Unterstützung von Kollaboration im Rahmen der Softwareentwicklung auf Basis Service-orientierter Architekturen

Thomas Karle

Forschung & Entwicklung
PROMATIS software GmbH
Pforzheimer Str. 160
76275 Ettlingen
thomas.karle@promatis.de

Andreas Oberweis

Institut AIFB
Universität Karlsruhe (TH)
Hertzstr. 16
76187 Karlsruhe
oberweis@aifb.uni-karlsruhe.de

Abstract: Im Rahmen der kollaborativen Software-Entwicklung muss die Zusammenarbeit weltweit verteilter Teams unterstützt werden. Die Realisierung von Anwendungssystemen erfolgt heutzutage vielfach auf Basis Service-orientierter Architekturen. Für die Unterstützung von Kollaboration in diesem Umfeld wird ein umfassendes formales System benötigt, das die Zusammenhänge dieser komplexen Lösungen präzise beschreibt. Der Beitrag stellt einen Petri-Netz-basierten Ansatz vor, mit dem Abläufe, Funktionen und Zusammenhänge auf verschiedenen Schichten einer Service-orientierten Architektur beschrieben werden können. Es werden insbesondere die Abläufe in Frontends und deren Zusammenspiel mit Geschäftsprozessmodellen und weiteren Schichten einer Service-orientierten Architektur betrachtet. Weiterhin wird beschrieben, wie der vorgestellte Ansatz Kollaboration in unterschiedlichen Phasen eines Software-Projekts unterstützen kann.

1 Einleitung

Viele Softwareanbieter oder Unternehmen, die sich aus strategischen Gründen für die Entwicklung firmeneigener Software entschieden haben, setzen für die Realisierung ihrer Produkte bzw. Projekte auf Offshoring [Su04]. Zunehmend werden auch Nearshore-Destinationen beispielsweise in Osteuropa attraktiv. Parallel dazu ermöglichen neue Arbeitsformen wie Telearbeit und Telekooperation, die sich durch die Verbreitung des Internets entwickelt haben, geographisch und zeitlich flexible Organisationsmodelle [BC05]. Bei der Realisierung von Systemen auf Basis moderner Service-orientierter Architekturen (SOA) sind Teammitglieder mit unterschiedlichen Kenntnissen und Fähigkeiten beteiligt. Sie realisieren Komponenten auf den verschiedenen Ebenen dieser Architekturen mit den jeweils benötigten Werkzeugen. Aufgrund der Größe der zu realisierenden Systeme, der Komplexität der Service-orientierten Architekturen und des Einsatzes moderner Entwicklungskonzepte (Extreme Programming, Aspekt-Orientierung) ist ein hohes Maß an Zusammenarbeit erforderlich. Weltweit verteilte Projektteams stellen eine weitere Herausforderung bei der Software-Entwicklung dar [CC05].

In Software-Projekten treten besondere Anforderungen hinsichtlich der Zusammenarbeit auf. Nachfolgend sind einige Beispielszenarien aufgeführt:

- Die Projektphasen Analyse, Design, Entwicklung, Test, Schulung und Inbetriebnahme werden bei Software-Projekten in der Regel von unterschiedlichen Personen durchgeführt. Eine effektive Kommunikation unter allen Beteiligten trägt maßgeblich zum Erfolg eines Projektes bei.

- Bei der Analyse müssen die fachlichen Aspekte mit den technischen Möglichkeiten für eine Umsetzung abgeglichen werden, d.h. in dieser Phase ist eine ausgeprägte Kommunikation zwischen den Teammitgliedern mit eher fachlichem Hintergrund und den eher technisch orientierten Systemarchitekten notwendig.

- Beim Design, d.h. dem detaillierten Entwurf der Software-Lösung, müssen neben den Teammitgliedern aus dem Fachbereich auch die Budget-Verantwortlichen miteingebunden werden, da die Entscheidungen, wie Umsetzungen erfolgen sollen, oft signifikante Auswirkungen auf die Kosten des Projekts haben. Die Hauptaufgabe ist hier, Transparenz für die Auswirkungen von Änderungen zu schaffen.

- Bei der agilen Software-Entwicklung, beispielsweise dem Einsatz von Extreme Programming, muss eine Umgebung bereitgestellt werden, in der eine enge Zusammenarbeit und Kommunikation innerhalb eines Teams, bis hin zum gemeinsamen Programmieren von einzelnen Funktionen unterstützt wird.

- In der Testphase bzw. im Rahmen des Betriebs von Software-Systemen müssen bei der Fehlerkorrektur ggf. frühere Teammitglieder hinzugezogen werden, um ein Problem beseitigen zu können.

Bei der Software-Entwicklung sind darüber hinaus die folgenden speziellen Rahmenbedingungen zu berücksichtigen:

- Es wird eine große Anzahl von einzelnen Komponenten erstellt, die in komplexen Gesamtsystemen zusammengefasst werden.

- Die einzelnen Komponenten können einzeln oft nicht betrieben bzw. getestet werden.

- Der Nachweis von Vollständigkeit und Korrektheit eines Gesamtsystems ist sehr aufwendig. Die Kosten für das nachträgliche Korrigieren von bereits implementierten Funktionen sind unverhältnismäßig hoch [CC05].

- Komponenten können parallel von verschiedenen Teammitgliedern bearbeitet werden.

Der Beitrag beschreibt im nachfolgenden Abschnitt zunächst den Aufbau moderner Service-orientierter Architekturen. In Abschnitt 3 werden die Grundlagen von Kollaboration erläutert. Anschließend werden in Abschnitt 4 die Mechanismen der Kollaboration im Rahmen der Entwicklung von Anwendungssystemen auf Basis Service-orientierter Architekturen untersucht. Abschnitt 5 beschreibt ein Petri-Netz-basiertes formales Modell, das als Grundlage für die Realisierung einer Umgebung für kollaborative Software-Entwicklung genutzt werden kann. Die Zusammenfassung und ein Ausblick auf weitere Schritte schließen den Beitrag.

2 Service-orientierte Architekturen

Die Realisierung von Geschäftsprozessen durch zu entwickelnde IT-Systeme erfolgt heutzutage vielfach auf Basis Service-orientierter Architekturen. Als Technik zur Implementierung von Service-orientierten Architekturen haben sich Web Services durchgesetzt [Do05]. Die für die Implementierung der Geschäftsprozesse benötigte Funktionalität wird auf mehrere Schichten verteilt [Du05]. Beispielsweise können komplexe Anwendungssysteme mit der Java Enterprise Edition (JEE) auf Basis einer solchen Architektur realisiert werden. Abbildung 1 zeigt den prinzipiellen Aufbau einer Service-orientierten Architektur und die Abhängigkeiten zwischen den Schichten.

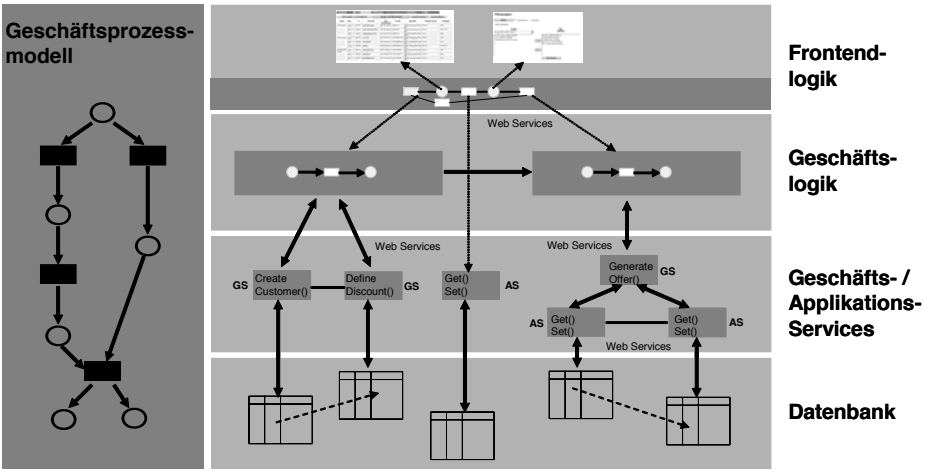

Abbildung 1: Realisierung von Geschäftsprozessen auf Basis Service-orientierter Architekturen

In der Datenbankschicht werden häufig relationale oder objekt-relationale Datenhaltungssysteme eingesetzt. Auch rein objekt-orientierte oder XML-basierte Datenhaltungssysteme sind in der Praxis zu finden. Viele relationale Datenbanken bieten inzwischen auch eine XML-basierte Sicht auf Daten, ohne dass die Daten redundant gehalten werden.

Der Zugriff auf die Daten wird in einer durchgängig Service-orientierten Architektur über die Geschäfts-/Applikations-Services-Schicht bereits mittels Web Services zur Verfügung gestellt. Die Web Services innerhalb dieser Schicht können in Applikations- (AS) und Geschäfts-Services (GS) unterschieden werden. Applikations-Services werden als wieder verwendbare Komponenten definiert, die nur innerhalb bestimmter Applikationen oder Geschäfts-Services genutzt werden können. Geschäfts-Services hingegen werden als wieder verwendbare Komponenten definiert, die direkt in unternehmensweiten oder unternehmensübergreifenden Geschäftsprozessen genutzt werden können.

Die Geschäftslogik wird einer separaten Schicht bereitgestellt. Sie greift auf die bereits Service-orientierte darunter liegende Schicht zu, welche Basisfunktionen, beispielsweise für den Zugriff auf die Datenbank oder auch komplexere Funktionen bereitstellt. In der Geschäftslogik-Schicht werden die Regeln für den Ablauf von Geschäftsprozessen implementiert. Für die Implementierung der Geschäftslogik auf Basis von Web Services hat sich die Business Process Execution Language (BPEL) etabliert. Mit BPEL können durch Orchestrierung von Web Services Geschäftsprozesse implementiert werden.

Die Steuerung des Frontends stellt die Schicht auf der Präsentationsebene bereit. Die Applikationslogik kann sich abhängig vom jeweiligen Frontend unterscheiden. Beispielsweise müssen auf einem mobilen Endgerät Eingabemasken für die Realisierung eines bestimmten Geschäftsprozesses anders dargestellt werden als auf einem PC über einen Web-Browser. Für die Steuerung der Abläufe im Frontend haben sich sogenannte Model View Controller (MVC) durchgesetzt [CST05]. Die Abläufe werden zum einen von der zugrunde liegenden Geschäftslogik und zum anderen von den Aktionen des Benutzers, beispielsweise durch Navigation über Menüs, Links oder Buttons bestimmt. Darüber hinaus muss entsprechend auf eintreffende Ereignisse reagiert werden.

3 Kollaboration

Interaktionsmechanismen, die auf dem computergestützten Austausch von Informationen basieren, werden unter dem Begriff *Computer Supported Cooperative Work* (CSCW) zusammengefasst [We96], [SDK99]. Das Forschungsgebiet CSCW beschäftigt sich mit dem Einsatz von Software zur Unterstützung von Zusammenarbeit und fasst Einflüsse aus den Forschungsgebieten Organisations- und Führungslehre, Psychologie, Informatik und Soziologie zusammen [Gesellschaft für Informatik, CSCW]. Als formale Interaktionsmechanismen mit unterschiedlichen Intensitätsgraden werden Kommunikation, Koordination, Kooperation und Kollaboration unterschieden [We96].

Kommunikation: Unter Kommunikation wird der Austausch von Informationen verstanden, wobei Informationen Daten bzw. Nachrichten in einem entsprechenden Anwendungskontext darstellen.

Koordination: Als Koordination werden Mechanismen bezeichnet, die lenkenden Einfluss auf Aktivitäten und Prozesse bei der Erstellung eines Produkts oder Erbringung einer Dienstleistung haben.

Kooperation: Als Kooperation wird die arbeitsteilige Leistungserbringung zwischen räumlich verteilten Aufgabenträgern oder Organisationen bezeichnet.

Kollaboration: Kollaboration ist ein Spezialfall der Kooperation bei der Teilaufgaben am selben Artefakt durch räumlich verteilte Aufgabenträger durchgeführt werden [Ga06], [St03].

Beispiele für Kollaboration sind die gemeinsame verteilte Erstellung oder Bearbeitung eines Dokuments, einer Software-Komponente oder der gemeinsame Entwurf eines Produkts im Rahmen der Konstruktion. Der Mechanismus der Koordination muss bei der Kollaboration um spezielle Fähigkeiten zur Sicherstellung eines konsistenten, einheitlichen und weiterverwertbaren Arbeitsergebnisses ergänzt werden. Hierzu werden, wie in Abbildung 2 dargestellt, neben dem Informations-, Wissens- und Projektmanagement, ein Konfigurationsmanagement und ein Zugriffs- und Berechtigungsmanagement benötigt. Kollaboration ist nicht auf computerbasierte Interaktionsmechanismen beschränkt. Auch andere Technologien mit denen die Zusammenarbeit von räumlich verteilten Aufgabenträgern unterstützt wird, beispielsweise das Telefon, werden mit einbezogen [KN05].

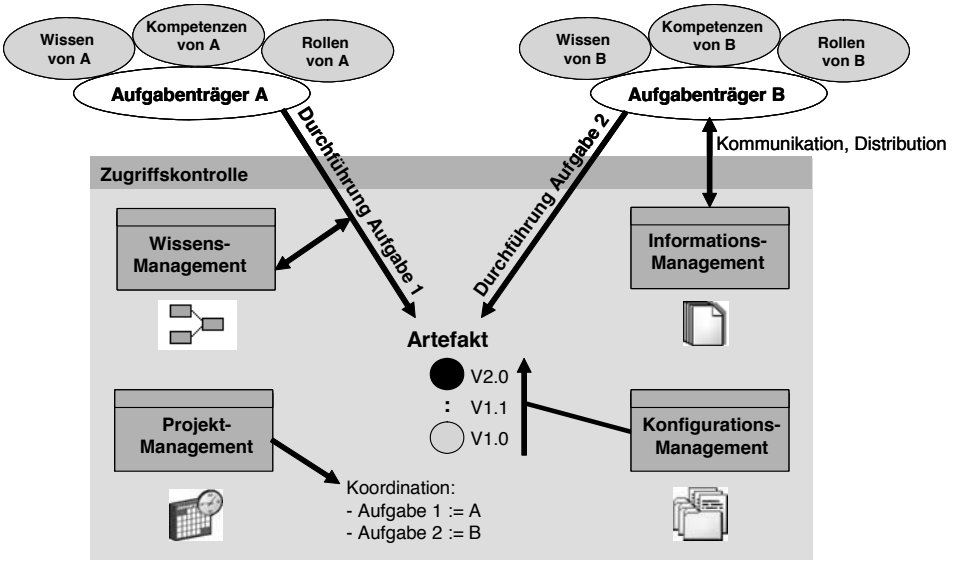

Abbildung 2: Kollaboration

Nachfolgend werden die grundlegenden Komponenten, die für den Aufbau eines Systems zur Unterstützung von Kollaboration benötigt werden, beschrieben.

Zugriffskontrolle: Der Zugang zu den Aufgaben und Informationen und die grundlegende Zugriffskontrolle für Kollaboration sollte durch ein Portal bereitgestellt werden. Portale beinhalten in der Regel bereits eine standardisierte Benutzerverwaltung, einen Login-Mechanismus und eine Zugriffskontrolle auf Ebene der Portalkomponenten, sogenannter Portlets.

Informationsmanagement: Es müssen den Beteiligten gezielt Informationen zur Verfügung gestellt werden. Für die Kommunikation können Komponenten für Mail-, Chat-, Foren, Messaging und Web-Conferencing als Bausteine in ein Kollaborations-Portal integriert werden. Darüber hinaus können über Content-Management-Komponenten sowohl Dokumente, als auch Informationen wie News über das Portal veröffentlicht werden. Durch den Einsatz von Workflow-Technologie kann der Informationsfluss zusätzlich gesteuert werden.

Projektmanagement: Über Funktionen wie Aufgabenverwaltung oder Terminverwaltung kann die Koordination erfolgen. Für die Steuerung und Überwachung kann hier ebenfalls Workflow-Technologie eingesetzt werden.

Wissensmanagement: Im Rahmen von Kollaboration wird Wissensmanagement durch eine Verknüpfung von zu erfüllenden Teilaufgaben und den dazu benötigten Informationen und Wissensträgern realisiert [FSW05]. Zu den benötigten Informationen zählen die definierten und umzusetzenden Prozesse aus der Analyse, weitere beschreibende Dokumente und die Dokumentation der benötigten Werkzeuge. Die zum Erfüllen der Teilaufgabe benötigten Werkzeuge wie beispielsweise eine CAD-Software für die gemeinsame Konstruktion eines Bauteils sollten im Rahmen des Kollaborations-Portals integriert bereitgestellt werden [GH98].

Konfigurationsmanagement: Konfigurationsmanagement wird benötigt, um den Beteiligten jederzeit einen konsistenten Zustand des aktuellen Entwicklungsstandes des gemeinsam zu realisierenden Artefakts anzuzeigen. Das Konfigurationsmanagement darf sich hierbei nicht nur auf die eigentliche Komponente beschränken, sondern muss auch beschreibende Dokumente, Modelle der Geschäftsprozesse oder weitere Informationen berücksichtigen. Der Zugriff auf eine bestimmte Version des zu realisierenden Artefakts sollte dann auch die zusätzlichen Informationen in der richtigen Version bereitstellen, beispielsweise die Dokumentation des aktuell benötigten Entwicklungswerkzeugs. Das Konfigurationsmanagement muss auch einen adäquaten Sperrmechanismus bereitstellen, der je nach Anwendungsfall unterschiedlich sein kann. Dies kann sowohl die Granularität der Sperren, als auch den Typ des Sperrmechanismus betreffen.

4 Kollaborative Software-Entwicklung

Bei der Realisierung von Systemen, basierend auf einer wie in Abschnitt 2 beschriebenen Service-orientierten Architektur, ist das Entwicklungsteam entsprechend der verschiedenen Schichten verteilt. D.h. es gibt Entwickler, die Datenbankobjekte implementieren und ggf. auch den Zugriff über entsprechende Web Services erstellen. Andere Entwickler realisieren Geschäfts-Services und Applikations-Services, die für die Implementierung der Geschäftslogik und Applikationslogik ebenfalls als Web Services bereitgestellt werden. Die Orchestrierung der Web Services zu Geschäftsprozessen mit BPEL erfolgt ggf. wieder durch andere Entwickler. Die Frontend-Programmierer implementieren beispielsweise mit Java die benötigten Applikationsbausteine und nutzen Model View Controller wie Struts oder JavaServer Faces für die Frontend-Steuerung.

Erschwerend kommt noch hinzu, dass die Teams durch Offshore-Entwicklung oder Outsourcing oft räumlich und zeitlich getrennt sind. Darüber hinaus sind zusätzlich noch die Systemanalytiker oder Mitarbeiter aus dem Fachbereich in die Realisierung involviert, beispielsweise wenn Änderungen an den Geschäftsprozessen vorgenommen werden. Aufgrund solcher Teamstrukturen werden Mechanismen für eine erfolgreiche räumlich verteilte Zusammenarbeit im Bereich Software-Entwicklung benötigt. Unter kollaborativer Software-Entwicklung versteht man die Anwendung der Konzepte und Methoden der Computer Supported Cooperative Work im Rahmen von Software-Projekten. Nachfolgend werden diese Mechanismen im Bereich der Realisierung von Anwendungssystemen auf Basis einer mehrschichtigen Service-orientierten Architektur untersucht.

5 Formales Modell

Die lose Kopplung der Schichten bringt Vorteile bzgl. der Flexibilität der einsetzbaren Technologien und der Handhabung der Flexibilität mit sich. Sie stellt jedoch eine neue Herausforderung bzgl. der Beschreibung der Abhängigkeiten zwischen den Schichten und damit des Gesamtsystems dar. Es gibt momentan kein Beschreibungsverfahren, das ein komplettes System schichtenübergreifend beschreibt und Abhängigkeiten über mehrere Schichten hinweg darstellt. Für die kollaborative Software-Entwicklung ist ein solches Beschreibungsverfahren als Basis essentiell, um beispielsweise systemgestützt Abhängigkeiten bei Änderungen erkennen zu können. Nachfolgend wird ein formales Beschreibungsverfahren erläutert, das für die kollaborative Software-Entwicklung auf Basis von Service-orientierten Architekturen dienen soll. Als einheitliche formale Beschreibungssprache werden Petri-Netze verwendet [Ba90]. Je nach Schicht werden entsprechende Varianten der Petri-Netze genutzt.

5.1 Geschäftprozessmodelle

Systemanalytiker erstellen in der Analysephase zusammen mit Mitarbeitern aus den Fachbereichen Modelle der zu realisierenden Geschäftsprozesse. Diese werden als Bestandteil von Pflichtenheften und Spezifikationen den Entwicklern bereitgestellt. Bereits in dieser Phase muss ein System zur Unterstützung von Kollaboration eine weltweit verteilte gemeinsame Erstellung der Geschäftsprozessmodelle durch entsprechendes Konfigurations-, Informations- Wissens- und Projektmanagement unterstützen. In späten Projektphasen werden auf Basis von Change Requests der Fachbereiche oftmals Prozesse geändert. Im Rahmen einer kollaborativen Software-Entwicklung sollte beides unterstützt werden. Die Änderungshäufigkeit in den späten Projektphasen kann durch Kollaboration in früheren Phasen deutlich reduziert werden [LLH06]. Die Bereitstellung der dokumentierten Prozesse aus der Analyse kann beispielsweise durch am Markt verfügbare web-fähige Content-Management-Systeme oder Wissensmanagement-Systeme erfolgen. Eine gezielte automatisierte Information von betroffenen Entwicklern über geplante oder durchzuführende Prozessänderungen in späteren Projektphasen kann damit jedoch nicht realisiert werden.

Die Voraussetzung hierzu ist eine Verknüpfung zwischen den Geschäftsprozessmodellen, deren Implementierung oder zumindest einer formalen Dokumentation der Implementierung und den beteiligten Mitarbeitern. Im formalen Modell werden zur Beschreibung der Geschäftsprozesse *Stellen/Transitions-Netze* verwendet. Abbildung 3 beschreibt einen Geschäftsprozess im CRM-Umfeld, der den Ablauf von einer Kundenanfrage bis hin zur Angebotserstellung beschreibt.

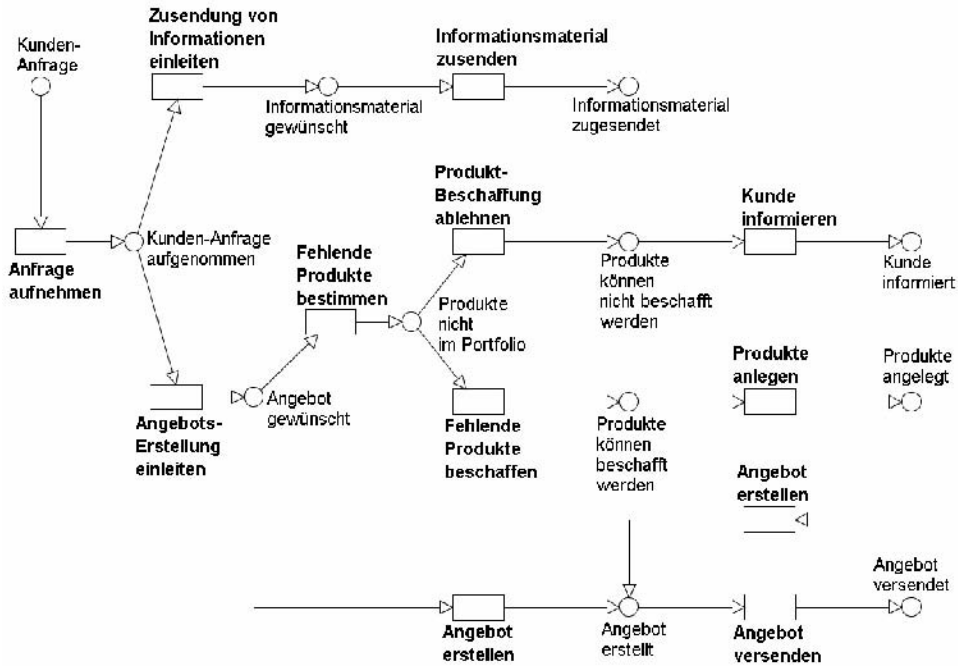

Abbildung 3: Geschäftsprozessmodell als Stellen/Transitions-Netz

5.2 Frontend-Schicht

Im Rahmen des Designs werden die Detailfunktionen und die Detailabläufe in den Applikationen, d.h. die Applikationslogik, festgelegt. In dieser Phase können neue Erkenntnisse aufgrund der Detailtiefe gewonnen werden, die zu Diskussionen und Anpassungen der bereits definierten Geschäftsprozesse aus der Analysephase führen. Sollen in späteren Projektphasen Änderungen an den darunter liegenden Web Services oder an der implementierten Geschäftslogik durchgeführt werden, so kann dies Auswirkungen auf die benötigte Funktionalität und die Abläufe in den Frontends haben. Umgekehrt können Änderungswünsche im Frontend auch Anpassungsanforderungen für Geschäftprozesse auslösen. Dieses Zusammenspiel sollte im Rahmen von kollaborativen Umgebungen unterstützt werden. Im formalen Modell wird die Applikationslogik durch so genannte *Frontend-Netze* beschrieben. Für die zuvor geschilderte Kollaboration müssen die Frontend-Netze mit den entsprechenden Geschäftsprozessmodellen und der darunter liegenden Geschäftslogik verknüpft sein.

Nachfolgend definieren wir für die Frontend-Schicht sogenannte Frontend-Netze, die den Ablauf innerhalb des Frontends beschreiben. Ein Frontend-Netz ist Tupel FN= <$\mathbf{P_P, P_{NE}, T_S, T_{NE}, F}$> das ein Petri-Netz mit folgenden Eigenschaften darstellt:

- Es gibt zwei Typen von Stellen: P_P = Seite und P_{NE} = Navigationselement/Ereignis. P_{NE} repräsentieren beispielsweise Links, Buttons, Menüpunkte oder Ereignisse.
- Es gibt zwei Typen von Transitionen: T_S = Systemaktion und T_{NE} = Navigations-/Ereignisaktion.
- Nach einer Stelle P_P sind nur Transitionen T_{NE} erlaubt.
- Nach einer Transition T_{NE} sind nur Stellen P_{NE} erlaubt.
- Eine Transition T_{NE} ist eine Vergröberung eines Petri-Netzes, das ein exklusives Oder abbildet, d.h. sie stellt eine Entscheidung des Anwenders bzw. das Eintreffen eines bestimmten Ereignisses dar.
- P_P enthält mindestens eine XSL-Datei, in dem die grafischen Attribute der Seite beschrieben sind. Darüber hinaus enthält P_P mindestens eine XML-Datei, in dem die Struktur der Inhalte, definiert ist.
- P_{NE} enthält mindestens eine XML-Datei, das den Input für die nachfolgende Aktion enthält.
- T_S schließt eine Seite bzw. startet eine weitere Seite oder startet die darunter liegende Geschäftslogik-Schicht. Der Zugriff auf die Geschäftslogik entspricht dem Aufruf eines Geschäfts-Services.

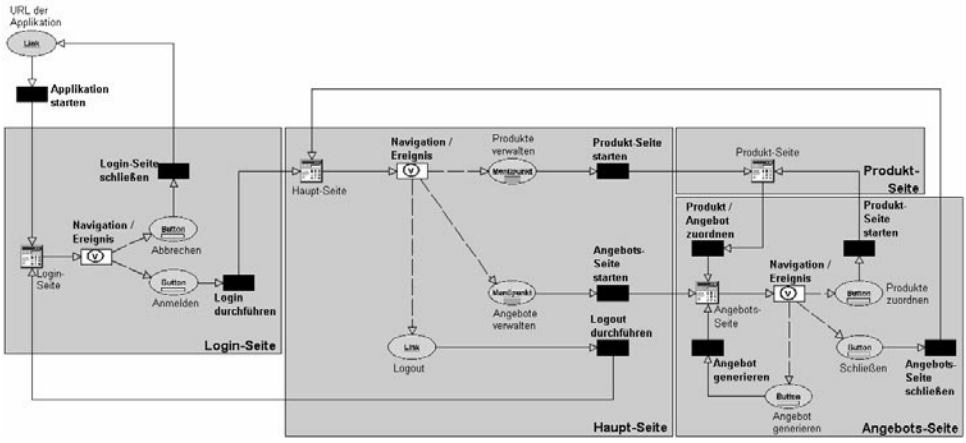

Abbildung 4: Applikationslogik als Frontend-Netz

Abbildung 4 zeigt ein Frontend-Netz, bei dem der Ablauf im Frontend einer web-basierten Applikation zur Unterstützung des oben aufgeführten Geschäftsprozesses modelliert ist. Die schwarzen Transitionen stellen Systemaktionen T_S dar, während die mit einem V markierten weißen Transitionen Navigations-/Ereignisaktionen T_{NE} repräsentieren. Die ovalen Stellen sind Navigationselemente bzw. Ereignisse vom Typ P_{NE}, während die als Icon dargestellten Seiten Stellen vom Typ P_P repräsentieren.

Zunächst wird vom Aufruf über eine URL die Applikation mit einer Login-Seite gestartet. Nach erfolgreichem Login wird die Hauptseite der Applikation gestartet. Über eine Menüstruktur kann der Anwender nun entweder die Produktseite starten, um zu prüfen ob sich ein bestimmtes Produkt im System befindet oder er kann die Angebots-Seite starten, um ein Angebot zu erstellen. Darüber hinaus kann sich der Anwender von der Hauptseite aus ausloggen. Wählt er die Angebots-Seite, dann kann er über den Aufruf der Produkt-Seite, die in diesem Fall zur Auswahl eines Produkts geöffnet wird, ein Produkt für das Angebot auswählen und zuordnen. Über den Button *Angebot generieren* wird ein darunter liegender Geschäfts-Service aufgerufen, der mit den übergebenen Parametern ein Angebot erstellt, das als XML-Datei zurückgegeben und auf der Angebots-Seite angezeigt wird. Dem Web Service werden beispielsweise ein Datum und die gewünschten Produkte mit der jeweiligen Anzahl übergeben. Mit diesen Parametern erzeugt der Web Service dann ein Angebot mit gültiger Angebotsnummer, Angebotsdatum, Gültigkeitsdatum, Produkten mit Preisen, einem Gesamtpreis, der Steuer und einem ausgewiesenen Rabatt basierend auf den Konditionen des jeweiligen Kunden.

5.3 Geschäfts- und Applikations-Services-Schicht

Im Rahmen des Designs werden die benötigten Funktionsbausteine definiert. Diese Funktionsbausteine sollen später implementiert, als Web Services zur Verfügung gestellt und in Applikationen als Applikations-Services oder in der Geschäftslogik als Geschäfts-Services verwendet werden. Änderungen der Web Services der Geschäfts- und Applikations-Services-Schicht können Auswirkungen auf die mit BPEL erstellte Geschäftslogik haben oder ziehen durchzuführende Anpassungen der Applikationslogik im Frontend oder auch der Datenstruktur in der Datenbank nach sich. Im formalen Modell werden die einzelnen Web Services der Geschäfts- und Applikations-Services-Schicht durch *Web-Service-Netze* [KM05] beschrieben. Abbildung 5 zeigt ein Beispiel für ein Web-Service-Netz, das einen Geschäfts-Service *WS_Generiere_Angebot* beschreibt, d.h. einen Web Service, der in der Geschäftslogik-Schicht genutzt werden kann. Der Web Service stellt eine Funktion bereit, die auf Basis einer XML-Datei, die ein Angebotsdatum, einen Kunden für den das Angebot erstellt werden soll und die für das Angebot ausgewählten Produkte enthält, ein entsprechendes Angebot generiert. Als Ergebnis werden die Angebotsdaten dann als XML-Datei bereitgestellt. Der Detailablauf innerhalb eines Web Services wird im formalen Modell durch ein XML-Netz beschrieben [LO03].

Die einzelnen Transitionen entsprechen dann entweder direkt Routinen, die in den im Projekt verwendeten Programmiersprachen, beispielsweise Java, implementiert sind oder noch implementiert werden sollen oder wiederum einfacheren Web Services, die hier intern verwendet werden.

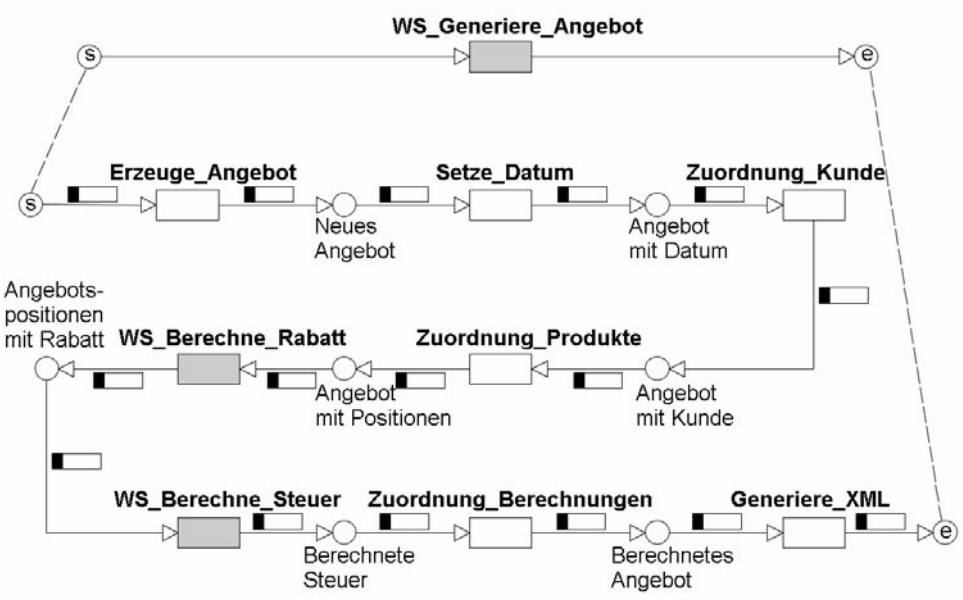

Abbildung 5: Web-Service-Netz mit Detailablauf als XML-Netz

5.4 Geschäftslogik-Schicht

Im Rahmen der Implementierung und des Betriebs wird Geschäftslogik eingerichtet und angepasst. Dies geschieht durch Orchestrierung der Web Services durch BPEL zu Geschäftsprozessen. Änderungen der Geschäftslogik haben direkte Auswirkungen auf die Geschäftsprozesse und dadurch auch auf deren Dokumentation, d.h. die Geschäftsprozessmodelle. Darüber hinaus können Änderungsanforderungen bzgl. der Geschäftslogik auch die durch Web Services realisierte Basisfunktionalität betreffen. Die durch BPEL bereitgestellte Geschäftslogik wird im formalen Modell durch die Verknüpfung von Web-Service-Netzen beschrieben [KM05].

5.5 Zusammenhänge zwischen den Schichten

Der Vorteil dieses komplett auf Petri-Netzen basierenden Ansatzes liegt in der Möglichkeit einer auf der gleichen Modellierungsmethode beruhenden Beschreibung der Zusammenhänge zwischen den Schichten. Nachfolgend werden die Zusammenhänge zwischen den verschiedenen Schichten und deren Beschreibung im formalen Modell dargestellt.

Geschäftsprozessmodell / Frontend: Eine systemtechnisch umgesetzte Transition eines Geschäftsprozessmodells wird einem oder mehreren Frontend-Netzen bzw. einem oder mehreren zusammenhängenden Teilbereichen von Frontend-Netzen zugeordnet. Dadurch wird die Beschreibung des Geschäftsprozesses direkt mit der formalen Beschreibung der Detail-Funktionalität und der Abläufe in einem entsprechenden Frontend verknüpft.

Geschäftsprozessmodell / Geschäftslogik: Ein systemtechnisch durch BPEL umgesetzter Bereich eines Geschäftsprozessmodells wird einem Bereich mit verknüpften Web-Service-Netzen der Geschäftslogik-Schicht zugeordnet.

Frontend / Geschäfts- und Applikations-Services: Eine Transition T_S (Systemaktion) eines Frontend-Netzes wird genau einem Web-Service-Netz der Geschäfts- und Applikations-Service-Schicht zugeordnet.

Geschäftslogik / Geschäfts- und Applikations-Services: Innerhalb der Geschäftslogik-Schicht werden die verfügbaren Geschäfts-Services zu implementierter Geschäftslogik orchestriert. Die einzelnen Geschäfts-Services einer mit BPEL umgesetzten oder umzusetzenden Geschäftslogik sind direkt in der Geschäfts-Logik-Schicht entsprechend der mit BPEL möglichen Ablauflogik (Sequenz, Entscheidung, Parallelverarbeitung etc.) angeordnet.

5.6 Kollaboration auf Basis des formalen Modells

Wenn im formalen Modell den jeweiligen Komponenten, d.h. den Geschäftsprozessmodellen, den Datenbankobjekten, den Applikations-Services, den Geschäfts-Services, den Abläufen in der Geschäftslogik und den Abläufen in den Frontends Mitarbeiter oder Mitarbeitergruppen zugeordnet werden, dann kann dies im Rahmen der kollaborativen Software-Entwicklung genutzt werden.

Basis für eine kollaborative Realisierung eines Anwendungssystems auf Basis Service-orientierter Architekturen ist ein feingranulares Konfigurations- und Zugriffsmanagement. Das beschriebene Modell kann als Grundlage für das Zugriffsmanagement und Steuerung von Änderungsprozessen dienen. Wird beispielsweise ein Objekt in der Datenbank-Schicht geändert oder gelöscht, dann kann diese Information automatisiert an einen Frontend-Entwickler gesendet werden, der aktuell eine Maske entwickelt, in der ein Web Service genutzt wird, der genau dieses persistente Objekt verwendet. Darüber hinaus können über diese Abhängigkeiten Sperren für den Zugriffschutz gesetzt werden. Hier sind sowohl optimistische als auch pessimistische Verfahren möglich. Bei einem pessimistischen Verfahren können bei einer Bearbeitung der persistenten Objekte, die das Objekt nutzenden Web Services für eine Bearbeitung anderer Entwickler gesperrt werden. Erst nach Freigabe des Objekts im Persistenz-Framework können die davon abhängigen Web-Services-Implementierungen wieder geändert werden. Bei einem optimistischen Verfahren lässt man bewusst Änderungen auf den verschiedenen Ebenen zu und prüft beim jeweiligen Zurückschreiben auf Konflikte.

Bei Änderungen und Erweiterungen können Zusammenhänge schnell erkannt werden. Durch die Zusammenhänge können Ansprechpartner für Diskussionen gefunden werden oder automatisch Informationen an entsprechende Mitarbeiter geschickt werden, wenn Komponenten erweitert, geändert oder gelöscht werden. Informationen können gezielt an den Komponenten innerhalb des Modells, für die der jeweilige Mitarbeiter verantwortlich ist, bereitgestellt werden. Das Modell kann als Basis für die Navigation zu den zu realisierenden Komponenten genutzt werden. Dem Benutzer können über einen solchen grafischen Zugang bereits Informationen über Sperren, durchgeführte Änderungen, verbundene Komponenten anderer Schichten, Bemerkungen anderer Teammitglieder, Termine oder weitere relevante Informationen angezeigt werden.

Ebenfalls auf der Basis des zuvor beschriebenen gemeinsamen Modells können Wissensmanagement-Strukturen im Rahmen der Realisierung implementiert werden. Die Verknüpfungen zwischen der in der Designphase erstellten Systemstruktur, in der die zu realisierenden Komponenten aufgeführt sind und den in der Analysephase definierten Geschäftsprozessen liefern den Entwicklern direkten Zugriff auf die fachlichen Informationen.

Im Bereich Projektmanagement können basierend auf dem formalen Modell Informationen über den Fertigstellungsgrad der Implementierung von Geschäftsprozessen automatisch ermittelt und angezeigt werden. Dies kann über die Zusammenhänge im formalen Modell aufgrund der Fertigstellungsgrade der einzelnen Komponenten auf Geschäftsprozessebene aggregiert berechnet werden. Die Koordination kann teilautomatisiert erfolgen, beispielsweise durch die Generierung einer Aufgabe für den Frontend-Entwickler aufgrund einer Änderung an einem Applikations-Service.

6 Zusammenfassung und Ausblick

Der Beitrag zeigt die besonderen Anforderungen bzgl. Kollaboration im Rahmen der Entwicklung von Anwendungssystemen auf Basis Service-orientierter Architekturen. Als Basis für die Unterstützung von Kollaboration in diesem Umfeld wird ein umfassendes formales System benötigt, das die Zusammenhänge dieser komplexen Lösungen beschreibt. Es wurde hier ein Petri-Netz-basierter formaler Ansatz vorgestellt, mit dem auch die Abhängigkeiten zwischen den Schichten beschrieben werden können. Aufbauend auf den beschriebenen Konzepten können Systeme zur Unterstützung von Kollaboration für die Software-Entwicklung realisiert werden. In solchen Systemen muss als Basis ein Repository mit einem entsprechenden Konfigurations- und Zugriffsmanagement für die zu verwaltenden Komponenten, Modelle, Dokumente und sonstigen Informationen realisiert werden. Darauf aufbauend können Funktionen für Kommunikation und Koordination aufgebaut werden, die aufgrund der im formalen System hinterlegten Zusammenhänge gesteuert und teilweise automatisiert werden können.

Literaturverzeichnis

[Ba90] Baumgarten B.: Petri-Netze – Grundlagen und Anwendungen, BI-Wissenschaftsverlag, Mannheim – Wien – Zürich, 1990.

[BC05] Ben E., Claus R.: Offshoring in der deutschen IT Branche – Eine neue Herausforderung für die Informatik, Informatik Spektrum, Band 28, Heft 1, Februar 2005, S. 34-39.

[CC05] Cook C., Churcher N.: Modelling and Measuring Collaborative Software Engineering, University of Canterbury, Christchurch, New Zealand, 2005.

[CST05] Cardone R., Soroker D., Tiwari A.: Using XForms to Simplify Web Programming, IBM Watson Research Center, New York, 2005.

[Do05] Dostal W., Jeckle M., Melzer I., Zengler B.: Service-orientierte Architekturen mit Web Services – Konzepte, Standards, Praxis, Spektrum Akademischer Verlag, München, 2005.

[Du05] Doubrovski V, Grundler J., Hogg K., Zimmermann O.: Service-Oriented Architecture and Business Process Choreography in an Order Management Scenario: Rationale, Concepts, Lessons Learned, OOPSLA 05, San Diego, 2005.

[FSW05] Fuchs-Kittowski F., Stahn P., Walter R.: Wissensmanagement und E-Collaboration - Ein Framework für Communities, Teams und Netze zur Unterstützung kooperativer Wissensarbeit, Fraunhofer Institut für Software- und Systemtechnik ISST, Berlin, 2003.

[Ga06] Gasson S.: Boundary Knowledge-Sharing in E-Collaboration, Proc. 38th Hawaii Intl. Conf. on System Sciences, Hawaii, 2005.

[GH98] Grundy J., Hosking J.: Serendipity: Integrated Environment Support for Process Modelling, Enactment and Work Coordination. In: Automated Software Engineering, Kluver Academic Publishers, Netherlands, 1998, S. 27 – 60.

[KM05] Koschmider A., Mevius M.: A Petri Net based Approach for Process Model Driven Deduction of BPEL Code, OTM Confederated International Conferences, Agia Napa, Cyprus, 2005.

[KN05] Kock N., Nosek J.,: Expanding the Boundaries of E-Collaboration, IEEE Transactions on Professional Communication, Vol. 48, No. 1, March 2005

[LO03] Lenz K., Oberweis A.: Interorganizational Business Process Management with XML Nets, in: H. Ehrig, W. Reisig, G. Rozenberg, H. Weber (Hrsg.): Petri Net Technology for Communication Based Systems. LNCS 2472, Springer-Verlag, 2003.

[LLH06] Lefebvre E., Lefebvre L., Hen G.,: Cross-Border E-Collaboration for New Product Development in the Automotive Industry, Proc. 39th Hawaii Intl. Conf. on System Sciences, Hawaii, 2006.

[SDK99] Swaby M., Dew P., Kearney P: Model-based Construction of collaborative systems, in: BI Technology Journal, Vol 17, No 4, 1999, S. 78 – 90.

[St03] Stoller-Schai D.: E-Collaboration: Die Gestaltung internetgestützter kollaborativer Handlungsfelder, Dissertation, Universität St. Gallen, Difo-Druck GmbH, 2003.

[Su04] Sury U.: Offshoring: Die rechtlichen Dimensionen, Informatik Spektrum, Forum Offshoring, Band 27, Heft 4, August 2004, S. 365 – 395.

[We96] Wendel T.: Computerunterstützte Teamarbeit – Konzeption und Realisierung eines Teamarbeitssystems, Dissertation, Institut AIFB, Universität Karlsruhe, Deutscher Universitätsverlag, 1996.