Faxantwort für Peter Strasser an

Manuskript **„Reengineering von Kernanwendungssystemen auf Großrechnern“** für die Zeitschrift Informatik-Spektrum

DOI 10.1007/s00287-003-0290-8

Korrespondenzautor: Wilhelm G. Spruth
Arbeitsbereich Technische Informatik, Institut für Informatik
Universität Tübingen
Sand 13
72076 Tübingen
E-mail: spruth@informatik.uni-tuebingen.de

weitere Autoren: J. Franz

Bitte überprüfen Sie die oben angegebenen Daten und korrigieren Sie sie gegebenenfalls.

☒ Bitte kreuzen Sie die gewünschten Optionen an und faxen Sie dieses Formular an die oben angegebene Nummer. Dies ist auch dann wichtig, wenn Sie keine Korrekturen haben. Vielen Dank. Die Erteilung der Rechtseinräumung ist für eine Veröffentlichung erforderlich!

I. Imprimatur

❑ Hiermit gebe ich mein Manuskript **direkt frei**. Korrekturen sind nicht notwendig.
Das Manuskript wird nicht zurückgefaxt.

❑ Ich habe **einzelne Seiten** des Manuskriptes korrigiert und diese dem Antwortfax beigelegt.
Nach ausgeführter Korrektur ist das Manuskript imprimiert.
Folgende Manuskript-Seiten habe ich beigelegt: _____

❑ Dieses Fax enthält das **komplette Manuskript** mit entsprechenden Korrekturwünschen. Nach ausgeführter Korrektur ist das Manuskript imprimiert. Mir ist bewusst, dass größere Korrekturen oder Änderungen an der Textlänge ein **späteres Publikationsdatum** nach sich ziehen können.

II. Sonderdrucke

❑ Ich wünsche kostenpflichtige Sonderdrucke meines veröffentlichten Manuskriptes.
Das ausgefüllte Formular „Sonderdruckbestellung“ liegt bei.

III. Rechtseinräumung

❑ In Erweiterung von § 38 Abs. 1 UrhG übertrage ich als erstgenannter Verfasser dem Springer-Verlag das ausschließliche Recht der Speicherung, Vervielfältigung, Verbreitung und Wiedergabe meines Beitrages – einschließlich des Rechts zur Übersetzung – für die Dauer des gesetzlichen Urheberrechts in gedruckter und elektronischer Form. Der Springer-Verlag ist berechtigt, die Nutzungsrechte am gesamten Beitrag oder in Teilen daraus in gedruckter oder elektronischer Form wahrzunehmen oder weiterzugeben.
Hiermit versichere ich – auch im Namen der Miturheber –, dass ich berechtigt bin, über die urheberrechtlichen Nutzungsrechte an dem oben genannten Beitrag zu verfügen.

Ort, Datum Unterschrift

Mit Erscheinen der gedruckten Ausgabe erscheint Ihr Beitrag in unserem Internetangebot. Sie finden ihn, wenn Sie Ihre Manuskriptnummer (DOI) s00287-003-0290-8 unter **http://link.springer.de/search.htm** eingeben.
Eingereichte Fotos oder Abbildungen werden nur auf ausdrücklichen Wunsch des Autors nach Veröffentlichung zurückgesandt.

Manuskript „Reengineering von Kernanwendungssystemen auf Großrechnern“ für die Zeitschrift Informatik-Spektrum
DOI: 10.1007/s00287-003-0290-8

Sonderdruckbestellung

Wenn Sie Sonderdrucke Ihres Artikels bestellen möchten, füllen Sie bitte dieses Formular aus und senden es zusammen mit Ihren Korrekturen an uns zurück. Sollten Sie erst nach dem Druck eines Heftes Sonderdrucke bestellen, beträgt die Mindestabnahme 300 Exemplare, da der Nachdruck erhöhten Aufwand bedeutet.

Zusätzlich zu den Sonderdrucken erhalten Sie per E-Mail eine PDF-Datei Ihres Beitrages zum persönlichen Gebrauch. Bitte kontrollieren und korrigieren Sie gegebenenfalls hier Ihre E-Mail-Adresse:

spruth@informatik.uni-tuebingen.de _____

Hiermit bestelle ich:

	Anzahl Sonderdrucke	**Preis in Euro inkl. Porto und Verpackung**
❑	50	350,00 €
❑	100	400,00 €
❑	200	550,00 €
❑	300	700,00 €
❑	400	850,00 €
❑	500	1.000,00 €
❑	über 500	Wir erstellen Ihnen ein individuelles Angebot.

Bitte geben Sie Ihre VAT-Nummer an:

_____

Kunden aus EU-Ländern ohne VAT-Nummer müssen den allgemein gültigen Steuerbetrag addieren.

Ich bezahle per:

❑ Kreditkarte
- ❑ Eurocard/Access/Mastercard
- ❑ American Express
- ❑ Visa/Barclaycard/BankAmericard

Kartennummer (inkl. Prüfziffern):

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Gültig bis: _ _ / _ _

❑ Rechnung

Rechnungsanschrift:

❑ siehe Lieferanschrift

❑ _____
_____
_____
_____
_____

Bitte verwenden Sie bei der Rechnungsstellung folgende Bestellnummer/folgendes Zeichen:

_____

Lieferung an:

❑ Wilhelm G. Spruth
Universität Tübingen
Institut für Informatik
Sand 13
72076 Tübingen
Germany

❑ _____
_____
_____
_____
_____

Datum / Unterschrift: _____

Bitte überprüfen Sie die oben angegebenen Daten und korrigieren Sie sie gegebenenfalls.
❑ Bitte kreuzen Sie die gewünschten Optionen an und faxen Sie dieses Formular an .

Informatik-Spektrum
© Springer-Verlag 2003
DOI 10.1007/s00287-003-0290-8

Hauptbeitrag

Reengineering von Kernanwendungssystemen auf Großrechnern

Prof. Dr.-Ing. Wilhelm G. Spruth (✉) · Joachim Franz

W. G. Spruth
Institut für Informatik, Universität Leipzig

W. G. Spruth
Arbeitsbereich Technische Informatik, Institut für Informatik, Universität Tübingen

J. Franz
IBM Business Consulting Services, Technology Strategy

W. G. Spruth
Arbeitsbereich Technische Informatik, Institut für Informatik, Universität Tübingen, Sand 13, 72076 Tübingen
✉ E-mail: spruth@informatik.uni-tuebingen.de

Zusammenfassung

Nach der Euro-Einführung und der Jahr-2000-Umstellung stellt der Umbau der heutigen Kernanwendungssysteme auf den Großrechnersystemen unter z/OS die nächste große Herausforderung dar, insbesondere bei Unternehmen im Financial-Services-Bereich (Banken und Versicherungen). Der vorliegende Beitrag führt in die Systemarchitektur einer „Transaktionsmaschine“ ein, die als Middlewarekomponente für das Reengineering von Kernanwendungssystemen verwendet werden kann. Die Transaktionsmaschine stellt einen neuartigen Ansatz dar, der geeignet ist, die Herausforderungen im zentralen Serverbereich der großen Unternehmen und staatlichen Organisationen zu adressieren. Der Fokus liegt hierbei auf dem Entwurf der abstrakten Systemarchitektur und deren Charakteristika, geprägt durch funktionale und nichtfunktionale Anforderungen. Als praktische Beispielszenarien werden Projekterfahrungen der Firma IBM im Umfeld von Abwicklungssystemen bei Transaktionsbanken aufgezeigt.

Schlüsselwörter

Kernanwendungssysteme · Reengineering · z/OS · Transaktion · STP

Summary

This Paper introduces a new System Architecture of a Transaction Engine which can be used as a middleware component to reengineer core application systems. Starting with the requirement analysis in an existing customer environment and mapping these to key design criteria's this paper leads to the design of an abstract System Architecture. At the end we show how this Transaction Engine could be used in a real customer environment as a platform for reengineering core application systems. We are using some practical example scenarios based on IBM's experience in the transaction banking area.

Keywords

Core Application Systems · Reengineeing · z/OS · Transaction · STP

Überblick

Moderne Client/Server-Systeme strukturieren ihre Anwendungen in die beiden Teile Geschäftslogik (Business-Logik, auch als Backend bezeichnet) und Präsentationslogik (auch als Frontend bezeichnet; s. Abb. 1).

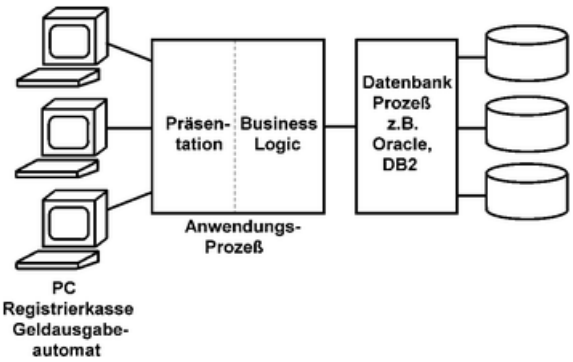

Abb. 1. Klassische Client/Server-Architektur mit der Aufteilung Frontend (Präsentation), Middle (Business) und Backend (Datenbank). Bei der 3-Tier-Anwendungsarchitektur wird der Anwendungsprozess über ein Model-Viewer-Controller-(MVC-)Prinzip abgebildet und die Datenbanken durch die Business-Logik eingebunden

---

Für die Präsentationslogik haben sich Java-Technologien, besonders Servlets, Applets und Java-Server-Pages sowie ihre Microsoft-Äquivalente als eine produktive Technologie weitgehend durchgesetzt.

Mit den EJBs und dem J2EE-Standard bemüht man sich, einen objektorientierten Ansatz auch für die Geschäftslogik einzusetzen. In der Tat sind eine Reihe von Großsystemanwendungen entstanden, bei denen heterogene Backendsysteme, z. B. OS/390, Solaris, openVMS, einschließlich ihrer unterschiedlichen Datenbanken, mit Hilfe von EJBs miteinander integriert wurden. Von dem vorhandenen Ziel, existierende Geschäftslogik durch neue EJB-Implementierungen zu ersetzen, sind wir jedoch noch sehr weit entfernt. Dies gilt besonders für die größeren Unternehmen, die in der Regel OS/390 mit IMS, CICS, VSAM, DB2, Oracle und Adabas als ihren zentralen Server einsetzen.

Vorhandene Java-Implementierungen von Internetanwendungen benutzen häufig Konnektoren, um mit Hilfe des Common-Connector-Frameworks oder der Java-Connector-Architektur (JCA) auf vorhandene Anwendungslogik zuzugreifen, die in der Form von Anwendungen unter CICS, IMS/DC, DB2 Stored Procedures oder Stapelverarbeitungsanwendungen auf dem Backend vorliegen.

Ein Reengineering dieser Anwendungen in Richtung Java hätte bei der Jahr-2000-Umstellung nahe gelegen; hiervon wurde aber nur in den seltensten Fällen Gebrauch gemacht. Es ist eher ein Trend zu erkennen, die heutigen Anwendungssysteme auf den Großrechnern in einer Übergangsphase auf eine neue zukünftige Anwendungsarchitektur durch „Wrapping“ zu isolieren und somit unabhängig von den Eingangskanälen (Front-Office) zu machen (Abb. 2). Dies kann jedoch nur ein Zwischenschritt sein, da dieser Ansatz die Kernanwendungen weitgehend unverändert lässt und die im Folgenden geschilderten Probleme nicht lösen kann.

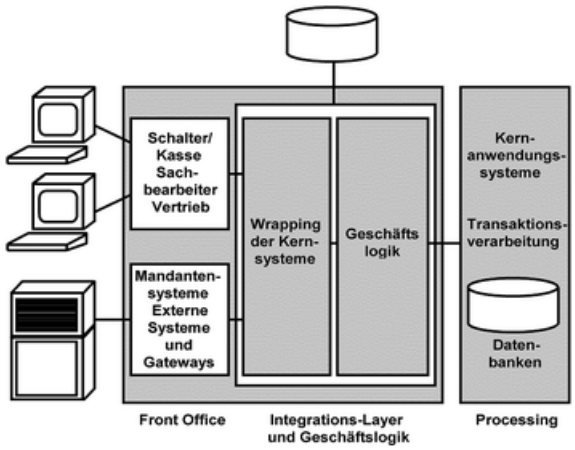

Abb. 2. Die Aufteilung der Anwendungsarchitektur in Front-Office (mit der Präsentationslogik), der Integrationsebene mit Teilen der Geschäftslogik und Processing mit der Transaktionsverarbeitung wird verwendet, um bestehende Kernanwendungssysteme durch den Integration-Layer zu wrappen

---

Es wird geschätzt, dass etwa 10 Millionen Mannjahre in die Entwicklung von unternehmenskritischen OS/390-Anwendungen unter CICS investiert wurden. Das bedeutet eine Investition von etwa einer Billion US-Dollar in OS/390-Anwendungssoftware unter CICS [3].

Die Wartung und ständige Anpassung an sich ändernde Unternehmensbelange stellen einen erheblichen Kostenfaktor für die Unternehmen dar. Dabei stellt sich heraus, dass Wartungskosten für Cobol-Programme deutlich niedriger liegen als für alle anderen Programme. Die Jahr-2000-Umstellungs-Kosten pro Function-Point betrugen im Durchschnitt für alle Sprachen 45 $; für Cobol-Programme lagen sie bei 28 $ [1]. Es werden deshalb auch sehr viele neue Anwendungen in Cobol geschrieben.

Die existierende Menge an Cobol-Programmen besteht aus etwa 180 Milliarden Codezeilen mit einer jährlichen Zuwachsrate von 5 Milliarden Codezeilen [2]. Nach Spector [3] sind derzeitig 20 Milliarden Zeilen CICS-Code in Benutzung.

Umfragen bei großen Unternehmen (typischerweise S/390 mit z/OS) zeigen, dass die nächsten großen Herausforderungen—im Zeitraum bis 2005—der Umbau und das Reengineering der heutigen Kernanwendungssysteme sein wird und muss. Hierbei ist nicht mehr an eine Ablösung der Großrechnersysteme und deren Anwendungen gedacht. Vielmehr soll durch ein Reengineering eine Ausrichtung auf das Kerngeschäft der Unternehmen erfolgen. Es steht hierbei eine engere Ausrichtung der IT auf die Geschäftsstrategie der Unternehmen im Vordergrund [4, 5]. Dies bedeutet im Gegensatz zum teilweisen Reengineering (process improvement) eine grundlegende Neukonzeption, von den Geschäftsprozessen bis hin zu den IT-Systemen der Unternehmen.

Durch die aktuell erreichte technologische Spitzenposition der z/Series-Server unter z/OS im Hinblick auf RAS (Reliability, Availability und Serviceability) und der hohen Verarbeitungsleistung (max. Durchsatz von Geschäftstransaktionen), eignen sich diese zentralen Server besonders gut als Zielplattform für weitere Serverkonsolidierungsaufgaben sowie als Basissystem für zukünftige Kernanwendungssysteme [6, 15].

Anforderungen

Im Rahmen mehrerer Projekte im Umfeld des Transaction-Banking (Abwicklungsbanken für Zahlungsverkehr und Wertpapierservice) zeigte sich ein Business-Process-getriebenes Vorgehen als ein möglicher Reengineering-Ansatz [7, 8]. Hierbei wird auf Basis der Neudefinition von Geschäftsprozessen für die Kernbereiche der Unternehmen eine STP-(Straight -Through-Processing-)Fähigkeit angestrebt. Finanzdienstleister definierten STP als einen vollautomatischen und standardisierten Geschäftsprozess ohne manuellen Eingriff in die Geschäftsverarbeitung vom Zeitpunkt des Abschlusses bis zum Zeitpunkt der vollständigen Abwicklung eines Geschäftes. Diese STP-fähigen Geschäftsprozesse dienen zusammen mit den funktionalen und nichtfunktionalen Anforderungen der Anwendungsentwicklung als fachliche Basis zur Ausrichtung der neuen Anwendungssysteme [9, 10]. Bei diesem Vorgehen geht man hierarchisch (top–down) vor, d. h. man gliedert definierte Hauptprozesse weiter in feinere Einheiten, die Teilprozesse. Ein Teilprozess wiederum besteht aus einem Netz von verbundenen Knoten, den Aktivitäten. Aktivitäten beschreiben die granularen fachlichen Tätigkeiten, die zur Erfüllung eines Geschäftes notwendig sind.

Wenn wir diesen fachlichen Entwurf der Geschäftsprozessdefinition nun auf den Anwendungsentwurf abbilden, ergibt sich grob eine Zuordnung des Prozessablaufes zu einer Steuerungslogik und eine Zuordnung der Aktivitäten zur Fachlogik (Abb. 3).

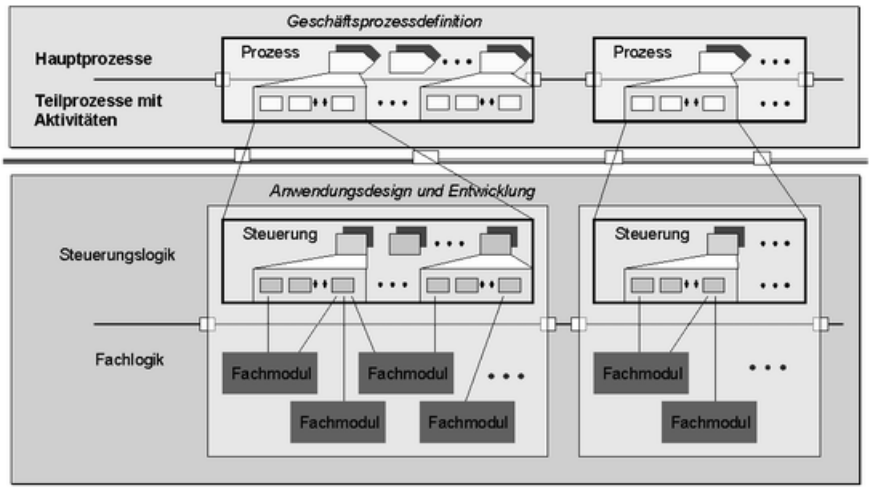

Abb. 3. Ausgehend von den STP-fähigen Geschäftsprozessen, wird das Anwendungsdesign und die Entwicklung durchgeführt

Eine zentrale Rolle spielt bei diesem Vorgehen die zukünftige Architektur der neuen Anwendungssysteme. Hierbei bildet man im Rahmen eines architekturzentrierten Vorgehens beim Entwurf verschiedene Architekturbereiche aus, welchen dann Architekturkomponenten (wie z. B. Geschäftsprozesse, Geschäftsobjekte, Fachkonzepte, Anwendungskomponenten, Middlewarekomponenten etc.) zugeordnet werden. In der Regel gliedert man diese Architekturbereiche in eine Facharchitektur (Geschäftsarchitektur), eine Anwendungsarchitektur und eine Systemarchitektur auf. Schwerpunkt der Facharchitektur ist hierbei eine Fachorientierung ohne einen direkten Bezug auf die IT (reine Geschäftssicht). Die Anwendungsarchitektur hat zur Aufgabe, die Anforderungen aus der Facharchitektur auf Anwendungssysteme umzusetzen, wobei eine Grenze zur Systemarchitektur auf der Ebene von Middlewarekomponenten zu sehen ist. Die Systemarchitektur enthält neben den Architekturkomponenten für die Middlewareebene auch Infrastruktur- und Systemkomponenten, die zum Betrieb der Anwendungssysteme notwendig sind.—Weitere Details bezüglich Architektur in der Anwendungsentwicklung wurden von Foegen [11] beschrieben.

Im weiteren Verlauf unseres Beitrags möchten wir konkret die mögliche Abbildung auf eine neuartige Systemarchitektur beschreiben, welcher eine zentrale Rolle in unserem gewählten Reengineeringansatz der Kernanwendungssysteme (KAS) zukommt. Kernanwendungssysteme sind Anwendungssysteme, die den Kern des Geschäftes bzw. der Geschäftsprozesse eines Unternehmens unterstützen. Sie sind aus diesem Grund die für die Geschäftsabwicklung wichtigsten Anwendungssysteme und für den operativen Erfolg eines Unternehmens absolut kritisch. Diese Kernanwendungssysteme sind bisher nahezu vollständig als Backend-Anwendungen (z. B. OS/390) in COBOL realisiert.
In mehreren großen Kundenprojekten (Anwendungssysteme >10 Mio. Zeilen Cobol-Code) haben sich eine Reihe von wesentlichen Anforderungen an eine zukünftige Anwendungs- und Systemarchitektur der neuen Kernanwendungssysteme herauskristallisiert. Diese wurden im Rahmen von Voruntersuchungen und Studien zur Neuausrichtung der Kernanwendungssysteme bei Kunden im Bankenumfeld belegt und sind in Anforderungskatalogen konsolidiert. Die folgende Liste zeigt exemplarisch einen konsolidierten Auszug:

- Mandantenfähigkeit (z. B. Mehrbankfähigkeit):
  Dies bedeutet im Wesentlichen die anwendungstechnische Unterstützung für eigenständige, juristisch unabhängige Unternehmen (z. B. eigenständige Bankinstitute mit eigenem

Kundennetz, Kundenstamm etc.). IT-Aspekte hierzu sind z. B. getrennte physische Datenhaltung; evtl. Verschlüsselung von Transaktionsdaten, falls Aufträge von verschiedenen Mandanten auf dem gleichem System verarbeitet werden; Ablaufverfolgung der Aufträge und Abrechnung der erbrachten Leistungen pro Mandant; etc. Der Grad der Mandantenunterstützung (strikte oder schwache Mandantentrennung) wird durch eine gemeinsame Vereinbarung (Vertrag) mit dem Serviceanbieter (z. B. einer Transaktionsbank) geregelt. Diese Vereinbarung wird auch als „Service Level Agreement" (SLA) bezeichnet und enthält noch weitere durch den Mandanten bestimmte „Quality-of-Service"-Parameter. Anhand der festgelegten SLA-Parameter bestimmt sich zum einen der Service (QoS), der für den Mandanten erbracht wird, zum anderen der Preisrahmen des übergebenen Geschäftsauftrags, der zur Abwicklung des Geschäftes berechnet wird.

- Mandanten und Kundeninformationen über ein Auftragsmanagement:
  Dies erlaubt eine Sicht auf aktuelle Aufträge mit Status, Verarbeitungskosten und berechneter Fertigstellung. Die Mandaten und evtl. deren Kunden sollen jederzeit in der Lage sein, übergebene (Geschäfts-)Aufträge online nachverfolgen zu können und aktuelle Statusinformationen über den Stand der Bearbeitung, die aktuellen Kosten und das voraussichtliche Ende der Auftragsbearbeitung zu erhalten. Auch ein nachträgliches Beschleunigen oder Rückstellen („Umpriorisierung") der Abarbeitung soll im Rahmen der vereinbarten Service Level Agreements „on the flight" möglich sein. Storno und Abbruch des Auftrages kann auch durch den Mandanten jederzeit durchgeführt werden.
- Möglichkeit der Integration und Anbindung externer und interner Softwarekomponenten:
  Hierunter zählen neben der Einbindung bzw. Integration von „zugekauften" Softwarekomponenten auch die Anbindung von nachgelagerten Systeme wie z. B. Dokumentenarchivierung oder Finanzbuchhaltung. Auch externe Zuliefersysteme wie Börsensysteme, Mandantensysteme, Systeme der Bundesbank etc. müssen im Rahmen des Prozessablaufs angebunden werden.
- Hohe Flexibilität in der Einführung und Änderung neuer Produkte für die Kernanwendungssysteme mit mandantenspezifischen Verarbeitungspfaden:
  Dies bedeutet, dass bei der Einführung und der Änderung von neuen Produkten des Serviceanbieters (z. B. Abwicklung von neuen Finanzprodukten im Rahmen der Riester-Rente oder Anpassung von Produkten an Mandantenwünsche, europäische bzw. internationale Regeln) die Abbildung auf das Kernanwendungssystem so gestaltet sein

soll, dass nur geänderte Abläufe zu einem bestehenden Basisprodukt eingebracht werden müssen, ansonsten die bestehenden Prozessabläufe aber mitverwendet werden können. Dies gilt auch bei Derivaten von Produkten mit geringer Abweichung im Prozessablauf. Die Einbringung und Änderung von Produkten soll am laufenden Kernanwendungssystem möglich sein.

- Reduktion der Kosten pro Transaktion in der Verarbeitung und Wartung:
  Dieser Punkt spiegelt eins der Hauptprobleme der heutigen „organisch gewachsenen" Kernanwendungssysteme wieder. Der Transaktionsdurchsatz soll (auch bei hohem Volumen >1 Mio./Trans. pro Stunde) in Bezug auf benötigte Ressourcen nahezu linear skalieren, wobei der Durchsatz für das Standardgeschäft (Basisprodukt) durch das System autonom optimiert werden soll. Dies führt zu einer Minimierung der Transaktionskosten durch Erreichen des Skalierungseffektes (Economy of Scales).
- Enge Integration der Kernanwendungssysteme in das Systemsmanagement:
  Dies ist unter der Anforderung eines 24 Stunden / 7 Tage unterbrechungsfreien Betriebs absolut notwendig und bedeutet eine Einbindung in ein Monitoring (Überwachung), Kapazitäts-/Performancemanagement, Konfigurations- und Softwaremanagement, wie auch das Management von Wiederanlauf und der Ausfallsicherheit.
- Unterstützung und Umsetzung mehrerer externer Formate:
  Zur Aufnahme und Übergabe von Geschäftsaufträgen an den Serviceanbieter durch Mandanten und externe Systeme sind eine Reihe von verschiedenen Eingangs- und Ausgangsformaten, Kommunikationswegen und Protokollen zu unterstützen. Diese sind durch das Kernanwendungssystem neutral zu behandeln, damit eine einheitliche, formatunabhängige Verarbeitung durchgeführt werden kann.

Aus den wesentlichen Kundenanforderungen und deren grobe Gewichtung in ihrer Bedeutung für eine Anwendungs- und Systemarchitektur wurde das in Abb. 4 dargestellte Ergebnis erarbeitet (IBM-Projekte bei Transaktionsbanken). Diese Bewertung diente als Basis, um das geeignete Architekturmuster für die Anwendungs- und Systemarchitektur zu finden und um eine Leitlinie für die Schlüsselkriterien („key design criterias") an eine neue Systemarchitektur festzulegen. Die grobe Einschätzung der Relevanz für die zukünftige Anwendungs- bzw. Systemarchitektur enthält bereits das Paradigma „strikte Trennung der Geschäfts- bzw. Fachlogik von der technischen Logik und Steuerung". Als dominante Eigenschaften einer zukünftigen Systemarchitektur wurden die Anforderungen „Performance", „Realtimefähigkeit", „Unterbrechungsfreier Betrieb", „Restart-Fähigkeit",

„Komponentenbasiertes System“, „Prozesssteuerung“ und „Multiplattformfähigkeiten“ ermittelt.

Kundenanforderungen an die Systemarchitektur	Relevanz für Anwendungsarchitektur [AA], Systemarchitektur [SA] ● Stark ◑ Mittel ○ Wenig	
Performance Täglich bis zu 50 Mio. Einzeltransaktionen in 5 Stunden In einer Stunde bis zu 20 Mio. Einzeltransaktionen	SA ●	AA ◑
Mandantenfähigkeit Bedienung verschiedener Mandanten und deren Kunden; Customizing im laufenden Betrieb; Billing und Tracking für Aufträge der Mandanten; Standard und Spezialabwicklung von Manadantenaufträgen	SA ◑	AA ●
Realtimefähigkeit Möglichkeit der direkten Bearbeitung der Transaktionen und Weiterleitung z.B. an Disposition und Buchung Unterstützung von Nicht-realtime-fähigen Schnittstellen wie Batch	SA ●	AA ○
Unterbrechungsfreier Betrieb (7 Tage/24 Stunden)	SA ●	AA ○
Restart-Fähigkeit	SA ●	AA ○
Komponentenbasiertes System Mehrfachverwendbarkeit von Komponenten Hinzufügen, verändern und ersetzen der Bankenprodukte im laufenden Betrieb	SA ●	AA ◑
Standardsoftware Anbindung bzw. Integration von Vendor Software; Teilweise Integration von heutigen Kernsystemen	SA ◑	AA ◑
Prozesssteuerung Service Level Steuerung, Transaktionsüberwachung, Verarbeitungssteuerung, Trennung von Geschäftslogik und technischer Logik	SA ●	AA ○
Multiplattformfähigkeit	SA ●	AA ◑

Abb. 4. Die Abbildung zeigt die wesentlichen Kundenanforderungen an die Systemarchitektur und deren Relevanz für Anwendungs- und Systemdesign

Komponenten der neuen Systemarchitektur

Wir haben bisher die Anforderungen und die abgeleiteten Architekturmerkmale an eine neue Systemarchitektur dargestellt. Eine mögliche Umsetzung wollen wir im Folgenden weiter vorstellen.

Die in Abb. 5 gezeigte Systemarchitektur der Transaktionsmaschine setzt zum einen die Anforderungen von Abb. 4 um, zum anderen bildet sie ein neues Verarbeitungskonzept ab. Dies geschieht mit dem Ziel, Geschäftsprozesse als Ganzes zu steuern und optimal unter Nutzung einer transaktionalen Verarbeitung (mit ACID-Eigenschaften) auszuführen. Der gewählte Begriff „Transaktionsmaschine“ spiegelt den Sachverhalt wieder, dass es sich bei dieser Middle-Ware-Komponente um eine transaktionsorientierte Prozesssteuerung handelt, die Aufträge entgegennimmt und diese autonom und orientiert an vereinbarten Serviceanforderungen flexibel steuert und abarbeitet. Das Ergebnis der Verarbeitung sind abgearbeitete Aufträge (Ergebnisaufträge).

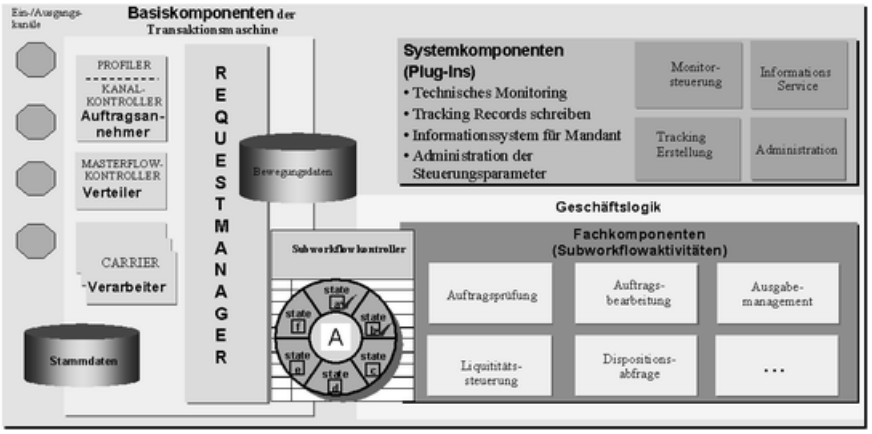

Abb. 5. Die Abbildung beschreibt die Gesamtdarstellung der Transaktionsmaschine und der Fachkomponenten

---

Bei der Abbildung von Geschäftsprozessen auf Transaktionssysteme hat man häufig das Problem von „Long Running Transactions“ zu lösen [13]. In der klassischen Transaktionsverarbeitung (CICS, IMS, DB2, TUXEDO) umgeht man dies durch die Aufteilung der Geschäftslogik in „Units of Works“ (UOW), die in Form von „Short Running Transactions“ ausführt werden. Diese lose gekoppelten Transaktionen erlauben jedoch nur wenig die Sicht auf den Geschäftsprozess als Ganzes.

Die in Abb. 5 gezeigte Transaktionsmaschine bietet die Möglichkeit, Geschäftslogik in der Form von „Short Running Transactions“ auszuführen (s. Komponente Subworkflowkontroller) und die Geschäftsprozesse als Ganzes durch die Komponente Masterflowkontroller abzubilden und zu steuern. Wir werden nun die Basiskomponenten der Transaktionsmaschine erläutern und anschießend im Rahmen eines Beispiels die Arbeitsweise zeigen.

– Kanalkontroller und Profiler: Nimmt eingehende Aufträge von verschiedenen Kanälen (z. B. Internet, Filetransfer, Datenträger, Remote Job Entry, Channel-to-Channel etc.) entgegen und versendet ausgehende Ergebnisse (Ergebnissaufträge, Responses) bzw. Anforderungen (externe Systemanfragen) über die verschiedenen Kanäle. Der Kanalkontroller formatiert alle eingehenden Aufträge in ein „neutrales“ Verarbeitungsformat, um eine einheitliche Verarbeitung zu gewährleisten. In der Ausgaberichtung werden die verschiedenen Ausgabeformate erzeugt. Man spricht hier auch von einem „Multi-Channel-Controller“. Die Profilerkomponente des Kanalkontrollers ermittelt die Verarbeitungsdomain (Cluster von Verarbeitungsknoten), die unter den geforderten Rahmenbedingungen, wie Service-Level-Vereinbarungen, Auftragstyp, Priorität, aktuelle Systemkapazität, benötigte Verarbeitungseinheiten (Service-Units)

etc., den Auftrag abarbeiten kann. Er führt eine adaptive serviceorientierte Lastverteilung zwischen Verarbeitungsdomainen mit der gleichen QoS-(Quality-of-Service-)Charakteristik durch.

- Requestmanager: Verwaltet die verschiedenen Warteschlangen und Ereignislisten der Transaktionsmaschine, über welche die verschiedenen Komponenten kommunizieren.
- Masterflowkontroller: Verteilt die Aufträge innerhalb einer Verarbeitungsdomain auf einen geeigneten Carrier. Die Attribute eines Carriers werden auf der Ebene der Masterflowdefinitionen festgelegt. Der Masterflowkontroller beinhaltet die Grobsteuerung der Prozesse bis auf die Ebene der fachlichen Aktivitäten. Er bildet den serviceorientierten Auftrag an die Verarbeitungsdomain auf eine lokale Workloadmanager Sicht ab (ressourcenorientiert).
- Carrier: Ist ein Container (Trägersystem), der eine gesicherte Transaktionsumgebung zur Ausführung der Geschäftslogik (Aktivitäten) bereitstellt. Er bildet quasi die „äußere" Hülle für eine fachliche Aktivität des Masterflowkontrollers, die auch die globalen und lokalen Daten (Nachrichten), die zwischen den Aktivitäten ausgetauscht werden, mit verwaltet.
- Subworkflowkontroller: Führt innerhalb des Carriers (Containers) die technischen Transaktionen auf Workflow-Ebene durch (short running). Der Subworkflowkontroller arbeitet nach dem Prinzip einer „Finite State Machine" (Kreissymbol mit States in Abb. 5). Hierbei ist die Abbildungsvorschrift so, dass eine fachliche Aktivität des Masterflowkontrollers auf eine Subworkflowkette passt. Eine Subworkflowkette besteht aus 1..n technischen Aktivitäten. Der Subworkflow kann 1..m technische Transaktionen umfassen (logical units of work). Diese Transaktionen müssen ACID-Eigenschaften (Atomicity, Consistency, Isolation, Durability) besitzen, die durch Services des CARRIERs (z. B. Encina, CICS, IMS, Tuxedo) mit unterstützt werden [12].

Fachkomponenten

Die Anwendungslogik (Abb. 5, Geschäftslogik) fügt sich in Form von einfachen, flexibel konfigurierbaren technischen Aktivitäten (Subworkflowaktivitäten) ein. Eine bestimmte Fachkomponente kann aus einer oder mehreren Subworkflowketten bestehen, die jeweils eine nicht unterbrechbare Abarbeitung (auf der Flowebene) von technischen Aktivitäten darstellen. Die Granularität der einzelnen technischen Aktivitäten ist hierbei entscheidend für den Durchsatz und die Komplexität in der Verwaltung des Anwendungssystems. Ein zu

grober Zuschnitt schränkt die Flexibilität bei den Kombinationsmöglichkeiten von technischen Aktivitäten und die notwendige Unterbrechbarkeit stark ein (long running transaction problem [13]). Ein zu feiner Zuschnitt stellt ein Mengenproblem der Verwaltung von technischen Aktivitäten in der Breite dar und produziert in Summe sehr lange Subworkflowketten mit entsprechend höherem „Dispatching"-Overhead seitens der Transaktionsmaschine. Außerdem kann hierbei der Bezug zu den fachlichen Aktivitäten der Geschäftsprozessebene (Masterflowkontroller) verloren gehen, wenn die technische Steuerungsebene zu feingranular arbeitet.

Systemkomponenten

Die Transaktionsmaschine ist über eine Plug-in-Schnittstelle jederzeit durch Systemkomponenten erweiterbar, die nicht in der Basis voll ausgeprägt sind (Abb. 5, Systemkomponenten-Plug-ins). Diese Schnittstelle bedient die operative Ablaufverfolgung (Monitoring), die fachliche Ablaufverfolgung (Tracking), die Verwaltung von Benutzern, Rollen, Rechte und Ablaufdefinitionen (Adminstration) sowie einen Auftragsinformationsdienst (Informationsservice).

Im folgenden Beispiel wollen wir die Arbeitsweise der Transaktionsmaschine erläutern. Die fachlichen Prozesse, die dem Beispiel als Basis dienen, sind mit Rücksicht auf die Übersichtlichkeit stark vereinfacht. Wir wollen die verschiedenen Ebenen der Transaktionsmaschine bezüglich der Steuerung (Masterflow und Subworkflow) und die Verarbeitung unter den verschiedenen Carriern zeigen.

Abbildung 6 zeigt die Definition eines fachlichen Prozesses (Zahlungsverkehr bei Banken) und seine Ableitungen auf den Masterflow und den Subworkflow mit den technischen Aktivitäten. In Abb. 7 sind die entsprechenden fachlichen Aktivitäten und die abgeleiteten technischen Aktivitäten, die für unser Beispiel von Interesse sind kurz beschrieben.

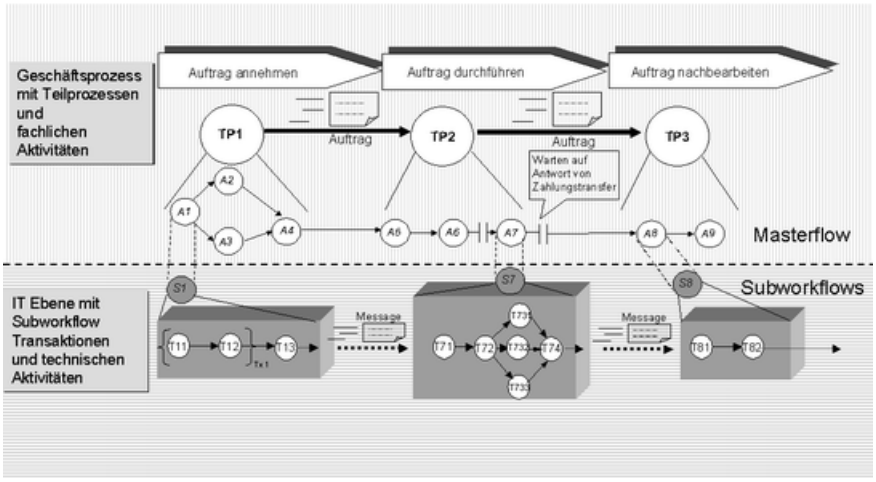

Abb. 6. Die Abbildung zeigt beispielhaft einen Geschäftsprozess mit seinen Teilprozessen und fachlichen Aktivitäten. Die Abbildung auf die IT-Ebene erfolgt durch Umsetzung auf Subworkflowtransaktionen mit ihren technischen Aktivitäten (unterer Teil der Abbildung)

Abb. 7. Die Aufstellung zeigt die fachlichen und technischen Aktivitäten aus dem Beispiel Geschäftsprozess zur Erläuterung der Transaktionsmaschine. Diese werden zum Teil im Textbeispiel verwendet

---

Nehmen wir nun an, eine Transaktionsbank für Zahlungsverkehrsservice erhält von einem Mandanten (z. B. einem großen Versicherer) einen Auftrag (Sammelauftrag) mit einer Million Überweisungspositionen an dessen weltweite Kunden (z. B. Rentenzahlungen aus Lebensversicherungen). Dieser Auftrag wird elektronisch über Filetransfer vom IT-System des Mandanten an die Transaktionsbank übermittelt. Hier kommt unsere Transaktionsmaschine zum Einsatz (Abb. 5).

Die Komponente Kanalkontroller empfängt den Auftrag über den Kanal FTP. Sie analysiert den eingehenden Auftrag, prüft die Berechtigung des Mandanten und führt eine Konvertierung des Auftrags in ein „neutrales“ Verarbeitungsformat durch. Der Profiler stellt

aus den Informationen des Auftrags (wie Mandantenkennung, Auftragstyp, Auftragspriorität und den vereinbarten Service-Level-Parametern) ein QoS-Profil zusammen. Wir nehmen z. B. an, dass der Auftrag zu einem QoS-Profil passt, das eine Verarbeitungszeit von 8 Stunden vorgibt. Der Profiler wählt anhand des QoS-Profils die geeignete Verarbeitungsdomain zur Abarbeitung des Auftrags aus. In unserem Fall sei dies z. B. ein Parallel-Sysplex-Cluster mit 3 Knoten unter z/OS, CICS und mehreren Batch-Adressräumen.

Der konvertierte Auftrag wird in einer der Eingangswarteschlangen des Requestmanagers eingefügt. Diese Warteschlangen werden von der Komponente Masterflowkontroller bearbeitet. Der Masterflowkontroller ermittelt die zu dem Auftrag benötigten Masterflowdefinitionen (s. Abb. 6). Der Masterflowkontroller startet den Masterflow auf dem Parallel-Sysplex-Cluster, beginned mit der Aktivität A1. Für A1 wurden im Rahmen der Subworkflowdefinitionen S1 die Attribute Online (schnelle Antwortzeit), CICS-Profil des Verarbeiters (Carrier) hinterlegt. Somit wird durch den Masterflowkontroller eine CICS-Instanz gestartet (oder eine bestehende Instanz mit dem gleichen Profil wieder verwendet) und der Auftrag übergeben. Der Carrier CICS startet nun eine Subworkflowkontroller-Instanz, die den Subworkflow S1 durchführt. S1 besteht aus den technischen Aktivitäten T11, T12 und T13. T11 stellt ein Cobol-Programm dar, das eine Plausibilitätsprüfung des Auftrags durchführt. T12 prüft die fachlichen Inhalte und bereitet die einzelnen Zahlungspositionen auf. T11 und T12 werden innerhalb einer Transaktionsklammer durchgeführt und committed (Abb. 6). Der Subworkflow S1 endet mit der technischen Aktivität T13, die eine Bestätigungsnachricht für den Auftraggeber erzeugt.

Die von S1 erzeugten externen Nachrichtendaten werden an den Masterflowkontroller übergeben, der die nächste Aktivität anstößt. Nachrichtendaten, die als Ausgabe bestimmt sind, werden über den Kanalkontroller an das entsprechende Mandantensystem weitergeleitet. Nachrichtendaten für die weitere Verarbeitung werden im Rahmen der Eingangsnachricht an die nächste Aktivität weitergegeben.

Schauen wir im Rahmen der weiteren Abarbeitung des Masterflows auf die Aktivität A7. Diese führt, fachlich gesehen, den Zahlungstransfer durch. Der zu A7 korrespondierende Subworkflow S7 wird durch die Attribute Online (mittlere Antwortzeit), CICS-Profil unter einem CICS-Carrier gestartet. Der Subworkflow S7 wird für die technischen Aktivitäten T731, T732, T733 parallel verarbeitet (Start der jeweiligen Transfers pro Bestimmungsland und Bank) und bei T74 wieder zusammengeführt und beendet. T731…T733 haben jeweils

einen Zahlungstransfer angestoßen, warten jedoch nicht, bis die Antworten von den Systemen der jeweiligen Empfängerbanken eintreffen. Aus diesem Grund hat der Masterflow nach der Aktivität A7 eine Unterbrechung (Symbol: parallele senkrechte Linien). Die Kommunikation mit den Systemen der Empfängerbanken wird über die Ausgabenachricht durch den Kanalkontroller durchgeführt, der dann auch die Antworten entgegennimmt und an den Masterflowkontroller die Antwortdaten übergibt. Dieser erkennt, dass der Masterflow bei A7 unterbrochen war und führt die Steuerung bei A8 weiter. Der Subworkflow S8 wurde mit den Attributen „Batch, tägliche Ausführung“ definiert, was dazu führt, dass er unter einem Carrier BATCH gestartet wird. T81 und T82 als technische Transaktionen von S8 führen eine Ergebnisauswertung für die Zahlungstransfers durch und erzeugen die entsprechenden finanztechnischen Buchungssätze usw.

Wichtig war es uns, in diesem Beispiel die Komponenten der Transaktionsmaschine in ihrer groben Arbeitsweise darzustellen und das Zusammenspiel von „Short running Transactions“ des Subworkflowskontrollers mit der „Long-Running“-Prozesssteuerung des Masterflowkontrollers aufzuzeigen. Im Detail sind natürlich auch die Aspekte der Rollen und Sicherheit, Rollback und Recovery, Monitoring und Ablaufverfolgung sowie Lastverteilung interessant. Dies würde jedoch den Rahmen diese Aufsatzes sprengen.

Durch das beibehaltene Prinzip der „Short Running Transactions“ über den Subworkflowkontroller ist es durch den Masterflowkontroller möglich, den Durchsatz an Aufträgen zu maximieren. Dies geschieht u. a. dadurch, dass Unterbrechungen auf der Masterflowebene (d. h. Wartezeiten, Ausgabe, Eingabe) durch die Bearbeitung von anderen Aufträgen gefüllt werden, um die Carrier (Tuxedo, Encina, CICS, IMS, BATCH etc.) der Transaktionsmaschine optimal auszulasten.

Für die Anwendungsarchitektur ergeben sich, vorgegeben durch den Architekturrahmen der Transaktionsmaschine, die folgenden Rahmenbedingungen und Aufgaben:

- Umsetzung der fachlichen Logik durch flexibel einsteckbare technische Aktivitäten;
- Definition eines Programmiermodells bzw. Frameworks für einfache technische Aktivitäten (z. B. einfache Java-Klassen „short running“, Cobol bzw. PL/I Programme im Sinne von Fast Path Transactions oder Single Segmented None-Conversational bei IMS);
- Tooling und Generierung der Ablaufdefinitionen mit Attributen (Transaktionskontext, Billing, Tracking);
- Verwaltung und Management der technischen Aktivitäten;

– Definition der Nachrichtenformate und Datenquellen, Datensenken für die technischen Aktivitäten.

Zur Einordnung der Transaktionsmaschine im Rahmen einer unternehmensweiten Architektur (Enterprise Application Architecture) wollen wir kurz zwei mögliche Szenarien zeigen. Das Szenario von Abb. 8 zeigt auf Basis einer J2EE-Architektur [14] die Einbindung des Enterprise-Information-Servers (EIS, Backoffice) über einen Middle-Tier-Server. Hierbei würde die von uns definierte Transaktionsmaschine ausschließlich auf dem Middle-Tier-Server ausgeprägt sein. Die einzelnen Fachaktivitäten werden dann über J2EE-Konnektoren auf der Basis von Enterprise Java Beans transaktionsgesichert eingebunden. Im dem hier gezeigten Szenario sind sie durch den Konnektor „IMS-Connect“ dargestellt.

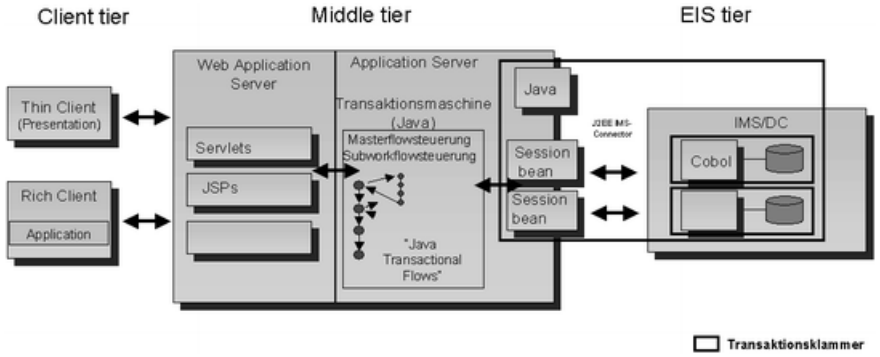

Abb. 8. Anwendung der Transaktionsmaschine. Architekturszenario auf Basis einer J2EE-Anwendungsarchitektur. Die Transaktionsmaschine läuft hierbei vollständig auf dem Middle-Tier-Server. Fachliche Aktivitäten in COBOL werden über J2EE-Konnektoren unter IMS eingebunden. Auf dem Middle-Tier-Server können zusätzlich fachliche Aktivitäten in Java mit eingebunden werden

---

Das zweite Szenario in Abb. 9 zeigt eine 3-Tier-Anwendungsarchitektur auf Basis der Transaktionsmaschine in einer IMS-Umgebung (ein ähnliches Szenario wäre auch für CICS denkbar). Hierbei läuft ein Teil der Transaktionsmaschine auf dem Middle-Tier-Server und ein Teil auf dem Backend-Server unter IMS als Trägersystem (CARRIER). Neben der Nutzung von IMS als Transaktionsmonitor bei der Abarbeitung des Subworkflows (technische Aktivitäten), spielt der hybride Modus in der Verarbeitung z. B. in Cobol, PL/I und Java eine entscheidende Rolle.

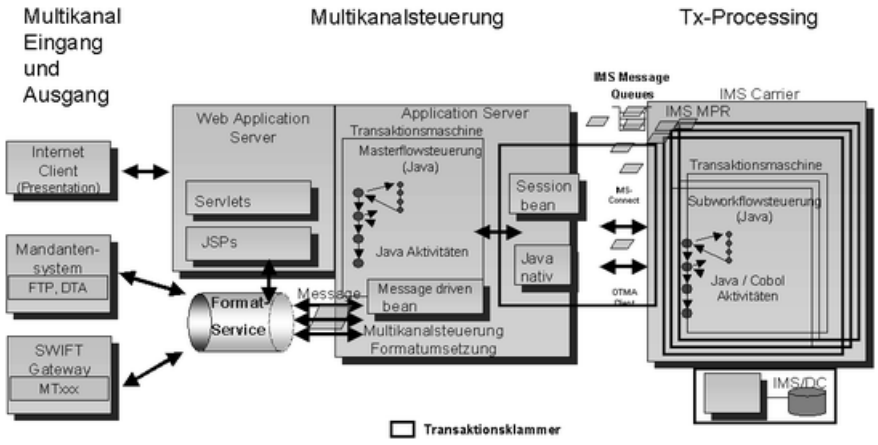

Abb. 9. Anwendung der Transaktionsmaschine. Architekturszenario auf Basis einer 3-Tier-Anwendungsarchitektur mit IMS-Subworkflowsteuerung. Die Transaktionsmaschine läuft verteilt auf einem Middle-Tier- und einem Backend-Server. Technische Aktivitäten werden in einem hybriden Modus (Cobol, PL/I und Java) als Subworkflows unter IMS transaktionsgesichert ausgeführt. Der Middle-Tier-Server übernimmt die Grobsteuerung (Masterflow) und die Multikanalsteuerung mit Formatumsetzung

---

Die zwei gezeigten Szenarien spiegeln nur einen Ausschnitt an möglichen Umsetzungsvarianten wider, die mit der beschriebenen Transaktionsmaschine denkbar sind. Die in Abb. 9 gezeigte verteilte Variante hat in Bezug auf die nichtfunktionalen Eigenschaften wie Durchsatz und Performance eindeutig Vorteile gegenüber der reinen J2EE-Variante von Abb. 8, da alleine schon die Frequenz der Aufrufe über die Konnektoren zum Backend sich um die mittlere Länge der Subworkflows reduziert. Andere wichtige Aspekte sind die Vermeidung bzw. Reduktion von 2-Phase-Committ-Transaktionen und die Verringerung der Sync-Point-Frequenz innerhalb des Subworkflows.

Es wäre auch denkbar, die Transaktionsmaschine unter Nutzung einer 2-Tier-Architektur auf der Basis einer Messaging-Middleware wie z. B. MQ-Series aufzusetzen und über entsprechende Koppelung (z. B. MQ-Series IMS/CICS Bridge) die Backend-Subworkflows anzusteuern.

Auch eine Umsetzung der Transaktionsmaschine als eine eigenständige Middlewarekomponente von Betriebssystemen ist ein möglicher Ansatz und stellt sicherlich unter den Aspekten wie Lastverteilung, Durchsatzoptimierung, Performance, Service-Level-Steuerung eine elegante Lösung dar. z/OS hat hierzu bereits heute wesentliche Komponenten wie Workloadmanager (WLM), Resource Recovery Services (RRS), Internal Resource Lock Manager (IRLM) und Activity Logging zu bieten, um nur einige zu nennen.

Wichtig ist es aber auch zu sehen, dass die dargestellte Transaktionsmaschine als Ganzes zurzeit nicht auf Basis eines „Standard“-Workflowmanagers umgesetzt werden kann. Dies ist zum einen in den hohen nichtfunktionalen Anforderungen wie z. B. dem Durchsatz mit begründet, aber auch in dem Ziel, sehr niedrige Transaktionskosten für die Abwicklung von Standardaufträgen erreichen zu können. STP erfordert eine sehr schlanke autonome Geschäftsprozesssteuerung, orientiert an Mandanten Serviceanforderungen ohne einen personenbezogenen Workflow.

Die Transaktionsmaschine stellt eine Weiterentwicklung und eine Verallgemeinerung bestehender Transaktionsmonitorkonzepte (TP-Monitor) dar.
Weitere Arbeiten sind auf dem Gebiet der Transaktionsmaschine notwendig, um das vorgestellte Konzept hinsichtlich der gesetzten Ziele und Anforderungen zum Erfolg zu führen. Dieses sind u. a.:

- Serviceorientierter Kanalkontroller, der auf der Basis von „echten“ Serviceparametern wie Bearbeitungszeit für einen Auftrag, Preis pro Auftrag, Auftragspriorität, Auftragsart, Sicherheit und Durchführungsgarantie eine geeignete Verarbeitungsdomain bestimmt. Er soll außerdem im heterogenen Umfeld die Einhaltung der geforderten SLAs sicherstellen und eine Lastverteilung für Aufträge an die Masterflowkomponente der Transaktionsmaschine durchführen. Dies stellt eine Erweiterung zum heutigen Stand der Technik dar. Heutige Workloadmanager arbeiten auf Basis nichtfunktionaler Kriterien wie Performance und Antwortzeiten in einem nahezu homogenen Clusterumfeld.
- Hochperformanter Subworkflowkontroller zur Abarbeitung der transaktionalen Workflows auf der Basis bestehender Transaktionsmonitore wie IMS, CICS, TUXEDO, ENCINA oder als eigenständige Java Virtual Machine im Online- und Batch-Modus. Hierbei ist die heutige J2EE-Architektur um eine transaktionale Workflowkomponente im Backend zu erweitern.
- Datenzugriffsdienste und Hochlasttransaktionsverarbeitung für den Datenbankbereich.
- Multikanalsteuerung und Formatumsetzung von Eingabe und Ausgabeformat auf Basis „neutraler“ Geschäftsformate.

Wissenschaftliche Tätigkeiten im Umfeld der Transaktionsmaschine finden bereits an der Universität Leipzig, Institut für Informatik statt. Ansprechpartner sind hierzu Herr Prof. W.G. Spruth und Herr Prof. U. Kebschull.

Zusammenfassung

Wir haben einen Ansatz für das Reengineering existierender Backend-Anwendungen beschrieben. Die Basis hierfür sind praktische Erfahrungen aus Großprojekten mit mehreren deutschen Banken.

Der Ansatz geht von einem Prozessmodell aus, das anschließend mit den Methoden der Informationstechnologie umgesetzt wird. Neuartig ist die Kombination einer „Business Process Flow Engine“ mit einer Reihe von Transaktionsverarbeitungsschritten. Hiermit wird ein als „Straight Through Processing“ (STP) bezeichneter vollautomatischer und standardisierter Geschäftsprozess ermöglicht, ohne manuellen Eingriff in die Geschäftsverarbeitung vom Zeitpunkt des Abschlusses bis zum Zeitpunkt der vollständigen Abwicklung eines Geschäftes.

STP wird als Workflow zu implementieren sein und aus einer ganzen Reihe von kurz laufenden Einzeltransaktionen mit ACID-Eigenschaften bestehen. STP als Ganzes benötigt aber selbst ebenfalls ACID-ähnliche Eigenschaften.

Der beschriebene Ansatz eignet sich besonders für die Neuausrichtung von so genannten Abwicklungssystemen mit hohen Anforderungen an Durchsatz und Ausfallsicherheit. Dieses könnte z. B. erfüllt sein für:

- Abwicklungsgeschäft bei Banken (Transaktionsbanken im Zahlungsverkehr, Wertpapierservice und in der Buchungsabwicklung);
- Massenabwicklung von Geschäftsaufträgen ohne direkte Benutzerinteraktion mit einer Notwendigkeit zur flexiblen Produktdefinition (z. B. Abwicklung von Rentenanträgen, Versicherungsabwicklung etc.);
- STP-orientierte Massenverarbeitung bei Banken und Versicherungen generell.

Die Transaktionsmaschine ist eine mögliche Realisierung. Sie implementiert Long-running-Geschäftsprozesse unter Nutzung der aktuellen technischen Plattform wie Transaktionsmonitore, Datenbanken und Betriebssysteme. Wesentliche Teile können mit Hilfe von J2EE und Enterprise Java Beans (Session Beans) implementiert werden. Für die kurz laufenden Einzeltransaktionen wird es erforderlich sein, auf die existierenden CICS- und IMS-Transaktionsmonitore zurückzugreifen, weil beim heutigen Stand der Technik nur hiermit die erforderlichen Antwortzeiten und Durchsatzeigenschaften erreichbar sein werden. Eine J2EE-Ankoppelung wird über die Java-Connector-Architektur erfolgen.

Die Transaktionsmaschine kann die folgenden Eigenschaften besitzen:

– flache, auf maximalen Auftragsdurchsatz optimierte STP-Ablaufsteuerung,
– integrierte Service-Level-Steuerung,
– Mandantenfähigkeit,
– prioritätengesteuertes Verarbeitungsdomainkonzept,
– Lastverteilung und Durchsatzoptimierung von Einzel- und Sammelaufträgen;
– Handling des kompletten Transaktionskontexts (Global Transaction Coordinator) wie
  – Start und Ende einer Transaktion,

Wir stellen uns die Transaktionsmaschine als ein Designpattern vor, mit dem zukünftige Großprojekte im Backendbereich verwirklicht werden können.

Literatur

1. Kappelmann, L.: Some strategic Y2 K blessings. IEEE Software, Vol. 17, No.2, March/April 2000, p. 42
2. Hardgrave, BC, Reed Doke, E.: Cobol in an object-oriented world. IEEE Software 17 (2), 28 (2000)
3. Spector, A.Z.: The IBM WebSphere Platform. http://www-3.ibm.com/developer/solutionsevent/pdfs/spector lunchtime_keynote
4. Datamonitor: European corporate banking technology strategie. Ref.-Code: BFTC0717, 05/2002
5. Datamonitor: Financial markets technology update, Driving IT Efficiencies. Ref.-Code: BFTC0688, 02/2002
6. Kebschull, U., Spruth, W.G.: Kommerzielle Großrechner als Ausbildungsaufgabe an Universitäten und Fachhochschulen. Informatik Spektrum 24(3), 140–144 (2001)
7. Hammer, M., Champy, J.: Reengineering the corporation. HarperBusiness 1993
8. Martin, J.: The great transition. New York: AMACOM 1995
9. Grover, V., Kettinger, W.J.: Business process change. Harrisburg: Idea Group Publishing 1995
10. Davenport, T.H.: Process Innovation. Boston: Harvard Business School Press 1993
11. Foegen, M., Battenfeld, J.: Die Rolle der Architektur in der Anwendungsentwicklung. Informatik Spektrum 24(5), 290–294 (2001)
12. Leymann, F., Roller,D.: Production workflow. Prentice Hall 2000
13. Bennett, B., Hahm, B., Leff, A., et al.: A distributed object oriented framework to offer transactional support for long running business processes. Lecture Notes in Computer Science, vol. 1795. Berlin Heidelberg New York: Springer 2001
14. J2EE Spezifikation und J2EE Connector Architecture Specification. Verfügbar auf Javasoft Website: http://www.javasoft.com/
15. Herrmann, P., Kebschull, U., Spruth, W.G.: Einführung in z/OS und OS/390. München: Oldenbourg 2002