K. Pousttchi, K. Turowski (Hrsg.), Mobile Economy: Transaktionen, Prozesse, Anwendungen und Dienste, GI-Edition, Lecture Notes in Informatics, 2004, pp. 147-161

Das eBill Presentment and Payment-System Com42Bill

Volker Gruhn[1], Lothar Schöpe[2], Alexander Schmitz[3]

[1] Universität Leipzig
Lehrstuhl für Angewandte Telematik / e-Business
04109 Leipzig
volker.gruhn@informatik.uni-leipzig.de

[2] Informatik Centrum Dortmund e.V.
Joseph-v.-Fraunhofer-Str. 20
44227 Dortmund
lothar.schoepe@icd.de

[3] Swisslog GmbH
Martin-Schmeisser-Weg 6-8
44227 Dortmund
schmitz@swisslog.com

Abstract: Die Erstellung und Versendung von Rechnungen in Papierform ist bei vielen Rechnungsstellern ein ebenso größer Kostenfaktor wie die Verfolgung von Zahlungseingängen und die Durchführung von Mahnungen. Aber auch Rechnungsempfänger müssen diese Papierrechnungen bearbeiten und verwalten. Gerade Unternehmen (Telekommunikationsunternehmen, Internetprovider, Kabelnetzbetreiber, GEZ), die monatlich eine Vielzahl von Rechnungen erstellen und versenden, versuchen schon heute durch den Einsatz von E-Mail beim Versenden von Rechnungen Aufwand und Kosten zu reduzieren. Jedoch wird hierdurch nicht die komplette finanzielle Transaktionskette – von der Rechnungsstellung bis zur Rechnungsbegleichung – abgedeckt, so dass immer noch ein Medienbruch stattfindet. Erst durch den Einsatz eines Electronic Bill Presentment and Payment-Systems (EBPP) kann die vollständige Unterstützung der finanzielle Transaktionskette erfolgen, wobei elektronische Zahlungssysteme, die auf der Seite der Rechnungsempfänger verwendet werden, integriert werden können. Darüber hinaus können EBPP-Systeme von Rechnungsstellern zu Marketing- und Vertriebszwecken verwendet werden. In diesem Beitrag werden verschiedene Realisierungsmodelle für EBPP-Systeme beschrieben und nach der Auswahl eines Modells, die Konzeption und Realisierung des EBPP-Systems Com42Bill vorgestellt. Durch das System Com42Bill ist der Zugriff auf Rechnungen über verschiedene mobile Endgeräte (z.B. PDA, Handy) möglich, da aufbauend auf einer Client/Server-Architektur ein „Thin Client“ realisiert wurde. Rechnungsempfänger können die Dienste des Systems Com42Bill daher zeitnah und standortunabhängig nutzen.

1 Einleitung

Ein Electronic Bill Presentment and Payment-System (EBPP) ist ein Softwaresystem, das die Durchführung von Transaktionen zwischen Rechnungsstellern und Rechnungsempfängern auf elektronischem Wege ermöglicht [So02]. Wie in Abbildung 1 dargestellt, ist die beliebteste Zahlungsart bisher immer noch die Bezahlung per Rechnung [Wu99]. Verunsichert durch Nachrichten über Betrug oder Falschabbuchungen misstrauen viele Kunden elektronischen Zahlungsarten, bei denen direkt auf ihr Konto zugegriffen wird. Die Entwicklung von neuen elektronischen Zahlungsmöglichkeiten und Zahlungstransaktionen wird damit auch weiterhin signifikanten Einfluss auf den Erfolg des Electronic Commerce haben.

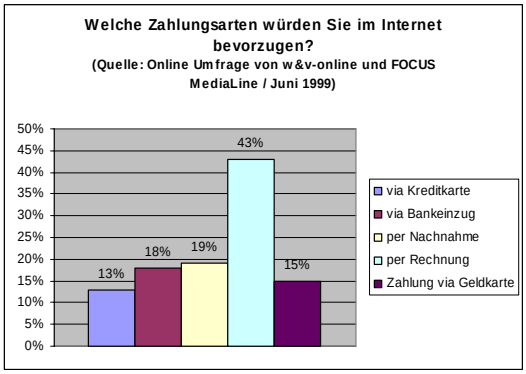

Abbildung 1: Welche Zahlungsarten würden Sie im Internet bevorzugen?

Die bewährte Papierrechnung hat aber sowohl für den Rechnungssteller als auch für den Rechnungsempfänger Nachteile. Dem Rechungsempfänger entstehen Kosten durch die Erstellung, den Versand und die Verwaltung der Rechnungen. Es

entsteht für den Rechnungsempfänger vermeidbarer Aufwand durch Erstellung eines Überweisungsträgers und den anschließenden Gang zur Bank. Hier helfen bereits Teillösungen wie das Homebanking oder M-Payment-Systeme (vgl. [Ki01], [MOS02]). Die Idee des EBPP-Systems ist es jedoch, einen Medienbruch von der Versendung einer Papierrechnung bis zur elektronischen Bezahlung zu vermeiden, um Kosten- und Arbeitsersparnisse sowohl für den Rechnungssteller als auch für den Rechnungsempfänger zu erreichen. Attraktiv auch durch weitere Möglichkeiten, wie dem Anbieten zusätzlicher Finanzdienstleistungen, z.B. der Finanzierung in Raten, oder auch einem automatisierten Mahnwesen. Ein weiterer Vorteil ist die Beschleunigung der Rechnungsübermittlung und somit auch der möglichen Bezahlung. Dem EBPP wurden laut Marktforschungen gute Chancen für die Zukunft eingeräumt [Ba99], [Ko99]. Obwohl diese Erwartungen – resultierend aus der wirtschaftlichen Gesamtsituation der letzten Jahre - nicht ganz eingetroffen sind, kann davon ausgegangen werden, dass aufgrund des Optimierungspotentials EBPP-Systeme auch in Zukunft im Zahlungsverkehr eine besondere Rolle haben werden.

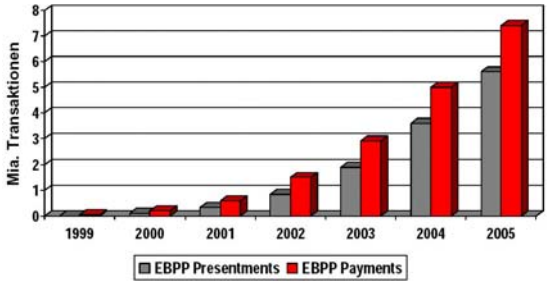

Abbildung 2: eBill Presentments & Payments in Europa [Ko99]

Betrachtet man zusätzlich noch Analysen (vgl. Abbildung 3) die belegen, dass der Internetzugang in Zukunft hauptsächlich von mobilen Geräten stattfinden wird, so wird auch deutlich, welche Ansprüche an ein modernes EBPP-System gestellt werden müssen, um unterschiedlichsten Clients den Zugang zum System zu ermöglichen.

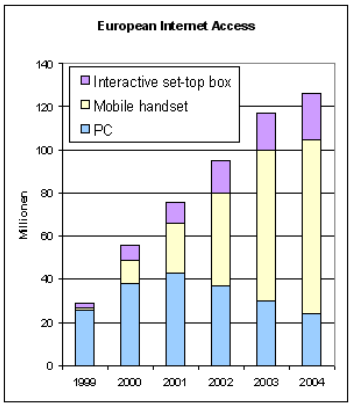

Abbildung 3: Internetzugangsarten in Europa [Cy00]

2 Realisierungsmodelle

Die gegenwärtig diskutierten Realisierungsansätze für ein EBPP-System sind das Thick Consolidator Modell, das Thin Consolidator Modell, sowie das Direct Billing Modell [Ko99], [Fr02], [Eb04]. Das Direct Billing Modell basiert auf einem direkten Kontakt zwischen einem Rechnungssteller und den Rechnungsempfängern. Bei den Consolidator Modellen agiert eine dritte Partei als Mittler für die Rechnungsbearbeitung. Die einzelnen Modelle werden im Weiteren detailliert vorgestellt.

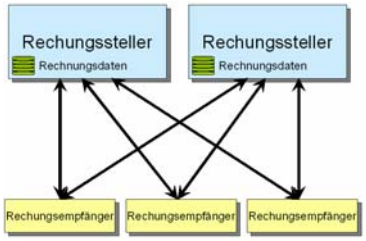

Abbildung 4: Direct Billing-Modell

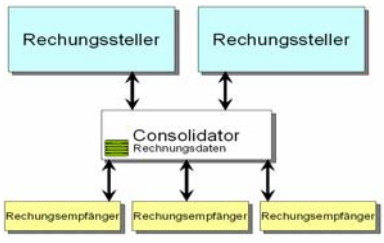

Abbildung 5: Thick Consolidator-Modell

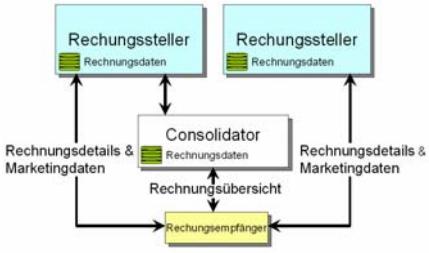

Abbildung 6: Thin Consolidator-Modell

2.1 Direct Billing

Direct Billing Systeme sind besonders bei Telekommunikationsunternehmen, Internetprovidern und Internetshops sehr verbreitet. Diese Unternehmen benutzen ihre eigenen Infrastrukturen und betreiben das EBPP-System bei diesem Modell selbst. Die Rechnungen werden auf elektronischem Wege dem Rechnungsempfänger zur Verfügung gestellt. 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 4).

2.2 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, Single Point of Contact, überprüfen und begleichen kann. Die Rechnungsdaten werden dem Rechnungsempfänger gegenüber also ausschließlich durch den Consolidator verwaltet.

2.3 Thin Consolidator

Über eine Schnittstelle zum Consolidator übermittelt der Rechnungssteller nur die notwendigsten Rechnungsdaten, welche der Rechnungsempfänger auf elektronischem Weg (z.B. 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.

3 Das Modell für das System Com42Bill

Bei der Realisierung eines EBPP-Systems, wie das System Com42Bill[1], müssen die drei Modelle gegeneinander abgewogen werden, um zu einer Lösung zu gelangen. Direct Billing Systeme sind anwenderspezifisch und nur mit hohem Aufwand an unterschiedliche Rahmenbedingungen anzupassen. Ihr Einsatz ist mit vergleichsweise hohem Aufwand für den Rechnungsempfänger verbunden. 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. Die Stärken des Thin Consolidator-Modells bestehen in der Verarbeitung nur der notwendigsten Rechnungsdaten und in dem direkten Kontakt des Rechnungsstellers zum Rechnungsempfänger, das durch ein System dieser Art nicht unterbrochen wird. Die Ergebnisse einer Experten-Untersuchung (vgl. [So02]), die Consolidator-Modellen im Vergleich eine positive Erfolgstendenz einräumen, bestätigen die Entscheidung, mit dem System Com42Bill einen Thin Consolidator zu realisieren (siehe Abbildung 7).

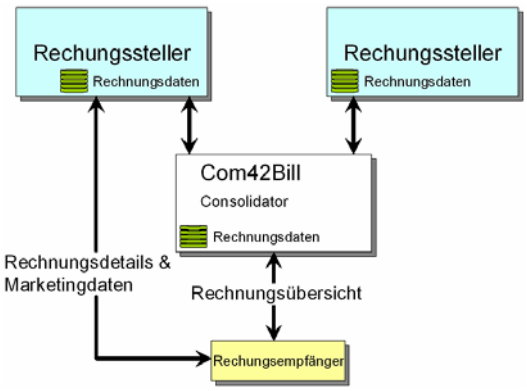

Abbildung 7: Modell für Com42Bill

Die Rechnungsdaten, Buchungsdaten und die Rechungsübersicht werden bei dem System Com42Bill durch Verwendung

[1] Das System Com42Bill (in Anlehnung an den Begriff „comfortable“) wurde in einem universitären Forschungsprojekt entwickelt, um Erfahrungen mit der Umsetzung des ebXML-Frameworks mittels eines J2EE-Applikationsservers zu gewinnen. Diese Erfahrungen sollen bei der Beantragung und Durchführung von weiteren Forschungsprojekten und Industrieprojekten genutzt werden.

von ebXML zwischen den verschiedenen Geschäftspartnern ausgetauscht. Bei ebXML handelt es sich um ein XML-basiertes Framework, welches lediglich den Rahmen spezifiziert, in dem die Daten übertragen werden. Obwohl dieses Framework noch nicht standardisiert ist, werden ebXML werden zurzeit hohe Durchsetzungschancen vorausgesagt, so dass der Einsatz dieses Frameworks sinnvoll erscheint [WHB01].

Zwischen dem System Com42Bill und dem Rechnungssteller wird ein Geschäftsprozess definiert, der den Austausch von Rechnungsdaten beschreibt. Hierbei werden mehrere einzelne Geschäftstransaktionen durchgeführt: Der Rechnungssteller sendet Rechnungen oder Stornos zu bereits existierenden Rechnungen und auch Stati zu bezahlten Rechnungen. Durch das System Com42Bill werden diese Daten verwaltet und dem Rechnungssteller zu jeder empfangenen Rechnung bzw. Storno eine Statusnachricht übermittelt. Diese Statusnachricht gibt an, ob die empfangenen Daten korrekt verarbeitet wurden oder ob Fehler bei dem Datenimport aufgetreten sind.

Der Geschäftsprozess, der zwischen dem System Com42Bill und dem Finanzdienstleister durchgeführt wird, beschreibt den Überweisungsvorgang. Durch das System Com42Bill werden Überweisungen zum Finanzdienstleister übermittelt. Die Geschäftsdaten zu jeder Geschäftstransaktion werden in XML übertragen [WHB01]. Hierzu existiert zu jeder Transaktion ein XML-Schema, welches den Aufbau der Daten definiert. Die Erweiterung auf andere Datenaustauschformate wie z.B. EDIFACT ist durch die flexible und erweiterbare Architektur des Systems Com42Bill möglich. Denkbar ist auch, durch einen weiteren vorgeschalteten Datenkonverter, die Daten zwischen dem bereits bestehenden XML-Format und einem andern Format zu konvertieren [Ge01].

4 Electronic Business XML (ebXML)

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) [Eb02]. Das Ziel ist es, ein Rahmenwerk zu schaffen, mit dem konsistente, robuste und integrationsfähige eBusiness-Anwendungen erstellt werden können und letztendlich dadurch ein einziger globaler eBusiness-Markt geschaffen wird, auf dem sich Unternehmen jeder Größe und jeder geografischen Lage treffen können.

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.

Die Zusammenführung von Geschäftspartnern wird durch Dokumente unterstützt, die mit der Sprache XML beschrieben werden müssen [To99]. Jeder Geschäftspartner, der ebXML nutzen möchte, muss die folgenden Dokumente bereitstellen.

Ein Collaboration Protocol Profile (CPP) dient der öffentlichen Bekanntmachung eines Unternehmens. Es beschreibt, welche Geschäftsprozesse, Nachrichtenformate, Kommunikationsschnittstellen etc. ein Unternehmen unterstützt. Auf Basis der Informationen des CPP können Unternehmen zueinander finden und dann alle weiteren Vereinbarungen für die Durchführung elektronischer Geschäftstransaktionen untereinander getroffen werden.

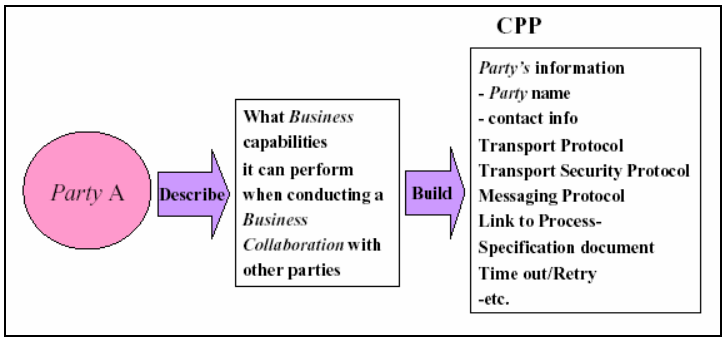

Abbildung 8: Collaboration Protocol Profile (CPP) [Eb02]

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

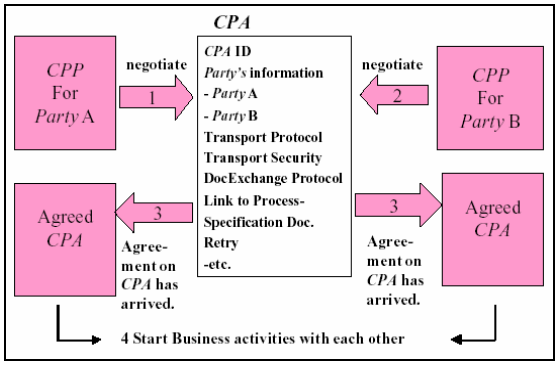

Abbildung 9: Collaboration Protocol Agreement (CPA) [Eb02]

Jedes CPP und CPA ist mit einem Business Process Specification Scheme (BPSS) verknüpft. In einem BPSS werden die Geschäftsprozesse der Unternehmen beschrieben.

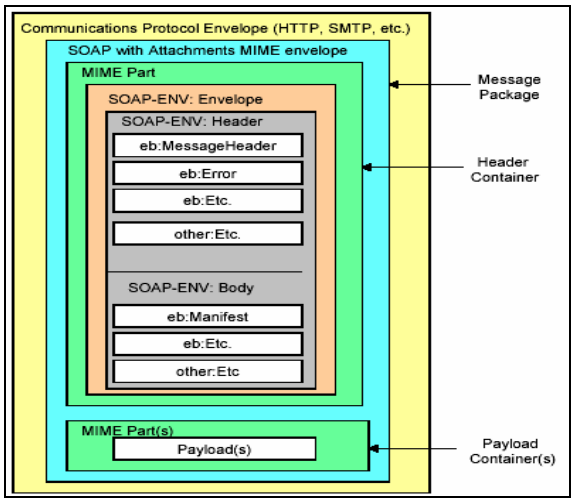

Abbildung 10: Struktur einer ebXML-Nachricht [Eb02]

Für die Verpackung der Geschäftsdaten wird eine Erweiterung von SOAP 1.1 (Simple Object Access Protocol) [Og02], [OM01], welches ebenfalls auf XML basiert, verwendet (siehe Abbildung 10). 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 [Ch01], [Je01].

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.

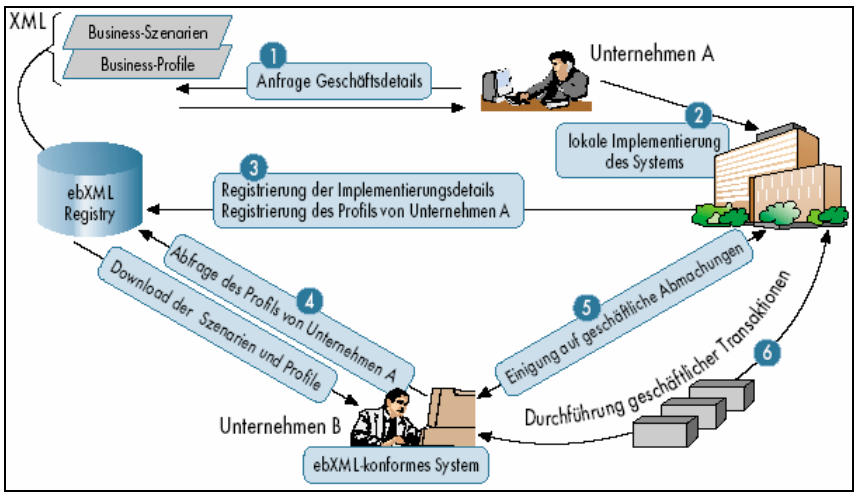

Abbildung 11: Beispielszenario [Eb02]

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. Unternehmen A plant und implementiert eine eigene ebXML-konforme Anwendung, dies kann eine Individualanwendung sein, aber auch eine fertige ebXML-Anwendung eines Drittherstellers. Um selbst von anderen Unternehmen gefunden werden zu können, muss Unternehmen A ein Unternehmensprofil (CPP) erstellen und in der ebXML-Registry ablegen. Unternehmen B sucht einen geeigneten Geschäftspartner. Bei der Suche in der ebXML-Registry nach Unternehmen, die die gesuchten Geschäftsprozesse unterstützen, findet es 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. 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. Der eigentliche Austausch von ebXML-Geschäftsnachrichten kann beginnen.

5 Architektur des Systems Com42Bill

Das System Com42Bill besteht aus fünf Komponenten, deren separate Aufgabenbereiche klar abgegrenzt sind. Der Zugriff einer Komponente auf Dienste einer anderen geschieht ausschließlich über definierte Schnittstellen. Eine vereinfachte Darstellung der Zusammenarbeit der Komponenten in der Gesamtarchitektur ist in Abbildung 12 zu sehen. So ist eine Austauschbarkeit und Unabhängigkeit der Komponenten gewährleistet. Als Basis für die Zusammenarbeit der Komponenten wird auf einen J2EE-Applikationsserver der Firma BEA zurückgegriffen, welcher die für ein solches System notwendige Eigenschaften, wie Transaktionsmanagement oder das Object Relational Mapping (O/R-Mapping) der Java-Objekte auf relationale Datenbanken, etc. zur Verfügung stellt (vgl. [GR02]). Durch Standards bei der Entwicklung mit einem J2EE-Applikationsserver hat das Softwaresystem Com42Bill deutlich an Qualität und Flexibilität gewonnen und kann problemlos auf Applikationsserver anderer Hersteller übertragen werden. Eine weitere, wichtige Entscheidung für die Entwicklung ist der Einsatz von RequestDictionaries. Diese ermöglichen schlanke, aber flexible Schnittstellen zwischen den Komponenten. Unter ein RequestDictionary ist ein Container-Objekt zu verstehen, das über die Dauer einer Anfrage an das System und deren Bearbeitung erhalten bleibt. In dem Container-Objekt können beliebige Objekte unter einem Namen abgelegt werden (Key Value Tupel). Während der Bearbeitung entnehmen Komponenten Informationen aus dem Dictionary und fügen eigene Objekte, die aus der Arbeit der Komponente resultieren hinzu. Für die Kollaboration der Komponenten müssen lediglich die Objektschlüssel und Objekttypen spezifiziert werden, die während der Verarbeitung benötigt werden.

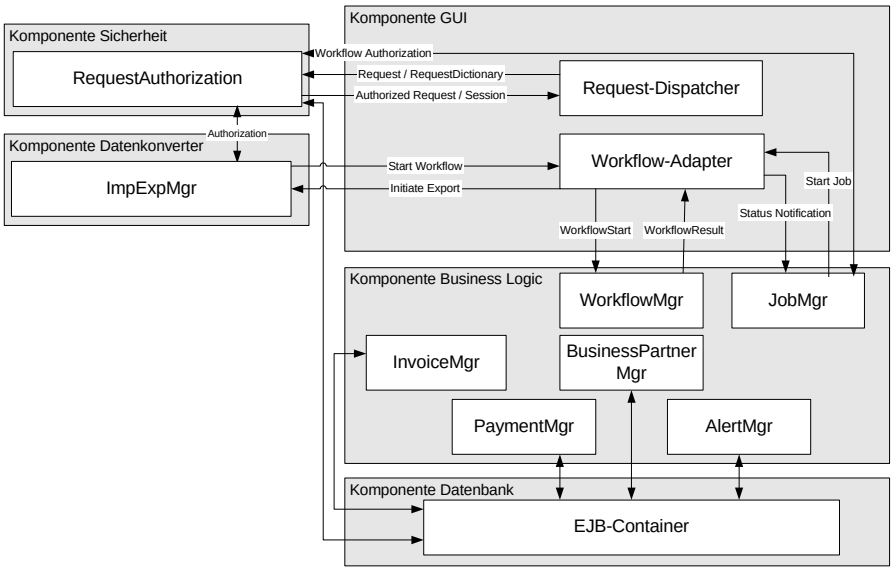

Abbildung 12: Zusammenarbeit der Komponenten in der Systemarchitektur

5.1 Komponente GUI

Bei der Gestaltung der grafischen Oberfläche des Systems Com42Bill wird besonderer Wert auf Benutzerfreundlichkeit und Softwareergonomie gelegt. Hierzu tragen eine überschaubare, einheitliche Seitenstruktur sowie die einfache Möglichkeit, die verschiedenen Funktionen des Systems direkt von der Startseite aus zu erreichen, ihren Teil bei (Abbildung 13). Ebenso erleichtern gezielte Hilfestellungen zu jedem Arbeitsschritt und ein Katalog von häufig gestellten Fragen die Bedienung des Systems.

Abbildung 13: Screenshot Com42Bill

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 unterschiedlichsten mobilen Endgeräte erzeugen. Der Rechnungssteller bspw. greift ausschließlich über das Internet auf das System Com42Bill zu, wofür eine HTML-Darstellung bereitgestellt wird. Eine Erweiterung auf andere Ausgabesprachen wie z.B. WML ist aber leicht realisierbar. Die gleiche XML/XSLT-Technologie wird auch bereits von anderen Projekten und Herstellern benutzt, um Ausgaben in Formate für beliebige mobile Engeräte zu transformieren ([BFG01], [Ce03]).

Als Resultat der Verarbeitung einer Anfrage an das System erhält die Komponente GUI von der Komponente Business Logic fachliche Objekte in Form von Entity Beans, diese müssen in einen XML-Datenstrom transformiert werden. 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, wie der Einsatz von Castor (www.castor.org) oder einer statischen Implementierung der Transformationen. Die letzte und aktuelle Lösung erfolgt über die Java Introspection API, welche die Struktur der Objekte dynamisch abfragen und somit die benötigten Daten extrahieren kann. Liegt nun der XML-Datenstrom vor, kann ein XSL-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. In Zusammenarbeit mit der Business Logic, die auch ereignisorientiert, z.B. beim Eintreffen neuer Rechnungen, Aktionen auslösen kann, ist es möglich sowohl das Push- als auch das Pull-Modell für den Kunden umzusetzen. D.h. der Benutzer muss das System nicht selbst regelmäßig auf neue Rechnungen überprüfen, sondern kann sich auch per SMS oder Electronc Mail informieren lassen.

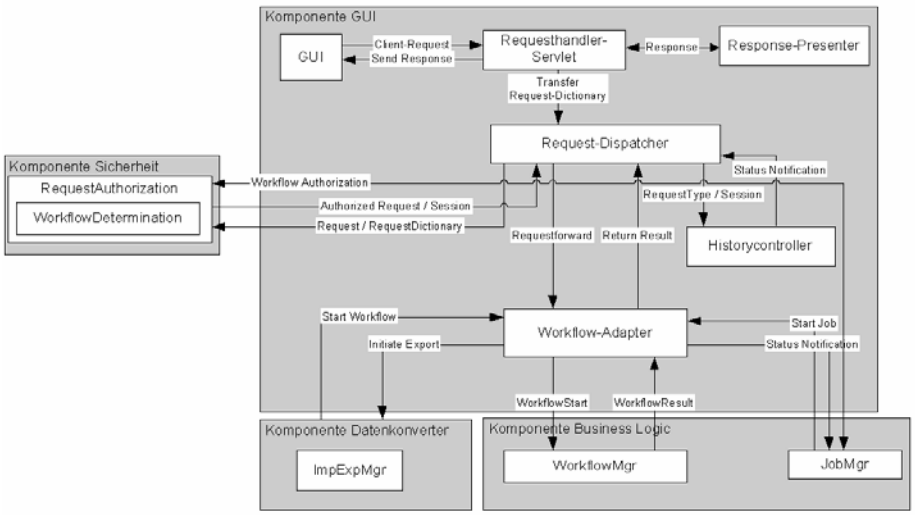

Abbildung 14: Teilarchitektur der Komponente GUI

- GUI:
  Unter diesen Bestandteil fallen alle zur Verfügung gestellten grafischen Benutzeroberflächen, auf die ein Geschäftspartner zugreifen kann.
- RequesthandlerServlet:
  Alle Daten, welche durch einen http-Request an das Servlet abgegeben werden, werden in einem RequestDictionary (Container-Objekt) hinterlegt. Nach einer Syntaxkontrolle der Eingabedaten wird der RequestDispatcher aufge-

rufen, welcher die weitere Abarbeitung des Requests einleitet. Die Ergebnisdaten werden schiesslich durch den ResponsePresenter in einen Ausgabedatenstrom konvertiert und an den Client übertragen.

- 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, wenn nicht, so erfolgt ein Anstoss des gewünschten Geschäftsprozesses bei der Business Logic. Ebenso erfolgt hier die Behandlung von Ausnahmen, welche durch die Komponente Sicherheit ausgelöst werden.
- HistoryController:
  Beim HistoryController ist eine Verbotsliste für nicht erlaubte Übergänge in der Navigationsstruktur der GUI hinterlegt. Der HistoryController arbeitet dabei ähnlich einem Zustandsautomaten, wobei durch das Ausführen von Geschäftsprozessen und dem anschließenden Anzeigen der Ergebnisse Zustände definiert werden. Die Behandlung unerlaubte Übergänge geschieht im RequestDispatcher. 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 Business Logic dar. Der WorkflowAdapter beauftragt den WorkflowMgr, welcher den Einstiegspunkt in die Business Logic darstellt, den Geschäftsprozess zu starten.
- ResponsePresenter:
  Der ResponsePresenter arbeitet die Ergebnisse des Prozesses, welche sich in dem vom RequesthandlerServlet erstellten RequestDictionary befinden, grafisch auf für die GUI-Anzeige. Mit Hilfe der BeanUtils [Ap01] 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.

5.2 Komponente Sicherheit

Diese Komponente ist dafür zuständig, alle Vorgänge zu überwachen und sicherheitskritische Vorgänge zu steuern. Diese Sicherheit wird durch eine Benutzerautorisierung sichergestellt, die kontrolliert, dass der Zugang nur über für den Benutzer erlaubte Schnittstellen erfolgt und dadurch dass die Autorisierung jeder Aktion eines Benutzers innerhalb des Systems überwacht wird, was eine missbräuchliche Nutzung des Systems verhindert. Die Sicherheitskomponente besteht - wie in Abbildung 15 dargestellt - im Wesentlichen aus vier Subkomponenten.

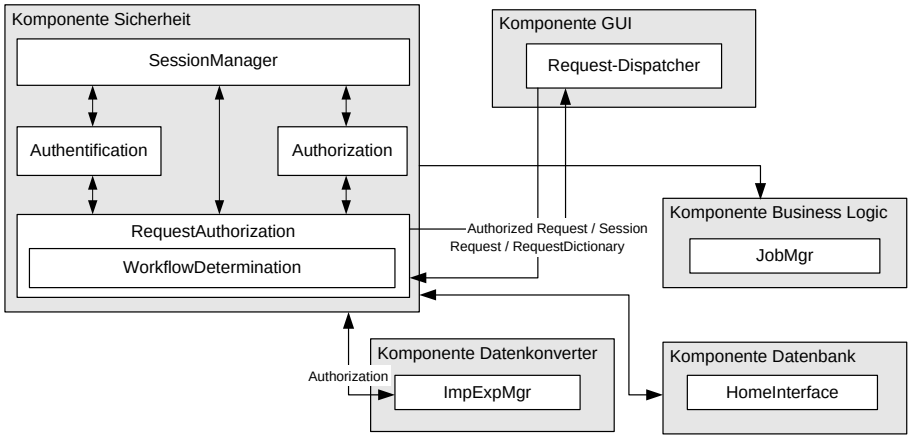

Abbildung 15: Teilarchitektur der Komponente Sicherheit

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.

- 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.
- 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 Gültigkeit einer Session wird mittels des SessionMgr überprüft, ist die Session ungültig oder abgelaufen, werden nur öffentliche Request erlaubt und die Benutzerinformationen aus der Session entfernt.
- Authentification:
  Hier findet die Verwaltung aller Benutzer des Systems statt. Hierzu gehören die Rechnungssteller, die Rechnungs-

empfänger sowie die Administratoren des Systems. Nur Benutzer, die dieser Untereinheit bekannt sind, können sich am System über die ihnen zugewiesene Schnittstelle anmelden.

- Autorization:
  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 dem System Com42Bill in erster Linie Geschäftsprozesse (Workflows) sind.

5.3 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 Aufruf kann sowohl zeit- als auch ereignisgesteuert ausgelöst werden, so dass auch bei dem Eintreffen neuer Rechnungen im System beispielsweise ein Geschäftsprozess ausgelöst werden kann. Ein Geschäftsprozess ist dabei eine zusammengehörige Folge von Aufgaben, welche nach bestimmten Regeln auf ein bestimmtes Ziel hin durchgeführt werden. Ein Beispiel hierfür ist die Durchführung einer Finanztransaktion, die aus verschiedenen Teilaufgaben besteht, 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.

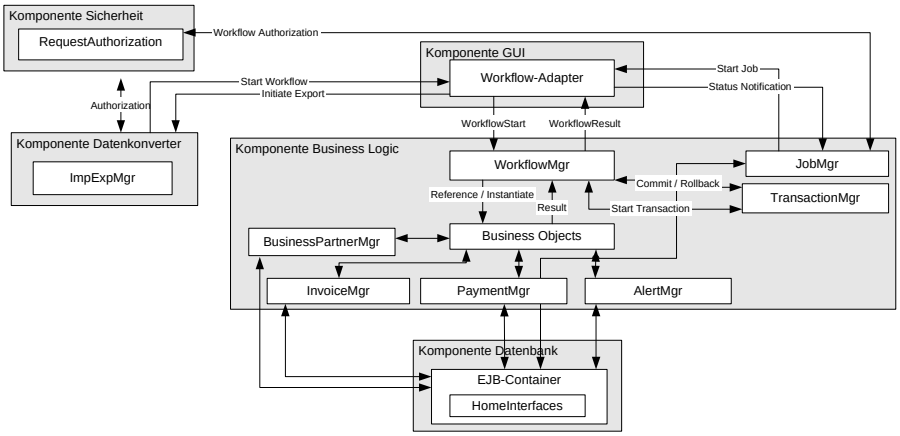

Abbildung 16: Teilarchitektur der Komponente Business Logic

5.3.1 Workflow-Engine

Workflows bestehen aus mehreren Teilabschnitten, welche jeweils kleine Schritte von der Geschäftslogik ausführen, die sog. Business Objects. Im Gegensatz zum reinen Einsatz von Session Beans sind die Teile der Geschäftsprozesse nicht mehr fest verdrahtet. Zur Änderung der Ablaufreihenfolge muss lediglich eine XML-Datei editiert werden, wodurch eine uneingeschränkte, flexible Wiederverwendbarkeit der einzelnen Business Objects garantiert wird. Um eine größtmögliche Flexibilität zu erreichen, müssen weitergehende Funktionen, wie Verzweigungen zwischen Workflows sowie Entscheidungs- und Zusammenführungsknoten, bereitgestellt werden. Für die Modellierung der Workflows bietet sich die Business-Process-Modelling-Language (BPML) der Business-Process-Modelling-Initiative (BPMI) [Bp02] an, einem Zusammenschluss aus vielen Vertretern der Wirtschaft. Dieses Vokabular enthält alle benötigten Funktionen.

Eine wichtige Teilaufgabe bei der Konzeption und Realisierung der Komponente Business Logic war es, die in XML modellierten Workflows auszuführen und in eine Aufrufreihenfolge von Business Objects 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.

- BusinessNode:
  BusinessNodes sind die einzigen Nodes, welche die Business Logic ausführen und direkt oder indirekt auf Entitäten arbeiten. Jeder BusinessNode ist für die Ausführung einer Instanz eines bestimmten Business Objects verantwortlich. Ein Business Object ist dabei eine Stateful SessionBean, welches in mehreren Contexten und somit mehreren Workflows auftreten kann und Aktivitäten ausführt.
- 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.

- 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.
- 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.
- SwitchCaseNode:
  Beim Zutreffen der Condition im SwitchCaseNode wird dieser ausgeführt. Die Condition besteht aus zwei oder drei Teilen, je nachdem, ob Vergleichswerte benötigt werden.
- DefinitionCase-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.
- SwitchDefaultNode:
  Der SwitchDefaultNode wird ausgeführt, wenn keine Condition für einen der SwitchCaseNodes zutrifft.
- JoinNode: Ein JoinNode führt vorher auseinandergegangene Workflows wieder zusammen. Um an einen JoinNode anzuknüpfen, benutzt man einen JumpNode.
- 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“.
- FaultNode:
  Durch die Ausführung eines FaultNodes wird ein Event für die Ausführung des zugehörigen OnFaultNodes ausgelöst.

5.3.2 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.

- Transaktionen:
  Transaktionen sind bei allen schreibenden Operationen auf Entitäten notwendig, ebenso wir beim Navigieren über Container-Managed-Relations, um die Datenkonsistenz zu bewahren bzw. sicherzustellen. Die Transaktionssteuerung erfolgt auf Workflowebene, 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).
- Properties:
  Um die Dynamik der Business Objects weiter zu erhöhen, können in WorkflowContexts Properties angelegt werden, welche aus Key-Value-Paaren bestehen. Diese Parameter werden den Business Objects übergeben. Hiermit können die Business Objects von außen konfiguriert werden. Hierdurch kann die Anzahl der Business Objects drastisch gesenkt werden, wenn zwei Business Objects ähnliche Aktionen durchführen müssen. Sie werden also parametrisierbar.
- Faults:
  Die bereits beschriebenen OnFault-Nodes werden in WorkflowContexts deklariert.

5.3.3 Das Manager-Konzept

Die Basisdienste der Business Logic, die Systemverwaltung, die Rechnungsverwaltung, Finanztransaktionen und das Mahnwesen werden in Form von Managern bereitgestellt, welche nach dem Singleton-Designpattern entwickelt wurden. Andere Komponenten arbeiten ausschließlich über Manager mit der Business Logic. Die Unterteilung des Systems in mehrere Komponenten wird hier konsequent fortgeführt. Somit wird es möglich, die verschiedenen Entwickler mit ihren Spezialfähigkeiten strikt zu trennen. Die Weiterentwicklung des Systems Com42Bill kann somit klar strukturiert erfolgen.

- WorkflowMgr:
  Der WorkflowMgr wird vom WorkflowAdapter aufgerufen. Dieser hat die Aufgabe, den Geschäftsprozess zu star-

ten 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.

5.4 Komponente Datenkonverter

Bei den durch das System 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.

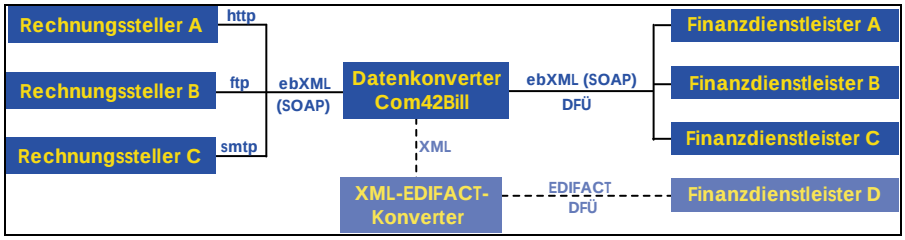

Abbildung 17: Kommunikation mit anderen Systemen

Rechnungsdaten, die von Rechnungsstellern erhalten werden, sind gewöhnlich nicht in einem einheitlichen Standard, sie werden deshalb vom Datenkonverter in ein eigenes Format zur einheitlichen Bearbeitung konvertiert. Für die Kompatibilität des Datenkonverters zu den unterschiedlichen Standards der Rechnungssteller ist der ebXML-Standard ein geeigneter Ansatz. Im Weiteren müssen die Daten der von den Rechnungsempfängern veranlassten Buchungen an die Finanzdienstleister übermittelt werden. Die Übermittlung und Konvertierung der Daten für die Kommunikation mit dem Finanzdienstleister erfolgt ebenfalls auf der Basis des ebXML-Standards. Bei Bedarf können die Buchungsdaten über einen separaten Konverter in EDIFACT [Mü95] konvertiert und dem Finanzdienstleister bereitgestellt werden (siehe Abbildung 17). Der Datenaustausch erfolgt in beiden Fällen online und völlig automatisiert, die Daten können so innerhalb des Systems zügig weiterverarbeitet werden. Welches Transportprotokoll für den Datentransport verwendet wird, ist unabhängig von dem verwendeten Austauschformat, so dass der Datenaustausch an die Erfordernisse des Rechnungsstellers angepasst werden kann. Grundlage der Architektur dieser Komponente ist das ebXML-Framework [Eb02], welches Mechanismen bereitstellt, unabhängig vom verwendeten Datenaustauschformat Dokumente mit Geschäftspartnern auszutauschen. Der Datenkonverter besteht aus den folgenden Subkomponenten (siehe Abbildung 18).

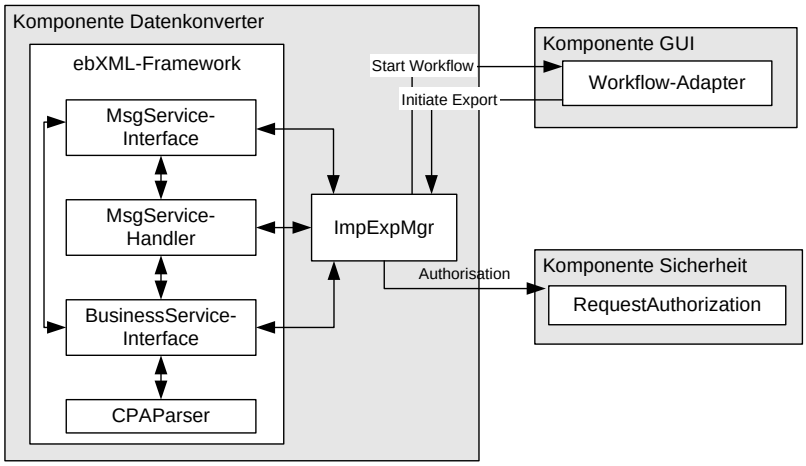

Abbildung 18: Teilarchitektur der Komponente Datenkonverter

- MsgServiceInterface:
  Das MsgServiceInterface ist die Schnittstelle für externe ebXML-Applikationen, die mit dem System Com42Bill elektronischen Datenaustausch betreiben möchten. Hier werden ebXML-Nachrichten über den MsgReceiver empfangen und über den MsgSender versendet.
- MsgServiceHandler:
  Hier werden SOAP-Nachrichten verarbeitet. Eingehende Nachrichten werden entpackt und in ihre elementaren Bestandteile zerlegt (Body, Header, Attachment) bzw. für ausgehende Nachrichten zusammengesetzt.
- BusinessServiceInterface:
  Bei der Kontrolle der ebXML-Geschäftsprozesse werden bei eingehenden Nachrichten 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.
- CPAParser:
  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. Als zentrale Technologie wurde DOM (Document Object Model) eingesetzt, um XML in Java zu konvertieren.
- ImpExpMgr:
  Der ImpExpMgr ist die zentrale Schnittstelle zu den anderen Komponenten des Systems. Die Authentifizierung und Autorisierung bei der Sicherheitskomponente sowie der Im- und Export von Daten in das System erfolgt ausschließlich über diese Schnittstelle

5.5 Komponente Datenbank

Die Komponente Datenbank besteht aus zwei Subkomponenten: Einem Datenbankmanagementsystem (DBMS) und einem EJB-Container, über den die Zugriffe auf die Datenbank stattfinden. Da dem Datenbankmanagementsystem eine relationale Datenbank zugrunde liegt, alle Komponenten jedoch mit Objekten arbeiten, werden die Daten der Objekte in Tabellenstrukturen abgebildet, was direkt durch den eingesetzten Applikationsserver (J2EE) unterstützt wird. Dieser unterstützt beispielsweise direkt m:n-Relationen innerhalb der Container, sowie die EJB-Query Language (EJB-QL), einer SQL-ähnlichen Abfragesprache, mit dessen Hilfe eigene Finder Methods der Home-Interfaces zum ermitteln bestimmter Datensätze geschrieben werden können. Die einzelnen Bestandteile der Komponente, welche in Abbildung 19 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 stehen aus Performanzgründen in einem Pool zur Verfügung. In diesem Connection Pool werden geöffnete JDBC-Verbindungen gehalten, über welche der Applikationsserver mit der Datenbank kommunizieren kann.
- DBMS:
  Als Datenbankmanagementsystem wird ein relationales DBMS (z.B. Oracle) eingesetzt.

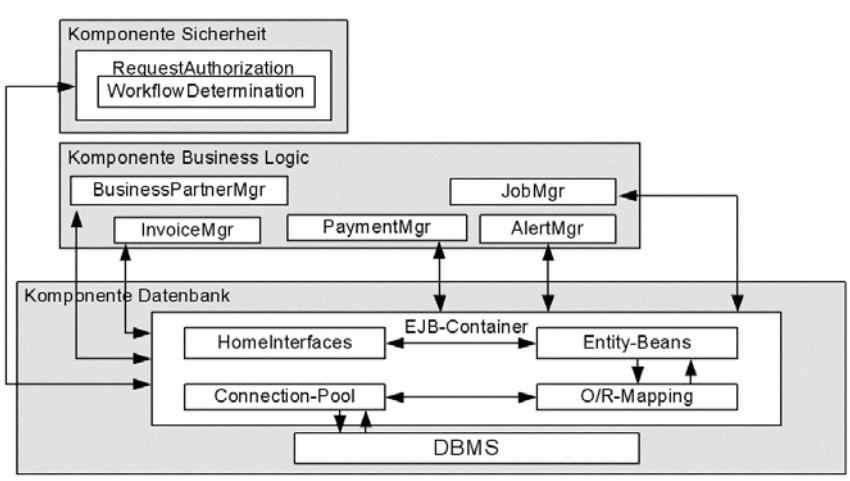

Abbildung 19: Teilarchitektur der Komponente Datenbank

6 Zusammenfassung und Schluss

Der Nachweis der Praktikabilität und der Praxisrelevanz des Systems Com42Bill in einem industriellen Kontext steht noch an. Obwohl das System in einem universitären Forschungsprojekt entwickelt wurde, brachten verschiedene Industrieunternehmen (Softwarehäusern und DV-Abteilungen bei Anwendern) bei der Vorstellung des Systems auf unterschiedlichen Veranstaltungen dem System ein großes Interesse entgegen.

Durch die Gewinnung eines industriellen Partners könnten weitere Erfahrungen und Probleme bei der Integration des Systems Com42Bill in eine existierende Hard- und Softwareinfrastruktur gewonnen und Aspekte wie Marktrelevanz, steuerliche Rahmenbedingungen, etc. überprüft werden.

Da jedoch durch den Einsatz eines Application Servers und der Nutzung der angebotenen Dienste das System Com42Bill adaptierbar und portierbar ist, wird ein hoher Grad an Skalierbarkeit erreicht. Die unterschiedlichen Systemkomponenten können unter Beibehaltung ihrer Schnittstellen ausgetauscht werden, z.B. die Komponente Datenkonverter kann leicht auf die verschiedenen Formate und Kommunikationskanäle von Rechnungsstellern angepasst werden, ohne dass das ebXML-Framework verwendet werden muss.

Diese Skalierbarkeit, Portierbarkeit und Adaptierbarkeit ist umso wichtiger, da sich bisher im europäischen Markt kein einheitliches EBPP-System durchgesetzt hat, obwohl dem Markt für EBPP-Systeme gute Chancen vorausgesagt werden. Sollte sich der Markt konsolidieren, kann das System Com42Bill in existierenden Hard- und Softwareinfrastrukturen integriert werden.

Erst durch eine kritische Masse von Kooperationspartnern (Banken, Telekommunikationsunternehmen, etc.) wird der Betrieb eines EBPP-Systems für den Rechnungssteller rentabel und für einen Rechnungsempfänger attraktiv. Diese kritische Masse ist in Europa bisher lediglich in der Schweiz durch das Unternehmen Paynet erfolgreich erreicht worden, in dem es mit den größten Schweizer Banken und einer Vielzahl von Rechnungsstellern zusammen arbeitet, die das System für den Kunden attraktiv machen.

Folglich kann ein EBPP-System nur durchsetzungsfähig sein, wenn es flexible Schnittstellen zu Systemen von Kooperationspartnern bietet, um mit wenig Aufwand mit diesen zusammenarbeiten zu können, und außerdem dem Kunden ein leicht bedienbares, überall erreichbares System zur Verfügung stellt.

Bei dem hier vorgestellten System Com42Bill werden diese Eigenschaften durch eine komponentenorientierte Systemarchitektur, durch den Einsatz des ebXML-Frameworks zum Datenaustausch und durch die Kombination von XML und XSLT zur Realisierung von Benutzungsoberflächen für diverse mobile Endgeräte erreicht.

7 Literaturverzeichnis

[Ap02] Apache Jakarta Project: BeanUtils, <http://jakarta.apache.org/commons/beanutils.html>, (15.12.2002)

[Ba99] Barling, Beth: Ovum Market Research 1999 – Bill Presentments & Payments in Europe, <www.ovum.com>, (05.01.2002)

[BFG01] Baumgartner, R., Flesca, S., Gottlob, G.: Visual Web Information Extraction with Lixto. In: Apers, P. (et.al.) (eds.): Proceedings of the 27th VLDB Conference, Morgan Kaufmann Press, 2001, S. 119-128

[BG98] Baker, S., Geraghty, R.: JAVA For Business Objects. In: Carmichel, A. (ed.) Developing Business Objects, SIGS, Cambridge University Press, 1998, S. 225-237

[Bp02] BPMI.org: The Business Process Management Initiative, <www.bpmi.org>, (20.07.2002)

[Ce03] Cellmania Inc.: mEnterprise – Product Features, <www.cellmania.com>, (28.10.2003)

[Ch01] Chester, T.M.: Cross-Platform Integration with XML and SOAP. In: IT Professional, Vol. 3, Issue 1, IEEE Computer Society Press, 2001, S. 26-34

[Cy00] Cytexx Media Devision: Strategy Report Europe – Mobile Marketing. Cytex Media GmbH, 2000

[Eb02] ebXML.org: Enabling a global electronic market, <www.ebxml.org>, (15.04.2002)

[Eb04] Goers Consult GmbH, EBPP – Modelle, <www.ebppinfo.de>, (06.01.2004)

[Fr02] Franck, C.: Electronic Bill Presentment and Payment. MA-Thesis, Fachhochschule Konstanz, 2002

[Ge01] Georg , B.: Begriffswirrwarr beim Datenaustausch. In: e-commerce magazin 8/2001, IWT Verlag, 2001, S. 40-42

[GS02] Gruhn, V., Schneider, A.: EJB 2.0 Anwendungen. Addison-Wesley, 2002

[Je01] Jepsen, T: SOAP Cleans up Interoperability Problems on the Web. In: IT Professional, Vol. 3, Issue 1, IEEE Computer Society Press, 2001, S. 52-55

[Ki01] Kieser, M.: Mobile Payment – Vergleich elektronischer Zahlungssysteme. In: Meier, A. (Hrsg.): Mobile Commerce, HMD Band 220, dpunkt Verlag, 2001, S. 27-36

[Ko99] Koch, B.: Electronic Bill Presentment & Payment (EBPP). Beweco GmbH, 1999

[Mü95] Müller-Berg, T.: EDI-Knigge, Springer Verlag, 1995

[MOS02] Mustafa, N., Oberweis, A., Schürr, T.: Mobile Banking und Sicherheit im Mobile Commerce. In: Silberer, G., Wohlfahrt, J., Wilhelm, T. (Hrsg.): Mobile Commerce – Grundlagen, Geschäftsmodelle, Erfolgsfaktoren. Gabler Verlag, 2002

[Og02] Ogbuji, U.: Using WSDL in SOAP-Applications, <http://www-4.ibm.com/spftware/developer/library/wssoap/index.html> (03.03.2002)

[OM01] O'Connell, P., McCrindle R.: Using SOAP to Clean up Configuration Management, In: Wu, P.: Proceedings of the COMPSAC'01 Conference, IEEE Computer Society Press, 2001, S. 555-560

[Pa02] Homepage der Schweizer Paynet AG, <www.paynet.ch> (20.02.2002)

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

[To99] Tolksdorf, R.: XML und darauf basierende Standards. In: Informatik Spektrum, Band 22, Heft 6, Springer Verlag, 1999, S. 407-421

[WHB01] Weitzel,T., Harder, T., Buxmann P.: Electronic Business und EDI mit XML, dpunkt Verlag, 2001

[Wu99] Werben & Verkaufen: Fachzeitschrift zu Werbung, Kommunikation und Marketing, Juni 1999 <http://www.wuv.de/servlet/wuv/community/umfrage_archiv.html> (12.03.2002)