Variable Migration von Workflows in *ADEPT*

Thomas Bauer, Peter Dadam
Abteilung Datenbanken und Informationssysteme
Universität Ulm
{bauer, dadam}@informatik.uni-ulm.de

Zusammenfassung

Bei großen, unternehmensweiten Workflow (WF)-Anwendungen können die Belastung der WF-Server und das Kommunikationsaufkommen in den zugrundeliegenden Teilnetzen zum Engpaß werden. In diesem Beitrag wird gezeigt, wie eine verteilte WF-Steuerung dergestalt realisiert werden kann, daß die Belastung der beteiligten Komponenten zur Ausführungszeit minimiert wird. Dazu kann die WF-Steuerung einer Instanz von einem WF-Server auf einen anderen migrieren. Ausgehend von den Bearbeiterzuordnungen wird zur Modellierungszeit für alle Aktivitäten jeweils der optimale WF-Server ermittelt. Da Bearbeiterzuordnungen von vorangegangenen Aktivitäten abhängen können, ist eine statische Serverzuordnung nicht immer angebracht. Deshalb werden sog. logische Serverzuordnungsausdrücke eingeführt, durch die eine dynamische Serverzuordnung ohne aufwendige Laufzeitanalysen möglich wird.

1 Motivation und Einführung

Workflow-Management-Systeme (WfMSe) stellen eine vielversprechende Technologie für die Realisierung vorgangsorientierter, unternehmensweiter verteilter Anwendungssysteme dar [LR97]. Durch die Separierung der Ablauflogik des Geschäftsprozesses (GP) vom eigentlichen Applikationscode lassen sich derartige Anwendungen in der Regel rascher und mit weniger Fehlerrisiken behaftet erstellen, als dies mit konventioneller Implementierungstechnik möglich ist, bei der die Prozeßlogik „hart verdrahtet“ im Applikationscode enthalten ist. Aus denselben Gründen lassen sich WfMS-basierte Anwendungssysteme in der Regel sehr viel rascher und kostengünstiger an Änderungen des Geschäftsprozesses anpassen. WfMS-basierte Anwendungssysteme weisen in den meisten Fällen die in Abb. 1 illustrierte Softwarearchitektur auf, die auch wir im folgenden unterstellen wollen. Ein oder mehrere WF-Server verwalten den aktuellen Status der laufenden Prozeßinstanzen, sie aktualisieren unter anderem (entweder aktiv

oder „on demand“) die Arbeitslisten der WF-Klienten und informieren die Klienten, welche Anwendung (mit welchen Aufrufparametern) mit einem bestimmten Prozeßschritt verbunden ist. Der Aufruf einer Applikationsfunktion, d.h. eines Anwendungsprogramms bzw. einer bestimmten Funktion eines Anwendungsprogramms, erfolgt in der Regel durch den jeweiligen Klienten. Wie in Abb. 1 angedeutet, „kennt“ nur der WF-Server den Prozeßablauf. Die Klienten rufen im wesentlichen nur einzelne Applikationsfunktionen auf. Änderungen im Prozeß schlagen daher, eine geeignete Implementierung der Prozeßablaufsteuerung und Applikationsfunktionen vorausgesetzt, nicht (oder nur in geringem Maße) auf die Implementierung der Klienten durch.

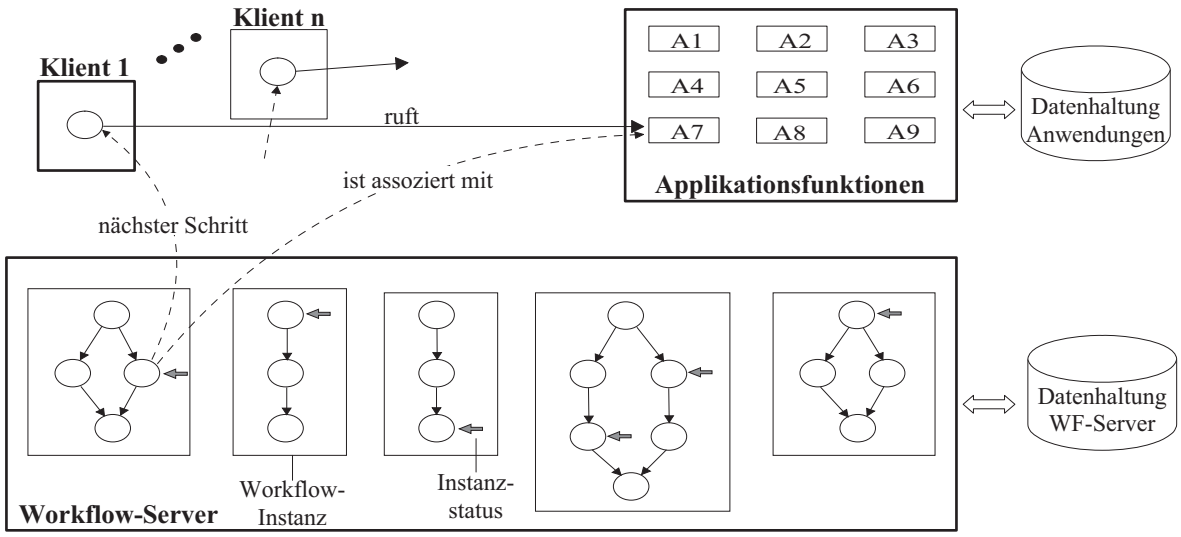

Abbildung 1 Aufbau eines WfMS-basierten Anwendungssystems.

Aber nicht nur die raschere Realisierung und Anpassung vorgangsorientierter Anwendungssysteme sprechen für den Einsatz eines WfMSs. Moderne WfMSe erlauben darüber hinaus, den einzelnen Prozeßschritten auf sehr variable Weise geeignete Bearbeiter zuzuweisen, denen der jeweilige Prozeßschritt, wenn er zur Ausführung ansteht, durch das System zur Bearbeitung angeboten wird. So kann man zum Beispiel festlegen, daß ein bestimmter Prozeßschritt gleichzeitig allen Personen angeboten wird, welche eine bestimmte Funktion („Rolle“) innehaben, oder man kann etwa bestimmen, daß es derselbe Bearbeiter sein soll, der auch den vorherigen Schritt ausgeführt hat, oder man kann das System anweisen, explizit eine andere Person für die Bearbeitung zu wählen, etwa um ein „Vier-Augen-Prinzip“ zu realisieren. Weitere Varianten ergeben sich in Verbindung mit Vertretungsregelungen.

Die oben skizzierte Funktionalität und Flexibilität haben aber auch ihren Preis. Sehr viele Ausführungsentscheidungen, allen voran die Bestimmung der für eine bestimmte Tätigkeit zu einem bestimmten Zeitpunkt konkret in Frage kommenden Bearbeiter, können WfMS-

seitig in der Regel erst zur Laufzeit getroffen werden. Dies bedeutet, daß in der Mehrzahl der Fälle für jeden zur Ausführung anstehenden Arbeitsschritt Kommunikation zwischen Server und Klient, in manchen Fällen sogar zwischen dem Server und allen seinen Klienten, erforderlich wird. Der Durchsatz bzw. das Antwortzeitverhalten eines WfMS-basierten Anwendungssystems wird daher ganz wesentlich von der Leistungsfähigkeit bzw. der Belastung der WF-Server und des Kommunikationsnetzes bestimmt. Insbesondere im Hinblick auf große, unternehmensweit ablaufende, „transaktionale" WfMS-basierte Anwendungen mit vielleicht hunderten von Benutzern und tausenden von aktiven WF-Instanzen müssen geeignete Konzepte entwickelt werden, die sowohl eine Überlastung der WF-Server als auch des Kommunikationssystems vermeiden helfen. In diesem Beitrag wollen wir uns diesbezüglich auf den Kommunikationsaspekt konzentrieren.

Wir gehen im folgenden davon aus, daß das Kommunikationsnetz, wie heute üblich, aus mehreren physischen Teilnetzen aufgebaut ist. Die Herausforderung besteht nun darin, das Kommunikationsaufkommen zwischen WF-Server(n) und WF-Klienten so auf die verschiedenen Teilnetze zu verteilen, daß kein Teilnetz überlastet ist. Der grundsätzliche Ansatz besteht darin, die WFs derart in logische Abschnitte aufzuteilen und jeden dieser Abschnitte durch einen geeigneten WF-Server so steuern zu lassen, daß die Kommunikation zwischen dem jeweils steuernden WF-Server und den für diesen „WF-Abschnitt" relevanten Klienten möglichst innerhalb eines Teilnetzes abgewickelt werden kann. Es wird also angestrebt, teilnetzübergreifende Kommunikation möglichst zu vermeiden. Dies bedeutet, daß die Steuerung einer WF-Instanz unter Umständen abschnittsweise von verschiedenen WF-Servern wahrgenommen wird. Es findet also eine „Migration" der „WF-Steuerung" statt.

In [BD97] haben wir ein Werkzeug bzw. ein Verfahren beschrieben, das dem WF-Modellierer zu analysieren erlaubt, welche Abschnitte einer gegebenen WF-Vorlage welchem WF-Server zur Steuerung zugeordnet werden sollten, um später, bei Ausführung von WF-Instanzen dieses Typs, eine möglichst günstige Lastverteilung zu erhalten. Das Verfahren analysiert im Prinzip, welche Rolle (Kompetenz) für die Ausführung einer Tätigkeit T erforderlich ist und welche Klienten in den Teilnetzen L_1 und L_2 diese Rolle aufweisen. Daraus werden dann die Wahrscheinlichkeiten abgeleitet, mit denen Aufgaben vom Typ T von einem Klienten in Teilnetz L_1 bzw. von einem Klienten in Teilnetz L_2 ausgeführt werden, und die erwarteten Kommunikationskosten bestimmt. Im eigentlichen Optimierungsschritt wird dann die WF-Vorlage so partitioniert und abschnittsweise den vorhandenen WF-Servern der Teilnetze zugeordnet, daß die Kommunikation zur Laufzeit zu einem (wahrscheinlich) hohen Grad jeweils lokal innerhalb eines Teilnetzes abgewickelt werden kann. Der für die Migration erforderliche Zusatzaufwand wird bei diesen Kostenabschätzungen natürlich berücksichtigt.

Das in [BD97] vorgestellte Verfahren führt diese Analyse und die Festlegung der WF-Server vollständig zur Modellierungszeit durch. Das heißt bereits zum Modellierungszeitpunkt wird die Entscheidung für oder gegen eine bestimmte Partitionierung gefällt und auch, welcher WF-Server ggf. die jeweilige „Abschnittssteuerung“ übernimmt. Man kann deshalb auch von einer „statisch definierten Migration“, im folgenden kurz „statische Migration“ genannt, sprechen. Die statische Migration führt dann zu guten Ergebnissen, wenn Bearbeiterzuordnungen der Art "$Rolle = Arzt$", "$Rolle = Pflegekraft \wedge Abteilung = Chirurgie$" oder "$Dr.\ Brinkmann \vee Dr.\ Frank$" verwendet werden, da in diesen Fällen die Menge der Bearbeiter (als Voraussetzung für die Berechnung der Wahrscheinlichkeiten (siehe oben)) bereits zur Modellierungszeit ermittelt werden kann.[1] Es kann jedoch auch Bearbeiterzuordnungen geben, die von anderen Aktivitäten abhängen (abhängige Bearbeiterzuordnungen). So kann z.B. festgelegt sein, daß derselbe Arzt, der einen Patienten untersucht hat (Aktivität x), auch den zugehörigen Arztbrief schreiben soll (Nachfolgeaktivität k'), oder daß ein Patient von einer Pflegekraft derjenigen Abteilung versorgt werden soll (Aktivität k), auf der er zuvor untersucht wurde (siehe Abb. 2). Erst nachdem Aktivität x ausgeführt wurde, steht in diesem Fall die Organisationseinheit (OE) der Bearbeiter der weiteren Aktivitäten (k und k') fest.

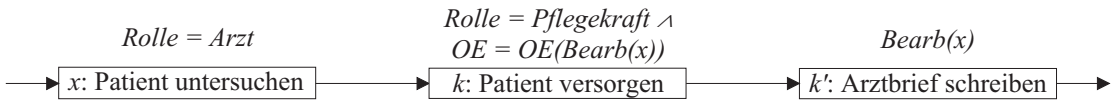

Abbildung 2 Beispiele für Bearbeiterzuordnungen, die von anderen Aktivitäten abhängig sind.

Abhängige Bearbeiterzuordnungen lassen sich im Prinzip zwar mittels bedingter Wahrscheinlichkeiten ebenfalls beschreiben und könnten daher im Prinzip in analoger Weise behandelt werden. Dies führt jedoch aufgrund der statistisch gesehen kleinen Fallzahl, in Verbindung mit einer meist großen kombinatorischen Vielfalt, in der Regel nicht zu wirklich brauchbaren Resultaten. Das heißt die Festlegung des WF-Servers zur Modellierungszeit wird in diesen Fällen häufig nicht zu den gewünschten optimalen Kommunikationskosten führen. Eine Festlegung zur Laufzeit wäre hier günstiger. Die Unterstützung der Serverauswahl zur Laufzeit, also von variablen Migrationen, ist das zentrale Thema dieses Beitrages.

Im nächsten Abschnitt werden die Voraussetzungen für die Anwendbarkeit des Verfahrens geklärt. Abschnitt 3 diskutiert die Problemstellung und mögliche Lösungsansätze. Den Kern des Beitrags bildet Abschnitt 4 mit der Beschreibung der Algorithmen zum Ermitteln einer optimalen Verteilung. In Abschnitt 5 werden verwandte Ansätze diskutiert. Der Artikel schließt mit einer Zusammenfassung und einem Ausblick auf weitere Arbeiten.

[1] Wobei vereinfachend unterstellt wird, daß sich an den Größenordnungen – und damit an den Wahrscheinlichkeitsverteilungen – bis zur Ausführung von Instanzen dieses WF-Typs nichts wesentlich ändert.

2 Einbettung und Rahmenbedingungen

Der nachstehend vorgestellte Ansatz beschreibt, wie Lastverteilung mittels variabler Migration in $ADEPT_{distribution}$, der verteilten Variante des *ADEPT*-Systems [DKR+95, RD98], realisiert wird. Um möglichst allgemeingültig zu bleiben, werden wir im folgenden jedoch von den speziellen Eigenschaften des *ADEPT*-Systems und den damit verbundenen speziellen Herausforderungen, wie die Unterstützung von Ad-hoc-Abweichungen, abstrahieren. In der weiteren Diskussion wird lediglich von den folgenden Annahmen ausgegangen:

1. Das WfMS verfügt über ein Organisationsmodell, in dem sowohl Personen sowohl mit OEen in der Aufbauorganisation als auch mit Rollen assoziiert werden können.
2. Die Zuordnung von Bearbeitern zu einem zur Bearbeitung anstehenden WF-Schritt (Tätigkeit) erfolgt in der Regel zur Laufzeit mittels eines Auswahlprädikats, wie z.B. "$Rolle = Arzt \wedge Organisationseinheit = Radiologie$". Es wird angenommen, daß sich, mit Ausnahme von abhängigen Bearbeiterzuordnungen, der tatsächliche Bearbeiter einer Aktivität unabhängig von anderen Arbeitsschritten ergibt.
3. Das WfMS verwendet mehrere Teilnetze sowie mehrere WF-Server. Zur Vereinfachung der Diskussion wird angenommen, daß jedes betrachtete Teilnetz über einen (und nur einen) eigenen WF-Server verfügt, und daß jeder dieser WF-Server im Prinzip für jeden WF-Typ und jeden WF-Abschnitt zur Steuerung eingesetzt werden kann. Jeder Benutzer des WfMSs (WF-Klient) hat ein fest zugeordnetes Teilnetz.
4. Jeder WF-Server kann alle beim WfMS angemeldeten WF-Klienten bedienen, nicht nur diejenigen, die sich in seinem Teilnetz (Domain) befinden.
5. Die potentiell möglichen Bearbeiter für einen gegebenen WF-Schritt befinden sich nicht notwendigerweise immer im selben Teilnetz.[2]

3 Problemstellung

Wie bereits oben erwähnt, ist es das Ziel unseres Ansatzes, Serverzuordnungen zu erhalten, durch welche die Kommunikationskosten minimiert werden. Dabei soll insbesondere der oben beschriebene Fall abhängiger Bearbeiterzuordnungen berücksichtigt werden. Für die Zuordnung von WF-Servern zu Aktivitäten gibt es im Prinzip mehrere Vorgehensweisen: Die einfachste und am häufigsten verwendete Methode ist, jeder Aktivität zur Modellierungszeit oder beim Start des WFs fest (statisch) einen Server zuzuordnen. Für WFs mit abhängigen

[2] Wäre dies immer der Fall, dann wäre keine variable Migration erforderlich. Alle Entscheidungen könnten bereits zur Modellierungszeit getroffen werden (statische Migration).

Bearbeiterzuordnungen, welche in der Praxis häufig anzutreffen sind, ist diese Lösung unbefriedigend. Die besten Serverzuordnungen würde man natürlich erhalten, wenn jeweils nach Beendigung einer Aktivität der WF-Server für die Nachfolgeaktivität aktuell bestimmt werden würde. Für diese Auswahl stünden dann die kompletten und aktuellen Laufzeitdaten der WF-Instanz zur Verfügung, so daß der tatsächlich am besten geeignete WF-Server bestimmt werden könnte. Leider scheidet diese Lösung im allgemeinen aus Aufwandsgründen aus. Das Durchführen der notwendigen Analysen würden die WF-Server stark belasten und damit die Performanz des WfMSs beeinträchtigen.[3]

In $ADEPT_{distribution}$ wird deshalb eine Strategie verfolgt, welche eine statische Vorberechnung (zur Modellierungszeit) mit dynamischen Aspekten (Berücksichtigung von Laufzeitdaten) kombiniert und so im wesentlichen die Vorteile der beiden Vorgehensweisen in sich vereint. Anstelle einer festen WF-Server-Zuordnung werden bei der Analyse, wo erforderlich bzw. sinnvoll, logische Serverzuordnungsausdrücke (im folgenden kurz: Serverzuordnungen) erzeugt, die Laufzeitdaten der WF-Instanz referenzieren und so auf einfache (und effiziente) Weise eine Festlegung des konkreten WF-Servers zur Laufzeit ermöglichen. In dem in Abb. 2 dargestellten Beispiel ergeben sich (variable) Serverzuordnungen der Art: $ServZuordn(k) =$ "Server im Domain des Bearbeiters von Aktivität x". Für die Aktivität x hingegen wird der Server weiterhin statisch gewählt. Für die Aktivitäten k und k' ist die OE ihrer Bearbeiter nach Ausführung von Aktivität x bekannt, so daß zur Ausführungszeit der optimale Server gewählt werden kann.

Die Frage ist nun, wie man zu geeigneten variablen Serverzuordnungen für die Aktivitäten kommt. Eine Möglichkeit wäre, daß sie vom WF-Modellierer bei der Modellierung einfach vorgegeben werden (analog zu den Bearbeiterzuordnungen). Das Problem hierbei ist, daß der WF-Modellierer, ohne weitere Unterstützung bzw. Information, die durch die Verteilung entstehende Last in den Teilnetzen in der Regel kaum wird abschätzen können. Es ist deshalb notwendig, daß er beim Ermitteln von geeigneten Serverzuordnungen seitens des WfMSs aktiv unterstützt wird. Das heißt, die WfMS-Modellierungskomponente muß für die in Betracht gezogenen Serverzuordnungen die Belastung jeder Systemkomponente (Server, Teilnetz, Gateway) geeignet abschätzen, so daß mögliche Überlastungen entdeckt und vermieden werden können.

Für diese Lastanalyse wird ein Kostenmodell benötigt, das es ermöglicht auszurechnen, für welche Komponenten welche Kosten (als gewichtete Summe von Übertragungsmenge, Verarbeitungsmenge etc.) bei der jeweils betrachteten WF-Server-Festlegung entstehen. Dieses

[3]Zur Erinnerung: Wir betrachten „Produktions-WfMSe“ mit potentiell sehr vielen gleichzeitig aktiven WF-Instanzen.

Kostenmodell und die Bestimmung der Serverzuordnungen wird im nächsten Abschnitt beschrieben.

4 Ermitteln der Serverzuordnungen

Im folgenden wird beschrieben, wie die optimale Verteilung der Aktivitäten einer WF-Vorlage auf WF-Server berechnet werden kann. Insbesondere wird auf die Bestimmung der Kosten und der Wahrscheinlichkeitsverteilungen (WVen) eingegangen.

4.1 Kostenmodell

Ein Kostenmodell zur Berechnung der Kosten bzw. zur Bestimmung der Last für die unter Performanz-Aspekten kritischen Komponenten des WfMSs (Server, Teilnetz, Gateway) muß zumindest die folgenden Kosten berücksichtigen:

- Kosten für den Parametertransfer beim Starten und Beenden von Aktivitätenprogrammen
- Kosten für das Aktualisieren von Arbeitslisten
- Migrationskosten
- Kosten für die Kommunikation von Aktivitätenprogrammen mit externen Datenquellen

Zusätzlich zu den bereits im WF- und Organisationsmodell vorhandenen Informationen werden je WF-Typ die (geschätzte) Häufigkeit von WF-Instanziierungen sowie je Aktivität die im Mittel zu kommunizierende Datenmenge (z.B. die Größe von Parametern)[4] benötigt.

Bezeichne im folgenden $prob_k(i,j)$ die Wahrscheinlichkeit, daß Aktivität k vom Server in Teilnetz i kontrolliert und von einem Benutzer in Teilnetz j bearbeitet wird. $prob'_{k,l}(i,j)$ sei die Wahrscheinlichkeit, daß beim Übergang von Aktivität k nach l von Server i nach j migriert werden muß. Diese Wahrscheinlichkeiten hängen von den gewählten Serverzuordnungen ab. Wie sie bestimmt werden können, wird später gezeigt. Für den Moment wollen wir sie als gegeben betrachten. Im folgenden werden Formeln entwickelt, die das anfallende bzw. zu transportierende Datenvolumen für jede Aktion (Aktivität ausführen, Migration, ...) beschreiben, und zwar getrennt für jede Komponente des WfMSs. Diese Formeln werden anschließend in eine Gesamtformel eingesetzt, mit der die in einer Komponente entstehende Last berechnet werden kann.

Der Erwartungswert für das Datenvolumen (im folgenden kurz: Datenvolumen) bei der Aktivitätenausführung (Transport der Ein- und Ausgabeparameterdaten) berechnet sich damit wie folgt: Das Datenvolumen, das an Server i für die Ausführung von Aktivität k entsteht,

[4] Es bietet sich an, diese Information bei der Modellierung gleich mitzuerfassen.

ergibt sich als Wahrscheinlichkeit, daß Server i die Aktivität k kontrolliert, multipliziert mit der zu transportierenden Datenmenge:

$Vol^{Akt}_{Server,i}(k) = \left(\sum_j prob_k(i,j)\right) \cdot (in\_parameter\_size_k + out\_parameter\_size_k)$

Die Belastung für die Teilnetze ergibt sich folgendermaßen: Im Teilnetz i findet Kommunikation statt, wenn entweder der Server in i liegt oder wenn ein Bearbeiter aus Teilnetz i gewählt wird. Trifft beides zu, so wird die Kommunikation nur einmal gezählt:

$Vol^{Akt}_{TN,i}(k) = \left(\sum_j prob_k(i,j) + \sum_{j \neq i} prob_k(j,i)\right) \cdot (in\_parameter\_size_k + out\_parameter\_size_k)$

Das anfallende Datenvolumen am Gateway von Teilnetz i nach j $(i \neq j)$ ergibt sich wie folgt:

$Vol^{Akt}_{GW,i,j}(k) = prob_k(i,j) \cdot in\_parameter\_size_k + prob_k(j,i) \cdot out\_parameter\_size_k$

Die Datenvolumina für das Aktualisieren der Arbeitslisten ($Vol^{AL}(k)$) und für die Kommunikation mit externen Datenquellen ($Vol^{Ext}(k)$) können auf ähnliche Weise berechnet werden (für Details siehe [BD98b]).

Von besonderem Interesse sind in dem betrachteten Kontext natürlich die Migrationskosten beim Übergang von Aktivität k nach l. Hierbei müssen sowohl ein- wie auch ausgehende Migrationen berücksichtigt werden. Das Datenvolumen für den Server i ergibt sich wie folgt:

$Vol^{Migr}_{Server,i}(k,l) = \sum_{j \neq i} \left(prob'_{k,l}(i,j) + prob'_{k,l}(j,i)\right) \cdot instance\_size_{k,l}$

Dabei ist zu beachten, daß keine Kommunikation stattfindet, wenn die Server für Aktivität k und l identisch sind (deshalb: $j \neq i$). Das durch die Migration verursachte Datenvolumen in den betroffenen Teilnetzen ist genau so groß wie bei den Servern:

$Vol^{Migr}_{TN,i}(k,l) = Vol^{Migr}_{Server,i}(k,l)$.

Die Gateways müssen hierbei die folgenden Datenmengen transportieren:

$Vol^{Migr}_{GW,i,j}(k,l) = prob'_{k,l}(i,j) \cdot instance\_size_{k,l}$

Für jede Systemkomponente müssen diese Datenmengen noch mit den Ausführungshäufigkeiten[5] der einzelnen Aktivitäten $h(k)$ bzw. der Migrationen $h'(k,l)$ multipliziert und für alle Aktivitäten aller WF-Typen aufaddiert werden, um die entsprechende Last dieser Komponente zu bestimmen. Für die Server ergibt sich die Last ($Load_{TN,i}$ und $Load_{GW,i,j}$ analog) wie folgt:

$$Load_{Server,i} = \sum_{WF} \sum_{k \in WF} \Big(h(k) \cdot Vol^{Akt}_{Server,i}(k) + h(k) \cdot Vol^{AL}_{Server,i}(k) + h(k) \cdot Vol^{Ext}_{Server,i}(k) + \sum_{l \neq k} h'(k,l) \cdot Vol^{Migr}_{Server,i}(k,l) \Big)$$

Ein globales Kostenmaß erhält man, indem man die Last aller Komponenten mit komponentenspezifischen Kostenfaktoren $C_{Server,i}$, $C_{TN,i}$ und $C_{GW,i,j}$ gewichtet und aufaddiert. Diese

[5] Die Ausführungshäufigkeiten berechnen sich aus den Starthäufigkeiten der einzelnen WF-Typen, der Wahrscheinlichkeit für den entsprechenden Zweig bei einem OR-Split und der durchschnittlichen Anzahl von Durchläufen von Schleifen. Diese Daten müssen vom WF-Modellierer vorgegeben werden.

Kostenfaktoren geben an, wie teuer es ist, ein Byte über den Server, das Teilnetz bzw. das Gateway zu transportieren. Der Modellierer kann damit beeinflussen, wie stark jede Komponente belastet werden soll. Durch einen großen Wert wird erreicht, daß z.B. eine WAN-Verbindung (Gateway) wenig verwendet wird. Die zu minimierende Zielfunktion ergibt sich damit als:
$K = \sum_i C_{Server,i} \cdot Load_{Server,i} + \sum_i C_{TN,i} \cdot Load_{TN,i} + \sum_i \sum_{j \neq i} C_{GW,i,j} \cdot Load_{GW,i,j}$
Im folgenden Abschnitt wird beschrieben, welche Serverzuordnungsausdrücke für die Aktivitäten möglich sind. In Abschnitt 4.3 wird gezeigt, wie die Serverzuordnungen so gewählt werden können, daß K minimal wird.

4.2 Serverzuordnungsausdrücke

Bei der statischen Migration werden als Serverzuordnungsausdrücke die Server-IDs von WF-Servern verwendet. Im Fall variabler Serverzuordnungen sind, wie oben bereits erwähnt, als Serverzuordnungen auch Ausdrücke erlaubt, die Laufzeitdaten der Instanz referenzieren. In den meisten praktisch relevanten Fällen wird man sich hierbei auf die Lokation des Servers oder des Bearbeiters der vorangegangenen Aktivitäten beschränken. In Sonderfällen können auch weitere WF-interne Daten (z.B. der Startzeitpunkt einer Aktivität), beliebige mathematische Funktionen, logische Ausdrücke oder die Einbeziehung WfMS-externer Daten (z.B. Parameterdaten von Aktivitäten) Sinn machen. Während Serverzuordnungen der erstgenannten Art (s.u.: 1. - 4.) systemseitig vom Modell abgeleitet werden können, müssen die Zuordnungsausdrücke für die Sonderfälle (5.) explizit vom Modellierer vorgegeben werden.
Mögliche Serverzuordnungen:

1. $ServZuordn(k) = S_i$
   Der Aktivität k wird der Server S_i fest (statisch) zugeordnet.
2. $ServZuordn(k) = Server(x)$
   Die Aktivität k soll vom selben Server kontrolliert werden wie die Aktivität x.
3. $ServZuordn(k) = Domain(Bearb(x))$
   Der Aktivität k wird der Server zugewiesen, der sich im Domain des Bearbeiters befindet, der die Aktivität x ausgeführt hat. Ist Aktivität k derselbe Bearbeiter zugeordnet wie Aktivität x, so ist der Server von Aktivität k durch diese Zuordnung stets optimal.
4. $ServZuordn(k) = f(Server(x))$ oder $ServZuordn(k) = f(Domain(Bearb(x)))$
   Auf die Serverzuordnungen kann noch eine Funktion f angewandt werden. Dies macht z.B. Sinn, wenn Aktivität k der Chef des Bearbeiters von Aktivität x zugeordnet ist. Dieser kann einem anderen Domain angehören (z.B. wenn er einer anderen Abteilung zugeordnet ist), so daß f eine Abbildung vom Domain des Mitarbeiters zu dem des Chefs darstellt. Wie

eine optimale Funktion f automatisch erzeugt werden kann, ist in [BD98b] beschrieben.

5. $ServZuordn(k)$ = beliebiger vorgegebener Ausdruck
   Wird vom Modellierer eine Serverzuordnung vorgegeben, die vom WfMS nicht analysiert werden kann, so muß er auch die für weitere Analysen benötigten WVen (siehe Abschnitt 4.4) vorgeben.

4.3 Bestimmung der optimalen Serverzuordnungen

Zum Ermitteln der optimalen Verteilung könnten im Prinzip nun für alle Aktivitäten und alle möglichen Serverzuordnungen die Kosten analysiert und die optimale Variante ausgewählt werden. Da ein WF aber durchaus $\#Akt > 100$ Aktivitäten umfassen kann, die jeweils alle ihre Vorgänger in der Serverzuordnung referenzieren können, scheidet diese Vorgehensweise wegen ihrer Komplexität $O(\#Akt^{\#Akt})$ im allgemeinen aus. Wir schlagen deshalb den nachfolgend beschriebenen Algorithmus 1 vor, der ein polynomiales Laufzeitverhalten hat. Er basiert auf einem Greedy-Ansatz und liefert für die praktisch relevanten Fälle das optimale Ergebnis bzw. eines, das dicht an diesem Optimum liegt.

Algorithmus 1 berechnet für jede Aktivität die optimale Serverzuordnung und legt diese im Array $ServZuordn$ ab. In der Phase 1 wird zunächst eine zulässige Ausganglösung bestimmt. Diese berücksichtigt noch keine Migrationskosten. Für jede Aktivität werden alle möglichen Serverzuordnungen ausprobiert. $PotServZuordn(k)$ liefert die entsprechende Menge für Aktivität k. Dies sind alle denkbaren Ausdrücke vom Typ 1 - 4 aus Abschnitt 4.2. Für Serverzuordnungen vom Typ 1 wird die Menge der Server-IDs $\{S_1, S_2 \ldots\}$ geliefert und für Typ 2 - 4 Ausdrücke der Form (z.B. Typ 2): $\{Server(x_1), Server(x_2) \ldots\}$, wobei die x_i die Vorgängeraktivitäten von k sind. Aus den potentiell möglichen Serverzuordnungen wird diejenige ausgewählt, die zu den minimalen Kosten für die Ausführung dieser Aktivität führt (*calculate_costs* verwendet hierzu das Kostenmodell aus Abschnitt 4.1). Da die Aktivitäten isoliert betrachtet werden, ergeben sich i.d.R. auch nicht rentable Migrationen. In Phase 2 wird überprüft, welche dieser Migrationen sinnvoll sind. Dazu wird ausgerechnet, ob es günstiger ist, eine Gruppe von Aktivitäten mit einer direkten Vorgänger- oder Nachfolgergruppe zusammenzufassen, so daß die Migration dazwischen entfällt. Die Motivation dafür ist, daß es sich nicht lohnt, wegen wenigen Aktivitäten die gesamte Prozeßinstanz zu einem Server hin und zurück zu migrieren. Der Algorithmus betrachtet zuerst kleine zusammenhängende Teilgraphen G von Aktivitäten, die alle vom selben Server kontrolliert werden. Diese werden mit Nachbar-Teilgraphen zusammengefaßt, wodurch immer größere Gruppen von Aktivitäten gebildet werden, zwischen denen keinen Migrationen stattfinden. Dies wird solange durch-

geführt, bis keine unrentablen Migrationen mehr vorhanden sind. Dieses Vorgehen reduziert die zu kommunizierende Datenmenge bei der Ausführung der WFs, da Gruppen von Aktivitäten genau dann zusammengefaßt werden, wenn dies die Kommunikationskosten reduziert.

Algorithmus 1: Berechnung der Serverzuordnung für einen WF-Typ (Prozeßvorlage)
Phase 1:
for each Aktivität $k \in$ *Prozeßvorlage* (in partieller Ordnung entsprechend Kontrollfluß) **do**
  $Cost_{min} = \infty$;
  for each $ServZuordn(k) \in PotServZuordn(k)$ **do**
    $Cost = calculate\_costs(ServZuordn, k)$; /* Kosten nur für Aktivität k berechnen */
    if $Cost < Cost_{min}$ **then** $ServZuordn_{opt}(k) = ServZuordn(k)$; $Cost_{min} = Cost$;
  $ServZuordn(k) = ServZuordn_{opt}(k)$;
Phase 2:
$Cost_{min} = calculate\_costs(ServZuordn, all)$; /* Kosten gesamter WF (mit Migrationskosten) */
for $i = 1$ **to** *#Aktivitäten(Prozeßvorlage)* **do**
  for each $G : |G| = i$, G ist max. Teilgraph mit $\forall l_1, l_2 \in G : Server(l_1) = Server(l_2)$ **do**
    for each $a \notin G : \exists\, l \in G : a =$ *Vorgänger(l)* $\vee$ $a =$ *Nachfolger(l)* **do**
      for each $l \notin G$ **do** $ServZuord_{neu}(l) = ServZuordn(l)$;
      for each $l \in G$ **do** $ServZuord_{neu}(l) = ServZuordn(a)$;
      $Cost_{neu} = calculate\_costs(ServZuordn_{neu}, all)$;
      if $Cost_{neu} < Cost_{min}$ **then**
        for each $l \in G$ **do** $ServZuordn(l) = ServZuordn_{neu}(l)$;
        $Cost_{min} = Cost_{neu}$;

4.4 Berechnung der Wahrscheinlichkeitsverteilungen

Nachdem erläutert wurde, welche Serverzuordnungen für eine Aktivität in Frage kommen und wie die durch sie entstehenden Kosten berechnet werden, bleibt die Frage, wie die WVen $prob_k(i,j)$ und $prob'_{k,l}(i,j)$ für eine zu untersuchende Verteilung berechnet werden können.

Da Serverzuordnungen von Laufzeitdaten der Instanz abhängen, kann dieselbe Aktivität k bei verschiedenen WF-Instanzen durch unterschiedliche Server kontrolliert werden. Die entsprechende Wahrscheinlichkeit, daß Aktivität k vom Server S kontrolliert wird, wird mit $\boldsymbol{P_k^S}$ bezeichnet. Da sich verschiedene Instanzen in unterschiedlichen OEen befinden können (und dabei unterschiedliche Server verwenden können), kann die Bearbeiter-WV jeweils unterschiedlich sein. Der Anteil an den Bearbeitern von Aktivität k aus dem Domain D, für den Fall, daß der Server S den Prozeßschritt k kontrolliert, wird in $\boldsymbol{U_k^{S,D}}$ festgehalten. Das Hauptproblem bei der Kostenanalyse besteht nun darin, für eine gegebene Verteilung die WVen P_k^S und $U_k^{S,D}$ zu berechnen. Sind diese erst bekannt, so können $prob_k(i,j)$ und damit die Kosten für die Aktivitätenausführung berechnet werden. Die Wahrscheinlichkeit, daß der Server S verwendet wird und ein Bearbeiter im Domain D die Aktivität k bearbeitet, beträgt $prob_k(S,D) = P_k^S \cdot U_k^{S,D}$. Tabelle 1 faßt die verwendeten WVen noch einmal zusammen; wie

Name	Bedeutung
$prob_k(i,j)$	Wahrscheinlichkeit, daß Aktivität k vom Server des Teilnetzes i kontrolliert und von einem Bearbeiter in Teilnetz j ausgeführt wird
$prob'_{k,l}(i,j)$	Wahrscheinlichkeit, daß beim Übergang von Aktivität k nach l vom Teilnetz i zum Teilnetz j migriert werden muß
P_k^S	Wahrscheinlichkeit, daß Aktivität k vom Server S (in Teilnetz S) kontrolliert wird
$U_k^{S,D}$	Anteil der Bearbeiter im Domain (Teilnetz) D an der Gesamtheit der Bearbeiter von Aktivität k, wenn Aktivität k vom Server S kontrolliert wird
$ServVert_k^S(U)$	Wahrscheinlichkeit, daß Aktivität k vom Server S kontrolliert wird, wobei der Bearbeiter U dieser Aktivität vorgegeben ist

Tabelle 1: Übersicht über die verwendeten Wahrscheinlichkeitsverteilungen.

$prob'_{k,l}(i,j)$ berechnet wird, wird in Abschnitt 4.5 erläutert.

4.4.1 Berechnung der Server-Wahrscheinlichkeitsverteilung P_k^S einer Aktivität

Die Server-WV P_k^S für die Aktivität k gibt an, mit welcher Wahrscheinlichkeit Server S der Server von Aktivität k ist. Sie ergibt sich aus der Serverzuordnung $ServZuordn(k)$ und der Server- bzw. Bearbeiter-WV der referenzierten Aktivität x. Diese WVen sind bei der Analyse von Aktivität k schon bekannt, da x vor k ausgeführt wird und die Prozeßvorlage in partieller Ordnung durchlaufen wird. Die Berechnung der Server-WV wird nun für die in Abschnitt 4.2 definierten Serverzuordnungen beschrieben. Der Fall 5 wird dabei (und im weiteren) nicht berücksichtigt, da bei dieser Serverzuordnung die WV vom Modellierer vorgegeben werden muß. P_k^S kann mit dem folgenden Algorithmus berechnet werden:

Algorithmus 2: Berechnung der Server-Wahrscheinlichkeitsverteilung P_k^S
case $ServZuordn(k) = S_i$: $P_k^S = \delta_{SS_i}$ [6]
/* Da die Aktivität k stets vom Server S_i kontrolliert wird, ist $P_k^S = 1$ für $S = S_i$ und sonst $= 0$. */
case $ServZuordn(k) = Server(x)$: $P_k^S = P_x^S$
/* Für Akt. k wird derselbe Server verwendet wie für x, weshalb die Server-WVen gleich sind.[7] */
case $ServZuordn(k) = Domain(Bearb(x))$: $P_k^i = U_x^i$ mit $U_x^D = \sum_S U_x^{S,D} \cdot P_x^S$
/* Der Server von Aktivität k ist im Domain des Bearbeiter von Aktivität x angesiedelt. Deshalb ergibt sich die Server-WV von k aus der Bearbeiter-WV von x. Da dabei nicht relevant ist, welcher Server Aktivität x kontrolliert hat, wird eine vom Server unabhängige Bearbeiter-WV U_x^i erzeugt, indem die Bearbeiter-WVen der einzelnen Server gewichtet aufaddiert werden. */
case $ServZuordn(k) = f(Server(x))$: $P_k^S = f(P_x^S)$
/* Die Berechnung von P_k^S erfolgt wie bei $ServZuordn(k) = Server(x)$, wobei auf das Ergebnis noch die Funktion f angewandt werden muß (für Details siehe [BD98b]). */

[6] Kroneckersymbol: $\delta_{ij} = 1$, falls $i = j$ und $\delta_{ij} = 0$, falls $i \neq j$.

[7] Dieser Aussage liegt die Annahme zugrunde, daß es zwischen x und k keine Verzweigungen gibt, die abhängig von der Wahl der Servers sind. Da solche Zusammenhänge kaum zu fassen sind, da sie Datenelemente betreffen, haben wir für Verzweigungs- und Schleifenbedingungen generell angenommen, daß ihr Ergebnis nicht vom aktuellen Server abhängt, außer der Modellierer gibt dies explizit vor.

case $ServZuordn(k) = f(Domain(Bearb(x)))$: $\quad P_k^S = f(U_x^i) \quad$ mit $U_x^D = \sum_S U_x^{S,D} \cdot P_x^S$
/* wie Fall $ServZuordn(k) = Domain(Bearb(x))$, dann auf das Ergebnis f anwenden */

4.4.2 Berechnung der Bearbeiter-Wahrscheinlichkeitsverteilung $U_k^{S,D}$

Die WV der Bearbeiter $U_k^{S,D}$ gibt an, mit welcher Wahrscheinlichkeit der Bearbeiter von Aktivität k aus dem Domain D stammt, wenn sie vom Server S kontrolliert wird. Sie ist vom konkret verwendeten Server S dieser Aktivität abhängig. Als Beispiel stelle man sich ein Krankenhaus mit 3 Abteilungen vor, die jeweils über einen eigenen Server verfügen. Für Patienten der Abteilung 1 sind ausschließlich Pflegekräfte von Abteilung 1 (Domain 1) zuständig. Sinnvollerweise wird in diesem Fall auch Server 1 verwendet. Damit ergibt sich eine Bearbeiter-WV $U_k^{1,D} = (1,0,0)$. Analog ergibt sich für Server 2 die WV $U_k^{2,D} = (0,1,0)$ und $U_k^{3,D} = (0,0,1)$ für Server 3. Die Bearbeiter-WV $U_k^{S,D}$ kann durch den folgenden Algorithmus bestimmt werden ($ServVert(k,U,OE)$ liefert die Server-WV von Aktivität k unter der Voraussetzung, daß der Benutzer U aus der OE OE diese Aktivität bearbeitet, s.u.):

Algorithmus 3: Berechnung der Bearbeiter-Wahrscheinlichkeitsverteilung $U_k^{S,D}$
$Bearb_k = \{U_k \mid U_k$ qualifiziert sich als Bearbeiter von Aktivität $k\}$;
for each $U_k \in Bearb_k$ **do**
  $D = Domain(U_k)$; $\quad OE = OE(U_k)$;
  $ServVert_k^S(U_k) = ServVert(k, U_k, OE)$;
  for each S **do** $U_k^{S,D} = U_k^{S,D} + ServVert_k^S(U_k)$;
normalisiere $U_k^{S,D}$ zeilenweise, so daß $\forall S : \sum_D U_k^{S,D} = 1$;

Der Algorithmus durchläuft alle Bearbeiter der Aktivität k und ermittelt jeweils den zugehörigen Domain D (aus dem Organisationsmodell). Außerdem bestimmt er den Server S, der die Aktivität kontrolliert, falls dieser Bearbeiter sie ausführt. Dann wird der Bearbeiter für den entsprechenden Server S und Domain D in der Bearbeiter-WV $U_k^{S,D}$ berücksichtigt. Die Berechnung von $U_k^{S,D}$ erfolgt also entsprechend der Definition der Bearbeiter-WV (siehe Abschnitt 4.4). Die Aktivität k kann trotz desselben Bearbeiters durch unterschiedliche Server – jeweils mit einer gewissen Wahrscheinlichkeit – kontrolliert werden. Diese Wahrscheinlichkeiten werden berechnet (s.u.) und im Vektor $ServVert_k^S(U)$ gespeichert. Der Bearbeiter wird anteilig bei jedem dieser Server berücksichtigt. Ein Beispiel hierfür ist eine Aktivität, deren Serverwahl von einer anderen Aktivität abhängt (also mehrere Server möglich sind). Wenn diese Aktivität immer vom selben Bearbeiter erledigt wird, muß dieser anteilsmäßig bei all diesen Servern berücksichtigt werden.

Die Berechnung der Server-WV $ServVert_k^S(U)$ für einen bestimmten Bearbeiter geschieht in ähnlicher Weise, wie für die bearbeiterunabhängige WV P_k^S in Abschnitt 4.4.1 beschrieben. Allerdings hängt $ServVert_k^S(U)$ nicht nur von der Serverzuordnung der betrachteten Aktivität

k ab, sondern auch von deren Bearbeiterzuordnung. Im folgenden werden einige Beispiele beschrieben, eine vollständige Diskussion aller Fälle, die durch Kombination der möglichen Server- und Bearbeiterzuordnungen entstehen, findet sich in [BD98b].

- Ist die Serverzuordnung statisch ($ServZuordn(k) = S_i$), so ist die Berechnung trivial, da der Server immer gleich ist: $ServVert_k^S(U) = \delta_{SS_i}$.
- Ist die Bearbeiterzuordnung unabhängig von anderen Aktivitäten (z.B. $Rolle = Arzt$), so ist die Server-WV unabhängig vom Bearbeiter U, weshalb die allgemeine Server-WV übernommen werden kann: $ServVert_k^S(U) = P_k^S$.
- In Abb. 3a wird Aktivität k vom selben Arzt bearbeitet wie Aktivität x. Der Server wird im Domain des Bearbeiters von Aktivität x allokiert. Es soll $ServVert_k^S(U)$ für $U =$ Dr. Brinkmann aus Domain 3 berechnet werden. Wegen $BearbZuordn(k) = Bearb(x)$ hat Dr. Brinkmann auch Aktivität x ausgeführt. Deshalb ist der Server von Aktivität k im Domain von Dr. Brinkmann angesiedelt. Da dies der Domain 3 ist, ergibt sich $ServVert_k^S(U) = (0, 0, 1)$.
- Wird, wie in Abb. 3b dargestellt, nicht derselbe Bearbeiter verwendet, sondern lediglich dieselbe OE, so ist das Verfahren etwas komplizierter. Es soll $ServVert_k^S(U)$ für Schwester Hildegard aus der OE Abteilung 2 berechnet werden. Dazu müssen alle Bearbeiter (derselben OE Abteilung 2) mit der Rolle Arzt durchlaufen werden, weil dies genau die Bearbeiter sind, die die Aktivität x ausgeführt haben können, wenn Aktivität k von Schwester Hildegard ausgeführt wird. Für jeden dieser Ärzte wird der Domain ermittelt, da er – falls dieser Arzt die Aktivität x ausgeführt hat – die Lokation des Servers bestimmt. Dieser Domain wird dann in $ServVert_k^S(U)$ berücksichtigt. Nach dem Durchlaufen aller potentieller Bearbeiter wird $ServVert_k^S(U)$ noch so normalisiert, daß $\sum_S ServVert_k^S(U) = 1$ gilt.

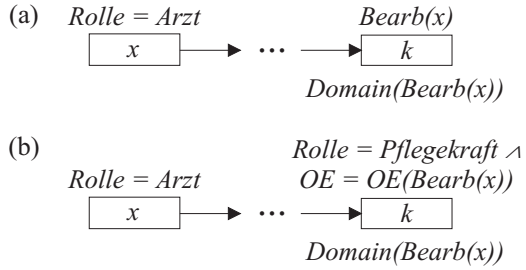

Abbildung 3 Beispiele zur Berechnung der Server-Wahrscheinlichkeitsverteilung $ServVert_k^S(U)$.

Zusätzlich zu diesen Aspekten ist zu beachten, daß es Benutzer gibt, die Teilzeit arbeiten, oder die nur einen Teil eines Arbeitstags mit dem WfMS arbeiten, oder Benutzer die in mehreren Domains jeweils einen Teil ihrer Arbeitszeit verbringen. Diese dürfen nicht wie Benutzer behandelt werden, die „full-time“ im selben Domain arbeiten, da sie weniger Last erzeugen. Ein solcher Sachverhalt läßt sich z.B. dadurch modellieren, daß jedem Benutzer U

des WfMSs ein Gewicht $Gewicht(U)$ zugeordnet wird. Dieses wird in Algorithmus 3 verwendet, um $ServVert_k^S(U)$ gewichtet zu addieren. Weitere Aspekte von Bearbeiter-Gewichtungen finden sich in [BD98b].

4.5 Migrationskosten

In Phase 1 von Algorithmus 1 werden nur die optimalen Serverzuordnungen der Aktivitäten betrachtet, ohne die Migrationskosten zu berücksichtigen. In Phase 2 werden auch diese Kosten mit einbezogen. Um sie berechnen zu können, muß die Wahrscheinlichkeit $prob'_{k,l}(i,j)$ bestimmt werden, mit der der WF vom Server i zum Server j migriert wird.

Die Wahrscheinlichkeiten für die Migrationen beim Übergang von Aktivität x nach k können in Form einer *Migrationsmatrix* angegeben werden. Der Eintrag der i-ten Zeile und j-ten Spalte gibt an, wie groß die bedingte Wahrscheinlichkeit ist, daß zum Server j migriert werden muß, wenn Aktivität x vom Server i kontrolliert wird. Es gilt also:
$M_{i,j}(x,k) = \mathrm{P}(\text{Server } j \text{ kontrolliert Aktivität } k \mid \text{Server } i \text{ kontrolliert Aktivität } x)$
Damit ergibt sich die Migrationswahrscheinlichkeit als:
$prob'_{x,k}(i,j) = P_k^i \cdot M_{i,j}(x,k)$
Im folgenden wird beschrieben, wie diese Matrix M ermittelt wird.

Aus den Serverzuordnungen kann man ableiten, welche Mengen von Aktivitäten einer WF-Instanz stets vom selben Server kontrolliert werden. Ist dies für die Aktivitäten x und k der Fall, so muß nie migriert werden, so daß sich die 1-Matrix ergibt. Andernfalls bietet die Nutzung der Server-WV P_k^S eine einfache Möglichkeit zur Approximation der Migrationswahrscheinlichkeiten. Die Server-WV gibt an, mit welcher Wahrscheinlichkeit sich die Instanz nach der Migration an Server j befindet. Die Migrationswahrscheinlichkeiten ergeben sich damit als:
$\forall i,j : M_{i,j}(x,k) = P_k^j$
Bei dieser Berechnung wird allerdings die Annahme vorausgesetzt, daß die Server der beiden Aktivitäten unabhängig voneinander gewählt werden. Diese Annahme ist aber häufig unrealistisch, da in vielen Fällen die Server- und Bearbeiterzuordnungen der beiden Aktivitäten jeweils von derselben OE abhängen werden. Dies kann wie folgt erkannt und behandelt werden:

Mit Hilfe der Bearbeiterzuordnungen werden Klassen von Aktivitäten gebildet, deren Bearbeiter stets derselben OE angehören (*OE-Klassen*). Dazu werden (einmal) alle Aktivitäten k der Prozeßvorlage durchlaufen. Aktivität k gehört derselben Klasse wie die von ihr referenzierte Aktivität x an, falls eine der folgenden Bearbeiterzuordnungen verwendet wird:

$BearbZuordn(k) = Bearb(x)$

$BearbZuordn(k) = OE(Bearb(x)) \wedge \ldots$

Zur Berechnung der Migrationswahrscheinlichkeiten wird Algorithmus 4 verwendet, wenn die Aktivität k und ihr Vorgänger x in derselben OE-Klasse liegen und in $ServZuordn(k)$ eine Aktivität dieser Klasse referenziert wird.

Algorithmus 4: Berechnung der Migrationswahrscheinlichkeiten (keine Unabhängigkeit)
for each S_x, S_k **do** $M_{S_x,S_k}(x,k) = 0$;
$Bearb_x = \{U_x \mid U_x$ qualifiziert sich als Bearbeiter von Aktivität $x\}$;
$Gesamtgewicht_x = \sum_{U_x \in Bearb_x} Gewicht(U_x)$;
for each $U_x \in Bearb_x$ **do**
  $ServVert_x^S(U_x) = ServVert(x, U_x, OE(U_x))$;
  $Bearb_k(U_x) = \{U_k \mid U_k$ qualifiziert sich als Bearbeiter von Aktivität k, wenn U_x Bearbeiter von $x\}$;
  $Gesamtgewicht_k(U_x) = \sum_{U_k \in Bearb_k(U_x)} Gewicht(U_k)$;
  for each $U_k \in Bearb_k(U_x)$ **do**
    $ServVert_k^S(U_k) = ServVert(k, U_k, OE(U_k))$;
    for each S_x, S_k **do**
      $M_{S_x,S_k}(x,k) = M_{S_x,S_k}(x,k) + \frac{Gewicht(U_x)}{Gesamtgewicht_x} \cdot \frac{Gewicht(U_k)}{Gesamtgewicht_k(U_x)} \cdot ServVert_x^{S_x}(U_x) \cdot ServVert_k^{S_k}(U_k)$;
normalisiere $M_{i,j}(x,k)$ zeilenweise, so daß $\forall i : \sum_j M_{i,j}(x,k) = 1$;

Algorithmus 4 berechnet die tatsächlichen Migrationswahrscheinlichkeiten, da er alle aufgrund von $BearbZuordn(k)$ erlaubten Kombinationen von Bearbeitern für die Aktivitäten x und k durchläuft, die Wahrscheinlichkeit ihres Auftretens berechnet und die bei diesem Paar auftretende Migration in M berücksichtigt. Die Wahrscheinlichkeit, daß Benutzer U_x die Aktivität x bearbeitet, beträgt $\frac{Gewicht(U_x)}{Gesamtgewicht_x}$. Die bedingte Wahrscheinlichkeit, daß Benutzer U_k unter dieser Voraussetzung Aktivität k bearbeitet, beträgt $\frac{Gewicht(U_k)}{Gesamtgewicht_k(U_x)}$. Mit diesen Wahrscheinlichkeiten gewichtet geht die Migration in M ein. Dabei werden (in den for-Schleifen) alle Fälle berücksichtigt, d.h. die Summe der Wahrscheinlichkeiten ergibt 1:

$$\sum_{U_x \in Bearb_x} \sum_{U_k \in Bearb_k(U_x)} \left(\frac{Gewicht(U_x)}{Gesamtgewicht_x} \cdot \frac{Gewicht(U_k)}{Gesamtgewicht_k(U_x)} \right)$$

$$= \sum_{U_x \in Bearb_x} \left(\frac{Gewicht(U_x)}{Gesamtgewicht_x} \cdot \sum_{U_k \in Bearb_k(U_x)} \frac{Gewicht(U_k)}{Gesamtgewicht_k(U_x)} \right)$$

$$\frac{1}{Gesamtgewicht_x} \cdot \underbrace{\sum_{U_x \in Bearb_x} \left(Gewicht(U_x) \right)}_{Gesamtgewicht_x} \cdot \frac{1}{Gesamtgewicht_k(U_x)} \cdot \underbrace{\sum_{U_k \in Bearb_k(U_x)} Gewicht(U_k) \Bigg)}_{Gesamtgewicht_k(U_x)} = 1$$

Außerdem gilt $\sum_{U_k \in Bearb_k(U_x)} \frac{Gewicht(U_k)}{Gesamtgewicht_k(U_x)} = 1$, so daß die Gewichtung des Bearbeiters U_x von Aktivität x nicht durch die Multiplikation mit $\frac{Gewicht(U_k)}{Gesamtgewicht_k(U_x)}$ beeinträchtigt wird. Ergibt die Berechnung von $ServVert_x^S(U_x)$ bzw. $ServVert_k^S(U_k)$ für das betrachtete Bearbeiterpaar U_x und U_k, daß mehrere Server betroffen sind, so werden die Bearbeiter anteilsmäßig bei jedem dieser Server berücksichtigt. Dadurch ergibt sich der Faktor $ServVert_x^{S_x}(U_x) \cdot ServVert_k^{S_k}(U_k)$. Wegen $\sum_{S_x} ServVert_x^{S_x}(U_x) = \sum_{S_k} ServVert_k^{S_k}(U_k) = 1$

gilt auch $\sum_{S_x} \sum_{S_k} ServVert_x^{S_x}(U_x) \cdot ServVert_k^{S_k}(U_k) = 1$, was beweist, daß das Gewicht des Bearbeiterpaares durch diesen zusätzlichen Faktor nicht beeinflußt wird.

Der mit dem soeben beschriebenen Algorithmus behandelte Fall ist in der Praxis sehr relevant, da WFs häufig für die Dauer mehrerer Aktivitäten innerhalb derselben OE verbleiben. Außerdem bilden die Benutzer einer OE oft exakt einen Domain. Werden in dem in Abb. 2 dargestellten Beispiel die Serverzuordnungen $ServZuordn(k) = Domain(Bearb(x))$ und $ServZuordn(k') = Domain(Bearb(k))$ verwendet, so kann aus diesen nicht geschlossen werden, daß für die beiden Aktivitäten stets derselbe Server verwendet wird. Werden zwei Server jeweils mit der Wahrscheinlichkeit 0,5 verwendet, so ergibt sich durch die einfache Approximation vom M durch P_k^S eine Migrationswahrscheinlichkeit von 50%. In Wirklichkeit tritt aber keine Migration auf. Der oben beschriebene Algorithmus berücksichtigt dies, da bei allen Kombinationen von Bearbeitern für die Aktivitäten k und k' jeweils derselbe Server (nämlich der in der OE dieser Bearbeiter) ermittelt wird, womit sich die 1-Matrix als Ergebnis M ergibt. Befindet sich lediglich der Großteil der Benutzer einer OE im selben Domain, so ergibt sich annähernd die 1-Matrix. In beiden Fällen wird erkannt, daß die Migrationskosten wesentlich niedriger sind, als wenn Unabhängigkeit angenommen worden wäre.

5 Diskussion

Dieser Abschnitt bietet einen Überblick über Architekturkonzepte für skalierbare WfMSe und stellt einige konkrete Ansätze vor. Für eine detaillierte Diskussion möchten wir auf [BD98a] verweisen. Das eine Extrem für die Architektur eines WfMSs ist ein *zentraler Server*. Diesem sind alle Aktivitäten zugeordnet. Da er deshalb die gesamte Systemlast bewältigen muß, stellt er einen potentiellen Flaschenhals dar. Einige Forschungsprototypen, die sich nicht primär um Skalierbarkeit kümmern (z.B. Panta Rhei [EG96]), und die meisten kommerziellen Systeme fallen in diese Kategorie. Das andere Extrem sind Systeme, die überhaupt keine Server verwenden. Die WFs migrieren zwischen den Benutzern, die sie bearbeiten. Auch hier sind keine Serverzuordnungen notwendig. Beispiele sind Exotica/FMQM [AMG+95] und INCAS [BMR94].

Zwischen diesen beiden Extremen liegen Ansätze, bei denen *mehrere WF-Server* verwendet werden, aber nicht jede Benutzermaschine gleichzeitig als Server fungiert. Da bei diesen Systemen eine Strategie für die Zuordnung der Server zu den Aktivitäten notwendig ist, werden sie nach diesem Kriterium klassifiziert. In [AKA+94] werden identische Replikate (Cluster) der WF-Engine verwendet. Eine WF-Instanz wird komplett von einem *zufällig ausgewählten Cluster* kontrolliert. Deshalb sind keine Serverzuordnungen notwendig. In [SM96] werden

die Systeme CodAlf und BPAFrame beschrieben, die den WF-Server einer Aktivität jeweils *bei der Anwendung* (z.B. beim DBMS) allokieren. Stehen mehrere Anwendungsserver zur Verfügung, so sorgt ein Trader für eine Verteilung der Last. Ein ähnlicher Ansatz wird von METEOR$_2$ [DKM+97, MSKW96, SK97] verfolgt. Die Anwendungslogik wird zu sog. Taskmanagern übersetzt. Für jede Aktivität entsteht so ein Taskmanager, der bei der zu dieser Aktivität gehörenden Anwendung plaziert wird. Generelle Nachteile dieses Vorgehens sind, daß zwischen dem WF-Server und den Bearbeitern u.U. eine weit entfernte Kommunikation notwendig wird und nicht jede Anwendung einen vorgegebenen Ort hat (z.B. ein Editor).

Die meisten Ansätze versuchen, wie $ADEPT_{distribution}$, die *Server nahe bei den Bearbeitern* der jeweiligen Aktivität zu plazieren. Bei MENTOR [WWWK96a, WWWK96b, WWK+97, MWW+98] werden State-/Activitycharts so partitioniert, daß sich die Äquivalenz der verteilten Ausführung zum zentralen Fall formal zeigen läßt. Da alle Bearbeiter einer Aktivität demselben „Processing Entity“ angehören müssen, bietet es sich an, diesen Server auszuwählen. Dieselbe Verteilungsstrategie wird von WIDE [CGP+96, CGS97] verwendet, wobei – anstelle einer Migration – entfernte Objektzugriffe mittels CORBA durchgeführt werden. In MOBILE [BHJ+96, HS96] werden die verschiedenen Aspekte eines WFs von verschiedenen Servern behandelt, mit dem Ziel deren Last zu reduzieren. Verschiedene WF-Typen können von verschiedenen Servern kontrolliert werden, es finden aber keine Migrationen statt.

$ADEPT_{distribution}$ bezieht in den Verteilungsalgorithmus die Kommunikationskosten mit ein und berücksichtigt damit, daß das Kommunikationssystem zum Flaschenhals werden kann. Es ist unseres Wissens der einzige Ansatz, bei dem eine Lastanalyse durchführt wird, um durch eine geeignete Verteilung die Kommunikationskosten zu minimieren. Bei den anderen Systemen gibt es keine variable Serverzuordnung, der Einfluß abhängiger Bearbeiterzuordnungen auf die Verteilung wird nicht berücksichtigt. Es gibt aber Systeme, die bei der Serverzuordnung flexibel sind [SM96], um eine Lastbalancierung zu erreichen oder den Ausfall von Komponenten zu kompensieren. Dies ist in $ADEPT_{distribution}$ ebenfalls vorgesehen, ist aber nicht Thema dieses Beitrags.

6 Zusammenfassung

WfMSe mit expliziter Prozeß- und Datenmodellierung und der dynamischen Zuordnung von Bearbeitern zu den auszuführenden Tätigkeiten sind eine vielversprechende Technologie für die Realisierung unternehmensweiter, vorgangsorientierter Anwendungssysteme. Die mit diesen Systemen erreichbare Flexibilität schlägt sich jedoch in einem relativ hohen Kommunikationsaufkommen zwischen den Steuerungskomponenten, den WF-Servern, und den An-

wendungskomponenten, den WF-Klienten, nieder. In unternehmensweiten WfMS-basierten Anwendungssystemen, mit hunderten von WF-Klienten und tausenden von aktiven WF-Instanzen spielt die daraus resultierende Belastung der WF-Server, aber auch der zugrundeliegenden (Teil-)Netze eine wesentliche Rolle für das Antwortzeitverhalten und damit für die Einsetzbarkeit überhaupt.

Eine Möglichkeit, um die Netzbelastung zu reduzieren, ist, die Kommunikation zwischen WF-Servern und ihren WF-Klienten jeweils möglichst lokal innerhalb eines Teilnetzes zu halten, d.h. teilnetzübergreifende Kommunikation nach Möglichkeit zu vermeiden. Dies ist dadurch zu erreichen, daß eine WF-Instanz nicht mehr komplett von dem initiierenden WF-Server gesteuert wird, sondern daß die Steuerung während der Ausführung ggf. auf andere WF-Server „migriert". In [BD97] haben wir eine Modellierungskomponente und deren formale Grundlagen (vor allem die Bestimmung von Wahrscheinlichkeitsverteilungen und Kostenformeln) beschrieben, die bei der Modellierung von WFs die Bestimmung geeigneter „WF-Abschnitte" erlaubt, die dann jeweils von einem anderen WF-Server gesteuert werden. Hierbei wurden die „Abschnittsserver" zur Modellierungszeit ermittelt und festgelegt. Dies führt dann zu guten Ergebnissen, wenn die möglichen Bearbeiter einer Tätigkeit z.B. allein durch ihre Rolle oder ihre Organisationseinheit beschrieben sind.

In diesem Beitrag haben wir untersucht, wie auch komplizierte Bearbeiterzuordnungen durch ein WfMS adäquat unterstützt werden können. Im Mittelpunkt standen hierbei sog. „abhängige" Bearbeiterzuordnungen, bei denen die in Frage kommenden Bearbeiter oder die Organisationseinheiten eines Arbeitsschrittes von in vorangegangenen getroffenen Zuordnungen abhängen; ein praktisch sehr relevanter Fall. Wir haben gezeigt, wie sich zum einen zur Modellierungszeit geeignete Abschätzungen durchführen lassen, und wie zum anderen die WF-Steuerung so realisiert werden kann, daß zur Laufzeit bei Bedarf dynamisch entschieden werden kann, welcher WF-Server für die Steuerung ausgewählt werden soll. Hierzu haben wir sog. Serverzuordnungsausdrücke sowie entsprechende Modelle für die Wahrscheinlichkeits- und Kostenabschätzungen eingeführt.

Literatur

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

[AMG+95] G. Alonso, C. Mohan, R. Günthör, D. Agrawal, A. El Abbadi und 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.

[BD97] T. Bauer und P. Dadam: *A Distributed Execution Environment for Large-Scale Workflow Management Systems with Subnets and Server Migration.* In: *Second IFCIS Conference on Cooperative Information Systems*, S. 99–108, Kiawah Island, SC, Juni 1997 (http://www.informatik.uni-ulm.de/dbis/papers/).

[BD98a] T. Bauer und P. Dadam: *Architekturen für skalierbare Workflow-Management-Systeme – Klassifikation und Analyse.* Technischer Bericht 98-02, Universität Ulm, Fakultät für Informatik, Januar 1998 (http://www.informatik.uni-ulm.de/dbis/papers/).

[BD98b] T. Bauer und P. Dadam: *Variable Migration von Workflows und komplexe Bearbeiterzuordnungen in ADEPT.* Technischer Bericht, Universität Ulm, Fakultät für Informatik, 1998 (erscheint demnächst).

[BHJ+96] C. Bußler, P. Heinl, S. Jablonski, H. Schuster und K. Stein: *Architektur von unternehmensweit einsetzbaren Workflow-Management–Systemen.* In: *Proceedings of MobIS 96, Rundbrief des GI-Fachausschusses 5.2*, S. 73–77, Oktober 1996.

[BMR94] D. Barbará, S. Mehrotra und M. Rusinkiewicz: *INCAS: A Computational Model for Dynamic Workflows in Autonomous Distributed Environments.* Technischer Bericht, Matsushita Information Technology Laboratory, Princeton, NJ, Mai 1994.

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

[CGS97] S. Ceri, P. Grefen und 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 und D. Worah: *ORBWork: A Reliable Distributed CORBA-based Workflow Enactment System for METEOR$_2$.* Technischer Bericht #UGA-CS-TR-97-001, Department of Computer Science, University of Georgia, Februar 1997.

[DKR+95] P. Dadam, K. Kuhn, M. Reichert, T. Beuter und M. Nathe: *ADEPT: Ein integrierender Ansatz zur Entwicklung flexibler, zuverlässiger kooperierender Assistenzsysteme in klinischen Anwendungsumgebungen.* In: *Proceedings GI/SI-Jahrestagung*, S. 677–686, Zürich, September 1995.

[EG96] J. Eder und H. Groiss: *Ein Workflow-Managementsystem auf der Basis aktiver Datenbanken.* In: J. Becker, G. Vossen (Herausgeber): *Geschäftsprozeßmodellierung und Workflow-Management.* International Thomson Publishing, 1996.

[HS96] P. Heinl und 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, Zürich, September 1996.

[LR97] F. Leymann und D. Roller: *Workflow-based Applications.* IBM Systems Journal, 36(1):102–123, 1997.

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

[MWW+98] P. Muth, D. Wodtke, J. Weißenfels, A. Kotz-Dittrich und G. Weikum: *From Centralized Workflow Specification to Distributed Workflow Execution.* Journal of Intelligent Information Systems, Special Issue on Workflow Management Systems, 10(2):159–184, März/April 1998.

[RD98] M. Reichert und P. Dadam: *$ADEPT_{flex}$ – Supporting Dynamic Changes of Workflows Without Loosing Control.* Journal of Intelligent Information Systems, Special Issue on Workflow Management Systems, 10(2):93–129, März/April 1998.

[SK97] A. Sheth und 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.

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

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

[WWWK96a] J. Weißenfels, D. Wodtke, G. Weikum und 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.

[WWWK96b] D. Wodtke, J. Weißenfels, G. Weikum und 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.