Informatik Forsch. Entw. (1999) 14: 203–217

Informatik
Forschung und Entwicklung

Verteilungsmodelle für Workflow-Management-Systeme – Klassifikation und Simulation

Thomas Bauer, Peter Dadam

Abteilung Datenbanken und Informationssysteme, Universität Ulm, Oberer Eselsberg, 89069 Ulm (e-mail: {bauer,dadam}@informatik.uni-ulm.de)

Eingegangen am 12. März 1999 / Angenommen am 8. Oktober 1999

Zusammenfassung. In unternehmensweiten Workflow-Management-Systemen (WfMS) kann die von der WF-Engine zu bewältigende Last sehr groß werden. Außerdem werden hohe Anforderungen an die Verfügbarkeit eines solchen Systems gestellt. Deshalb wurden in der Literatur zahlreiche Architekturen für skalierbare WfMS vorgeschlagen, die auf unterschiedlichen Verteilungsmodellen für die WF-Engine basieren. Im vorliegenden Beitrag werden diese Verteilungsmodelle analysiert, verglichen und klassifiziert. Aufbauend auf diese Klassifikation wird für zwei Beispielszenarien die bei den verschiedenen Verteilungsmodellen entstehende Last simuliert und verglichen.

Schlüsselwörter: Workflow-Management, Architekturen, verteilte Ausführung, Skalierbarkeit, Simulation

Abstract. In enterprise-wide workflow management systems (WfMS) the workflow engine may have to cope with a very high load. In addition, the availability of such a system must be high. Many architectures for scalable WfMS have been proposed in the literature, which are based on different distribution models of the workflow engine. These distribution models are analyzed, compared, and classified. Based on this classification, two example scenarios are used to simulate and compare the load resulting for the different distribution models.

Key words: Workflow management, architectures, distributed execution, scalability, simulation

CR Subject Classification: H.4.1, H.1.0, C.2.4

1 Einleitung

WfMS ermöglichen die rechnerunterstützte Ausführung von Geschäftsprozessen in einer verteilten Systemumgebung. Aufgrund von wirtschaftlichen Zwängen ist das Interesse an solchen Systemen in den letzten Jahren stetig gewachsen. Ihr entscheidender Vorteil ist, daß sie helfen, große Anwendungssysteme überschaubarer zu gestalten. Dazu wird der applikationsspezifische Code der verwendeten Anwendungen von der Ablauflogik („Prozeßlogik“) getrennt. Anstelle eines großen monolithischen Programmpakets erhält man nun, zumindest logisch gesehen, einzelne Anwendungsprogramme („Anwendungskomponenten“), welche die auszuführenden Aktivitäten repräsentieren. Ihre Abfolge wird in einer separaten Kontrollfluß-Definition festgelegt, welche die Ausführungsreihenfolge (Sequenz, Verzweigung, Parallelität, Schleifen) der einzelnen Aktivitäten bestimmt. Häufig wird auch der Datenfluß zwischen den Aktivitäten explizit modelliert. Das WfMS sorgt dafür, daß nur solche Aktivitäten ausgeführt werden können, die der Ablauflogik zufolge zur Bearbeitung anstehen. Diese werden in die Arbeitslisten autorisierter Bearbeiter eingefügt. Welche Benutzer zur Bearbeitung einer bestimmten Aktivität autorisiert sind, wird meist durch eine Rolle und evtl. zusätzlich eine Organisationseinheit (OE, z.B. Abteilung, Zweigstelle o.ä.) festgelegt, welchen die entsprechenden Bearbeiter angehören müssen. Dabei ist es durchaus möglich, daß mehrere Benutzer ermittelt werden, die für die Bearbeitung einer bestimmten Aktivität in Frage kommen.

Wir gehen in dieser Arbeit von dem folgenden *Referenzmodell* für WfMS aus: Ein WfMS besteht aus einer WF-Engine und den WF-Clients. Die *WF-Engine* steuert die Ausführung der WF-Instanzen entsprechend dem in den WF-Vorlagen modellierten Kontrollfluß und verwaltet die WF-Daten. Eine WF-Engine besteht aus einem oder mehreren *WF-Servern*, die jeweils über eine eigene *WF-Datenbank* (WF-DB) verfügen. In dieser sind die folgenden WF-Daten gespeichert: die (weitgehend statischen) WF-Vorlagen (Schemainformation), der aktuelle Zustand der WF-Instanzen dieses Servers, die Werte der Datenelemente dieser WF-Instanzen und evtl. Information über alte Zustände. Der *WF-Client* ermöglicht einem Benutzer die zugehörige Arbeitsliste anzuzeigen. Wenn dieser daraus eine Aktivitäteninstanz auswählt, so wird das entsprechende Aktivitätenprogramm gestartet.

Im folgenden soll das zugrundegelegte *Ausführungsmodell* erläutert werden. Dabei wird nur auf Aspekte eingegangen, die für das weitere Verständnis dieser Arbeit wichtig sind. Wir betrachten die Ausführung eines (Ausschnitts eines) WF, der aus der Sequenz der Aktivitäten A und B

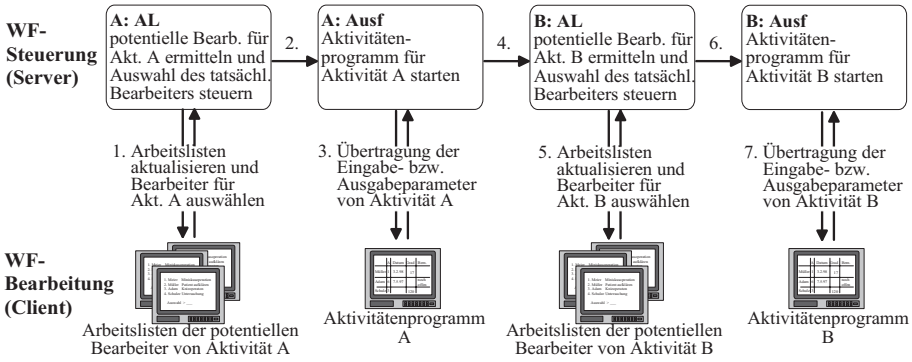

Abb. 1. Steuerung und Bearbeitung eines WF, der aus den Aktivitäten A und B besteht

besteht. In Abb. 1 sind die hinsichtlich der späteren Diskussion interessanten Schritte aufgeführt. Der WF-Server ermittelt die potentiellen Bearbeiter von Aktivität A und fügt Aktivität A in deren Arbeitslisten ein. Ein Benutzer, der diese Aktivität bearbeiten möchte, wählt sie aus seiner Arbeitsliste aus; der WF-Server synchronisiert diesen Vorgang (A: AL). Anschließend startet der WF-Server oder Client das entsprechende Aktivitätenprogramm für diesen Benutzer (A: Ausf). Dies findet i.d.R. zumindest für den Client-Teil des Programms auf dem Rechner dieses Bearbeiters statt. Zum Starten müssen die Eingabedaten zu dem Programm übertragen werden. Nach Beendigung des Programms und der Übertragung der Ausgabedaten zum Server erfolgt derselbe Ablauf für Aktivität B (B: AL und B: Ausf).

Im einfachsten Fall würden alle diese Aktionen auf demselben Rechner ausgeführt werden. Dieser Fall ist aber wenig sinnvoll, da die Benutzer des WfMS und die verwendeten Anwendungen meist räumlich verteilt sind. Deshalb findet die WF-Bearbeitung in der Regel verteilt statt. Die Last für die WF-Engine ist bei großen (unternehmensweiten) Anwendungen sehr groß, da nach Beendigung jeder Aktivität nicht nur die Nachfolgeaktivitäten bestimmt werden müssen, sondern auch alle für deren Bearbeitung geeigneten Benutzer. Dies erfordert einen hohen Aufwand, da zur Festlegung der potentiellen Bearbeiter einer Aktivität i.d.R. komplexe Auswahlprädikate verwendet werden können. Für jede Aktivitäteninstanz und jeden Benutzer muß ein solcher Ausdruck ausgewertet werden. Anschließend müssen auch noch die Arbeitslisten der hierbei ermittelten Bearbeiter aktualisiert werden. Der hierdurch entstehende Aufwand führt wegen mangelnder Skalierbarkeit bei vielen der heute kommerziell verfügbaren WfMS dazu, daß diese bereits bei relativ kleinen Benutzerzahlen an ihre Leistungsgrenze stoßen [GHS95]. Um dem abzuhelfen, wird häufig auch die WF-Steuerung verteilt, d.h., es werden mehrere WF-Server verwendet. Die Art und Weise, wie die Aufgaben auf die verschiedenen WF-Server verteilt werden, wird in diesem Beitrag als *Verteilungsmodell* bezeichnet. Der Vergleich der verschiedenen Verteilungsmodelle ist der zentrale Aspekt dieser Arbeit. Da jede der in Abb. 1 dargestellten Aktionen potentiell auf einem anderen Rechner ausgeführt werden kann, stellen die Pfeile 1 bis 7 mögliche Kommunikationen dar. In [BD98a] wurde für einen Versicherungs-WF mit 150 Sachbearbeitern und moderaten Annahmen über Parametergrößen berechnet, daß sich alleine schon durch den Parametertransfer zu und von den Aktivitätenprogrammen ein Datenvolumen von 8 Mbit/sec ergibt. Damit ist ein Ethernet-basiertes Local Area Network (LAN) schon durch die Kommunikation zwischen der WF-Engine und den Clients überlastet. Ein ähnliches Beispiel findet sich in [BD97]. Solche Lastprobleme treten vor allem dann auf, wenn die Parameterdaten der Aktivitätenprogramme sehr umfangreich sind (z.B. Multimediadaten wie Bilder und Videos, eingescannte Briefe, mit Textverarbeitungssystemen erzeugte Dokumente), da diese für jede einzelne Aktivitäteninstanz über das LAN transportiert werden müssen. Insbesondere bei Multimediadaten können auch moderne Netzwerktechnologien an ihre Leistungsgrenze stoßen. Deshalb wird im folgenden nicht nur die Belastung der WF-Server, sondern auch die des Kommunikationssystems betrachtet. Die Minimierung der Kommunikationslast ist noch wichtiger, wenn das WfMS über ein Wide Area Network (WAN) weiträumig verteilt eingesetzt wird. Durch ein geeignetes Verteilungsmodell kann dann nicht nur die Überlastung der WF-Server und des Kommunikationssystems verhindert werden, sondern es können auch die Antwortzeiten und die Verfügbarkeit verbessert werden. Falls sich nämlich der WF-Server „näher“ bei den Bearbeitern der Aktivitäten befindet, so ist die WF-Ausführung weniger auf das WAN angewiesen.

Viele verschiedene Aspekte eines WfMS haben Einfluß auf die erzeugte Last. Durch ein geeignetes Verteilungsmodell ist es aber möglich, diese Last so aufzuspalten und auf die einzelnen Systemkomponenten (Server, Teilnetze und Gateways) zu verteilen, daß eine Skalierbarkeit des Gesamtsystems durch Hinzufügen und Einbeziehung entsprechender Systemkomponenten erzielbar wird. Die folgenden Betrachtungen sind dabei im wesentlichen unabhängig davon, wie z.B. die Aktualisierung der Arbeitslisten und der Datenfluß konkret realisiert sind (siehe [BD98a] für eine Übersicht), ob Backup-Server zur Erhöhung der Verfügbarkeit eingesetzt werden (siehe z.B. [KAGM96]), welches Transaktionskonzept [Elm92, Ley97] zugrunde gelegt wird und ob dynamische Änderungen an laufenden WF-Instanzen [RD98] unterstützt werden oder nicht. Alle diese Aspekte schlagen sich letztlich „nur“ in entsprechenden Kostenfaktoren in den einzelnen Kostenmodellen nieder und werden im folgenden deshalb nicht mehr explizit diskutiert.

Das nächste Kapitel befaßt sich mit der Analyse und Klassifikation von Verteilungsmodellen für WfMS. Kapitel 3 beschreibt die Simulation der unter den verschiedenen Verteilungsmodellen entstehenden Last. Kapitel 4 faßt die gewonnenen Erkenntnisse zusammen.

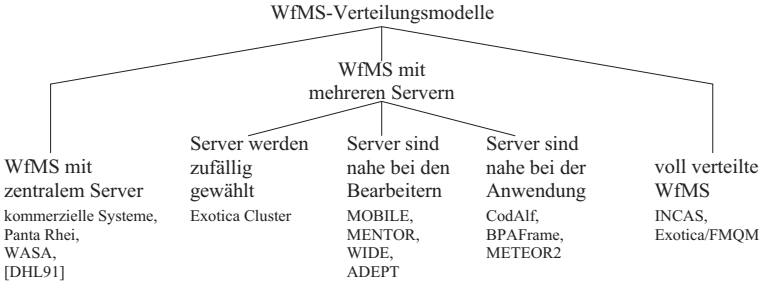

Abb. 2. Verteilungsmodelle für WfMS und Einordnung entsprechender Systeme

2 Verteilungsmodelle

Skalierbarkeit und hohe Verfügbarkeit von WfMS wird durch Replikation von Systemkomponenten und ihre geeignete Verteilung auf mehrere Rechner und Teilnetze erreicht. Die dafür möglichen Verteilungsmodelle werden im folgenden beschrieben, klassifiziert und analysiert und ihnen konkrete kommerzielle Systeme sowie Vorschläge aus dem Bereich der Forschung zugeordnet. Die Konzepte werden hierbei bezüglich ihres Leistungsverhaltens miteinander verglichen. Dazu wird untersucht, wie sich die Last für die Steuerung der Prozeßschritte und das Aktualisieren der Arbeitslisten in der Belastung der Server und der Teilnetze niederschlägt. In einem unternehmensweiten WfMS können diese Werte sehr groß werden. Damit keine Komponente überlastet wird, muß ihre Belastung unter die jeweils vorgegebene Maximalkapazität der Komponente gedrückt werden können. Dies ist z.B. dann gegeben, wenn die Last in jeder Komponente $O(\text{Gesamtlast}/x)$ beträgt, wobei x eine konfigurierbare Größe sein muß, wie etwa die Anzahl der Server-Replikate im System. Ist diese Bedingung durch ein Verteilungsmodell erfüllt, so kann dieses als skalierbar bezeichnet werden.

Abbildung 2 zeigt eine Kategorisierung der Verteilungsmodelle und die Einordnung konkreter Systeme. Die beiden Extrempunkte des Spektrums sind Systeme mit nur einem zentralen Server und voll verteilte Systeme, bei denen jede Komponente Serveraufgaben wahrnimmt. Dazwischen liegen Systeme, die mehr als einen Server verwenden, aber trotzdem noch die Trennung zwischen Client und Server aufrechterhalten, so daß es weniger Server als Clients (Benutzerrechner) im System gibt. Die Übergänge zwischen den Klassen sind fließend, und natürlich gibt es Systeme, die Mischformen darstellen. Im folgenden werden die Verteilungsmodelle aber in ihrer Reinform dargestellt, um die Unterschiede deutlich herausarbeiten zu können.

2.1 WfMS mit zentralem Server

Systeme dieser Kategorie verwenden eine zentrale Serverkomponente (vgl. Abb. 3). Dies ist zumindest eine zentrale WF-Datenbank mit nur einem DB-Server. Meist wird auch nur ein WF-Server verwendet, es gibt aber auch Systeme, bei denen sich mehrere WF-Server die zentrale Datenbank teilen.

Die Leistungsfähigkeit eines solchen Systems ist im wesentlichen durch den zentralen Server bestimmt bzw. beschränkt, da der Server die volle Last für die Ausführung der Aktivitäten und für die Aktualisierung der Arbeitslisten zu tragen hat. Entsprechend stark wird das Teilnetz des DB-Servers belastet. Ein solches System ist deshalb nur für eine relativ kleine Anzahl von Benutzern (einige Dutzend) geeignet. Werden mehrere WF-Server verwendet, so läßt sich die Last zwischen ihnen aufteilen, das zentrale DBMS bleibt aber weiterhin ein potentieller Flaschenhals.

Die Beseitigung bzw. Abmilderung dieses potentiellen Engpasses erfordert relativ rasch den Einsatz von (teuren) Hochleistungssystemen, z.B. in Form von Mehrprozessorsystemen und von Mehrrechner-DBMS. Ob sich damit die erforderliche Leistung erzielen läßt, hängt stark davon ab, inwieweit man die Steuerung der einzelnen WF-Instanzen sowie deren Daten so auf die beteiligten Teilsysteme verteilen kann, daß eine gute Lastbalancierung erreicht wird und wenig Zugriffskonflikte auftreten. Dies führt zur selben Problemstellung wie die Verteilung der Gesamtlast auf mehrere (dezentrale) WF-Server. Wir werden hierauf in Abschnitt 2.2 näher eingehen. Aber selbst wenn man dadurch unter günstigen Voraussetzungen vermeiden kann, daß der zentrale WF-Server bzw. das zentrale DBMS zum Flaschenhals wird, so bleibt als Nachteil, daß das Teilnetz des (zentralen) Hochleistungssystems zum Engpaß werden kann. Hinzu kommt, daß weit(er) entfernte Benutzer in der Regel mit schlechteren Antwortzeiten bedient werden.

In die Kategorie „zentraler Server“ fallen viele kommerzielle Systeme wie **WorkParty** [Sie95], **ProMInanD** [Kar94] oder **FlowMark** [IBM96] bis Version 2.1. Sie verwenden einen zentralen WF-Server oder zumindest eine zentrale WF-Datenbank (mit zentralem DB-Server), die zur Ausführung jedes Teilschritts benötigt wird. Auch einige Forschungsansätze (z.B. **Panta Rhei** [EG96], **WASA** [WHKS98], **[DHL91]**) setzen eine zentrale Kontrollinstanz voraus.

FlowMark [IBM96] ermöglicht ab Version 2.2, einen Subprozeß in einem anderen FlowMark-System (Local Domain) ausführen zu lassen. Dadurch entsteht eine Mischform aus einer zentralen Architektur und einer Architektur, bei der mehrere Server verwendet werden, die jeweils nahe bei den jeweiligen Bearbeitern angesiedelt sind (siehe Abschnitt 2.2.2). Eine solche Kooperation von zentralen WfMS ist nützlich, wenn ein Teil eines Prozesses von anderen Bearbeitern bzw. in anderen OE ausgeführt wird als der Rest. Dieser Teil wird dann von dem anderen WfMS kontrolliert. Es ist hierbei zu beachten, daß es sich bei den verschiedenen Local Domains um getrennte Systeme handelt, die lediglich kooperieren. Dies zeigt sich zum Beispiel daran, daß Benutzer, die Aktivitäten in verschiedenen Abschnitten eines solchen aufgespaltenen Prozesses ausführen sollen, in allen zugehörigen Systemen bekannt gemacht werden müssen.

Im Optimalfall verteilt sich die Last gleichmäßig auf alle beteiligten (zentralen) Systeme. Auch der Aufwand für das Aktualisieren der Arbeitslisten wird aufgeteilt, wenn die Benutzer jeweils nur in wenigen Local Domains aktiv und gleichmäßig auf sie verteilt sind. Der Aufwand für das Ändern einer Verteilung ist allerdings sehr groß, da entsprechende Teilprozesse für die einzelnen Teilsysteme modelliert werden müssen, d.h., das WF-Modell verändert werden muß.

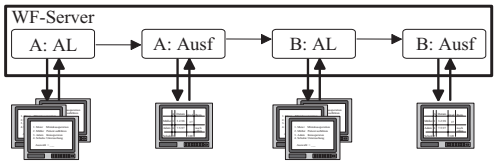

Abb. 3. Ein zentraler WF-Server bearbeitet alle WF-Instanzen des WfMS

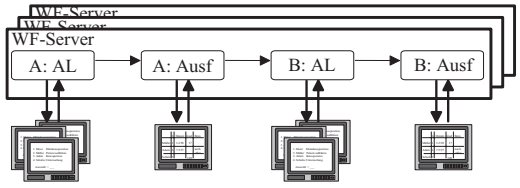

Abb. 4. Der Cluster für eine WF-Instanz wird bei deren Start zufällig gewählt

2.2 WfMS mit mehreren Servern

Systeme, bei denen der WF-Server mehrfach und auf verschiedene Maschinen verteilt vorhanden ist, bilden die im folgenden beschriebene Kategorie. Für die Verteilung der Server gibt es drei Ansätze: Man wählt den WF-Server nach dem Zufallsprinzip oder man versucht, aus dem Ort des Servers Vorteile zu ziehen. Hier gibt es zwei Möglichkeiten: Man kann für die Steuerung einer Aktivität den Server verwenden, der sich bei den vorgesehenen Bearbeitern befindet oder den, der sich auf dem Knoten der zugehörigen Anwendung befindet.

2.2.1 Server werden zufällig gewählt

Bei diesem Ansatz wird der Server für eine WF-Instanz zufällig gewählt (Abb. 4). Die Server sind identische Replikate der WF-Engine und bestehen aus einer WF-Datenbank und einem WF-Server (WF-Cluster). Alle Prozeßtypen können von jedem Cluster ausgeführt werden; eine Prozeßinstanz verbleibt in dem Cluster, in dem sie gestartet wurde. Ein Cluster ist nicht an eine OE gebunden.

In der WF-Datenbank sind die replizierte Schemainformation, die Instanzen des Clusters und der diesen Cluster betreffende Teil der Arbeitslisten aller Benutzer gespeichert. Jeder Client muß eine Verbindung mit jedem Cluster aufbauen, da in jedem Cluster Aufträge für ihn vorhanden sein können.

Durch die geeignete Wahl des Clusters beim Prozeßstart (zufällig, zyklisch, lastabhängig) kann eine Lastbalancierung erreicht werden. Dadurch läßt sich die durch die Aktivitätenausführung entstehende Last (Parametertransfer zum Client) auf die WF-Datenbanken verteilen. Der Ausfall einer WF-Datenbank blockiert zwar alle Instanzen des zugehörigen Clusters, die WF-Instanzen anderer Cluster sind hiervon aber nicht betroffen. Da die Benutzer normalerweise auch WF-Instanzen anderer Cluster bearbeiten, können i.d.R. alle Benutzer, trotz des Ausfalls eines Clusters, zumindest eingeschränkt weiterarbeiten.

Für das Aktualisieren der Arbeitslisten stellen die Cluster ein potentielles Problem dar. Von einem Cluster werden jeweils WF aller WF-Typen kontrolliert, weshalb alle Benutzer des WfMS für diesen Cluster relevant sind. Deshalb muß ein Cluster Teile der Arbeitslisten aller Benutzer verwalten, weshalb er einen potentiellen Flaschenhals bildet. Die Anzahl der zu verwaltenden Arbeitslisten kann also durch die Verwendung mehrerer Cluster nicht verringert werden, was bei sehr großen Benutzerzahlen ein Problem sein kann.

Ein anderes Problem ergibt sich durch die Belastung des Netzwerks. Durch die Erhöhung der Anzahl der Teilnetze (auf welche die Cluster verteilt werden) läßt sich zwar ihre Kommunikationslast senken, aber durch die zufällige Wahl des Clusters beim Start einer Instanz werden auch Cluster mit ungünstiger Lage gewählt. Dieses Problem ist bei weiträumig verteilten Systemen besonders gravierend. So kann die Bearbeitung eines Prozesses zwar geographisch beschränkt sein, aber durch die unglückliche Wahl des Clusters für eine Instanz ständige WAN-Kommunikation zur Steuerung notwendig werden.

Beim **Exotica-Cluster**-Ansatz [AKA+94] bestehen die Cluster aus je einer WF-Datenbank und mehreren WF-Servern. Ein Client muß sich mit nur einem WF-Server jedes Clusters verbinden, da durch die gemeinsame WF-Datenbank alle Server eines Clusters über alle notwendigen Informationen verfügen.

Die Verwendung mehrerer WF-Server in einem Cluster erhöht die Verfügbarkeit, da ein Client beim Ausfall eines WF-Servers einen anderen Server dieses Clusters verwenden kann. Der Ausfall der WF-Datenbank blockiert natürlich weiterhin den gesamten betroffenen Cluster. Die Verwendung mehrerer WF-Server je Cluster reduziert die Last pro Server, die durch die Aktivitätenausführung und das Aktualisieren der Arbeitslisten entsteht. Da die Server eines Clusters alle Benutzer bedienen müssen, bildet die WF-Datenbank beim Aktualisieren der Arbeitslisten aber weiterhin einen Flaschenhals.

2.2.2 Server sind nahe bei den Bearbeitern

Die im folgenden beschriebenen Ansätze versuchen aus der gezielten Wahl des WF-Servers Vorteile zu ziehen. Dazu wird jeweils der WF-Server verwendet, der nahe bei den Bearbeitern liegt. Dieser Ansatz basiert auf der Annahme, daß die meisten Aktivitäten von Benutzern ausgeführt werden, die (fast) alle derselben OE angehören. Diese stellt somit einen guten Ort für den WF-Server dar. Eine OE kann jede Art von organisatorischer Einheit sein. Allerdings wird in diesem Zusammenhang gefordert, daß die OE geographisch beschränkt ist.

Die Konzepte dieser Kategorie können danach unterschieden werden, ob eine Prozeßinstanz den WF-Server wechseln kann, also eine Migration[1] möglich ist (Abb. 5b), oder nicht (Abb. 5a). Eine solche Migration ist sinnvoll, wenn es Prozesse gibt, die nacheinander in verschiedenen OE bearbeitet werden. Man denke an einen Prozeß, der zuerst die Verkaufsabteilung durchläuft und dann den Versand. Sind diese weit voneinander entfernt, so ist es wesentlich billiger, den gesamten Prozeß einmal zu migrieren, als die WAN-Verbindung für jeden weiteren Teilschritt zu verwenden. Bei den Ansätzen, bei denen keine Migration möglich ist, wird von der Annahme ausgegangen, daß ein kompletter Prozeß weitgehend zu nur einer OE gehört. Aktivitäten, die in anderen OE ausgeführt werden, werden ebenfalls von diesem, dann natürlich nicht optimalen Server gesteuert.

[1] Bei einer Migration müssen alle Instanzdaten zum neuen Server kopiert werden, bevor dieser die Kontrolle über die Instanz übernehmen kann. Dies sind zum einen die Parameterdaten und zum anderen die Zustandsinformation der Instanz (z.B. in Form einer Ablaufhistorie).

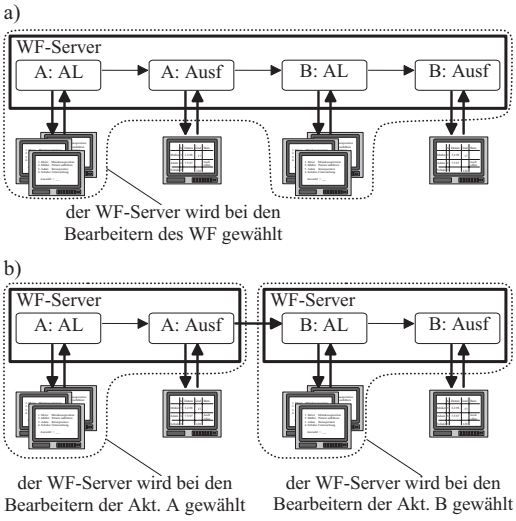

Abb. 5. WF-Server nahe bei den potentiellen Bearbeitern, Granularität: **a** gesamter Prozeß (ohne Migration), **b** einzelne Aktivitäten (mit Migration)

Wird die Anzahl dieser „entfernten“ Aktivitäten zu groß, so kann die durch ihre Ausführung erzeugte Kommunikationslast sehr hoch werden.

Ist eine Migration von Prozeßinstanzen möglich, so kann im Prinzip jede Aktivität von dem für sie optimalen Server kontrolliert werden. Allerdings gehen einige dieser Ansätze von der einschränkenden Annahme aus, daß alle potentiellen Bearbeiter einer Aktivität derselben OE angehören. Die Bearbeiterauflösung erfolgt dann nur lokal für die Benutzer im Domain dieses WF-Servers, eine globale Bearbeiterauflösung findet nicht statt.

Wir analysieren zuerst die Variante ohne Migration. Die Server sind jeweils für verschiedene Prozeßtypen zuständig. Da deren Laufzeitdaten voneinander unabhängig sind, ist keine Synchronisation zwischen den WF-Servern notwendig. Die Last für die Steuerung der Aktivitätenausführung wird unter den Servern aufgeteilt. Das Verteilen der Last ist etwas schwierig, da Prozesse nur als Ganzes einem Server zugeordnet werden können. Sollte ein Prozeßtyp alleine schon einen Server überlasten, gibt es keine geeignete Verteilung.

Auch wenn Migrationen möglich sind, teilt sich die Last für die Steuerung der Aktivitätenausführung zwischen den WF-Servern auf. Es kommen zwar die Migrationskosten hinzu, aber die durch die Aktivitätenausführung erzeugte Kommunikationslast ist geringer, da durch die Migration für jede Aktivität ein geeigneter WF-Server ausgewählt werden kann.

Zur Abschätzung der Belastung der Server durch das Aktualisieren der Arbeitslisten nehmen wir generell an, daß ein Benutzer nicht in alle Prozeßtypen bzw. Aktivitätentypen involviert ist. Deshalb ist jeder Server nur für einen Teil der Benutzer zuständig, weshalb er auch nur einen Teil der Last bewältigen muß. Erfolgt die Bearbeiterauflösung nur lokal, d.h., ist der Server nur für die Bearbeiter seines Domains zuständig, so erzeugt das Aktualisieren der Arbeitslisten besonders wenig Last. Allerdings hat diese Variante den Nachteil, daß beim Ausfall eines Servers alle seine Benutzer blockiert sind, da sie ihre Aufträge nur von ihm erhalten. Außerdem sind manchmal Aktivitäten wünschenswert, die in mehreren OE bearbeitet werden können (z.B. Einholen einer Unterschrift von einem beliebigen Abteilungsleiter). Diese sind bei lokaler Bearbeiterauflösung nicht modellierbar. Des weiteren schränkt dieses Konzept die Replikation der Server ein. Bei Überlastung eines Servers ist es nicht möglich, die Last auf mehrere Server zu verteilen, da diese dann getrennte OE mit disjunkten Benutzern darstellen würden.

MOBILE [HS96, Jab97] verfolgt das Ziel, durch Replikation der Server und Partitionierung der Daten die Last für die einzelnen Server zu minimieren und dadurch eine möglichst hohe Gesamtlast zu bewältigen. Dazu werden die Aufgaben eines WF-Servers in sogenannte Aspekte zerlegt, wie z.B. den funktionalen Aspekt, den Kontollfluß und den Datenfluß. Jeder dieser Aspekte wird durch einen eigenen Server realisiert, die durch einen Systemkern verbunden sind. Ist ein solcher Server überlastet, so wird er repliziert.

Das Synchronisationsproblem bei Änderung der Daten dieser Server wurde folgendermaßen gelöst: Statische Daten, wie z.B. Schemadaten, werden repliziert. Die Server für verschiedene Aspekte haben keine gemeinsamen Daten, außer der statischen WF-Id; diese wird repliziert. Die anderen Daten lassen sich nach den Aspekten partitionieren. Die verschiedenen Replikate eines Servers für denselben Aspekt verwenden zwar dieselben dynamischen Instanzdaten, aber jeder Workflow-Typ hat eine korrespondierende OE und damit einen fest zugeordneten Server. Es werden also gesamte Prozesse einem WF-Server zugeordnet, eine Migration ist nicht vorgesehen. Entsprechend dieser Zuordnung lassen sich diese Daten partitionieren.

Die Aufteilung in Aspekte bringt eine Reduktion der Last um bestenfalls einen Faktor, der der Anzahl der verschiedenen Aspekte entspricht. Da dies aber ein kleiner (konstanter) Faktor ist (gemäß [HS96]: 6), führt dies zu keiner signifikanten Reduktion der Last. Daß sich die Last gleichmäßig auf die Aspekte-Server verteilt, ist zudem unrealistisch, da deren Aufgaben völlig unterschiedlichen Aufwand erfordern. Hinzu kommt, daß für bestimmte Operationen (wie z.B. die dynamische Restrukturierung eines Ablaufgraphen) fast alle Aspekte benötigt werden, so daß durch die Verteilung auf mehrere Server ein zusätzlicher Kommunikations- und Synchronisationsaufwand entsteht.

In [SNS99] wurde der MOBILE-Ansatz noch erweitert. Beim Start eines (Sub-)WF, also zur Laufzeit, wird entschieden, von welchem WF-Server dieser kontrolliert werden soll. Bei dieser Entscheidung werden Rechte, Gewichte (gewünschte Ressourcenzuordnung) und Kosten (Leistungsfähigkeit von Rechnern und Kommunikationsverbindungen) berücksichtigt. Es findet allerdings keine Migration der Prozeßinstanz statt, sondern die Subprozesse werden entfernt gestartet. Beim Start und bei der Beendigung jedes Sub-WF muß deshalb mit dem Server des Vater-WF kommuniziert werden, auch wenn der nächste Sub-WF wieder vom selben Server kontrolliert wird.

Das **MENTOR**-System [WWWK96a, WWWK96b, WWK+97, MWW+98] basiert auf einigen Standardkomponenten, wie dem Transaktionsmonitor TUXEDO [UNI92], CORBA [OMG95] und Statemate, einem Werkzeug zur Modellierung und Ausführung von State-/Activitycharts. Dabei wird in einem Statechart der Kontrollfluß und in einem Activitychart der zugehörige Datenfluß und der funktionale Aspekt eines Workflows modelliert. Kernidee des in diesem Projekt verfolgten Ansatzes ist es nun, die State-/Activitycharts so zu partitionieren, daß eine zum zentralen Fall äquivalente verteilte Ausführung entsteht und jede Aktivität in einem zuvor für sie nach OE festgelegten „Processing Entity“ ausgeführt wird. Der entsprechende Server ist somit nur für die ihm zugeordneten Aktivitäten und für die seinem Processing Entity angehörigen Benutzer zuständig. Dies ist einer der Ansätze, die auf eine globale Bearbeiterauflösung verzichten.

An den Zerlegungspunkten des State-/Activitycharts erfolgt ein Wechsel des Servers, d.h., die komplette Prozeßinstanz migriert zu einem anderen Server. Es werden nur globale Variablen verwendet, deren Änderungen durch einen in jedem Server vorhandenen Communication-Manager propagiert werden. Alle Kommunikationen werden durch ein zwei-Phasen-Commit-Protokoll (2PC) geschützt, weshalb die verwendeten Anwendungen das XA-Protokoll unterstützen müssen.

WIDE [CGP+96, CGS97] verfolgt einen ähnlichen Ansatz, allerdings ist die Skalierbarkeit in diesem Projekt nur ein Teilaspekt. Es wird auch ein Transaktionsmanagement auf verschiedenen logischen Ebenen (geschach-

telte Transaktionen für Aktivitäten bzw. Sagas für Prozesse) angeboten. Mit Hilfe von aktiven Regeln ist zudem eine Ausnahmebehandlung möglich. Die Idee zur Verteilung ist analog zu MENTOR. Allerdings ist noch keine Migration vorgesehen, sondern die Daten verbleiben an einem Ort, und es wird entfernt mit Hilfe von CORBA-Diensten auf sie zugegriffen. Dies kann zu den schon erwähnten hohen Kosten bei mehrfachem weit entfernten Zugriff führen.

In **ADEPT** [BD97, BD98b] wird außer den Servern auch die Netzstruktur betrachtet, da auch Teilnetze durch zuviel Kommunikation überlastet werden können. Deshalb wird versucht, die Kommunikationskosten zu minimieren, indem für jede Aktivität der optimale Server gewählt wird. Dazu wird eine Aktivität meist von dem Server kontrolliert, in dessen Teilnetz sich die meisten Benutzer mit einer passenden Rolle befinden. Die Aktivität wird aber auch geeigneten Benutzern in anderen Teilnetzen angeboten (globale Bearbeiterauflösung). – Es gibt Aktivitäten, die bei verschiedenen WF-Instanzen in unterschiedlichen OE (z.B. Abteilungen) ausgeführt werden, woraus sich jeweils ein anderer optimaler Server ergibt. Ein Beispiel hierfür ist eine Aktivität „Patient für die Operation vorbereiten“, die von einer Pflegekraft der Station ausgeführt wird, auf der der betroffene Patient liegt. Deshalb wird in ADEPT für solche Aktivitäten der Server jeweils in der zugehörigen OE gewählt (*variable Serverzuordnung* [BD98b, BD99]). – Aufgrund von Überlastung oder Ausfall eines Servers kann es vorteilhaft sein, von der vorgeplanten Serverzuordnung dynamisch abzuweichen. Dies wird realisiert, indem Serverbeschreibungen einzelner WF-Instanzen vom WfMS zur Laufzeit verändert werden.

Um die Verteilung zu optimieren, werden Algorithmen verwendet, die zur Modellierungszeit laufen, also die Server nicht zusätzlich belasten. Die Algorithmen berechnen eine Verteilung, durch welche die Kommunikationskosten minimiert werden und analysieren die Last für die einzelnen Komponenten (WF-Server, Teilnetze und Gateways). Bei diesen Kosten wird die Kommunikation für das Aktualisieren der Arbeitslisten und für den Parametertransfer bei der Aktivitätenausführung berücksichtigt. Aber auch die Migrationskosten und die Kommunikationen von Anwendungen mit externen Datenquellen werden mit einbezogen.

Die Belastung für die WF-Server, WF-Datenbanken und Teilnetze wird bei diesem Ansatz verteilt. Die Migrationskosten sind dabei niedriger als bei MENTOR und WIDE, da nicht bei jedem Wechsel der OE migriert werden muß.

2.2.3 Server sind nahe bei der Anwendung

Eine weitere Möglichkeit, um aus der Wahl des Ortes des WF-Servers Vorteile zu ziehen, ist ihn dort zu plazieren, wo das Anwendungsprogramm der entsprechenden Aktivität läuft (Abb. 6). Diese Variante wird vor allem von Systemen verwendet, die auf einer objektorientierten Infrastruktur (z.B. CORBA) basieren. Die Anwendung ist dabei als Objekt gekapselt, das an einem bestimmten Ort im verteilten System allokiert ist. Das entsprechende Teilnetz oder sogar derselbe Rechner wird auch für den WF-Server der zugehörigen Aktivität gewählt. Die Prozeßinstanz ist ein Objekt, das zu dem jeweiligen WF-Server migriert. Eine andere Möglichkeit ist, daß lediglich Objektreferenzen zwischen den Servern wandern und diese dann bei der Verwendung der zugehörigen Parameter entfernt auf die entsprechenden Objekte zugreifen.

Die durch die Aktivitätenausführung erzeugte Last läßt sich dadurch auf die einzelnen Server verteilen. Die bei diesem Konzept entstehende Migrationslast ist allerdings recht groß, da das Instanzenobjekt bei fast jedem Teilschritt migriert (selbst wenn er vom selben Bearbeiter wie der Vorgängerschritt ausgeführt wird), weil gewöhnlich eine andere Anwendung verwendet wird. Falls vom WF-Server entfernt auf die Daten zugegriffen wird, verringert dies zwar die Migrationslast, aber die Last durch die Aktivitätenausführung ist wesentlich größer, da die Anwendungen mehrfach auf ein (evtl. weit) entferntes Objekt zugreifen. Ein generelles Problem des Ansatzes ist, daß die Verteilung der Last davon abhängt, wie häufig die einzelnen Anwendungen aufgerufen werden. Dies läßt sich nicht immer steuern.

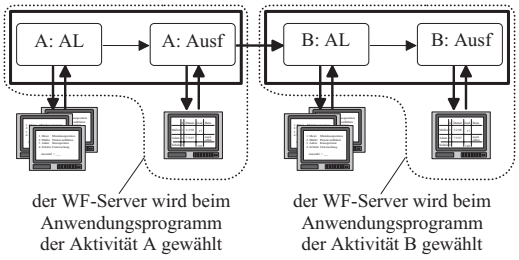

Abb. 6. Der WF-Server wird nahe bei Anwendung plaziert

Es ist schwierig, die Belastung der Server durch das Aktualisieren der Arbeitslisten zu bestimmen. Ein Server muß prinzipiell alle Benutzer bedienen, da die Verteilung nicht an OE (z.B. Abteilungen) orientiert ist, sondern an Anwendungen. Dadurch kann er zu einem Flaschenhals werden. Allerdings werden i.d.R. die meisten Benutzer nur wenige Anwendungen aufrufen dürfen, da sie ihre Rolle auf bestimmte Tätigkeiten einschränkt. Da deshalb jede Anwendung nur von bestimmten Benutzern verwendet wird, müssen nur deren Arbeitslisten gewartet werden, was die Last für den WF-Server reduziert. Diese Annahme muß aber nicht gelten. Es sind Szenarien denkbar, in denen es Abteilungen gibt, deren Mitarbeiter zwar getrennte Prozesse bearbeiten, jedoch mit denselben Anwendungen. In einem solchen Fall wird jede Anwendung von jedem Benutzer aufgerufen, wodurch die Skalierbarkeit eingeschränkt wird.

Dieser Ansatz geht zudem von der Annahme aus, daß jede Anwendung einen eindeutig bestimmbaren Ort hat. Das ist auch der Grund dafür, daß dieses Verteilungsmodell vor allem von Ansätzen verwendet wird, die auf einer objektorientierten Infrastruktur basieren. Da das Aktivitätenprogramm dann als Objekt gekapselt ist, hat es stets einen bestimmbaren Ort. Bei einer Datenbankanwendung wäre dieser z.B. durch den Rechner bestimmt, auf dem das DBS läuft. Problematisch sind Anwendungen, die auf mehreren Rechnern installiert sind oder über das Netzwerk gestartet werden, wie z.B. ein Editor, eine Textverarbeitung oder eine Eingabemaske. Solche Anwendungen werden immer auf dem Rechner des (a priori unbekannten) Benutzers ausgeführt, der die Aktivität bearbeitet. Da durch die Anwendung kein Ort vorgegeben wird, ist dieser Ansatz nicht ohne weiteres verwendbar, weil dann kein WF-Server für die entsprechende Aktivität bestimmt werden kann. Man könnte nun für den WF-Server einen beliebigen Ort wählen, dies wäre aber bezüglich der Kommunikationskosten sehr ungünstig. Es bietet sich an, den WF-Server in diesen Fällen wie bei den Ansätzen aus Abschnitt 2.2.2 so zu wählen, daß er nahe bei den Bearbeitern der Aktivität liegt. – In der Regel werden Anwendungen wohl häufig auf Rechnern der OE laufen, zu der ihre Bearbeiter gehören, so daß die letzten beiden Verteilungsmodelle zu ähnlichen Ergebnissen führen.

In [SM96] werden die verwandten Systeme **CodAlf** und **BPAFrame** beschrieben, die auf unterschiedlicher objektorientierter Middleware basieren. Während CodAlf eine objektorientierte Erweiterung des OSF DCE [Sch93] verwendet, basiert BPAFrame auf CORBA [OMG95]. Wie oben

beschrieben, wird jede Anwendung in einem Objekt gekapselt. Ein solches Objekt hat einen festen Ort – an dem sich auch der zugehörige WF-Server befindet – und wird als Runtime-System bezeichnet. Prozeßtypen werden als Objekttypen implementiert, die Prozeßinstanzen sind somit als Objekte realisiert. Sie enthalten außer den Anwendungsdaten auch noch die Prozeßbeschreibung. Diese mobilen Objekte migrieren zum Ort des Anwendungsobjektes der nächsten Aktivität. Dieser Ort wird von einer Komponente (Trader) ermittelt, die als Directory Service dient (also angibt, wo sich ein geeignetes Objekt befindet) und außerdem eine Lastverteilung vornimmt. Der erforderliche entfernte Zugriff auf Daten wird durch die verwendeten Middleware-Dienste realisiert. Dasselbe gilt für den Zugriff der Benutzer auf die entfernten Runtime-Systeme.

Wie oben bereits ausgeführt wurde, hängt bei diesem Ansatz die Last eines WF-Servers davon ab, wie häufig die zugehörige Anwendung verwendet wird. Falls es aber möglich ist, den Anwendungsserver zu replizieren, so sorgt der Trader für eine Verteilung der Last.

METEOR$_2$ [DKM+97, MSKW96, SK97] verwendet einen Ansatz, der dem eben beschriebenen sehr ähnlich ist. Er basiert ebenfalls auf CORBA. Allerdings ist die Ablaufbeschreibung nicht in einem mobilen Instanzenobjekt enthalten, sondern sie wird in ihre Teilschritte (jeweils mit Kanten zu vorangehenden und nachfolgenden Aktivitäten) zerlegt. Aus dieser Information wird für jede Aktivität ein Teil des WF-Servers (Taskmanager genannt) erzeugt, der die Ausführung genau dieser Aktivität steuert. Ein Taskmanager kann prinzipiell an einem beliebigen Ort allokiert werden, allerdings wird auch hier der Ort der Anwendung vorgeschlagen. Im Gegensatz zu den beiden letzten Ansätzen migrieren keine Instanzenobjekte zwischen den Taskmanagern. Statt dessen werden beim Weiterschalten des Prozesses zur nächsten Aktivität Referenzen auf die verwendeten Datenobjekte übergeben. Bei der Ausführung einer Aktivität muß dann auf die i.d.R. entfernt liegenden Datenobjekte zugegriffen werden. Schließlich wird in METEOR$_2$ der Ort der Anwendungen und der Taskmanager eindeutig festgelegt, so daß die Aufgabe des Traders entfällt. Dadurch ist aber auch keine dynamische Lastbalancierung möglich.

Der Ansatz betrachtet auch noch weitere Aspekte, wie den Wiederanlauf nach Fehlern oder die effiziente Ermittlung des aktuellen Zustands einer Prozeßinstanz. Zu diesem Zweck gibt es einen zentralen Monitoring-Service, dem die Taskmanager Änderungen der Zustände melden. Außerdem werden ihm Referenzen auf verwendete Daten geschickt, um die Daten beim Wiederanlauf nach Fehlern rekonstruieren zu können.

Außer der Last der WF-Server ist bei diesem Ansatz auch noch die Belastung des Monitoring-Services interessant. Da diese zentrale Komponente bei der Ausführung jeder Aktivität über Änderungen informiert werden muß, ist ihre Last proportional zu der Last, die durch die Aktivitätenausführung erzeugt wird. Sie ist allerdings nur ein sehr kleiner Bruchteil dieser Last, da lediglich Referenzen auf Daten übertragen und nur einfache Lese- und Schreiboperationen durchgeführt werden. Aus diesem Grund ist eine zentrale Komponente bis zu einer sehr hohen Systemlast ausreichend. Etwas schlechter sieht die Analyse für das Teilnetz aus, in dem der Monitoring-Service angesiedelt ist. Es werden zwar nur kleine Pakete übertragen, dafür aber sehr viele. Der Monitoring-Service stellt auch ein Problem für die Verfügbarkeit dar. Da er für das Recovery notwendig ist, führt sein Ausfall zum Systemstillstand. Eine Replikation ist nicht vorgesehen, sie würde im Fall einer Netzpartitionierung auch nicht verhindern, daß der abgeschnittene Teil (u.U. ist das der größere Teil) blockiert ist.

2.3 Voll verteilte WfMS

Ein WF-Server und das zugehörige Teilnetz sind potentielle Engpässe eines Systems. Diese können vermieden werden, indem man auf (explizite) Server verzichtet und die entsprechende Funktionalität in jedem Client realisiert. Die komplette WF-Instanz migriert dann nach Beendigung einer Aktivität und der Ermittlung des nächsten Bearbeiters zu dessen Rechner (siehe Abb. 7). Das heißt, jeder Arbeitsplatzrechner fungiert quasi als „Mini-Server“ für seine lokalen Anwendungen. Beim Ermitteln dieses Bearbeiters wird eine verteilte Synchronisation notwendig. Hierbei stellt sich die Frage, woher ein Client alle angemeldeten Benutzer mit

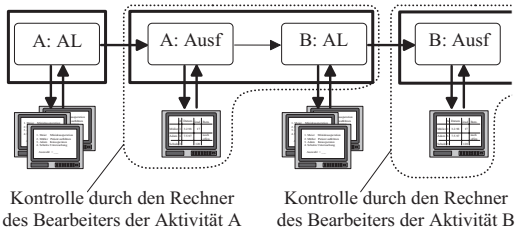

Abb. 7. WF-Kontrolle durch den Rechner des Benutzers, der die aktuelle Aktivität ausführt

einer passenden Rolle für die Nachfolgeaktivität kennt. Dies ist notwendig, um einen entsprechenden Eintrag in deren Arbeitslisten einfügen zu können. Dieses Problem wird von manchen Systemen dadurch umgangen, daß man auf die Bearbeiterauflösung verzichtet und jeder Aktivität einen eindeutigen Bearbeiter zuordnet.

Die durch die WF-Ausführung entstehende Last verteilt sich auf alle Arbeitsstationen des Systems. Hierdurch ist die Last pro Komponente („Mini-Server“) kleiner als bei den in Abschnitt 2.2 beschriebenen Systemen. Das ist auch notwendig, da die Arbeitsstationen keine leistungsstarken Maschinen, sondern nur die Rechner der Benutzer sind. Die Migrationslast ist bei voll verteilten Ansätzen besonders hoch, weil die Prozeßinstanz bei fast jedem Teilschritt migriert. Der dafür erforderliche Aufwand teilt sich ebenfalls auf die Arbeitsstationen aller Benutzer auf. Wird auf eine Bearbeiterauflösung verzichtet, so wird der Aufwand für die verteilte Synchronisation vermieden. Dies stellt aber einen Verlust an Funktionalität bzw. Flexibilität dar, der nicht bei jeder Anwendung akzeptiert werden kann. Wird eine Bearbeiterauflösung durchgeführt, so verteilt sich die dadurch entstehende Last auf die vielen Arbeitsstationen der Benutzer, weshalb die Last für eine einzelne Maschine klein ist. Da es keine WF-Server gibt, bei denen sich die Kommunikation konzentriert, gibt es keine Flaschenhälse im Kommunikationsnetzwerk. Die Belastung der Teilnetze stellt also kein Problem dar, wenn die Arbeitsstationen auf ausreichend viele Teilnetze verteilt werden. Problematisch ist, daß die Zustandsinformation nicht in Servern verfügbar, sondern über die Benutzerrechner verteilt ist. Deshalb muß der WF „gesucht“ werden, wenn nach seinem aktuellen Zustand gefragt wird, was zu einem hohen Aufwand führt.

INCAS [BMR96] ist ein System, das auf eine Bearbeiterauflösung verzichtet. Der Name kommt von einem INformation CArrier, der jeweils zur Arbeitsstation der nächsten Aktivität migriert. Dieser INCA enthält den gewünschten Dienst, Regeln, die den Daten- und Kontrollfluß beschreiben, die Daten der Instanz und Atomaritätsanforderungen. Auch die Arbeitsstationen verfügen über Regeln, die mit denen des INCA zusammen verwendet werden, um eine Aktivität auszuführen und die nächste Aktivität zu berechnen. Für diese Nachfolgeaktivität wird, ebenfalls anhand der Regeln, die zugehörige Arbeitsstation berechnet. Es findet keine Bearbeiterauflösung statt, somit ist auch keine Synchronisation zwischen den potentiellen Nachfolgern notwendig. Bei jedem Weiterschalten zur nächsten Aktivität wird der INCA verändert. Er wird allerdings nicht modifiziert, sondern es wird eine neue Version von ihm erzeugt, die zusammen mit der alten migriert. Da dadurch die Prozeßinstanz sehr groß wird, ist die durch die Migration verursachte Last besonders hoch. Ein Vorteil dieser Vorgehensweise ist jedoch, daß sich Regeln auch auf alte Versionen der Daten beziehen können. Das gesamte System wird durch Regeln gesteuert. Die Regelmenge ist sogar dynamisch, d.h., bei der Ausführung einer Aktivität können Regeln erzeugt oder verändert werden. Auch die Transaktionssemantik der Aktivitäten wird

mittels Regeln definiert. Allerdings ist eine große, verteilte und dazu noch dynamische Regelmenge schwer zu durchschauen.

Bei **Exotica/FMQM** [AMG+95] wird nach Beendigung einer Aktivität eine globale Bearbeiterauflösung durchgeführt. Die Kommunikation findet (gesichert) über Persistent-Queues statt. Der Ablauf wird wie in FlowMark durch einen Graphen mit Kontroll- und Datenkonnektoren beschrieben. Dieser Graph wird so auf die Knoten verteilt, daß jeder Knoten diejenigen Teile des Graphen erhält, die Aktivitäten mit den Rollen seiner Benutzer enthalten. Ist die Bearbeitung einer Aktivität beendet, so werden Nachrichten an alle Knoten geschickt, die für Aktivitäten verantwortlich sind, zu denen von der aktuellen Aktivität aus Kontroll- oder Datenkanten führen. Die Daten werden bei dieser Methode schrittweise verteilt, sie migrieren nicht mit der Prozeßinstanz.

Wegen der Bearbeiterauflösung kann eine Instanz nicht einfach an den Knoten der nachfolgenden Aktivität gesendet werden, sondern dieser muß erst ermittelt werden. Dazu informiert der Vorgängerknoten alle potentiellen Nachfolger über die zu vergebende Aktivität, indem er die entsprechende Information in deren Message-Queues schreibt. Diese holen sich dann die Instanz aus dessen Ausgabe-Queue, wenn sie die nächste Aktivität ausführen wollen. Die Synchronisation erfolgt durch das Transaktionskonzept des Queueing-Systems. Durch die Auswertung der von einer Aktivität ausgehenden Datenkonnektoren lassen sich aber lediglich die Aktivitäten ermitteln, die diese Daten potentiell benötigen. Es ist jedoch noch nicht bekannt, an welchen Knoten diese später ausgeführt werden, so daß die Instanzdaten zu ihnen transportiert werden können. Die bei diesem Ansatz erzeugte Last hängt stark davon ab, wie effizient der persistente Message-Queue-Dienst implementiert ist, insbesondere ob das entfernte Lesen (effizient) unterstützt wird.

Tabelle 1 bietet eine Übersicht über die wichtigsten Eigenschaften der in diesem Kapitel diskutierten Verteilungsmodelle. Im nächsten Kapitel wird diese qualitative Analyse durch quantitative Untersuchungen untermauert.

Tabelle 1. Die wichtigsten Eigenschaften der untersuchten Verteilungsmodelle

Modell	Eigenschaften
WfMS mit zentralem Server:	
	• der WF-Server und dessen Teilnetz bilden potentielle Flaschenhälse
	• es gibt wenig Möglichkeiten zur Optimierung der Lage des WF-Servers
WfMS mit mehreren Servern	
Server werden zufällig gewählt:	
	• ist einfach zu realisieren, kein Zusatzaufwand durch Migrationen
	• keine Optimierung bzgl. Kommunikationskosten möglich
	• jeder Cluster (WF-DB) muß Arbeitslisten aller Benutzer verwalten
Server sind nahe bei den Bearbeitern:	
	• Last verteilt sich auf Server, gute Verteilung der Kommunikationslast möglich
	• ohne Migration: Server ist bzgl. einzelner Aktivitäten nicht optimal
	• lokale Bearbeiterauflösung: eingeschränkte Funktionalität
	• variable Serverzuordnungen: weitere Reduzierung der Kommunikationslast
Server sind nahe bei der Anwendung:	
	• erfordert, daß Aktivitätenprogramme einen fest zugeordneten Ort haben
	• meist in Kombination mit objektorientierter Infrastruktur verwendet
	• ist häufig ähnlich wie das Verteilungsmodell „Server bei den Bearbeitern“
voll verteilte WfMS:	
	• keine Flaschenhälse im System vorhanden
	• viele Migrationen (bei jedem Wechsel des Bearbeiters)
	• problematisch: sehr hoher Grad an Verteilung der Information
	• häufig muß jede Aktivität einen fest zugeordneten Bearbeiter haben

3 Simulation der Verteilungsmodelle

Um einen quantitativen Vergleich zu ermöglichen, wurde die WF-Ausführung für folgende Verteilungsmodelle simuliert (vgl. Tabelle 2): Zentraler Server, zufällig gewählter Server (Cluster), Server nahe bei den Bearbeitern mit/ohne Migration und mit/ohne globaler Bearbeiterauflösung, variable Serverzuordnung, Server bei der Anwendung und voll verteiltes WfMS. Um einen wirklich objektiven Vergleich der Verteilungsmodelle zu erhalten, wäre an sich die Simulation sehr vieler Szenarien erforderlich. Da dies hier aus Platzgründen nicht möglich ist, haben wir lediglich zwei WF-Typen simuliert, hierbei aber darauf geachtet, daß keine „unfairen Parameter“ verwendet werden. Das heißt, die Szenarien wurden so ausgewählt, daß die Stärken und Schwächen aller Verteilungsmodelle zum Tragen kommen.

Das verwendete Simulationsprogramm ermöglicht es, die WF-Ausführung unter verschiedenen Verteilungsmodellen zu simulieren. Es berechnet für die WF-Server, die Teilnetze und Gateways die in einem realen WfMS durch die WF-Ausführung entstehende Last. Dabei werden die Kosten für den Transfer von Parameterdaten zu und von den Aktivitätenprogrammen, für das Aktualisieren der Arbeitslisten und für die Migrationen berücksichtigt. Dem Simulationsprogramm werden Beschreibungen der zu simulierenden WF-Typen, die Anzahl der Instanzen und die Eigenschaften der Benutzer (z.B. Rolle, Teilnetz) vorgegeben. Die konkreten Zeitpunkte für den Start der WF-Instanzen werden zufällig im Simulationszeitraum gewählt. Die Bearbeitungszeiten der Aktivitäten sind in einem vorgegebenen Zeitintervall gleichverteilt. Nach Beendigung einer Aktivität durch einen Benutzer wird simuliert, daß dieser zufällig einen Eintrag aus seiner Arbeitsliste auswählt und die zugehörige Aktivität ausführt.

Da für das Simulationsergebnis der stationäre Zustand relevant ist, wurde mit dem Verfahren von Welch [Wel83, Lan92] die Ausdehnung der Einschwingphase ermittelt, so daß die zugehörigen Werte verworfen werden konnten. Außerdem wurde die Simulation 10000 mal durchgeführt und die Ergebnisse gemittelt. Dadurch wurde erreicht, daß sich die 90%-Konfidenzintervalle für die Erwartungswerte mit $< \pm 0,1\%$ Abweichung um die jeweiligen Mittelwerte bewegen.

3.1 Simulation I

3.1.1 Modellannahmen

Zum Vergleich der Verteilungsmodelle wurde die Ausführung des in Abb. 8 dargestellten (stark vereinfachten) Klinik-WF simuliert. Wir beschreiben zunächst die gewählte Abbildung auf die verschiedenen Verteilungsmodelle und erläutern dann die Simulationsergebnisse. Der WF beginnt damit, daß ein Patient in die Ambulanz kommt, wo er untersucht und eine Diagnose erstellt wird (Akt. 1-6). Dazu muß eine Blutprobe im Labor untersucht werden (Akt. 5). Falls notwendig, wird er auf einer von 3 möglichen Stationen aufgenommen, versorgt und behandelt (Akt. 7-14). Nach der Entlassung des

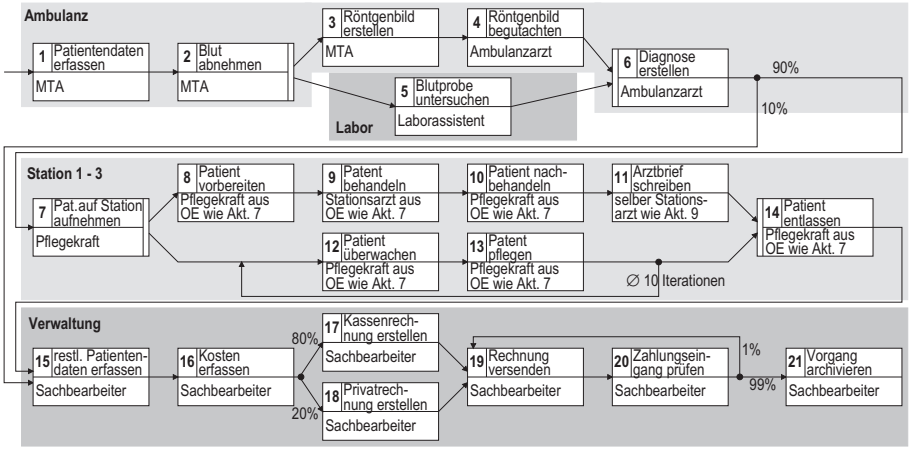

Abb. 8. Für die Simulation verwendeter WF: Aktivitäten mit Bearbeiterzuordnung

Patienten wird der WF in der Verwaltung fortgesetzt und schließlich beendet (Akt. 15-21). Für die Bearbeitung wurden 35 Benutzer veranschlagt (2 Ambulanzärzte, 3 MTA, je Station 2 Ärzte und 6 Pflegekräfte, 4 Sachbearbeiter, 2 Laborassistenten). Jede Station, die Ambulanz, die Verwaltung und das Labor bilden jeweils ein eigenes Teilnetz. Während der Simulationsdauer von 20 Tagen werden 500 Instanzen des Klinik-WF simuliert. Die konkreten Größen der Parameterdaten sind aus Platzgründen und zur Erhöhung der Übersichtlichkeit nicht angegeben. Da Lastangaben im folgenden immer relativ zum zentralen Fall erfolgen, haben sie auch keinen Einfluß auf das Simulationsergebnis. Eine Verdoppelung der Datenmenge führt bei allen Verteilungsmodellen zu einer Verdoppelung der Last und beeinflußt damit deren Vergleich nicht.

Die Abbildung des Klinik-WF auf die verschiedenen Verteilungsmodelle erfolgt wie in Tabelle 2 dargestellt.

Modell 1: Zentraler Server. Der WF-Server befindet sich im Teilnetz der Ambulanz. Mögliche Alternativen hierzu werden später diskutiert.

Modell 2: Server zufällig. Der Server für eine gesamte WF-Instanz wird zufällig ausgewählt.

Modell 3: Keine Migration. Das optimale Teilnetz für den WF-Server ist das der Verwaltung.

Modell 4: Keine globale Bearbeiterauflösung. Ohne globale Bearbeiterauflösung müssen die Stationen 1-3 einen gemeinsamen Domain (ein Teilnetz mit nur einem WF-Server) bilden, damit die Aktivitäten 7-14 von den Benutzern aller 3 OE bearbeitet werden können. Deshalb verteilt sich die Gesamtlast bei diesem Ansatz auf insgesamt nur 4 Domains.

Modell 5: Migration und globale Bearbeiterauflösung. Bei diesem Verteilungsmodell werden die Aktivitäten 7-11 und 14 vom Server der Verwaltung kontrolliert und nicht von dem einer Station, um Kommunikationskosten einzusparen.

Modell 6: Variable Serverzuordnung. Variable Serverzuordnungen ermöglichen, daß die Aktivitäten 8-14 vom Server der Station kontrolliert werden, aus der die Bearbeiter stammen. Aktivität 7 wird vom Server der Ambulanz gesteuert, da vor ihrer Ausführung die betroffene Station noch nicht bekannt ist.

Tabelle 2. Zuordnung der Aktivitäten zu den WF-Servern bei den simulierten Verteilungsmodellen

Verteilungsmodell	Anzahl Serv.	Teiln.	Zuordnung der WF-Server zu den Aktivitäten
1. zentraler Server	1	6	Akt. 1–21: Ambulanz
2. Server zufällig	6	6	zufälliger Server für gesamte Instanz
3. keine Migration	6	6	Akt. 1–21: Verwaltung
4. keine globale Bearbeiterauflösung	4	4	Akt. 1–4, 6: Ambulanz, Akt 5: Labor, Akt. 7–14: Station, Akt. 15–21: Verwaltung
5. Migration und glob. Bearbeiterauflösung	6	6	Akt. 1–6: Ambulanz, Akt. 12–13: Station 1, Akt. 7–11, 14–21: Verwaltung
6. variable Serverzuordnung	6	6	Akt. 1–7: Ambulanz, Akt. 8–14: Station des Bearbeiters von Akt. 7, Akt. 15–21: Verwaltung
7. bei der Anwendung	6	6	Akt. 2–4, 6: Ambulanz, Akt. 5: Labor, Akt. 7–14: Station 1, Akt. 1, 15–21: Verwaltung
8. voll verteilt	35	6	Akt. 1–21: Rechner des Bearbeiters der Aktivität

Modell 7: Bei der Anwendung. Für jede Aktivität muß festgelegt werden, wo die zugehörige Anwendung angesiedelt ist. Dabei wurde – wann immer möglich – angenommen, daß die Anwendung im Teilnetz der zugehörigen Bearbeiter liegt. Andernfalls ergibt sich eine wesentlich höhere Belastung für die Teilnetze. Allerdings ist diese Annahme nicht für alle Aktivitäten möglich. Die Aktivitäten 1 und 15 (Patientendaten erfassen) verwenden dieselbe Anwendung, werden aber in unterschiedlichen OE bearbeitet. Aktivität 4 wurde (entsprechend ihrer Bearbeiter) dem Labor zugeordnet. Um unnötige Migrationen zu vermeiden, wurde angenommen, daß die Anwendungen der Aktivitäten 7-14 alle auf dem Server derselben Station installiert sind.

Modell 8: Voll verteilt. Da es keine expliziten Server gibt, sind die in Tabelle 2 angegebenen 35 Server die Rechner der 35 Benutzer, die die WF-Steuerung übernehmen.

Bei der Auswahl des Prozesses für die Simulation wurde darauf geachtet, daß alle für die Verteilungsmodelle relevanten Aspekte berücksichtigt sind und eine ausgewogene Mischung erreicht wird. Würden bestimmte Aspekte fehlen (z.B. die Möglichkeit zur Migration), so würden verschiedene Verteilungsmodelle zusammenfallen. Wenn hinge-

gen bestimmte Aspekte dominieren würden (z.B. ständiger Wechsel der OE), so wären die Ergebnisse nicht sehr aussagekräftig. Deshalb wurde ein WF gewählt, dessen Bearbeiter in verschiedenen OE angesiedelt sind (Ambulanz, Labor, Stationen, Verwaltung). Außerdem gibt es Aktivitäten (7–14), die je nach Instanz in unterschiedlichen OE (Station 1-3) bearbeitet werden. Schließlich enthält der Prozeß Verzweigungen, Parallelität und Schleifen, und die Belastung der Bearbeiter ist nur so groß, daß ihre Arbeitslisten nicht „überlaufen".

3.1.2 Ergebnis der Simulation I

Für jedes Verteilungsmodell werden in Abb. 9 die Gesamtlast aller WF-Server, die Last pro WF-Server und die durchschnittliche Belastung der Teilnetze angegeben. Für die Gateways wird nicht die Belastung pro Gateway, sondern die insgesamt umgesetzte Last angegeben, weil letztere der ggf. über ein WAN zu kommunizierenden Datenmenge (und damit den entstehenden Kosten) entspricht. Da absolute Lastangaben (in Bytes/sec) wenig aussagekräftig sind, wird für die Lastangaben die Last des zentralen Falls als Vergleichswert (100) verwendet und die anderen Ergebnisse werden relativ zu diesem Wert angegeben. Ein Wert von 100 bedeutet dabei „dieselbe Last wie im zentralen Fall" und nicht etwa „100% Auslastung der Komponente". Es soll die bei den verschiedenen Verteilungsmodellen entstehende Last verglichen und nicht die Überlastung einer konkreten Komponente geprüft werden. Aus diesem Grund betrachten wir Lastsummen und -mittelwerte. In verteilten WfMS sind alle WF-Server, Teilnetze und Gateways „logisch" gleichartig, und es werden stets verschiedene WF-Typen gleichzeitig ausgeführt. Deshalb gibt es keinen Grund, warum zwischen den Komponenten signifikante Ungleichbelastungen auftreten sollten. Eine Ausnahme davon bilden zentrale WfMS, weil es bei diesen genau ein Teilnetz mit einem WF-Server gibt. Da dieses besonders stark belastet ist, wird es im folgenden gesondert betrachtet.

Bei den Verteilungsmodellen ohne Migration (2. und 3.) ist die Servergesamtlast identisch mit dem zentralen Fall, ansonsten ist sie höher. Dabei fällt auf, daß sie bei einem voll verteilten WfMS (8.) extrem hoch ist. Die Last pro Server ist bei den Ansätzen 2 - 8 viel kleiner als für den zentralen Server. Die durchschnittliche Teilnetzlast ist meist ähnlich zum zentralen Fall, nur durch variable Serverzuordnungen (6.) wird sie deutlich reduziert. Die Belastung der Gateways ist bei Ansätzen mit der Möglichkeit zur Migration (4.-8.) generell niedriger.

3.1.3 Diskussion der Ergebnisse der Simulation I

Modell 1: Zentraler Server. Die durchschnittliche Teilnetzlast hängt davon ab, in welchem Teilnetz der WF-Server angesiedelt ist. Konkret ergibt sich beim Teilnetz der Ambulanz die Last 100, bei den Stationen 101,1, bei der Verwaltung 98,6 und beim Labor 111,2. Es wurde also mit der Ambulanz ein „recht guter Fall" ausgewählt (da es im zentralen Fall nur einen WF-Server im System gibt, kann dessen Lage nicht für einzelne WF-Typen optimiert werden). Die Last im Teilnetz des WF-Servers ist unabhängig von dieser Wahl stets 334,6. Diese hohe Last führt schnell zur Überlastung dieses Teilnetzes. Die Belastung des WF-Servers ist unabhängig davon, in welchem Teilnetz er angesiedelt ist. Da es nur einen Server im System gibt, ist die Gesamtlast der Server identisch mit der Last pro Server. Diese Gesamtlast kann von anderen Verteilungsmodellen nicht unterboten werden, da im zentralen Fall keine Kommunikation zur Synchronisation (z.B. Migrationen) notwendig ist. Die Gateway-Last wird bei diesem Verteilungsmodell alleine von den 5 Gateways umgesetzt, die das Teilnetz des WF-Servers mit denen von Bearbeitern verbinden. Um die Belastung der Teilnetze mit jener der Gateways vergeichen zu können, sei noch angemerkt, daß die von den Gateways umgesetzte Datenmenge 44,3% der in den Teilnetzen insgesamt kommunizierten Datenmenge entspricht.

Modell 2: Server zufällig. Hier ist ebenfalls keine Synchronisation zwischen den WF-Servern notwendig, da die Instanzen unabhängig voneinander ausgeführt werden und eine Instanz komplett von einem Server kontrolliert wird. Deshalb erreicht die Gesamtlast den optimalen Wert 100. Diese Last verteilt sich gleichmäßig auf die 6 WF-Server. Wegen der zufälligen Wahl des Servers findet keine Optimierung seines Ortes statt, weswegen sich für die Netz- und Gatewaylast sogar ein schlechterer Wert als im zentralen Fall ergibt.

Modell 3: Keine Migration. Wird der Server bei den Bearbeitern plaziert und sind keine Migrationen möglich, so erreicht die Serverlast den minimalen Wert 100. Da für verschiedene WF-Typen unterschiedliche Server verwendet werden können, verteilt sich diese Last auf die 6 Server. In der Simulation fällt die gesamte Last natürlich für den Server der Verwaltung an, da nur ein WF-Typ simuliert wird. Trotz der niedrigen Gesamtlast ist das Ergebnis für die Netzlast nicht besonders gut, da der Ort des WF-Servers nur für die gesamte WF-Instanz und nicht für einzelne Aktivitäten optimiert werden kann. Deshalb liegt der Server für die Aktivitäten 1-14 im falschen Teilnetz.

Modell 4: Keine globale Bearbeiterauflösung. Weil zum Teilnetz der Bearbeiter jeder Aktivität migriert werden muß, entstehen nicht rentable Migrationen (z.B. wegen Aktivität 5 ins Labor). Deshalb resultiert eine recht hohe Gesamtlast. Da sich diese auf nur 4 Server verteilt, ergibt sich bei diesem Ansatz die höchste Last pro Server. Aus denselben Gründen entsteht auch eine hohe Last pro Teilnetz. Prinzipiell wäre zwar vorstellbar, für den „zusammengesetzten Domain" der Stationen 3 Teilnetze zu verwenden, es entstünde aber keine Verbesserung für das Teilnetz des Servers. Dieses Teilnetz würde mit der gesamten Kommunikation des Domains belastet, da der WF-Server an jeder Kommunikation des Domains beteiligt ist. Der Grund dafür ist, daß die Clients der 3 Teilnetze ausschließlich mit diesem Server kommunizieren (wegen der lokalen Bearbeiterauflösung), und daß der Server die gesamte Kommunikation mit anderen Teilnetzen (Migrationen) abwickelt. Die Gesamtlast der Gateways ist bei diesem Ansatz besonders niedrig, da die 3 Stationen ein gemeinsames Teilnetz verwenden, so daß zwischen ihnen keine Kommunikation über ein Gateway stattfindet. Diese

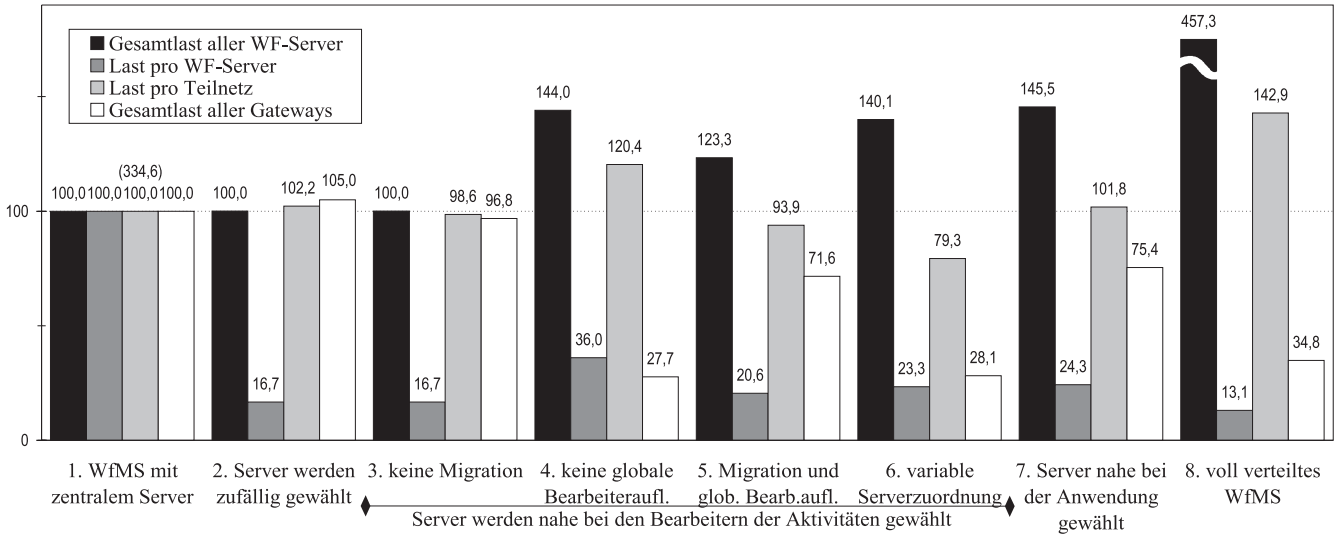

Abb. 9. Ergebnis der Simulation I

Last wird aber von nur 6 – und nicht wie bei den anderen Verteilungsmodellen 15 – Gateways umgesetzt.

Modell 5: Migration und globale Bearbeiterauflösung. Die erzeugte Gesamtlast ist kleiner als beim letzten Ansatz, da auf nicht rentable Migrationen verzichtet werden kann. Dies sind die Migration zur Aktivität 5 und die zu einer Station wegen den Aktivitäten 7-11 und 14. Daß letztgenannte Aktivitäten der Verwaltung zugeordnet werden, spart für einen Großteil der Instanzdaten eine zusätzliche Migration. Für die (mehrfach ausgeführten) Aktivitäten 12 und 13 ist es hingegen rentabel, sie vom Server einer Station kontrollieren zu lassen, da dann in 1/3 der Fälle die Bearbeitung auf dieses Teilnetz beschränkt ist. Die Migrationskosten für den entsprechenden Teil der Instanzdaten sind niedriger, als die durch die ständige Verwendung des falschen Servers entstehenden Kosten. Durch die Migrationen ist die Gesamtlast der Server natürlich größer als bei den Ansätzen 1, 2 und 3. Daraus ergibt sich eine höhere Last pro Server als bei verteilten Modellen ohne Migration. Sie ist aber deutlich niedriger als beim vorherigen Ansatz. Die Netzlast wird durch diesen Ansatz reduziert; die Last pro Teilnetz ist niedriger als bei den bisher beschriebenen Verteilungsmodellen.

Modell 6: Variable Serverzuordnungen. Weil alle Instanzdaten zusätzlich zum Server der entsprechenden Station migrieren müssen (Aktivität 8-14), ist die Gesamtlast und die Last pro Server größer als bei statischer Serverzuordnung. Da keine nicht rentablen Migrationen ausgeführt werden, sind diese Werte aber immer noch besser als bei den Ansätzen 4 und 7. Durch die variable Serverzuordnung ergibt sich das beste Ergebnis für die Belastung der Teilnetze. Das Ergebnis bei statischer Serverzuordnung (5.) ist 18% schlechter, alle anderen Ansätze erzeugen mindestens 24% mehr Last. Dies zeigt, daß variable Serverzuordnungen und das Ermitteln der optimalen Verteilung adäquate Mittel sind, um die Netzlast zu reduzieren.

Modell 7: Bei der Anwendung. Die unrentablen Migrationen (z.B. nach Aktivität 1) führen – verglichen mit den anderen Ansätzen, bei denen die Server geeignet gewählt werden – zu der höchsten Gesamtlast. Der Wert für die Netzlast ist nicht ganz so schlecht, da für die meisten Aktivitäten der optimale Server verwendet wird. Allerdings wirken sich auch hier die hohen Migrationskosten negativ aus.

Modell 8: Voll verteilt. Da es keine (expliziten) Server gibt, ist die in Abb. 9 für die Server angegebene Last die Gesamtlast aller Arbeitsstationen der Benutzer („Mini-Server“). Dieser Wert ist viel größer, als bei den bisher diskutierten Verteilungsmodellen, da die komplette WF-Instanz nach jeder Aktivität migriert (außer sie wird vom selben Benutzer wie ihr Vorgänger bearbeitet). Diese Gesamtlast verteilt sich auf die Arbeitsstationen der 35 Benutzer. Die Last pro Rechner ist kleiner als bei den Ansätzen mit Servern. Da aber die Arbeitsplatzrechner der Benutzer und nicht leistungsstarke Server damit belastet werden, ist der Wert vergleichsweise hoch. Die Belastung der Gateways ist ziemlich niedrig, da ein WF stets in das Teilnetz der Station migriert wird (Aktivität 7-14), zu der die entsprechenden Bearbeiter gehören (vgl. variable Serverzuordnungen). Migrationen zwischen Bearbeitern derselben OE finden lokal in einem Teilnetz statt, was die Gateways nicht belastet, aber zu der höchsten Belastung der Teilnetze führt.

3.2 Simulation II

3.2.1 Modellannahmen

Für die Simulation II wurde ein Prozeß gewählt, wie er für viele Umgebungen typisch ist. Während in der Simulation I kaum Abhängigkeiten zwischen den Bearbeitern der einzelnen Aktivitäten unterstellt werden, wird nun der in der Praxis häufig auftretende Fall betrachtet, daß derartige Abhängigkeiten bestehen. Es wird hier unterstellt, daß die OE des Bearbeiters der Startaktivität die OE (Station 1-6) für 9 darauf folgende Aktivitäten bestimmt. Jede Station verfügt über ein eigenes Teilnetz. Jeweils nach 3 bzw. 6 dieser Aktivitäten wird eine einzelne Aktivität im Labor ausgeführt, wegen der sich eine Migration nicht lohnt. Insgesamt ergibt sich die folgende Aktivitätenabfolge:

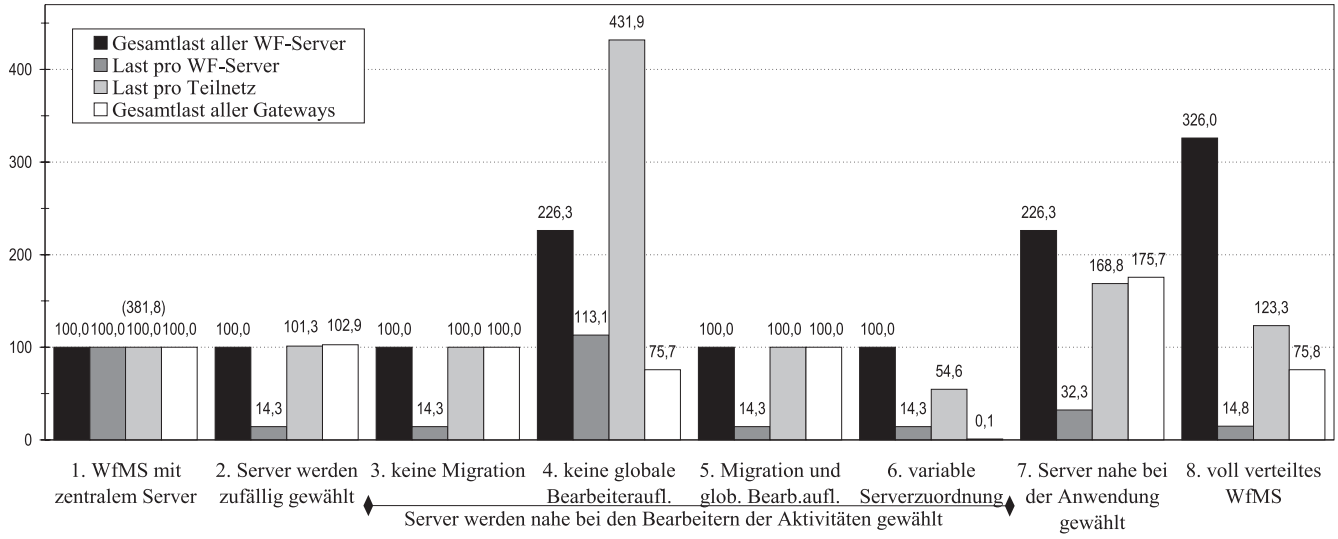

Abb. 10. Ergebnis der Simulation II

Startaktivität, $3\times$ Station i, $1\times$ Labor, $3\times$ Station i, $1\times$ Labor, $3\times$ Station i

3.2.2 Ergebnis der Simulation II

Bei dem in Abb. 10 dargestellten Simulationsergebnis wurde die Last des zentralen Falls wieder als Vergleichswert 100 verwendet. Auffällig ist die hohe Last bei Verteilungsmodellen, die unrentable Migrationen erzwingen (4., 7. und 8.). Die Kommunikationslast ist nur bei variablen Serverzuordnungen niedriger als im zentralen Fall.

3.2.3 Diskussion der Ergebnisse der Simulation II

Modell 1: Zentraler Server. Es wurde der (optimale) Server von Station 1 gewählt. Befindet sich der Server bei einer anderen Station, so ergibt sich dasselbe Ergebnis, beim Labor ergibt sich eine Teilnetzlast von 109,1. Die Last im Teilnetz des Servers beträgt bei allen Fällen 381,8. Bei diesem Szenario liegt die insgesamt von den Gateways zu bewältigende Kommunikationslast bei 45,5% der Gesamtlast der Teilnetze.

Modell 2: Server zufällig. Auch hier ergibt sich die Gesamtlast 100, da keine Migrationen stattfinden. Diese verteilt sich gleichmäßig auf die 7 WF-Server. Die Belastung des Netzwerks ist etwas größer als bei 1, da in $1/7$ der Fälle der Server des Labors gewählt wird. Da dieser Server sehr ungünstig ist, muß häufig über Teilnetzgrenzen hinweg kommuniziert werden.

Modell 3: Keine Migration. Die Gesamtlast ist identisch mit der des zentralen Falls. Diese Last verteilt sich rechnerisch auf die 7 WF-Server des Systems. Auch die Belastung der Teilnetze ist so groß wie im zentralen Fall.

Modell 4: Keine globale Bearbeiterauflösung. Aus den in Abschnitt 3.1 geschilderten Gründen müssen die Stationen bei lokaler Bearbeiterauflösung einen gemeinsamen Domain bilden. Die unrentablen Migrationen zum Server des Labors führen zu einer hohen Gesamtlast. Da sich diese auf nur 2 WF-Server bzw. 2 Teilnetze verteilt, ist auch deren Last sehr groß.

Modell 5: Migration und globale Bearbeiterauflösung. Es ergibt sich die gleiche Verteilung und damit dasselbe Ergebnis wie bei 3., da die in Frage kommenden Migrationen zum Labor unrentabel sind.

Modell 6: Variable Serverzuordnung. Die Startaktivität wird (statisch) vom Server der Station 1 kontrolliert. Alle weiteren Aktivitäten werden vom Server der betroffenen Station gesteuert, weil sich Migrationen ins Labor nicht lohnen. Da – außer der auf die Startaktivität folgenden (billigen) Migration – keine Migrationen stattfinden, ist die Gesamtlast der Server nicht meßbar größer als im zentralen Fall. Damit erreicht die Last pro Server den optimalen Wert 100. Die Kommunikationslast liegt bei 54,6. Werte unter 50 sind nicht möglich, da 50 bedeutet, daß jede Kommunikation anstelle von 2 Teilnetzen im zentralen Fall (Server und Client) nur noch ein Teilnetz belastet. Wie wichtig die richtige Wahl des Servers ist, zeigt sich auch an der von den Gateways umgesetzten Datenmenge. Diese beträgt hier nur 0,1 und ist damit wesentlich kleiner als bei allen anderen Verteilungsmodellen.

Modell 7: Bei der Anwendung. Die Migrationen zum Server des Labors wirken sich negativ aus. Wie bei der Simulation I wurde wieder wohlwollend angenommen, daß sich aufeinanderfolgende Anwendungen im selben Teilnetz befinden, so daß zwischen den Aktivitäten einer Station keine Migrationen notwendig sind.

Modell 8: Voll verteilt. Durch die ständigen Migrationen ergibt sich eine sehr hohe Gesamtlast für die WF-Server. Wie bei der Simulation I schlägt sich diese in der Belastung der Teilnetze und nicht so sehr in der Belastung der Gateways nieder.

4 Zusammenfassung

In diesem Beitrag wurden drei fundamentale Klassen von Verteilungsmodellen für WfMS identifiziert. Dies sind zum einen Systeme mit einem zentralen Server. Da dieser einen potentiellen Engpaß darstellt, muß bei einer hohen Benutzerzahl ein entsprechend leistungsstarker und damit teurer Rechner verwendet werden. Die dafür aktuell zur Verfügung stehende Technologie bildet eine Obergrenze für die Leistungsfähigkeit des WfMS. Weitere Klassen bilden Systeme mit mehreren Servern und Systeme ohne Server, bei denen die Ablaufsteuerung vollständig verteilt ist. Innerhalb dieser Klassen wurden noch Unterklassifikationen vorgenommen. Einige kommerzielle Systeme und Vorschläge aus dem Bereich der Forschung wurden bezüglich dieser Klassifikation eingeordnet und auf ihre Skalierbarkeit hin untersucht. Mit Hilfe einer Simulation wurden die verschiedenen Verteilungsmodelle quantitativ verglichen.

Um ein geeignetes Verteilungsmodell für ein WfMS in einem konkreten Anwendungsfall auswählen zu können, ist es notwendig, die jeweilige Anwendung genau zu analysieren. Ggf. sollten die verschiedenen Verteilungsmodelle dazu durch eine Simulation dieser Anwendung verglichen werden. Generell ist ein zentrales WfMS nur für Anwendungen geeignet, bei denen die Anzahl der Benutzer beschränkt ist. Voll verteilte Ansätze führen wegen ihrem hohen Grad an Verteilung der Information zu Nachteilen. Sie eignen sich eigentlich nur, wenn fast jeder Aktivität eindeutig ein Bearbeiter zugeordnet ist. Haben Sequenzen von Aktivitäten denselben Bearbeiter, so ergibt sich ein ähnliches Verhalten wie bei variablen Serverzuordnungen: der WF migriert einmal zum Rechner dieses Bearbeiters (in der entsprechenden OE) und verbleibt dort für die Ausführung mehrerer Aktivitäten. Da sich der WF sogar auf dem richtigen Rechner (nicht nur in der richtigen OE) befindet, entfällt dann sogar die Kommunikation für den Transfer der Parameterdaten des Aktivitätenprogramms.

Für unternehmensweite Anwendungen ist aber meist ein Verteilungsmodell mit mehreren Servern am geeignetsten. Bei diesem kann die Last auf eine überschaubare und geeignet gewählte Anzahl von WF-Servern verteilt werden. Es empfiehlt sich normalerweise nicht, den WF-Server zufällig zu wählen, da dann kein Vorteil aus dessen Wahl gezogen werden kann. Den Server nahe bei der Anwendung zu plazieren ist schwierig, da für Anwendungen häufig kein Ort vorgegeben ist. Außerdem kann diese Strategie zu schlechten Antwortzeiten führen, falls der WF-Server durch diese Festlegung weit von den Clients entfernt liegt. Deshalb ist es für das Kommunikationsverhalten i.d.R. am günstigsten, den Server nahe bei den Bearbeitern der Aktivitäten zu wählen. Ob mittels variabler Serverzuordnungen weitere Verbesserungen erreicht werden können, hängt von der betrachteten Anwendung ab. Dies ist immer dann der Fall, wenn die Bearbeiter der Aktivitäten von den Bearbeitern anderer Aktivitäten abhängen. Durch eine Einschränkung der Funktionalität eines WfMS ist ebenfalls eine Verringerung der Kommunikationskosten möglich. Werden z.B. in der zu realisierenden Anwendung die Prozesse fast komplett innerhalb nur einer OE ausgeführt, so kann auf Migrationen verzichtet werden. Gehören alle Bearbeiter einer Aktivität jeweils zur selben OE, so ist keine globale Bearbeiterauflösung notwendig.

Eine geeignete Verteilung eines WfMS erhöht dessen Skalierbarkeit und Verfügbarkeit. Aber auch für andere Anforderungen an ein WfMS ist dessen Verteilung eine wichtige Voraussetzung:

- Eine OE wird durch einen eigenen WF-Server vom Rechenzentrum unabhängiger und kann deshalb flexibler agieren.
- Bei einem weltweit verteilten Unternehmen wird der Einsatz eines WfMS durch die Verwendung mehrerer WF-Server überhaupt erst ermöglicht.
- Ein WfMS und damit die zugehörige Organisation kann durch das Hinzufügen von Servern dynamisch wachsen.
- Durch die Integration von Kunden in ein WfMS kann eine bessere Kundenorientierung erreicht werden.

Es bleibt noch zu bemerken, daß es noch ein großes Potential für die Entwicklung von Architekturen für unternehmensweite WfMS gibt. Dabei ist aber zu beachten, daß die Architektur auch Einfluß auf die Funktionalität eines Systems haben kann. Für die Leistungsfähigkeit ist es sicher von Vorteil, die Prozeßbeschreibungen zu zerteilen und zu kompilieren, so daß für jeden Knoten ein eigener Scheduler entsteht (vgl. METEOR$_2$ [MSKW96]). Allerdings macht dies dynamische Modifikationen der Ablaufbeschreibung [RD98, RBD99] nahezu unmöglich. Diese setzen faktisch eine interpretierende Ausführung des Prozeßgraphen voraus. Um beispielsweise feststellen zu können, ob irgendwelche Eingabeparameter durch die Modifikation einer WF-Instanz (z.B. Auslassen einer Aktivität) nicht mehr versorgt sind, muß zudem der gesamte Prozeßgraph vorliegen oder mit vertretbarem Aufwand konstruierbar sein. Aber auch andere Aspekte, die in heutigen Systemen ebenfalls noch nicht realisiert sind, stellen gewisse Anforderungen an die Architektur. Beispiele hierfür sind das Zeitmanagement, die Verwendung von Backup-Servern oder die Integration mobiler Clients.

Danksagung. Wir danken unseren Kollegen Manfred Reichert und Clemens Hensinger sowie den anonymen Gutachtern und dem Herausgeber für ihre hilfreichen Anregungen.

References

[AKA+94] G. Alonso, M. Kamath, D. Agrawal, A. El Abbadi, R. Günthör, C. Mohan. Failure Handling in Large Scale Workflow Management Systems. Technical Report RJ9913, IBM Almaden Research Center, November 1994

[AMG+95] G. Alonso, C. Mohan, R. Günthör, D. Agrawal, A. El Abbadi, M. Kamath. Exotica/FMQM: A Persistent Message-Based Architecture for Distributed Workflow Management. In: *Proceedings of the IFIP Working Conference on Information Systems for Decentralized Organisations*, Trondheim, August 1995

[BMR96] D. Barbará, S. Mehrotra, M. Rusinkiewicz. INCAs: Managing Dynamic Workflows in Distributed Environments. *Journal of Database Management, Special Issue on Multidatabases*, 7(1):5–15 (1996)

[BD97] T. Bauer, P. Dadam. A Distributed Execution Environment for Large-Scale Workflow Management Systems with Subnets and Server Migration. In: *Proc. Second IFCIS Confe-*

rence on Cooperative Information Systems, S. 99–108, Kiawah Island, SC, Juni 1997

[BD98a] T. Bauer, P. Dadam. Architekturen für skalierbare Workflow-Management-Systeme – Klassifikation und Analyse. Ulmer Informatik-Berichte 98-02, Universität Ulm, Fakultät für Informatik, Januar 1998

[BD98b] T. Bauer, P. Dadam. Variable Migration von Workflows in ADEPT. Ulmer Informatik-Berichte 98-09, Universität Ulm, Fakultät für Informatik, September 1998

[BD99] T. Bauer, P. Dadam. Efficient Distributed Control of Enterprise-Wide and Cross-Enterprise Workflows. In: *Proceedings Workshop Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications, 29. Jahrestagung der GI (Informatik '99)*, S. 25–32, Paderborn, Oktober 1999

[CGP+96] F. Casati, P. Grefen, B. Pernici, G. Pozzi, G. Sánchez. WIDE: Workflow Model and Architecture. CTIT Technical Report 96-19, University of Twente, 1996

[CGS97] S. Ceri, P. Grefen, G. Sánchez. WIDE – A Distributed Architecture for Workflow Management. In: *7th International Workshop on Research Issues in Data Engineering*, Birmingham, April 1997

[DKM+97] S. Das, K. Kochut, J. Miller, A. Sheth, D. Worah. ORBWork: A Reliable Distributed CORBA-based Workflow Enactment System for METEOR$_2$. Technical Report #UGA-CS-TR-97-001, Department of Computer Science, University of Georgia, Februar 1997

[DHL91] U. Dayal, M. Hsu, R. Ladin. A Transactional Model for Long-Running Activities. In: *Proceedings of the 17th International Conference on Very Large Data Bases*, S. 113–122, Barcelona, September 1991

[EG96] J. Eder, H. Groiss. Ein Workflow-Managementsystem auf der Basis aktiver Datenbanken. In J. Becker, G. Vossen, ed., Geschäftsprozeßmodellierung und Workflow-Management. San Francisco, CA: International Thomson Publishing, 1996

[Elm92] A.K. Elmagarmid. Database Transaction Models for Advanced Applications. Los Altos, CA: Morgan Kaufmann, 1992

[GHS95] D. Georgakopoulos, M. Hornick, A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases* 3(2):119–152 (1995)

[HS96] P. Heinl, H. Schuster. Towards a Highly Scaleable Architecture for Workflow Management Systems. In: *Proceedings of the 7th International Workshop on Database and Expert Systems Applications, DEXA'96*, S. 439–444, Zurich, September 1996

[IBM96] IBM. FlowMark – Modeling Workflow, Version 2 Release 2, Document Number: SH19-8241-01, 2. edition, Februar 1996

[Jab97] S. Jablonski. Architektur von Workflow-Mangement-Systemen. *Informatik Forsch. Entw., Themenheft Workflow-Management*, 12(2):72–81 (1997)

[KAGM96] M. Kamath, G. Alonso, R. Günthör, C. Mohan. Providing High Availability in Very Large Workflow Management Systems. In: *Proceedings of the 5th International Conference on Extending Database Technology*, S. 427–442, Avignon, März 1996

[Kar94] B. Karbe. Flexible Vorgangssteuerung mit ProMInanD. In U. Hasenkamp, S. Kirn, M. Syring, editor, *CSCW – Computer Supported Cooperative Work*, S. 117–133. Reading, MA: Addison-Wesley 1994

[Lan92] H. Langendörfer. Leistungsanalyse von Rechensystemen: Messen, Modellieren, Simulation. München: Carl Hanser 1992

[Ley97] F. Leymann. Transaktionsunterstützung für Workflows. *Informatik Forsch. Entw., Themenheft Workflow-Management* 12(2):82–90 (1997)

[MSKW96] J. A. Miller, A. P. Sheth, K. J. Kochut, X. Wang. CORBA-Based Run-Time Architectures for Workflow Management Systems. *J. Database Management, Special Issue on Multidatabases* 7(1):16–27 (1996)

[MWW+98] P. Muth, D. Wodtke, J. Weißenfels, A. Kotz-Dittrich, G. Weikum. From Centralized Workflow Specification to Distributed Workflow Execution. *J. Intell. Inform. Syst., Special Issue on Workflow Management Systems* 10(2):159–184 (1998)

[OMG95] OMG. Object Management Architecture Guide. New York: Wiley 1995

[RBD99] M. Reichert, T. Bauer, P. Dadam. Enterprise-Wide and Cross-Enterprise Workflow-Management: Challenges and Research Issues for Adaptive Workflows. In: *Proceedings Workshop Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications, 29. Jahrestagung der GI (Informatik '99)*, S. 56–64, Paderborn, Oktober 1999

[RD98] M. Reichert, P. Dadam. ADEPT$_{flex}$ – Supporting Dynamic Changes of Workflows Without Losing Control. *J. Intell. Inform. Syst., Special Issue on Workflow Management Systems* 10(2):93–129 (1998)

[Sch93] A. Schill. DCE – Das OSF Distributed Environment, Einführung und Grundlagen. Berlin Heidelberg New York: Springer-Verlag 1993

[SM96] A. Schill, C. Mittasch. Workflow Management Systems on Top of OSF DCE and OMG CORBA. *Distrib. Syst. Eng.* 3(4):250–262 (1996)

[SNS99] H. Schuster, J. Neeb, R. Schamburger. A Configuration Management Approach for Large Workflow Management Systems. In: *Proceedings of the International Joint Conference on Work Activities Coordination and Collaboration*, San Francisco, Februar 1999

[SK97] A. Sheth, K. J. Kochut. Workflow Applications to Research Agenda: Scalable and Dynamic Work Coordination and Collaboration Systems. In: *Proceedings of the NATO Advanced Study Institute on Workflow Management Systems and Interoperability*, S. 12–21, Istanbul, August 1997

[Sie95] Siemens Nixdorf. WorkParty – Benutzerhandbuch, Version 2.0, August 1995

[UNI92] UNIX System Laboratories. TUXEDO System Release 4.1 – Product Overview and Master Index. Englewood Cliffs, NJ: Prentice Hall 1992

[WWK+97] G. Weikum, D. Wodtke, A. Kotz-Dittrich, P. Muth, J. Weißenfels. Spezifikation, Verifikation und verteilte Ausführung von Workflows in MENTOR. *Informatik Forsch. Entw., Themenheft Workflow-Management* 12(2):61–71 (1997)

[WWWK96a] J. Weißenfels, D. Wodtke, G. Weikum, A. Kotz-Dittrich. The Mentor Architecture for Enterprise-wide Workflow Management. In: *Proceedings of the NSF Workshop on Workflow and Process Automation in Information Systems*, S. 69–73, Athens, Mai 1996

[Wel83] P.D. Welch. The Statistical Analysis of Simulation Results. In S.S. Lavenberg, editor, *Computer Performance Modeling Handbook*, S. 267–329. London: Academic Press 1983

[WHKS98] M. Weske, J. Hündling, D. Kuropka, H. Schuschel. Objektorientierter Entwurf eines flexiblen Workflow-Management-Systems. *Informatik Forsch. Entw.* 13(4):179–195 (1998)

[WWWK96b] D. Wodtke, J. Weißenfels, G. Weikum, A. Kotz-Dittrich. The Mentor Project: Steps Towards Enterprise-Wide Workflow Management. In: *Proceedings of the 12th IEEE International Conference on Data Engineering*, S. 556–565, New Orleans, LA, März 1996

THOMAS BAUER studierte Informatik in Ulm. Seit seinem Diplom 1995 ist er wissenschaftlicher Mitarbeiter der Universität Ulm, Abteilung Datenbanken und Informationssysteme. Seine Interessen liegen in den Bereichen Workflow-Management und Skalierbarkeit.

PETER DADAM ist seit 1990 Professor für Informatik an der Universität Ulm und Leiter der Abteilung Datenbanken und Informationssysteme. Davor war er Leiter der Abteilung Advanced Information Management am Wissenschaftlichen Zentrum der IBM in Heidelberg. Seine aktuellen Arbeitsgebiete sind: Verteilte, kooperative Informationssysteme, Workflow-Management, Datenbanktechnologie und deren Anwendungen in anspruchsvollen medizinischen und technisch/wissenschaftlichen Anwendungsgebieten.