UNIVERSITÄT LEIPZIG

LPZ E-BUSINESS
applied telematics

ALEXANDER SCHMITZ,
LOTHAR SCHÖPE

SOFTWARESYSTEM COM42BILL

eBILL PRESENTMENT AND PAYMENT SOLUTIONS
INNOVATIONSBERICHT Nr. 2

INTERNES MEMORANDUM

Universität Leipzig · Institut für Informatik
Lehrstuhl für Angewandte Telematik/e-Business
Prof. Dr. Volker Gruhn

Klostergasse 3 · D-04109 Leipzig
Telefon +49 (341) 97 – 32330
Telefax +49 (341) 97 – 32339
info@lpz-ebusiness.de · www.lpz-ebusiness.de

ISSN 1612-3131

Das Softwaresystem

Alexander Schmitz

Lothar Schöpe

Juni 2003

Inhalt

1 Einleitung

Ein Electronic Bill Presentment and Payment-System (EBPP) ist ein Softwaresystem, das den Ablauf von Transaktionen zwischen Rechnungssteller und Rechnungsempfänger auf elektronischem Wege ermöglicht. Dieses System soll Unternehmen die Möglichkeit bieten, Rechnungen über ein elektronisches, einfach zugängliches und einfach bedienbares Medium zur Verfügung zu stellen, sowie den Endverbraucher in die Lage versetzen, diese Rechnungen zu überprüfen und zu bezahlen.

Durch elektronische Medien wird der Handel erleichtert. Daraus resultieren neue Bedürfnisse an Abrechnungssysteme, um die Möglichkeiten der elektronischen Medien während der gesamten Transaktion zu nutzen und die Nachteile der klassischen Zahlungsarten zu vermeiden.

Wie Abbildung 1 zeigt, ist die beliebteste Zahlungsart die Bezahlung per Rechnung [Wuv99]. Bei dieser Art der Zahlung hat der Kunde die Möglichkeit, seine Zahlungen vor der endgültigen Durchführung noch einmal zu kontrollieren. Verunsichert durch Nachrichten über Betrug oder Falschabbuchungen misstrauen viele Kunden Zahlungsarten, bei denen direkt auf ihr Konto zugegriffen wird.

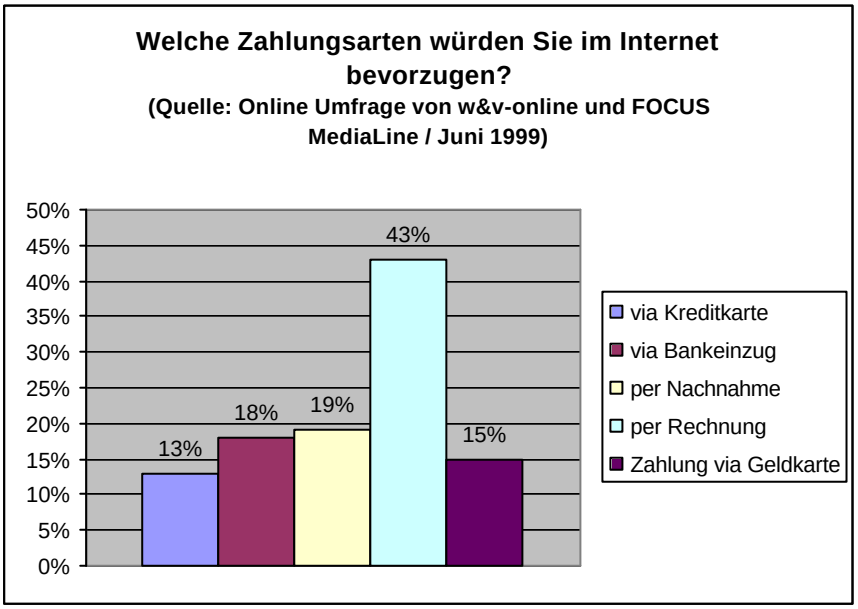

Abbildung 1: Welche Zahlungsarten würden Sie im Internet bevorzugen?

Typischerweise werden Rechnungen in Papierform der Ware beigelegt oder separat zugesandt. Diese Art der Rechnungsübermittlung ist jedoch sowohl auf Rechnungssteller- als auch auf Rechnungsempfängerseite sehr kostenintensiv. Während bei dem Rechnungssteller die Kosten in der Verwaltung und beim Versand sowie der Nachverfolgung der Rechnung anfallen, ist es für den Rechnungsempfänger aufwändig, einen Überweisungsträger auszufüllen und diesen zur Bank zu bringen, wobei hier jedoch vielfach elektronische Teillösungen im Einsatz sind, wie z.B. die Bezahlung von Rechnungen per Homebanking oder M-Payment-Systeme wie zum Beispiel Paybox [Pay01]. Hier beginnt die Idee eines EBPP-Systems. Wenn der Bezahlvorgang elektronisch erfolgt, kann auch die Rechnungsstellung elektronisch erfolgen; der Medienbruch zwischen elektronischem Medium zur Bestellung und ggf. zur Lieferung und einer Rechnungspräsentation auf Papier ist vermeidbar. Weiterhin können dem Rechnungsempfänger in einer zentralen Rechnungsverwaltung zusätzlich Finanzdienstleistungen, wie zum Beispiel Finanzierungen oder Transportversicherungen,

angeboten werden. Die Verlagerung des Zahlungsverkehrs auf elektronische Systeme sorgt für eine Zeitverkürzung zwischen Rechnungsstellung und –ankunft bei dem Kunden. Dies ermöglicht eine schnellere Bezahlung, etwa wenn vor der Lieferung bezahlt werden soll. Des Weiteren kann der Rechnungssteller von einer sicheren Übertragung ausgehen, so dass verlorene oder vergessene Rechnungen als Verzugsgrund entfallen. Zudem entfallen Papierverbrauch und Versand, wodurch mittel- bis langfristig Zeit- und Kosteneinsparungen für die Unternehmen erzielt werden können. Die Option, dass ein derartiges System auch das Mahnwesen übernimmt, bietet ebenfalls Einsparungspotential auf der Seite des Rechnungsstellers.

In diesem Bericht werden die Ergebnisse des internen Forschungsprojekts Com42bill beschrieben.

2 Realisierungsmodelle

2.1 Allgemein

Gegenwärtig werden verschiedene Realisierungsansätze diskutiert, wie eine Implementierung eines EBPP-Systems konzipiert werden könnte. Zwei davon dominieren auf dem heutigen EBPP-Markt – das Thick Consolidator Modell auf der einen, sowie das Thin Consolidator Modell auf der anderen Seite. Ein weiteres Modell stellt das Buyer Direct Modell dar. Es konnte sich bisher nur im B2B-Bereich durchsetzen.

Das Direct Billing Modell setzt einen direkten Kontakt zwischen einem Rechnungssteller und den entsprechenden Rechnungsempfängern voraus. Bei dem Consolidator Modell allgemein agiert eine dritte Firma als Mittler für den Rechnungsvorgang zwischen dem Rechnungssteller und dem Rechnungsempfänger. Aufgrund des Umfangs der Rechnungsdetails, welche der Consolidator zu verwalten hat, unterscheidet man zwei Konzepte. Der Thick Consolidator zeichnet sich durch die Abspeicherung sämtlicher Rechnungsdetails im System aus, wohingegen der Thin Consolidator nur eine Auswahl von Rechnungsdaten zu verwalten hat. Ferner sieht das Thick Consolidator Modell keinen direkten Kontakt zwischen Rechnungsstellern und Rechnungsempfängern während der Zahlungsabwicklung vor. Im Vergleich zu Direct Billing und den Consolidator –Modellen setzt das Buyer Direct Modell einen Teil des EBPP-Software-Systems auf Seiten des Rechnungs*empfängers* voraus.

2.2 Direct Billing

Direct Billing Systeme sind besonders bei Telekommunikationsunternehmen, Internetprovidern und Internetshops sehr verbreitet. Für diese Unternehmen bietet sich der elektronische Übertragungsweg wie selbstverständlich an, da sie letztendlich selbst in diesem Bereich vertreten sind. Der Rechnungssteller betreibt das EBPP-System bei diesem Modell also selbst. Auf elektronischem Wege stellt er die Rechnungen den Empfängern zur Verfügung. Diese sind zugleich seine Kunden und ihm somit auch bekannt, so dass alle zahlungsrelevanten Daten bereits vorhanden sind, wodurch eine zusätzliche Registrierung entfällt (siehe Abbildung 2).

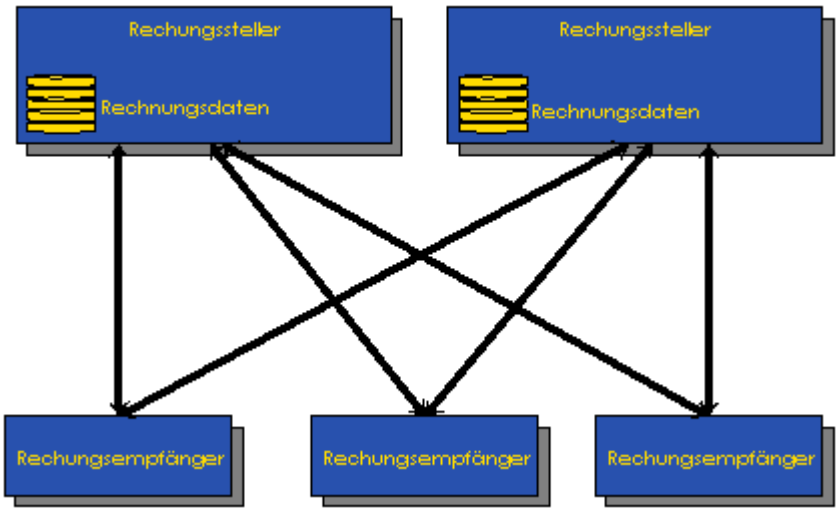

Abbildung 2: Direct Billing-Modell

2.3 Thick Consolidator

Die Rechnungen mit sämtlichen Details, also einer Aufstellung der einzelnen Rechnungsposten, Zahlungsfristen, etc., werden von dem Rechnungssteller an den Consolidator weitergegeben. Diese Rechnungen können dann über den Consolidator auf elektronischem Wege individuell abgerufen werden. Im Idealfall agiert der Consolidator als Verwalter für mehrere Rechnungssteller eines Rechnungsempfängers, der somit diverse Rechungen ohne hohen Aufwand an einem zentralen Punkt überprüfen und begleichen kann. Die Rechnungsdaten werden dem Rechnungsempfänger gegenüber also ausschließlich durch den Consolidator verwaltet.

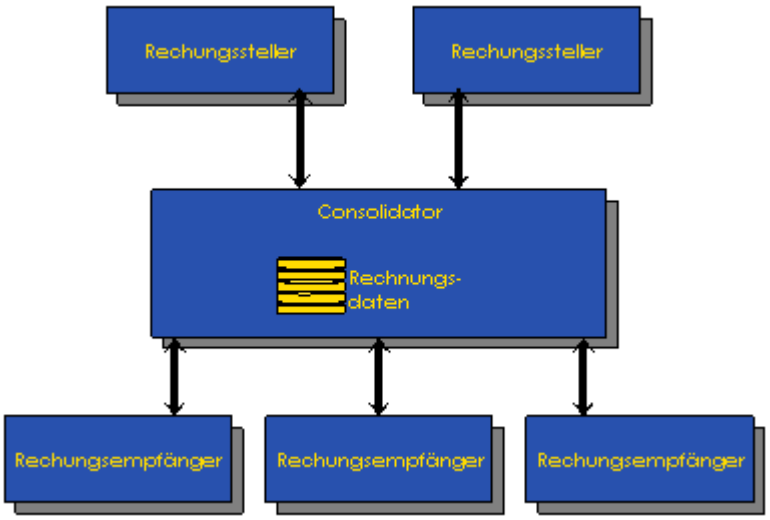

Abbildung 3: Thick Consolidator-Modell

2.4 Thin Consolidator

Über eine Schnittstelle zum Consolidator übermittelt der Rechnungssteller nur die notwendigsten Rechnungsdaten, welche der Rechnungsempfänger über das Internet einsehen kann. Der Rechnungsempfänger erhält dadurch eine eher grobe Rechnungsübersicht. Um zusätzliche Rechnungsdetails zu erhalten, wird er über einen Verweis auf den Server des jeweiligen Rechnungsstellers umgeleitet. Die Details über die jeweiligen Rechnungen werden also bei dem Rechnungssteller gespeichert, der direkte Kontakt zwischen Rechnungsstellern und -empfängern bleibt erhalten. Dadurch erhalten Rechnungssteller zusätzlich die Möglichkeit, Werbung oder sonstige kundenrelevante Informationen den Rechnungsdetails beizufügen.

Nur die zur Bezahlung der Rechnungen notwendigen Informationen bleiben bei dem Consolidator, der schließlich mit der Ausführung der Zahlung beauftragt wird.

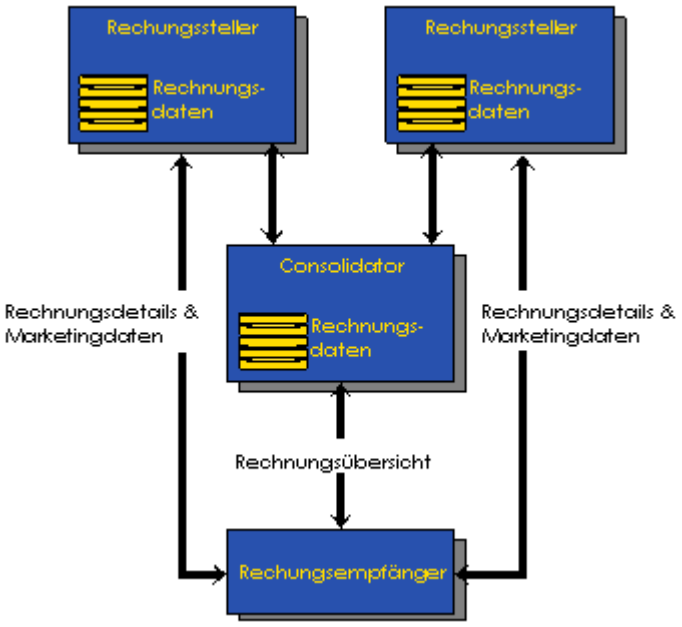

Abbildung 4: Thin Consolidator-Modell

3 Das Modell für Com42Bill

Der Vergleich der drei Modelle zur Realisierung eines EBPP-System zeigt folgendes Bild:

Direct Billing-Systeme sind anwenderspezifisch und nur mit hohem Aufwand an unterschiedliche Rahmenbedingungen anzupassen. Systeme, die dem Buyer Direct-Modell folgen, haben sich Marktstudien zufolge bisher nicht durchsetzen können. Ihr Einsatz ist mit vergleichsweise hohem Aufwand für den Rechnungsempfänger verbunden [So02].

Nur das Consolidator-Modell verspricht eine Realisierung, die vielen verschiedenen potenziellen Kunden gerecht wird: Branchenunabhängigkeit und die Möglichkeit, sämtliche Arten von Waren und Dienstleistungen in unterschiedlichen Zahlungsweisen zu bezahlen sind Vorteile, die mit den anderen Lösungen nicht erreicht werden.

Bei der möglichen Ausprägung des Consolidator-Modells fiel die Wahl auf den Thin Consolidator, der den Austausch und die Verarbeitung nur der notwendigsten Rechnungsdaten vorsieht. Im Vergleich zum Thick Consolidator fallen weniger Daten an, was die Datenhaltungs- und Entwicklungskosten senkt. Eine weitere Stärke des Thin Consolidator-Modells ist, dass der direkte Kontakt des Rechnungsstellers zum Rechnungsempfänger durch ein System dieser Art nicht unterbrochen wird. Dies ist eine Eigenschaft, die für Rechnungssteller ausschlaggebend sein dürfte, im Vergleich der bestehenden Systeme auf Com42Bill zu setzen. Die Ergebnisse einer Experten-Untersuchung [So02], die Consolidator-Modellen im Vergleich eine positive Erfolgstendenz einräumen, bestätigen die Entscheidung, mit Com42Bill einen Thin Consolidator zu realisieren (siehe Abbildung 5).

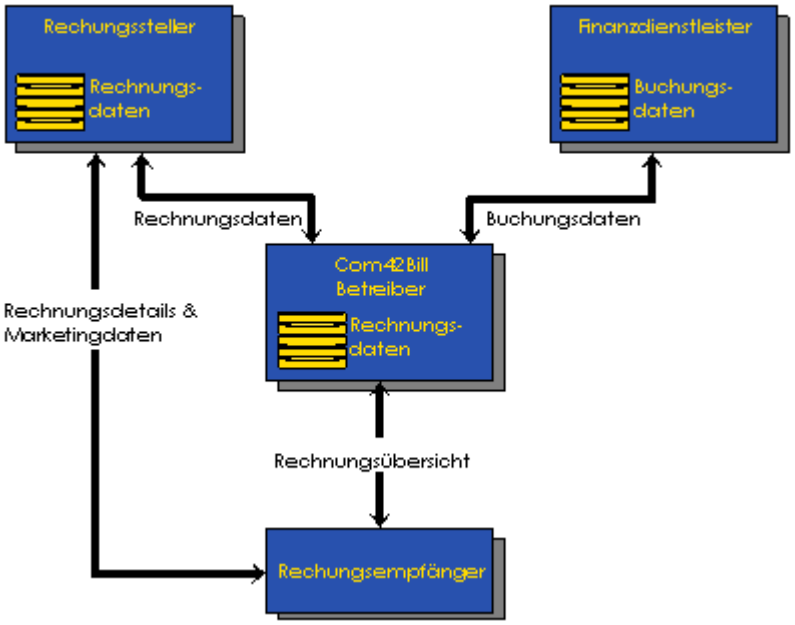

Abbildung 5: Modell für Com42Bill

Die an Com42Bill–Transaktionen beteiligten Parteien lassen sich in vier Gruppen einteilen:

Die Rechnungssteller übermitteln die Rechnungsdaten ihrer Kunden an den Betreiber von Com42Bill. Hierfür bietet Com42Bill eine Programmschnittstelle an. Über eine Weboberfläche kann der Rechnungssteller den Status seiner übertragenen Rechnungen einsehen und seine Daten wie z.B. die Rechnungen verwalten.

Die Rechungsempfänger verwenden Online-Schnittstellen (Web, WAP) von Com42Bill, um ihre persönlichen Rechnungsdaten einzusehen und die Bezahlung ihrer Rechnungen zu veranlassen. Voraussetzung hierfür ist die einmalige Registrierung bei dem Betreiber von Com42Bill. Der Zugang zum System wird durch einen Benutzernamen und ein Passwort ermöglicht. Weiterhin kann sich ein Rechnungsempfänger per E-Mail über Fälligkeiten oder neu eingetroffene Rechnungen informieren lassen.

Die Finanzdienstleister führen die Zahlungsaufträge, die von den Rechnungsempfängern erteilt werden, aus. Durch die flexible Architektur von Com42Bill und den konsequenten Einsatz von offenen Standards wie z.B. ebXML kann das System ohne großen Aufwand an verschiedene Übertragungsprotokolle und Datenformate der Finanzdienstleister und Rechnungssteller angepasst werden.

Der Betreiber des Systems fungiert als Mittler zwischen Rechnungssteller, Rechnungsempfänger und Finanzdienstleister. Die von dem Rechnungssteller erhaltenen Rechnungsdaten werden dem Rechnungsempfänger übermittelt. Nach einer Zahlungsfreigabe des Rechnungsempfängers werden die Zahlungsdaten an einen Finanzdienstleister zur Ausführung übermittelt. Diese Funktionen erfordern kein Eingreifen des Betreibers, die Daten werden automatisch durch das System an die richtigen Empfänger übertragen. Um den reibungslosen Betrieb zu gewährleisten, hat der Betreiber jedoch jederzeit die Möglichkeit, über eine einfach zu bedienende Administrationsoberfläche in den Betrieb einzugreifen.

4 Die Architektur

4.1 Allgemein

Die Systemarchitektur gibt eine Übersicht über die einzelnen Komponenten, welche gemeinsam das EBPP-System bilden. Bereits Es wurde beschlossen das System in fünf Komponenten aufzuteilen. Jede Komponente hat einen klar abgegrenzten Aufgabenbereich. Der Zugriff einer Komponente auf Dienste einer anderen geschieht ausschließlich über definierte Schnittstellen. So ist eine Austauschbarkeit und Unabhängigkeit der Komponenten gewährleistet.

Einige Entscheidungen bzgl. der Architektur haben systemweite Auswirkungen. Zunächst ist der Beschluss zu nennen, einen J2EE-Applikationsserver als Grundlage für das System einzusetzen. Diese Entscheidung ist aus der Überlegung entstanden, dass es im Rahmen eines Forschungsprojektes nahezu unmöglich ist, ein zufriedenstellendes Rahmenwerk für ein komplettes Softwaresystem zu entwerfen und entwickeln. Als schwierigste Punkte sind hier das Transaktionsmanagement sowie das Objekt-Relationale-Mapping der Java-Objekte auf relationale Datenbankstrukturen zu nennen. Diese Überlegungen haben zu der Ansicht geführt, dass durch den Einsatz eines Applikationsservers das Softwaresystem Com42Bill deutlich an Qualität und Flexibilität gewinnen und somit durch die Einhaltung von Standards auch eine höhere Marktfähigkeit erlangen kann. Im Endeffekt hat sich gezeigt, dass diese Entscheidung richtig war. Gerade im Bereich der Persistierung und des objekt-relationalen Mappings hat das EJB-Konzept geholfen, einen gewissen Qualitätsstandard zu erreichen, der dank unserer Entwicklungsumgebung auch problemlos auf Applikationsserver anderer Hersteller übertragen werden kann.

Die zweite systemweite Entscheidung ist der Einsatz von RequestDictionaries. Hierunter sind Container-Objekte zu verstehen, in denen unter einem Namen beliebige Objekte abgelegt werden können (Key-Value-Paare). Die RequestDictionaries werden beim Eintreffen jeglicher Anfragen an das System erzeugt und bleiben mindestens bis zum Versenden der zugehörigen Antwort im System bestehen, maximal jedoch bis zum Beenden der zugehörigen Benutzersitzung. Hierdurch können die Schnittstellen zwischen den einzelnen Komponenten sehr klein gehalten werden. Beim Hinzufügen von benötigten Objekten müssen daher keine Schnittstellen angepasst werden. Jede Komponente nimmt sich die benötigten Informationen aus dem RequestDictionary und fügt eigene Objekte, die aus der Arbeit der Komponente resultieren, hinzu. Der einzige Nachteil eines solchen Mechanismus ist die Notwendigkeit der genauen Spezifikation aller Objekttypen und der zugehörigen Schlüssel durch die jeweiligen Komponenten. Hier muss also das Vorhandensein der benötigten Objekte sichergestellt werden. Der Verwaltungsoverhead, der durch das Anlegen und Synchronhalten der benötigten Konstanten (diese bilden die Keys) entstand, hat sich teilweise als etwas nervenaufreibend erwiesen. Hier war ziemlich viel Sorgfalt bei den Komponententeams notwendig, zumal die unterschiedliche Handhabung meist erst während der Integration auffiel. Trotzdem haben sich die schlanken Schnittstellen als sehr angenehm erwiesen. Man muss nur an wenigen Stellen auf deren Einhaltung achten.

4.2 Komponente GUI

Bei der Gestaltung der grafischen Oberfläche von Com42Bill wird besonderer Wert auf Benutzerfreundlichkeit und Softwareergonomie gelegt. Ein modernes, ausgewogenes Design und ein umfangreiches Hilfesystem sind maßgebende Eigenschaften dieses Systems.

Es wurde eine Oberfläche erstellt, die es ihren Benutzern ermöglicht, die Bedienung und Navigation möglichst einfach und unkompliziert zu erleben. Eine große Rolle spielen eine überschaubare Seitenstruktur sowie die einfache Möglichkeit, die verschiedenen Funktionen, wie beispielsweise die Übersicht über offene Rechnungen, direkt von der Startseite aus zu

erreichen. Das einheitliche Design aller Seiten des Systems trägt dazu bei, es allen Benutzern zu erleichtern, das System wiederzuerkennen und sich darin zurechtzufinden.

Das Kennenlernen des Com42Bill-Web-Interfaces wird durch gezielte Hilfestellungen unterstützt. So wird zum Beispiel jeder Schritt der Rechnungsübersicht verständlich erklärt und erläutert, welche Daten von dem Endbenutzer erwartet werden und in welcher Form diese in die einzelnen Eingabefelder einzutragen sind.

Einen weiteren Auszug aus dem Hilfesystem stellen die häufig gestellten Fragen (FAQ) dar, die bei den meisten Anfangsproblemen die erste Anlaufstelle sind. Auch diese sind selbstverständlich von jeder Seite über das Menü per Mausklick zu erreichen.

Die technische Seite der GUI-Komponente bereitet die darzustellenden Daten in das aktuell benötigte Format auf. Der Zugriff auf das System erfolgt von Seiten der Rechnungsempfänger nur per Internet; hierfür wird eine HTML-Darstellung der Benutzungsoberfläche bereitgestellt. Eine Erweiterung auf andere Ausgabesprachen, wie bspw. WML, ist leicht machbar. Wählt der Rechnungsempfänger eine Funktion aus, leitet die GUI die zur Ausführung nötigen Schritte bei den anderen Komponenten ein.

Diese Komponente stellt ein Web-Publishing Framework bereit, mit dem sich dynamisch Webinhalte generieren und anzeigen lassen. Der wichtigste architektonische Aspekt ist der Einsatz von XML (eXtensible Markup Language) und XSLT (XML-Stylesheet-Language-Transformation). Mit diesen Mechanismen lassen sich auf einfache Weise dynamisch Inhalte für die verschiedensten Ausgabemedien erzeugen. Die Komponente GUI erhält von der Business Logic fachliche Objekte in Form von Entity-Beans, welche zur Anzeige ermittelt werden. Diese müssen nun zunächst in einen XML-Datenstrom transformiert werden. An dieser Stelle ist der einzige Nachteil dieser Technologie zu sehen. Da die meisten J2EE-Applikationsserver Mechanismen anbieten, benötigte Attribute der persistenten Entity-Beans erst bei explizitem Abruf nachzuladen (Lazy Loading), ist es von großer Wichtigkeit, bei der Transformation in XML auch nur die wirklich zur Anzeige benötigten Daten zu übersetzen. Für die Transformation wurden im Laufe der Zeit verschiedene Ansätze diskutiert und ausprobiert. Zunächst wurde die Einsetzbarkeit von Castor (www.castor.org) geprüft. Das Team kam jedoch zu dem Schluss, das Castor für diesen Zweck noch nicht weit genug entwickelt ist. Bei komplexeren Objekten und Relationen gab es keine zufriedenstellenden Lösungen. Der zweite Versuch war eine statische Implementierung der Transformation. Hierbei wurden die XML-Tags „per Hand“ erzeugt. Diese Lösung stieß ebenfalls schnell an ihre Grenzen. Die letzte und aktuelle Lösung erfolgt über die Java-Introspection-API, welche die Struktur der Objekte dynamisch abfragen und somit die benötigten Daten extrahiert werden kann. Liegt nun der XML-Datenstrom vor, kann ein XML-Stylesheet mit Hilfe eines XSLT-Prozessors auf diesen angewendet werden. Durch diese Transformation wird der fertige Ausgabe-Datenstrom erzeugt, welcher zurück zum Client übertragen wird. Lediglich durch den Austausch eines solchen Stylesheets kann bereits ein anderes Ausgabeformat, wie z.B. PDF erzeugt werden.

Vom Einsatz eines fertigen XML-Publishing-Frameworks wurde abgesehen, da die angebotenen Funktionalitäten die benötigten bei weitem übersteigen und meist auf den Einsatz ohne Zuhilfenahme von Applikationsservern ausgelegt sind. Nach Ansicht des Komponententeams GUI würde der Einarbeitungsaufwand den gewonnenen Nutzen in diesem Falle übersteigen. Bei der Umsetzung der Architektur hat das Komponententeam weitestgehend die Subkomponenten direkt in Klassen umgesetzt. Aus Sicht der Architekten bestehen hier noch Optimierungsmöglichkeiten im Bereich des Klassendesigns. Mehrere Hilfsklassen für bestimmte Funktionen erhöhen die Übersicht. Ebenso wären die Einsatzmöglichkeiten von J2EE-Architektur-Patterns zu prüfen.

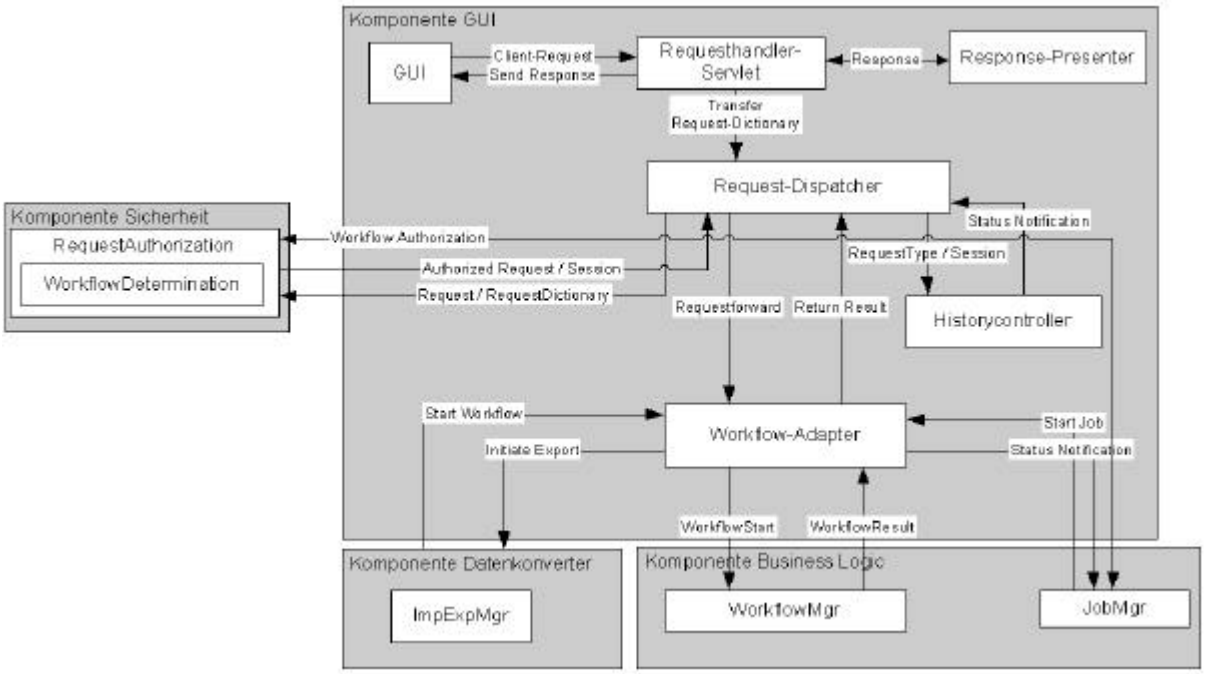

Abbildung 6: Teilarchitektur der Komponente GUI

GUI: Unter diesen Bestandteil fallen alle zur Verfügung gestellten grafischen Benutzeroberflächen, auf die ein Geschäftspartner zugreifen kann.

RequesthandlerServlet: Dies ist der Startpunkt der Verarbeitung aller von der Weboberfläche gestellten Requests. Die vom Eingabemedium abgeschickten Datenformulare bzw. gestarteten Aktionen werden hier entgegen genommen. Das Servlet erhält einen Eingabestrom an Daten, der aufzubereiten ist. Alle Daten, welche sich am http-Request befinden, werden in einem RequestDictionary (Container-Objekt) hinterlegt. Im nächsten Schritt wird die Syntaxkontrolle für die Eingabedaten gestartet. Im Anschluss wird der RequestDispatcher aufgerufen, welcher die weitere Abarbeitung des Requests einleitet. Zuletzt wird der ResponsePresenter zur Erzeugung des Ausgabedatenstroms angestoßen, bevor das Servlet die Daten zurück zum Client überträgt.

RequestDispatcher: Diese Komponente ist die zentrale Verwaltungsstelle der ClientRequests. Der Dispatcher delegiert die Request-Autorisierung und Authentifizierung bei der Sicherheit. Des Weiteren wird geprüft, ob die Anfrage aus dem Cache des Historycontrollers beantwortet werden kann, um die Business Logic nicht unnötig zu belasten. Ebenso erfolgt die Behandlung von Fehlermeldungen, welche im Fall einer fehlerhaften Autorisierung bzw. Authentifizierung durch die Komponente Sicherheit ausgelöst werden. Zuletzt wird die Ausführung des Geschäftsprozesses bei der Business Logic initiiert.

HistoryController: Beim HistoryController ist eine Verbotsliste für nicht erlaubte Übergänge in der Navigationsstruktur der GUI hinterlegt. Hierdurch wird für jeden Request ersichtlich, ob dieser gültig ist und sich somit der Benutzer „an dieser Stelle“ befinden darf. Mit Hilfe dieser Subkomponente lässt sich sicherstellen, dass sich jeder Benutzer mit seiner Session in einem genau definierten Zustand innerhalb des Systems befindet. Der HistoryController arbeitet dabei ähnlich einem Zustandsautomaten, wobei durch das Ausführen von Geschäftsprozessen und dem anschließenden Anzeigen der Ergebnisse Zustände definiert werden. Mit Hilfe der Verbotsliste werden alle unzulässigen Zustandsübergänge festgelegt. Bei

jedem Request wird also der aktuelle Zustand der Session festgestellt und mit Hilfe der aktuellen Anfrage entschieden, ob ein entsprechender Zustandsübergang ausgeführt werden darf, also ob die Anfrage abgearbeitet werden kann. Andernfalls wird der Zustand nicht geändert und der Benutzer erhält eine entsprechende Meldung oder die vorhergehende Seite erneut angezeigt.

Des Weiteren ist im Historycontroller ein Caching-Mechanismus implementiert. Statische Daten können permanent hinterlegt werden, dynamische Daten werden nach dem Ablauf von drei Minuten aus dem Cache entfernt.

WorkflowAdapter: Diese Einheit stellt die einzige Schnittstelle von der Requestverwaltung zur BusinessLogic dar. Der WorkflowAdapter beauftragt den WorkflowMgr, welcher den Einstiegspunkt in die Business Logic darstellt, den Geschäftsprozess zu starten. Ebenso nimmt er die Ergebnisse der Geschäftsprozessausführung entgegen. Der WorkflowAdapter wird auch von der Komponente Datenkonverter und vom JobMgr der Komponente Business Logic zum Starten von Geschäftsprozessen benutzt.

ResponsePresenter: Der ResponsePresenter arbeitet die Ergebnisse des Prozesses, welche sich in dem vom RequesthandlerServlet erstellten RequestDictionary befinden, grafisch auf für die GUI-Anzeige und sendet einen Datenstrom entweder direkt oder über das RequesthandlerServlet an diese. Mit Hilfe der BeanUtils [APA01] aus dem Apache-Jakarta-Projekt werden die Objekte in ein XML-Dokument umgewandelt, welches anschließend vom Xalan-XSLT-Prozessor und einem XSL-File in ein HTML-Dokument weiterverarbeitet wird.

4.3 Komponente Sicherheit

Diese Komponente ist dafür zuständig, alle Vorgänge zu überwachen und sicherheitskritische Vorgänge zu steuern. Es werden Sicherheitsrichtlinien definiert, um ein einheitliches Sicherheitskonzept zu realisieren. Ziel aller Maßnahmen ist ein hoher Sicherheitsstandard, der gewährleistet, dass Daten korrekt und unverändert übertragen werden. Weiterhin muss sichergestellt werden, dass ein Benutzer nur auf die ihn betreffenden Daten zugreifen kann.

Die Sicherheitsrichtlinien stellen eine Grundlage für die Konzeption und Implementierung der einzelnen Komponenten und deren Subkomponenten dar. Sie legen fest, über welche Protokolle mit Programmen außerhalb des Systems wie z.B. WWW-Browsern kommuniziert werden soll.

Bevor ein Rechnungsempfänger bzw. Rechnungssteller das System verwenden kann, muss er sich zunächst anmelden. Die Authentifizierung erfolgt durch eine Benutzeridentifikation und ein individuelles Passwort. Nach erfolgter Anmeldung werden die diesem Benutzer zugewiesenen Zugriffsrechte überprüft. Dadurch wird sichergestellt, dass ein Rechnungssteller bzw. –empfänger nur die Aktionen durchführen kann, für die er autorisiert ist. Insbesondere bedeutet dies, dass sich Rechnungssteller bzw. Rechnungsempfänger nur über die ihnen zugedachten Schnittstellen anmelden können. Während der Ausführung werden bei jeder gewählten Funktion die Berechtigungen überprüft, um missbräuchliche Nutzung des Systems zu verhindern und Kompromittierungsversuche rechtzeitig zu erkennen.

Die Sicherheitskomponente besteht - wie in Abbildung 7 dargestellt - im Wesentlichen aus vier Subkomponenten. Des Weiteren sind Funktionalitäten wie z.B. Datenübertragungsprotokolle beteiligt, welche sich jedoch im Architekturbild nicht zentral zusammenfassen lassen, sondern am gesamten Datenaustausch sowohl innerhalb als auch außerhalb des Systems beteiligt sind.

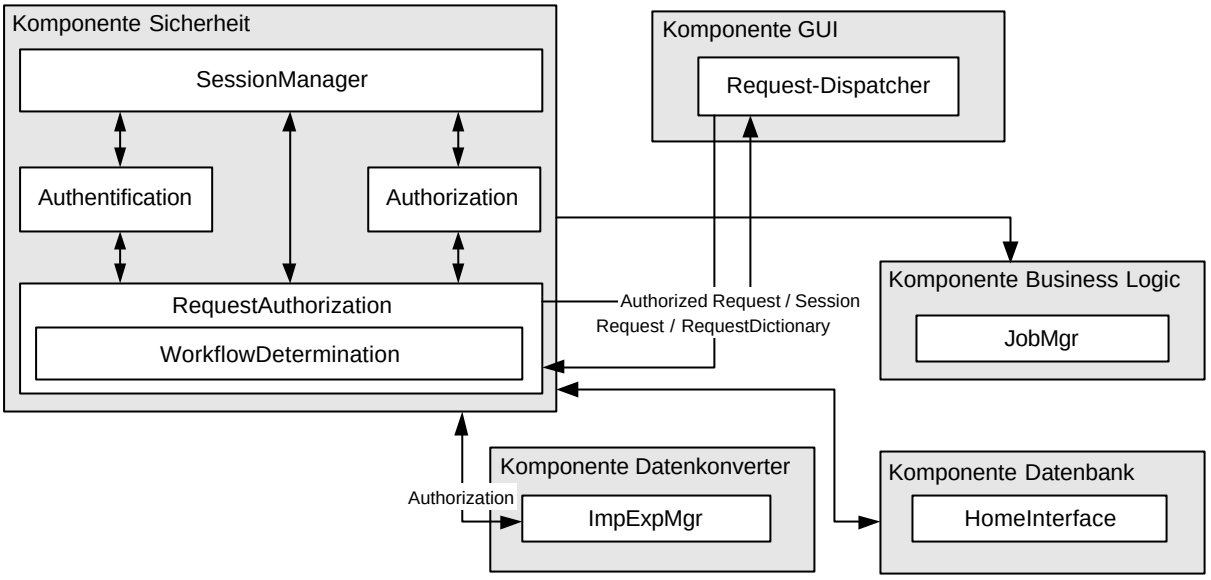

Abbildung 7: Teilarchitektur der Komponente Sicherheit

RequestAuthorization: Die RequestAuthorization ist die erste Anlaufstelle bei der Bearbeitung eines jeden Requests. An dieser Stelle wird die Art des Requests identifiziert und an die entsprechende Untereinheit zur Bearbeitung weitergeleitet. Einzige Ausnahme stellen öffentliche Seiten dar, die ohne weitere Autorisierung angezeigt werden dürfen, da sie nicht auf sensible Daten zugreifen. Diese werden direkt von der RequestAuthorization autorisiert.

SessionMgr: Der SessionMgr ist für die Verwaltung der Benutzersessions verantwortlich. Er bietet Funktionalität zum Auffinden und Bearbeiten von Sessions. Der Einsatz von Sessions ermöglicht das Behalten von Anmeldeinformationen über mehrere Requests. Die Überprüfung auf Gültigkeit einer Session erfolgt über mehrere Schritte. Zuerst wird überprüft, ob eine Session dem SessionMgr bekannt ist. Ist dies nicht der Fall, ist die Session ungültig. Ansonsten wird überprüft, ob die Session abgelaufen ist. In diesem Fall werden die Benutzerinformationen aus der Session entfernt und nur öffentliche Requests erlaubt.

Authentification: Hier findet die Verwaltung aller Benutzer des Systems statt. Hierzu gehören die Rechnungssteller, die Rechnungsempfänger sowie die Administratoren des Systems. Nur Benutzer, die dieser Untereinheit bekannt sind, können sich am System über die ihnen zugewiesene Schnittstelle anmelden.

Authorization: Mit Hilfe dieser Untereinheit erfolgt die Zugriffssteuerung für die einzelnen Geschäftsprozesse. An dieser Stelle ist hinterlegt, welche Benutzergruppe welchen Zugriff auf das System erlangen darf. Diese Funktionalität lässt sich unter dem Oberbegriff *Role Based Access Control* (RBAC) zusammenfassen. Die Rechte beziehen sich dabei auf Objekte, welche geschützt werden müssen. Die Benutzer nehmen nun eine oder mehrere Rollen ein und erhalten darüber bestimmte Rechte auf die Objekte, welche bei Com42Bill in erster Linie Geschäftsprozesse (Workflows) sind. Die höheren Level der RBAC führen komplexere Strukturen wie beispielsweise Hierarchien ein.

4.4 Komponente Business Logic

Die Komponente Business Logic bildet das Kernstück des Systems. Ihre Aufgabe ist es, alle im System anfallenden Geschäftsprozesse abzubilden und bei entsprechendem Aufruf abzuarbeiten. Ein Geschäftsprozess ist dabei eine zusammengehörige Folge von Aufgaben, welche nach bestimmten Regeln auf ein bestimmtes Ziel hin durchgeführt werden. Als Beispiel ist die Durchführung einer Finanztransaktion zu nennen. Von der Beauftragung der Transaktion

durch den Rechnungsempfänger bis zum Empfang einer Statusmeldung am Ende der Transaktion sind verschiedene Teilaufgaben zu erfüllen, welche als Gesamtheit einen Geschäftsprozess darstellen. Die gesamte Verwaltung der Workflows, welche Geschäftsprozesse im System repräsentieren, wird durch die im Folgenden beschriebene Workflow-Engine durchgeführt.

4.4.1 Workflow-Engine

Durch den Einsatz von Workflows hat die Komponente sehr viel an Flexibilität gewonnen. Workflows bestehen aus mehreren Teilabschnitten, welche jeweils kleine Schritte von der Geschäftslogik ausführen, die sog. Business-Objekte. Im Gegensatz zum reinen Einsatz von Session Beans sind die Teile der Geschäftsprozesse nicht mehr fest verdrahtet. Außerdem muss bei einer Änderung der Ablaufreihenfolge nicht mehr der Source-Code angepasst werden, sondern nur noch die XML-Datei. Hierdurch wird auch die uneingeschränkte Wiederverwendbarkeit der einzelnen Business-Objekte garantiert, die nun in mehrere Workflows eingebunden werden können. Abbildung 8 stellt dies am Beispiel einer Zahlungsanweisung dar.

Um eine größtmögliche Flexibilität zu erreichen, müssen weitergehende Funktionen, wie Verzweigungen zwischen Workflows sowie Entscheidungs- und Zusammenführungsknoten, bereitgestellt werden. Aufgrund der Modellierung der Workflows in XML unterliegt man bei der Umsetzung und Entwicklung der Infrastruktur keinen Einschränkungen. Als XML-Sprache für die Workflow-Definitionen viel die Wahl auf die Business-Process-Modelling-Language (BPML) der Business-Process-Modelling-Initiative (BPMI) [BPM00], einem Zusammenschluss aus vielen Vertretern der Wirtschaft. Dieses Vokabular enthält alle benötigten Funktionen.

Die Ausführung der Geschäftsprozesse muss durch eine zentrale Einheit gesteuert und überwacht werden. Mit dieser Kontrolleinheit interagieren die anderen Komponenten mit Ausnahme der Datenbank, die von Teilkomponenten der Business Logic direkt angesprochen wird.

Es lassen sich drei Möglichkeiten unterscheiden, wie ein Geschäftsprozess gestartet werden kann. Die erste Möglichkeit ist der Aufruf durch die Komponente GUI im Rahmen einer Benutzerinteraktion. Mögliche Interaktionen sind zum Beispiel die Bezahlung einer Rechnung oder die Pflege der angegebenen Kontendaten. Die zweite Möglichkeit ist der Prozessstart durch den Datenkonverter, zum Beispiel nach dem erfolgten Import neuer Rechnungsdaten. Die dritte Möglichkeit ist die zeitgesteuerte Ausführung von Standard-Aufträgen, wie zum Beispiel die Mahnungsversendung oder die Durchführung von Finanztransaktionen.

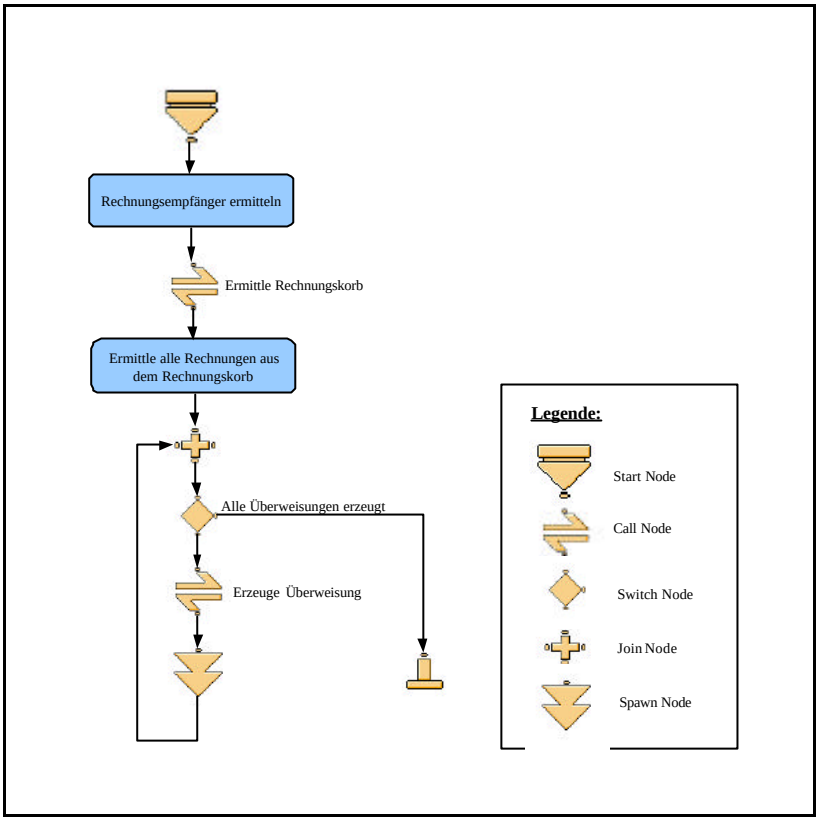

Abbildung 8: Beispiel-Workflow

Eine Teilaufgabe des Komponententeams Business Logic war es, die Auswertung des in XML modellierten Workflows durchzuführen und in eine Aufrufreihenfolge von Business-Objekten umzusetzen. Hierzu wurde der Betrieb der Engine in zwei Phasen eingeteilt, die Initialisierungsphase und die Ausführungsphase. Während der Initialisierungsphase werden die XML-Dateien eingelesen und mit Hilfe von Castor in eine Objektstruktur übersetzt. Zu den wichtigsten BPML-Objekten existieren nun entsprechende Gegenstücke in der Komponente, welche rekursiv erzeugt und parametrisiert werden. Am Ende besteht ein Workflow nur noch aus einer Aneinanderreihung von verschiedenen Workflow-Nodes, wobei jeder Node seinen Nachfolger kennt. Die Ausführung besteht also lediglich aus dem rekursiven Aufruf der einzelnen Nodes, wobei jeder Node andere Aktionen durchführt.

4.4.1.1 BusinessNode

BusinessNodes sind die einzigen Nodes, welche die sogenannte „Business Logic“ ausführen und direkt oder indirekt auf Entitäten arbeiten. Jeder BusinessNode ist für die Ausführung einer Instanz eines bestimmten Business-Objekts verantwortlich. Ein Business-Objekt ist dabei eine stateful SessionBean, welches in mehreren Contexten und somit mehreren Workflows auftreten kann und Aktivitäten ausführt. Die Implementierung der BusinessObjects als stateful SessionBeans hat sich als keine gute Wahl erwiesen. Ursprünglich war vorgesehen, während der Initialisierungsphase eine Instanz des SessionBeans zu erzeugen und diese während der Ausführung immer wieder zu benutzen. Beim Testen haben sich jedoch zwei schwierige Problemfelder ergeben. Zum einen sind stateful SessionBeans nicht Multithreading-fähig. Man kann sie lediglich im Deployment-Deskriptor so modifizieren, dass die Aufrufe sequentiell ausgeführt werden und der Container somit keine Exceptions ausgelöst. Dieser Umstand ist bereits ein Ausschlusskriterium für eine Enterpriseanwendung, wo jedem Request ein neuer

Thread zugewiesen wird. Das zweite Problem besteht darin, dass der Container die Instanz des SessionBeans vernichtet, sobald darin eine Exception auftritt. Als Workaround wurde daher die Erzeugung der Bean-Instanzen in die Ausführungsphase der BusinessNodes ausgelagert, so dass für jeden Request eine neue Referenz vom Container geholt wird. Es wäre zu überlegen, ganz auf den Einsatz von SessionBeans zu verzichten. Die beiden herausragenden Funktionalitäten von SessionBeans, wie das Lifecycle-Management durch den Container und das Transaktionsmanagement werden eigentlich nicht benötigt, sondern von der Workflow-Engine übernommen.

4.4.1.2 CallNode

Der CallNode ist für die Ausführung von Subworkflows zuständig. Am Ende der Subworkflow-Ausführung kehrt die Engine wieder zum CallNode zurück, und der aufrufende Workflow wird weiter ausgeführt. Hiermit kann eine gewisse Modularisierung der Workflows erreicht werden. Es können beispielsweise immer wiederkehrende Aufgaben (sogenannte Prefix-Workflows) in den aktuellen Flow inkludiert werden.

4.4.1.3 JumpNode

Ähnlich wie der CallNode führt ein JumpNode ebenfalls einen Subworkflow aus. Er kann jedoch nur am Ende eines Workflows stehen und kehrt nicht mehr zurück. Auch dieser Node kann das Auftreten von zwei gleichen Handlungsabläufen in verschiedenen Workflows verhindern und sorgt für mehr übersicht.

4.4.1.4 SwitchNode

SwitchNodes sind Entscheidungsknoten, welche Abfragen auf dem Dictionary durchführen und daraufhin den nächsten auszuführenden Knoten auswählen. Ein SwitchNode enthält 0 bis n SwitchCaseNodes. Jeder SwitchCase enthält dabei eine Condition, sowie eine Aktivitätenliste und einen optionalen Kontext. Sollte keine Condition der jeweiligen Cases zutreffen, wird der SwitchDefaultNode ausgeführt, welcher keine Condition mehr enthalten muss.

4.4.1.5 SwitchCaseNode

Beim Zutreffen der Condition im SwitchCaseNode wird dieser ausgeführt. Die Condition besteht aus zwei oder drei Teilen, je nachdem, ob Vergleichswerte benötigt werden.

4.4.1.6 Definition Case-Conditions

Die einzelnen Teile der Conditions werden in dem XML-File eingetragen und durch Semikolons getrennt. Der erste Teil besteht immer aus dem zu überprüfenden Dictionary-Key. Der zweite Teil wird von dem Type-Code der Operation gebildet. Bei Nichtexistenzabfragen muss dann noch im dritten und letzten Teil der Condition der Vergleichswert angegeben werden (vgl. Tabelle 1).

Bedeutung	Type-Code	Beschreibung
Defined	1	Fragt auf dem Dictionary ab, ob ein Key existiert
NOT Defined	2	Fragt auf dem Dictionary ab, ob ein Key nicht existiert
Equals	3	Gleichheitsabfrage auf Strings
!Equals	4	Umgekehrte Gleichheitsabfrage auf Strings
=	5	Gleichheitsabfrage auf Integer
!=	6	Verneinte ...
<	7	Integeroperation
>	8	Integeroperation
<=	9	Integeroperation
>=	10	Integeroperation

Tabelle 1: Conditionen

4.4.1.7 SwitchDefaultNode

Der SwitchDefaultNode wird ausgeführt, wenn keine Condition für einen der SwitchCaseNodes zutrifft.

4.4.1.8 JoinNode

Ein JoinNode führt vorher auseinandergegangene Workflows wieder zusammen. Um an einen JoinNode anzuknüpfen, benutzt man einen JumpNode.

4.4.1.9 OnFaultNode

OnFaultNodes können in allen Kontexten definiert werden und stellen Eventhandler für Nicht-System-Fehler dar. Das bedeutet, dass die Ausführung durch solch einen fachlichen Fehler nicht mit einer Exception unterbrochen wird, sondern der Workflow einen anderen Weg „einschlägt“. Jeder OnFaultNode besitzt einen Code, über den er angesprochen werden kann.

4.4.1.10 FaultNode

Durch die Ausführung eines FaultNodes wird ein Event für die Ausführung des zugehörigen OnFaultNodes ausgelöst. Der Code des FaultNodes muss dabei dem des OnFaultNodes entsprechen.

4.4.2 WorkflowContext

Der WorkflowContext stellt eine Umgebung für die Ausführung der Workflows bereit. Jeder Workflow muss mindestens einen WorkflowContext enthalten. Zusätzlich kann in jeder ComplexActivity (SequenceNode, OnFaultNode, SwitchCaseNode, SwitchDefaultNode) ein eigener WorkflowContext enthalten sein. Mit den jeweiligen Contexts wird eine Hierarchie aufgebaut. Das bedeutet, dass in jedem aktuellen Context auch die Informationen eines ParentContext abrufbar sind.
Zurzeit sind folgende drei Funktionalitäten innerhalb der WorkflowContexts verfügbar.

4.4.2.1 Transaktionen

Transaktionen sind bei allen schreibenden Operationen auf Entitäten notwendig (WebLogic ist zusätzlich der Meinung, dass diese auch zum Navigieren über Container-Managed-Relations notwendig sind), um die Datenkonsistenz zu bewahren bzw. sicherzustellen. Da die Granularität der BusinessObjects möglichst hoch sein sollte, reicht es nicht aus, Transaktionen auf einzelne BusinessObjects zu beschränken. Somit erfolgt die Transaktionssteuerung auf Workflowebene. Es ist daher möglich, in einem beliebigen Context eine Transaktion zu starten.

Diese Transaktion wird spätestens am Ende der Workflowausführung verarbeitet (commit) oder vorher beim Auftreten einer Exception zurückgesetzt (rollback). Vorgesehen ist weiterhin, dass in naher Zukunft Transaktionen auch durch einen Eintrag im Workflow an bestimmter Stelle verarbeitet werden können. Dies wird notwendig sein, wenn man im gleichen Workflow Daten ausliest, in dem sie angelegt wurden.

4.4.2.2 Properties

Um die Dynamik der BusinessObjects weiter zu erhöhen, können in WorkflowContexts Properties angelegt werden, welche aus Key-Value-Paaren bestehen. Diese Parameter werden den BusinessObjects übergeben. Hiermit können die BusinessObjects von außen konfiguriert werden. Hierdurch kann die Anzahl der BusinessObjects drastisch gesenkt werden, wenn zwei Objects ähnliche Aktionen durchführen müssen. Sie werden also parametrisierbar.

4.4.2.3 Faults

Die bereits beschriebenen OnFault-Nodes werden in WorkflowContexts deklariert.

4.4.3 Das Manager-Konzept

Die Komponente Business Logic stellt vier Arten von Basisdiensten zur Verfügung, welche im Rahmen der Ausführung eines Geschäftsprozesses in Anspruch genommen werden: Der erste Dienst stellt Möglichkeiten bereit, die drei Arten von Vertragspartnern (Rechnungssteller, Rechnungsempfänger, Finanzdienstleister) im System zu verwalten. Der zweite Dienst betrifft die Pflege der Rechnungen. Für den Rechnungssteller bietet dieser Dienst die Möglichkeit zur Rechnungsübermittlung, aber auch zur Nachverfolgung und Stornierung bzw. Gutschrift. Der Rechnungsempfänger kann seine Rechnungen einsehen oder zur Zahlung freigeben. Der nächste Dienst befasst sich mit der Verwaltung von Finanztransaktionen. Aufgabe dieses Dienstes ist die Zusammenstellung und Bereitstellung von Transaktionsdaten. Der Rechnungsempfänger kann weiterhin den Status seiner Buchungen verfolgen. Der vierte Dienst stellt das Mahnwesen bereit und übernimmt die Benachrichtigung des Kunden im ersten Schritt per E-Mail.

Die Basisdienste werden in Form von Managern bereitgestellt, welche nach dem Singleton-Designpattern entwickelt wurden. Aufgrund des Einsatzes der Manager bleibt der Benutzer der Entitäten in den Business Objects nahezu von der Verwaltung von EntityBeans und dem dafür benötigten Know-How unberührt. Ebenso wenig muss er sich mit SQL auseinandersetzen. Die Unterteilung des Systems in mehrere Komponenten wird hier konsequent fortgeführt. Somit wird es möglich, die verschiedenen Entwickler mit ihren Spezialfähigkeiten strikt zu trennen. Auf unterster Ebene befinden sich die EJB-Entwickler, welche sich am besten mit J2EE-Technologien auskennen müssen. Bei (u.U. anderen) Beteiligten muss Wissen vorhanden sein, wie man EJBs benutzt. Die Business-Objekte werden von Java-Entwicklern implementiert, welche auf die Dienste der Manager zugreifen können. Auf oberster Ebene im Bereich der Komponente Business Logic werden die Geschäftsprozesse – im Idealfall grafisch – entwickelt. Hierfür ist nun kein Wissen über Programmiersprachen mehr notwendig. Die Weiterentwicklung von Com42Bill kann somit klar strukturiert erfolgen.

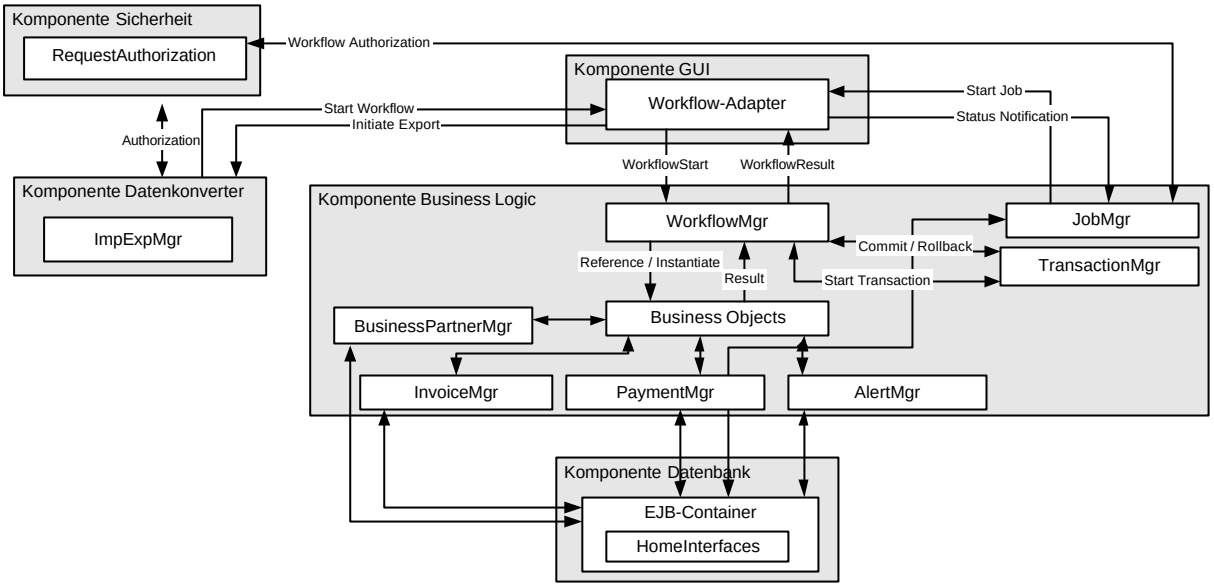

Abbildung 9: Teilarchitektur der Komponente Business Logic

WorkflowMgr: Der WorkflowMgr wird vom WorkflowAdapter aufgerufen. Dieser hat die Aufgabe, den Geschäftsprozess zu starten und zu überwachen. Zu Beginn des Prozesses wird nach Bedarf eine Transaktion gestartet. Im Anschluss werden die einzelnen Workflow-Nodes ausgeführt.

Business Objects: Die Business Objects sind die Komponenten der Business Logic, welche statusbehaftet sein können. Die Business Objects werden im Rahmen eines Geschäftsprozesses referenziert und bilden einen Teil des Prozesses. Sie sind als stateful SessionBeans implementiert. Die Business Objects operieren nun auf den nachfolgend genannten Managern zur Laufzeit eines Geschäftsprozesses.

TransactionMgr: Der Transaktionsmanager hat die Aufgabe, neue Transaktionen zu erstellen und deren Verlauf aufzuzeichnen. Beim Abschluss des Geschäftsprozesses muss die Transaktionen entweder komplett verarbeitet werden (commit) oder komplett rückgängig gemacht werden (rollback). Diese Funktionalität wird durch den Applikationsserver zur Verfügung gestellt und ist über die standardisierte Java Transaction API (JTA) und das Java Naming and Directory Interface (JNDI) zugreifbar.

BusinessPartnerMgr: Dieser Manager verwaltet die Repräsentationen der Rechnungssteller / Rechnungsempfänger und Finanzdienstleister. Die Kernaufgabe ist das Führen der zugehörigen Konten.

InvoiceMgr: Der InvoiceMgr verwaltet alle Vorgänge, welche in Beziehung zur Verarbeitung von Rechnungen stehen.

PaymentMgr: Der PaymentMgr stellt Methoden bereit, Finanztransaktionen auf Basis der Entitäten durchzuführen, bzw. deren Durchführung für einen späteren Zeitpunkt zu planen.

AlertMgr: Dieser Manager verwaltet das Mahnwesen sowie weitere Benachrichtigungsdienste, wie z.B. Benachrichtigungen über fällige oder neu eingetroffene Rechnungen.

JobMgr: Der JobMgr initiiert alle zeitgesteuerten Geschäftsprozesse. Hierzu greift er auf den WorkflowAdapter zu. Der JobMgr hat auch die Möglichkeit, im Anschluss an die Durchführung eines Geschäftsprozesses einen Datenexport zu initiieren.

4.5 Komponente Datenkonverter

Im Rahmen der von Com42Bill angebotenen Dienstleistungen spielt der Austausch von Geschäftsdaten eine entscheidende Rolle. Das System muss Rechnungsdaten vom Rechnungssteller entgegennehmen und Überweisungsaufträge an Finanzdienstleister übermitteln können.

Die Herausforderung beim elektronischen Austausch von Daten besteht im Allgemeinen darin, dass diese von beiden Seiten „verstanden“ werden müssen, um sie weiterverarbeiten zu können. Dieses „Verständnis“ wird vom Datenkonverter gewährleistet.

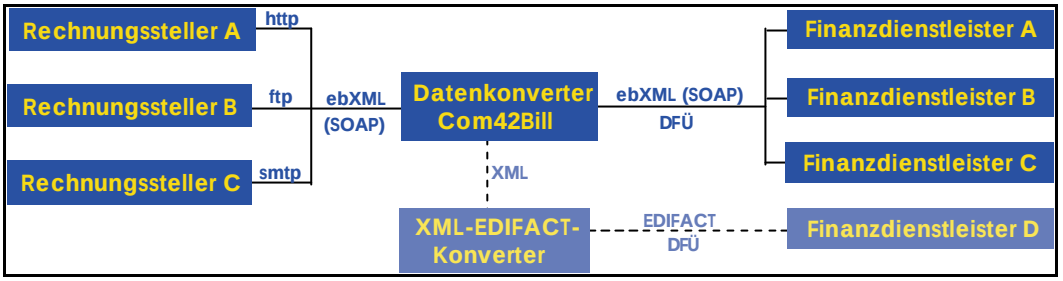

Abbildung 10: Kommunikation von Com42Bill mit anderen Systemen

Der Datenkonverter empfängt Rechnungsdaten von einem Rechnungssteller. Da diese jedoch für gewöhnlich nicht in einem standardisierten Format vorliegen, werden sie in ein eigenes Format konvertiert, um die Daten intern weiterverarbeiten zu können. Hierfür muss vor der ersten Übermittlung ein gemeinsames Austauschformat definiert werden. Ein geeigneter Ansatz hierfür ist der ebXML-Standard.

Im Weiteren müssen die Daten der von den Rechnungsempfängern veranlassten Buchungen an die Finanzdienstleister übermittelt werden. Hierzu kann der Datenkonverter die Daten in ein Format konvertieren, welches der Finanzdienstleister verarbeiten kann. Die Übermittlung der Daten erfolgt ebenfalls auf der Basis des ebXML-Standards. Bei Bedarf können die Buchungsdaten über einen separaten Konverter in EDIFACT konvertiert und dem Finanzdienstleister bereitgestellt werden (siehe Abbildung 10).

Der Datenaustausch erfolgt in beiden Fällen völlig automatisiert. Der Empfang der Rechnungsdaten erfolgt über eine Online-Schnittstelle, d.h. über eine direkte Datenverbindung, so dass die Daten innerhalb des Systems zügig weiterverarbeitet werden können. Welches Transportprotokoll für den Datentransport verwendet wird, ist unabhängig von dem verwendeten Austauschformat. Dadurch kann der Datenaustausch mit Com42Bill einfach an die Erfordernisse eines Rechnungsstellers angepasst werden. Die Buchungsdaten werden dagegen aus Sicherheitsgründen über eine DFÜ-Verbindung an den Finanzdienstleister übermittelt. Der Datenkonverter stellt die Programmschnittstelle zu den Vertragspartnern bereit. Grundlage der Architektur dieser Komponente ist das ebXML-Framework [Ebx02], welches Mechanismen bereitstellt, unabhängig vom verwendeten Datenaustauschformat Dokumente mit Geschäftspartnern auszutauschen.

4.5.1 ebXML – eine allgemeine Einführung

Die Vereinigten Nationen (UN/CEFACT) und die Organisation für die Förderung Strukturierter Informationsstandards (OASIS) haben in Zusammenarbeit ein weltweites Projekt gestartet: die Electronic Business XML Initiative (ebXML). Das Ziel ist es, ein Rahmenwerk zu definieren, bei dem XML in einer standardisierten Art und Weise für den Austausch elektronischer Geschäftsdokumente verwendet wird. Dieses Rahmenwerk soll die Schaffung konsistenter,

robuster und integrationsfähiger eBusiness-Anwendungen ermöglichen und letztendlich dadurch einen einzigen globalen eBusiness-Markt schaffen, auf dem sich Unternehmen jeder Größe und jeder geografischen Lage treffen können.

Das Rahmenwerk stellt keine konkreten Implementierungen bereit, sondern beschreibt detailliert die Schnittstellen der einzelnen Komponenten und stellt zahlreiche Informationsmodelle bereit (siehe ebXML-Spezifikationen).

4.5.1.1 Geschäftsprozesse

ebXML basiert auf einer geschäftsprozessorientierten Sichtweise. Dabei beschreibt ein Geschäftsprozess detailliert, wie und wann ein Geschäftsteilnehmer bestimmte Rollen einnimmt, welche Verbindungen es zu anderen Geschäftsteilnehmern gibt und welche Verantwortlichkeiten in diesem Zusammenhang auf den einzelnen Geschäftsteilnehmern lasten. Es geht bei der Spezifikation von Geschäftsprozessen nicht darum, zu beschreiben, wie die Dokumente (z. B. Rechnungsdaten) auszusehen haben, die zwischen den Geschäftsteilnehmern ausgetauscht werden. Vielmehr gilt es zu beschreiben, wie die Interaktion zwischen den Geschäftsteilnehmern im Rahmen eines ebXML-Geschäftsprozesses (z. B. Bezahlvorgang) aussieht.

4.5.1.2 Geschäftspartnerprofile

Die Zusammenführung von Geschäftspartnern wird unterstützt durch XML Dokumente. Jeder Geschäftspartner, der ebXML nutzen möchte, muss die folgenden Dokumente bereitstellen:

Collaboration Protocol Profile (CPP):

Ein CPP dient der öffentlichen Bekanntmachung eines Unternehmens. Es beschreibt, welche Geschäftsprozesse, Nachrichtenformate, Kommunikationsschnittstellen etc. ein Unternehmen unterstützt.

Auf Basis dieser Informationen können Unternehmen zueinander finden. Wenn ein Unternehmen ein anderes Unternehmen mit einem passenden CPP gefunden hat, können die weiteren Vereinbarungen für die Durchführung elektronischer Geschäftstransaktionen untereinander getroffen werden.

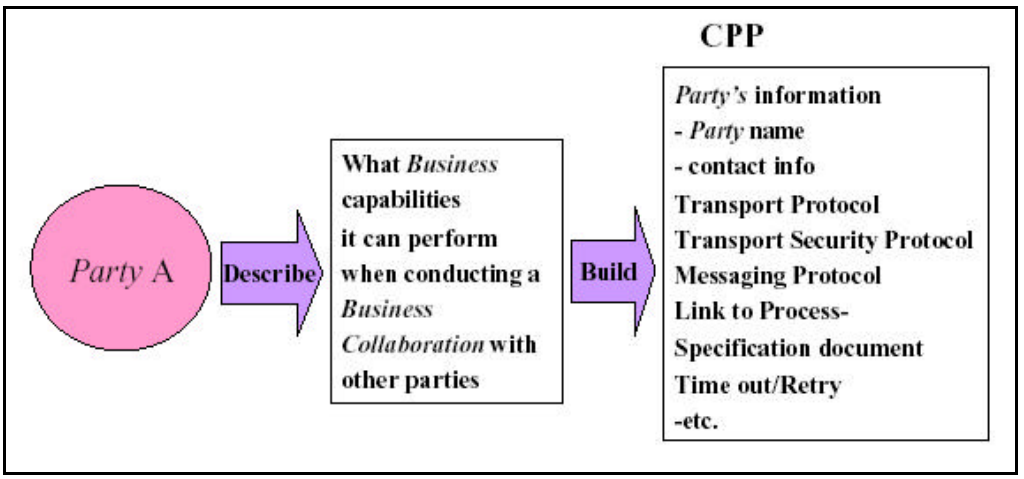

Abbildung 11: Collaboration Protocol Profile (CPP)

Collaboration Protocol Agreement (CPA):

Die Vereinbarungen zwischen zwei Unternehmen resultieren in einem sog. CPA. Ein CPA ist also ein Vertrag zwischen zwei Geschäftspartnern, in dem die Geschäftstransaktionen sowie die technischen Schnittstellen für einen automatisierten Datenaustausch beschrieben werden.

Grundlage für ein CPA sind die beiden CPPs der Geschäftspartner. Ein CPA wird von den Geschäftspartnern manuell erzeugt. Dieser Prozess soll zukünftig durch spezielle Tools vereinfacht werden.

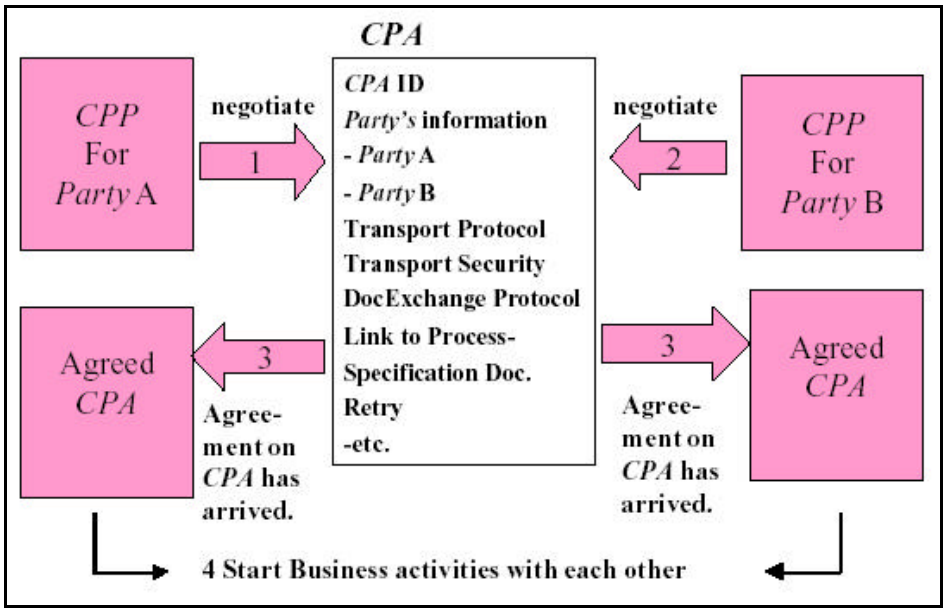

Abbildung 12: Collaboration Protocol Agreement (CPA)

Business Process Specification Schema (BPSS):

Jedes CPP und CPA ist mit einem BPSS verknüpft. In einem BPSS werden die Geschäftsprozesse der Unternehmen beschrieben.

4.5.1.3 Die ebXML-Registry

Die ebXML-Registry ist die zentrale Registrierungsstelle der ebXML-Community. Hier werden sämtliche ebXML-Dokumente der Unternehmen gespeichert. Dies ist also eine Plattform, in der sich Unternehmen über andere Unternehmen informieren und evtl. eine Partnerschaft via ebXML eingehen können.

Diverse Realisierungen von ebXML-Registries existieren bereits:

- Korea (http://www.xeni.co.kr/services/formSearchContent.jsp)
- Taiwan (http://www.xml.org.tw)

4.5.1.4 Beispielszenario

Um einen Überblick über ebXML zu gewinnen, wird im Folgenden ein Beispielszenario betrachtet, in dem die Unternehmen A und B miteinander eine Geschäftsbeziehung

eingehen. Zu Beginn dieses Szenarios benutzt Unternehmen B bereits ein ebXML-konformes System, während Unternehmen A noch nicht über ein derartiges System verfügt.

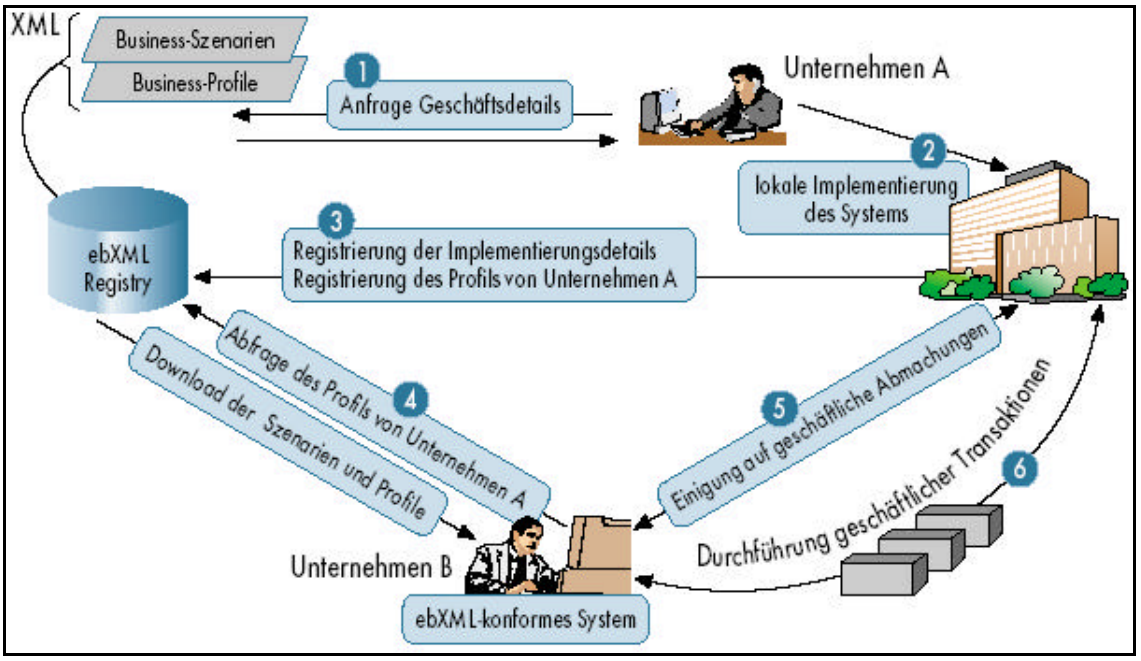

Abbildung 13: Beispielszenario

1. Unternehmen A verfügt noch nicht über ein ebXML-konformes System, möchte aber zukünftig ein solches System nutzen. Daher fragt Unternehmen A in einer ebXML-Registry die nötigen Informationen (Implementierungsdetails, ebXML-Spezifikationen) ab.

2. Unternehmen A plant und implementiert eine eigene ebXML-konforme Anwendung. Dabei ist es nicht unbedingt notwendig, eine Individualanwendung zu erstellen. Vielmehr ist zu erwarten, dass es in Zukunft auch fertige ebXML-Anwendungen von Drittherstellern gibt, die ein Unternehmen nur für die eigenen Belange konfigurieren muss.

3. Um selbst von anderen Unternehmen gefunden werden zu können, muss Unternehmen A ein Unternehmensprofil (CPP) erstellen und in der ebXML-Registry ablegen.

4. Unternehmen B sucht einen geeigneten Geschäftspartner. Es durchsucht daher die ebXML-Registry nach Unternehmen, die die gesuchten Geschäftsprozesse unterstützen. Dabei findet es zum Beispiel Unternehmen A und fordert dessen CPP an. Dieses Profil enthält alle Daten, die Unternehmen B braucht, um die Geschäftsinteressen von Unternehmen A sowie die dafür notwendigen Protokolle feststellen zu können.

5. Unternehmen B sendet nun eine Nachricht an Unternehmen A, in dem es sein Interesse an einer Geschäftsbeziehung mitteilt. Falls auch Unternehmen A an einer Geschäftsbeziehung interessiert ist, muss ein Vertrag (CPA) zwischen den beiden Unternehmen festgelegt werden.

6. Der eigentliche Austausch von ebXML-Geschäftsnachrichten kann beginnen.

4.5.1.5 Beschreibung von Geschäftsprozessen

Geschäftsprozesse werden im Rahmen von ebXML durch das Business Process and Information Model (BPIM) beschrieben. Über spezielle Modellierungs-Tools können hier die Geschäftsregeln definiert und erstellt werden. Die Modellspezifikationen stellen somit sicher,

dass jede Geschäftsregel von jedem Teilnehmer verwendet und vor allem verstanden werden kann. Die Modellierung erfolgt in UML, anschließend werden die Modelle in XML-Syntax transferiert.

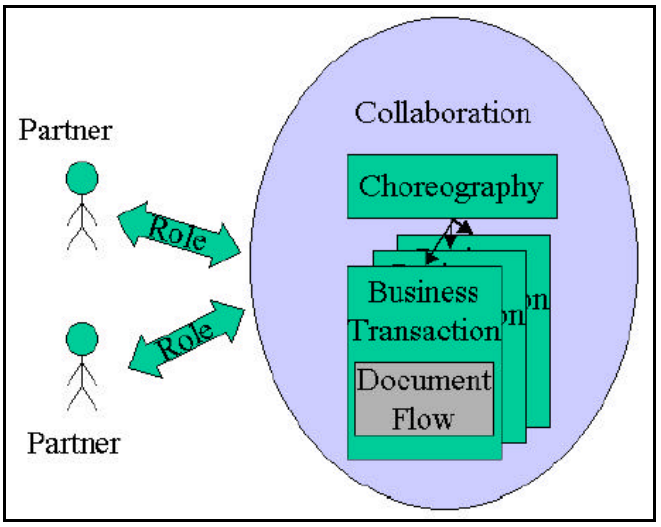

Abbildung 14: Struktur von Geschäftstransaktionen

Die Struktur von Geschäftstransaktionen in der ebXML-Welt wird in Abbildung 14 dargestellt. Geschäftstransaktionen (business transactions) sind mit einem Dokumentenfluss (document flow) verknüpft und werden durch eine Choreographie (choreography) in ihrer Abfolge gesteuert. Die Choreographie wird durch eine sog. Collaboration zusammengefasst und von den Geschäftspartnern als solche referenziert.

Ein konkreter Ablauf könnte wie folgt aussehen: Zwei oder mehr Geschäftspartner nehmen über Rollen an einer Collaboration teil. Beispiele für Rollen sind „Rechnungssteller" oder „Rechnungsempfänger". Sie geben die Position wieder, die ein Geschäftspartner innerhalb einer Collaboration einnimmt. Der Datenaustausch erfolgt dabei über Dokumentenflüsse im Rahmen von Geschäftstransaktionen. Durch Choreographien wird schließlich die Reihenfolge, in der diese Geschäftstransaktionen auszuführen sind, definiert. Die einzelnen Begriffe werden im Folgenden nochmals näher erläutert:

Collaboration:

Eine Collaboration bildet eine Menge von Geschäftstransaktionen und dadurch einen unternehmensübergreifenden Geschäftsprozess ab. Jedem Geschäftspartner wird mindestens eine Rolle in einer Collaboration zugeordnet. Es gibt zwei Arten von Collaborations:

- Binary Collaborations: es nehmen zwei Geschäftspartner teil.
- Multiparty Collaborations: es nehmen mehr als zwei Geschäftspartner teil. Multiparty Collaborations werden aus Binary Collaborations „zusammengesetzt".

Geschäftstransaktion:

Eine Geschäftstransaktion ist atomar, d. h. sie kann nicht mehr in Unter-Geschäftstransaktionen unterteilt werden. Sie wird zwischen zwei Partnern mit gegensätzlichen Rollen (anfordernde und antwortende Rolle) ausgeführt. Ein weiteres

Merkmal von Geschäftstransaktionen ist, dass sie nur als Ganzes entweder erfolgreich sein oder fehlschlagen können.

Dokumentenfluss:

Ein Dokumentenfluss ist die Realisierung einer Transaktion in Form von Dokumenten, die die Geschäftspartner untereinander austauschen. Dabei gibt es immer ein „Request"-Dokument und optional, falls eine Antwort notwendig ist, auch ein „Response"-Dokument.

Choreographie:

Die Choreographie legt fest, in welcher Reihenfolge Geschäftstransaktionen innerhalb einer Binary Collaboration ausgeführt werden sollen. In der UML-Ansicht von ebXML-Geschäftsprozessen wird die Choreographie als Aktivitätsdiagramm dargestellt.

4.5.1.6 Messaging-Service

Der Messaging-Service stellt im ebXML-Rahmenwerk die Komponente dar, die sich um Transport, Routing & Packaging von Geschäftsdaten kümmert.

Im Einzelnen bietet der Messaging-Service folgende Funktionen:

- Eindeutige Identifikation der Nachricht
- Bereitstellung von Absender- und Empfängerangaben
- Angabe von Routing-Informationen, d.h. des nächsten ebXML-Message Service Handlers
- Signatur der Nachricht zur eindeutigen Identifikation des Absenders und zur Sicherstellung der Unversehrtheit der Nachricht
- Verschlüsselung der Nachrichten
- Empfangsbestätigung
- Fehlerbenachrichtigung

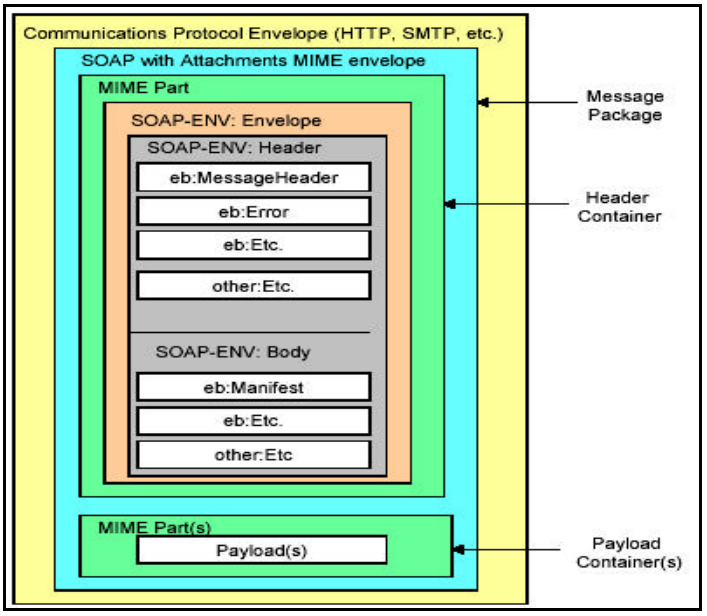

Abbildung 15: Struktur einer ebXML-Nachricht

Für die Verpackung der Geschäftsdaten wird eine Erweiterung von SOAP 1.1 (Simple Object Access Protocol), welches ebenfalls auf XML basiert, verwendet (siehe Abbildung 15).

Im SOAP Envelope sind lediglich Informationen enthalten, die die Empfänger-Schnittstelle benötigt, um die Nachricht im Rahmen eines Geschäftsprozesses zu identifizieren. Die eigentlichen Geschäftsdaten befinden sich in einem Attachment. Als Attachment kann neben XML jedes beliebige Format gewählt werden, so dass Geschäftsdaten in beliebigen Austauschformaten übertragen werden können.

Für die Übertragung der Daten werden die üblichen Übertragungsprotokolle (HTTP, SMTP etc.) verwendet.

4.5.2 ebXML – Einsatz bei Com42Bill

Die Wahl auf den Einsatz von ebXML fiel aus mehreren Gründen. Zunächst einmal handelt es sich um ein XML-basiertes Framework, welches lediglich den Rahmen spezifiziert, in dem die Daten übertragen werden. EbXML werden zur Zeit hohe Durchsetzungschancen vorausgesagt. Da es sich noch nicht um einen allgemein akzeptierten Standard handelt, ist es für ein internes Forschungsprojekt besonders interessant, sich mit dieser neuen Technologie auseinander zusetzen.

Eigene Recherchen haben ergeben, dass im Sektor der Finanzdienstleister lediglich Datenträgeraustausch (DTAUS) und EDIFACT eingesetzt wird, woran sich in naher Zukunft auch nichts ändern dürfte. Eine interne Diskussion hat ergeben, dass eine Implementierung einer EDIFACT-Schnittstelle vom Aufwand den Rahmen eines internen Forschungsprojekts sprengen würde. Wie bereits in der Seminarphase im Vortrag „SGML-basierter Datenaustausch“ erläutert, gelten individuelle EDIFACT-Anbindungen als sehr teuer und schwer zu handhaben, wodurch sie für viele Rechnungssteller nicht in Frage kommen würden. Aus diesen Gründen wurde beschlossen, zunächst nur einen Austausch auf Basis von ebXML zu realisieren und eine EDIFACT-Anbindung in der Architektur zunächst nur vorzusehen.

4.5.2.1 Die Schnittstellen zu den Geschäftspartnern

Aus den o. g. Gründen ließen sich keine realen Geschäftspartner mit einer ebXML-Schnittstelle finden. Um die Schnittstelle des Datenkonverters testen zu können, wurden daher sowohl für den Rechnungssteller als auch für den Finanzdienstleister Dummy-Schnittstellen implementiert. Für jeden Partner wurden die entsprechenden CPPs und daraus hervorgehend die CPAs zwischen Com42Bill und einem fiktiven Rechnungssteller sowie Com42Bill und einem fiktiven Finanzdienstleister erzeugt.

4.5.2.2 Die unterstützten Geschäftsprozesse

Com42Bill unterstützt die folgenden Geschäftsprozesse, die in einem BPSS beschrieben sind.

Zwischen Com42Bill und dem Rechnungssteller wird ein Geschäftsprozess ausgeführt, der den Austausch von Rechnungsdaten beschreibt. Hierbei werden mehrere einzelne Geschäftstransaktionen durchgeführt: Der Rechnungssteller sendet Rechnungen oder Stornos zu bereits existierenden Rechnungen. Com42Bill speist diese Daten in sein System ein und sendet dem Rechnungssteller zu jeder empfangenen Rechnung bzw. Storno eine Statusnachricht. Diese Statusnachricht gibt an, ob die empfangenen Daten korrekt verarbeitet wurden oder ob Fehler bei dem Datenimport Fehler aufgetreten sind. Wenn ein Rechnungsempfänger schließlich eine Rechnung bezahlt hat, informiert der Rechnungssteller nach dem Eingang der Zahlung Com42Bill durch eine entsprechende Nachricht. Diese Nachricht wird ebenfalls durch eine Statusnachricht von Com42Bill quittiert. Aufgrund dieser

Information vom Rechnungssteller kann der Status einer bezahlten Rechnung bei Com42Bill entsprechend gesetzt werden.

Der Geschäftsprozess, der zwischen Com42Bill und dem Finanzdienstleister durchgeführt wird, beschreibt den Überweisungsvorgang. Com42Bill sendet Überweisungen zum Finanzdienstleister. Falls die Überweisungsdaten fehlerhafte Informationen enthalten, wird vom Finanzdienstleister eine Fehlernachricht gesendet.

4.5.2.3 Die unterstützten Datenaustauschformate

Die Geschäftsdaten werden bei den Dokumentflüssen jeder Geschäftstransaktion in XML übertragen. Hierzu existiert zu jedem Dokumentfluss ein XML-Schema, welches den Aufbau der Daten vorgibt.

Die Erweiterung auf andere Datenaustauschformate ist möglich. Zum einen ist die Architektur des Datenkonverters flexibel gestaltet, so dass andere Formate ohne großen Aufwand integriert werden können. Andererseits ist es auch denkbar, durch einen weiteren vorgeschalteten Datenkonverter, die Daten zwischen dem bereits bestehenden XML-Format und einem andern Format zu konvertieren.

4.5.3 Architektur

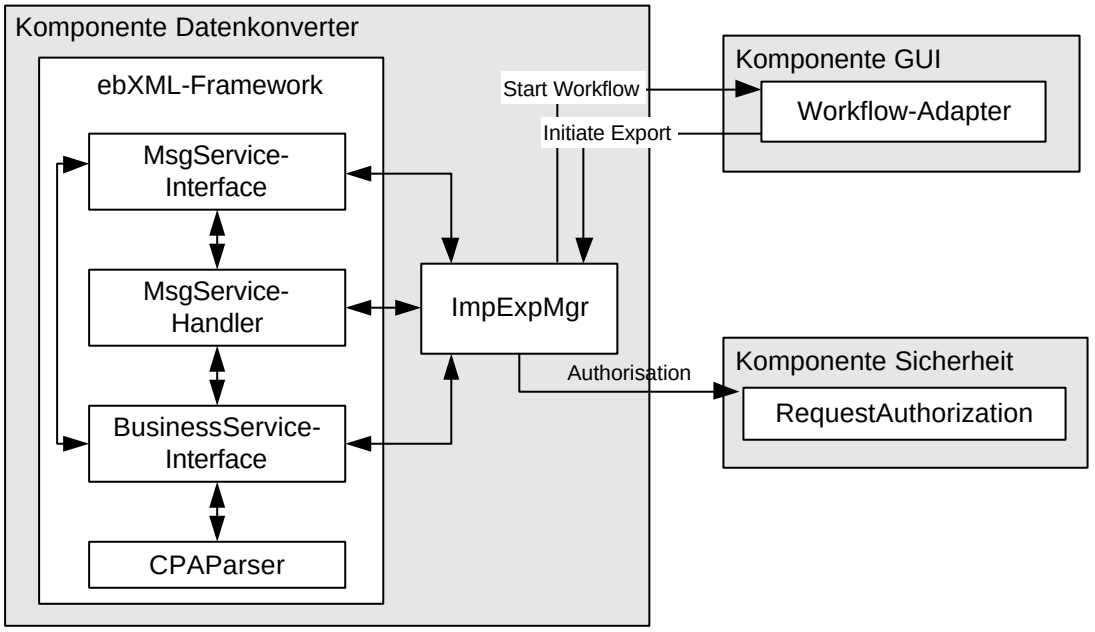

Abbildung 16: Teilarchitektur der Komponente Datenkonverter

Der Datenkonverter besteht aus den folgenden Subkomponenten (siehe Abbildung 16).

4.5.3.1 MsgServiceInterface

Funktion: Das MsgServiceInterface ist die Schnittstelle für externe ebXML-Applikationen, die mit Com42Bill elektronischen Datenaustausch betreiben möchten. Hier werden ebXML-Nachrichten über den MsgReceiver empfangen und über den MsgSender versendet.

Technologie: Hier wurde JAXM 1.1 (Java API for XML Messaging) eingesetzt. Diese API stellt Funktionalitäten zum Generieren und Senden von SOAP-Nachrichten bereit.

Der MsgReceiver ist ein JAXM-Servlet. Dieses verwendet einen Listener, der über einen HTTP-SOAP-Channel SOAP-Nachrichten empfangen kann.

Der MsgSender ist ein stateless Session Bean und benutzt ebenfalls einen HTTP-SOAP-Channel, um SOAP-Nachrichten zu versenden.

4.5.3.2 MsgServiceHandler

Funktion: Hier werden SOAP-Nachrichten verarbeitet.

Eingehende Nachrichten werden entpackt und in ihre elementaren Bestandteile zerlegt (Body, Header, Attachment). Aus den einzelnen Bestandteilen werden Informationen, die für den auszuführenden Workflow relevant sind, extrahiert. Das sind z. B. Daten, die zur Authentifizierung benötigt werden, sowie Informationen über den Typ der eingegangenen Nachricht.

Umgekehrt werden hier auch ausgehende Nachrichten zusammengesetzt, bevor sie über den MsgSender versendet werden.

Technologien: Hier kamen ebenfalls JAXM 1.1 sowie stateful und stateless Session Beans zum Einsatz.

4.5.3.3 BusinessServiceInterface

Funktion: Hier wird die Ausführung der ebXML-Geschäftsprozesse kontrolliert.

Bei eingehenden Nachrichten werden abhängig vom Typ der Nachricht und des enthaltenen Attachments (Business Document) die Geschäftsdaten aus dem XML Attachment extrahiert und konvertiert und anschließend der entsprechende Workflow bei der Business Logic gestartet.

Beim Export von Geschäftsdaten wird entsprechend der zu startende ebXML-Geschäftsprozess initiiert. Zu sendende Daten werden anschließend in ein entsprechendes XML-Attachment konvertiert.

Technologien: Zum Einsatz kamen Castor 0.3.9.21 (Open Source Data Binding Framework for Java), JAXM 1.1 sowie stateless Session Beans.

4.5.3.4 CPAParser

Funktion: Das CPA, welches als XML-Dokument vorliegt und die relevanten Informationen für die Ausführung eines ebXML-Geschäftsprozesses enthält, wird geparst. Die benötigten Daten werden ausgelesen und an anderer Stelle weiter verwendet.

Technologie: Als zentrale Technologie wurde DOM (Document Object Model) eingesetzt, um XML in Java zu konvertieren.

4.5.3.5 ImpExpMgr

Funktion: Der ImpExpMgr ist die zentrale Schnittstelle zu den anderen Komponenten des Systems. Die Authentifizierung und Authorisierung bei der Sicherheitskomponente sowie der Im- und Export von Daten in das System erfolgt ausschließlich über diese Schnittstelle.

Technologie: Auch diese Schnittstelle wurde als stateless Session Bean implementiert.

4.6 Komponente Datenbank

Die Komponente Datenbank besteht aus zwei Subkomponenten: Einem Datenbankmanagementsystem (DBMS) und einem EJB-Container, über den die Zugriffe auf die Datenbank stattfinden.

Die Aufgabe des DBMS besteht in der Speicherung der Daten, die die Arbeitsgrundlage für alle Komponenten des Softwaresystems Com42Bill darstellen. Außer dem EJB-Container hat keine andere Komponente direkten Zugriff auf die Datenbank. Grundsätzlich werden zwei Arten von Daten unterschieden: Globale Daten des Systems, die allen beteiligten Komponenten zur Verfügung stehen sollen, sowie Daten, die dem exklusiven Zugriff einzelner Komponenten des Systems unterliegen. Für alle gilt der Grundsatz der zentralen Datenhaltung. So wird eine komfortable Administration möglich; einfache Datensicherung durch zentrale Backups ist eine weitere Stärke, die hierdurch erreicht wird.

Da dem Datenbankmanagementsystem eine relationale Datenbank zugrunde liegt, alle Komponenten jedoch mit Objekten arbeiten, werden die angeforderten Daten in Entity Beans gemappt, welche die Objektrepräsentationen der einzelnen Datensätze/Datenbankviews darstellen. Andersherum wird beim Speichern von neuen Daten eine Abbildung von Objekt in Tabellenstrukturen vorgenommen. Dies bezeichnet man als Objekt-Relationales-Mapping (O/R-Mapping).

Auf das Datenbankmanagementsystem wird über eine Java Database Connectivity (JDBC) – Schnittstelle zugegriffen. Somit kann das System leicht auf eine andere Datenbank als die vom Entwicklerteam vorgesehene angepasst werden.

Die Komponente Datenbank profitiert maßgeblich vom Einsatz eines J2EE-Applicationservers. Das komplette O/R-Mapping kann mit Hilfe der Container Managed Persistence (CMP) automatisch durchgeführt werden. Des Weiteren muss man sich nicht um die Konsistenz der Daten kümmern, da sowohl die Datenbankmanagementsysteme als auch die Applikationsserver Transaktionsmonitore/-manager besitzen, welche diese Aufgabe übernehmen. Die Wahl beim Datenbankmanagementsystem fiel auf die Version 8i der Oracle – Datenbank. Diese Entscheidung ist hauptsächlich mit der Verbreitung und somit der sehr guten Unterstützung bei den Applicationservern zu begründen. Beim Applicationserver entschied man sich für das Produkt WebLogic 6.1 von der Firma Bea. Hiermit liegt eine stabile und erprobte Version zugrunde, welche sich problemlos in die meisten Entwicklungsumgebungen integrieren lässt.

Die Entity-Enterprise Java Beans wurden nach der Version 2.0 der Spezifikation implementiert. Die zwei wesentlichen Neuerungen dieser Version sind zum einen Container-Managed-Relations. Hierbei übernimmt der Container beispielsweise die Verwaltung von m:n-Relationen. Hiermit entfällt das eigenhändige Management der Zwischentabelle. Die zweite Neuerung ist die EJB-Query Language (EJB-QL). Hierbei handelt es sich um eine SQL-ähnliche Abfragesprache, mit dessen Hilfe eigene Finder-Methoden der Home-Interfaces zum ermitteln bestimmter Datensätze geschrieben werden können.

Bei der Umsetzung der Komponente bestand durch den Einsatz von EJBs kein großer Gestaltungsspielraum. Das gesamte Objektmodell wurde implementiert und enthält somit bereits einigen Spielraum für Erweiterungen, wie z. B. die Umsetzung der Länderanpassung des Systems.

Die einzelnen Bestandteile der Komponente, welche in Abbildung 17 schematisch dargestellt sind, werden abschließend erläutert.

Home-Interfaces: Die Home-Interfaces stellen die Lebenszyklusmethoden für die Entity-Beans zur Verfügung. Hiermit ist es möglich, neue Entitäten zu erzeugen, vorhandene aufzufinden und sie letztendlich auch wieder zu entfernen.

Entity-Beans: Diese Objekte stellen die Repräsentationen der Datenbankdatensätze/-views dar.

O/R-Mapping: Beim Einsatz einer relationalen Datenbank müssen die Attribute des Entity-Beans auf relationale Strukturen abgebildet werden. Diese Aufgabe übernimmt das O/R-Mapping.

Connection Pool: Die Verbindungen zwischen EJB-Applikationsserver und Datenbank sollten aus Performanzgründen in einem Pool zur Verfügung gestellt werden. In diesem Connection Pool werden geöffnete JDBC-Verbindungen gehalten, über welche der Applikationsserver mit der Datenbank kommunizieren kann. Die Implementierung erfolgt in der Regel durch den Applikationsserver.

DBMS: Das Datenbankmanagementsystem ist die klassische relationale Datenbank.

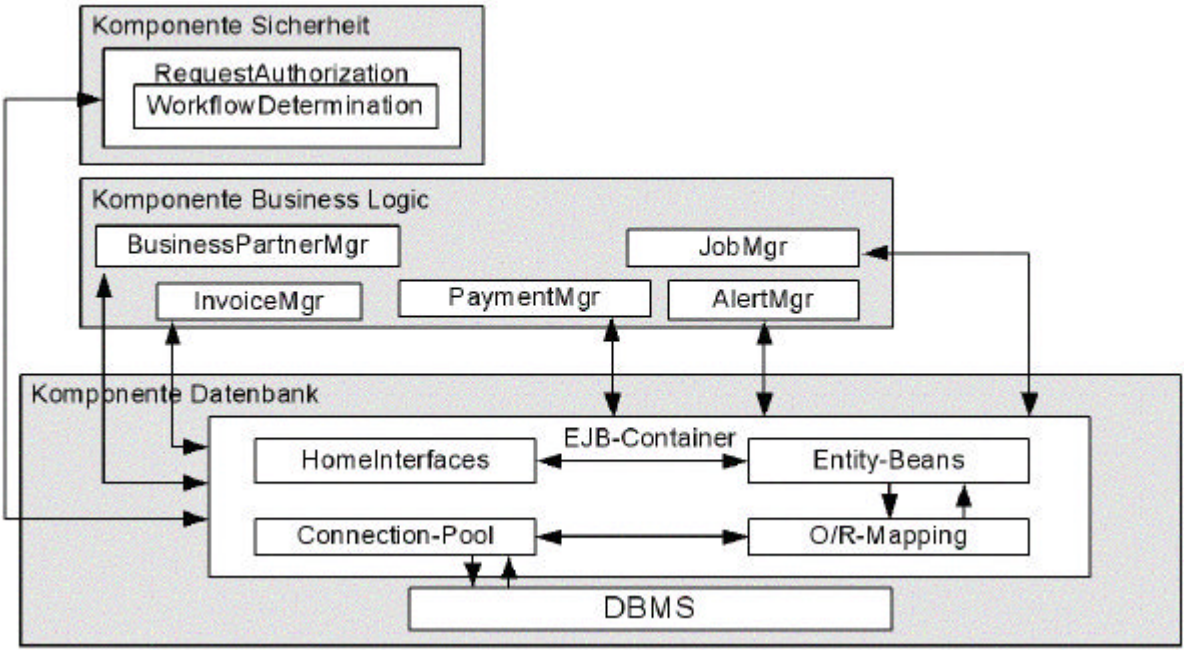

Abbildung 17: Teilarchitektur der Komponente Datenbank

5 Funktionalität

5.1 Allgemein

Alle Funktionen von Com42Bill werden über eine Web-Schnittstelle bereitgestellt. Das bedeutet, dass die Kommunikation zwischen Benutzern (Betreiber und Rechnungsempfänger) und dem Server, auf dem Com42Bill läuft, über HTML-Seiten und –Formulare stattfindet. Eine Umsetzung auf WML-Seiten für WAP-fähige Mobilgeräte ist in der Architektur vorgesehen, jedoch in dem realisierten System noch nicht umgesetzt.

5.2 Funktionen der Benutzeroberfläche

5.2.1 Begrüßung

Dem Benutzer werden auf der Begrüßungsseite kurz die Ziele von Com42Bill bekannt gemacht. Über diese Seite hat der Benutzer Zugriff auf alle anderen Elemente von Com42Bill, so kann er sich beispielsweise über das Quick-Login in der rechten Spalte sofort anmelden. Auch die aktuellen Nachrichten und die wichtigsten Punkte der FAQ sind in der rechten Spalte aufgeführt.

Abbildung 18: Begrüßungsseite

Die weiteren Seiten von Com42Bill kann der Benutzer über den Menüpunkt MyCom42Bill erreichen. Befindet sich der Mauszeiger über diesem Button, öffnet sich ein Popup-Menü. Durch einen Druck auf den Button selbst kann der Benutzer eine Übersichtseite der Funktionen von Com42Bill erreichen.

Abbildung 19: Popup-Menü

5.2.2 Aktuelles

Com42Bill hat die Möglichkeit, den Benutzer auf der Seite Aktuelles mit aktuellen Informationen zu versorgen. Hier können zum Beispiel Umstellungen oder Verbesserungen des Systems angekündigt oder bekannt gemacht werden.

Abbildung 20: Aktuelles

5.2.3 MyCom42Bill

MyCom42Bill ist eine geschützte Seite, für die der Benutzer bereits angemeldet sein muss. Sie bietet einen Überblick über die Funktionen, die in Com42Bill aufgerufen werden können. Außerdem zeigt sie einen Kurzüberblick über die Rechnungen des Benutzers in der rechten Spalte.

Abbildung 21: MyCom42Bill

5.2.4 Login

Will der Benutzer nicht das Quick-Login benutzen, so kann er auch eine Login-Seite über das Popup-Menü des Com42Bill-Buttons aufrufen. Nach einem erfolgreichen Login wird der Benutzer direkt zur Übersichtsseite MyCom42Bill weitergeleitet.

Abbildung 22: Login

5.2.5 Rechnungsübersicht

Die Rechnungsübersicht bietet dem Benutzer Kontrolle über seine Rechnungen. Mittels der Suchfunktion kann er sich nur gewünschte Rechungen anzeigen lassen. Die Übersicht zeigt den Rechnungsstatus, die Fälligkeit und den Rechnungssteller.

Abbildung 23: Rechnungsübersicht

Mittels des Buttons in der Spalte „Zahlen?“ kann er jede offene Rechnung zum Rechnungskorb hinzufügen oder aus diesem entfernen. Rechnungen im Rechnungskorb sind zur Bezahlung vorgemerkt. Den Bezahlvorgang selbst kann der Benutzer über den Button „Bezahlen“ unter den Rechnungen starten.

5.2.6 Bezahlvorgang

Der Bezahlvorgang läuft in mehreren Schritten ab. Zunächst wird dem Benutzer der Rechnungskorb gezeigt, also die Rechnungen, die zur Bezahlung vorgemerkt wurden. Diese Rechnungen können in dieser Übersicht auch noch entfernt werden, um sie nicht mit diesem Bezahlvorgang zu begleichen. Durch den Button „Zurück“ kann der Benutzer in den Schritten des Bezahlvorganges jederzeit auf die vorherige Seite gelangen bzw. den Bezahlvorgang abbrechen.

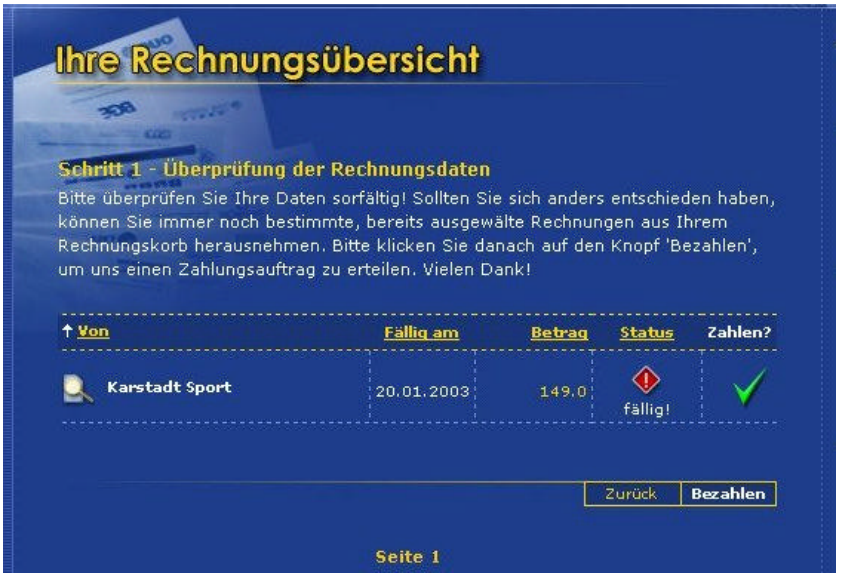

Abbildung 24:Bezahlvorgang – Schritt 1

Im nächsten Schritt wählt der Benutzer die gewünschte Bezahlungsmethode aus. Er kann unter allen Bezahlmethoden wählen, die er in Com42Bill konfiguriert hat. Zusätzlich kann er noch einen späteren Zeitpunkt für die Bezahlung durch die Datumseingabe festlegen.

Abbildung 25: Bezahlvorgang – Schritt 2

Nach der Wahl des Bezahlvorgangs erfolgt die Bestätigung der Bezahlung durch den Knopf „Bezahlen“. Der Benutzer bekommt zur Bestätigung eine entsprechende Meldung des Systems, dass sein Auftrag gemäß seiner Angaben ausgeführt wird.

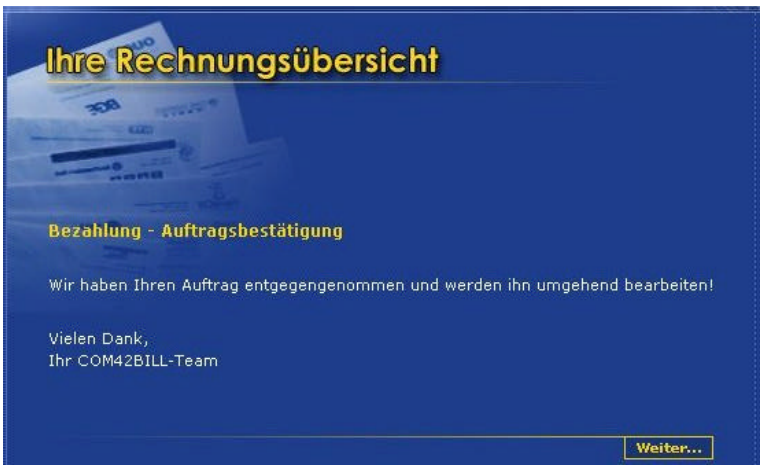

Abbildung 26: Bezahlvorgang – Bestätigung

5.2.7 Persönliche Daten

Auf dieser Seite wird dem Benutzer die Möglichkeit gegeben, seine persönlichen Daten zu ändern, also das Passwort, die Anschrift und ähnliche Informationen.

Abbildung 27: Persönliche Daten ändern

5.2.8 Zahlungsweise ändern

Hier kann der Benutzer seine Konten verwalten. Er kann neue Konten anlegen, die dann für den Bezahlvorgang zur Verfügung stehen, Konten löschen oder auch Kontoinformationen überarbeiten.

Abbildung 28: Zahlungsmethode ändern

In dem Dialog für das Anlegen bzw. Editieren der Zahlungsmethode kann der Benutzer die Art des Kontos angeben und die zugehörigen Informationen des Kreditinstituts. Vorgesehen waren unterschiedliche Arten von Konten für die Bezahlung, also zum Beispiel Kreditkarten, umgesetzt wurden im Laufe des Projekts aber vorerst nur Bankkonten.

Abbildung 29: Kontodaten ändern

5.2.9 Logout

Durch den Aufruf der Seite Logout meldet sich der Benutzer explizit ab. Seine derzeitige Anmeldung wird für ungültig erklärt und die geschützten Bereiche sind ab jetzt nicht mehr zugänglich. Für die erfolgreiche Abmeldung erhält er auf dieser Seite eine Bestätigung.

Abbildung 30: Logout

5.2.10 Registrierung

Neue Benutzer, die sich bei Com42Bill anmelden wollen, müssen sich für den Dienst registrieren. Die Registrierung läuft in mehreren Schritten ab. Zunächst wird dem Benutzer der Registriervorgang erklärt.

Abbildung 31: Registriervorgang – Erklärung

Danach wird nach den persönlichen Daten verlangt, also Name, Anschrift und Kontaktinformationen.

Abbildung 32: Registriervorgang – Persönliche Daten

Als nächstes wählt der Benutzer einen Login-Namen und ein Passwort aus. Sollte der Login-Name bereits vergeben sein, so kann er die Seite nicht mittels der Bestätigung verlassen und muss erst einen anderen Namen eingeben. Zusätzlich kann er auswählen, ob und wie er über eingehende Rechnungen informiert werden möchte.

Abbildung 33: Registriervorgang – Wahl des Logins

Im dritten Schritt muss eine Zahlmethode eingegeben werden, damit der Benutzer für eingehende Rechnungen Bezahlvorgänge abwickeln kann.

Abbildung 34: Registriervorgang – Kontoinformationen eingeben

Vor der abschließenden Bestätigung der Registrierung werden nochmals alle eingegebenen Daten zur Kontrolle angezeigt. Wenn der Benutzer Fehler entdeckt, so können diese durch ein Korrigieren in den vorhergehenden Schritten des Registriervorganges behoben werden. Diese Seiten kann der Benutzer über den Zurück-Button erreichen.

Abbildung 35: Registriervorgang - Kontrollansicht

Abschließend wird dem Benutzer die erfolgreiche Registrierung bestätigt.

5.2.11 Support

Auf der Seite Support werden dem Benutzer Kontaktinformationen für eventuell benötigte Hilfe angezeigt. Zusätzlich wird auf die FAQ-Seiten von Com42Bill verwiesen, auf denen der Benutzer eventuell bereits die Antworten zu seinen Fragen finden kann.

Abbildung 36: Support

5.2.12 Frequently Asked Questions

Auf dieser Seite werden häufig gestellte Fragen und deren Antworten präsentiert. Auf diese Weise kann der interessierte Benutzer sich selbst über häufig erfragte Funktionen und Eigenschaften von Com42Bill informieren.

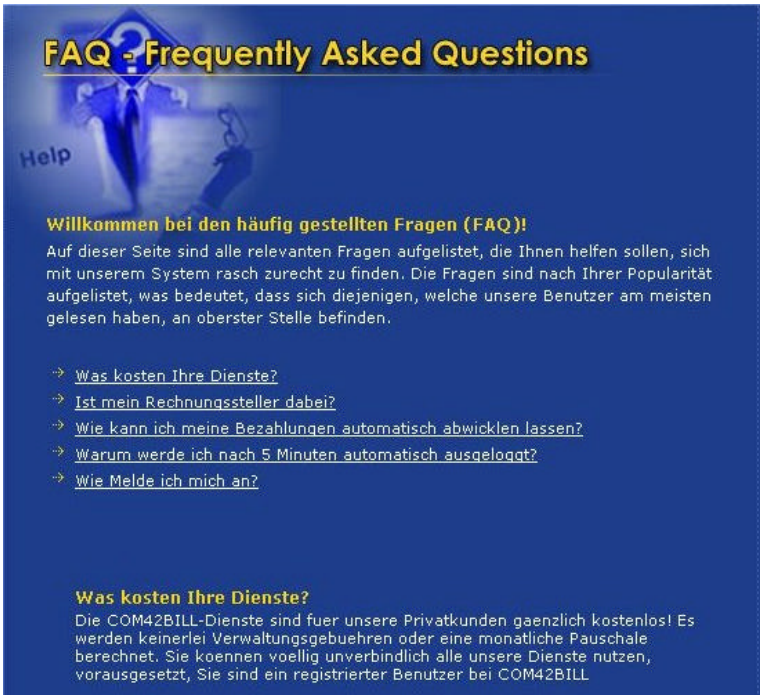

Abbildung 37: FAQ

5.2.13 Impressum / Kontakt

Das Impressum stellt in Kurzform das interne Forschungsprojekt vor. Bei einem „Echteinsatz“ würde hier das nach §6 Teledienstegesetz (TDG) geforderte Impressum stehen [TDG02].

5.3 Funktionen der Administrationsoberfläche

5.3.1 Begrüßung

Um die Administrationsfunktionen optisch deutlich von der Benutzeroberfläche zu trennen, ist die Administrationsoberfläche komplett in grün gehalten. Um dennoch einen Widererkennungseffekt zu erreichen, entspricht das Design der Seite im Wesentlichen dem der Benutzerseite. Zunächst muss sich der Administrator anmelden:

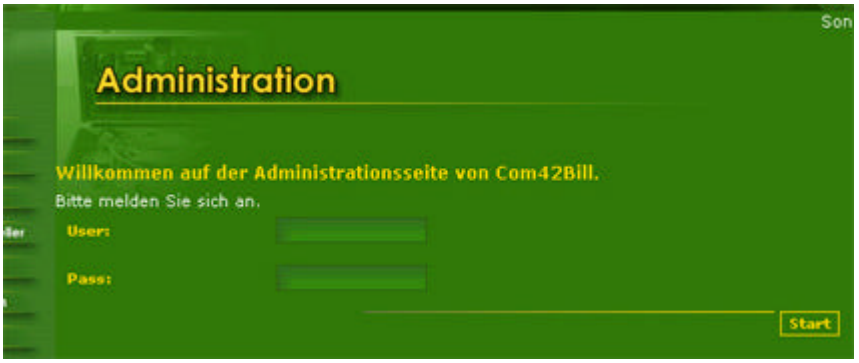

Abbildung 38: Anmeldung für den Administrator

In der vorliegenden Version gibt es nur einen administrativen Benutzer, Benutzername ist „Administrator“. Weitere administrative Benutzer mit eingeschränkten Rechten sind zwar

vorgesehen, aber nicht realisiert, da die Pflege von Benutzergruppen und –rechten nicht umgesetzt wurde.

Die Begrüßungsseite für den Administrator sieht wie folgt aus:

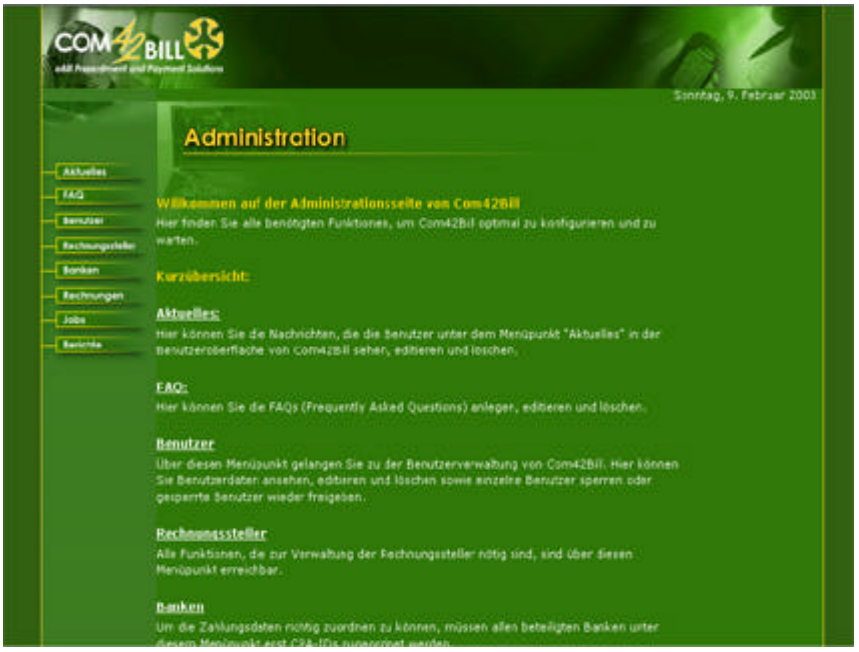

Abbildung 39: Begrüßungsseite für den Administrator

Der prinzipielle Aufbau entspricht dem der Benutzeroberfläche. Die gewünschte Aktion bzw. der zu administrierende Teilbereich wird auf der linken Seite ausgewählt. In der aktuellen Version teilt sich der Administrationsbereich in acht Teilbereiche.

5.3.2 Aktuelles

Auf dieser Seite können die Nachrichten bearbeitet werden, die dem Benutzer (Rechnungsempfänger) in der rechten Spalte der Benutzeroberfläche angezeigt werden.

Abbildung 40: Administrationsseite "Aktuelles"

Im oberen Bereich kann eine neue Nachricht eingegeben werden. Ein Klick auf „Speichern“ speichert die neue Nachricht. Im unteren Bereich werden die vorhandenen Nachrichten angezeigt. Zu jedem Artikel gibt es einen zugehörigen Link „Diesen Artikel löschen“.

5.3.3 FAQ

Der FAQ – Bereich der Benutzeroberfläche soll dem Rechnungsempfänger Hilfestellungen bei der Bedienung von Com42Bill geben. Zunächst gelangt der Administrator zu einer Übersicht aller bereits vorhandenen Artikel:

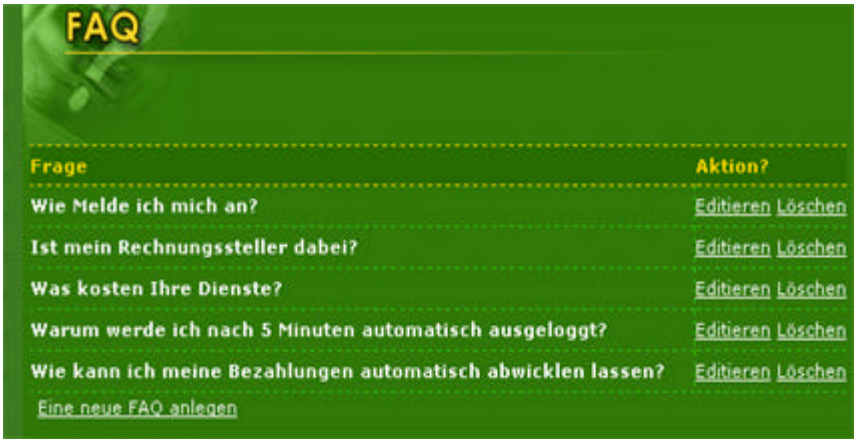

Abbildung 41: FAQ-Übersicht

Über die entsprechenden Links können die einzelnen FAQ-Artikel editiert und gelöscht werden. Über den Link „Eine neue FAQ anlegen“ gelangt man zu einem Eingabeformular für FAQs, das im Wesentlichen wie das zum Bearbeiten einer FAQ aussieht:

Abbildung 42: FAQ editieren

Wie in allen anderen Formularen auch, gelangt der Administrator durch Klick auf „Abbruch“ wieder zu der Übersicht, ein Klick auf „Speichern“ speichert die gemachten Eingaben.

5.3.4 Benutzer

Zunächst landet der Administrator auf einer Übersichtsseite, auf der der zu bearbeitende Benutzer ausgewählt werden kann:

Abbildung 43: Benutzerauswahl

Die Löschung wird direkt von dieser Seite ausgelöst. Zum Editieren gelangt der Administrator auf folgende Seite:

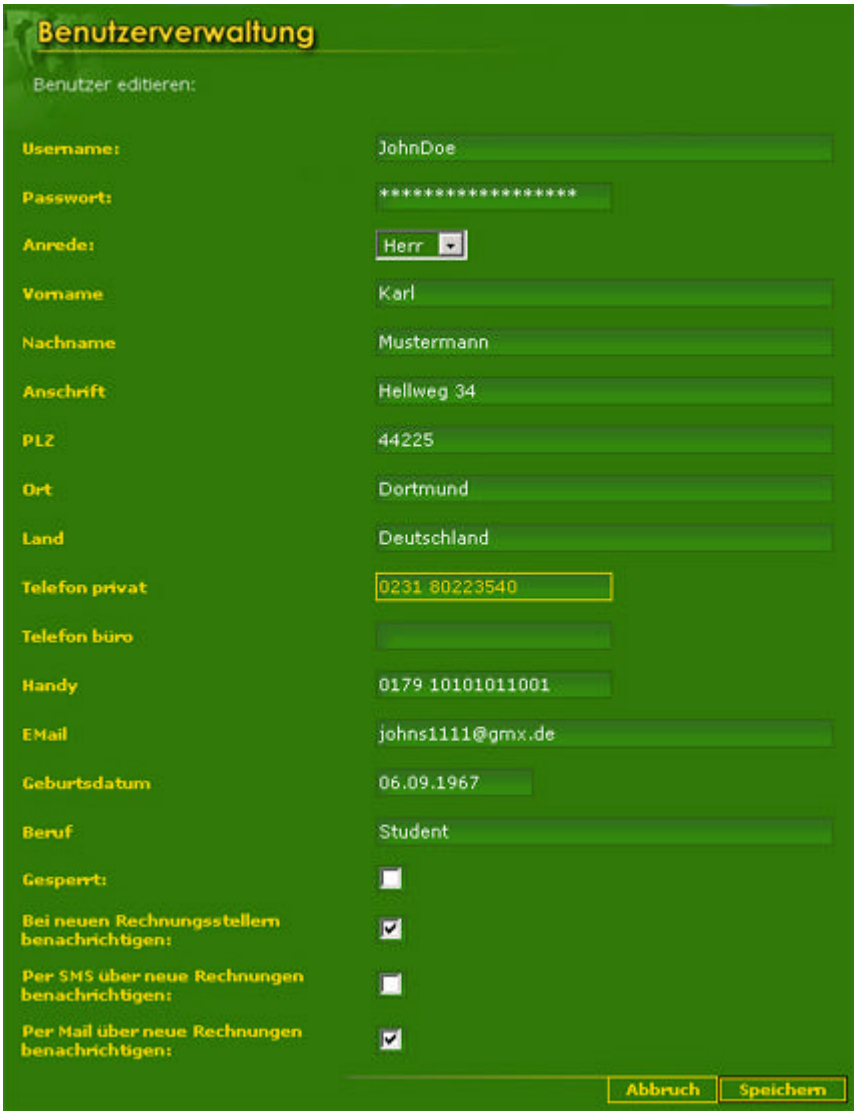

Abbildung 44: Benutzer editieren

Alle wesentlichen Daten zu dem Benutzer können hier bearbeitet werden. Ein Klick auf „Abbruch“ bringt den Administrator zurück zur Übersicht, ein Klick auf „Speichern“ speichert die vorgenommenen Änderungen. Im unteren Bereich der Seite kann ein Konto des Rechnungsempfängers ausgewählt werden:

Abbildung 45: Kontenauswahl

Die Kontodaten können auch geändert werden:

Abbildung 46: Konto editieren

Vorgesehen waren Bankkonten und Kreditkarten; realisiert wurde nur die Eingabe von Bankkonten. Die Neuanlage eines Bankkontos ist hier nicht vorgesehen; dies kann nur der Benutzer selbst.

5.3.5 Rechnungssteller

Da die Rechnungssteller, anders als die Rechnungsempfänger, keine eigene Oberfläche haben, um Ihre Daten zu ändern, gelangt man hier zu der einzigen Stelle, an der die Daten von Rechnungsempfängern bearbeitet werden können. Zunächst wird man wie bei den Benutzern auf eine Übersichtsseite geführt:

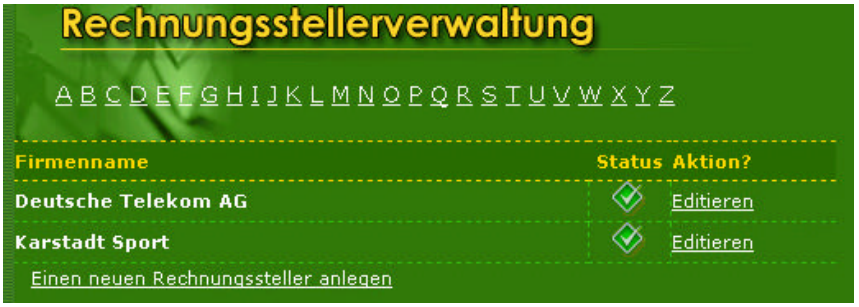

Abbildung 47: Rechnungssteller auswählen

Ein Rechnungssteller kann nicht gelöscht werden. Wird ein Rechnungssteller vom System ausgeschlossen, wird dieser gesperrt (siehe folgenden Abschnitt). Die Löschung eines Rechnungsstellers ist eine kritische Angelegenheit, da Zahlungsverkehrsdaten einige Jahre aufbewahrt werden müssen; des weiteren ist unklar, wie lange nach der letzten Rechnung noch Reklamationen möglich sein müssen. Daher haben wir auf eine endgültige Löschung eines Rechnungsstellers verzichtet.

Zu jedem angelegten Rechnungssteller gibt es einen Link „Editieren“, der zu der Bearbeitungsseite für Rechnungssteller führt:

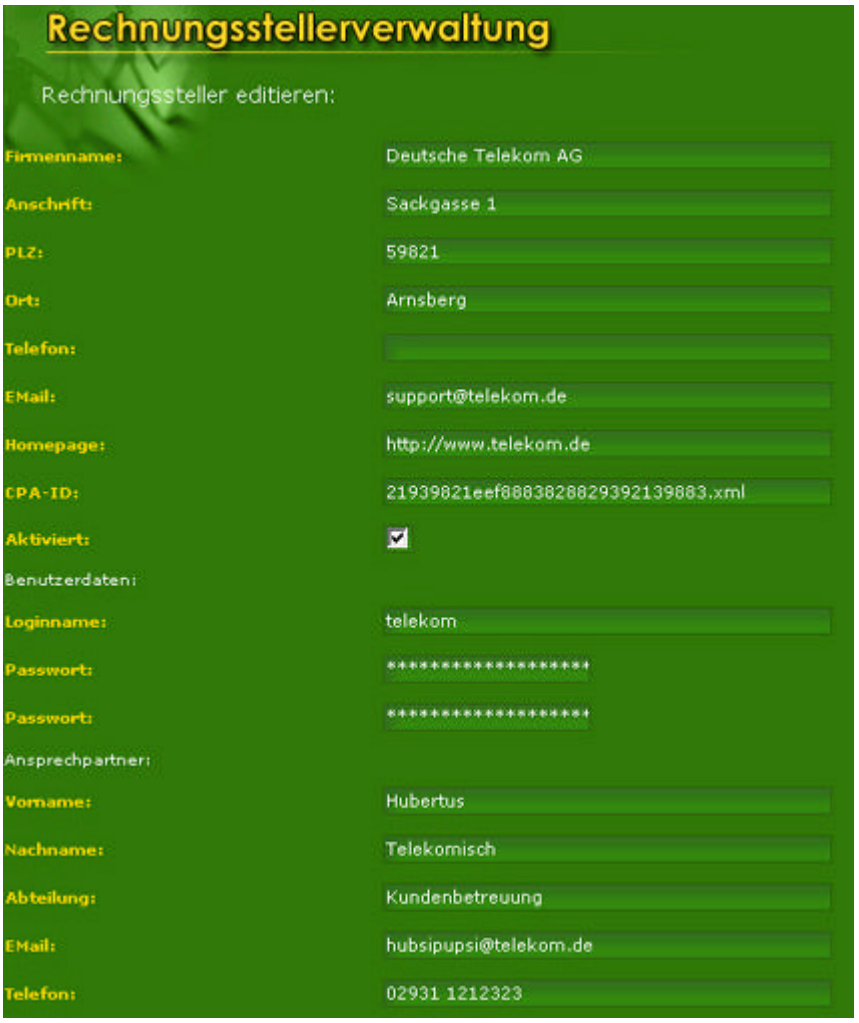

Abbildung 48: Rechnungssteller editieren

Durch die Checkbox „Aktiviert“ kann ein Rechnungssteller, wie bereits erwähnt, gesperrt werden.

Im unteren Bereich des Formulars kann ein Finanzkonto zur Bearbeitung ausgewählt werden. Dieser Bereich sieht aus wie beim Rechnungsempfänger (vgl. Abbildung 45). Auch hier können derzeit nur Bankkonten eingegeben werden (das Formular zur Neuerfassung sieht genauso aus; daher wird auf eine Abbildung verzichtet).

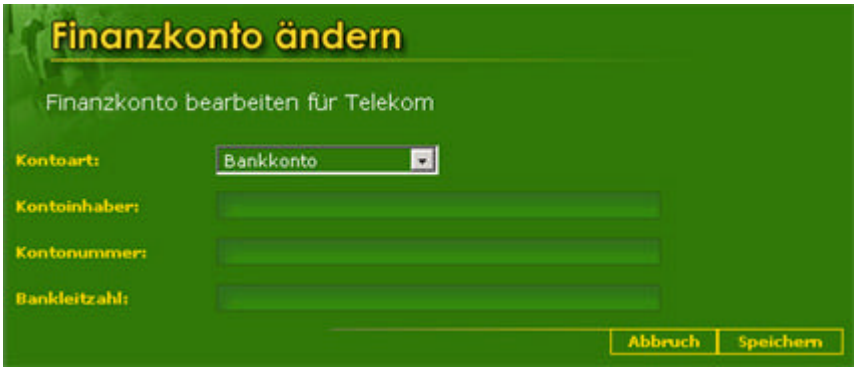

Abbildung 49: Finanzkonto editieren

5.3.6 Banken

Die Daten aller Banken in Deutschland wurden in die Datenbank importiert. Ein Ablauf zur Pflege der Bankdaten wurde nicht implementiert, da die Änderungen von der Deutschen Bank in einem festgelegten Format herausgegeben werden und daher leicht importiert werden können.

5.3.7 Rechnungen

Da der Betreiber eines Systems, das hauptsächlich auf Kundenaktionen basiert, in jede Aktion eingreifen können muss, gibt es einen Bereich zur Administration der Rechnungen. Diese können über die folgende Maske gesucht werden:

Abbildung 50: Rechnungen suchen

Weitergehende Funktionen sind hier nicht vorhanden. Bei einem Echteinsatz muss der Betreiber an dieser Stelle die Möglichkeit haben, in Rechnungsvorgänge einzugreifen bzw. sich weitere systeminterne Details zu Zahlungsvorgängen anzusehen.

5.3.8 Jobs

Hier können regelmäßig auszuführende Arbeitsabläufe angelegt und bearbeitet werden. Der prinzipielle Aufbau ist wie bei der Benutzerkonfiguration oder der Rechnungsstellerkonfiguration:

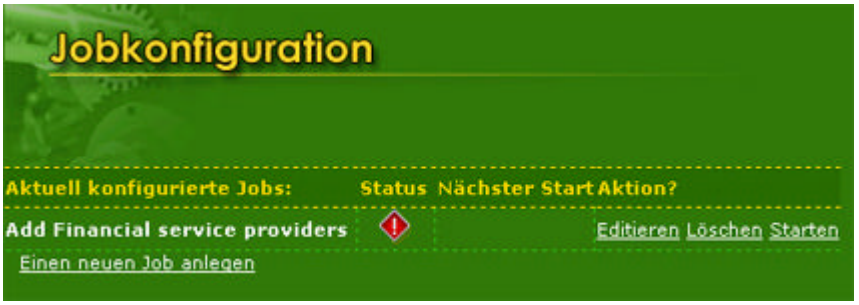

Abbildung 51: Jobübersicht

Hier kann ein Job sofort gestartet werden (für wiederkehrende Jobs, die jedoch nicht regelmäßig ausgeführt werden). Diese Funktion ist bislang lediglich vorgesehen; ein Sofortstart von Jobs ist derzeit nicht möglich. Außerdem kann der Job hier gelöscht werden. Ein Klick auf „Editieren“ bringt den Administrator zu folgendem Formular:

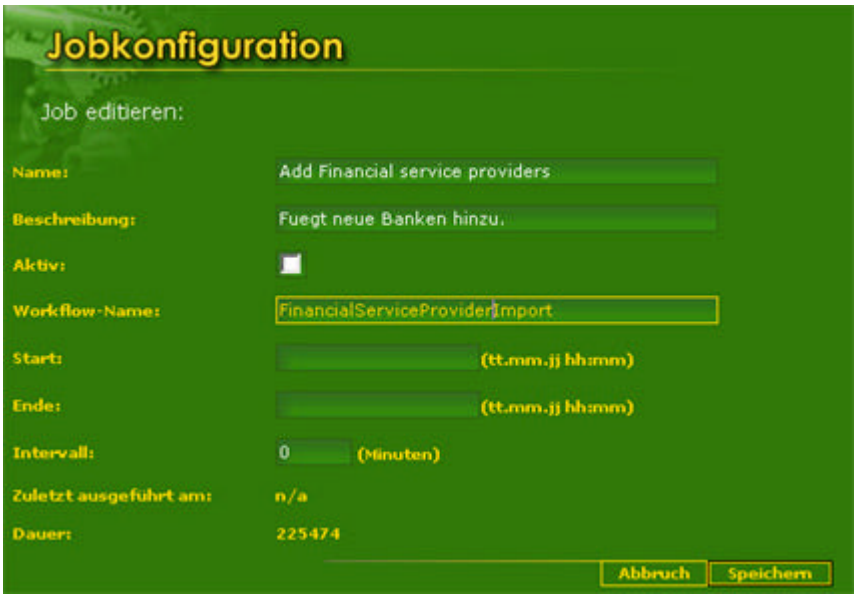

Abbildung 52: Job konfigurieren

Alle Angaben, die für einen Job relevant sind, werden hier erfasst. Des Weiteren können auf dieser Seite Zusatzinformationen eingesehen werden, hier z.B. das Datum der letzten Ausführung sowie die Dauer der letzten Ausführung.

6 Zusammenfassung und Schluss

Während der Projektlaufzeit hat sich weiter herausgestellt, dass das Thema Electronic Bill Presentment und Payment sowohl in der Forschung als auch in der Wirtschaft immer noch aktuell ist.

Es ist daher weiter geplant, die Ergebnisse dieses internen Forschungsprojekts mit Partnern aus der Wirtschaft (Versicherungs-, Finanzdienstleister) zu diskutieren und Anknüpfungspunkte für weitere – auch kooperative – Arbeiten zu finden.

Ein besonderer Anknüpfungspunkt ergibt sich durch die Nutzung von Telematiktechniken für mobile Endgeräten – deren technische Möglichkeiten immer besser werden (Farbdisplay, Java, etc.).

7 Literaturverzeichnis

[APA01] Apache Jakarta Project (2001): **BeanUtils,** <http://jakarta.apache.org/commons/beanutils.html>, 15.12.2002

[As02] Ausarbeitungen zum Seminar **Application Server (2002),** Universität Dortmund 2002, <www.ls10.de>

[BPM00] BPMI.org (2000): **The Business Process Management Initiative**, <www.bpmi.org>, 20.07.2002

[Bei01] Beijing, Cambridge, Farnham, et al. (2001): **XML Pocket Reference**, O'Reilly, 2001

[BMI97] Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie und Bundesministerium für Gesundheit (1997): **Telematik im Gesundheitswesen – Perspektiven der Telemedizin in Deutschland**, München, August 1997

[Boe86] B. W. Boehm (1986): **A Spiral Model of Software Development and Enhancement**, In: Software Engineering Notes (1986) 11, 22-42

[Bro02] Bronstein, Ilja N. u.a. (2000): **Taschenbuch der Mathematik**, Verlag Harry Deutsch AG, Thun, 2000

[C42B02_2] Zwischenbericht der Projektgruppe 411 „Com42Bill“, interner Bericht, Universität Dortmund - Lehrstuhl für Software-Technologie, 2002

[C42B02] Webseite des Projekts Com42Bill; z.Zt. sind die Ausarbeitungen zu dem Seminar nur über die Autoren bzw. die Projektgruppe erhältlich. Kontaktadressen siehe www.Com42Bill.de

[Che01] T. M. Chester (2001): Cross-Platform Integration with XML and SOAP, **IT Pro**, September / Oktober 2001, S. 26-34

[Ebx02] ebXML.org (2002): **Enabling a global electronic market**, <www.ebxml.org>, (15.04.2002)

[Geo01] B. Georg (2001): **Begriffswirrwarr beim Datenaustausch,** e-commerce magazin (2001), August, S. 40-42

[Has02] Hasanbegovic, D., Schmitz, A. (2002): **E-Bill Anwendungen – Systeme, Produkte, Anbieter,** PG-Seminar PG411, Dortmund / Nordhelle 2002

[Kah01] B. Kahlbrandt (2001): **Skript zur Vorlesung Software-Engineering II**, 16.12.2001, <http://www.informatik.fh-hamburg.de/~khb/st4se2/st4se2.html> (17.03.2002)

[Kru98] P. Kruchten (1998): **The Rational Unified Process – An Introduction**, 1998, Addison-Wesley

[IESE02] Fraunhofer Institute of Experimental Software Engineering (IESE) (2002): **Vorgehensmodell (V-Modell)**, <http://www.iese.fhg.de/VModell/Intro/vm.introMain.html> (18.03.2002)

[ISO00] **EN ISO 8402 / EN ISO 9000:2000 / EN ISO 9001:2000 / EN ISO 9004:2000** <www.din.de, www.iso.ch>

[Jec01] M. Jeckle (2001): **XML Tutorial**, <http://www.jeckle.de/vorlesung/xml/script.html>, 22.02.2002

[Jep01] T. Jepsen (2001): **SOAP Cleans up Interoperability Problems on the Web**, IT Pro, Januar / Februar 2001, S. 52-55

[Kie01] Kieser, M. (2001): Mobile Payment – Vergleich elektronischer Zahlungssysteme, **HMD 220**, 27-36

[Kla01] A. Klafs (28.11.2001): **Electronic Payment**, <http://www.systor.com/dl_know_campus_vorl_ecommerce_e_payment.pdf> (04.03.2002)

[Kro96] Kron,Thomas (1996): **Secure Electronic Transaction,** <www.vsb.cs.uni-frankfurt.de/lehre/WS_96-97/kron> (05.03.2002)

[LBE01] Landesverband des Bayerischen Einzelhandels e.V.(2001): **Positionspapier Telematik im Verkehr**, München, Januar 2001

[Mül02] Müller, D., Schlich, B. (2002): **Softwareentwicklungsprozesse**, PG-Seminar PG411, Dortmund / Nordhelle 2002

[Mül99] G. Müller-Ettrich (1999): **Objektorientierte Prozessmodelle – UML einsetzten mit OOIC, V-Modell, Objectory**, Addison-Wesley

[Mus02] Mustafa, N., Oberweis, A., Schnurr, T. (2002): Mobile Banking und Sicherheit im Mobile Commerce. In Silberer, G., Wohlfahrt, J., Wilhelm, T. (Hrsg.) (2002): **Mobile Commerce – Grundlagen, Geschäftsmodelle, Erfolgsfaktoren.** Gabler Verlag.

[Obe98] Oberschelp, Walter (1998): **Rechneraufbau und Rechnerstrukturen**, 7. Auflage, München; Wien: Oldenbourg, 1998

[Ogb00] U. Ogbuji (2000): **Using WSDL in SOAP-Applications**, <http://www-4.ibm.com/spftware/developer/library/wssoap/index.html> (03.03.2002)

[OM01] P. O'Connell und R. McCrindle (2001**): Using SOAP to Clean up Configuration Management**, Institute of Electrical and Electronics Engineers, 0-7695-1372-7/01, S. 555-560

[Pay01] Paybox (2001): **Zahlungsdienst per Handy**, <www.paybox.de>, (20.03.2002)

[Pe01] Mathias Peick (2001): **Mut zur Lücke – Erfahrungen mit Versionsmanagement**, iX 2001, Heft 1, S.91ff.

[RSA02] RSA Security Inc. (2002): **RSA Laboratories FAQ about todays Cryptography**, Version 4.1, <www.rsasecurity.com/rsalabs/faq/index.html>

[Sch00] Oliver Schade (2000): **Kontroletti – Werkzeuge zur Versionsverwaltung**, iX 2000, Heft 9, S.102ff.

[So02] Solomon To, Ray Plant (2002): **EBPP – Is this the end of the paper bill?** <http://www.fujixerox.com.au/business_solutions/finan_ser.jsp> (02.04.2002)

[SQI02] Software Quality Institute (SQI) (2002): **An Introduction to the Documents**, <http://www.sqi.gu.edu.au/spice/suite/intro.html> (19.03.2002)

[Sta01] G. Starke (2001): Datenaustausch im Web, **IT FOKUS**, Heft 7, Juni 2001, S. 72-75

[Ste00] Dirk Stelzer (2000), **Qualitätsmanagement in der Softwareentwicklung**, Computer Reseller News Nr. 13/2000, 13.März 2000

[TDG02] Teledienstgesetz (TDG) in der Fassung vom 1.1.2002, §6: **Allgemeine Informationspflichten**

[Udd00] Uddi.org (2000), **UDDI Technical White Paper**, <http://www.uddi.org> (15.03.2002)

[Wel99a] J. D. Wells (1999): **The XP Philosophy**, <http://www.extremeprogramming.org/Kent.html> (13.03.2002)

[Whb01] T. Weitzel, T. Harder, P. Buxmann (2001): **Electronic Business und EDI mit XML**, dpunkt-Verlag, 2001

[Wie02] T. Wieland (2002): **Werkzeuge für die Softwareentwicklung**, <http://www.cpp-entwicklung.de/cpplinux/cpp_main/node8.html> (3.3.2002)

[Wuv99] **Werben & Verkaufen** (1999): Fachzeitschrift zu Werbung, Kommunikation und Marketing, Juni 1999 <http://www.wuv.de/servlet/wuv/community/umfrage_archiv.html> (12.03.2002)

[Ze00] Andreas Zeller, Jens Krinke (2000): **Programmierwerkzeuge**, 1. Aufl., Heidelberg, dpunkt Verlag 2000

Danksagung

Wir bedanken uns bei allen wiss. Mitarbeitern und Studenten, die in der Zeit vom April 2002 – März 2003 an diesem Projekt mitgearbeitet und zu diesem Bericht beigetragen haben: Timo Albert, Zahir Amiri, Dino Hasanbegovic, Narcisse Kemogne Kamdem, Christian Kotthoff, Dennis Müller, Matthias Niggemeier, Andre Pavlenko, Stefan Pinschke, Alireza Salemi, Bastian Schlich und Ursula Wellen.