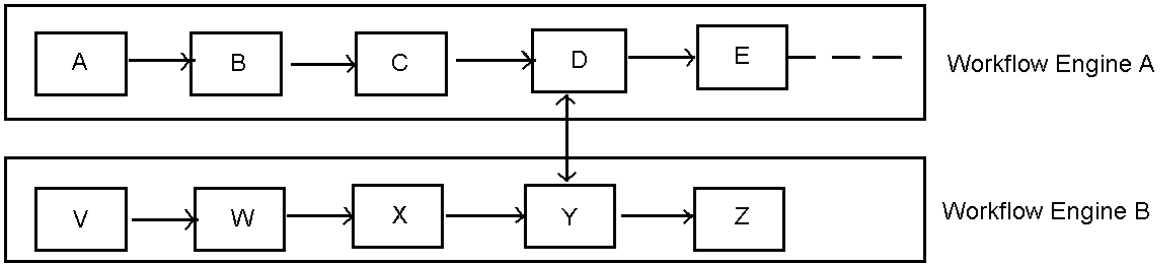


Abbildung 21: Synchronisierte Prozess-Ausführung[17]

Dieses parallele, synchronisierte Ausführungsmodell wird vorerst nicht Bestandteil zukünftiger Standards der WfMC sein. Dementsprechend ist es in existierenden Standardisierungs-Vorschlägen (vgl. Kapitel 5) nicht umgesetzt, was wiederum die breite Nutzung dieser Standards erheblich einschränkt.

4.5 Prozessdatenaustausch

Um einen Workflow auf einem entfernten WFMS starten zu können, muss diesem WFMS die Prozessdefinition bekannt sein. Hierfür gibt es mehrere Möglichkeiten. Erstens kann die komplette

[16] Aus [WFMC96], Seite 13

[17] Aus [WFMC96], Seite 13

Prozessdefinition beim Aufruf mit übergeben werden. Die Prozessvorlage muss für diesen Fall in einem einheitlichen Prozessdatenformat vorliegen, damit sie von dem gerufenen System interpretiert werden kann. Zweitens können die Prozessdefinitionen auch an einer zentralen Stelle gespeichert werden, auf die beide Systeme Zugriff haben. Hier muss die Prozessdefinition ebenfalls in einem einheitlichen Prozessdatenformat vorliegen. Als letzte Möglichkeit kann die Prozessdefinition auch beim entfernten WFMS gespeichert sein. Es muss dann nur gewährleistet sein, dass das aufrufende System Kenntnis der Aufrufparameter dieses Prozesses hat, um die für seinen Aufruf benötigten Daten richtig übergeben zu können.

4.6 Operationen der abstrakten Spezifikation

In der *Abstract Specification* der WFMC werden 16 abstrakte Operationen für die Interoperabilität zwischen WFMS definiert. Sie sollen alle Formen der WFMS-Interoperabilität abdecken. Diese Operationen werden aufgeteilt in *Request Messages und Notification Messages*.

Request Messages dienen dazu Anfragen bzw. Aufträge an andere WF-Engines zu schicken. Sie werden dann sofort mit einer *Response Message* beantwortet, in der das Ergebnis der Anfrage zurückgegeben wird. *Notification Messages* werden dazu genutzt, um während der Ausführung eines Prozesses die aufrufende Workflow-Engine über Änderungen oder das Erreichen bestimmter Ausführungspunkte zu informieren. Jede *Notification Message* wird wieder mit einer *Response Message* beantwortet. Sie zeigt an, dass die *Notification Message* erhalten und verstanden wurde.

Getrennt nach *Request Messages* und *Notification Messages* gibt es folgende Operationen:

- ***Create Process Instance***
  Diese Operation übergibt eine Prozessdefinition, die die empfangende Workflow-Engine instanzieren und ausführen soll. Ein weiteres Flag, das *return_flag,* legt fest, ob es sich bei dem Aufruf um einen verketteten oder verschachtelten Aufruf handelt (vgl. Abschnitt 4.4). Abhängig davon werden an die aufrufende Workflow-Engine Rückgabewerte geliefert oder nicht.

- ***Get Process Instance State***
  Die Anfrage `GetProcessInstanceState` liefert den Status[18] einer Prozess-Instanz zurück, die auf einer anderen Workflow-Engine ausgeführt wird. Dies ist z.B. dann wichtig, wenn Subprozesse eine relativ lange Laufzeit haben, und die aufrufende Workflow-Engine prüfen will, ob ein Subprozess noch läuft oder schon vollständig ausgeführt bzw. abgebrochen wurde.

- ***Set Process Instance Attributes***
  Prozessdefinitionen können nach der Instanzierung nicht sofort ausgeführt werden, da sie erst mit den entsprechenden Daten versorgt werden müssen. Mit dem Befehl `SetProcess-`

[18] Die WFMC definiert für Prozess-Instanzen in [WFMC99] die folgenden Zustände: `notStarted`, `running`, `suspended`, `completed`, `aborted` und `terminated`. (Vgl. [WFMC99], Seiten 169ff)

`InstanceAttributes` können Attribute einer Prozess-Instanz gesetzt werden. Dabei wird eine Liste mit Attributen und ihren Werten übergeben, die im Subprozess gesetzt werden sollen.

- ***Get Process Instance Attributes***
  Diese Anfrage liefert den Wert der angefragten Attribute der Prozess-Instanz zurück. Damit kann eine Workflow-Engine prüfen, wie weit die Ausführung einer Prozess-Instanz auf einer anderen Workflow-Engine fortgeschritten ist. Dies lässt sich z.B. realisieren, indem in einem Attribut der Ausführungsforschritt einer Prozess-Instanz mitgeschrieben wird und dieser dann mit Hilfe von `GetProcessInstanceAttributes` abgefragt wird.

- ***Start Process Instance***
  Mit Hilfe der Operation `StartProcessInstance` wird die Ausführung einer zuvor mit `CreateProcessInstance` erzeugten Prozess-Instanz gestartet.

- ***Abort Process Instance***
  Diese Nachricht bricht die Ausführung einer Workflow-Instanz auf einer entfernten Workflow-Engine ab. Es werden die Ausführung der jeweils aktiven Arbeitsschritte abgebrochen und die aktuellen Daten verworfen.

- ***Terminate Process Instance***
  Diese Nachricht beendet die Ausführung einer Workflow-Instanz auf einer entfernten Workflow-Engine. Dies ist z.B. dann nötig, wenn die Ausführung der WF-Instanz vor dem normalen Ende des Prozesses beendet werden soll. Im Gegensatz zu `Abort Process Instance` werden alle laufenden Arbeitsschritte zu Ende geführt. Wenn alle laufenden Schritte abgeschlossen sind, wird der gesamte Prozess beendet.

- ***Change Process Instance State***
  Diese Nachricht ändert den Status einer Prozess-Instanz, die auf einer entfernten Workflow-Engine ausgeführt wird (z.B. vom Status *unterbrochen* in den Status *wiederaufgenommen* oder umgekehrt). Damit lässt sich erreichen, dass die Ausführung eines Subprozesses auf einer entfernten Workflow-Engine unterbrochen und – falls nötig – wieder aufgenommen wird.

- ***List Process Instances***
  Die Operation `ListProcessInstances` liefert eine Liste von Prozess-Instanzen zurück, die bestimmten Kriterien genügen (z.B. alle Prozesse eines bestimmten Benutzers). Diese Operation wird beispielsweise von Management-Tools benötigt, um Prozesse aufzufinden und anschließend ihren Status abfragen zu können.

- ***Relinquish Process Instance***
  Diese *Request Message* bewirkt, dass der Bezug zwischen aufrufender und ausführender Workflow-Engine nicht mehr aufrechterhalten wird. Die Ausführung wird komplett an die

Workflow-Engine, die den Subprozess steuert, übergeben und es werden keine *Notification Messages* an die aufrufende Workflow-Engine mehr verschickt.

- ***Process Instance Started***
  Die *Notification Message* `ProcessInstanceStarted` benachrichtigt die aufrufende Workflow-Engine, dass mit der Ausführung der von ihr erzeugten Prozess-Instanz begonnen wurde. Dies kann wichtig sein, wenn der Start der Ausführung noch von anderen Vorbedingungen abhängig ist.
- ***Process Instance Aborted***
  Mit dieser *Notification Message* kann die ausführende Workflow-Engine die aufrufende Engine benachrichtigen, dass die Ausführung einer Prozess-Instanz abgebrochen wurde.
- ***Process Instance Terminated***
  Mit dieser *Notification Message* kann die ausführende Workflow-Engine die aufrufende Engine benachrichtigen, dass die Ausführung einer Prozess-Instanz beendet wurde.
- ***Process Instance Completed***
  Mit dieser *Notification-Message* kann die ausführende Workflow-Engine die aufrufende Engine benachrichtigen, dass deine Prozess-Instanz komplett ausgeführt und regulär beendet wurde.
- ***Process State Changed***
  Mit dieser *Notification-Message* kann die ausführende Workflow-Engine die aufrufende Engine darüber benachrichtigen, dass der Status einer Prozess-Instanz geändert wurde.
- ***Process Attributes Changed***
  Die *Notification Message* `ProcessAttributesChanged` benachrichtigt die Workflow-Engine des Eltern-Prozesses, dass sich relevante Prozess-Daten bzw. -Attribute bei der Ausführung des Subprozesses geändert haben. Dies ist z.B. dann wichtig, wenn die aufrufende Workflow-Engine über bestimmte Meilensteine der Ausführung eines Subprozesses informiert werden soll.

4.7 Zusammenfassung

Dieses Kapitel hat die abstrakte Spezifikation der WfMC vorgestellt. Sie soll als Grundlage für zukünftige Interoperabilitäts-Standards dienen. Dabei wird auf einer recht abstrakten Art und Weise spezifiziert, wie Interoperabilität zwischen verschiedenen WFMS ermöglicht werden kann.

Die Spezifikation geht nur von zwei möglichen Arten der Zusammenarbeit aus: dem verschachtelten und dem verketteten Aufruf eines entfernten WFMS. Die für die Praxis wichtigste Form der Zusammenarbeit, der parallel synchronisierte Fall, liegt außerhalb des Fokus dieser Spezifikation. Denn oftmals gibt es in verschiedenen Unternehmensbereichen bereits existierende prozessorientierte Anwendungen, welche im unternehmensweiten oder -übergreifenden Fall synchronisiert werden müssen. Dagegen bestehen im allgemeinen Fall nicht immer strenge

hierarchische Beziehungen zwischen den einzelnen im Unternehmen ablaufenden Prozessen. Gerade dies aber wird von den WfMC-Standards unterstellt. Es ist deshalb wichtig auch parallel ablaufende Prozesse verschiedener WFMS synchronisieren zu können.

Des weiteren wird von der *Abstract Specification* nicht spezifiziert wie der Datenaustausch konkret ablaufen soll. Hierzu werden nur verschiedene Möglichkeiten genannt, ohne anzugeben, welche dieser Optionen in späteren Standards umgesetzt werden soll.

Wie Interoperabilitäts-Schnittstellen implementiert werden sollen wird von der abstrakten Spezifikation nicht festgelegt. Dies soll in konkreten „Bindings" (siehe Kapitel 5) spezifiziert werden und könnte z.B. mittels E-Mail oder MAPI geschehen. Jedoch muss für die entsprechenden Transportmechanismen immer garantiert werden können, dass entsprechende Nachrichten auch ankommen bzw. veränderte oder doppelt gesendete Nachrichten als solche auch erkannt werden.

Weitergehende Abhängigkeiten zwischen Prozessen verschiedener Unternehmensbereiche oder gar verschiedener Unternehmen, z.B. zeitliche Abhängigkeiten zwischen Prozessschritten, bleiben unberücksichtigt.

Eigentlich werden von der *Abstract Specification* nur die Operationen, die Interoperabilität zwischen WFMS ermöglichen sollen, recht ausführlich diskutiert. Dabei werden z.B. Operationen für das Starten, das Beenden oder die Parameterübergabe definiert. Es gibt jedoch keine Möglichkeit, den genauen Status einer Prozessausführung, also wie weit die Ausführung auf dem entfernten WFMS fortgeschritten ist, abzufragen. Lediglich ein Metastatus ist vorgegeben. Er gibt an, ob ein Prozess läuft, beendet oder abgebrochen wurde.

Ebenso wenig sind Operationen vorgesehen, die eine dynamische Änderung des Ablaufs eines Workflow-Prozesses ermöglichen.

Die Abstrakte Spezifikation wurde 1996 veröffentlicht. Bis heute liegt sie immer noch in der Version 1.0 vor, das heißt sie wurde nicht mehr an die erweiterten Anforderungen aus der Praxis (z.B. E-Commerce, B2B) angepasst. Dies bedeutet, das mittels Standards, die auf dieser Spezifikation beruhen, nur recht einfache, hierarchische Abläufe gekoppelt werden können, ohne dass andere Abhängigkeiten Berücksichtigung finden.

5 Interoperabilitätsstandards

Dieses Kapitel gibt einen Überblick über die sich abzeichnenden bzw. vorgeschlagenen Standards, die auf den Vorgaben der *Abstract Specification* der WfMC (vgl. Kapitel 4) basieren. Zuvor wird kurz auf die generellen Vor- und Nachteile von Standards eingegangen.

5.1 Vor- und Nachteile von Interoperabilitätsstandards

Um eine Kommunikation zwischen verschiedenen Software-Systemen zu ermöglichen, müssen die übertragenen Informationen zwischen den beteiligten Parteien abgestimmt werden. Prinzipiell gibt es zwei Möglichkeiten wie diese Kommunikation realisiert werden kann: Zum einen bilateral, das heißt jeweils nur zwischen zwei beteiligten Systemen, zum anderen über eine einheitliche, für alle Systeme gültige Standard-Schnittstelle (siehe Abbildung 22b). Bei `n` verschiedenen Systemen werden bei einer bilateralen Kommunikation `n*(n-1)/2` Schnittstellen benötigt[19], bei einer standardisierten Schnittstelle sind dagegen nur `n` Schnittstellenimplementierungen vorzunehmen.

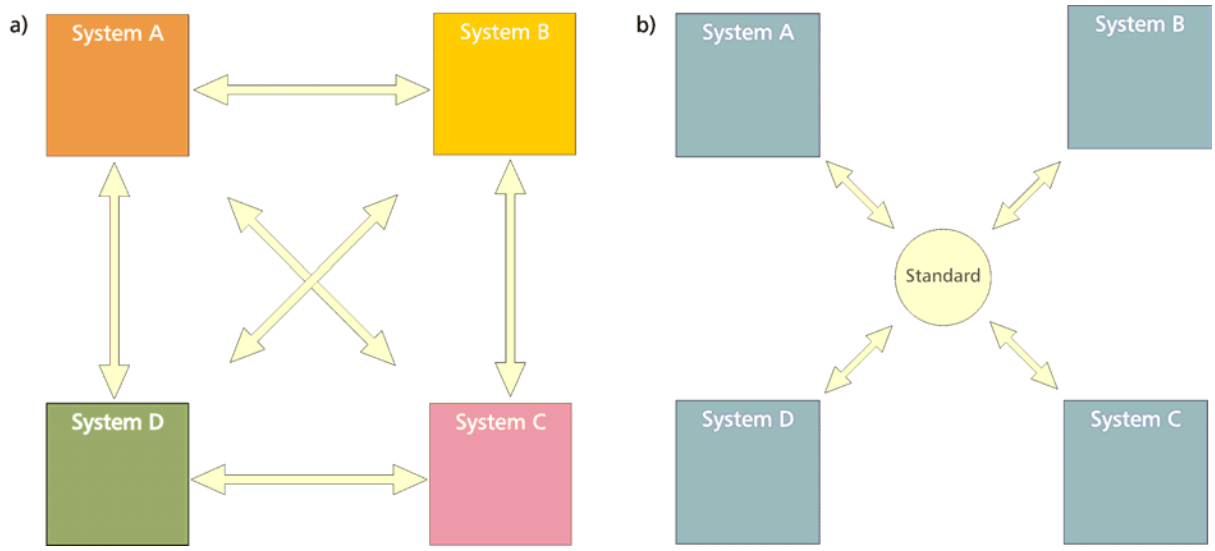


Abbildung 22: Bilaterale und normierte Schnittstellen zwischen verschiedenen Systemen

Da auch standardisierte Schnittstellen nachteilig sein können, werden im Folgenden zunächst allgemeine Vor- und Nachteile einer Standardisierung diskutiert.

[19] Im Beispiel aus Abbildung 22a werden bei 4 verschiedenen Systemen bereits 6 (=4*3/2) verschiedene Paare von Austauschformaten benötigt.

5.1.1 Vorteile

- ***Kostenreduzierung***

  Arbeiten verschiedene Systeme auf der Grundlage bilateraler Vereinbarungen zusammen, sind im Allgemeinen eine Vielzahl verschiedener Datenformate und Schnittstellen zu pflegen. Verwendet man im Gegensatz dazu eine standardisierte Schnittstelle, muss man die Schnittstellen-Implementierungen nur einmal vornehmen und kann diese dann beliebig oft nutzen. Auch der Änderungsaufwand im laufenden Betrieb ist wesentlich geringer, da er nur an einer Stelle vorgenommen werden muss, unabhängig von der Anzahl der beteiligten Systeme.

- ***Zeitersparnis***

  Standards stehen sofort zur Verfügung und erlauben somit eine schnelle Zusammenarbeit verschiedener Systeme – ohne zeitaufwendige Verabredungen der Teilnehmer über Austauschformate und Datenmodelle. Dies ist heute ein sehr wichtiger Aspekt, um die immer wieder geforderte Reduzierung der Anlaufzeiten zu realisieren.

- ***Qualitätsverbesserung***

  Normierte Schnittstellen sind aufgrund der größeren Erfahrung der Entwickler häufig qualitativ besser als eigenentwickelte Austauschformate. In der Regel werden sie von einer großen Zahl von Personen und Institutionen konzipiert, so dass Fehler frühzeitig erkannt werden können.

- ***Verringertes Investitionsrisiko***

  Durch die Verbindlichkeit von Standards verringert sich das Risiko einer Investition in ein EDV-System. Man hat die Sicherheit auch zukünftig mit wechselnden Geschäftspartnern ohne großen Aufwand zusammenarbeiten zu können, da sie die gleichen Standards benutzen. Auch ist eine Migration auf ein Nachfolgesystem idealerweise ohne Datenverluste möglich.

- ***Größere Flexibilität in der Zusammenarbeit***

  Die Vereinheitlichung der Schnittstellen reduziert die Abhängigkeit der Unternehmen untereinander und erlaubt somit die freie Auswahl konkurrierender Systeme und Partner nach rein wirtschaftlichen Gesichtspunkten. Beispielsweise ist ein Wechsel von Lieferanten bei Einhaltung standardisierter Schnittstellen einfacher möglich.

- ***Unabhängigkeit vom eigenen Personal***

  Im Gegensatz zu bilateral vereinbarten Schnittstellen oder eigenentwickelten Konvertierungsformaten lässt sich ein Datenaustausch mit Hilfe von Standards auch dann realisieren, wenn kein oder nur unzureichend qualifiziertes Personal im Unternehmen zur Verfügung steht. Durch eine Standardisierung erreichen die Unternehmen somit auch eine größere Unabhängigkeit von Einzelpersonen.

5.1.2 Nachteile

- ***Mangelnde Effizienz***

  Standards werden im Hinblick auf den breiten Anwenderkreis meist so allgemein entwickelt, dass alle möglichen Fälle abgedeckt werden können. Somit zeigen entsprechende Implementationen im Vergleich zu Eigenentwicklungen meist ein schlechteres Performance-Verhalten, welches zugunsten einer größtmöglichen Allgemeingültigkeit hingenommen wird.

- ***Vielfalt von Standards***

  Für gleiche oder ähnliche Anforderungen finden sich oft alternative Standards (auch im Workflow-Umfeld). Dadurch kann es vorkommen, dass zwei Produkte verschiedene Standards unterstützen und somit nicht miteinander kommunizieren können.

- ***Haftungsprobleme***

  Ein Standard wird meist von einer Institution verabschiedet, die aber keine Haftung für Fehler übernimmt, die durch die Nutzung dieses Standards entstehen.

- ***Sehr lange Normierungsprozesse***

  Das wahrscheinlich größte Problem der Standardisierung sind die langen Normierungsprozesse. So dauert es von den ersten Vorschlägen bis zur Verabschiedung eines Standards meist recht lange. In diesem Zeitraum entwickeln sich nicht selten proprietäre Lösungen, die ebenfalls sehr gut funktionieren und sich schwer wieder vom Markt verdrängen lassen. Das heißt die Durchsetzung des Standards gestaltet sich dann entsprechend schwierig.

5.2 Entwicklungsgeschichte der Interoperabilitätsstandards

Auf Basis der *Abstract Specification* der WfMC [WFMC96] haben sich verschiedene Vorschläge etabliert, von denen inzwischen einige als Standard verabschiedet worden sind.

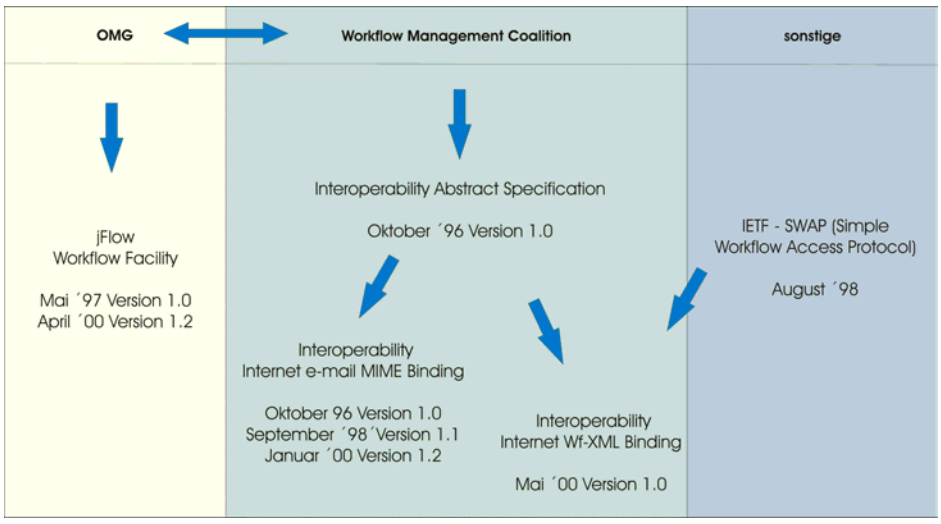


Abbildung 23: Schema der Entwicklungsgeschichte der Interoperabilitätsstandards

Die einzelnen Standards sind in Abbildung 23 in der zeitlichen Reihenfolge ihres Erscheinens dargestellt. Grundlage dieser Vorschläge ist immer die *Abstract Specification* der WfMC. Von Seiten der WfMC wurde zum einen ein Standard zur Kommunikation über E-Mail-Nachrichten (Internet E-Mail MIME Binding) [WFMC98b, WFMC00] verabschiedet, zum anderen wurde, beeinflusst durch die SWAP-Spezifikation der IETF [Swen98], die Kommunikation über XML-Nachrichten (WF-XML) standardisiert [WFMC00a]. Die OMG hat kurz nach Erscheinen des WfMC-Referenz-Modells ihre WF Facilitiy (siehe auch 2.5.2.2) [OMG99] als Standard verabschiedet.

Die einzelnen Vorschläge und Standards werden in den folgenden Abschnitten vorgestellt und bewertet.

5.3 WfMC E-Mail Binding

Der E-Mail MIME-Binding[20] Standard beruht auf der Kommunikation verschiedener WFMS mit Hilfe von MIME Nachrichten [WFMC98b]. Die erste Version (1.0) wurde kurz nach Erscheinen der *Abstract Specification* im Oktober 1996 veröffentlicht. Dieser Standard liegt inzwischen in der Version 1.2 (Stand vom Januar 2000) vor [WFMC00].

Dabei werden, im Gegensatz zu der *Abstract Specification,* nur die folgenden Operationen definiert:

- ***CHANGE PROCESS INSTANCE STATE***
- ***CREATE PROCESS INSTANCE***
- ***GET PROCESS INSTANCE ATTRIBUTES***
- ***GET PROCESS INSTANCE STATE***
- ***PROCESS INSTANCE ATTRIBUTES CHANGED***
- ***PROCESS INSTANCE STATE CHANGED***
- ***SET PROCESS INSTANCE ATTRIBUTES***
- ***START CONVERSATION***
- ***STOP CONVERSATION***

Mit Hilfe dieser Funktionen lassen sich auf einem entfernten WFMS (Sub-)Prozess-Instanzen erzeugen und (Sub-) Prozesszustände abfragen. Die Funktionsweise der einzelnen Operationen entspricht der in der *Abstract Specification* getroffenen Festlegung (vgl. Abschnitt 4.6).

Die Operationen werden in Form von MIME-Nachrichten zum entfernten WFMS übertragen und dann von diesem interpretiert.

5.3.1 Beispiel einer Konversation

Eine Beispielkonversation eines Subprozesses zeigt Abbildung 24. Man erkennt, dass pro Nachricht auch mehrere Operationen übermittelt werden können. Auf jede Nachricht wird eine Antwort-

[20] MIME steht für *Multipurpose Internet Mail Extension*

nachricht an das aufrufende WFMS zurückgeschickt, die die korrekte Ausführung der Anfrage bestätigt.

Zuerst wird vom WFMS A die Konversation aufgenommen (`StartConversation`) und beim entfernten WFMS B eine neue Prozess-Instanz erzeugt (`CreateProcessInstance`). Anschließend werden die einzelnen Parameter der neuen Prozess-Instanz übergeben (`SetProcess-InstanceAttibute`) und mit ihrer Ausführung auf dem WFMS B begonnen (`ChangeProcess-InstanceState`), indem der Status des Workflow-Prozesses auf *running* gesetzt wird. Nach beendeter Ausführung meldet das entfernte WFMS B, dass die Prozess-Ausführung beendet wurde (`ProcessInstanceStateChanged`). Dazu liefert sie den neuen Status *completed* zurück, woraufhin die Konversation vom WFMS A beendet wird (`StopConversation`).

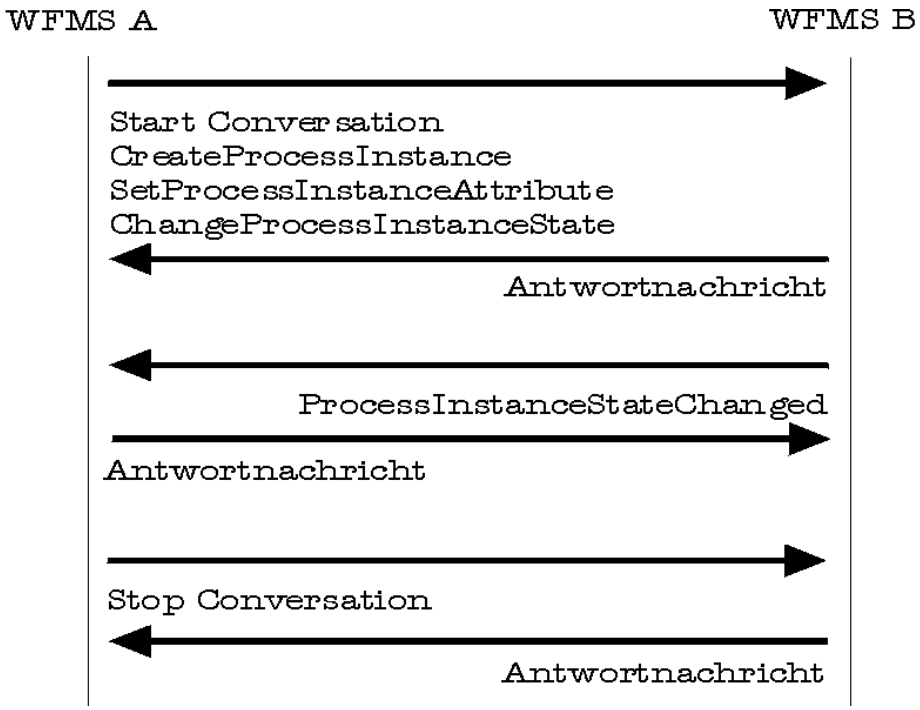


Abbildung 24: Beispielkonversation mit MIME-Nachrichten

5.3.2 Aufbau einer MIME-Nachricht

MIME-Nachrichten sind in den RFCs[21] 2045-2049 definiert. Demnach muss eine MIME-Nachricht den folgenden Aufbau haben:

[21] Request for Comment

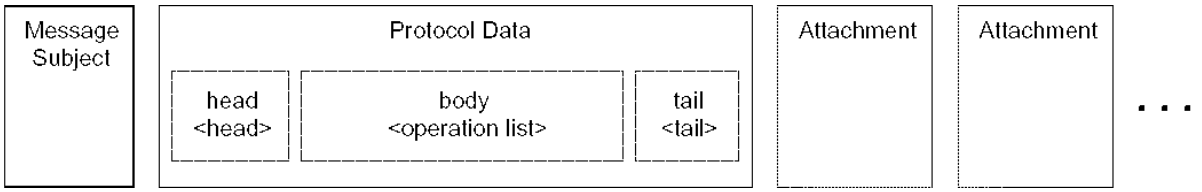


Abbildung 25: Aufbau einer MIME-Nachricht[22]

Eine MIME-Nachricht kann entweder als reine Textnachricht gesendet werden (*Content type:* `text/plain`) oder, falls Attachments (z.B. Dokumente) angehängt werden sollen, als sogenannte Multipart-Nachricht (*Content type*: `multipart/mixed`).

Das *Message Subject* enthält die Art der Nachricht (*Request*/*Response*/*Error*) sowie eine *Sequenz-* und eine *Conversation-ID*. Die *Sequenz-ID* gibt Auskunft zur Reihenfolge der gesendeten Nachrichten einer Konversation. Die *Conversation–ID* besteht aus zwei Teilen, der eindeutigen *Conversation–ID* des Quell-WFMS und der eindeutigen *Conversation–ID* des Ziel-WFMS. Bei der Initial-Nachricht einer Konversation ist die *Conversation-ID* unvollständig, da das Ziel-System erst nach Erhalt der Nachricht eine eigene ID generiert. Ein Beispiel für ein *Message Subject* ist :

```
wfmc-if4-request[12]495+AB010DFF&&
```

Dabei handelt es sich um eine *Request-Nachricht*, mit der Sequenznummer 12 und der *Conversation-ID 495* (Quell-System) bzw. *AB010DFF* (Ziel-System). Abgeschlossen wird das *Message Subject* immer mit "&&".

Der Head-Teil der Nachricht (vgl. Abbildung 25) beinhaltet nochmals das *Message-Subject* (vgl. oben). Falls Mailsysteme das *Message-Subject* bei der Übertragung der Mail abschneiden sollten, liegt also immer noch die komplette *Conversation–ID* vor, um die Nachricht korrekt zuordnen zu können. Nach dem *Message-Subject* im Head-Teil wird noch ein Zeitstempel angefügt:

```
wfmc-if4-request[0]K1234+&1998-04-25T04:25:39Z&&
```

Im Body-Teil der Nachricht (vgl. Abbildung 25) werden die einzelnen Operationen sequentiell übermittelt. Eine einzelne Konversation beginnt immer mit der Operation `StartConversation` und endet mit der Operation `EndConversation`. Wie bereits erwähnt umfasst eine Konversation üblicherweise mehrere Nachrichten, wobei eine Nachricht wiederum mehrere Operationen einschließen kann.

Den einzelnen Operationen einer Nachricht werden (nach einem "?") ihre Parameter mit übergeben, wobei der Parameter-Bezeichner und der eigentliche Parameter-Wert jeweils mit einem "=" getrennt und mit einem "&" abgeschlossen werden (vgl. Abbildung 26 und Abbildung 27). Zeilenumbrüche werden dabei ignoriert. Der End-Begrenzer vor einer neuen Operation ist "&&".

[22] Aus [WFMC00], Seite 8

Der letzte Teil der Nachricht (*Tail*) enthält noch eine Prüfsumme. Er besteht aus dem Schlüsselwort "end" sowie - in Klammern - der eigentlichen Prüfsumme. Diese wird nach dem Fletscher-Algorithmus berechnet [Flet82]. Dabei wird die komplette Nachricht angefangen mit "w" von *wfmc-request* bzw. *wfmc-response* bis zum "d" von *end* bei der Berechnung dieser Prüfsumme zugrunde gelegt.

Falls Attachments angehängt werden, folgen diese in den weiteren Abschnitten der Multipart-Nachricht. Sie werden nicht in die Berechnung der Prüfsumme mit einbezogen.

Der genaue Aufbau einer MIME-Nachricht ist in [WFMC00], in Form einer BNF-Grammatik (*Backus-Naur-Format*), definiert (siehe Anhang E).

Im Folgenden sind zwei Beispielnachrichten dargestellt, zum einen eine Multipart-Nachricht mit Anhängen, zum anderen eine reine Textnachricht. Dabei sind die einzelnen Teile der Nachricht unterschiedlich grau hinterlegt (*Message Subject, Head, Body, Tail, Attachments*).

```
From: Workflow Engine X <xyz@wfmc.org>
To: Workflow Engine Y <y@wfmc2.org>
Subject: wfmc-if4-request[0]K1234+&
MIME-Version: 1.0
Content-type: multipart/mixed; boundary="simple
boundary"
This is the preamble. It is part Zero, and it is
ignored.
--simple boundary
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&OpID=1&
ProductID=MagicWorkflow/5.0&
SourcePID=24&SourceConversationID=K1234&&
CreateProcessInstance?ProcessDefinitionID=Open Account&
Profile=chain&OpID=2&&
SetProcessInstanceAttributes?OpID=3&Number=4&
Name=copies&Type=WMTINT8&Value=3&
NAME=description&TYPE=WMTTEXT&VALUE=Two lines of text,
thi
s is the first line%0D%0AThis is the second line.&
name=file1&type=WMTAttachment&value=2&
name=file2&type=WMTATTACHMENT&value=3&&
ChangeProcessInstanceState?OpID=4&State=open.running&&
StopConversation?OpID=5&&
end(7224)
--simple boundary
Content-type: text/plain; charset=us-ascii
This is an attachment. It is attachment number two
(name=file1&).
The attachment could be any context type, but in this
case it is text/plain.
--simple boundary
Content-type: text/plain; charset=us-ascii
This is another attachment. It is attachment number
three (name=file3&).
This attachment could be any context type, but in this
case it is text/plain.
--simple boundary—
This is the epilogue. It is part four, and it is also
ignored.
```

Message Subject

Head

Body

Tail

Attachment

Attachment

Abbildung 26: Multipart MIME Nachricht[23]

[23] Aus [WFMC00], Seite 17

```
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-response[1]K1234+ABC&1998-04-25T05:00:33Z&&
StartConversation?ErrorCode=0&SourceConversationID=K1234&
TargetConversationID= ABC&TargetNodeID=abc@wfmc.org&
Version=1.1&ProductID=BetterMagicWorkflow/8.1&OpID=1&&
CreateProcessInstance?ErrorCode=0&OpID=2&
TargetPID=32&state=open.notRunning.NotStarted&&
SetProcessInstanceAttributes?ErrorCode=0&OpID=3&
Number=4&name=copies&name=description&
Name=file1&name=file2&&
ChangeProcessInstanceState?ErrorCode=0&state=open.running&OpID=
4&&
StopConversation?ErrorCode=0&OpID=5&&
end(27234)
```

Head

Message Subject

Body

Tail

Abbildung 27: Reine Text-MIME Nachricht[24]

5.3.3 Sicherheitsaspekte beim Nachrichtenaustausch

Um gewährleisten zu können, dass der Nachrichtentausch zwischen zwei WFMS korrekt funktioniert, werden verschiedene Sicherheitsmechanismen vorgeschlagen. Sie sollen sicherstellen, dass MIME-Nachrichten stets vollständig, in der richtigen Reihenfolge, unverändert und nicht doppelt zum Ziel-WFMS gelangen.

Dazu zählt zuerst die bereits erwähnte Prüfsumme, die es ermöglicht, abgeschnittene bzw. veränderte Nachrichten zu erkennen. Mit Hilfe der Sequenznummer lässt sich ebenfalls eindeutig bestimmen, ob die Nachrichten in der richtigen Reihenfolge und nicht mehrfach beim Ziel-WFMS angekommen sind. Dabei wird auf folgendes Nummerierungsschema zurückgegriffen:

Nachricht	**Sequenznummer**
Anfrage von der Quell-Workflow-Engine	Beginnt mit 0 (Start Conversation) Nächste Sequenznummer := Letzte gesendete Nummer + 4
Antwort von der Ziel-Workflow-Engine	Nächste Sequenznummer := Letzte empfangene Nummer + 1
Anfrage von der Ziel-Workflow-Engine	Beginnt mit 2 Nächste Sequenznummer := Letzte gesendete Nummer + 4
Antwort von der Quell-Workflow-Engine	Nächste Sequenznummer := Letzte empfangene Nummer + 1

Tabelle 2: Sequenznummer der MIME-Nachrichten

Jede Anfrage-Nachricht muss also eine gerade und jede Antwortnachricht eine ungerade Sequenznummer haben. Bei jeder Anfrage der Quell-Workflow-Engine ist die Sequenznummer durch vier teilbar, die Sequenznummer bei Anfragen der Ziel-Workflow-Engine lässt sich erst bei Subtraktion von 2 durch 4 teilen. Somit ist gewährleistet, dass jede Nachricht eindeutig zuzuordnen ist.

Welche Hersteller dieses MIME-Binding unterstützen, lässt sich im Anhang A anhand des Eintrags „IF4“ (Interface 4 = E-Mail Binding) in der Spalte „*Unterstütztes WfMC Interface*“ erkennen.

[24] Aus [WFMC00], Seite 18

5.4 Simple Workflow Access Protocol (SWAP)

Zwei Jahre nach dem MIME-Binding der WfMC wurde 1998, bedingt durch die Entwicklung von XML [W3C00], ein Alternativvorschlag veröffentlicht. Das *Simple Workflow Access Protocol* [Swen98] sollte ein einfaches, internetbasiertes Workflow-Protokoll etablieren, mit dessen Hilfe Workflow-Instanzen (auf entfernten WFMS) gestartet, gesteuert und überwacht werden können. Die Grundidee von SWAP ist es, mit Hilfe von XML-Nachrichten auf Basis des weit verbreiteten Hyper-Text-Transfer-Protocol (HTTP) Interoperabilität von WFMS zu ermöglichen.

Dabei wurden die folgenden Ziele in den Vordergrund gestellt [BoKa99]:

- ***Initialisierung von Prozess-Instanzen:***
  Das Generieren und Aufrufen von entfernten (Sub-)Workflow-Instanzen sowie das Übergeben der dazugehörigen Parameter-Werte (bzw. der Verweise auf benötigte Ressourcen und Daten) müssen möglich sein.
- ***Überwachung von Prozess-Instanzen***
  Hierzu gehört die Abfrage des aktuellen Prozess-Status und des Verlaufs der bisherigen Prozess-Ausführung eines Workflows.
- ***Kontrolle der Prozess-Instanzen***
  Es sollen Abfragen und Änderungen des Status von entfernten Workflow-Instanzen möglich sein. Des weiteren sollen Workflow-Instanzen vorübergehend unterbrochen, fortgesetzt und beendet werden können.
- ***Benachrichtigung bei Änderungen***
  Bei Änderungen (z.B. des Workflow-Status) müssen entsprechende Benachrichtigungen an die beteiligten Engines geschickt werden.

SWAP stellt, im Gegensatz zum MIME-Binding, sehr wenige Operationen zur Verfügung. Sie ermöglichen es aber, die oben genannten Anforderungen zu erfüllen. Als Transportprotokoll ist HTTP vorgeschrieben, die MIME-Spezifikation (vgl. Abschnitt 5.3) dagegen lässt diesen Punkt offen.

Der SWAP-Vorschlag bewog die WfMC, neben dem oben beschriebenen E-Mail-MIME-Binding, auch die Möglichkeit der Kommunikation über XML-Nachrichten [W3C00] näher zu betrachten. Daraus ergab sich das WF-XML Binding der WfMC, welches im nächsten Abschnitt besprochen wird. Da SWAP nicht als Standard verabschiedet wurde, sondern nur als Grundlage für das WF-XML Binding der WfMC diente, wird auf die einzelnen Operationen der Spezifikation an dieser Stelle nicht mehr genauer eingegangen. Diese sind in [SWAP98] nachzulesen.

5.5 WfMC WF-XML Binding

Ausgehend von dem SWAP-Vorschlag wurde von der WfMC, alternativ zu dem E-Mail-MIME-Binding, noch eine weitere Möglichkeit der Kommunikation mittels XML-Nachrichten [W3C00]

verabschiedet [WFMC00a]. Dieser Standard erweitert die Möglichkeiten seiner Vorgänger. Er basiert auf einem gut strukturierten XML-Grundprotokoll. Dieses Protokoll verwendet Nachrichtenstrukturen, die jeweils aus einem Header- und einem Datenteil bestehen. Es besteht ferner die Möglichkeit des synchronen und asynchronen Nachrichtenaustauschs. Des weiteren ist der Austausch unabhängig vom darunter liegenden Transportprotokoll. Die Grundzüge von XML werden im folgenden Abschnitt vorgestellt.

5.5.1 Grundlagen von XML

Die *eXtensible Markup Language* (XML) wurde 1996 vom *World Wide Web Consortium* (W3C) für den Austausch strukturierter Daten entwickelt [W3C00]. Es handelt sich um eine Metasprache zur Definition von Auszeichnungssprachen. Sie ist eine echte Untermenge des SGML-Standards (*Standard Generic Markup Language*). Somit können auch vorhandene SGML-Werkzeuge mit XML arbeiten. Ziel war es, eine Sprache zu entwickeln, die die Mächtigkeit von SGML besitzt, die aber leicht erlernbar ist. In den letzten Jahren hat XML stark an Bedeutung gewonnen, weil es durch die strikte Trennung von Format und Inhalt ein geeignetes Datenformat zum Austausch und zur Speicherung von Informationen darstellt.

Eine XML-Datei zur Speicherung von Adressdaten könnte z.B. folgendes Format haben:

```
<?xml version="1.0"?>
<adressbuch>
 <eintrag>
   <nachname>Mustermann</nachname>
   <vorname>Max</vorname>
   <strasse>Musterstrasse 1</strasse>
   <plz>12345</plz>
   <ort>Musterstadt</ort>
   <telefon>01234/56789</telefon>
   <email>max@mustermann.de</email>
 </eintrag>
 <eintrag>
   <nachname>Müller</nachname>
   <vorname>Franz</vorname>
   <telefon>0731/124567</telefon>
   <email>Franz@mueller.de</email>
   <email>fm7@student.uni-ulm.de</email>
 </eintrag>
 <eintrag>
   <nachname>Umann</nachname>
   <vorname>Silke</vorname>
   <telefon>08382/98491</telefon>
 </eintrag>
</adressbuch>
```

Hierbei handelt es sich um ein *wohlgeformtes* XML-Dokument (*well-formed document*). Dies liegt immer dann vor, wenn es in seiner Syntax den W3C-Spezifikationen von XML 1.0 entspricht.

Ein wesentliches Kriterium beim Austausch von Daten ist die Einhaltung eines definierten Formats. Für XML wurde eine spezielle Sprache zur Beschreibung solcher Formate entwickelt, die sogenannte *Document Type Definition* (DTD). Sie definiert den strukturellen Aufbau und die logischen Elemente

einer Klasse von Dokumenten (Dokumenttypen). Entspricht ein Dokument nach seinem Aufbau und Struktur einer DTD, so handelt es sich um ein *gültiges* (*valid*) Dokument.

Für das obige Adressbuch-Beispiel ergibt sich folgende DTD:

```
<?xml version="1.0"?>
<!ELEMENT adressbuch (eintrag)+>
<!ELEMENT eintrag (nachname,vorname,strasse?,plz?,ort?,telefon+,email*)>
<!ELEMENT nachname (#PCDATA)>
<!ELEMENT vorname (#PCDATA)>
<!ELEMENT strasse (#PCDATA)>
<!ELEMENT plz (#PCDATA)>
<!ELEMENT ort (#PCDATA)>
<!ELEMENT telefon (#PCDATA)>
<!ELEMENT email (#PCDATA)>
```

Das Adressbuch besteht aus mehreren Einträgen, wobei jeder Eintrag eine Reihe von weiteren Elementen enthält. Manche davon sind zwingend vorgeschrieben, andere können keinmal, einmal oder mehrmals vorkommen (vgl. Tabelle 3).

Die Elemente wie Name oder Vorname bestehen aus beliebigem Text (*Parsed Character Data,* PCDATA) und dürfen keine weiteren Unterelemente enthalten.

Operator	Bedeutung
?	Element muss einmal oder keinmal vorhanden sein
+	Element muss einmal oder mehrmals vorhanden sein
*	Element muss keinmal, einmal oder mehrmals vorhanden sein
k.A.	Element muss einmal vorhanden sein
,	Sequenz (gefolgt von)
|	Alternative (oder)

Tabelle 3: Wiederholungs- und Verkettungsoperatoren der XML-Spezifikation

Mittels XML-Parsern lassen sich XML-Dokumente auf ihre Konformität zu einer bestimmten DTD prüfen. Allerdings lassen sich mit DTDs Dokumente nur oberflächlich spezifizieren. Die genaue Angabe eines Datenformats (bzw. Datentyps) für bestimmte Elemente (z.B. PLZ oder E-Mail-Adresse) ist nicht möglich.

XML-Schema beseitigt diese Mängel und bietet wesentlich mächtigere Konstrukte zur Spezifikation von Struktur, Inhalt und Semantik von XML Dokumenten. Neben der Nutzung der eingebauten Datentypen wie *string*, *real* oder *bool*, ermöglichen sie die Definition eigener Datentypen. Ein weiterer Vorteil der XML-Schemata ist, dass sie selbst in XML formuliert sind, so dass kein spezieller Parser wie im Fall von DTDs erforderlich ist [W3C01a,W3C01b,W3C01c].

Die Einfachheit von XML einerseits und die Möglichkeit der Beschreibung komplexer Daten andererseits machen XML zu einem geeigneten Format zur Speicherung und zum Austausch unterschiedlicher Daten in heterogenen Umgebungen. Daher ist die Bedeutung von XML in den letzten Jahren enorm gewachsen und es entstehen zahlreiche XML-Applikationen, wie z.B. die *Extensible Hyper Text Markup Language* (XHTML) [W3C00b] oder das *Simple Object Access Protocol* (SOAP) [W3C00a].

5.5.2 WF-XML

5.5.2.1 Basismodell

Wie in SWAP beruht der Vorschlag der WF-XML Spezifikation auf folgendem Modell:

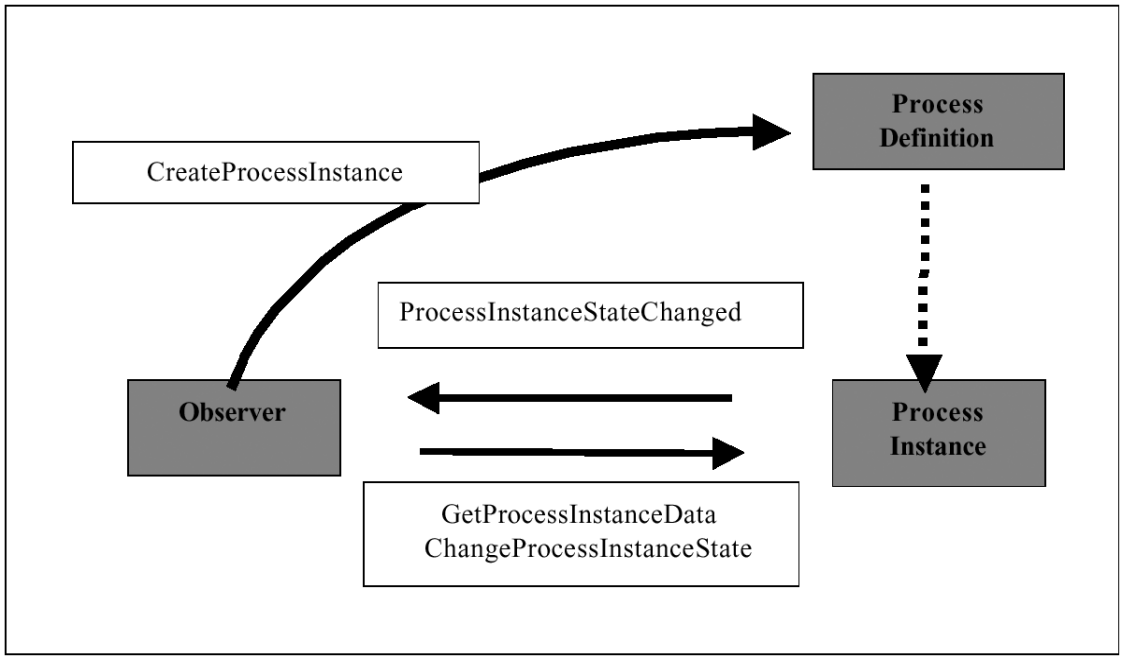


Abbildung 28: Prozessmodell der WF-XML Spezifikation[25]

Es besteht aus drei wichtigen Elementen, der Prozessdefinition, der Prozess-Instanz sowie dem Observer. Die Prozessdefinition (*Process Definition*) stellt dabei die Funktionalität zum Erzeugen von Prozess-Instanzen zur Verfügung. Eine Prozess-Instanz (*Process Instance*) entspricht der konkreten Ausführung einer bestimmten Prozess-Definition. Sie ist über eine Prozess-ID, die eindeutig ist und somit nur einmal vorkommt, ansprechbar. Bei der Erzeugung einer Prozess-Instanz wird diese automatisch gestartet. Der Observer ist ein Beobachtungsobjekt. Er ermöglicht es einer Prozess-Instanz Statusänderungen (z.B. die Beendigung der Prozess-Instanz) an den Observer zu kommunizieren. Dies ist z.B. dann von Bedeutung, wenn die aufrufende Workflow-Engine mit der Ausführung eines Prozesses auf das Terminieren eines Subprozesses auf einer entfernten Workflow-Engine wartet. In diesem Fall wird die aufrufende WF-Engine als Observer definiert, der bei Beendigung des Subprozesses entsprechend benachrichtigt wird.

5.5.2.2 Aufbau einer XML-Nachricht

Der Aufbau einer solchen XML-Nachricht ist folgendermaßen definiert:

---

[25] Aus [WFMC00a], Seite 9

```
<?xml version="1.0"?>
<WfMessage Version="1.0">
  <WfTransport/>
  <WfMessageHeader>
    ...
  </WfMessageHeader>
  <WfMessageBody>
    ...
  </WfMessageBody>
</WfMessage>
```

Jede XML-Nachricht ist ein XML 1.0 Dokument, bestehend aus drei Teilen, dem *WfTransport*-Element, dem *WfHeader*-Element und dem *WfBody*-Element. *WfTransport* dient ausschließlich für Transportprotokoll-spezifische Zwecke, um den Zusammenhang einzelner Nachrichten erkennen zu können (z.B. bei asynchroner Kommunikation). Es beinhaltet benutzerspezifische Daten, die unverändert in der Antwort-Nachricht wieder zurückgesendet werden. Anhand dieser Daten kann der Zusammenhang von Nachrichten von der Workflow-Engine erkannt werden.

Der Header enthält entweder ein *Request*- oder ein *Response-Element*, je nachdem ob es sich bei der Nachricht um einen Aufruf oder eine Antwort handelt. Ein *Request-Element* beinhaltet zusätzlich noch ein Feld, das angibt, ob eine Antwort erwartet wird oder nicht. Hier gibt es die drei Möglichkeiten „ja", „nein" oder „nur im Fehlerfall". Des weiteren gehört zu dem Header ein Key-Element, das die URL des Ziel-WFMS der aufgerufenen Prozess-Instanz speichert. Bei der Operation `CreateProcessInstance` enthält das Key-Element die Prozess-ID der Prozess-Vorlage, die instanziert werden soll.

5.5.2.3 Unterstützte Operationen

Im Body der XML-Nachricht werden die eigentlichen Operationen [Haye00] und die Daten zu ihrer Ausführung übermittelt (vgl. Abbildung 28). Im Gegensatz zum MIME-Binding kann allerdings immer nur eine Operation pro Nachricht versendet werden. Im einzelnen werden die folgenden Operationen unterstützt:

- ***Create Process Instance***

  Mit Hilfe der Operation `CreateProcessInstance` wird eine bekannte Prozess-Definition instanziert. Die Prozess-Instanz wird erzeugt, mit ihren Parameterdaten verknüpft und anschließend ausgeführt. In Anlehnung an das Beispiel aus Abschnitt 2.6 könnte der Produktionsauftrag zur Herstellung eines Notebooks mit den Ausstattungsmerkmalen (Typ=Notebook;Serie=Inspirion;Nummer=7500;Ausstattung=DVD-Laufwerk) wie folgt aussehen:

```
<?xml version="1.0"?>
<WfMessage Version="1.0">
 <WfTransport/>
 <WfMessageHeader>
   <Request ResponseRequired ="Yes">
   </Request>
   <Key>http://www.xyz.com/Wfengine?id=1199827
   </Key>
 </WfMessageHeader>
 <WfMessageBody>
   <CreateProcessInstance.Request StartImmediately="true">
     <ObserverKey>http://www.Acme.com/wfx456
```

```
      </ObserverKey>
      <ContextData>
        <Computer>
        <Type>notebook</Type>
        <Series>Inspiron</Series>
        <Number>7500<Number>
        <Option>DVD</Option>
        </Computer>
      </ContextData>
    </CreateProcessInstance.Request>
  </WfMessageBody>
</WfMessage>
```

Hier wird auch der Observer-Key mit übergeben. Er entspricht der URL des Observers, der bei Statusänderungen bzw. bei Beendigung des Prozesses informiert werden soll.

Die Antwort auf diese Nachricht (*Response Required*) kann zwei mögliche Ausprägungen haben: Es ist ein Fehler aufgetreten oder aber der Prozess konnte erfolgreich gestartet werden.

Im Fehlerfall wird eine Exception-Nachricht zurückgesendet, die Aufschluss über die Art der eingetretenen Ausnahme gibt. Ist beispielsweise die in der obigen Request-Nachricht angegebene Prozess-Definition auf dem Ziel-WFMS unbekannt, wird der Fehler 502[26] erzeugt. Dies führt zu folgender Response-Nachricht:

```
<?xml version="1.0"?>
<WfMessage Version="1.0">
  <WfTransport/>
  <WfMessageHeader>
    <Response/>
    <Key>http://www.xyz.com/WfEngine?id=1199827
    </Key>
  </WfMessageHeader>
  <WfMessageBody>
    <CreateProcessInstance.Response>
      <Exception>
        <MainCode>502</MainCode>
        <Type>F</Type>
        <Subject>Invalid Process Definition</Subject>
        <Description>Cannot create instance
        </Description>
      </Exception>
    </CreateProcessInstance.Response>
  </WfMessageBody>
 </WfMessage>
```

War dagegen der Start der Prozess-Instanz erfolgreich, wird eine Antwort-Nachricht zurückgeschickt, die den Key der Prozess-Instanz enthält. Er ist der Prozess-Instanz während ihrer kompletten Ausführungsdauer eindeutig zugeordnet. Eine solche Antwortnachricht besitzt folgende Gestalt:

```
<?xml version="1.0"?>
<WfMessage Version="1.0">
  <WfTransport/>
  <WfMessageHeader>
    <Response/>
    <Key>http://www.xyz.com/Wfengine?id=1199827
    </Key>
  </WfMessageHeader>
  <WfMessageBody>
    <CreateProcessInstance.Response>
```

[26] siehe [WFMC00a], Seite 20 bzw. Anhang D *(Exception Codes)*

```
    <ProcessInstanceKey>
      http://www.dell.com/WfcXML1199
    </ProcessInstanceKey>
    </CreateProcessInstance.Response>
  </WfMessageBody>
</WfMessage>
```

In diesem Beispiel hat der Herstellungsprozess des Notebooks den *ProcessInstanceKey* „http://www.xyz.com/WfcXML1199“, über welchen er für zukünftige Operationen zugreifbar ist.

- ***Get Process Instance Data***

  Die Operation `GetProcessInstanceData` ermöglicht es, den jeweils aktuellen Wert von Variablen einer Prozess-Instanz abzufragen. Wird z.B. der Fortschritt der Fertigung eines Computers in einer bestimmten Variablen mitprotokolliert, kann er mittels der folgenden Nachricht abgefragt werden:

```
<?xml version="1.0"?>
<WfMessage Version="1.0">
  <WfTransport/>
  <WfMessageHeader>
    <Request ResponseRequired ="Yes">
    </Request>
    <Key>http://www.xyz.com/WfcXML1199</Key>
  </WfMessageHeader>
  <WfMessageBody>
    <GetProcessInstanceData.Request>
      <ResultDataAttributes>
        <Fertigungsstatus/>
      </ResultDataAttributes>
    </GetProcessInstanceData.Request>
  </WfMessageBody>
</WfMessage>
```

Die Antwortnachricht hat in diesem Fall den folgenden Aufbau.

```
<?xml version="1.0"?>
<WfMessage Version="1.0">
  <WfTransport/>
  <WfMessageHeader>
    <Response/>
    <Key> http://www.xyz.com/WfcXML1199</Key>
  </WfMessageHeader>
  <WfMessageBody>
    <GetProcessInstanceData.Response>
      <Fertigungsstatus>Endkontrolle</Fertigungsstatus>
    </GetProcessInstanceData.Response>
   </WfMessageBody>
</WfMessage>
```

Die Antwortnachricht besagt, dass sich der Prozess momentan im Stadium der Endkontrolle befindet, wie man am Wert des Elements <`Fertigungsstatus`> erkennt.

- ***Change Process Instance State***

  Um den Status einer Prozess-Instanz zu ändern, wird die Operation `ChangeProcess-InstanceState` benötigt. Eine solche Statusänderung wird z.B. erforderlich, wenn die Ausführung eines Prozesses abgebrochen werden soll (z.B. für den Fall, dass der Kunde ein anderes Produkt bestellen will). In einem solchen Fall muss eine laufende Prozess-Instanz vom

Zustand *running* in den Zustand *terminated* bzw. *aborted* gesetzt werden können. Mittels folgender Nachricht lässt sich z.B. der oben gestartete Prozess abbrechen:

```
<?xml version="1.0"?>
<WfMessage Version="1.0">
 <WfTransport/>
 <WfMessageHeader>
   <Request ResponseRequired ="Yes">
   </Request>
   <Key>http://www.xyz.com/WfcXML1199</Key>
 </WfMessageHeader>
 <WfMessageBody>
   <ChangeProcessInstanceState.Request>
     <State>closed.abnormalCompleted.terminated
     </State>
   </ChangeProcessInstanceState.Request>
 </WfMessageBody>
</WfMessage>
```

Als Antwort liefert das System die erfolgreiche Beendigung des Prozesses zurück.

```
<?xml version="1.0"?>
 <WfMessage Version="1.0">
 <WfTransport/>
 <WfMessageHeader>
   <Response/>
   <Key>http://www.dell.com/WfcXML1199</Key>
 </WfMessageHeader>
 <WfMessageBody>
   <ChangeProcessInstanceState.Response>
     <State>closed.abnormalCompleted.terminated
     </State>
   </ChangeProcessInstanceState.Response>
 </WfMessageBody>
</WfMessage>
```

- ***Process Instance State Changed***

  Die Operation `ProcessInstanceStateChanged` benachrichtigt den Observer, der bei Start eines Prozesses angegeben wurde, über Statusänderungen des Prozesses. In unserem Beispiel wird der Observer z.B. beim Abbruch des Prozesses entsprechend informiert. Dies ist mittels folgender XML-Nachricht möglich:

```
<?xml version="1.0"?>
 <WfMessage Version="1.0">
 <WfTransport/>
 <WfMessageHeader>
   <Request ResponseRequired ="No">
   <Request/>
   <Key>http://www.Acme.com/wfx456</Key>
 </WfMessageHeader>
 <WfMessageBody>
   <ProcessInstanceStateChanged.Request>
     <ProcessInstanceKey>
     http://www.xyz.com/WfcXML1199
     </ProcessInstanceKey>
     <State>closed.abnormalCompleted.terminated
     </State>
     <ResultData>
       <Order>ACM00456</Order>
       <Account>ACM-400-2460</Account>
       <Amount>150.00</Amount>
     </ResultData>
     <LastModified>1999-12-25T15:10:35Z
     </LastModified>
   </ProcessInstanceStateChanged.Request>
 </WfMessageBody>
</WfMessage>
```

Der gegebene Überblick zu den elementaren Operationen der WF-XML Spezifikation soll an dieser Stelle ausreichen. Als Erweiterung dazu sind Operationen vorgesehen, die es ermöglichen, Prozess-Instanzen eines entfernten WFMS nach bestimmten Kriterien aufzulisten (`ListInstances`). Darüber hinaus besteht die Möglichkeit, bestimmte Prozessesdaten zu ändern (`SetData`).

Als sehr wichtige Erweiterung ist die Operation `QueryInterface` anzusehen. Sie ermöglicht es, im Rahmen der definierten Sicherheitsbeschränkungen, von unbekannten Prozessdefinitionen die Aufrufparameter abzufragen. Insbesondere können die Aufrufparameter einer Prozess-Vorlage noch zur Laufzeit von einem entfernten System abgefragt werden. Dadurch kann der entsprechende Aufruf dynamisch generiert werden, ähnlich wie man es von verschiedenen Komponentenmodellen (z.B. CORBA, Enterprise Java Beans) her kennt.

Alle für die Zukunft möglichen Erweiterungen des WF-XML Standards sind in [WFMC00a] im Anhang B nachzulesen.

5.5.2.4 Übertragungsprotokoll

Die WF-XML-Spezifikation macht keine Angaben über das Transportmedium, schlägt aber der Einfachheit halber das HTTP-Protokoll vor. Jedoch können z.B. auch E-Mail oder CORBA genutzt werden. Auch Aspekte, die die Datensicherheit betreffen, sind außerhalb des Fokus dieser Spezifikation. Hier bietet sich bei der Verbindung über HTTP z.B. die Nutzung von *Secure Sockets Layer* (SSL) an. Auch die im Message-Body übergebenen Werte sind nicht fix definiert, da sie abhängig von den jeweiligen Anforderungen der Prozesse sind. All diese Punkte sollen durch einen sogenannten *Interoperability Contract*, also einen Vertrag zwischen den Herstellern deren WFMS zusammenarbeiten sollen, geklärt werden und als bilaterale Vereinbarung zwischen den beiden Parteien als Basis der Zusammenarbeit dienen.

5.6 OMG jFlow Workflow Facility

Die OMG hat in Zusammenarbeit mit der WfMC mit jFlow einen weiteren Standard für WFMS, basierend auf der OMA-Architektur (vgl. Abschnitt 2.5.2.1), verabschiedet. Dieser Abschnitt stellt kurz die Objekte vor, die für die Interoperabilität im jFlow -Standard von Bedeutung sind.

OMG jFlow	WfMC WF-XML
WfProcessMgr	**ProcessDefinition**
WfProcess	**ProcessInstance**
WfReqester	**Oberserver**

Tabelle 4: Vergleich OMG Objektmodell mit WfMC WF-XML

Das Objektmodell der jFlow-Spezifikation hat einen ähnlichen Aufbau wie die WF-XML-Spezifikation (siehe Tabelle 4). Dabei entspricht das *ProcessDefinition*-Objekt der WF-XML-Spezifikation dem *WfProcessMgr*-Objekt der *jFlow-Spezifikation*. Eine Prozess-Instanz (*ProcessInstance*) entspricht in der *jFlow-Spezifikation* einem *WfProcess*-Objekt. Um eine Prozess-Instanz zu erzeugen, wird die

Methode `create_process` des *WfProcessMgr*-Objekts aufgerufen. Dadurch wird eine Prozess-Instanz, ein *WfProcess*-Objekt, erzeugt, welche mittels der Methoden `start` und `set_context` gestartet und mit den entsprechenden Parametern versorgt wird. Die Operation `GetProcessInstanceState` der WF-XML-Spezifikation entspricht der Methode `change_state` des *WfProcess*-Objekts. Um Parameter einer laufenden Prozess-Instanz abfragen zu können, ist in der jFlow Spezifikation die *WfProcess* Methode `get_result` definiert. Sie entspricht der Operation `GetProcessInstanceData` des WF-XML-Bindings. Das jFlow *WfRequester*-Objekt, welches die Prozessausführung überwacht, entspricht dem Wf-XML *WfObserver*.

Zusätzlich definiert die jFlow-Spezifikation noch die Objekte *WfActivity*, *WfResource* und *WfAssignment* (vgl. dazu das Objektmodell von jFlow, Abschnitt 2.5.2.2). Da CORBA-kompatible Systeme aufgrund ihrer Architektur nur untereinander kommunizieren, wird auf diesen Ansatz nicht detaillierter eingegangen.

5.7 Zusammenfassung

In diesem Kapitel wurde ein Überblick zu allen bisher im Workflow-Bereich veröffentlichten Interoperabilitätsstandards gegeben. Sie basieren allesamt auf der *Abtract Specification* der WFMC (siehe Kapitel 4).

Als wichtigster Vorschlag existiert, neben MIME-Binding und jFlow-Spezifikation, das WF-XML-Binding. Es beruht auf XML-Nachrichtentausch und definiert einen Satz an Operationen, die für verschiedene WFMS eine Interoperabilität zwischen unter- und übergeordneten Workflows ermöglicht. Trotz Standardisierung dieser Operationen und der zugrunde liegenden XML-Nachricht ist aber immer noch ein bilateraler Interoperabilitätsvertrag zwischen je zwei WFMS-Herstellern nötig. Er regelt z.B. das Übertragungsprotokoll und Sicherheitsaspekte.

Die Nichtstandardisierung des Übertragungsprotokolls bzw. der dadurch entstehende Abstimmungsaufwand zwischen Herstellern erschwert die Implementierung des WF-XML-Standards erheblich, insbesondere für kleinere Hersteller. Deshalb sind präzise Standards nötig. Es gibt zwar Bestrebungen seitens der WFMC, ein auf HTTP basierendes Binding [WFMC00b] zu verabschieden, dies liegt aber erst als Einreichung vor.

Wie schon bei der *Abstract Specification* der WFMC angemerkt, beruht das zugrunde liegende Interoperabilitäts-Modell auf einer sehr hierarchischen und starren Sichtweise. Parallele synchronisierte Ausführung, Zeitabhängigkeiten zwischen verschiedenen Workflows oder dynamische Änderungen und ihre Auswirkungen auf andere abhängige Workflows werden aktuell nicht betrachtet.

Auch Semantikaspekte bleiben bei allen bisherigen Bestrebungen vollständig ausgeklammert. Es gibt z.B. keine Möglichkeit der Definition bzw. Verwendung einer normierten Fachterminologie für

Branchenanwendungen (wie z.B. bei CORBA med/finance). Dies schränkt jedoch die Interoperabilitätsmöglichkeiten von vornherein ein.

Im Kapitel 7 wird noch genauer auf Aspekte und Anforderungen eingegangen, die man an zukünftige WFMS stellt.

6 Interoperabilitätsschnittstelle in einem bestehenden WFMS

Nachdem gezeigt wurde welche Standards es im Bereich Interoperabilität gibt, soll nun anhand eines bestehenden Systems geprüft werden, ob diese Standards von den Softwareherstellern auch angenommen werden. Dies wird am Beispiel von IBM MQ Series Workflow [IBM99] durchgeführt. IBM ist aktives Mitglied der WfMC und sollte somit die Bestrebungen der WfMC unterstützen. Auf andere WFMS-Vertreter (z.B. Staffware) wird anschließend kurz eingegangen.

6.1 IBM MQ Series Workflow

MQ Series Workflow [IBM99], das Workflow-Management-System von IBM, baut größtenteils auf den Konzepten von IBM Flowmark 2.3 [IBM96] auf. Es wurde nach der Eingliederung von Flowmark in die MQ Series Reihe [IBM01] in MQ Series Workflow umbenannt und liegt derzeit in der Version 3.2 vor. Zur Kommunikation der einzelnen Komponenten untereinander ist das Transaktionssystem MQ Series [IBM01a] im Lieferumfang enthalten. Es garantiert eine verlässliche Kommunikation der einzelnen Komponente. Inzwischen wurde die Version 3.3 von MQ Series Workflow angekündigt, die sich hauptsächlich durch eine bessere Web-Integration auszeichnen wird [IBM01b].

6.1.1 Software-Architektur von MQ Series Workflow

Bei MQ Series Workflow sind Buildtime und Runtime voneinander getrennt [IBM99a]. Die Buildtime dient der Modellierung von Arbeitsabläufen und der Definition von Systemressourcen. Dazu steht ein grafischer Editor für die Erstellung von Prozessmodellen und die Organisations-Modellierung zur Verfügung. Des weiteren lassen sich mit Hilfe der Buildtime externe Programme, durch Zuordnung zu einzelnen Aktivitäten, in die Arbeitsabläufe einbinden. Zu diesem Zweck müssen für die verschiedenen Applikationen geeignete Wrapper (die auch für die Versorgung bzw. Übernahme von Parameterdaten zuständig sind) bereitgestellt werden.

Vorhandene Workflow-Modelle, die im *Flow-Definition-Language*-Format (FDL) vorliegen, lassen sich in die Buildtime importieren und weiterbearbeiten. Exportmöglichkeiten bestehen ebenfalls wieder in das FDL-Format sowie in das HTML-Format zur Dokumentation der modellierten Prozesse.

Die modellierten Arbeitsabläufe werden anschließend im FDL-Format in die Runtime importiert, wo sie als Workflow-Schema gespeichert und von den Serverkomponenten verwaltet werden.

Die Runtime von IBM MQ Series WF dient zur Ausführung und Überwachung der mit Hilfe der Buildtime definierten Workflow-Modelle. Dabei ist MQ Series Workflow als 3-Schichten Modell konzipiert [IBM99]. Diese Schichten sind im Folgenden schematisch dargestellt:

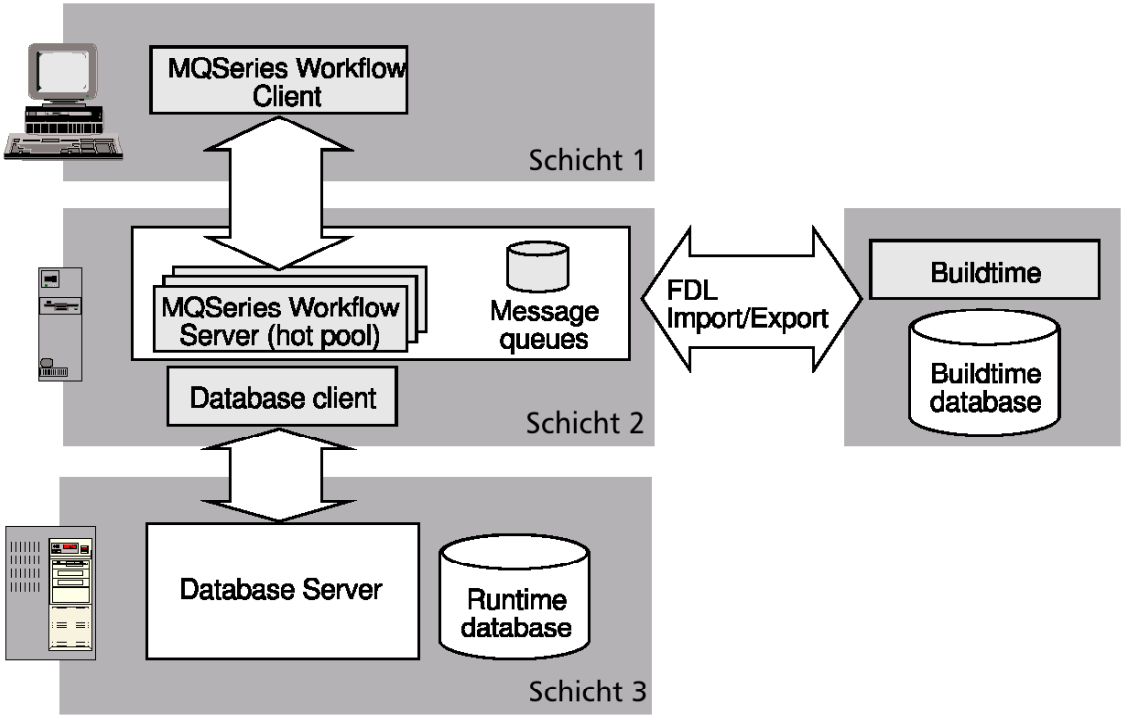


Abbildung 29: 3-Schichten Modell von MQ Series Workflow[27]

- ***Schicht 1: Client-Komponenten***

  Die Schicht 1 repräsentiert die Client-API von MQ Series Workflow sowie die Clients, die diese API nutzen. Clients sind verantwortlich für die Programmausführung externer Applikationen sowie für Benutzer-Interaktionen (z.B. Start einer Aktivität aus einer Arbeitsliste).

  Die Kommunikation mit dem Workflow-Server kann dabei auf zwei verschiedene Arten ablaufen: Zum einen über CORBA und dem Internet InterORB-Protokoll (IIOP), zum anderen über die MQ Series eigene Nachrichtenschicht mit ihren verschiedenen Nachrichten-Queues.

- ***Schicht 2: Server-Komponenten und Buildtime-Komponenten***

  Die Serverkomponenten und die Buildtime-Komponenten sind auf der zweiten Schicht angesiedelt. Die Serverkomponente ist verantwortlich für die Ausführung der einzelnen Workflow-Instanzen zur Laufzeit. Die Buildtime-Komponenten dienen der Definition von Workflow-Modellen und der Abbildung der Organisationsstruktur des Unternehmens.

  Die einzelnen Komponenten der Schicht 2 können auf verschiedenen Maschinen zwecks Lastverteilung ausgeführt werden. Dabei wird die Kommunikation der einzelnen Server-Komponenten sowie die Kommunikation zwischen Server- und Buildtime-Komponenten wieder über die Nachrichten-Schlangen von MQ Series abgewickelt.

[27] Aus [IBM99], Seite 27

- ***Schicht 3: Datenbank Server***
  Die Schicht 3 repräsentiert den Datenbank-Server. Die Datenbank beinhaltet alle Workflow-relevanten Daten für ein MQ Series Workflow System, u.a. auch Status- und Setup-Informationen. Zur Kommunikation zwischen Datenbank-Server und ihrem Client auf Schicht 2 wird das Datenbank-eigene Transportprotokoll verwendet.

6.1.2 Interoperabilitäts-Schnittstellen in MQ Series Workflow

Zur Anbindung externer Systeme, hier kann es sich sowohl um einfache Anwendungen als auch um komplexe WFMS handeln, bietet MQ Series Workflow eine XML-Nachrichten-Schnittstelle an. Diese hat zwei generelle Funktionalitäten. Erstens kann man mit Hilfe einer XML-Nachricht von einer externen Anwendung, innerhalb des MQ Series WFMS entsprechende Prozess-Instanzen ausführen. Dies geschieht durch sogenannte *Request Messages*. Zweitens lassen sich aus MQ Series Workflow heraus andere Applikationen mittels XML-Nachrichten ansprechen und innerhalb dieser Applikationen bestimmte Aktionen ausführen. Diese werden *Invoke Messages* genannt.

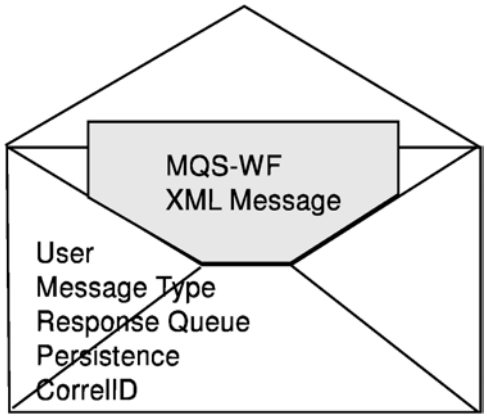


Abbildung 30: Nachrichtenformat einer MQ Series Workflow XML-Nachricht[28]

Der Aufbau dieser Nachricht entspricht keinem standardisierten, sondern einem von IBM eingeführten proprietären Format. Dieses ist folgendermaßen definiert:

Eine Nachricht besteht immer aus zwei Teilen, einem *MQ Series Message Descriptor* (MQMD) und der eigentlichen XML-Nachricht.

- ***UserIdentifier***
  Identifiziert den Benutzer bzw. das System (z.B. bei einem automatischen Aufruf eines Subprozesses), das die Nachricht verfasst hat. Bei *Request Messages*, die an ein MQ Series Workflow-System geschickt werden, wird diese ID dem Anwender zugeordnet, in dessen Auftrag die Anfrage ausgeführt wird. Sie dient außerdem der Authentifizierung des Benutzers.

[28] Aus [IBM99b], Seite 183

Bei *Invoke Messages* beinhaltet die ID den Anwender, auf dessen Auftrag hin die externe Aktivität gestartet wird.

- ***MessageID***
  Die MessageID ist ein festgelegter String mit Wert „FMCXML“, der die Nachricht als MQ Series Workflow XML-Nachricht kennzeichnet.
- ***ReplyToQueue/ReplyToQueueManager***
  Diese Felder enthalten den Namen der Nachrichten-Warteschlange und des Nachrichten-Warteschlangen-Managers, an welche die Antwort geschickt werden soll.
- ***Persistence***
  Gibt an, ob die Nachricht persistent oder transient ist, also nur eine bestimmte Gültigkeitsdauer hat. Bei Anfragen an MQ Series Workflow erhält die Antwort den gleichen Persistence-Modus wie die Anfrage. XML-Nachrichten, die von MQ Series Workflow verschickt werden, sind immer persistent. Antwortnachrichten der aufgerufenen Anwendung sollten ebenfalls persistent sein.
- ***Expiry***
  Hier kann man die Gültigkeitsdauer für transiente Nachrichten in Zehntelsekunden angeben. Persistente Nachrichten sollten auf “unendlich“ (`MQEI_UNLIMITED`) gesetzt werden. Für Anfragen, die an MQ Series Workflow gestellt werden, entspricht die Gültigkeitsdauer der Antwortnachricht der Gültigkeitsdauer der Anfrage abzüglich ihrer Ausführungszeit.
- ***CorrelID***
  Die Korrelations-ID wird dazu genutzt, um Anfragen und Antwortnachrichten zueinander in Beziehung setzen zu können. Antworten, die von MQ Series Workflow geschickt werden, enthalten die gleiche *CorrelID* wie die Anfrage. Bei Anfragen von MQ Series Workflow an andere Systeme sollten die Antworten ebenfalls die gleiche Korrelations-ID enthalten.

Die eigentliche XML-Nachricht basiert auf dem XML 1.0 Standard. Sie besteht wiederum aus zwei Teilen, einem *Message Header* und einem zweiten Teil mit den prozessrelevanten Daten.

Der *Message Header* beinhaltet ein Flag, das angibt, ob eine Antwort auf die Anfrage geschickt werden soll oder nicht. Auch wenn keine Antwort gewünscht wird, kann es im Fehlerfall wichtig sein, dass die aufrufende Anwendung entsprechend informiert wird. Somit gibt es drei Möglichkeiten für das Antwortverhalten des Systems („No“ = keine Antwort / „IfError“ = nur im Fehlerfall / „Yes“ = Antwort). Des weiteren enthält der *Message Header* den Eintrag *User Context*. Dieser Eintrag in der Anfrage-Nachricht entspricht dem der Antwort-Nachricht. Dies dient dazu, dass die entsprechende Antwort-Nachricht der Anfrage-Nachricht im externen System zugeordnet werden kann. Die MQMD, die den gleichen Zweck erfüllt, dient der internen Datentransportsteuerung von MQ Series und ist dem externen System somit nicht bekannt.

Die prozessrelevanten Daten werden in sogenannten Containern, das heißt Datenstrukturen mit XML-Tags, übergeben. Sie repräsentieren die einzelnen Attribute.

Ein Beispiel für die Datenstruktur Kunde ist:

```
Kundendaten:
  -Kundennummer:   Zahl
  -Name:           String
  -Vorname:        String
  -Strasse:        String
  -PLZ:            Zahl
  -Ort:            String
```

Abgebildet in eine XML-Datenstruktur sieht obiges Beispiel folgendermaßen aus:

```
<Kundendaten>
  <Kundennummer>0815</Kundennummer>
  <Name>Mustermann</Name>
        <Vorname>Max</Vorname>
        <Strasse>Musterstrasse 45</Strasse>
        <PLZ>12345</PLZ>
        <Stadt>Musterstadt</Stadt>
</Kundendaten>
```

Diese Container können auch ineinander geschachtelt werden.

6.1.2.1 Request Messages

Externe Anfragen an MQ Series Workflow (z.B. eine Online-Kreditanfrage aus einem Web-basierten System) werden mit sogenannten *Request Messages* über das MQ-Series-Workflow-XML-Nachrichten-Interface gestellt [IBM99b]. Der Ablauf einer XML-Nachrichten-Anfrage ist in Abbildung 31 grafisch dargestellt und wird im Folgenden erklärt.

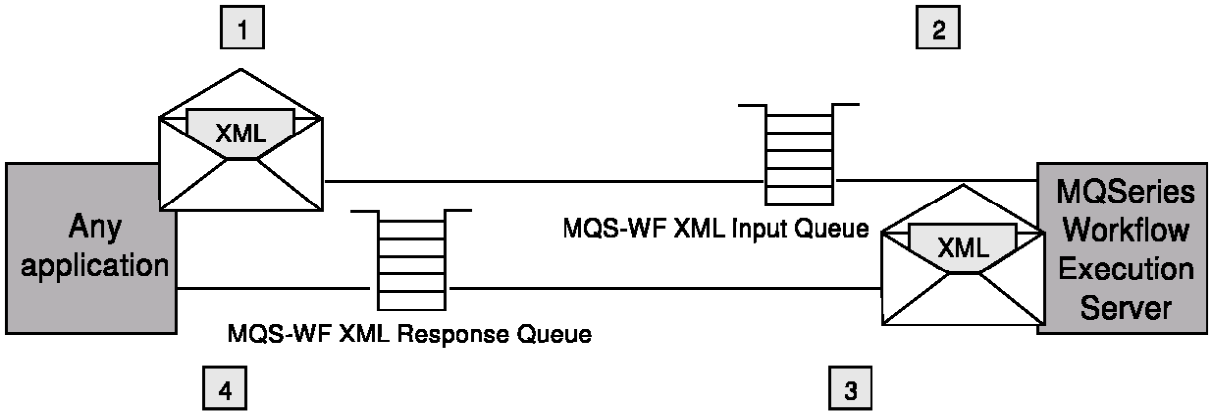


Abbildung 31: Ablauf der Kommunikation bei *Request Messages*[29]

Ablauf der Kommunikation bei Request Messages:

1. Eine externe Anwendung generiert eine MQ Series Workflow XML-Nachricht und leitet sie in die Eingangs-Warteschlange des MQ Series Wf Execution Server.

[29] Aus [IBM99b], Seite 189

2. Der MQ Series Workflow-Ausführungs-Server nimmt die Nachricht entgegen und führt die Anforderung aus.
3. Der MQ Series Workflow-Ausführungs-Server generiert eine XML-Antwort-Nachricht und leitet diese in die MQ Series Workflow Nachrichtenausgangs-Warteschlange, wobei der Name der Warteschlange Teil der MQMD der Eingangsnachricht ist.
4. Die externe Anwendung liest die eingegangene Nachricht und generiert eine Antwort.

Mittels dieser Schnittstelle können nur die beiden MQ Series Workflow Befehle `CreateAndStartInstance()` und `ExecuteProcessInstance()` (siehe [IBM99b] ausgeführt werden.

Eine Beispielnachricht, welche eine Online-Kreditanfrage über die XML-Nachrichten Schnittstelle an ein MQ Series Workflow System stellt, um damit eine entsprechende Prozess-Instanz zu starten, hat den folgenden Aufbau:

```
<?xml version="1.0" standalone="yes"?>
<WfMessage>
  <WfHeader>
    <ResponseRequired>Yes</ResponseRequired>
    <UserContext>This data is sent back in response</UserContext>
  </WfHeader>
  <ProcessTemplateExecute>
    <ProcTemplName>OnlineKreditanfrage</ProcTemplName>
    <ProcInstName>Kreditanfrage #658321</ProcInstName>
    <KeepName>true</KeepName>
    <ProcInstInputData>
      <Kreditdaten>
        <Kunde>
          <Name>User1</Name>
          <Account>4711</Account>
          <Account>0007</Account>
        </Kunde>
        <Amount>100000</Amount>
        <Currency>CurrencyX</Currency>
      </Kreditdaten>
    </ProcInstInputData>
  </ProcessTemplateExecute>
</WfMessage>
```

Hier wird der Befehl *ProcessTemplateExecute* mit der Prozess-Definition „Online-Kredit-Anfrage" gestartet und die erforderlichen Kundendaten, wie Name und Kreditbetrag, übergeben. Anschließend muss vom MQ Series Workflow-System eine Antwort (*ResponseRequired*) generiert und an den Aufrufenden zurückgeschickt werden.

6.1.2.2 Invoke Messages

Um aus MQ Series Workflow externe Applikationen ansprechen zu können, werden sogenannte *Invoke Messages* über das MQ Series Workflow XML-Nachrichten Interface geschickt [IBM99b].

Die externe Applikation muss als *User-defined Program Execution Server* (UPES) in der MQ Series Workflow Buildtime definiert worden sein.

Die Ausführung eines externen Aufrufs kann zum einen synchron, zum anderen asynchron erfolgen. Beim synchronen Aufruf, dem Standardfall, wartet das WFMS so lange mit der weiteren Prozess-Ausführung, bis die *Completion*-Nachricht vom UPES zurückgesandt wird. Hingegen wird beim

asynchronen Aufruf die weitere Ausführung sofort fortgesetzt, eine eventuelle *Completion*-Nachricht wird ignoriert. Somit ist es auch nicht möglich prozessrelevante Daten zurückzuliefern.

Der Ablauf dieser XML-Nachrichten-Anfrage ist in Abbildung 32 grafisch dargestellt und wird im folgenden erklärt.

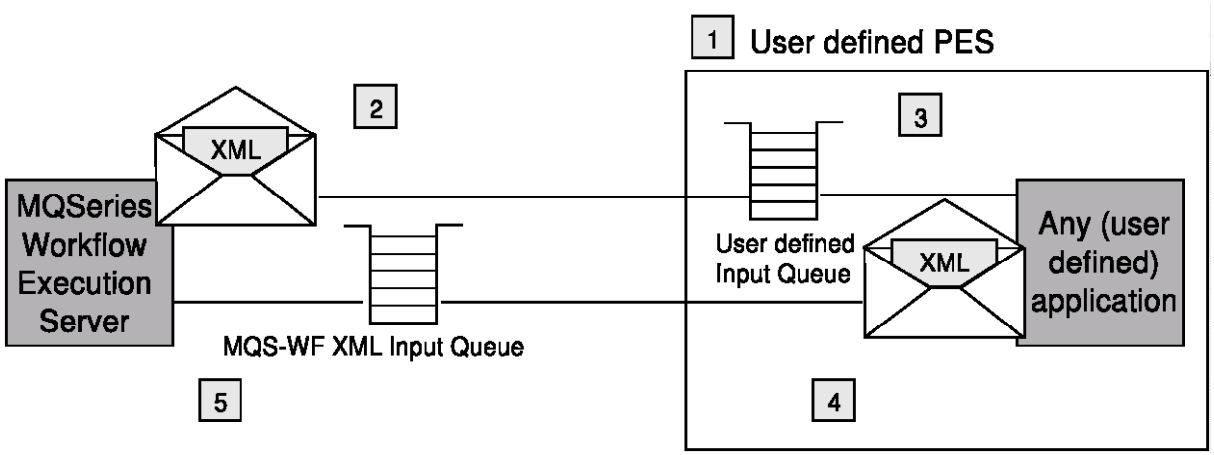


Abbildung 32: Ablauf der Kommunikation bei *Invoke Messages*[30]

Ablauf der Kommunikation bei Invoke Messages:

1. Es muss in der MQ Series Workflow Buildtime ein UPES definiert worden sein.
2. Wenn eine externe Aktivität gestartet werden soll, schickt der MQ Series Workflow Server eine definierte *Invocation Message* zum UPES.
3. Die externe Applikation überwacht ihre Eingangs-Warteschlange. Sobald eine XML-Nachricht eintrifft, liest sie diese Nachricht und führt sie aus.
4. Wenn diese Aktivität beendet ist, generiert die externe Applikation eine Antwortnachricht, sofern verlangt (siehe Messageheader), und leitet sie in die in der MQMD definierte Nachrichtenwarteschlange.
5. Der MQ Series Workflow Ausführungsserver liest die Antwortnachricht, verarbeitet diese weiter und ändert den Status der externen Aktivität entsprechend.

Eine Beispielnachricht, die aus MQ Series Workflow eine Bestellung in ein externes System platziert, hätte den folgenden Aufbau:

```
<ActivityImplInvoke>
  <ActImplCorrelID>FFABCEDF0123456789FF</ActImplCorrelID>
  <Starter>user@systemGroup</Starter>
  <ProgramID>
    <ProcTemplID>84848484FEFEFEFE</ProcTemplID>
    <ProgramName>PerformOrder</ProgramName>
    <ImplementationData>
      <ImplementationPlatform>AIX</ImplementationPlatform>
      <ProgramParameters>/custNo=1234</ProgramParameters>
      <ExeOptions>
```

[30] Aus [IBM99b], Seite 192

```
        <PathAndFileName>/usr/local/bin/perforder</PathAndFileName>
        <WorkingDirectoryName>/usr/local/data</WorkingDirectoryName>
        <InheritEnvironment>true</InheritEnvironment>
        <StartInForeGround>true</StartInForeGround>
        <AutomaticClose>true</AutomaticClose>
        <WindowStyleVisible>true</Visible>
        <RunInXTerm>true</RunInXTerm>
      </ExeOptions>
    </ImplementationData>
    <ProgramInputData>
      <!-- Another container goes here... -->
    </ProgramInputData>
    <ProgramOutputDataDefaults>
      <!-- And yet another container ... -->
    </ProgramOutputDataDefaults>
</ActivityImplInvoke>
```

Hier werden auf einem externen System eine Bestellung des Kunden mit der Nummer 1234 ausgeführt und die notwendigen Daten mit übergeben. Eine mögliche Antwort auf obige Anfrage hat beispielsweise den folgenden Aufbau:

```
<ActivityImplInvokeResponse>
  <ActImplCorrelID>FFABCEDF0123456789FF</ActImplCorrelID>
  <ProgramRC>0</ProgramRC>
  <ProgramOutputData>
    <!-- Another container comes here... -->
  </ProgramOutputData>
</ActivityImplInvokeResponse>
```

Die Antwort wird anhand der *Correlation-ID* dem Aufruf zugeordnet. Der Aufbau der obigen IBM MQ Series Workflow XML-Nachrichten beruhen auf der Dokument Typ Definition (DTD) aus Anhang C. Der Aufbau dieser DTD ist immer abhängig von der externen Anwendung, die angesprochen werden soll, also davon welche Aufrufparameter bzw. welche anderen Workflow-relevanten Daten benötigt werden.

6.1.3 Bewertung

Obwohl IBM Flowmark, wie aus [WFMC01c] ersichtlich, konform zum Interface 4 der WfMC war und IBM selbst Mitglied der WfMC ist, besteht in MQ Series Workflow nur die Möglichkeit einer einfachen XML-Nachrichten-Schnittstelle. Diese erinnert auf den ersten Blick zwar an das WF-XML-Binding der WfMC (vgl. Abschnitt 5.5), bei genauerer Betrachtung erkennt man allerdings, dass man damit nur externe Applikationen starten bzw. Prozess-Instanzen innerhalb von MQ Series Workflow von außen instanzieren kann. Das Aufrufformat entspricht nicht der Spezifikation der WfMC. Auch ist es nicht möglich, die Ausführung einzelner externer Prozesse zu überwachen bzw. die Ausführung zu unterbrechen oder anzuhalten, wie dies z.B. mit den Operationen `ChangeProcessState` oder `GetProcessInstanceDate` (vgl. Kapitel 4) der Wf-XML-Spezifikation möglich wäre. Auch eine Abfrage aller im Moment aktiven Prozess-Instanzen, wie in der Erweiterung des WF-XML-Binding vorgesehen (`ListInstance`), oder eine Benachrichtigungsmöglichkeit des aufrufenden WFMS (`Notify`) sind nicht vorgesehen.

IBM MQ Series Workflow entspricht also in keinster Weise den Anforderungen, die das WF-XML-Binding an ein WFMS stellt, zumal mit WF-XML der Aufruf von internen Prozessen von außen komplett anders abläuft als bei der Ausführung von externen Prozessen durch MQ Series Workflow.

Laut [WFMC01c] ist die Unterstützung der Schnittstellen 2,4 und 5 des WfMC Referenzmodells geplant – wann dies jedoch umgesetzt werden soll, wird nicht angegeben.

6.2 Andere Vertreter

Neben IBM ist Staffware mit dem Produkt Staffware 2000 [Staffware01] ein wichtiger Vertreter im Bereich Workflow-Management. Trotz ausgiebiger Recherche war von Staffware Deutschland keine Informationen über die Unterstützung des WfMC Interface 4 zu bekommen. Laut Aussagen eines Mitarbeiters von Staffware Großbritannien existiert ein optionales Modul zur Unterstützung dieses Interfaces. Nähere Informationen waren jedoch auch darüber nicht in Erfahrung zu bringen.

Im Anhang A findet sich eine Tabelle der WfMC mit den wichtigsten WFMS-Herstellern [WFMC01c]. Ihr kann entnommen werden, welche Systeme welche Schnittstellen der WfMC unterstützen bzw. ob eine solche Unterstützung für einen späteren Zeitpunkt geplant ist.

7 Fortschrittliche Anforderungen und Aspekte

Wie bereits erwähnt, erfüllen heutige Workflow-Management-Konzepte und -Standards bei weitem nicht alle Anforderungen, um alle möglichen Geschäftsvorfälle sinnvoll elektronisch unterstützten zu können. Dieses Kapitel gibt einen Überblick über wichtige Ansprüche, die man an zukünftige WFMS stellen wird, die jedoch in heutigen WFMS bzw. Workflow-Standards noch nicht berücksichtigt worden sind.

- ***Ad-hoc-Abweichungen zur Laufzeit***

  Heutige WFMS erzwingen zur Laufzeit meist die starre Ausführung des modellierten Workflows. Abweichungen von dem vormodellierten Ablauf sind dagegen nicht oder nur stark eingeschränkt möglich. Dies vermindert die breite Einsetzbarkeit von WFMS signifikant. Um Ad-Hoc-Änderungen während der Laufzeit einer Prozess-Instanz vornehmen zu können (z.B. Einfügen, Löschen oder Verschieben eines Prozess-Schrittes), muss diese dynamisch abänderbar sein, und zwar ohne dass dadurch andere Workflow-Instanzen (desselben Typs) beeinträchtigt werden [ReDa98]. Entsprechende Optionen sind in vielen Anwendungsbereichen (z.B. Engineering- oder Krankenhaus-Bereich) unerlässlich, da es hier immer wieder zu außerplanmäßigen Änderungen kommen kann.

  Die standardmäßige Unterstützung von Ad-hoc-Änderungen erfordert für die Zukunft die Definition neuer bzw. die Erweiterung existierender Standards. Insbesondere muss normiert werden, welche Ad-hoc-Änderungen von einem adaptiven WFMS standardmäßig unterstützt werden sollen. Neben Operationen für die Durchführung von Kontrollfluss- und Datenfluss-änderungen müssen zukünftige Standards auch Möglichkeiten zur Modifikation anderer Workflow-Aspekte (z.B. Bearbeiterzuordnungen, Zeitabhängigkeiten, Aktivitätenprogramme) vorsehen. Wichtige Forderungen in diesem Zusammenhang betreffen Aspekte wie Korrektheit, Vollständigkeit, Minimalität oder Sicherheit.

  Die Unterstützung dynamischer Workflow-Änderungen macht eine Erweiterung aller Schnittstellen des WfMC-Referenzmodells erforderlich. So müssen für den Anwendungs-entwickler weitaus mächtigere Programmierschnittstellen als bisher realisiert werden, um die Änderungsmöglichkeiten des WFMS für Anwendungen sinnvoll nutzbar zu machen. Das heißt die API für die Realisierung von Klientenprogrammen (Interface 2 und 3) muss um Operationen zur Festlegung von Ad-hoc-Abweichungen erweitert werden.

  Aber auch für die anderen Schnittstellen des WfMC-Referenzmodells sind im Zusammenhang mit adaptiven WFMS Erweiterungen unerlässlich. Sollen Ad-hoc-Änderungen auch für system- bzw. organisationsübergreifende Workflows möglich sein, müssen das Interface 4 der WfMC bzw. die damit in Verbindung stehenden Standards (vor allem WF-XML) entsprechend erweitert

werden. Auch in Bezug auf das Interface 5 sind sinnvolle Erweiterungen denkbar. So ergeben sich in einem adaptiven WFMS, das kontrollierte Abweichungen vom vormodellierten Workflow zulässt, während der Workflow-Ausführung naturgemäß sehr viel aussagekräftigere Audit-Daten als im Fall starrer Workflow-Ausführungen. Durch die lückenlose Protokollierung aller Abweichungen, zusammen mit den normalen Ausführungsdaten, ist der jeweilige Prozessablauf jederzeit nachvollziehbar. Dadurch ist nicht nur eine verbesserte Qualitätssicherung möglich, sondern es eröffnen sich auch völlig neue Perspektiven im Hinblick auf die kontinuierliche Optimierung der Prozesse. Grundlage für die breite Nutzung dieser Möglichkeiten sind erweiterte Standards für das Auditing von Prozessabläufen (inklusive durchgeführter Abweichungen) und den programmseitigen Zugriff auf diese Informationen. Dies würde auch den herstellerübergreifenden Einsatz „neutraler" Analysewerkzeuge (z.B. für das Process Mining) ermöglichen.

Schließlich erfordert die Unterstützung dynamischer Workflow-Änderungen auch Erweiterungen des Workflow-Metamodells selbst. So muss zum Beispiel sehr präzise angegeben werden können, welche Benutzer bzw. Benutzer welcher Rollen entsprechende Änderungen vornehmen können sollen. Entsprechende Änderungsrechte müssen abhängig von Prozessmodellen und -instanzen, Benutzerrollen, organisatorischen Zuordnungen usw. erfolgen können. Das bedeutet auch, dass das Interface 1 um entsprechende Beschreibungsmöglichkeiten erweitert werden muss.

- ***Abbruch von Prozessen***

  Der Abbruch von Prozessen ist in den heutigen Systemen zwar möglich, jedoch werden Daten-Abhängigkeiten nicht ausreichend betrachtet. Es wird lediglich die Ausführung eines Prozesses beendet. Ein *Rollback*, wie man es von Transaktionen her kennt, ist dagegen nicht möglich. Vor allem im unternehmensübergreifenden Fall ist es immens wichtig, dass die Konsistenz der Daten gewahrt bleibt und dass beim Abbruch von Prozessen sämtliche Anwendungsdaten und Workflow-Daten (wie z.B. Bestände, Bestellmengen, u.ä.) wieder in einen konsistenten Stand überführt werden.

  Auch was die Unterstützung von Prozessabbrüchen betrifft, sind Erweiterungen der bisherigen Standards erforderlich. Zwar gibt es einfache Operationen für den Abbruch von Prozess-Instanzen (vgl. z.B. das Interface 4 der WfMC bzw. WF-XML), diese beschränken sich aber auf reine Statusänderungen (z.B. Überführung eines entfernt ausgeführten Sub-Workflow vom Status `running` in den Status `aborted`). Darüber hinausgehende Aspekte, wie transaktionale Eigenschaften des abgebrochenen Workflows (z.B. semantisches Rollback des abgebrochenen Workflows, Kompensationssphären), Autorisierungskonzepte (z.B. „Wer ist überhaupt berechtigt laufende Prozesse abzubrechen?") oder Semantikfragestellungen (z.B. „Welches Verhalten zeigt das WFMS beim Abbruch von Prozessen?") bleiben dagegen ausgeklammert. Ihre Berücksichtigung erfordert weitreichende Änderungen aller Schnittstellen des WfMC-

Referenzmodells, ähnlich wie dies im Zusammenhang mit Ad-hoc-Workflow-Änderungen bereits diskutiert wurde. Daher wird an dieser Stelle auf eine ausführliche Diskussion verzichtet.

- ***Temporale Aspekte***

  Der Faktor Zeit wird von den heutigen Workflow-Modellen noch so gut wie gar nicht berücksichtigt, sieht man einmal von einfachen Deadlines für die Ausführung von Prozessen oder einzelnen Prozess-Schritten ab [AAH98]. Um Anwender sinnvoll unterstützen und sie rechtzeitig auf drohende Terminverletzungen hinweisen zu können, sind weiterreichende Konzepte von Nöten. Beispielsweise müssen für viele Anwendungen auch minimale und maximale Zeitabstände zwischen Aktivitäten sowie extern festgelegte Termine spezifiziert und zur Laufzeit überwacht werden können [Gri97]. Konzepte hierfür sind aus den Bereichen Netzplantechnik seit längerem bekannt, wurden bisher aber von WFMS noch nicht in ausreichendem Maße umgesetzt.

  Die standardmäßige Unterstützung temporaler Aspekte durch ein WFMS erfordert entsprechende Erweiterungen des WfMC-Referenzmodells. Notwendig sind unter anderem Beschreibungskonzepte für die Definition von Zeitbeschränkungen (Interface 1), Programmierschnittstellen zur Festlegung bzw. Änderung von Terminen für einzelne Prozessinstanzen (Interface 2 und 3), erweiterte Interoperabilitätsschnittstellen (z.B. Abfrage der maximalen Zeitdauer eines entfernt ausgeführten Sub-Workflow; Interface 4) sowie der Einbezug bestimmter Zeitdaten beim Auditing (Interface 5) von Workflow-Ausführungen (z.B. „Wie häufig kam es bei der Ausführung von Workflows eines bestimmten Typs zu Terminüberschreitungen?“).

- ***Sicherheitsaspekte***

  Um eine breite Akzeptanz von unternehmensübergreifenden Workflow-Prozessen zu erreichen, muss gewährleistet sein, dass Sicherheitsaspekte bedacht werden. Das heißt es muss verschiedene Zugriffrechte auf entfernte WFMS geben. Dazu gehört das Lesen und Ändern von Prozessdefinitionen, das Starten, Beenden und Abbrechen von Prozessinstanzen sowie die Statusabfrage von laufenden Instanzen. So darf z.B. ein Mitbewerber nicht die Möglichkeit haben, die Prozessvorlagen von Konkurrenten über entsprechende Interoperabilitätsschnittstellen abzufragen, etwa wenn er die eigenen Prozesse mit diesem Wissen optimieren will. Momentan ist dies in noch keinem Standard explizit definiert, es muss jeweils über einen bilateralen Interoperabilitätsvertrag zwischen zwei Herstellern vereinbart werden.

- ***Statusabfragen***

  Das derzeitige Modell der WFMC beinhaltet nur die Möglichkeit, den aktuellen Status einer laufenden Prozess-Instanz mittels eines Metastatus abzufragen. Dieser schließt aber nur Informationen darüber ein, ob eine Prozess-Instanz momentan ausgeführt wird, pausiert oder ob die Ausführung schon beendet wurde. Somit sind weitere Operationen notwendig, die es ermöglichen, auch übergreifend den genauen Ausführungsfortschritt zu bestimmen (z.B. welche

Prozess-Schritte wurden bereits ausgeführt? Welche werden aktuell bearbeitet? usw.). Entsprechende Möglichkeiten sind z.B. für organisationsübergreifende SCM-Anwendungen (z.B. zwischen Automobilherstellern und -zulieferern) essentiell. Je nach zugrundeliegender Berechtigung der aufrufenden Stelle, können diese Informationen mehr oder weniger umfangreich ausfallen.

- ***Schema-Evolution***

  Hierunter versteht man die Möglichkeit, Änderungen an Prozess-Vorlagen vorzunehmen und diese dann, soweit sinnvoll und möglich, auch auf schon laufende Prozess-Instanzen zu propagieren [CCPP98, JoHe98]. Dazu zählt auch die Verwaltung verschiedener Vorlagen für verschiedene Geschäftspartner, da diese unter Umständen jeweils andere Ansprüche in Bezug auf die Ausführung bestimmter Prozesse haben.

  Ein weiterer wichtiger Aspekt im Zusammenhang mit der Evolution von Workflow-Schemata ist die Verwaltung von Schema-Versionen. Weder in kommerziellen WFMS noch in den in dieser Arbeit behandelten Workflow-Standards gibt es vernünftige Versionierungskonzepte. Dasselbe gilt für die Propagierung von Änderungen auf bereits laufende Instanzen. In MQSeries Workflow etwa gibt es weder für die Versionierung von Workflow-Schemata noch für die Propagierung von Schemaänderungen auf laufende Instanzen eine sinnvolle Unterstützung.

8 Diskussion und Zusammenfassung

Diese Arbeit hat einen Überblick zu aktuellen Standards zur Verbesserung der Interoperabilität zwischen WFMS und zur unternehmensübergreifenden Prozesskopplung gegeben. Die WfMC, als wichtigste Standardisierungsgruppe im Bereich Workflow-Management, hat mit ihrem *Workflow Reference Model* eine Referenzarchitektur vorgestellt, die als Grundlage aller weiteren Standards dient. Um Interoperabilität zu ermöglichen, wurde bereits sehr früh die *Abstract Specification* veröffentlicht. Auf dieser Spezifikation sollen alle Standards für die Zusammenarbeit von WFMS unterschiedlicher Hersteller beruhen. Allerdings wird in dieser Spezifikation auf eine sehr abstrakte Art und Weise die Interoperabilitätsschnittstelle beschrieben, so dass viele für die Praxis relevanten Aspekte ausgeklammert bleiben.

Ausgehend von der *Abstract Specification* wurden mehrere Standards verabschiedet, von denen das WF-XML-Binding der Wichtigste ist. Dieser auf XML-Nachrichten beruhende Standard bietet die Möglichkeit, für hierarchisch strukturierte, von verschiedenen WFMS gesteuerte Workflow-Prozessen über ein Transport-Protokoll (z.B. http) Aufrufdaten auszutauschen. Es handelt sich um einen sehr einfachen Standard mit wenigen Operationsaufrufen, der sehr leicht umzusetzen ist. Allerdings beschränkt sich WF-XML auf den Start und das Ausführen von Sub-Workflows auf entfernten WFMS und bietet keine Möglichkeiten, zeitliche oder kausale Abhängigkeiten zwischen einzelnen Prozessschritten unternehmensübergreifend zu beachten.

Um zu prüfen, ob die Standardisierungsbemühungen der WFMC auch in der Praxis umgesetzt werden, wurde das Workflow-Produkt MQ Series Workflow von IBM, einem wichtigen Mitglied der WFMC, näher betrachtet. IBM setzt derzeit jedoch keine der offiziellen WFMC-Interoperabilitäts-Standards in ihrem Produkt ein. Lediglich eine XML-Schnittstelle ist verfügbar, mittels der sich entfernte Sub-Prozesse starten lassen. Allerdings ist das von IBM verwendete Aufrufformat von IBM proprietär und (noch) nicht konform zu WF-XML.

Um eine breitere Akzeptanz erreichen zu können, müssen die WfMC-Standards stark erweitert werden. Dazu gehören z.B. Interoperabilitätsmodelle für voneinander unabhängige WF-Modelle (z.B. Reihenfolgebeziehungen zwischen einzelnen Schritten) sowie eine genaue Definition des Transport-Protokolls. Auch Sicherheitsaspekten muss in zukünftigen Standards mehr Beachtung geschenkt werden. Des weiteren sind zeitliche Aspekte (z.B. Termine, Zeitabstände, Start- und Endzeiten) sowie Flexibilitätsanforderungen (z.B. Ad-Hoc-Änderungen oder Schema-Evolutionen) bei künftigen Standards zu berücksichtigen.

Alles in allem sind die Standardisierungsbemühungen der WfMC in jüngster Zeit etwas ins Stocken geraten. Die bestehenden Standards müssen baldmöglichst an die erwähnten Anforderungen angepasst werden, um für zukünftige Anwendungsszenarien einsetzbar zu sein. Man kann gespannt

sein, ob mit Jon Pyke, dem neuen Vorsitzenden der WfMC, frischer Wind in diese Bemühungen kommt – ansonsten wird die WFMC ihre wichtige Rolle als Standardisierungsinstanz im Bereich Workflow-Management auf mittlere Sicht verlieren. In der Praxis wird die Industrie ihre eigenen proprietären Standards implementieren, wodurch die Zusammenarbeit von verschiedenen Systemen verschiedener Hersteller noch mehr erschwert wird.

Literaturverzeichnis

[AAH98] N. Adam, V. Atluri, W. Huang*: Modeling and Analysis of Workflows Using Petri Nets*, In: Journal of Intelligent Inf. Systems, Vol. 10, Nr. 2, S. 131-158, 1998

[AIIM01] Hompepage der Association for Information and Image Management, http://www.aiim.org, August 2001

[BoKa99] G.A. Bolcer, G. Kaiser: *SWAP – Leveraging the Web to manage Workflow*, in: Internet Computing, S. 85-88, Januar/Februar 1999

[CCPP98] F. Casati, S. Ceri, B. Pernici und G. Pozzi: *Workflow Evolution*. Data & Knowledge Engineering, Vol. 24, Nr.3, S.211–238, 1998

[DaRe00] P. Dadam, M. Reichert: Vorlesungsskript Workflow-Management-Systeme, Vorlesung, Universität Ulm, Wintersemester 2000/2001

[DRK00] P. Dadam, M. Reichert und K. Kuhn: Clinical Workflows - The Killer Application for Process-oriented Information Systems? In: Proc. 4th Int. Conf. on Business Information Systems, Posen, S. 36-59, April 2000

[Flet82] J. Fletcher: An Arithmetic Checksum for Serial Transmissions. IEEE Transactions on Communication, Vol. 30, No. 1, S. 247-252, Januar 1982

[Gri97] M. Grimm: *ADEPT-time - Temporale Aspekte in flexiblen Workflow-Management-Systemen*, Universität Ulm, Diplomarbeit, 1997

[HaCh95] M. Hammer; J. Champy: Business Reengineering, Campus Verlag, 5. Auflage, 1995, Frankfurt - New York

[Hast99] Ch. Hastedt-Markwardt: Workflow-Management-Systeme, Ein Beitrag der IT zur Geschäftsprozess-Orientierung & Optimierung – Grundlagen, Standards und Trends, in: Informatik Spektrum, Jahrgang 22, S. 99-109, Springer Verlag, 1999

[Haye00] J.G. Hayes et al: *Workflow Interoperability, Standards for the Internet*, in: Internet Computing, Vol.4, No.3, S. 37-45, Mai/Juni 2000

[IBM01] IBM: IBM MQ Series Familiy, Überblick über die MQ Series Produktfamilie, http://www-4.ibm.com/software/ts/mqseries, Stand 09/2001

[IBM01a] IBM: IBM MQ Series, Produktbeschreibung MQ Series, http://www-4.ibm.com/software/ts/mqseries/messaging, Stand 09/2001

[IBM01b] IBM: IBM MQ Series Workflow Version 3.3, Produktbeschreibung, http://www-4.ibm.com/software/ts/mqseries/workflow/v3r3, Stand 09/2001

[IBM96] IBM: IBM FlowMark, *Modeling Workflow*, Version 2, Release 3. IBM Deutschland Entwicklung GmbH. Document Number SH19-8241-02. Wien 1996.

[IBM99] IBM: IBM MQSeries Workflow, *Concepts and Architecture Version 3.2.1*, Third Edition, September 1999

[IBM99a] IBM: IBM MQSeries Workflow, *Administration Guide Version 3.2*, Third Edition, Juni 1999

[IBM99b] IBM: IBM MQSeries Workflow, *Programming Guide Version 3.2.1*, Fifth Edition, September 1999

[JoHe98] G. Joeris, O. Herzog: *Managing Evolving Workflow Specifications,* In: Proc.3rd IFCIS Conf. on Cooperative Information Systems, New York, August 1998

[KnMeZe00] G. Knolmayer, P. Mertens, A. Zeier: SCM auf Basis von SAP-Systemen – Perspektiven der Auftragsabwicklung für Industriebetriebe. Springer-Verlag, Berlin, 2000

[LeRo00] F. Leymann; D. Roller: Production Workflow – Concepts and Technique, Prentice Hall, 1. Auflage, New Jersey, 2000

[Mar01] U. Martschat: Vergleich und Bewertung von Production Workflow-Management-Systemen, Universität Ulm, Diplomarbeit, August 2001

[OMG00] OMG: *Workflow Management Facility*, Version 1.2, April 2000

[OMG01] Hompage der Object Management Group (OMG) http://www.omg.org, August 2001

[OMG95] OMG: *Object Management Architecture Guide*, Version 3.0, 13. Juni 1995

[OMG95a] OMG: *Common Facilities Architecture*, Revision 4.0, November 1995

[OMG95b] OMG: CORBA: Common Object Request Broker Architecture and Specification. Framingham, MA. 1995.

[OMG98] OMG: *CORBAservices: Common Objects Services Specification*, Dezember 1998

[OMG99] OMG: *Workflow Management Facility*, OMG Document Number: bom/99-03-01, 9. März 1999

[PIF01] Homepage des Process Interchange Format (PIF) Projekts, http://ccs.mit.edu/pif, August 2001

[PSL01] Homepage des Process Specification Language (PSL) Projekts, http://www.mel.nist.gov/psl, August 2001

[PSLoJ] The Process Specification Language (PSL): Overview and Version 1.0 Specification

[Rapp00] Reinhold Rapp: *Customer Relationship Management. Das neue Konzept zur Revolutionierung der Kundenbeziehungen*, Campus Verlag, Frankfurt, 2000

[ReDa98] M. Reichert, P. Dadam: *ADEPTflex - Supporting Dynamic Changes of Workflows Without Loosing Control*, Journal of Intelligent Information Systems, Vol. 10, No. 2, S. 93-129, März 1998

[ReDa00] M. Reichert, P. Dadam: *Geschäftsprozessmodellierung und Workflow-Management – Konzepte, Systeme und deren Anwendung*, in: Industrie Management, Modellierung und Simulation, 3/2000, GITO-B Verlag

[RSD97] M. Reichert, B. Schultheiß, P. Dadam: Erfahrungen bei der Entwicklung vorgangsorientierter, klinischer Anwendungssysteme auf Basis prozessorientierter Workflow-Technologie , Proc. 42. Jahrestagung der GMDS (GMDS'97), Ulm, S. 181-187, September 1997

[Staffware01] Homepage der Firma Staffware, http://www.staffware.com, August 2001

[Swen98] K. Swenson: *Simple Workflow Access Protocol (SWAP), Internet Draft, 1998*

[W3C00] World Wide Web Consortium: Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6 October 2000, http://www.w3.org/TR/REC-xml, August 2001

[W3C00a] World Wide Web Consortium: Simple Object Access Protocol (SOAP) 1.1, W3C Note 08 May 2000, http://www.w3.org/TR/SOAP, August 2001

[W3C00b] World Wide Web Consortium: XHTML 1.0: The Extensible Hyper Text Markup Language - A Reformulation of HTML 4 in XML 1.0, W3C Recommendation 26 January 2000, http://www.w3.org/TR/xhtml1, August 2001

[W3C01a] World Wide Web Consortium: XML Schema Part 0: Primer, W3C Recommendation, 2 May 2001, http://www.w3.org/TR/xmlschema-0, August 2001

[W3C01b] World Wide Web Consortium: XML Schema Part 1: Structures, W3C Recommendation, 2 May 2001, http://www.w3.org/TR/xmlschema-1, August 2001

[W3C01c] World Wide Web Consortium: XML Schema Part 2: Datatypes, W3C Recommendation, 2 May 2001, http://www.w3.org/TR/xmlschema-2, August 2001

[WFMC00] Workflow Management Coalition: *Workflow Standard – Interoperability,* Internet e-mail MIME Binding, WFMC-TC-1018, Version 1.2, Januar 2000

[WFMC00a] Workflow Management Coalition: *Workflow Standard – Interoperability, Wf-XML Binding*, WFMC-TC-1023, Version 1.0, Mai 2000

[WFMC00b] Workflow Management Coalition: Workflow Standard – Interoperability XML-HTTP binding, Document Status: Submission 2.0, 8 February, 2000

[WFMC01] Workflow Management Coalition: *Workflow Process Definition Interface- XML Definition Language,* WFMC-TC-1025, Draft 0,03a, 22. Mai 2001

[WFMC01a] Workflow Management Coalition: *Workflow Process Definition Interface- XML Definition Language, XML-DTD,* http://www.wfmc.org/standards/docs/xpdl.dtd, August 2001

[WFMC01b] Workflow Management Coalition: *Workflow Standards and Associated Documents,* http://www.wfmc.org/standards/docs/Stds_diagram.pdf, August 2001

[WFMC01c] Workflow Management Coalition: *Conformance to Interface Standards,* http://www.wfmc.org/standards/conformance.htm, August 2001

[WFMC01d] Workflow Management Coalition: *Published Documents,* http://www.wfmc.org/standards/docs.htm, August 2001

[WFMC01e] Homepage der Workflow Management Coalition, http://www.wfmc.org, August 2001

[WFMC95] Workflow Management Coalition: The Workflow Reference Model. WFMC-TC00-1003, Version 1.1, Januar 1995

[WFMC96] Workflow Management Coalition: *Workflow Standard – Interoperability, Abstract Specification*, WFMC-TC-1012, Version 1.0, October 1996

[WFMC96a] Workflow Management Coalition: Workflow Client Application (Interface 2), Application Programming Interface (WAPI), Specification, WFMC-TC-1009, Version 1.1, 15. Mai 1996

[WFMC97] Workflow Management Coalition: Workflow Client Application (Interface 2), Application Programming Interface (WAPI) Naming Conventions, WFMC-TC-1013, Version 1.4, 1. November 97

[WFMC98] Workflow Management Coalition: *Workflow Management Application Programming Interface (Interface 2&3)* Specification WFMC-TC-1009 Version 2.0 Juli 1998

[WFMC98a] Workflow Management Coalition*: Audit Data Specification*, Document Number WFMC-TC-1015, Version 1.1, *22 Sep 1998*

[WFMC98b] Workflow Management Coalition: *Workflow Standard – Interoperability,* Internet e-mail MIME Binding, WFMC-TC-1018, Version 1.1, 25. September 1998

[WFMC99] Workflow Management Coalition: *Terminologie & Glossary.* WFMC-TC-1011, Version 3.0, Februar 1999

[WFMC99a] Workflow Management Coalition: *Interface 1: Process Definition Interchange, Process Model*, WfMC TC-1016-P,Version 1.1 (Official release), Oktober 1999

[WFMC99b] Workflow Management Coalition: *Interface 1: Process Definition Interchange, Q&A and Examples*, WfMC TC-1016-X,Version 7.01 (Draft), 1. Januar 1999

Abkürzungsverzeichnis

AIIM	Association for Information an Image Management
API	Application Programming Interface
BPR	Business Process Reengineering
CORBA	Common Object Request Broker Architecture
CWAD	Common Workflow Audit Data
DDE	Dynamic Data Exchange
DMS	Dokumenten-Management-System
DTD	Document Type Definition
EAS	Elektronisches Archiv-System
FDL	Flow Definition Language
IETF	Internet Engineering Task Force
IIOP	Internet InterORB Protocol
IDL	Interface Definition Language
IF	Interface
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
MAPI	Message API
MIME	Multipurpose Internet Mail Extension
MQMD	MQSeries Message Descriptor
NIST	National Institute of Standards and Technology
ODMA	Open Document Management API
OLE	Object Linking and Embedding
OMG	Object Management Group
OMA	Object Management Architecture
ORB	Object Request Broker
PCDATA	Parsed Character Data
PIF	Process Interchange Format
PSL	Process Specification Language
RFC	Request for Comment

RFP	Request for Proposal
SGML	Standard Generic Markup Language
SSL	Secure Sockets Layer
SWAP	Simple Workflow Access Protocol
UPES	User-defined Program Execution Server
URI	Universal Resource Identifier
URL	Universal Resource Locator
W3C	World Wide Web Consortium
WAPI	Workflow Application Programming Interface
WF	Workflow
WfMC	Workflow Management Coalition
WFMS	Workflow Management System
WPDL	Workflow Process Definition Language
XML	Extensible Markup Language
XPDL	XML Process Definition Language

Abbildungsverzeichnis

Abbildung 1: Schematischer Aufbau eines WFMS 10
Abbildung 2: Struktur eines Workflow-Management-System 11
Abbildung 3: Trennung von Ablauflogik und Anwendungscode 12
Abbildung 4: Das Workflow-Referenzmodell der WfMC 14
Abbildung 5: Workflow Standards und verwandte Dokumente der WfMC 15
Abbildung 6: Prozess Metamodell 16
Abbildung 7: Das Konzept des Prozess-Definitions-Austausches 17
Abbildung 8: Zusammenhang von Prozessdefinitionen, -instanzen und Arbeitslisten 18
Abbildung 9: Funktionsweise der Tool Agenten 20
Abbildung 10: Referenzmodell der Object Management Architecture (OMA) 23
Abbildung 11: Joint Workflow Management Facility Model 25
Abbildung 12: Schematische Darstellung der Ausgangsituation 29
Abbildung 13: Darstellung einer herkömmlichen Verfügbarkeitsanfrage 30
Abbildung 14: Darstellung einer verknüpften Verfügbarkeitsanfrage 31
Abbildung 15: Ablaufgraph einer Verfügbarkeitsanfrage 32
Abbildung 16: Ablaufgraph eines Bestellvorgangs 33
Abbildung 17: Schema der Statusabfrage einer Bestellung 34
Abbildung 18: Schema der Stornierung einer Bestellung 35
Abbildung 19: Verkettete Prozess-Ausführung 40
Abbildung 20: Verschachtelte Prozess-Ausführung 41
Abbildung 21: Synchronisierte Prozess-Ausführung 41
Abbildung 22: Bilaterale und normierte Schnittstellen zwischen verschiedenen Systemen 46
Abbildung 23: Schema der Entwicklungsgeschichte der Interoperabilitätsstandards 48
Abbildung 24: Beispielkonversation mit MIME-Nachrichten 50
Abbildung 25: Aufbau einer MIME-Nachricht 51
Abbildung 26: Multipart MIME Nachricht 52

Abbildung 27: Reine Text-MIME Nachricht 53

Abbildung 28: Prozessmodell der WF-XML Spezifikation 57

Abbildung 29: 3-Schichten Modell von MQ Series Workflow 66

Abbildung 30: Nachrichtenformat einer MQ Series Workflow XML-Nachricht 67

Abbildung 31: Ablauf der Kommunikation bei *Request Messages* 69

Abbildung 32: Ablauf der Kommunikation bei *Invoke Messages* 71

Tabellenverzeichnis

Tabelle 1: Übersicht über die fünf Schnittstellen des WfMC Referenz Modells 14

Tabelle 2: Sequenznummer der MIME-Nachrichten 53

Tabelle 3: Wiederholungs- und Verkettungsoperatoren der XML-Spezifikation..... 56

Tabelle 4: Vergleich OMG Objektmodell mit WfMC WF-XML 62

Anhang

A) WFMC Interface-Konformität

Überblick über wichtige WFMS-Hersteller und deren Konformität zur WFMC [WFMC01c]:

Anbieter	Produkt/Version	Release Datum	Unterstütztes WfMC Interface	Implementiert
Action Technologies Inc	ActionWorks Metro 5.0	27.04.1998	IF2/3 IF4	Prototypes Demonstrated
BancTec/Plexus Software	FloWare V 5.0	Sep 98	IF1 IF2/3 IF4	Yes Prototype Demonstrated Planned
BOC GmbH	ADONIS V2.0/V3.0	1998	IF1 IF5	Prototype Trial Developments
Concentus	KI Shell 2.0/3.0	25.06.1997	IF1 IF2/3 IF4 IF5	No Yes Yes No
CSE Systems	CSE/WorkFlow Gold Edition Cibon	Sep 97 31.03.1999	IF1,IF2/3,IF4 IF1,IF2/3 IF4	Yes Yes JointFlow Supported
FileNET	Visual WorkFlo 3.0 Integrated WorkFlo 1.22	Sep 98	IF4	Yes
HandySoft Corp	Handy*Solution V5.0	30.09.1998	IF2/3 IF4	Planned Planned
Hitachi Ltd	Groupmax Workflow V5 WorkCoordinator	Mai 99 Nov 98	IF4 IF2/3 IF4	Yes Yes Planned
IBM	Flowmark 2.3 MQ Series Workflow	Dez 97 Jun 98	IF2/3/4 IF1 IF2/4/5 OMG JointFlow	Yes Trial developments Planned

INSIEL Sp.A	Office241	Jun 98	IF2/3	Yes
	OfficeFlow		IF4	Prototype
	V2.6.9a		IF5	Planned
IDS Prof. Scheer GmbH	ARIS Toolset 3.2	1997	IF1	Prototype
	Modelling and BPR Tool		IF2/3,IF4,IF5	NO
KAISHA-Tec	Modeler Pro V 1.2	Mai 97	IF1	Yes
Ley GmbH	COSA Workflow 2.0	31.03.1997	IF1,IF2/3,IF4,IF5	No
Meta Software Corporation	WorkFlow Analyzer V 4.0	Jun 98	IF1	WPDL 7.04 Beta Implemented 1.0 Rel sched 6/99
PROMATIS	INCOME Workflow V 1.0	01.04.1998	IF1,IF2/3	Yes
			IF4	No
			IF5	Yes
SAP	SAP WebFlow/ SAP Business Workflow	1997 R/3. Rel. 3.1	IF2/3	Implemented
		2000		
		Technology Release 4.6C		
			Wf-XML	Implemented
Staffware	Staffware 97	04.07.1998	IF1	Trial developments done
	Staffware 2000		IF2/3	Prototype demonstrated
			IF4	Optional Module
			IF5	Trial Developments
		05.01.2001	Wf-XML 1.0	Implemented
Technology Deployment International	WebDeploy: WorkFlow 1.2	Jun 98	IF2	Yes
TIBCO	TIB/InConcert	2000	IF 1	Prototype demonstrated
		1996	IF 2/3	Prototype demonstrated
		1999	IF 4	Prototype demonstrated
		planned	IF 5	TIB/InConcert supports the abstract Audit Data Specification
Versata Inc	Versata Interaction Server (VIS)	TBA	Wf-XML v1.0	Implemented

B) DTD WF-XML

Die folgende Dokument-Typ-Definition (DTD) beschreibt den Aufbau der im WF-XML Binding definierten Operationen im XML-Format:

```
<!-- W f-XML DTD, Revision 1.0 Final - 11 April, 2000

If a DOCTYPE declaration is required to parse this set of declarations,
the following line should be prepended to this file:

<!DOCTYPE WfMessage PUBLIC "-//WfMC//DTD Wf-XML 1.0//EN"
"http://www.wfmc.org/standards/docs/Wf-XML-1.0.dtd" [

and the following line appended: ]>

-->

<!-- ~~~~~~~~~~~~~~~~~~ Entity Declarations ~~~~~~~~~~~~~~~~~~ -->
<!-- The ISOLangs entity provides the choices for the ResponseLang attribute of
the Request element. These language codes are taken from the ISO 639:1988
standard, which can be used for further clarification of the names of each
language and can be obtained from http://www.iso.ch/cate/d4766.html. Additional
information is also available at: http://www.oasis -open.
org/cover/iso639a.html. -->

<!ENTITY % ISOLangs
"(aa|ab|af|am|ar|as|ay|az|ba|be|bg|bh|bi|bn|bo|br|ca|co|cs|cy|da|de|dz|el|en|eo|
es|et|eu|fa|fi|fj|fo|fr|fy|ga|gd|gl|
gn|gu|ha|hi|hr|hu|hy|ia|ie|ik|in|is|it|iw|ja|ji|jw|ka|kk|kl|km|kn|ko|ks|ku|ky|la
|ln|lo|lt|lv|mg|mi|mk|ml|mn|mo|mr|ms|
mt|my|na|ne|nl|no|oc|om|or|pa|pl|ps|pt|qu|rm|rn|ro|ru|rw|sa|sd|sg|sh|si|sk|sl|sm
|sn|so|sq|sr|ss|st|su|sv|sw|ta|t
e|tg|th|ti|tk|tl|tn|to|tr|ts|tt|tw|uk|ur|uz|vi|vo|wo|xh|yo|zh|zu)">


<!-- The following two entities are used to define the request and response
elements for each operation. -->

<!ENTITY % OperationRequest "(CreateProcessInstance.Request |
GetProcessInstanceData.Request | ChangeProcessInstanceState.Request |
ProcessInstanceStateChanged.Request)">

<!ENTITY % OperationResponse "(CreateProcessInstance.Response |
GetProcessInstanceData.Response | ChangeProcessInstanceState.Response |
ProcessInstanceStateChanged.Response)">

<!-- The ProcessInstanceData entity defines the properties of a process instance
that may be obtained using the GetPrcoessInstanceData operation. -->

<!ENTITY % ProcessInstanceData "(Name | Subject | Description | State |
ValidStates | ObserverKey | ResultData | ProcessDefinitionKey | Priority |
LastModified)+">

<!-- This is the list of valid states defined by the WfMC for version 1.0 of Wf-
XML. -->

<!ENTITY % vstates "(open.notrunning | open.notrunning.suspended | open.running
| closed.completed | closed.abnormalCompleted
|closed.abnormalCompleted.terminated |
closed.abnormalCompleted.aborted)*">

<!-- ~~~~~~~~~~~~~~~~~~ Element Declarations ~~~~~~~~~~~~~~~~~~ -->

<!-- Root element -->
```

```
<!ELEMENT WfMessage (WfTransport?, WfMessageHeader, WfMessageBody)>
<!ATTLIST WfMessage Version CDATA #REQUIRED>

<!-- ~~~~~~~~~~ WfTransport ~~~~~~~~~~~~ -->

<!-- Used for transport-specific information, such as special security or
asynchronous processing. -->

<!ELEMENT WfTransport (CorrelationData?)>

<!ELEMENT CorrelationData (#PCDATA)>

<!-- ~~~~~~~~ WfMessageHeader ~~~~~~~~~~ -->

<!-- Information generally used in all messages, helpful for preprocessing. -->
<!ELEMENT WfMessageHeader ((Request | Response), Key)>

<!ELEMENT Request EMPTY>
<!ATTLIST Request ResponseRequired (Yes | No | IfError) #REQUIRED
ResponseLang %ISOLangs; #IMPLIED>

<!ELEMENT Response EMPTY>

<!-- The URI of the resource. -->
<!ELEMENT Key (#PCDATA)>

<!-- ~~~~~~~~~ WfMessageBody ~~~~~~~~~~ -->

<!ELEMENT WfMessageBody (%OperationRequest; | %OperationResponse;)>
<!ELEMENT CreateProcessInstance.Request (ObserverKey?, Name?, Subject?,
Description?, ContextData)>

<!ATTLIST CreateProcessInstance.Request StartImmediately (true|false) #FIXED
"true">

<!ELEMENT ObserverKey (#PCDATA)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Subject (#PCDATA)>
<!ELEMENT Description (#PCDATA)>

<!ELEMENT ContextData ANY>

<!ELEMENT GetProcessInstanceData.Request (ResultDataSet?)>
<!ELEMENT ResultDataSet %ProcessInstanceData;>

<!ELEMENT State %vstates;>
<!ELEMENT ValidStates %vstates;>
<!ELEMENT open.notrunning EMPTY>
<!ELEMENT open.notrunning.suspended EMPTY>
<!ELEMENT open.running EMPTY>
<!ELEMENT closed.completed EMPTY>
<!ELEMENT closed.abnormalCompleted EMPTY>
<!ELEMENT closed.abnormalCompleted.terminated EMPTY>
<!ELEMENT closed.abnormalCompleted.aborted EMPTY>

<!ELEMENT ResultData ANY>

<!ELEMENT ProcessDefinitionKey (#PCDATA)>
<!ELEMENT Priority (#PCDATA)>
<!ELEMENT LastModified (#PCDATA)>

<!ELEMENT ChangeProcessInstanceState.Request (State)>

<!ELEMENT ProcessInstanceStateChanged.Request (ProcessInstanceKey, State,
ResultData?, LastModified?)>
```

```
<!ELEMENT ProcessInstanceKey (#PCDATA)>

<!ELEMENT CreateProcessInstance.Response (ProcessInstanceKey | Exception)>

<!ELEMENT GetProcessInstanceData.Response (%ProcessInstanceData; | Exception)>

<!ELEMENT ChangeProcessInstanceState.Response (State | Exception)>

<!ELEMENT ProcessInstanceStateChanged.Response (Exception?)>

<!ELEMENT Exception (MainCode, SubCode?, Type, Subject, Description?)>
<!ELEMENT MainCode (#PCDATA)>
<!ELEMENT SubCode (#PCDATA)>
<!ELEMENT Type (#PCDATA)>
```

C) DTD IBM MQ Series Workflow XML Message

Die folgende Dokument-Typ-Definition (DTD) beschreibt den Aufbau der Request-Messages und Invoke-Messages der XML-Nachrichten-Schnittstelle von IBM MQ Series Workflow:

```
<!-- FmcXMLIF.dtd == DTD for MQSeries Workflow messages -->
<!-- Message ======================================== -->
<!ELEMENT WfMessage
( WfMessageHeader?,
( ProcessTemplateCreateAndStartInstance
| ProcessTemplateCreateAndStartInstanceResponse
| ProcessTemplateExecute
| ProcessTemplateExecuteResponse
| ActivityImplInvoke
| ActivityImplInvokeResponse
| GeneralError ) ) >
<!-- ================================================
Workflow Message Header
================================================ -->
<!ELEMENT WfMessageHeader (ResponseRequired?,UserContext?)>
<!-- Opaque -->
<!ELEMENT UserContext (#PCDATA) >
<!-- Enumerated type -->
<!ELEMENT ResponseRequired (#PCDATA)>
<!-- Expected values: {No,IfError,Yes} -->
<!-- ================================================
Specific Messages
================================================ -->
<!-- ProcessTemplateCreateAndStart ================== -->
<!ELEMENT ProcessTemplateCreateAndStartInstance
( ProcTemplName,
ProcInstName,
KeepName,
ProcInstInputData ) >
<!ELEMENT ProcessTemplateCreateAndStartInstanceResponse
( ProcessInstance
| Exception ) >
<!-- ProcessTemplateExecute ========================= -->
<!ELEMENT ProcessTemplateExecute
( ProcTemplName,
ProcInstName,
KeepName,
ProcInstInputData ) >
<!ELEMENT ProcessTemplateExecuteResponse
( ( ProcessInstance,
ProcInstOutputData )
| Exception ) >
<!-- ActivityImplInvoke ============================= -->
<!ELEMENT ActivityImplInvoke
( ActImplCorrelID,
Starter,
ProgramID,
(ImplementationData)*,
ProgramInputData,
ProgramOutputDataDefaults ) >
<!ELEMENT ActivityImplInvokeResponse
( ActImplCorrelID,
( ProgramRC,
ProgramOutputData )
| Exception ) >
<!-- GeneralError =================================== -->
<!ELEMENT GeneralError (Exception) >
<!-- ================================================
Data Structures
```

```
================================================ -->
<!-- Named Entities ================================== -->
<!ENTITY %CONTAINER "CreditData | InsuranceData | Address | Customer|...">
<!ELEMENT ProcInstInputData (%CONTAINER;) >
<!ELEMENT ProcInstOutputData (%CONTAINER;) >
<!ELEMENT ProgramInputData (%CONTAINER;) >
<!ELEMENT ProgramOutputData (%CONTAINER;) >
<!ELEMENT ProgramOutputDataDefaults (%CONTAINER;) >
<!-- Process Instance ================================ -->
<!ELEMENT ProcessInstance
( ProcInstID,
ProcInstName,
ProcInstParentName?,
ProcInstTopLevelName,
ProcInstDescription?,
ProcInstState,
LastStateChangeTime,
LastModificationTime,
ProcTemplID,
ProcTemplName,
Icon,
Category? ) >
<!-- Program ID ====================================== -->
<!ELEMENT ProgramID
( ProcTemplID,
ProgramName ) >
<!-- Implementation Data ============================= -->
<!ELEMENT ImplementationData
( ImplementationPlatform ,
ProgramParameters,
( ExeOptions
| DllOptions
| ExternalOptions ) ) >
<!ELEMENT ExeOptions
( PathAndFileName,
WorkingDirectoryName?,
Environment?,
InheritEnvironment,
StartInForeGround,
AutomaticClose,
WindowStyle?,
RunInXTerm ) >
<!ELEMENT DllOptions
( PathAndFileName,
EntryPointName,
ExecuteFenced,
KeepLoaded )
<!ELEMENT ExternalOptions
( ServiceName,
ServiceType,
InvocationType,
ExecutableName,
ExecutableType,
IsLocalUser,
IsSecurityRoutineCall,
CodePage,
TimeoutPeriod,
TimeoutInterval?,
IsMappingRoutineCall,
MappingType?,
ForwardMappingFormat?,
ForwardMappingParameters?,
BackwardMappingFormat?,
BackwardMappingParameters? ) >
<!-- Exception ======================================== -->
```

```
<!ELEMENT Exception
(Rc, Parameters, MessageText?, Origin) >
<!-- Message text is optional, as it will be ignored
in messages being sent *to* the Wf server. -->
<!ELEMENT Parameters
(Parameter*) >
<!-- Data Elements ================================== -->
<!-- Booleans -->
<!ELEMENT AutomaticClose (#PCDATA) > <!-- Expected values: {true, false} -->
<!ELEMENT DllV2Compatible (#PCDATA) > <!-- Expected values: {true, false} -->
<!ELEMENT ExecuteFenced (#PCDATA) > <!-- Expected values: {true, false} -->
<!ELEMENT InheritEnvironment (#PCDATA) > <!-- Expected values: {true, false} -->
<!ELEMENT IsLocalUser (#PCDATA) > <!-- Expected values: {true, false} -->
<!ELEMENT IsMappingRoutineCall (#PCDATA) > <!-- Expected values: {true, false} -
->
<!ELEMENT IsSecurityRoutineCall (#PCDATA) > <!-- Expected values: {true, false}
-->
<!ELEMENT KeepLoaded (#PCDATA) > <!-- Expected values: {true, false} -->
<!ELEMENT KeepName (#PCDATA) > <!-- Expected values: {true, false} -->
<!ELEMENT RunInXTerm (#PCDATA) > <!-- Expected values: {true, false} -->
<!ELEMENT StartInForeGround (#PCDATA) > <!-- Expected values: {true, false} -->
<!-- Strings -->
<!ELEMENT BackwardMappingFormat (#PCDATA) >
<!ELEMENT BackwardMappingParameters (#PCDATA) >
<!ELEMENT Category (#PCDATA) >
<!ELEMENT EntryPointName (#PCDATA) >
<!ELEMENT Environment (#PCDATA) >
<!ELEMENT ExecutableName (#PCDATA) >
<!ELEMENT ExecutableType (#PCDATA) >
<!ELEMENT ForwardMappingFormat (#PCDATA) >
<!ELEMENT ForwardMappingParameters (#PCDATA) >
<!ELEMENT Icon (#PCDATA) >
<!ELEMENT InvocationType (#PCDATA) >
<!ELEMENT MappingType (#PCDATA) >
<!ELEMENT MessageText (#PCDATA) >
<!ELEMENT Origin (#PCDATA) >
<!ELEMENT Parameter (#PCDATA) >
<!ELEMENT PathAndFileName (#PCDATA) >
<!ELEMENT ProcInstDescription (#PCDATA) >
<!ELEMENT ProcInstName (#PCDATA) >
<!ELEMENT ProcInstParentName (#PCDATA) >
<!ELEMENT ProcInstTopLevelName (#PCDATA) >
<!ELEMENT ProcTemplName (#PCDATA) >
<!ELEMENT ProgramName (#PCDATA) >
<!ELEMENT ProgramParameters (#PCDATA) >
<!ELEMENT ServiceName (#PCDATA) >
<!ELEMENT ServiceType (#PCDATA) >
<!ELEMENT Starter (#PCDATA) >
<!ELEMENT WorkingDirectoryName (#PCDATA) >
<!-- Opaque -->
<!ELEMENT ActImplCorrelID (#PCDATA) >
<!ELEMENT ProcInstID (#PCDATA) >
<!ELEMENT ProcTemplID (#PCDATA) >
<!-- Numbers -->
<!ELEMENT CodePage (#PCDATA) >
<!ELEMENT ProgramRC (#PCDATA) >
<!ELEMENT Rc (#PCDATA) >
<!ELEMENT TimeoutInterval (#PCDATA) >
<!-- Timestamps YYYY-MM-DD-hh.mm.ss.000000 (000000 milliseconds) -->
<!ELEMENT LastModificationTime (#PCDATA) >
<!ELEMENT LastStateChangeTime (#PCDATA) >
<!-- Enumerated types -->
<!ELEMENT ImplementationPlatform (#PCDATA) > <!-- Expected values:
{ OS2, AIX,
HPUX, Windows95,
```

```
WindowsNT, OS390,
Solaris, Linux } -->
<!ELEMENT ProcInstState (#PCDATA) > <!-- Expected values:
{ Ready, Running,
Finished, Terminated,
Suspended, Terminating,
Suspending,Deleted } -->
<!ELEMENT WindowStyle (#PCDATA) > <!-- Expected values:
{ Visible, Invisible,
Minimized, Maximized } -->
<!ELEMENT TimeoutPeriod (#PCDATA) > <!-- Expected values:
{ TimeInterval
Forever Never } -->
<!-- Container ======================================== -->
<!ELEMENT CreditData (...) >
<!ELEMENT OrderData (...) >
<!ELEMENT InsuranceData (...) >
<!ELEMENT Address (...) >
<!ELEMENT Customer (...) >
```

D) Exception Codes des WF-XML Binding

Die folgende Liste beinhaltet die dreistelligen Fehler-Codes des WF-XML-Binding. Sie sind in 7 Hauptkategorien aufgeteilt. Diese Codes werden im Fehlerfall zurückgeliefert, um Aufschluss über die eingetretene Ausnahme zu geben.

WfMessageHeader-specific 100 Series	
These exceptions deal with missing or invalid parameters in the header.	
WF_PARSING_ERROR	100
WF_ELEMENT_MISSING	101
WF_INVALID_VERSION	102
WF_INVALID_RESPONSE_REQUIRED_VALUE	103
WF_INVALID_KEY	104
WF_INVALID_OPERATION_SPECIFICATION	105
Data 200 Series	
These exceptions deal with incorrect context or result data	
WF_INVALID_KEY	200
WF_INVALID_CONTEXT_DATA	201
WF_INVALID_RESULT_DATA	202
Authorization 300 Series	
A user may not be authorized to carry out this operation on a particular resource, e.g., may not create a process instance for that process definition.	
WF_NO_AUTHORIZATION	300
Operation 400 Series	
The operation can not be accomplished because of some temporary internal error in the workflow engine. This error may occur even when the input data is syntactically correct and authorization is permitted.	
WF_OPERATION_FAILED	400
Resource Access 500 Series	
A valid Key has been used, however this operation cannot be currently invoked on the specified resource.	
WF_NO_ACCESS_TO_RESOURCE	500

WF_INVALID_PROCESS_DEFINITION	502
Operation-specific 600 Series	
These are the more operation specific exceptions. Typically, they are only used in a few operations, possibly a single one.	
WF_INVALID_STATE_TRANSITION	600
WF_INVALID_OBSERVER_FOR_RESOURCE	601
Extensibility 800 Series	
An additional exception main code is provided to allow implementations of the WF-XML specification to return additional exceptions	
WF_OTHER	800

E) Aufbau von MIME Nachrichten im BNF-Format

Die folgende Auflistung stellt den Aufbau der MIME-Nachrichten des E-Mail-MIME-Bindung der WFMC im BNF-Format dar:

```
<message Body>     ::=   <head> <operation list> <tail>

<head>             ::=   <message type> <sequence> <conversation id>
                         "&" <timestamp> <encoding> <terminator>

<message type>     ::=   "wfmc-if4-request"
                       | "wfmc-if4-response"
                       | "wfmc-if4-error"   "(" <message error> ")"

<sequence>         ::=  "[" <message id> "]"

<conversation id>  ::=   <source conversation id> "+"
                       | <source conversation id> "+"
                         <target conversation id>

<encoding>         ::=   "," "encoding" "=" <encoding name>
                       | <empty>

<operation list>   ::=   <op>
                       | <op> <operation list>
                       | <empty>

<op>               ::=   <operation> "?" <name-value list> <extended>
                         <terminator>

<extended>         ::=   "&extended&"  <name-value list>
                       | <empty>

<name-value list>  ::=   <name> "=" <value>
                       | <name> "=" <value> "&" <name-value list>

<terminator>       ::=   "&&"

<tail>             ::=   "end" "(" <checksum> ")"
```

Where:

<checksum>	This is the message checksum from the first byte of this part (always a "w" of either wfmc-if4-request, wfmc-if4-response, or wfmc-if4-error) to the last byte before the "(" (always the "d" of "end"). See section 7.2.2 Tail. This is a string representation of an integer value.
<empty>	This is a syntactical placeholder, which indicates no input.
<encoding name>	Implementations may recognize several encodings, including Unicode and JIS. This is a US-ASCII string value. The encoding name values are case insensitive. For Unicode (ISO/IEC 10646) use the following values: UTF-8 UTF-16 ISO-10646-UCS-2 ISO-10646-UCS-4 For JIS X-0208-1997 use the following values: ISO-2022-JP Shift_JIS EUC-JP

US-ASCII and the ISO-8859 family should not be used as encoding values because they are part of the text/plain charset

`<message error>`

A message interchange error. Used to tell the remote engine that a sent message was not processed because of a message interchange error. This is an US-ASCII string representation of an integer value.

wfmc-if4-error indicates an interchange error relating to either a wfmc-if4-request or wfmc-if4-response. The `<message id>` of the wfmc-if4-error must be the same as the `<message id>` of the message in error. The <operation list> must be empty (a wfmc-if4-error message only contains <head> and <tail>).

Valid values are:

201 -- Truncated message
202 -- Modified Message (Bad Checksum)
203 -- Invalid or unsupported encoding
204 -- Expired Message
205 -- Invalid Message
206 -- Invalid ConversationID
207 -- Invalid Sequence
208 -- Invalid Timestamp
209 -- Invalid Escape Sequence

`<message id>`

Message number. This is a string representation of an integer value. See section 9.2.6 Message Id.

`<name>`

This is the name of a parameter, as defined in the operation, or the name of an extended attribute (if it appears after "&extended&"). The name values are case insensitive.

Valid values are:

ActivityID
AErrorCode
ContractID
ErrorCode
ErrorText
Language
MessageID
Name
Number
OpID
ProcessDefinitionID
ProcessID
ProductID
Profile
RootPID
SourceAID
SourceBDefName
SourceNodeID
SourcePID
SourceRoleID
SourceConversationID
SourceUserID
State
TargetNodeID
TargetBDefName
TargetPID
TargetRoleID
TargetConversationID
TargetUserID
Timestamp
Type
Value
Version

<operation>	This is the name of an operation as described in section 11 Operations. The operation values are case insensitive. Valid values are: ChangeProcessInstanceState CreateProcessInstance GetProcessInstanceAttribute GetProcessInstanceState ProcessInstanceAttributeChanged ProcessInstanceStateChanged SetProcessInstanceAttributes StartConversation StopConversation
<sequence>	There are two variants of <sequence>, one with a range and one without. Request messages do not use the range variant. The range variant is only used by response messages in which not all the operations are being responded in a single message.
<source conversation Id>	Identifies the source engine on this conversation. This is a string value. See section 9.2.4 Conversation id. The maximum length for the source conversation id is 128 bytes, and the minimum length is 1 byte. Note that "+" and "&" must not appear in <conversation id>. If they are necessary, they must be escaped.
<target conversation Id>	Identifies thetarget engine on this conversation. This is a string value. See section 9.2.4 Conversation id. The maximum length for the target conversation id is 128 bytes, and the minimum length is 1 byte. Note that "+" and "&" must not appear in <conversation id>. If they are necessary, they must be escaped.
<timestamp>	Time stamp of the message. It must use a UTC time value. See section 9.1.4.3 DateTime Format (WMTDateTime).
<value>	This is the value of the field or attribute. The type of the value is predefined by the operation.

Note that <encoding name>, <name>, and <operation> are case insensitive. For example: ErrorCode, errorcode, errorCODE, and ERRORcode are all valid <name>s.

Erklärung

Christian Ergenschäfter
Matrikel-Nr.: 330233

Hiermit versichere ich, dass ich die Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Wasserburg, den 30.11.2001 _____