Proc. KnowTech 2000, Leipzig, September 2000

ADEPT – Realisierung flexibler und zuverlässiger unternehmensweiter Workflow-Anwendungen

Manfred Reichert, Thomas Bauer, Peter Dadam

Universität Ulm
Fakultät für Informatik, Abt. Datenbanken und Informationssysteme (DBIS)
E-Mail: {reichert, bauer, dadam}@informatik.uni-ulm.de, URL: www.informatik.uni-ulm.de/dbis

Zusammenfassung *Die Unterstützung unternehmensweiter und -übergreifender Geschäftsprozesse stellt für Workflow-Management-Systeme (WfMS) eine besondere Herausforderung dar: Es sind sehr viele Organisationseinheiten (auch externe) involviert, die Prozesse können langlaufend sein (Wochen, Monate), sie müssen rasch an neue Gegebenheiten anpassbar sein (Prozess-Redesign) und bei Bedarf muss im Einzelfall auch ad hoc vom geplanten Ablauf abgewichen werden können (z.B. durch Auslassen, Einfügen oder Verschieben von Prozessschritten). Die resultierenden Anwendungen müssen - auch im Fall von Ad-hoc-Abweichungen - für EDV-Laien einfach bedienbar sein, die WfMS-basierten Anwendungen müssen robust und stabil laufen (auch im Kontext von dynamischen Ablaufänderungen) und das WfMS muss auch bei einer großen Anzahl von Benutzern und Prozessinstanzen performant sein. Heutige WfMS erlauben bislang so gut wie keine Abweichung vom vormodellierten Ablauf. Prozesse der oben beschriebenen Art können deshalb als eine „Killer-Anwendung" für diese Systeme betrachtet werden. Zukünftige WfMS werden jedoch, sofern sie ein solches Anwendungsszenario adäquat unterstützen können, in vielen anspruchsvollen Anwendungsdomänen einsetzbar sein. Im Rahmen des ADEPT-Projektes arbeiten wir seit 1994 intensiv an den technologischen Grundlagen und der Entwicklung eines WfMS der nächsten Generation, das alle diese Aspekte (und noch viele mehr) ganzheitlich und sehr grundlegend adressiert. Der realisierte ADEPT-WfMS-Prototyp weist die Implementierbarkeit und das Zusammenspiel der entwickelten Konzepte auf eindrucksvolle Weise nach und zeigt, dass Flexibilität, Robustheit und Effizienz keine Widersprüche sein müssen. Der vorliegende Beitrag erläutert die zugrundeliegende Problemstellung, die technologischen Herausforderungen sowie die Einsatzperspektiven für ein solches System.*

1 Einleitung

Durch das rasante Wachstum des elektronischen Handels im Internet („E-Business"), durch die zunehmende Globalisierung der Märkte, durch neue Kooperationsformen zwischen Unternehmen („Virtual Enterprises") und vieles mehr, gewinnt die Unterstützung unternehmensweiter und -übergreifender Geschäftsprozesse immer mehr an Bedeutung [1]. Heutige betriebliche Informationssysteme bieten jedoch keine ausreichenden Möglichkeiten für die computerbasierte Steuerung und Überwachung von Geschäftsprozessen [2]. Die elektronische Informationsverarbeitung erfolgt daten- und funktionsbezogen und ist nicht für eine vorgangsorientierte, bereichsübergreifende Koordination von Tätigkeiten ausgelegt. In den einzelnen Bereichen gibt es Anwendungssysteme zur Unterstützung spezieller Funktionen, wie die Erfassung eines Kundenauftrags, das Erstellen eines Angebots oder das Schreiben einer Rechnung. Die logische Verknüpfung dieser Funktionen, d.h. der dahinterstehende Ablauf (engl. Workflow, WF) existiert dagegen meist nur in den Köpfen der prozessbeteiligten Personen oder in Form von Organisationsanweisungen, falls solche vorhanden sind. Dies führt in der Praxis häufig dazu, dass zu erledigende Aufgaben übersehen oder Abhängigkeiten zwischen ihnen (Reihenfolgen, zeitliche Einschränkungen) nicht beachtet werden. Die Folgen sind Unterlassungsfehler, falsche Entscheidungen, unnötige Aufgabenwiederholungen bzw. -verzögerungen sowie lange Leerlauf-, Transport- und Liegezeiten [2, 3].

Vorgangsorientierte Anwendungssysteme, die die Durchführung von Geschäftsprozessen aktiv koordinieren und überwachen, die Mitarbeiter ablaufbezogen informieren und die Prozesse lückenlos do-

kumentieren, die die Ressourcen- und Terminplanung unterstützen und die auf drohende Terminverletzungen oder unterlassene Tätigkeiten hinweisen, können hier eine signifikante Verbesserung der Prozesse bewirken [2]. Prozessorientierte WfMS, wie MQSeries Workflow oder Staffware, bieten einen vielversprechenden Ansatz zur Realisierung solcher Anwendungen [4, 5]. Sie zerlegen das Anwendungssystem in einen Ablauf- und einen Funktionsteil – genauer eine Sammlung von Funktionsbausteinen. Zu diesem Zweck muss für jeden zu unterstützenden Ablauftyp eine sog. *Prozessvorlage* modelliert und im System hinterlegt werden. Sie definiert, welche Tätigkeiten (sog. *Aktivitäten*) von welchen Personen (*Rollen*) in welcher Reihenfolge und unter welchen Bedingungen rechnergestützt durchgeführt werden sollen. Für die Vorlagendefinition bedient man sich grafischer Beschreibungssprachen. Neben den Prozessen müssen auch die Organisationsstrukturen des Unternehmens und Bearbeiterrollen - soweit sie für die Ablaufsteuerung relevant sind - modelliert werden.

Um die Bearbeitung von Aktivitäten rechnergestützt bewerkstelligen zu können, müssen ihnen Funktions- bzw. Programmbausteine zugeordnet werden. Diese werden, zumindest aus logischer Sicht, separat entwickelt und können in unterschiedlichen Prozessvorlagen wiederverwendet werden [4]. Dabei kann es sich um einfache Bildschirmmasken handeln, über die Informationen ein- und ausgegeben werden, es können aber auch (existierende) Anwendungskomponenten sein, die an dieser Stelle eine bestimmte Anwendungsfunktion (z.B. „Auftrag erfassen“, „Rechnung erstellen“) realisieren [6]. Um zur Laufzeit einen neuen Vorgang zu erzeugen, muss ein berechtigter Akteur zuerst eine Prozessvorlage aus dem Repository auswählen. Nach der Festlegung vorgangsspezifischer Daten wird dann vom WfMS eine neue *WF-Instanz* generiert und gestartet, die dann über ihre komplette Lebensdauer hinweg vom System begleitet wird. Die Laufzeitkomponenten steuern und überwachen die Ausführung der WF-Instanzen, verwalten instanzspezifische Daten, bieten die zur Ausführung anstehenden manuellen Arbeitsschritte den zuständigen Akteuren in Arbeitslisten an, führen automatische Schritte selbständig durch und rufen die für die Ausführung einer Aktivität passenden Funktionsbausteine auf. Insgesamt bietet die beschriebene Trennung von Prozesslogik und Anwendungscode die Möglichkeit, die Anwendungsentwicklung durch die Bereitstellung geeigneter Systemdienste drastisch zu vereinfachen, etwa in Bezug auf die Ablaufsteuerung, die Arbeitslistenverwaltung oder die Fehler- und Ausnahmebehandlung. Insbesondere wird der Anwendungsentwickler von systemnahen Aspekten, wie Rechnerkommunikation, Logging oder Recovery entlastet, wodurch sich die Zuverlässigkeit des resultierenden Anwendungssystems erhöht.

Bei der Realisierung prozessorientierter Anwendungen ist zu beachten, dass sie häufiger abgeändert werden müssen als funktionsorientierte Anwendungssysteme, die tendenziell relativ stabil sind [7, 8, 9]. Ein WfMS ist deshalb nur dann auf breiter Basis sinnvoll einsetzbar, wenn sich die von ihm unterstützten Anwendungen rasch an geänderte Ablauf- und Organisationsstrukturen anpassen lassen. Dies ist bei heutigen WfMS entweder gar nicht oder nur mit Einschränkungen möglich, so dass sie nur für einige wenige Anwendungsbereiche einsetzbar sind [8]. Ein weiteres Manko heutiger WfMS besteht darin, dass sie eine starre Ausführung des vormodellierten WF erzwingen, d.h. von der einmal definierten Prozessvorlage kann zur Ausführungszeit nicht oder nur unter erheblichen Einschränkungen abgewichen werden [9]. Ein durch den Rechner erzwungenes starres schematisches Vorgehen bei der täglichen Arbeit stößt allerdings in vielen Bereichen auf wenig Akzeptanz, so dass die Einsatzmöglichkeiten der Systeme eingeschränkt sind [3, 10]. Stattdessen sollte es für berechtigte Akteure jederzeit möglich sein, in die Kontrolle laufender WF-Instanzen einzugreifen und sie bei Bedarf spontan an geänderte Realweltsituationen anzupassen, etwa durch das Auslassen, Einfügen oder Verschieben von Prozessschritten [11]. Im allgemeinen werden solche Adaptionen immer dann notwendig, wenn während der Ausführung eines Workflows neue Informationen oder Arbeitsschritte eingebracht oder bereits definierte Eigenschaften, z.B. Bearbeitungsreihenfolgen von Aktivitäten, nachträglich geändert werden sollen. Für diesen Fall muss die Bearbeitung der abzuändernden WF-Instanz teilweise unterbrochen, die gewünschte Modifikation ihres Ausführungsgraphen, ihrer Attribute oder ihres Status vorgenommen und anschließend mit der Instanzausführung fortgefahren werden. Allerdings dürfen solche Ad-hoc-Änderungen in der Folge niemals zu Konsistenzproblemen (z.B. Lost Updates) oder einem instabilen Systemverhalten (z.B. Aufruf von Anwendungsfunktionen mit fehlenden Parameterdaten) führen. Dies ist bei existierenden WfMS nicht gewährleistet (vgl. [11]).

Der Abschnitt 2 erläutert wichtige technologische Herausforderungen für WfMS. In Abschnitt 3 zeigen wir, wie diese im ADEPT-Projekt aufgegriffen werden und welche Konzepte hieraus hervorgegangen sind. Der Beitrag schließt mit einer kurzen Zusammenfassung.

2 Technologische Herausforderungen

Vorgangsorientierte Informationssysteme müssen in der Lage sein, ein breites Spektrum an Arbeitsprozessen flexibel und effizient zu unterstützen. Hieraus ergeben sich für das zugrundeliegende WfMS die folgenden technologischen Herausforderungen (vgl. [3]):

1. Das WfMS muss einen ausdrucksstarken Formalismus für die WF-Modellierung (sog. *WF-Metamodell*) bereitstellen, der für Entwerfer verständlich ist und der die konsistente Beschreibung aller relevanten Prozessaspekte (Kontroll- und Datenfluss, zeitliche Beschränkungen, organisatorische Aspekte, usw.) gestattet.
2. Das WfMS muss ein zuverlässiges und robustes Ausführungsverhalten besitzen. Aus diesem Grund sollte bereits zur Modellierungszeit ausgeschlossen werden, dass es zur Laufzeit infolge einer fehlerhaften WF-Modellierung zu ungewollten „Überraschungen“ kommt (z.B. Blockierungen bei der WF-Ausführung, nicht erfüllbare zeitliche Einschränkungen, Aufrufe von Funktionsbausteinen bei fehlenden oder unvollständigen Parameterdaten).
3. Das WfMS muss temporale Aspekte unterstützen. Neben Deadlines für Aktivitäten müssen zeitliche Abhängigkeiten zwischen ihnen (z.B. minimale oder maximale Zeitabstände) verwaltet werden. Die Einhaltung dieser Zeitrestriktionen muss systemseitig überwacht werden. Treten Probleme auf (z.B. drohende Terminverletzungen) sind die Benutzer darüber zu informieren .
4. Im Einzelfall muss für WF-Instanzen auch ad hoc vom geplanten Ablauf, d.h. vom modellierten WF, abgewichen werden können, etwa durch das Auslassen, Einfügen oder Verschieben von Prozessschritten. Solche dynamischen WF-Änderungen dürfen nicht zu Konsistenz- oder Korrektheitsverletzungen führen. Das bedeutet, dass WfMS-basierte Anwendungen auch im Anschluss an dynamische Änderungen robust und stabil laufen müssen. Die hierzu notwendigen On-the-fly-Modellanalysen müssen effizient durchführbar sein, was mit zunehmender Ausdrucksmächtigkeit des verwendeten WF-Metamodells jedoch immer schwieriger wird.
5. Die resultierenden Anwendungen müssen - auch im Fall von Ad-hoc-Abweichungen - für EDV-Laien einfach bedienbar sein. Insbesondere sollte die mit der Festlegung einer Ad-hoc-Änderung verknüpfte Komplexität vor Anwendern verborgen bleiben, etwa in Bezug auf das Re-Mapping der Ein- und Ausgabeparameter der von der Änderung betroffenen Funktionsbausteine oder die Behandlung fehlender Daten nach dem Löschen von Prozessschritten. Ebensowenig sollte sich der Benutzer um die Anpassung von WF-Zuständen kümmern müssen. Stattdessen muss das WfMS geeignete Modifikationsoperatoren auf einem hohen Abstraktionsniveau anbieten (kein Pop-Up eines Modell-Editors).
6. Von einem WfMS werden häufig Daten verarbeitet, an die hinsichtlich Datenschutz und Datensicherheit strenge Maßstäbe angelegt werden müssen. Die Rechte einer Person für den Zugriff auf diese Daten sind dabei über die Rolle oder Funktion geregelt, welche sie gerade einnimmt. Bereits im statischen Fall sind die realen Rollenverteilungen und Vertretungsregelungen häufig sehr komplex und stellen hohe Anforderungen an das WfMS, insbesondere im Hinblick auf die Pflege von Organisationsmodellen. Im dynamischen Fall kommt erschwerend hinzu, dass durch Änderungen des Ablaufgraphen keine "Datenschutzlücken" entstehen dürfen. Aus diesem Grund muss sehr fein steuerbar sein, wer in welcher Rolle und in welchen WF-Zuständen welche Änderungen durchführen bzw. welche Informationen (z.B. Ausführungshistorie, Änderunghistorie usw.) abrufen darf. Entsprechende Einschränkungen sind auch deshalb sinnvoll, weil einzelne Anwender oft nur eine eingeschränkte Sichtweise auf Abläufe besitzen. Dadurch können WF-Änderungen, die aus der Sicht des Einzelnen durchaus sinnvoll erscheinen, im Widerspruch zu übergeordneten Interessen (Einhaltung von Fristen, organisatorische Regelungen usw.) stehen. Diese Problematik ist bei unternehmensweiten oder –übergreifenden Workflows verstärkt ausgeprägt.
7. Um unternehmensweite und -übergreifende Workflows angemessen unterstützen zu können, muss das WfMS auch bei einer großen Anzahl von Benutzern und Prozessintanzen performant arbeiten.

Obwohl bereits sehr umfangreich, ist diese Liste noch nicht vollständig. Weitere wichtige Anforderungen betreffen Evolution von WF-Schemata und – soweit sinnvoll und möglich – die Propagierung dieser Änderungen auf bereits laufende WF-Instanzen, die Handhabung von Inter-Workflow-Abhängigkeiten, die komponentenbasierte Entwicklung von WF-Anwendungen, das semantische Rollback von Workflows beim Auftreten logischer Fehler oder den Umgang mit Medienbrüchen. Zukünftige WfMS werden, sofern sie diesen Anforderungen gerecht werden, in vielen anspruchsvollen Anwendungsdomänen einsetzbar sein. Im Rahmen des ADEPT-Projektes arbeiten wir seit 1994 intensiv an den technologischen Grundlagen und der Entwicklung eines WfMS der nächsten

Generation, das alle diese Aspekte (und noch einige mehr) ganzheitlich und sehr grundlegend adressiert.

3 Der ADEPT-Ansatz zur Realisierung prozessorientierter Informationssysteme

Den im vorangegangenen Abschnitt skizzierten Anforderungen kann man nicht dadurch gerecht werden, indem man sie isoliert voneinander betrachtet. Im ADEPT-Projekt versuchen wir deshalb die verschiedene Facetten prozessorientierter Informationssysteme integriert zu behandeln: Benutzerschnittstellen, Modellierungsfragestellungen und -werkzeuge, planbare und nicht planbare Ausnahmebehandlungen, Flexibilität und dynamische WF-Änderungen, WF-Schema-Evolution, temporale Aspekte, Inter-Workflow-Abhängigkeiten und Skalierbarkeit. Aus Platzgründen können wir an dieser Stelle nicht auf alle diese Aspekte eingehen. Stattdessen konzentrieren wir uns auf dynamische WF-Änderungen und Skalierbarkeitsfragestellungen. Weitergehende Informationen zum ADEPT-Projekt finden man in [9, 11, 12, 13, 14].

3.1 $ADEPT_{base}$ – Das ADEPT-Workflow-Metamodell

ADEPT stellt für den WF-Modellierer ein ausdrucksstarkes WF-Metamodell bereit, das es erlaubt, Arbeitsprozesse möglichst natürlich und in einer für die Anwender verständlichen Form abzubilden. Darüber hinaus ist die effiziente Überprüfung bzw. Sicherstellung wichtiger WF-Qualitätseigenschaften möglich, etwa im Hinblick auf die Aktivierbarkeit von WF-Aktivitäten, die Verklemmungsfreiheit des modellierten WF oder die Korrektheit der definierten Datenflüsse. ADEPT verwendet dazu das graphbasierte WF-Metamodell $ADEPT_{base}$, das die integrierte Beschreibung der verschiedenen Aspekte eines WF ermöglicht [2, 11]. Für die Kontrollflussmodellierung wird ein blockbasierter Beschreibungsansatz verfolgt, bei dem Sequenzen, Verzweigungen und Schleifen als logische Blöcke mit jeweils genau einem Ein- und Ausgangsknoten modelliert werden. Solche Kontrollblöcke können geschachtelt sein, dürfen sich aber nicht überlappen. Um die Ausdrucksmächtigkeit des Metamodells zu erhöhen, werden zusätzliche Konstrukte angeboten, etwa zur Beschreibung verschiedener Arten von „Wartet-Auf"-Beziehungen zwischen Aktivitäten paralleler Zweige oder zur Vormodellierung ausnahmebedingter Vorwärts- und Rückwärtssprünge im Kontrollfluss (z.B. partielles Rollback des WF). Desweiteren stellt $ADEPT_{base}$ auch Konstrukte zur Modellierung von Datenflüssen, zeitlichen Einschränkungen (z.B. Mindest- oder Maximalzeitabstände zwischen Aktivitäten) und organisatorischen Aspekten bereit. Ein Beispiel eines ADEPT-WF-Modells zeigt Abbildung 1.

Auf Ausführungsebene wird jede WF-Instanz um Zustandsinformationen angereichert, die vom WF-Server für die WF-Steuerung benötigt werden. Zur Unterstützung der Kontrollflusssteuerung werden den Knoten und Kanten des Ausführungsgraphen einer WF-Instanz entsprechende Zustandsmarkierungen zugeordnet. Für ihre Handhabung gibt es wohldefinierte Regeln, die festlegen, unter welchen Graphmarkierungen eine WF-Aktivität aktiviert werden darf und welche Folgemarkierungen sich im Anschluss an ihre Ausführung ergeben können [11]. Dabei bleiben die Zustandsmarkierungen bereits abgearbeiteter bzw. nicht mehr ausführbarer WF-Bereiche beim Fortschreiten des WF erhalten, zumindest solange die entsprechenden Graphregionen nicht wiederholt (z.B. nach Schleifenrücksprüngen) durchlaufen werden. Dadurch können Informationen zum bisherigen Verlauf der WF-Ausführung direkt aus den aktuellen Graphmarkierungen abgeleitet werden, was für die effiziente Überprüfung der Anwendbarkeit dynamischer Änderungsoperationen vorteilhaft ist [11].

3.2 $ADEPT_{flex}$ – Unterstützung dynamischer WF-Änderungen

Berechtigte Anwender können in ADEPT dynamische WF-Änderungen auf einem hohen Abstraktionsniveau festlegen. Die Grundlage hierfür bildet das $ADEPT_{flex}$-Kalkül, das einen vollständigen Satz an Modifikationsoperatoren anbietet, mit denen sich die Struktur, die Attribute oder der Status von sich in Ausführung befindlichen WF-Instanzen abändern lassen [11]. Unterstützt werden Operationen für das Hinzufügen, das Löschen oder das Verschieben einzelner Aktivitäten oder ganzer Kontrollblöcke, für die Abänderung von Datenflüssen, für die Modifikation einzelner Ausführungsattribute (z.B. Ressourcen- oder Bearbeiterzuordnungen, Verzweigungsprädikate, Zeitattribute) oder für das kontrollierte Zurücksetzen der WF-Bearbeitung. Durch ihre kombinierte Anwendung können auch semantisch höherwertige Änderungen realisiert werden, etwa für das dynamische Einfügen einer Aktivität zwischen zwei Knotenmengen oder für die Umsetzung von Ad-hoc-Vorwärtssprüngen (z.B. vorzeitige Bearbeitung einer Aktivität mit Nachholen der übersprungenen Schritte).

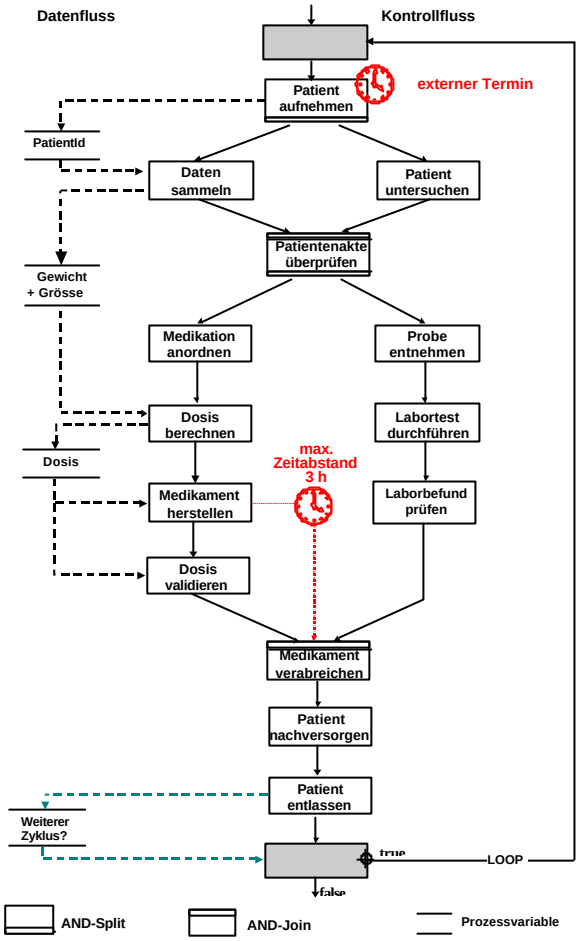

Abb. 1: *WF-Modellierung in ADEPT*

WF-Instanz-Änderungen sind in ADEPT nur zulässig, wenn sie die Korrektheit und Konsistenz des WF nicht verletzen. Zur Erfüllung dieser Forderung müssen alle WF-Aspekte und ihre Wechselwirkungen beachtet werden, selbst wenn sie nicht unmittelbar Gegenstand der beabsichtigten Modifikation sind. Bei Änderungen der Kontrollflussstruktur sorgt ADEPT z.B. selbständig dafür, dass die Blockstrukturierung aufrechterhalten bleibt und dass keine Verklemmungssituationen eintreten. Dies wird durch die Vorgabe geeigneter Vor- und Nachbedingungen für die Anwendung von Änderungsoperationen erreicht. Des weiteren wird bei Anwendung von Operationen wie dem Einfügen, Löschen oder Verschieben von Aktivitäten (inkl. ihrer Schreizugriffe auf Prozessvariablen) sichergestellt, dass im weiteren Verlauf der WF-Ausführung keine Inkonsistenzen (z.B. Lost Updates) oder Fehler (z.B. Aufruf von Aktivitätenprogrammen mit unvollständig versorgten Eingabeparametern) auftreten können. Sind z.B. obligate Eingabeparameter einer Aktivität nach dem Löschen einer Vorgängeraktivität nicht mehr versorgt, muss die Löschoperation entweder abgebrochen werden oder es sind begleitende Anpassungen vorzunehmen (z.B. kaskadierendes Löschen datenabhängiger Schritte, Installation von Nachforderungsdiensten, die bei der Aktivierung des datenabhängigen Schrittes gerufen werden, usw.) [9].

$ADEPT_{flex}$ sorgt auch selbständig dafür, dass nach der Anwendung einer dynamischen Änderung wieder ein WF mit konsistentem Zustand resultiert. Um dies sicherzustellen, schränken wir die Anwendbarkeit von Änderungsoperationen durch den aktuellen Zustand der jeweiligen WF-Instanz (d.h. den Knoten- und Kantenmarkierungen ihres Ausführungsgraphen) ein. Beispielswiese dürfen bereits abgeschlossene Aktivitäten nicht mehr gelöscht oder ihre Ausführungsattribute nachträglich verändert werden. Ebensowenig darf eine neue WF-Aktivität in einen bereits abgearbeiteten Teil eines Ausführungsgraphen eingefügt werden. Entsprechende Statusüberprüfungen werden *vor* Anwendung einer Änderungsoperation bzw. -transaktion durchgeführt. Darüber hinaus wird *nach* der strukturellen Abänderung eines Ausführungsgraphen sein Status neu bewertet und ggf. angepaßt, z.B. durch die Zuordnung von Markierungen zu neu eingefügten Knoten und Kanten. Anschließend kann mit der WF-Ausführung in einem konsistenten Zustand fortgefahren werden (vgl. [9]).

Das ADEPT-WfMS stellt ebenfalls sicher, dass bei der Anwendung einer Änderungstransaktion wieder ein WF mit konsistentem, d.h. erfüllbarem Zeitplan resultiert (siehe [2]). Die Einhaltung dieser Forderung wird nicht nur bei der Änderung von Zeitattributen geprüft, sondern auch in Verbindung mit Modifikationen der Kontrollflussstruktur. Beispielsweise können sich durch das Einfügen einer Aktivität (mit bekannter minimaler und maximaler Zeitdauer), die frühesten Anfangszeitpunkte nachfolgender Aktivitäten nach hinten verschieben. Je nachdem, ob dadurch zuvor festgelegte Termine noch erfüllbar sind oder nicht, können die notwendigen Anpassungen automatisch durch das WfMS erfolgen, oder sie müssen in Interaktion mit den Anwendern ermittelt werden.

3.3 ADEPT$_{distribution}$ – Skalierbarkeit durch verteilte WF-Steuerung

Unternehmensweite und –übergreifende WF-Anwendungen sind charakterisiert durch eine große Zahl von Benutzern und gleichzeitig aktiven WF-Instanzen [1]. Dadurch kann die Belastung für WF-Server sehr groß werden. Bereits die Durchführung einer WF-Aktivität macht die Übertragung mehrerer Nachrichten zwischen WF-Server und WF-Clients erforderlich, etwa um Parameterdaten zu transferieren, Arbeitslisten zu aktualisieren, Funktionsbausteine zu rufen oder Daten zwischen gerufenen Funktionen und externen Datenquellen auszutauschen. Insgesamt kann das zu bewältigende Kommunikationsaufkommen sehr groß werden, was bei zunehmender Anzahl von WF-Instanzen zur Überlastung von Teilnetzen und zu schlechten Antwortzeiten führt. Erschwerend kommt für unternehmensübergreifende Workflows hinzu, dass die Kommunikation teilweise über relativ langsame Weitverkehrsnetze abgewickelt werden muss. Spätestens dann stellt die Belastung des Kommunikationssystems einen kritischen Faktor für den Gesamtdurchsatz des Systems dar [15].

Um die im unternehmensweiten Einsatz aufkommende Gesamtlast bewältigen zu können, muss ein WfMS skalierbar sein. In ADEPT erzielen wir Skalierbarkeit, indem wir die Systemlast auf mehrere WF-Server verteilen. Zu diesem Zweck haben wir mit ADEPT$_{distribution}$ ein Ausführungsmodell entwickelt und implementiert, bei dem die Kontrolle einer WF-Instanz ggf. abschnittsweise durch verschiedene WF-Server erfolgen kann [12]. Zu diesem Zweck wird der Ablaufgraph einer Prozessvorlage in *Partitionen* unterteilt (vgl. Abb. 2). Jeder Partition wird ein WF-Server zugeordnet, der zur Laufzeit die Koordination ihrer Aktivitäten übernimmt. Kommt es während der Ausführung einer WF-Instanz zu einem Übergang zwischen zwei Partitionen, so findet eine *Migration* statt, bei der ihre Kontrolle vom aktuellen Server an den Server der Zielpartition übertragen wird. Bevor dieser Server mit der Ausführung fortfahren kann, müssen instanzspezifische Daten (z.B. Informationen zum aktuellen Ausführungsstatus, aktuelle Werte bestimmter Prozessvariablen) transferiert werden. Durch diesen Ansatz werden die Aktivitäten paralleler Bearbeitungszweige ggf. nebenläufig durch unterschiedliche WF-Server koordiniert. Um dabei die Kommunikation lokal zu halten, müssen diese Server in ADEPT nicht notwendigerweise Kenntnis über den Bearbeitungsstatus der von anderen Servern kontrollierten Teilzweige bzw. Aktivitäten besitzen.

Ein zentrales Anliegen des ADEPT$_{distribution}$-Ansatzes ist es, die Partitionierung von WF-Ablaufgraphen und die Zuordnung von WF-Servern so zu wählen, dass das Kommunikationsaufkommen bei der WF-Ausführung minimiert wird. Zu diesem Zweck werden für den WF-Entwickler Verfahren bereitgestellt, mit denen sich optimale Serverzuordnungen bzw. optimale Partitionierungen automatisch berechnen lassen. Dazu verwenden wir Kostenformeln, die es uns erlauben, die Güte einer gewählten Verteilung zu bewerten. Berücksichtigt werden u.a. Kosten für den Transfer von Parameterdaten, für die Aktualisierung von Arbeitslisten sowie für die Durchführung von Migrationen. Dabei wird eine Aktivität in den meisten Fällen demjenigen Server zugeordnet, der sich am nächsten bei den zugehörigen Bearbeitern befindet. Da Migrationen ebenfalls Kommunikationskosten verursachen und die Server belasten, werden sie allerdings nur eingesetzt, wenn sie das Gesamtkommunikationsverhalten verbessern.

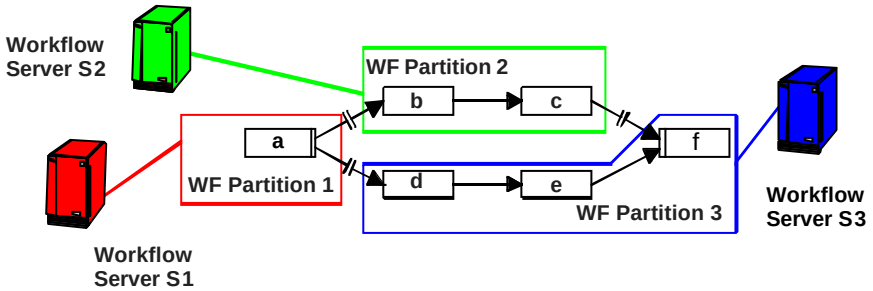

Abb. 3-2: *Partitionierung von Workflows und Migration der Workflow-Kontrolle*

4 Zusammenfassung

WF-Technologie besitzt das Potenzial, die Entwicklung vorgangsorientierter Anwendungen nachhaltig zu verändern. Faktisch wird durch ihren Einsatz die Realisierung und der Betrieb prozessorientierter Anwendungssysteme im größeren Stil überhaupt erst möglich. Dabei sollten, wie im Fall von ADEPT, die einzelnen Funktionsbausteine eines WF als isolierte, wiederverwendbare Komponenten implementiert werden können, deren Eingabeparameter beim Aufruf von der Laufzeitumgebung des WfMS versorgt werden und die lediglich dafür sorgen müssen, dass nach ihrer Beendigung korrekte Werte für Ausgabeparameter erzeugt werden. Alle anderen Aufgaben der Ablaufsteuerung und -überwachung (inkl. Ausnahme- und Fehlerbehandlungen) sollen durch das WfMS übernommen werden. WF-Technologie bietet die Chance, zu einer gänzlich neuen Art der Entwicklung verteilter Informationssysteme zu gelangen, bei der Anwendungen durch die (graphische) Beschreibung von Prozessvorlagen und durch das Einstecken vorgefertigter Programmbausteine in diese Vorlagen entwickelt werden. Spätere Prozessänderungen und daraus resultierende Anpassungen der Anwendungssysteme können bei diesem Ansatz relativ einfach durchgeführt werden. Wurde bei der Implementierung der Funktionsbausteine sorgfältig vorgegangen, kann z.B. die Reihenfolge der Arbeitsschritte eines WF geändert oder es können neue Schritte hinzugenommen werden, ohne dass hiervon die bereits existierenden Funktionsbausteine betroffen sind. Schließlich können WfMS dazu beitragen, funktionsorientierte Anwendungen prozessorientiert zu integrieren und so eine gemeinsame Basis für die Verwaltung von Arbeitsabläufen zu schaffen.

Heutige WF-Technologie ist noch zu limitiert, um ein breiteres Spektrum an Prozessen angemessen unterstützen zu können. Für die nahe Zukunft ist jedoch zu erwarten, dass fortschrittliche WfMS, wie das ADEPT-System, am Markt erscheinen werden, die eine mächtigere Plattform für die Implementierung prozessorientierter Informationssysteme bereitstellen werden. Der ADEPT-WfMS-Prototyp [14] ist derzeit eines der funktional mächtigsten und flexibelsten WfMS. Er weist die Implementierbarkeit und das Zusammenspiel der im ADEPT-Projekt entwickelten Konzepte auf eindrucksvolle Weise nach und zeigt, dass Flexibilität, Robustheit und Effizienz keine Widersprüche sein müssen. Der Prototyp zeigt aber auch, dass solche High-End-WfMS große Software-Systeme sind, die leicht die Code-Komplexität eines High-End-DBMS erreichen. Die derzeitige Implementierung des ADEPT-WfMS umfaßt ca. 120.000 Programmzeilen Java-Code.

Literatur

1. Dadam, P.; Reichert, M. (eds.): *Proc. Workshop on Enterprise-Wide and Cross-Enterprise Workflow-Management: Concepts, Systems, Applications*, 29. Jahrestagung der GI (Informatik'99), Paderborn, Oktober 1999.
2. Reichert, M.; Dadam, P.; Mangold, R.; Kreienberg, R.: *Computerbasierte Unterstützung von Arbeitsabläufen im Krankenhaus – Konzepte, Technologien und deren Anwendung*. Zentralbl Gynakol, Vol 122, Januar 2000, S. 56–70.
3. Dadam, P.; Reichert, M.; Kuhn, K.: *Clinical Workflows - The Killer Application for Process-oriented Information Systems*? Proc. 4th Int'l Conf. on Business Information Systems (BIS'2000), Posen, April 2000, S. 36–59.
4. Leymann, F.; Roller, D.: *Production Workflow – Concepts and Techniques*. Prentice Hall, 2000.
5. Jablonski, S.; Böhm, M.; Schulze, W. (Hrsg.): *Workflow-Management: Entwicklung von Anwendungen und Systemen*. Dpunkt, 1997
6. Reichert, M.; Dadam, P.: *Geschäftsprozeßmodellierung und Workflow-Management – Konzepte, Systeme und deren Anwendung. Industrie Management*, Vol. 16, No. 3, 2000, S. 23–27.
7. Hammer, M.; Stanton, S.: *The Reengineering Revolution*. Harper Collins Publ., 1995
8. Casati, F.; Ceri, S.; Pernici, B.; Pozzi, G.: *Workflow Evolution*. Data & Knowledge Engineering, Vol. 24, No. 3, Januar 1998, S. 211-238
9. Reichert, M.; Hensinger, C.; Dadam, P.: *Supporting Adaptive Workflows in Advanced Application Environments*. Proc. EDBT-Workshop on Workflow Management Systems, Valencia, 1998, S. 100-109
10. Weske, M.: *Flexible Modeling and Execution of Workflow Activities*. Proc. 31st Hawaii Int'l. Conf. System Sciences (HICSS-31), Software Technology Track (Vol. VII), 1998, S. 713-722

11. Reichert, M.; Dadam, P.: *ADEPT$_{flex}$ – Supporting Dynamic Changes of Workflows Without Losing Control*. Journal of Intelligent Information Systems, Special Issue on Workflow Management Systems, Vol. 10, No. 2, März 1998, S. 93–129
12. Bauer, T.; Dadam, P.: *Efficient Distributed Workflow Management Based on Variable Server Assignments*. Proc. 12tn Conf. on Advanced Information Systems Engineering, Stockholm, Juni 2000, S. 94-109.
13. Dadam, P.; Kuhn, K.; Reichert, M.; Beuter, T.; Nathe, M.: *ADEPT: Ein integrierender Ansatz zur Entwicklung flexibler, zuverlässiger kooperierender Assistenzsysteme in klinischen Anwendungsumgebungen*. Proc. 25. GI–Jahrestagung, Zürich, September 1995, S. 677–686
14. Hensinger, C.; Reichert, M.; Bauer, T.; Strzeletz, T.; Dadam, P.: *ADEPT$_{workflow}$ – Advanced Workflow Technology for Adaptive, Enterprise-wide Processes*. Demo-Proc. of the 7tn Int'l Conf. on Extending Database Technology (EDBT'2000), Konstanz, März 2000.
15. Bauer, T.; Dadam, P.: *Verteilungsmodelle für Workflow-Management-Systeme – Klassifikation und Simulation*, Informatik Forschung und Entwicklung, Vol. 14, No. 4, Dezember 1999, S. 203-217.