Dominik Kuropka

Spezifikation und Implementierung einer verteilten persistenten Workflow-Engine auf der Basis von CORBA

Diplomarbeit

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de/ abrufbar.

Impressum:

Copyright © 1998 GRIN Verlag
ISBN: 9783638101073

Dominik Kuropka

Spezifikation und Implementierung einer verteilten persistenten Workflow-Engine auf der Basis von CORBA

GRIN Verlag

Westfälische Wilhelms-Universität Münster
Institut für Wirtschaftsinformatik
Lehrstuhl für Informatik

Diplomhausarbeit

zum Thema

Spezifikation und Implementierung einer verteilten persistenten Workflow-Engine auf der Basis von CORBA

im Fach Informatik

Themensteller: -
Betreuer: -
vorgelegt von: Dominik Kuropka
Abgabetermin: 1. April 1998

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Seit der Einführung von computergestützter Informationstechnologie in den Büro- und Produktionsalltag in den 70er Jahren ist das Ziel eines durchgehenden, computergestützten, unternehmensweiten Ablauf- und Ausführungsmanagements nicht erreicht worden. Wie läßt sich dieses Faktum in Bezug auf ein so schnell expandierendes Fachgebiet, wie die Informatik / Wirtschaftsinformatik, in dem scheinbar alles auf Grund der rasanten Computerentwicklung (spätestens in ein paar Jahren) möglich ist, in Einklang bringen?

Eine Betrachtung der Historie der elektronischen Datenverarbeitung vermag die Situation aufzuhellen: Anfang der 70er Jahre zog die computergestützte Informationstechnologie in Form von Datenbanksystemen in den Büroalltag ein. Es wurden zentralisierte Großrechnersysteme verwendet, auf denen einige wenige Applikationen installiert waren, deren Hauptaufgabe das Archivieren und statistische Auswerten von Informationen war. Ein Datenaustausch zu anderen Systemen war kaum erforderlich und auch der Zugang zu den Systemen war einigen Spezialisten vorbehalten.

Den großen Durchbruch erlebte die computergestützte Informationstechnologie in den 80er Jahren, als es möglich geworden war, die Rechner klein genug zu gestalten, so daß sie an einem Büroarbeitsplatz installiert werden konnten. Das Zeitalter der Personalcomputer begann: Textverarbeitung, Tabellenkalkulation und andere Applikationen standen auf einmal vielen Mitarbeitern zur Verfügung. Die Rechnerleistungen stiegen schnell weiter an und auch die Kommunikationstechnologie, in Form von Vernetzung der Systeme, nahm zu. Am Ende dieser Entwicklung stand eine Vielzahl von mächtigen Applikationen, die jeweils einen Inselbereich der Büroarbeit effizient unterstützen konnten, aber es fehlte ein Gesamtkonzept. Durch die verschiedene Darstellung von Informationen innerhalb der unterschiedlichen Anwendungssysteme war häufig eine manuelle Konvertierung von Daten zwischen verschiedenen Formaten nötig. Damit trat das Phänomen auf, daß sich die Produktivität der Büroarbeit trotz immer besserer technischer Werkzeuge nicht erhöhen ließ, weil an den Schnittstellen der Werkzeuge Arbeitszeit für Konvertierungen verschwendet wurde. Der Ruf nach einer gesamtheitlichen Lösung wurde laut.

Diese gesamtheitliche Lösung sollten in den 90ern die Workflow-Management-Systeme darstellen. Diese Vermögen an Hand von Modellen den Ablauf der Büroarbeit zu steuern und integerieren die im Unternehmen verwendeten Applikationen zu einem

Gesamtkonzept. Nach anfänglichen Erfolgen stellten sich jedoch auch hier Probleme ein: Die Workflow-Management-Systeme sind zu starr und unflexibel für dynamische Unternehmensbereiche. Die heutigen verfügbaren Workflow-Management-Systeme trennen strikt zwischen einer *Buildtime*- und einer *Runtime*-Umgebung [zM96]. In der Buildtime erstellt ein Modellierer ein Modell der logischen Struktur und der informationsbezogenen Zusammenhänge von Applikationen an Hand der Unternehmensstruktur. Dieses Modell wird beim Übergang zur Runtime zumeist compiliert und in ein ausführbares Programm übersetzt, welches die Ablaufsteuerung übernimmt und zu den Applikationen Schnittstellen zur Verfügung stellt.

Für einfach strukturierte und im Zeitablauf statische Aufgaben sind die heutigen Workflow-Management-Systeme gut geeignet. Probleme treten insbesondere in hoch dynamischen Bereichen auf, in denen entweder zu Beginn einer Aufgabe noch nicht der vollständige Ablauf bekannt ist oder sich laufend ändert. Auch der Versuch, Workflow-Management-Systeme durch die Einführung von Ausnahmebehandlungen flexibler zu gestalten, half nicht für alle Aufgabenstellungen. Die einzige vielversprechende Lösung ist, auf eine strikte Trennung von Runtime und Buildtime zu verzichten und somit dem Anwender zu ermöglichen, auch zur Laufzeit von Workflow-Instanzen Änderungen an diesen vornehmen zu können.

Das Ziel dieser Diplomarbeit ist die Spezifikation und Implementierung eines flexiblen Workflow-Management-Systems, basierend auf dem in [WHKS97] vorgestellten Objektmodell. Das Workflow-Management-System soll sich durch seine Verwendbarkeit auf heterogenen Systemen, Verteilung und Persistenz von Workflow-Modellen und -Instanzen auszeichnen. Zu diesem Zweck wird das System auf der Basis von *CORBA* in der Programmiersprache *Java* spezifiziert und implementiert (siehe [WHH+98]). Zur Erhaltung der Persistenz der Datenobjekte wird eine Schnittstelle zu einer Datenbank definiert und für die am Institut vorhandene *Oracle*-Datenbank programmiert.

Die Kapitel 2 und 3 führen kurz in die Thematik der Workflows und der Workflow-Modellierung ein und stellen verschiedene Ansätze für Workflow-Management-Systeme vor. Die Realisierung der Persistenz wird in Kapitel 4 ausführlich behandelt. Die Implementierung des Workflow-Management-Systems wird in Kapitel 5 erläutert und es wird beispielhaft die Darstellung von Workflows in dem System im 6. Kapitel erklärt. Abschließend wird im letzten Kapitel ein Resumee gezogen, das die Erfahrungen, die bei der Implementierung des Systems gemacht worden sind, umfaßt und einige Ideen für die zukünftige Weiterentwicklung des Systems vorstellt.

Kapitel 2

Workflow-Management

2.1 Einführung

Eine Aktivität ist eine definierte und abgeschlossene Tätigkeit, die unter Umständen von verschiedenen, aber definierten *Agenten* ausgeführt werden kann. Im Zusammenhang mit Workflow-Management ist unter einem Agenten eine Person oder ein Softwaresystem zu verstehen. Eine Menge von Aktivitäten, die in einem logischen Zusammenhang zu einander stehen und inhaltlich abgeschlossen sind, wird als *Workflow* bezeichnet.

Die Aufgabe eines Workflow-Management-Systems ist die Koordinierung von Workflows, die durch ein Workflow-Modell von außen vorgegeben werden. Ein Modell ist als ein immaterielles Abbild eines Realweltausschnittes für die Zwecke eines Subjekts zu verstehen [BS96]. Konkret bedeutet dieses für den Fall eines Workflow-Modells, daß alle Informationen über die logischen Abhängigkeiten, die sich daraus ergebende Aktivitätenreihenfolge, die zur Koordinierung benötigten Informationen und die Organisationstruktur eines Unternehmens im Workflow-Modell enthalten sein müssen.

Workflow-Modelle können durch Sprachen beschrieben werden [JBS97], [WV98]. Im folgenden wird zur Beschreibung von Modellen die in [WHKS97] beschriebene, graphbasierte Workflow-Sprache verwendet, bei der Workflow-Modelle durch geschachtelte gerichtete Graphen repräsentiert werden.

2.2 Eigenschaften von Workflows

Damit das Workflow-Management-System seiner koordinierenden Aufgabe gerecht werden kann, muß es einige relevante Laufzeit-Informationen über Workflows haben, an Hand derer Entscheidungen getroffen werden können, die den weiteren Arbeitsablauf steuern. Daher ist es sinnvoll, daß Workflows neben kontrollrelevaten Strukturen (Kontrollkonnektoren), die den logischen Ablauf vorgeben, auch Parameter haben. Über Parameter, können vorangehende Workflows auf einen Workflow Einluß nehmen,

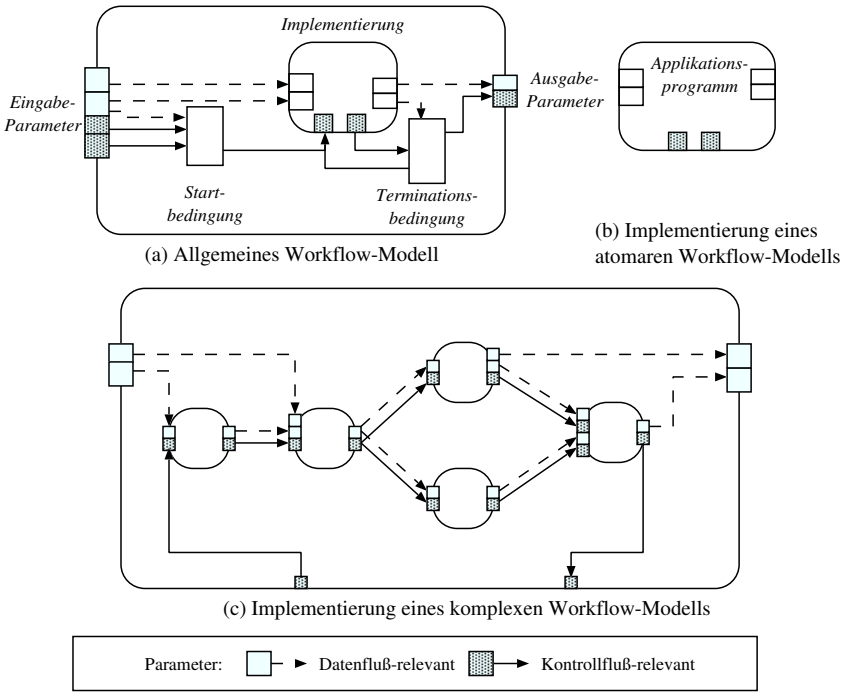

Abbildung 2.1: Struktur von Workflow-Modellen (Quelle:[WHKS97])

indem die Eingabeparameter des einen mit den Ausgabeparmetern des anderen verbunden werden. Hier stehen zwei Konzepte zur Auswahl: *globale Parameter* oder *Schnittstellenparameter*. In Workflow-Sprachen, die mit globalen Parametern arbeiten, kann jeder Workflow auf einen Parameter schreibend oder lesend zugreifen [WV98]. Sprachen dagegen, die Schnittstellenparameter verwenden, definieren für jeden Workflow eine Menge von Eingabe- und Ausgabeparametern, die über Datenfluß-Konnektoren miteinander verbunden werden [WV98]. Eine solche Modellierung wird in dieser Arbeit verwendet, weil die Datenflüsse in Workflow-Modellen so deutlicher zu erkennen sind und Workflows leichter wiederverwendbar sind, weil es nicht zu Seiteneffekten[1] auf Modellebene kommen kann.

Wie in Abbildung 2.1 (b) sichtbar, hat ein Workflow neben den datenflußrelevanten Parametern auch zwei Kontrollfluß-relevante Parameter. Diese kontrollflußrelevanten Parameter gewinnen im Zusammenhang mit komplexen Workflows (siehe nächster Abschnitt) an Bedeutung. Während Datenfluß-relevante Parameter einen beliebigen Datentypen haben können, sind Kontrollfluß-relevante Parameter immer von einem speziellen Datentyp, der den Wertebereich {*true signaled*, *not signaled*, *false signaled*} hat. Der erste (linke) Kontrollfluß-relevante Parameter des atomaren Workflows gibt an, ob ein Workflow ausgeführt werden soll. Dieses ist dann der Fall, wenn der Parameter den Wert *true signaled* hat. Der zweite (rechte) Kontrolfluß-relevante Parameter hat solange den Wert *not signaled*, wie der Workflow nicht ausgeführt wurde. Ist

[1] Mit Seiteneffekten ist hier die schon aus den Programmiersprachen bekannte Problematik von globalen Parametern (Variablen) gemeint, die insbesondere beim nachträglichen Verändern auftritt, bedingt durch die Eigenschaft von globalen Variablen, ihnen nicht ansehen zu können an welchen Stellen und im welchen kausalen Zusammenhang Zugriffe auf diese geschehen.

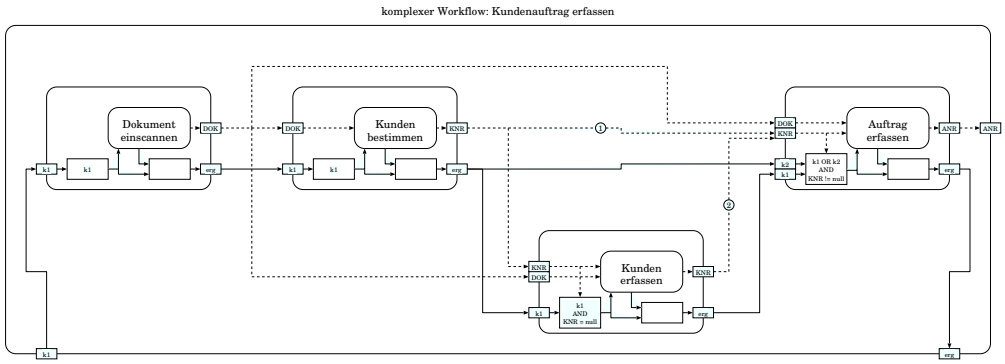

Abbildung 2.2: Beispiel für einen komplexen Workflow

der Workflow erfolgreich abgeschlossen worden, dann wird der Wert auf *true signaled*, ansonsten *false signaled* gesetzt.

2.2.1 Atomare Workflows und komplexe Workflows

Es lassen sich zwei Arten von Workflows unterschieden: *atomare* und *komplexe* Workflows. Ein atomarer Workflow zeichnet sich dadurch aus, daß er innerhalb des Workflow-Management-Systems keine sichtbare innere Struktur hat, entweder weil er sich auf Grund seiner Natur nicht weiter unterteilen läßt, oder weil eine weitere Unterteilung für das Workflow-Management-System nicht von Bedeutung ist.

Komplexe Workflows unterscheiden sich von atomaren Workflows nach außen betrachtet nicht, sie haben jedoch eine im Modell spezifizierte innere Struktur. Ein komplexer Workflow besteht aus beliebig vielen *Subworkflows*, die über Datenkonnektoren und Kontrollkonnektoren miteinander im Kontext des *Superworkflows* verbunden sind (siehe Abbildung 2.1 (c)). Um in einem Workflow-Modell verschiedene Ausführungspfade definieren zu können, erzwingt die hier verwendete Sprache, im Kontext eines Superworkflows, neben der Definition einer Terminationsbedingung, die Definition einer Startbedingung, die, wie in Abbildung 2.1 (a) gezeigt, sowohl die kontrollflußrelevanten als auch die datenflußrelevanten Eingabeparameter auswerten kann.

Abbildung 2.2 zeigt ein Beispiel für einen komplexen Workflow. Der Workflow stellt die Auftragserfassung einer Unternehmung dar. Zunächst wird ein eingegangener Auftrag eingescannt und es wird überprüft, ob der Kunde schon in der Kundendatenbank existiert. Ist das der Fall, dann gibt der Workflow „Kunden bestimmen“die Kundennummer zurück, ansonsten wird keine Kundennummer zurückgegeben (also *null*). Ist keine Kundennummer zurückgegeben worden, so startet der Workflow „Kunden erfassen“und gibt als Ergebnis die Kundennummer des neuen Kunden zurück. Da der Parameter „KNR“des Workflows „Auftrag erfassen“von zwei Datenflußkonnektoren referenziert wird, muß dieser Zugriff synchronisiert werden. Dies geschieht über Prioritäten (kleiner Kreis mit Ziffer an den Datenflußkonnektoren). Da die Priorität für den Datenfluß von „Kunden erfassen“höher ist als von „Kunden bestimmen“ist die Kundennummer von „Kunden erfassen“ausschlaggebend, sofern dieser Workflow gestartet wird. Zu allerletzt werden die Daten mit der gegebenen Kundennummer erfaßt und der komplexe Workflow ist mit der Auftragsnummer als Rückgabewert beendet.

2.2.2 Agenten- / Rollenkonzept

Wird an Hand eins Workflow-Modell ein konkreter Workflow erstellt, spricht man vom Instantiieren eines Workflow-Modells. Ist eine Modell-Instanz erstellt worden, dann kommt irgendwann der Zeitpunkt, an dem atomare Workflows ausgeführt werden sollen. Da ein Workflow zu seiner Ausführung einen Agenten braucht, muß ein geeigneter zunächst gefunden werden. Diesen Vorgang nennt man *Rollenauflösung*.

Wie schon zu Beginn dieses Kapitels erwähnt, muß in einem Workflow-Modell auch die Organisationsstruktur eines Unternehmens abgebildet werden, um geeignete Agenten zur Ausführung von atomaren Workflows zu finden, die sowohl die Kompetenz, als auch die Fähigkeit haben, ein bestimmte Aktivität auszuführen. Zur Darstellung der Fähigkeiten und Kompetenzen von Agenten verwendet man das Konzept der *Rollen*. Ein Agent hat verschiedene Rollen, die er einnehmen kann und ein Workflow steht zu Rollen in Beziehung, die ein Agent haben muß, um ihn ausführen zu können.

Üblicherweise geht die Rollenauflösung so vor sich, daß Agenten, sofern diese human sind, über Workflows, die von ihnen ausgeführt werden dürfen, in Form einer *Worklist* informiert werden. In diesem Zusammenhang ist es sinnvoll, noch weitere Eigenschaften für Workflows zu modellieren, um zum Beispiel zu verhindern, daß Workflows, aus welchen Gründen auch immer, überhaupt nicht, oder zu spät ausgeführt werden. Hierzu gibt es insbesondere die Konzepte der *Priorisierung* und der *Deadline*. Bei der Priorisierung bekommt jeder Workflow eine eigene statische Priorität, die seine Wichtigkeit ausdrückt. Dieses Konzept kann flexibilisiert werden, indem sich die effektive Priorität eines atomaren Workflows von seiner eigenen und der seines Subworkflows berechnet; sehr sinnvoll im Zusammenhang mit Aufgaben, in denen sowohl unwichtige als auch sehr hochpriorisierten Tätigkeiten anfallen. Das zweite Konzept der Deadline sieht einen spätesten Zeitpunkt vor, an dem ein Workflow bearbeitet oder beendet sein muß. Dieses Konzept ist dynamisch, weil sich die Workflow-Wichtigkeit im Verlaufe der Zeit ändern kann. Besser geeignet als die Priorisierung ist dieses Konzept insbesondere bei Terminsachen.

2.3 Anforderungen an ein Workflow-Management-System

Bei Workflow-Management-Systemen handelt es sich um sehr komplexe Softwareprodukte, die nicht ad hoc implementiert und nachträglich ohne weiteres in ihrer Funktionalität erweitert werden können. Vor der Implementierung eines Workflow-Management-Systems sind vielmehr zunächst Überlegungen theoretischer Art notwendig um ein in der Praxis anwendbares System zu erstellen. Dies ist darin begründet, daß ein Workflow-Management-System nicht eine vollständige Applikation im herkömmlichen Sinne ist, sondern vielmehr eine Plattform darstellt, in die bestehende und zukünftige Systeme eingebunden werden sollen. Das heißt, daß das Workflow-Management-System als Mittler zwischen Applikationen verstanden werden kann, der aus einer Menge von Einzelapplikationen eine zusammenhängende, unternehmensweite Anwendung zur Unterstützung der Unternehmensziele schafft. Da sich die Ziele einer Unternehmung kurzfristig und langfristig ändern oder an die Zeit anpassen müssen, ist nicht nur das Workflow-Management-System, sondern auch die verwendeten

Applikationen Änderungen unterworfen. Daher müssen von vorneherein Überlegungen angestellt werden, das System „zukunftskompatibel“zu gestalten.

2.3.1 Workflow-Meta-Modell-bezogene Anforderungen

Die theoretischen Überlegungen zu einem Workflow-Management-System umfassen insbesondere das *Workflow-Meta-Modell*. Die Definition des Workflow-Meta-Modells legt fest welche Workflow-Modelle das System erlaubt und verarbeiten kann, somit ist ein konkretes Workflow-Modell einer Unternehmung eine Instanz des Workflow-Meta-Modells [JBS97].

Aus diesem Grunde ist eine wohlüberlegte Modellierung des Workflow-Meta-Modells von entscheidender Bedeutung für das Workflow-Management-System. Ein Workflow-Meta-Modell sollte die folgenden Anforderungen erfüllen, um „Zukunftskompatibilität“zu gewährleisten [Jab96]:

Erweiterbarkeit: Workflow-Management-Systeme werden in Unternehmen eingeführt, um verschiedene Anwendungsbereiche zu integrieren, die durch unterschiedlichste Workflows charakterisiert sind. Daher es ist unmöglich von Anfang an bei der Systemspezifikation alle benötigten Eigenschaften und Modellelemente von Workflows vorherzusehen, so daß das Workflow-Meta-Modell in der Art flexibel gestaltet sein muß, daß auch neue Workflow-Typen oder Kontrollkonstrukte nachträglich in das Modell eingefügt werden können.

dynamische Anpassungsfähigkeit: Das Workflow-Meta-Modell sollte die Möglichkeit vorsehen, bereits gestartete Workflows nachträglich zu ändern. Das heißt, es muß für einen gestarteten komplexen Workflow zum Beispiel möglich sein, Subworkflows, die noch nicht ausgeführt worden sind, zu überspringen oder neue Subworkflows einzufügen etc.

Gründe, die eine solche Flexibilität erzwingen, sind zum einen die Problematik von Fehlern bei der Ausführung von Workflows. Selbst ein Workflow-Meta-Modell, welches eine sehr ausführliche Fehlerbehandlung für Workflow-Modelle zur Verfügung stellt, wird niemals alle möglichen Fehlerquellen abfangen können, allein auf Grund der Tatsache, daß die Fehlerursache sich außerhalb des Workflow-Management-Systems befinden kann. Des weiteren können Änderungen an der Umgebung von Unternehmen kurzfristiger oder langfristiger Art auch Änderungen an Workflow-Modellen erzwingen.

Wiederverwendbarkeit: Auf Grund der Tatsache, daß der Aufwand für die Erstellung eines Workflows ein nicht zu unterschätzender Kostenfaktor ist, sollte das Workflow-Meta-Modell eine Wiederverwendung von Workflows und anderen Modell-Elementen gestatten. Das bedeutet zum Beispiel für einen Subworkflow, daß er keine Annahmen über irgendwelche Eigenschaften seines Superworkflows treffen sollte, um problemlos in den Kontext eines anderen Superworkflows eingebunden werden zu können.

Offenheit: Workflows können die verschiedensten Bereiche einer Unternehmung umspannen, in denen die unterschiedlichsten Hard- und Softwareplattformen ihre Verwendung finden. Daher sollte ein Workflow-Modell keine Annahmen über die Beschaffenheit von Applikationen treffen, um keine Restriktionen in den Anbindungsmöglichkeiten von Anwendungen zu haben. Das bedeutet, daß ein Workflow-Modell eine geeignete Schnittstelle zwischen der Workflow-Repräsentation einer Applikation und der Applikation selbst zur Verfügung stellen muß.

2.3.2 Implementierungsbezogene Anforderungen

Neben den theoretischen dürfen die praktischen Überlegungen zur Implementierung des Workflow-Management-Systems nicht vernachlässigt werden, damit das System auch ergonomische Aspekte erfüllt. Leider sind die implementierungsbezogenen Anforderungen und die Workflow-Meta-Modell-bezogenen Anforderungen nicht vollständig orthogonal zueinander, so daß bei der Entwicklung des Systems eine enge Kooperation zwischen Meta-Modell-Designer und den Programmierern notwendig ist. Folgende Anforderungen gilt es zu berücksichtigen [JBS97]:

Offenheit: Wie bei den Workflow-Meta-Modell-bezogenen ist die Offenheit des Systems bei den implementierungsbezognenen Aspekten von entscheidender Bedeutung. Unter Offenheit im Kontext der Systemimplementierung ist jetzt allerdings nicht nur die Fähigkeit des Systems zur Einbindung von verschiedenen Anwendungen gemeint, sondern auch die Möglichkeit das gesamte System (mit Anpassungen) auf eine Vielzahl von Plattformen zu portieren. Insbesondere im Zusammenhang mit der Offenheit von Workflow-Management-Systemen ist die Verwendung von Middleware-Produkten, wie zum Beispiel CORBA, sinnvoll.

Skalierbarkeit: Da ein Workflow-Management-System im Verlauf seines Einsatzes in der Unternehmung bei erfolgreicher Anwendung immer mehr Bereiche umfassen wird, muß von vorneherein mit einer steigenden Anzahl von Anfragen an das System gerechnet werden. Daher ist es enorm wichtig, das System skalierbar, das heißt an die Zahl der Anfragen anpassbar, zu gestalten. In diesem Zusammenhang ist es sinnvoll, über die Verteilung des Systems zu reden (siehe übernächster Punkt).

Zuverlässigkeit: Auf Grund der zentralen Bedeutung des Workflow-Management-Systems für eine Unternehmug ist ein zuverlässiger und sicherer Betrieb unerläßlich. Das System muß nicht nur Persistenz und Fehlertoleranz aufweisen, sondern muß auch eine geringe Ausfallzeit haben. Das bedeutet, die Workflow-Engine muß entweder auf mehreren Rechnern repliziert oder verteilt sein.

Verteilbarkeit: Auch wenn die Verteilung eines Systems nicht gerade zu den einfachsten Disziplinen der Informatik gehört, bietet sie sich im Zusammenhang mit Workflows auf Grund ihrer Struktur an. Atomare Workflows werden

weitestgehend autonom ausgeführt und könnten somit als eigener Knoten des verteilten Systems gesehen werden. Auf diese Weise ist, eine genügende Anzahl von Rechner vorausgesetzt, eine Skalierbarkeit des Systems gegeben, wenn die Workflows möglichst an dem Ort ausgeführt werden, an dem sich auch der Agent befindet.

Ein Nachteil der Verteilung im Zusammenhang mit Workflow-Management-Systemen ist das Monitoring. Da beim Monitoring von Workflows im allgemeinen sehr viele Daten über Subworkflows etc. benötigt werden, hat bei dieser Problematik der zentralisierte Ansatz naturgemäß Vorteile.

2.4 Kommerzielle Workflow-Management-Systeme

Es gibt eine Vielzahl kommerzieller Workflow-Management-Systeme auf dem Markt, von denen hier zwei kurz vorgestellt werden sollen:

2.4.1 FlowMark von IBM

FlowMark ist ein seit 1994 von der Firma IBM distributiertes, zentrales Workflow-Management-System. Das System ist auf Grund seiner Client-Server Architektur zentral, weil es aus logischer Sicht gesehen, genau einen Server gibt (der physisch gespiegelt sein kann). Das heißt, daß alle Workflow-Daten auf einem Knoten vereinigt sind. FlowMark kann auf heterogenen Plattformen, auf Grund der breiten Basis an unterstüzten Systemen (OS/2, AIX, Windows, HP-UX), verwendet werden.

In FlowMark werden *Prozesse* (komplexe Workflows) in Form von gerichteten Graphen modelliert, die aus *Aktivitäten* (atomaren Workflows) bestehen, die über *Kontrollflußkonnektoren* miteinander logisch verknüpft werden. Aktivitäten haben jeweils genau einen *Input-* und einen *Output-Container*, die über *Datenflußkonnektoren* mit Containern anderer Aktivitäten im Prozeß verknüpft sein können [zM96].

FlowMark implementiert strikt das Runtime-Buildtime Konzept [RzM96]. Das heißt Workflow-Modelle werden zur Buildtime in einem anderen Datenformat gehalten als zur Runtime. Zur Konvertierung eines Buldtime-Modells wird ein Compiler verwendet, der das Modell in ein fertiges Programm übersetzt. Dieses wirkt sich sehr restriktiv auf die dynamische Veränderbarkeit von Workflows aus. Bei FlowMark ist es nicht möglich in den Ablauf von Workflow-Instanzen nach ihrem Start einzugreifen oder ihr Modell zu ändern. Die Veränderungen an einem Workflow-Modell werden erst dann wirksam, wenn sich der Agent, der einen neuen Workflow startet, in das System erneut einloggt. Dafür bietet FlowMark die Möglichkeit Prozeßadministratoren zu bestimmen, die im Falle eines Scheiterns von Workflow-Instanzen über die Situation informiert werden. Daneben ist ein statisches Exceptionhandling für Aktivitäten möglich.

Die Wiederverwendung von Workflows ist in FlowMark auf Grund der direkten Zuordnung von Containern zu Workflows gut gestaltet. Einzig die Tatsache, daß ein Workflow genau zwei Container hat (einen Input- und einen Outputcontainer) wirkt

etwas befremdlich. Ein Container enthält mehrere Datenfelder, die bei einem Datenfluß von einem Input- zu einem Outputcontainer über eine *Mapping*-Operation einander zugeordnet werden (siehe Meta-Modell FlowMark in [zM96]). Somit kann einem Workflow-Modell nicht auf einen Blick angesehen werden, welche Datenfelder eines Outputcontainers in welche Datenfelder eines Inputconatiners übertragen werden.

2.4.2 WorkParty von Siemens

WorkParty ist, wie FlowMark, ein zentrales Workflow-Management-System, daß 1992 von der Siemens Nixdorf AG für die Windows-Plattform entwickelt wurde. Ein *Prozeß* (komplexer Workflow) in WorkParty ist ein gerichteter Graph, dessen Knoten aus Aktivitäten bestehen, die durch Konnektoren miteinander verbunden sind. Die Konnektoren können nicht frei platziert werden, sondern es werden dem Modellierer vordefinierte Ablaufkonstrukte zur Verfügung gestellt, die die Modellierung eines syntaktisch korrekten Workflowmodells sicherstellen [zM96].

Auch in WorkParty wird zwischen einer Runtime und einer Buldtime unterschieden, dennoch ist das verwendete Konzept flexibler als bei FlowMark. Wie bei FlowMark können auch bei WorkParty schon gestartete Workflow-Instanzen nicht mehr beeinflußt werden, aber es ist bei WorkParty möglich, in Workflow-Modellen Aktivitäten leer zu lassen. Wird eine leere Aktivität ausgeführt, dann hat der Benutzer zur Laufzeit die Möglichkeit, entweder ein schon fertig modelliertes Workflow-Template an dieser Stelle ausführen zu lassen, oder *ad hoc* einen eigenen Workflow zu modellieren (der natürlich auch keine Aktivitäten enthalten darf, was dem Überspringen der Leer-Aktivität entspricht).

Die Wiederverwendung von Workflow-Modellen ist mit WorkParty möglich, aber auf Grund der Art der Parameterdarstellung im Workflow-Meta-Modell nicht gut gelöst. In WorkParty werden Daten in *Akten* gespeichert, die für jede Workflow-Instanz neu erzeugt werden, aber die für alle Aktivitäten der Workflow-Instanz global sind. Das heißt, jede Aktivität ist in der Lage diese Akten zu beschreiben oder zu lesen, ohne daß dieses explizit über Datenflußkonnektoren ausgedrückt wird. Somit ist der Modellierer bei der Wiederverwendung gezwungen, darauf zu achten, daß das wiederverwendete Workflow-Modell mit seinen verwendeten Akten nicht in ungewünschte Konflikte mit den Akten der anderen schon vorhandenen Workflows gerät. Insbesondere bei nachträglichen Änderungen an Modellen, die von anderen Benutzern erstellt worden sind, ist mit gefährlichen Seiteneffekten auf Grund von zufälligen, nicht semantisch begründeten, gleichen Aktenbenennungen möglich (siehe Workflow-Meta-Modell zu WorkParty in [zM96]).

2.5 Der erste WASA Prototyp

Im Jahre 1997 ist am Lehrstuhl für Informatik des Institutes für Wirtschaftsinformatik der Westfälischen Wilhelms-Universität Münster im Rahmen einer Diplomarbeit [Wit97] der erste WASA Prototyp entstanden. Dieser Prototyp zeichnet sich dadurch aus, daß er sowohl dynamische Änderungen an Workflows erlaubt, als auch für heterogene Rechnerumgebungen geeignet ist.