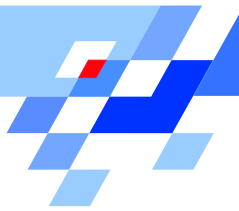
UNIVERSITÄT DORTMUND

Fachbereich Informatik
Lehrstuhl für Software-Technologie



M E M O Nr. 135

Endbericht der Projektgruppe Com42Bill (PG 411)

Timo Albert
Christian Kotthoff
Alireza Salemi
Zahir Amiri
Dennis Müller
Bastian Schlich
Dino Hasanbegovic
Matthias Niggemeier
Alexander Schmitz
Narcisse Kemogne Kamdem
Andre Pavlenko
Volker Gruhn
Lothar Schöpe
Ursula Wellen

März 2003

Internes Memorandum des
Lehrstuhls für Software-Technologie
Prof. Dr. Ernst-Erich Doberkat
Fachbereich Informatik
Universität Dortmund
Baroper Straße 301

D-44227 Dortmund

ISSN 0933-7725

Universität Dortmund
Fachbereich Informatik
Lehrstuhl für Software-Technologie

Endbericht der Projektgruppe 411

Timo Albert, Zahir Amiri, Dino Hasanbegovic

Narcisse Kemogne Kamdem, Christian Kotthoff

Dennis Müller, Matthias Niggemeier

Andre Pavlenko, Alireza Salemi,

Bastian Schlich, Alexander Schmitz

Volker Gruhn, Lothar Schöpe, Ursula Wellen

Version 1.0 vom 27.03.2003

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.

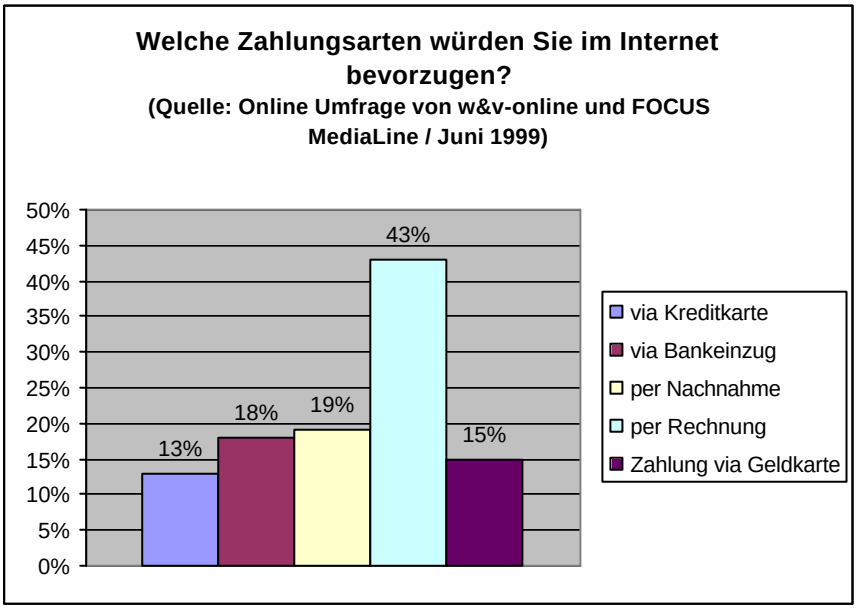


Abbildung 1: Welche Zahlungsarten würden Sie im Internet bevorzugen?

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.

Der hier vorliegende Endbericht beschreibt die Arbeitsergebnisse der Projektgruppe Com42Bill, die die obigen Überlegungen in ein Softwaresystem umgesetzt hat. Nach einer Seminarfahrt, dessen Ergebnisse in Kapitel 3 vorgestellt werden, wurde zunächst ein Modell entwickelt, das das Grundgerüst für das zu implementierende System darstellt. Dieses in Kapitel 4 vorgestellte Modell wurde weiterentwickelt; Ergebnis ist die Architektur des Systems (Kapitel 5) sowie das zugehörige Datenmodell (Kapitel 7). Kapitel 6 beschreibt die Funktionalität des Systems. In Kapitel 8 folgt die Beschreibung der eingesetzten Hard- und Software. Die Berichte des Projektmanagements und der Qualitätsmanager schliessen sich diesem an. Den Abschluss bildet ein Fazit, in dem die Erfahrungen der PG-Teilnehmer zusammengefasst sind. Im Anhang befinden sich Erläuterungen und Beschreibungen der eingesetzten bzw. entwickelten Technik.

2 Was ist eine Projektgruppe?

Im Rahmen eines Studiums nimmt ein Informatikstudent hauptsächlich an Vorlesungen teil. Diese vermitteln theoretische Grundlagen und Spezialwissen. Dabei sind die Anwendungsgebiete eher abstrakt formuliert; der praktische Einsatz dieses Wissens wird hierbei nicht erlernt.

Um diesem Problem zu begegnen, gibt es Projektgruppen. Die Projektgruppe ist ein zentraler Bestandteil des Hauptstudiums jedes Informatikstudenten der Universität Dortmund. Diese Pflichtveranstaltung umfasst zwei Semester mit jeweils ca. 15 Semesterwochenstunden. Acht bis zwölf Studenten nehmen an einer Projektgruppe teil. Die Gruppe wird von zwei Lehrstuhlmitarbeitern betreut. Diese Betreuer beraten und unterstützen die Projektgruppe.

Projektgruppen zielen darauf ab, verschiedene praktische Fähigkeiten zu vermitteln und diese zu trainieren. Da Problemstellungen im Allgemeinen zu komplex bzw. zu umfangreich sind, um diese allein lösen zu können, muss das Problem innerhalb der Gruppe aufgeteilt werden. Dabei wird auch der Umgang mit gruppendynamischen Prozessen erlernt. Durch die Konfrontation mit neuen Problemen reicht es nicht aus, bekanntes Wissen anzuwenden. Vielmehr sind die Teilnehmer einer Projektgruppe gezwungen, sich über neue Techniken zu informieren, diese an ein Problem anzupassen und anschließend anzuwenden.

Die Teilnehmer der Projektgruppe bekommen ein Problem vorgestellt. Dieses Problem müssen sie selbständig analysieren und strukturiert Lösungen dazu erarbeiten. Am Ende wird die Lösung implementiert; es werden also alle Phasen eines Softwareentwicklungsprozesses durchlaufen. Die Darstellung der Projektgruppe nach Außen und die Kommunikation mit der Außenwelt regelt die Gruppe in eigener Regie. Die Betreuer kontrollieren das Vorgehen und stehen der Gruppe in Problemfällen mit Ratschlägen zur Seite.

3 Seminarphase

3.1 Allgemein

Um das Wissen aller Teilnehmern einer Projektgruppe auf eine homogene Basis zu stellen, werden in einer Seminarphase verschiedene Vorträge zu den unterschiedlichen Aspekten, die sich aus der Zielsetzung der Projektgruppe ergeben, gehalten. Die Seminarphase findet in der Regel vor dem eigentlichen Beginn der Projektgruppe in Form einer Exkursion statt, um das Kennenlernen der Teilnehmer untereinander zu fördern. Es folgt eine Zusammenfassung der Ausarbeitungen zu den Vorträgen, die Ausarbeitungen selbst sind auf der Internetseite des Projekts [C42B02] erhältlich.

3.2 Softwareentwicklungsprozesse

Mit steigenden Ansprüchen an aktuelle Softwaresysteme wird das Augenmerk immer mehr auf Softwareentwicklungsprozesse gelenkt, die diese Ansprüche erfüllen können. Aufgrund verschiedener Anforderungen bei der Softwareentwicklung sind unterschiedliche Softwareentwicklungsprozesse entstanden.

Man muss bei der Betrachtung der Prozesse mehrere Begriffe unterscheiden. Ein Vorgehensmodell beschreibt auf abstrakte oder generische Weise, wie ein Softwaresystem entwickelt wird. Ein Prozessmodell beschreibt die Aktivitäten, deren Reihenfolge sowie deren Ziele und Elementarmethoden eines Prozesses. Ein Prozessleitfaden ist das Ergebnis der Anpassung eines Prozessmodells an ein Unternehmen [Mül99].

Während Anfang der siebziger Jahre das Wasserfallmodell entstand, welches den Entwicklungsprozess in Phasen unterteilt, existieren heutzutage eine Vielzahl von Modellen, welche deutlich komplexere Strukturen vorweisen. Das Spiralmodell [Boe86] basiert auf dem Wasserfallmodell. Es arbeitet in Zyklen, welche den Phasen des Wasserfallmodells entsprechen. Ein weiteres Modell ist das Prototyping Modell [Kah01]. Es gibt verschiedene Ansätze des Prototyping. Hier sind vor allem exploratives, evolutionäres, experimentelles und rapid prototyping zu erwähnen. Das V-Modell [IESE02] ist 1992 vom deutschen Verteidigungsministerium veröffentlicht worden. Es sollte die Softwareentwicklung im Bereich der Bundeswehr regeln. Mittlerweile ist dieses Modell aber für alle Bundesaufträge verpflichtend. Viele Unternehmen haben dieses Modell zum internen Standard gemacht. Das V-Modell ist in 4 Submodelle unterteilt. Die Entwicklung selbst wird im Submodell Systemerstellung (SE) dargestellt. Die Submodelle Qualitätssicherung (QS), Konfigurationsmanagement (KM) und Projektmanagement (PM) begleiten diese Entwicklung. Jedes dieser Submodelle hat seine eigenen Aktivitäten, Produkte und Rollen. Der Rational Unified Process (RUP) [Kru98] ist eigentlich kein Entwicklungsprozess, sondern ein kommerzielles Komplettpaket, das einen Entwicklungsprozess enthält. Der RUP unterteilt analog zum Wasserfallmodell die Entwicklung in vier Phasen. Abgeschlossen wird eine Phase durch einen Meilenstein. Sollten die Anforderungen eines Meilensteins nicht ausreichend erfüllt sein, so muss die Phase wiederholt werden. Extreme Programming [Wel99a] ist eine neue Entwicklungsmethodik, die in den letzen Jahren bekannt geworden ist. Es ist ein leichtgewichtiger Entwicklungsprozess, welcher auf vier Werten basiert. Diese Werte lauten Kommunikation, Einfachheit, Rückkopplung und Mut. Im IBM-OOTC-Prozess [Müll99] ist eine Phase die kleinste Prozesseinheit. Eine Phase ist nicht, wie bei den anderen Modellen, eine zeitliche Einteilung, sondern ein sich wiederholender Vorgang, welcher sich auf einen Entwicklungsaspekt konzentriert. Jede Phase erstellt ihr eigenes Produkt. Der IBM-OOTC-Prozess beschreibt diese Phasen und Produkte und definiert, welche Tätigkeiten in den Phasen durchzuführen sind.

Nicht zuletzt werden auch Entwicklungsmodelle als weiterzuentwickelnde Produkte verstanden. Dieser Aufgabe widmen sich die Qualitätsmodelle. Das CMM teilt Softwareentwicklungsprozesse in 5 Qualitätsstufen ein. Je nach Einordnung in eine dieser Stufen werden unterschiedliche Maßnahmen zur Erreichung des Stufenziels vorgeschrieben. Bootstrap baut auf denselben Qualitätsstufen auf, welche im CMM definiert wurden. Es teilt jedoch diese Stufen in Viertel ein, um eine genauere Einordnung zu erlangen. Außerdem wird das Unternehmen in drei Bereiche eingeteilt. Diese Bereiche lauten Organisation, Methodik und Technologie. SPICE (Software Process Improvement and Capability Determination) [SQI02] wurde 1993 ins Leben gerufen, um einen internationalen Standard auf Basis der ISO 9001 Norm zur Qualitätssicherung bei der Softwareentwicklung zu entwickeln. SPICE baut teilweise auf CMM auf und erweitert den Umfang dieser Methode. Es enthält keine Vorgehensweise zur Beseitigung von Schwächen.

Zur Unterstützung der Entwicklung existieren verschiedene Notationen. Die am häufigsten eingesetzten Notationen sind UML, Ereignisgesteuerte Prozessketten (eEPK) und Petri Netze.

3.3 Applikationsserver

Ein Application Server ist eine Lösung für viele Client/Server Problematiken: monolithische und nur teuer zu wartende Applikationen, erhöhte Netzwerkbelastung wegen der Datenverarbeitung auf dem Client, schlechte Wiederverwendbarkeit der Applikationslogik. Durch Einführung einer zusätzlichen Schicht – der Applikationsschicht – wird die Applikationslogik von den Clients in einen Application Server verlagert. Die Clients beschäftigen sich nur mit der Entgegennahme von Benutzer-Events und mit der Steuerung des Benutzer-Interface (Abbildung 2).

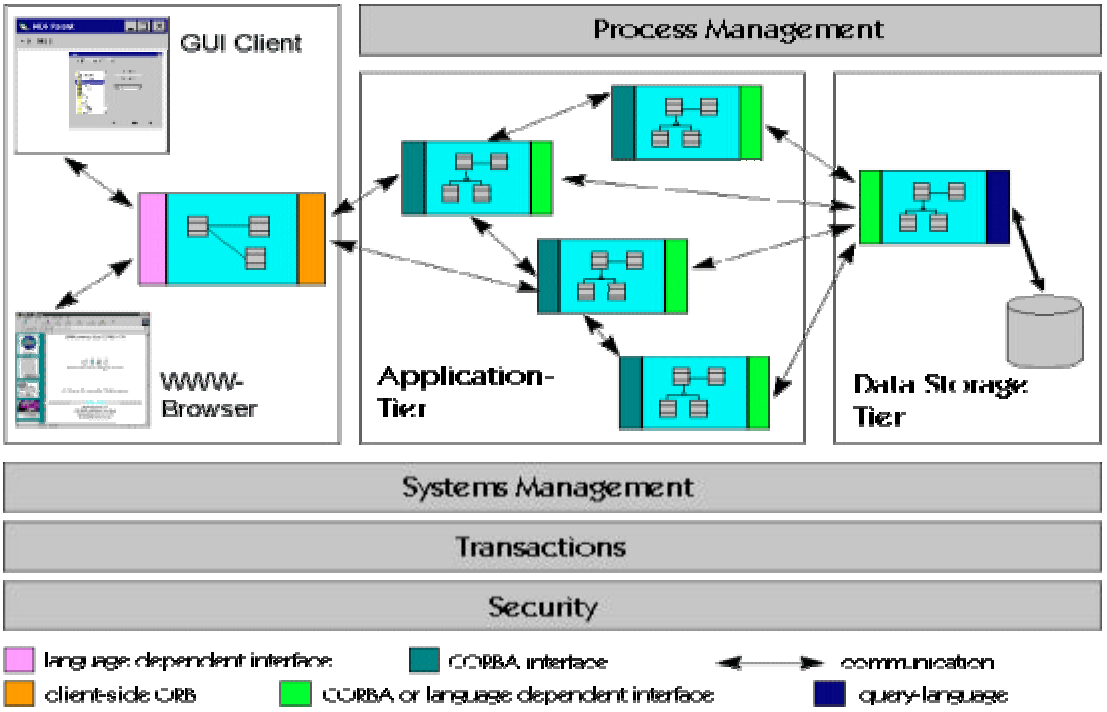


Abbildung 2: Aufbau eines 3- und mehrschichtigen Systems

Als Standard, der das Zusammenspiel zwischen Application Server und Komponenten genau regeln soll, wurde Enterprise Java Beans (EJB, aktuell in der Version 2.0) festgelegt. EJB-Komponenten werden Enterprise Beans genannt und bedienen sich der Programmiersprache Java, die sich aufgrund ihrer Vorteile (klare Trennung von Interface/Implementierung, Plattformunabhängigkeit...) auf diesem Gebiet durchgesetzt hat. Als rein serverseitige

Komponenten enthalten EJB's keine grafische Benutzeroberfläche, sondern implementieren die reine Funktionalität einer Applikation (Business Logic). Clients nutzen die von den Beans angebotenen Dienste, indem sie sich mit diesen mittels geeigneter Protokolle verbinden. Die Realisierung einer grafischen Benutzeroberfläche ist dabei nicht zwingend erforderlich.

Ein weiterer wichtiger Standard im Bereich der Java Application Server ist die Java 2 Platform Enterprise Edition Spezifikation von Sun Microsystems, die die Standards beschreibt, an die sich Hersteller/Entwickler von zertifizierter Enterprise-Software halten müssen. Als reine Spezifikation lässt J2EE, die zurzeit in Version 1.3 vorliegt, den Herstellern bei der Implementierung freie Hand, was teilweise wiederum zu Problemen der Inkompatibilität führt, da Spezifikationen mitunter unterschiedlich ausgelegt werden.

Da die vollständige Betrachtung der Eigenschaften auch nur eines Application Servers den Rahmen dieses Abstracts sprengen würde, wird daher für Interessierte auf das Seminar Java Application Server des WS2000 verwiesen. [AS02]

3.4 Konfigurationsverwaltung und Projektdokumentation

Ziel dieses Abschnitts ist es, die prinzipielle Arbeitsweise von Software-Konfigurations-Management (SKM)-Werkzeugen vorzustellen. Darüber hinaus wird eine Auswahl an Werkzeugen kurz präsentiert.

Software-Konfigurationsmanagement (SKM) ist die Aufgabe, Änderungen - und damit die Evolution - eines Softwaresystems zu organisieren und zu kontrollieren. Das Ergebnis einer Änderung an einer Software-Komponente wird als Version bezeichnet und wird von einem SKM-Werkzeug durch automatische Vergabe einer Versionsnummer dokumentiert. Unter einer Konfiguration versteht man die Zusammenfassung aller Komponenten eines SW-Systems zu einer Einheit [Ze00].

Um SKM in automatisierter Form anzuwenden, bedient man sich sogenannter SKM-Werkzeuge. Aufgaben hierbei sind die Rekonstruktion (Rückgriff auf eine vorherige Version einer Komponente), die Koordination (Vermeidung von gleichzeitiger Bearbeitung einer Komponente durch verschiedene Entwickler) und die Identifikation (Dokumentation von Unterschieden zwischen einzelnen Versionen).

Das Repository eines SKM- Werkzeugs ist ein zentrales Archiv, in dem alle Versionen aller Komponenten eines Softwaresystems abgelegt werden. Mit dem Begriff Sandbox wird der Arbeitsbereich bzw. das Arbeitsverzeichnis bezeichnet, in dem die Änderungen an den Komponenten vorgenommen werden, wobei der Inhalt des Repository zunächst von den Änderungen unberührt bleibt.

Durch eine check in - Funktion werden Komponenten aus dem Arbeitsbereich in das Archiv kopiert. Das Werkzeug ermittelt dabei die Änderung gegenüber der vorherigen Version und vergibt automatisch eine Versionsnummer bzw. Revisionsnummer. Bei der check out-Funktion wird in umgekehrter Weise die aktuellste Kopie der gewählten Komponenten in den Arbeitsbereich kopiert [Pe01].

Durch eine lock-Funktion können Dateien nach einem check out für andere Benutzer mit einem Schreibschutz versehen werden. Durch diese Funktion kann eine Datei immer nur sequentiell von allen Benutzern bearbeitet werden. Um Bearbeitungsvorgänge zu parallelisieren, bieten viele SKM- Werkzeuge die Möglichkeit, Versionierungen in verschiedene Pfade aufzuteilen und zu einem späteren Zeitpunkt wieder zusammenzuführen.

Das *Revision Control System* (RCS) ist ein kostenloses Werkzeug, mit dem nur Dateien verwaltet werden können, Unterverzeichnisse werden nicht unterstützt. *CVS* (Concurrent Version

System) kann im Gegensatz zu RCS auch ganze Verzeichnisbäume bzw. komplette Software-Projekte versionieren. *SCCS* (Source Code Control System) ist ähnlich wie RCS sehr rudimentär, allerdings ohne Portierungen nach Windows und ohne grafische Benutzerschnittstelle. Als kommerzielle Varianten verfügbarer SKM - Werkzeuge sind beispielsweise *PVCS Version Manager* , *RATIONAL Clearcase* oder *Microsoft Visual Source Safe* zu nennen [Sch00, Wie02].

Zur Projektdokumentation gehören die Beschreibung von Schnittstellen und Funktionen und die Dokumentation der durchgeführten Änderungen (Versionsgeschichte). Änderungen können mit Hilfe der SKM-Werkzeuge dokumentiert werden. Beschreibungen und Design-Dokumente müssen durch das SKM-Werkzeug mitverwaltet werden.

3.5 Telematik-, Mobilfunk- und Netzwerktechniken

Dieser Vortrag dient der Einführung in die Bereiche Telematik, Mobilfunk- und Netzwerktechniken.

Aus der Definition des Begriffs *Telematik* geht hervor, dass unter dem Begriff Telematik der gemeinsame Einsatz von Informatik und Telekommunikationstechnik zu verstehen ist [BMI97]. Als ein einfaches Beispiel kann man sich zwei Rechner vorstellen, die über Mobilfunk miteinander kommunizieren. Zu den populärsten Einsatzgebieten der Telematik gehören das Gesundheits- und Verkehrswesen. Im Gesundheitswesen werden durch Telematik Kommunikationserleichterungen und Effizienzsteigerungen durch Rationalisierungsprozesse erreicht. Zudem können durch Schaffung integrierter Versorgungsketten Qualitätsverbesserungen erzielt werden. Durch den Einsatz der Telematik im Straßenverkehr wird die bestehende Infrastruktur effizienter genutzt und damit der Verkehrsfluss verbessert. Weitere Folge ist eine Optimierung der Transport- und Verkehrsabläufe. Ferner werden bei Einsatz von Telematikstrukturen im Strassenverkehr auch eine Verringerung der Umweltbelastung und eine Erhöhung der Verkehrssicherheit prognostiziert [LBE01].

Das Kapitel über Mobilfunktechniken gibt einen Überblick über die Technik, die dem Mobilfunk zu Grunde liegt, angefangen mit einem kurzen Ausflug in die analoge Vergangenheit des Mobilfunks über im Augenblick verwendete Techniken bis hin zu modernen Verfahren, die in Zukunft die Mobilfunktechnik bestimmen sollen.

Grundlegend für Netzwerktechniken sind die Referenzmodelle ISO/OSI und TCP/IP. Die wesentliche Idee ist hierbei, datenverarbeitungsorientierte Funktionen, transportorientierte Funktionen und physikalische Übertragung getrennt voneinander zu betrachten und zu realisieren [Obe98]. So wird die Komplexität bei der Planung und dem Entwurf des Rechnernetzes reduziert.

3.6 XML-basierte Datenaustauschformate

Der automatisierte Austausch von Daten zwischen Maschinen ist seit mehreren Jahrzehnten für Geschäftstransaktionen unerlässlich. Electronical Data Interchange (EDI) ist hierbei der Obergriff für den „unternehmensübergreifenden Austausch von strukturierten Geschäftsdokumenten zwischen Rechneranwendungen unter Nutzung von Datenformatstandards und Kommunikationswegen“ [Geo01]. Als Kommunikationsweg hat sich in den letzten Jahren das Internet mit seinen Standardprotokollen weltweit etabliert. Somit steht den Geschäftspartnern ein weitverbreitetes kostengünstiges Medium zur Verfügung.

Im Bereich der Datenformatstandards basieren alle heutigen Auszeichnungssprachen auf der *Standardized Generalized Markup Language* (SGML), welche die allgemeinste und ausdrucksstärkste Sprache darstellt. In Anschluss an die hauptsächlich für die Präsentation von Inhalten gedachten *Hypertext Markup Language* (HTML) wurde die *eXtensible Markup*

Language (XML) entwickelt. Diese nimmt eine Trennung zwischen der Struktur der Dokumente und den Inhalten vor, welche separat definiert werden können. Somit ist die Sprache problemlos erweiterbar, da eigene Strukturen definiert werden können [Jec01, Bei01].

XML bildet die Grundlage für sogenannte Web Services. Diese Dienste sind Softwarekomponenten, die via *Remote-Procedure-Calls* (RPC) über das Internet genutzt werden können. Das *Simple Object Access Protocol* (SOAP) ermöglicht dabei das standardisierte Durchführen von RPCs, mit Hilfe von UDDI können Web Services gefunden werden. WSDL dient schließlich der Beschreibung der Web Services [Che01, OM01, Jep01, Sta01, Udd00, Ogb00].

Web Services dienen unter anderem dem automatischen Datenaustausch zwischen Softwaresystemen. Die für den Austausch benötigten Datenformate basieren vorwiegend auf dem XML-Standard. XML-basierte Datenaustauschformate lassen sich wie folgt in drei verschiedene Kategorien einteilen: Die Kategorie *Frameworks* legt lediglich die Spezifikationen für den strukturierten Nachrichten-/Dokumentenaustausch zwischen verschiedenen Partnern fest. Mit Hilfe der Gruppe *Functionsn* werden branchenunabhängige Geschäftsprozesse definiert, somit sind erste Bausteine vorhanden, um die Nachrichten zu strukturieren. Zum Nachrichtenaustausch innerhalb einer Branche beziehungsweise von einer Versorgungskette werden bei der Kategorie *Verticals* die XML-Vokabulare feingranulierter spezifiziert. Von der ersten bis zur dritten Kategorie erhalten die Austauschformate stets mehr Struktur, wodurch einerseits die Ausdruckfähigkeit abnimmt, andererseits wird durch fortschreitende Standardisierung der globale Austausch von Geschäftsdokumenten entproblematisiert [Whb01].

3.7 Mobile Commerce

In den letzten Jahren gewann Electronic Commerce (E-Commerce), also die Abwicklung von Handel und Dienstleistungen über elektronische Medien, enorm an Bedeutung. Eine spezielle Variante des ECommerce stellt Mobile Commerce (M-Commerce) dar. Der Konsument benutzt hierbei ein mobiles Endgerät wie z. B. ein Mobiltelefon oder einen PDA, um Geschäfte online abzuwickeln. Diese Mobilität bietet dem Konsumenten hohe Flexibilität durch Ortsunabhängigkeit.

Die weite Verbreitung mobiler Endgeräte stellt bereits eine optimale Ausgangsbasis für M-Commerce dar. Die Anwendungsmöglichkeiten sind aber aufgrund der momentan verfügbaren Technologien noch sehr eingeschränkt. Einerseits bieten die auf dem GSM- und GPRS-Standard basierenden Mobilfunknetze zu geringe Datenübertragungsraten, andererseits zeichnen sich insbesondere WAP-Mobiltelefone durch mangelnde Darstellungsmöglichkeiten aus. Erst mit Zukunftstechnologien wie UMTS kann das Potential von M-Commerce voll ausgeschöpft werden. Besonders aufgrund der zukünftig möglichen Übertragung von Bild- und Videodaten ist hier eine Vielzahl an neuen Dienstleistungen zu erwarten.

Ein Beispiel für einen Bereich, in dem bereits Dienstleistungen angeboten werden, ist Mobile Payment (M-Payment). Hierbei gibt es verschiedene Ansätze, um das Mobiltelefon als Zahlungsmittel zu etablieren. Der deutsche Anbieter „Paybox“ sowie der französische Anbieter „ItiAchat“ bieten Lösungen an, die sich aber in ihrer Handhabung deutlich voneinander unterscheiden. Andere Anbieter wie z. B. „Payitmobile“ befinden sich noch im Teststadium [Kie01].

Ein weiterer wichtiger Dienstleistungsbereich ist Mobile Banking (M-Banking). Dieser soll nach Aussagen von Analysten die Bankgeschäfte zukünftig revolutionieren. Bereits Anfang der 90er Jahre wurde von mehreren großen Finanzinstituten wie z. B. der Dresdner Bank der Aktienhandel via SMS angeboten, allerdings ohne großen Erfolg. Ein wesentlich

erfolgreicheres Beispiel ist die auf Java basierende Applikation „youtrade on Palm“, die vom Schweizer Finanzinstitut Credit Suisse eingesetzt wird. Hierdurch hat der Kunde die Möglichkeit, seine Aktiengeschäfte via PDA auszuführen [Mus02].

3.8 Sicherheit

Sicherheit in offenen Netzen wird in Zukunft ohne die Verwendung kryptographischer Verfahren nicht zu gewährleisten sein. Das technologische Know-how sowie hardware- und softwarebasierende Verfahren zum Schutz der Vertraulichkeit, der Integrität und der Zuordnungsfähigkeit von über Netze zu übermittelnden Nachrichten und zu speichernden Daten ist vorhanden. In der Telekommunikation kann Verschlüsselungstechnik von den Betreibern der Dienste eingesetzt werden, beispielsweise im Mobilfunknetz. Es besteht aber auch die Möglichkeit, dass die Teilnehmer ihre zu übermittelnden Informationen selbst schützen, indem sie entsprechende Verschlüsselungs- und Signaturtechniken einsetzen. In dem Vortrag wurde der Schwerpunkt auf die Kryptographie gesetzt. Es wurden zuerst Verfahren der Kryptographie vorgestellt [Bro02], danach folgten als Beispiele zwei Algorithmen für das symmetrische Verfahren, nämlich DES (Data Encryption Standard) und IDEA (International Data Encryption Algorithm) und einen Algorithmus für das asymmetrische Verfahren, RSA (Rivest, Shamir und Adleman) und deren Ergebnisbeurteilung [RSA02]. Dann wurden die digitale Signatur sowie die Verfahren zur Erstellung und Überprüfung von Signaturen behandelt. Danach wurde ein Überblick über Zertifikate und deren Ausstellung gegeben. Im Weiteren wurden die Authentifizierung und im Anschluss Anwendungen wie SSL (Secure Socket Layer) und SHTTP für kryptographische Systeme präsentiert [Kro96].

3.9 E-Bill Anwendungen

Electronic Billing Presentment and Payment –Systeme (EBPP) sind eine Applikation oder eine Menge von Applikationen, die den Ablauf der Transaktion zwischen Rechnungssteller und Rechnungsempfänger auf elektronischem Wege ermöglichen. Diese Systeme sollen Unternehmen und Endverbrauchern die Möglichkeit bieten, Rechnungen über ein elektronisches, einfach zugängliches und bedienbares Medium zur Verfügung zu stellen, zu überprüfen und zu bezahlen.

Um ein solches System realisieren zu können, werden hohe Anforderungen an die Sicherheit und Effizienz unter Bezugnahme der gesetzlichen Regelungen und Vorschriften gesetzt. Für die Realisierung der Systeme werden 3 unterschiedliche Modelle herangezogen: Buyer Direct-, Thick Consolidator- und Thin Consolidator-Modell.

Direct Billing-Systeme sind sehr Anwender-spezifisch und nur mit hohem Aufwand an unterschiedliche Rahmenbedingungen anzupassen. Hierbei agiert in der Regel ein Unternehmen als Rechnungssteller nur für seine eigenen Kunden.

Systeme, die dem Buyer Direct-Modell folgen, haben sich Marktstudien zufolge bisher nicht durchsetzen können. Hierbei wird ein funktionierendes EBPP-System auf der Seite des Rechnungsempfängers vorausgesetzt. Bislang sind derartige Lösungen ausschließlich im B2B-Bereich anzutreffen, da ihre Realisierung aufgrund hoher Kosten auf den herkömmlichen Endkundenbereich nicht übertragbar ist.

Nur das Consolidator-Modell verspricht eine Realisierung, die viele verschiedene Parteien zufrieden stellen könnte. Bei diesem Modell wird, wie bereits erwähnt, zwischen dem Thick und dem Thin Consolidator unterschieden. Hierbei agiert eine dritte Firma als Vermittler zwischen den Rechnungsempfängern auf der einen und den Rechnungsstellern auf der anderen Seite (Consolidator). Das Thick Consolidator Modell sieht eine vollständige Speicherung der Rechnungsdaten auf dem Consolidator-System vor, wohingegen das Thin Consolidator Modell die Aufbewahrung der detaillierten Rechnungsdaten bei dem

Rechnungssteller voraussetzt. Der Consolidator bietet den Rechnungsempfängern in diesem Fall nur eine grobe Rechnungsbersicht, wobei der Rechnungsempfänger auf Wunsch über einen Verweis auf die Seiten des Rechnungsstellers die restlichen Details seiner Rechnungen einsehen kann. Auf diesem Weg bleibt der Direktkontakt zwischen einem Unternehmen und seinen Kunden erhalten.

Die Entwicklung des EBPP begann als eine Menge von Einzellösungen unterschiedlicher Anbieter, die diese Systeme als Direktanbieter zum Eigenzweck entwickelten. Aufgrund der Nähe zu den elektronischen Medien waren diese Unternehmen hauptsächlich in der Internet- und Telekommunikationsbranche tätig. Die steigende Bedeutung des E-Commerce lässt ebenso dem EBPP eine größere Bedeutung zukommen, da es logisch erscheint, auf elektronischem Wege erworbene Waren über den gleichen Weg ohne einen Medienbruch, also der Rechnung auf Papier, zu bezahlen.

Da die Wirtschaftlichkeit dieser Systeme stark von der Masse der sie nutzenden Kunden abhängig ist, stellen Benutzerfreundlichkeit, Effizienz und Funktionsvielfalt auf der einen, Akzeptanz und Vertrauen der Kunden auf der anderen Seite die entscheidenden Faktoren bei der Durchsetzung und Standardisierung des jeweiligen Systems dar. Als durchsetzungsfähigstes Modell der EBPP-Systeme wird von Fachleuten das Thin Consolidator Modell angesehen, denn dieses bietet allen Beteiligten die größte Flexibilität. Der Kunde kann also seine Rechnungen zentral abarbeiten, während die Unternehmen den überlebenswichtigen Draht zu ihnen behalten.

4 Das Modell für Com42Bill

Der Vergleich der drei Modelle zur Realisierung eines EBPP-System (siehe Kapitel 3.9) 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 in der Entwurfsphase 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 besteht ferner darin, dass der direkte Kontakt des Rechnungsstellers zum Rechnungsempfänger durch ein System dieser Art nicht unterbrochen wird. 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.

Die an Com42Bill–Transaktionen beteiligten Parteien lassen sich in vier Gruppen einteilen, die in Abbildung 3 zusammen mit ihrer Relation zum Betreiber von Com42Bill dargestellt sind.

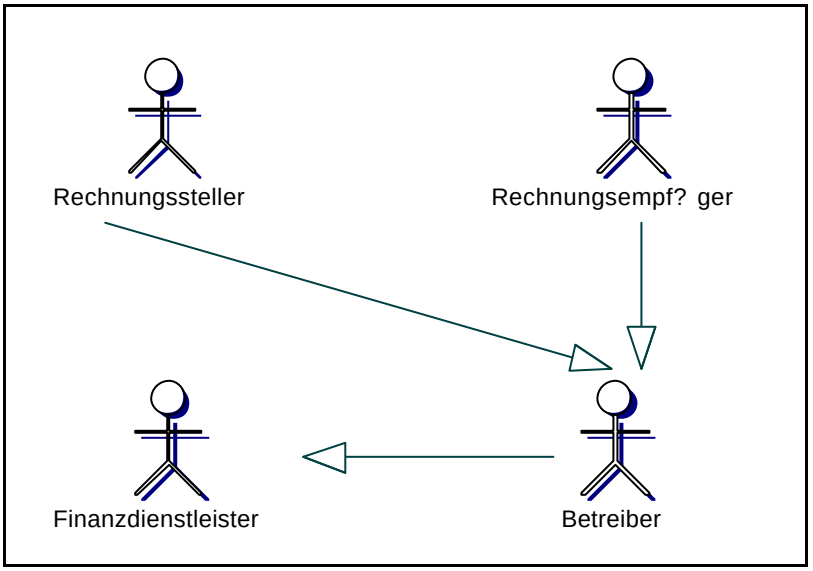


Abbildung 3: Rollenmodell

Der **Rechnungssteller** sollte die Möglichkeit erhalten, sich über eine Weboberfläche jederzeit über den Status seiner Transaktionen informieren zu können. Von dieser Idee wurde vor allem mangels Zeit abgesehen. Der Ausweg bestand darin, die benötigten Informationen über die ebXML-Schnittstelle des Datenkonverters an den Rechnungssteller zu schicken.

Somit beschränkt sich die Schnittstelle zu dem Com42Bill-System hauptsächlich auf die Kommunikation über die genannte standardisierte Schnittstelle. Dadurch erhält der Rechnungssteller folgende Datenaustauschszenarien:

1. Er ist in der Lage, die entsprechenden Rechnungsdaten zur weiteren Verarbeitung an Com42Bill zu übergeben.
2. Die Statusänderung des Bezahlungsvorgangs wird simultan vom Rechnungssteller an Com42Bill weitergegeben, wodurch der Rechnungsempfänger sich jederzeit über den Status seiner Rechnungen informieren kann.

Solche Szenarien sind vollständig implementiert worden. Die Registrierung des Rechnungsstellers bei Com42Bill geschieht ausschließlich offline mit Unterzeichnung eines gemeinsamen Vertrags zwischen den beiden Parteien. Dabei werden alle geschäftsrelevanten Daten ausgetauscht, die eigentliche Freischaltung in dem Com42Bill-System geschieht durch das Anlegen eines neuen Rechnungsstellers auf der Administratoroberfläche. Diese Aufgabe obliegt somit dem Betreiber von Com42Bill.

Die **Rechungsempfänger** sollten ursprünglich die Möglichkeit erhalten, sowohl über Web- als auch W@P-Schnittstellen mit Com42Bill zu arbeiten. Aufgrund hoher Anforderungen an die webbasierte Oberfläche blieb letztendlich keine Zeit mehr, die Handyvariante zu integrieren. Aus heutiger Sicht der rasanten Technologieentwicklung wäre eine WAP-Implementierung sogar als Fehlgriff zu betrachten, wenn man bedenkt, dass viele vielversprechende WAP-Projekte gescheitert sind. Über eine Mobillösung sollte erst im Zuge der Einführung von UMTS nachgedacht werden.
Die Rechnungsempfänger können folgende Abläufe über die Webseite tätigen:
Um seine persönlichen Rechnungsdaten einzusehen oder die Bezahlung seiner Rechnungen zu veranlassen, braucht der Rechnungssteller sich lediglich mit seinem Benutzernamen und seinem Com42Bill-Passwort anzumelden, wonach er sein persönliches Com42Bill-Portal erreicht. Die Voraussetzung hierfür ist die einmalige Registrierung bei dem Betreiber von Com42Bill. Diese geschieht direkt über die Weboberfläche. Der Benutzer wird dabei ausführlich über jeden Registrierungsschritt informiert und kann bei Bedarf auf die ausreichend vorhandenen Hilfestellungen zurückgreifen.
Weiterhin kann sich ein Rechnungsempfänger per E-Mail über Fälligkeiten oder neu eingetroffene Rechnungen informieren lassen. Zusätzlich war die Möglichkeit vom SMS-Versand zu denselben Zwecken angedacht, die hohen Anschaffungskosten eines SMS-Gateways hinderten uns leider daran, diese Idee umzusetzen. Diese Option ist jedoch ein Bestandteil der Komponente Business Logic und kann bei Bedarf problemlos aktiviert und parallel zur E-Mail-Benachrichtigung eingesetzt werden.

Die **Finanzdienstleister** führen die Zahlungsaufträge, die von den Rechnungsempfängern erteilt werden, aus. Die Voraussetzung für eine erfolgreiche Zusammenarbeit mit dem Com42Bill-System ist die Implementierung der ebXML-Schnittstelle von Seiten des Finanzdienstleisters, welche standardmäßig über die Datenkonverter-Komponente angeboten wird. Man soll an dieser Stelle bedenken, dass dieser Standard immer noch Zukunftsmusik bleibt, man jedoch fest von der Annahme ausgeht, dass immer mehr Finanzdienstleister diesen Standard akzeptieren und in ihre Kommunikationsschnittstellen integrieren werden. Somit bleibt auch Com42Bill sehr gut für die Zukunft des elektronischen Zahlungsverkehrs gerüstet. Was den konkreten Datenaustausch zwischen den Finanzinstituten und Com42Bill betrifft, so lässt sich dieser folgendermaßen zusammenfassen:

Bei einer zur Bezahlung freigegebenen Rechnung gelangen die Informationen über die Komponente Business Logic zum Datenkonverter. Dieser seinerseits bereitet die Rechnungsdaten für den Austausch über die genannte ebXML-Schnittstelle auf und verschickt diese an das entsprechende Finanzinstitut. Die Statusabfrage ist jederzeit möglich, bei erfolgten Transaktionen oder evtl. aufgetretenen Fehlern informiert der Finanzdienstleister die Com42Bill-Gegenstelle umgehend. Dieser wiederum gibt die Daten in Com42Bill-interner

Form an die Businesslogic-Komponente weiter, welche wiederum die GUI-Komponente (also den Rechnungssteller) mit neu eingetroffenen Daten konfrontiert.

Somit wurden alle am Anfang festgesetzten Ziele der Kommunikation zwischen dem Finanzdienstleister und Com42Bill weitestgehend realisiert.

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.

Die Administrationsoberfläche bietet dem Betreiber eine einfache und sehr bequeme Möglichkeit, beinahe alle Parameter des Systems zu verwalten, sowie interne Abläufe zu kontrollieren. Anfangs war zusätzlich angedacht, eine Art statistische Auswertung der Geschäftsprozesse zu implementieren. Es stellte sich jedoch heraus, dass der Umfang dieser Aufgabe so enorm ist, dass man durchaus eine weitere Projektgruppe lange Zeit damit beschäftigen könnte. Abgesehen von dieser Ausnahme wurden alle Anforderungen umgesetzt. Somit erhält der Betreiber ein mächtiges Werkzeug zur Verwaltung seines gesamten Systems.

5 Die Architektur

5.1 Allgemein

Die Systemarchitektur gibt eine Übersicht über die einzelnen Komponenten, welche gemeinsam das EBPP-System bilden. Bereits in der Seminarphase wurde eine Aufteilung in fünf Komponenten beschlossen, welche sich auch in Personengruppen, den Komponententeams, widerspiegeln. 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 den Überlegungen der Teilnehmer entstanden, dass es im Rahmen einer Projektgruppe 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. Nach Ansicht der PG-Teilnehmer überwiegen die Vorteile eines solchen Einsatzes gegenüber dem massiven Einarbeitungsaufwand, der nahezu alle Komponenten betrifft. Im Endeffekt hat sich gezeigt, dass diese Entscheidung richtig war. Es mussten zwar während der gesamten Implementierungsphase immer wieder neuartige Probleme im Umgang mit dem Applikationsserver gelöst werden, es hat jedoch jeder eine Menge Erfahrungen sammeln können, die einem evtl. auch im späteren Berufsleben noch zugute kommen werden. 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.

Es war sehr interessant zu sehen, wie unterschiedlich die ursprüngliche Architektur von den Komponententeams umgesetzt wurde. Von sehr vielen Zehn-Zeiler-Klassen bis zu Klassen mit 500 Zeilen ist alles dabei. Teilweise wurden die Subkomponenten direkt in Klassen umgesetzt. Daher werden die Komponenten vermutlich mit sehr unterschiedlichem Aufwand auf

Änderungen reagieren können. Die Anpassbarkeit an neue Anforderungen können wir „leider“ nicht mehr erproben.

Im Folgenden werden die Teilarchitekturen der einzelnen Komponenten vorgestellt.

5.2 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.

Die Komponente GUI weist folgende Architekturübersicht auf:

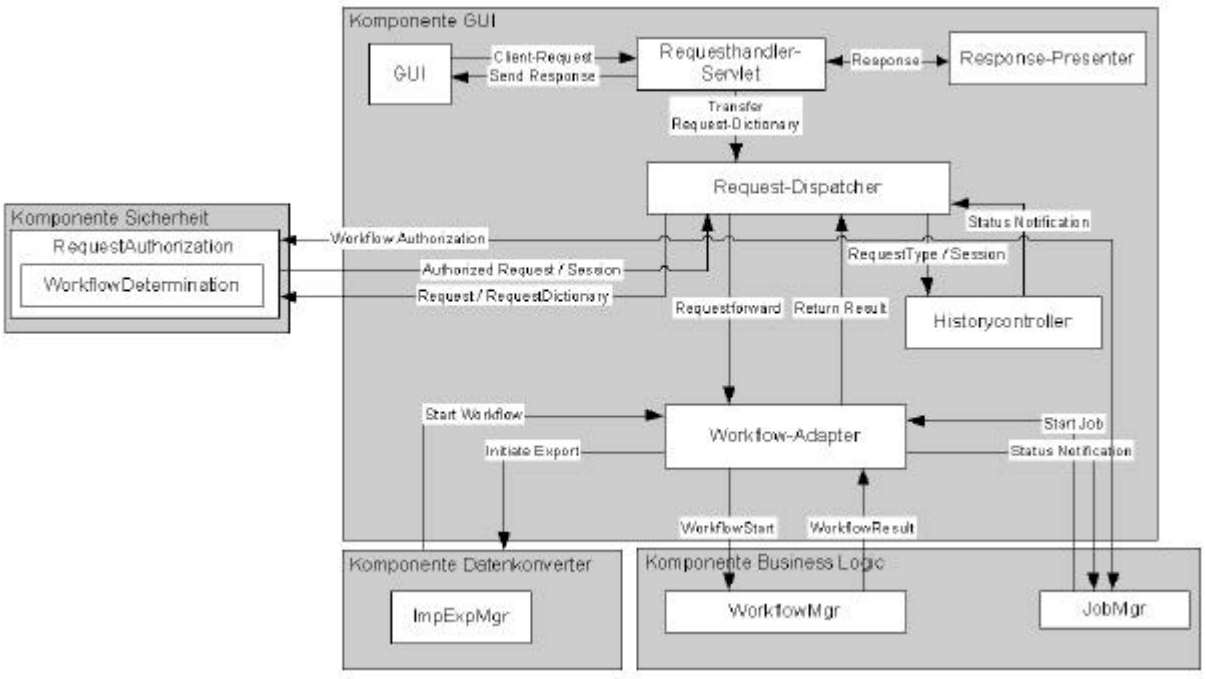


Abbildung 4: Teilarchitektur der Komponente GUI

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.

5.3 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 5 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.

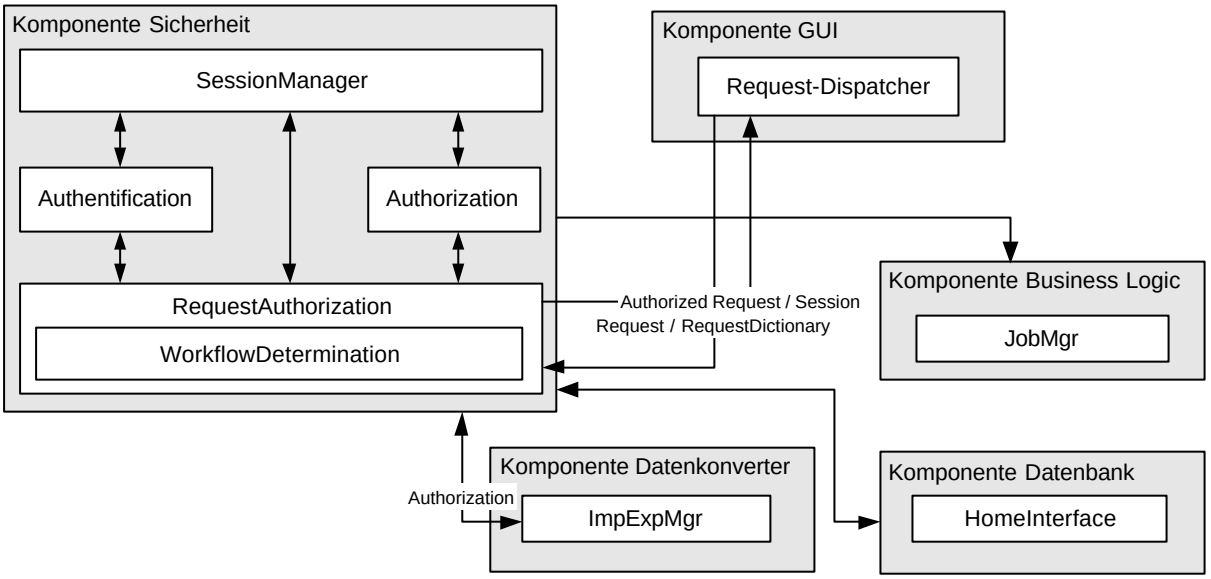


Abbildung 5: Teilarchitektur der Komponente Sicherheit

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.

5.4 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.

5.4.1 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 6 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.

Im Folgenden wird auf die technische Umsetzung eingegangen.

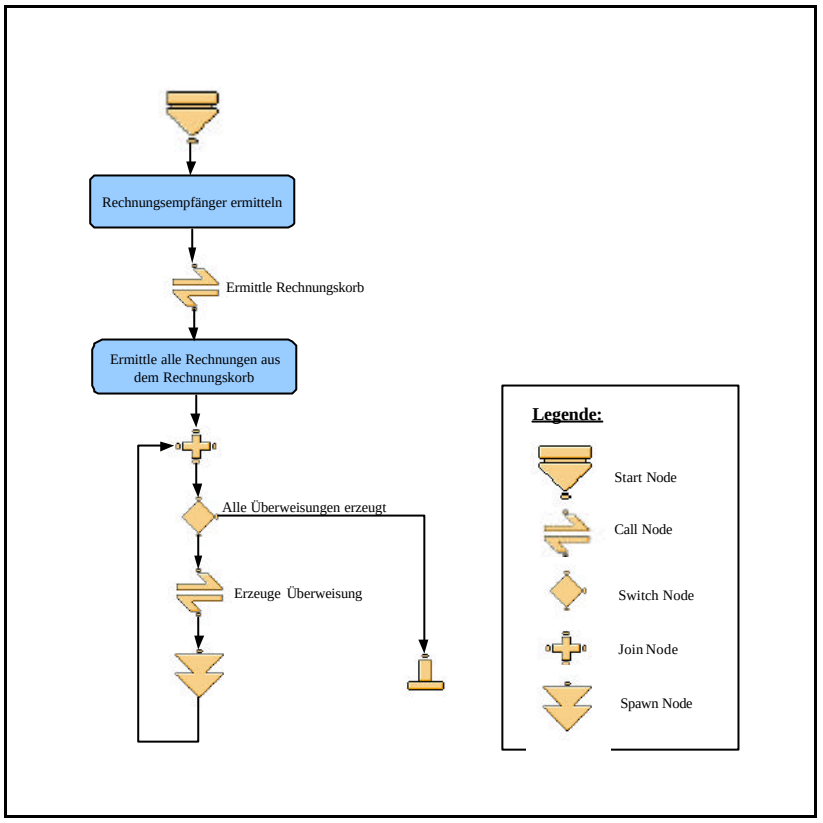


Abbildung 6: Beispiel-Workflow

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.

Im Folgenden werden die verfügbaren Nodes erläutert.

5.4.1.1 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.

5.4.1.2 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.

5.4.1.3 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.

5.4.1.4 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.

5.4.1.5 SwitchCaseNode

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

5.4.1.6 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.

Die nachfolgende Tabelle gibt einen Überblick über die möglichen Conditions.

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

5.4.1.7 SwitchDefaultNode

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

5.4.1.8 JoinNode

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

5.4.1.9 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.

5.4.1.10 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.

5.4.2 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.

5.4.2.1 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.

5.4.2.2 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.

5.4.2.3 Faults

Die bereits beschriebenen OnFault-Nodes werden in WorkflowContexts deklariert.

5.4.3 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.

Im Anschluss werden die Subkomponenten kurz erläutert.

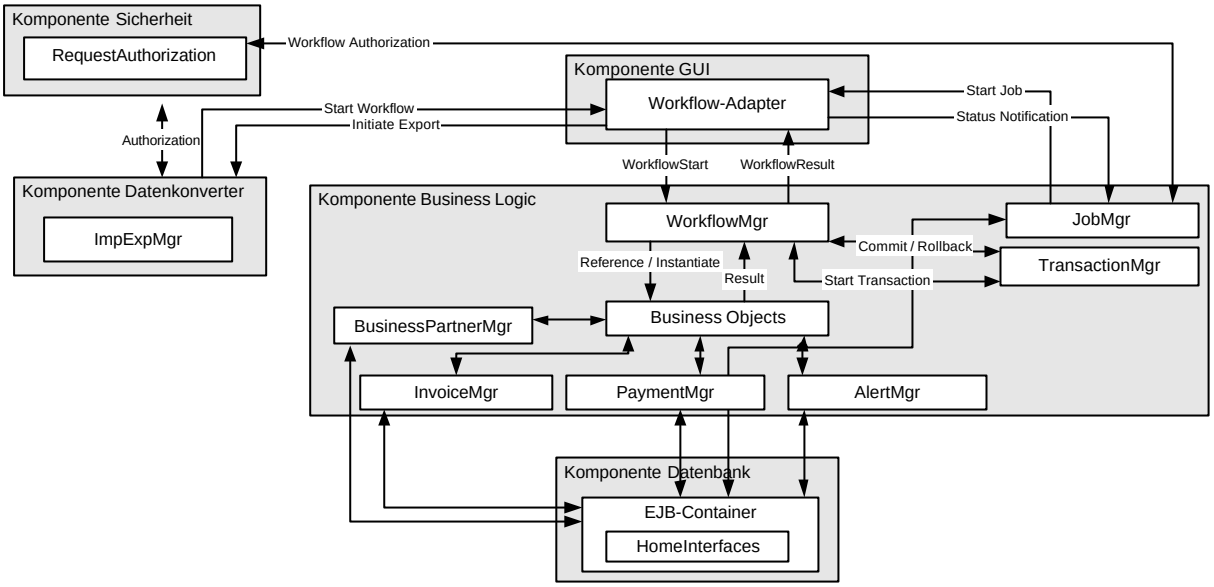


Abbildung 7: Teilarchitektur der Komponente Business Logic

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.

5.5 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.

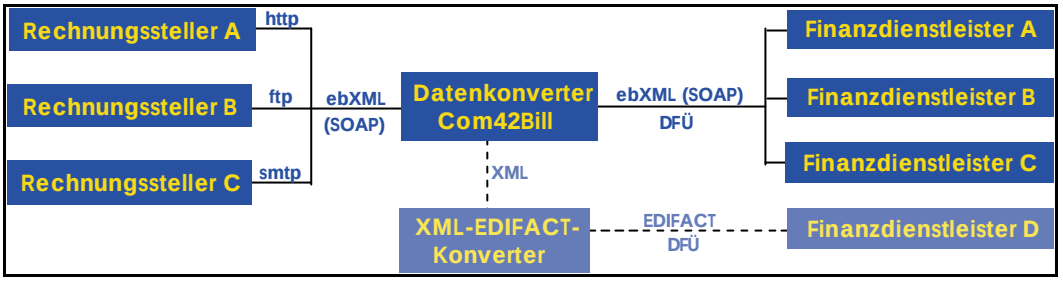


Abbildung 8: Kommunikation von Com42Bill mit anderen Systemen

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, der in Kapitel 5.5.1 näher beschrieben wird.

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 8).

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.

5.5.1 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).

5.5.1.1 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.

5.5.1.2 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.

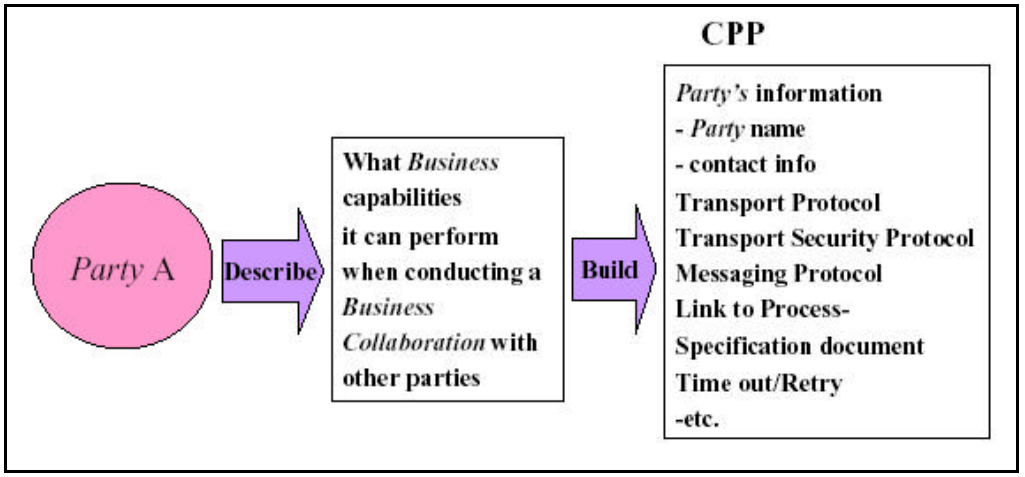


Abbildung 9: Collaboration Protocol Profile (CPP)

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.

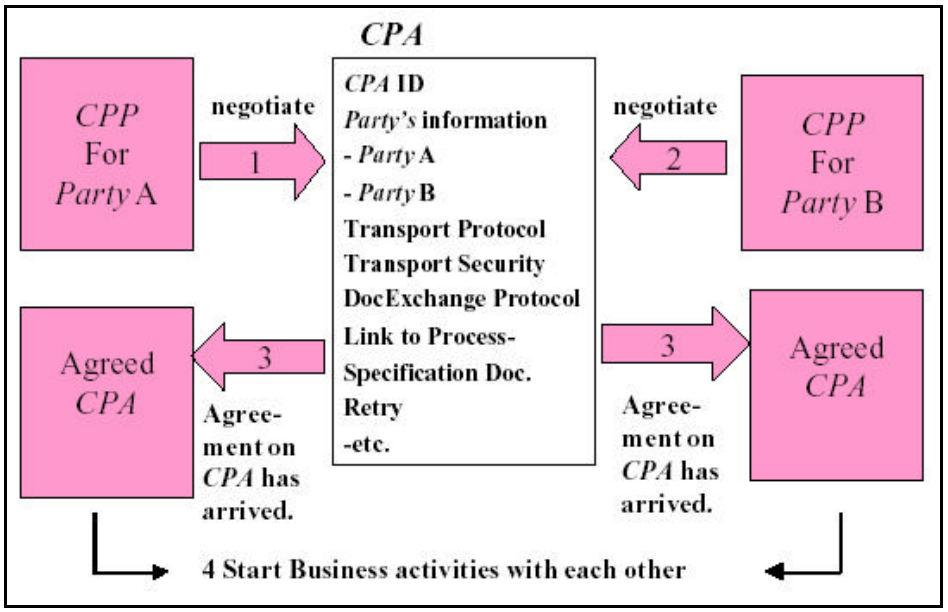


Abbildung 10: Collaboration Protocol Agreement (CPA)

Business Process Specification Schema (BPSS):

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

5.5.1.3 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)

5.5.1.4 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.

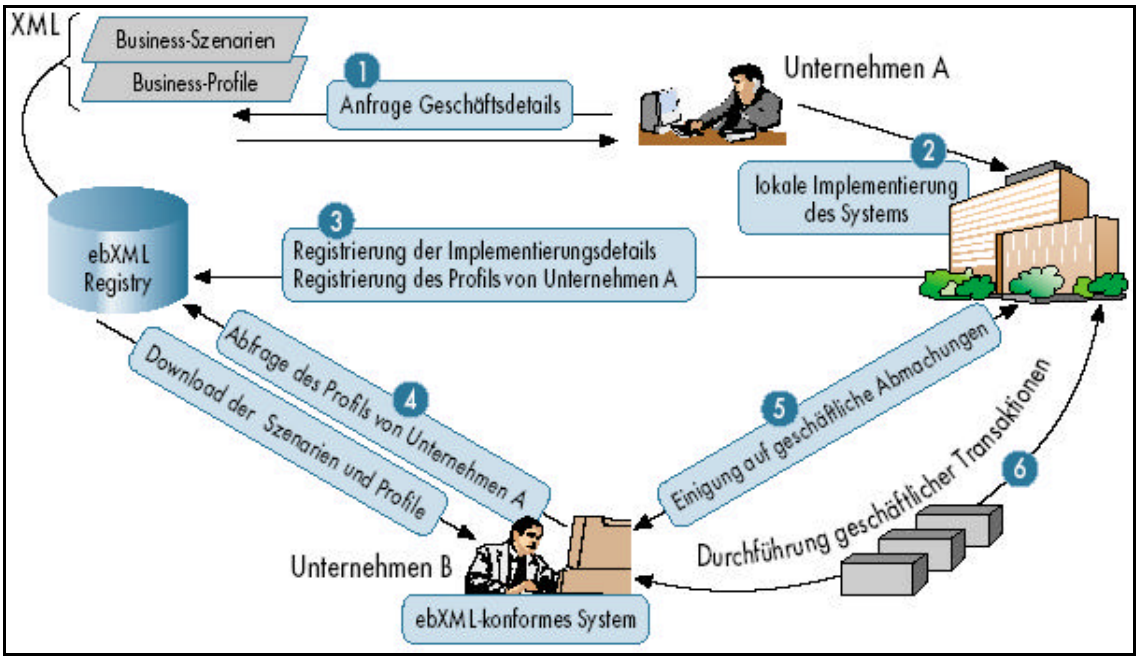


Abbildung 11: Beispielszenario

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.

5.5.1.5 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.

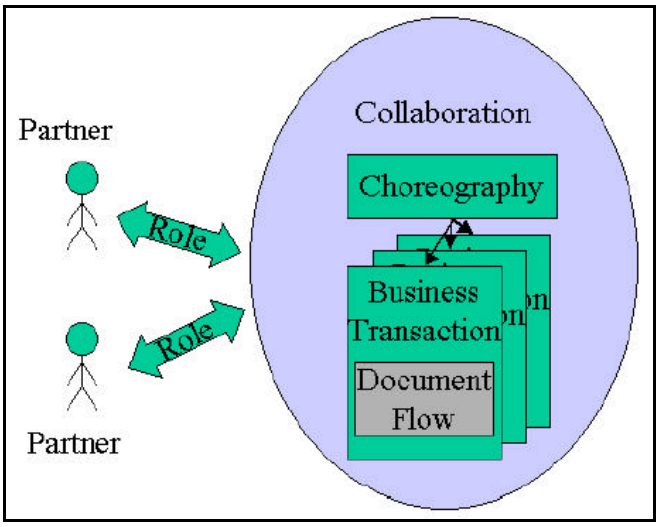


Abbildung 12: Struktur von Geschäftstransaktionen

Die Struktur von Geschäftstransaktionen in der ebXML-Welt wird in Abbildung 12 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.

5.5.1.6 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

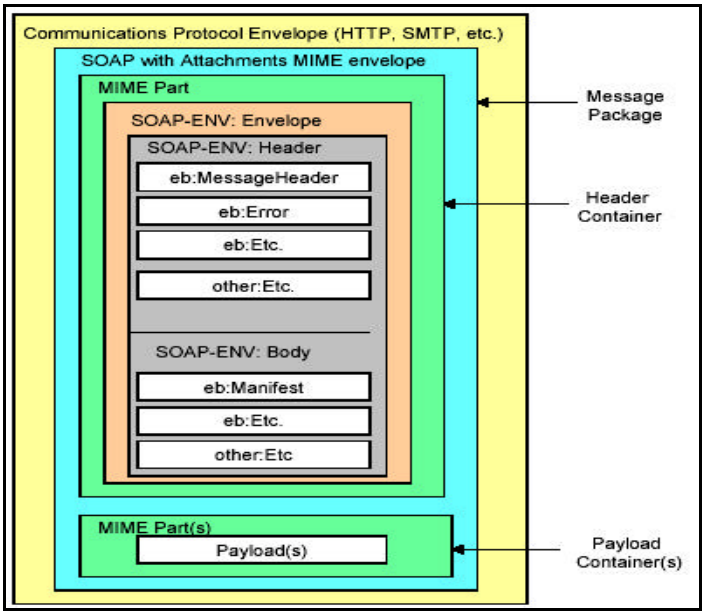


Abbildung 13: Struktur einer ebXML-Nachricht

Für die Verpackung der Geschäftsdaten wird eine Erweiterung von SOAP 1.1 (Simple Object Access Protocol), welches ebenfalls auf XML basiert, verwendet (siehe Abbildung 13).

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.

5.5.2 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 eine Projektgruppe 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 einer Projektgruppe 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.

5.5.2.1 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.

Die Dokumente befinden sich im Anhang C.

5.5.2.2 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.

Das BPSS befindet sich in Anhang C.

5.5.2.3 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.

Die XML-Schemata sind im Anhang C abgebildet.

5.5.3 Architektur

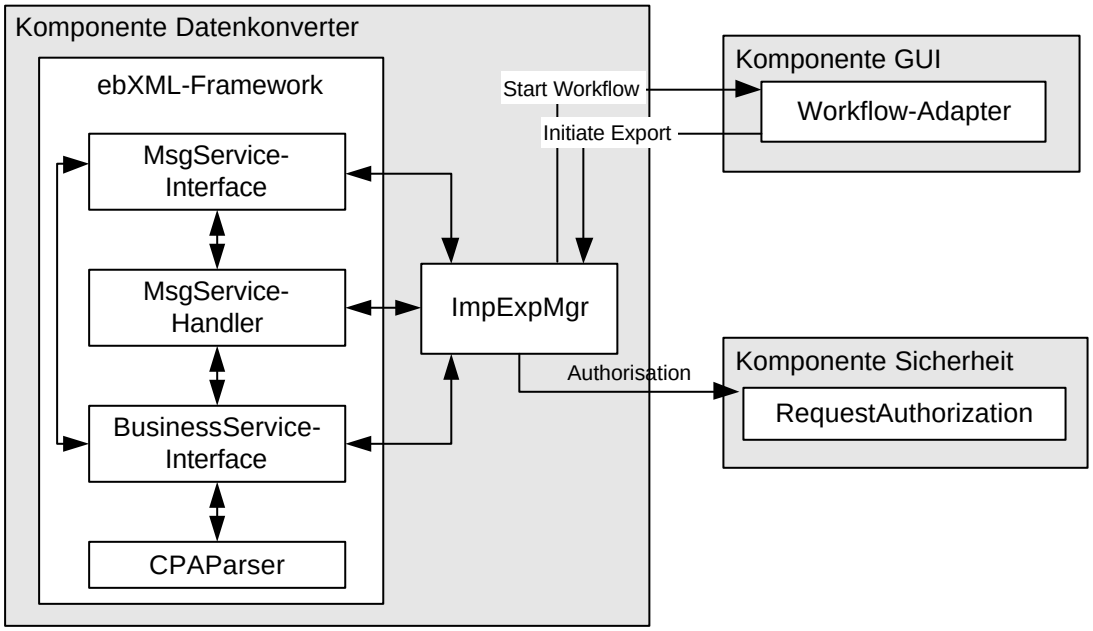


Abbildung 14: Teilarchitektur der Komponente Datenkonverter

Der Datenkonverter besteht aus den folgenden Subkomponenten (siehe Abbildung 14).

5.5.3.1 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.

5.5.3.2 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.

5.5.3.3 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.

5.5.3.4 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.

5.5.3.5 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.

5.6 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 15 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.

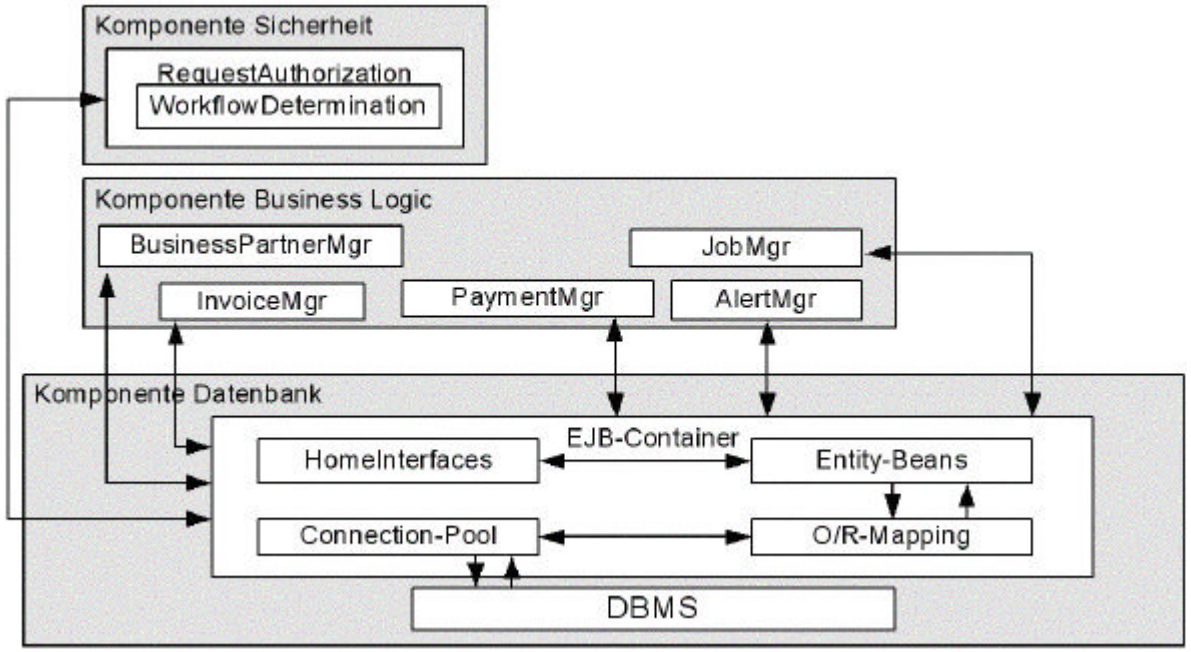


Abbildung 15: Teilarchitektur der Komponente Datenbank

6 Funktionalität

6.1 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.

Die einzelnen Anforderungen, die mit dem System umgesetzt werden sollten, können im Zwischenbericht [C42B02_2] dieser Projektgruppe nachgelesen werden.

Im Folgenden werden die Funktionen von Com42Bill beschrieben; geplante, aber nicht umgesetzte Funktionen werden im Text erwähnt.

6.2 Funktionen der Benutzeroberfläche

6.2.1 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 16: 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 17: Popup-Menü

6.2.2 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 18: Aktuelles

6.2.3 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 19: MyCom42Bill

6.2.4 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 20: Login

6.2.5 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 21: 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.

6.2.6 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.

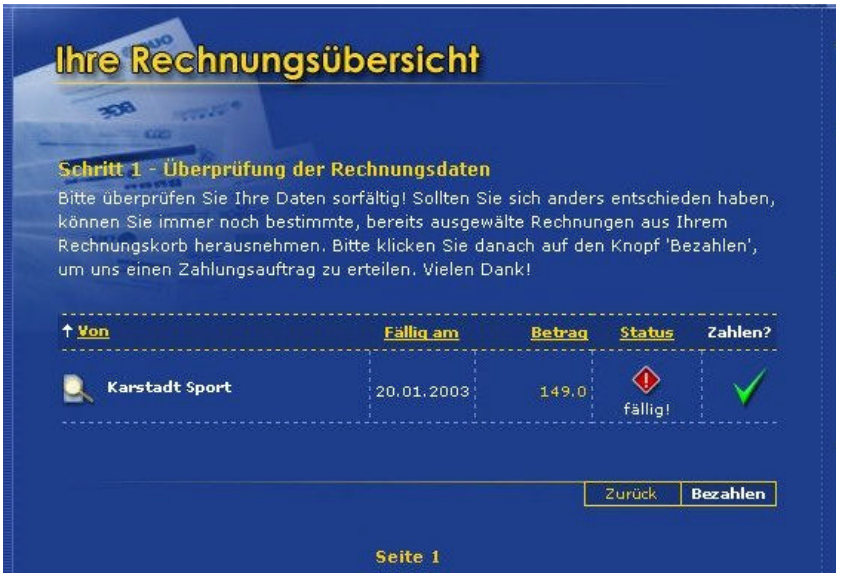


Abbildung 22:Bezahlvorgang – Schritt 1

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 23: 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.

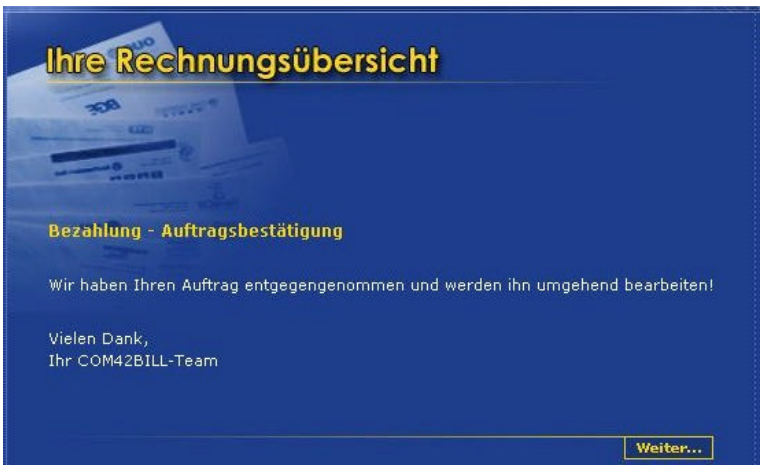


Abbildung 24: Bezahlvorgang – Bestätigung

6.2.7 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 25: Persönliche Daten ändern

6.2.8 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 26: 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 27: Kontodaten ändern

6.2.9 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 28: Logout

6.2.10 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 29: Registriervorgang – Erklärung

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

Abbildung 30: 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 31: Registriervorgang – Wahl des Logins

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

Abbildung 32: 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 33: Registriervorgang - Kontrollansicht

Abschließend wird dem Benutzer die erfolgreiche Registrierung bestätigt.

6.2.11 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 34: Support

6.2.12 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.

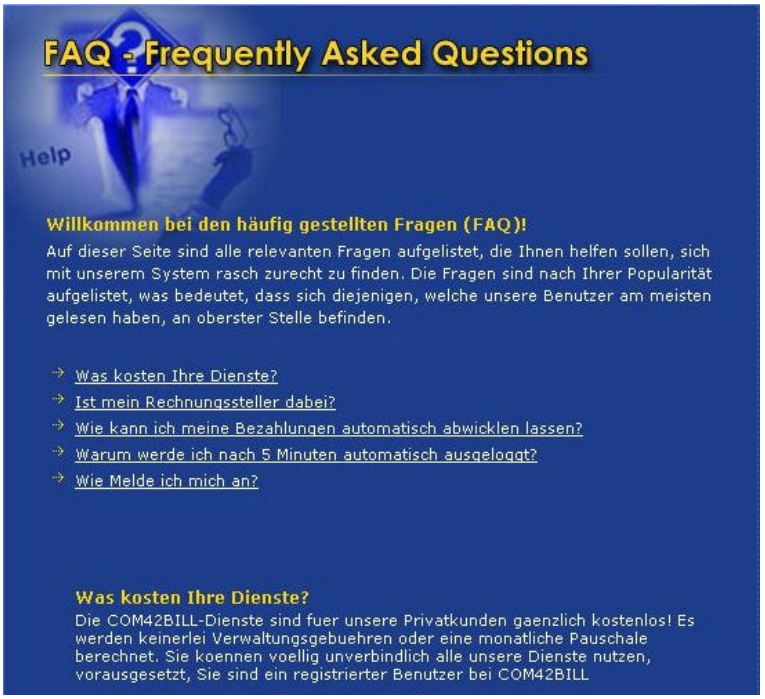


Abbildung 35: FAQ

6.2.13 Impressum

Das Impressum stellt in Kurzform die Projektgruppe 411 vor. Neben dem Zeitraum der Entwicklung werden die Ziele und gewünschten Ergebnisse der Projektgruppe vorgestellt. Bei einem „Echteinsatz“ würde hier das nach §6 Teledienstegesetz (TDG) geforderte Impressum stehen [TDG02].

6.2.14 Kontakt

Auf der Kontaktseite findet man Informationen zu den Betreuern der Projektgruppe sowie die Emailadresse der Entwicklergruppe von Com42Bill.

6.3 Funktionen der Administrationsoberfläche

6.3.1 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:

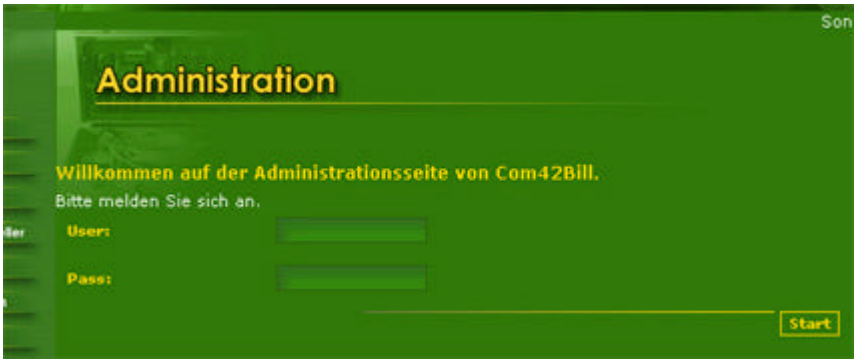


Abbildung 36: Anmeldung für den Administrator

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:

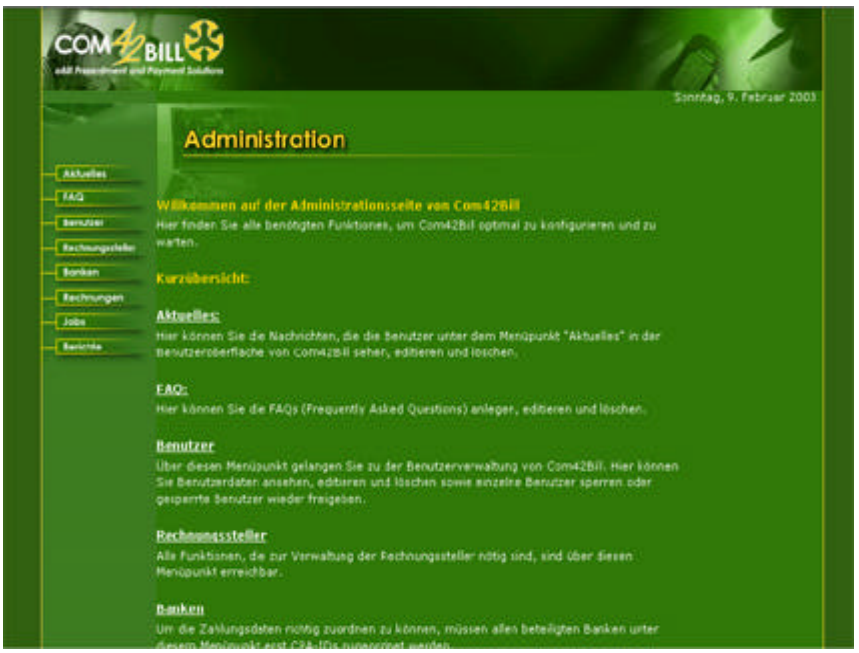


Abbildung 37: Begrüßungsseite für den Administrator

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.

6.3.2 Aktuelles

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

Abbildung 38: 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“.

6.3.3 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:

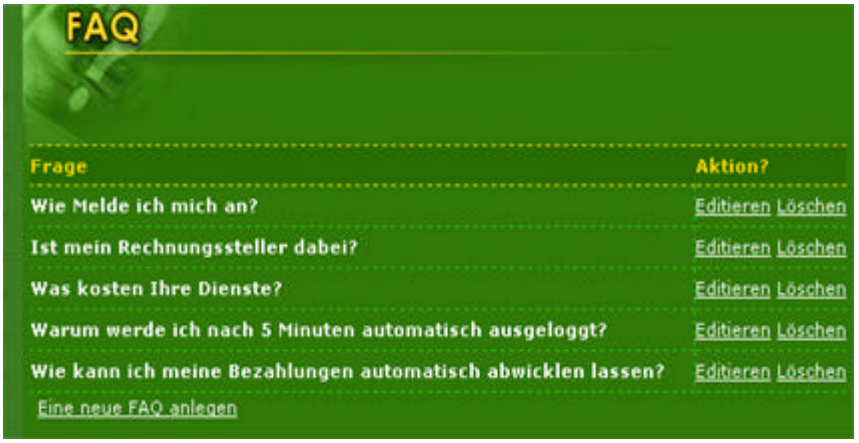


Abbildung 39: FAQ-Übersicht

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 40: 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.

6.3.4 Benutzer

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

Abbildung 41: Benutzerauswahl

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

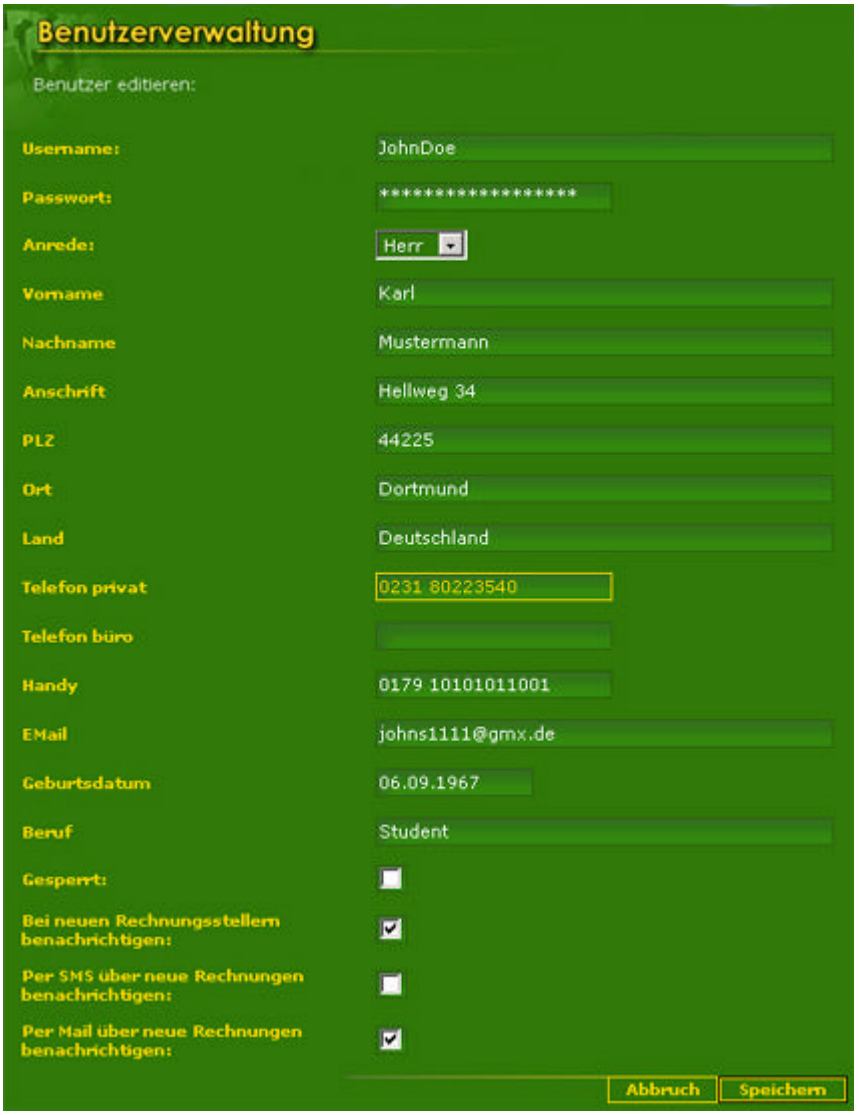


Abbildung 42: Benutzer editieren

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 43: Kontenauswahl

Die Kontodaten können auch geändert werden:

Abbildung 44: 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.

6.3.5 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:

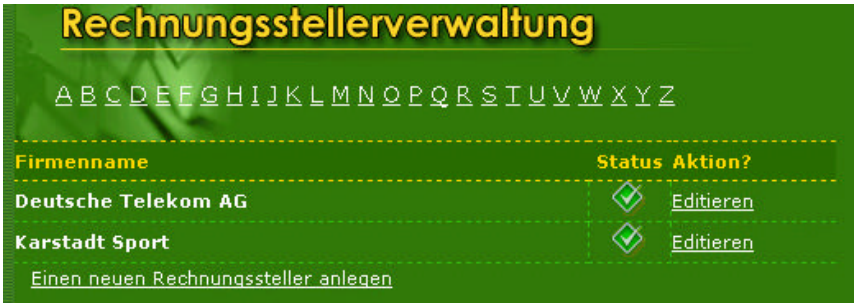


Abbildung 45: Rechnungssteller auswählen

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:

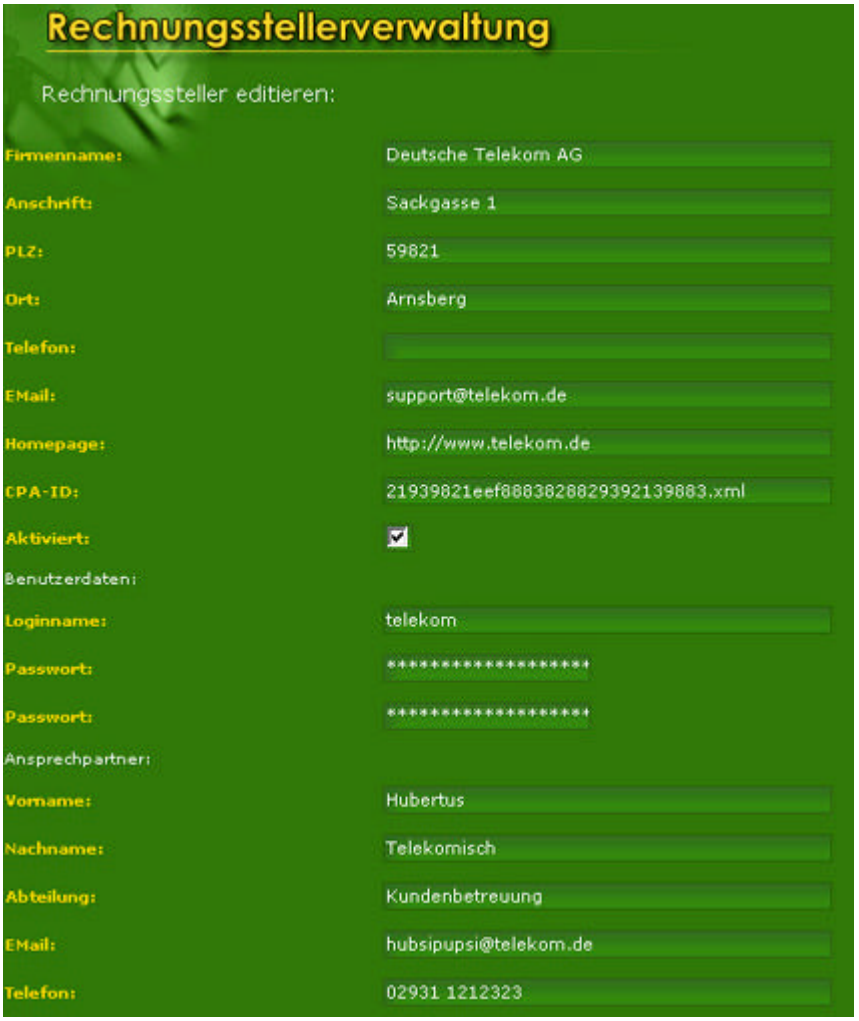


Abbildung 46: Rechnungssteller editieren

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 43). Auch hier können derzeit nur Bankkonten eingegeben werden (das Formular zur Neuerfassung sieht genauso aus; daher wird auf eine Abbildung verzichtet).

Abbildung 47: Finanzkonto editieren

6.3.6 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.

6.3.7 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 48: 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.

6.3.8 Jobs

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

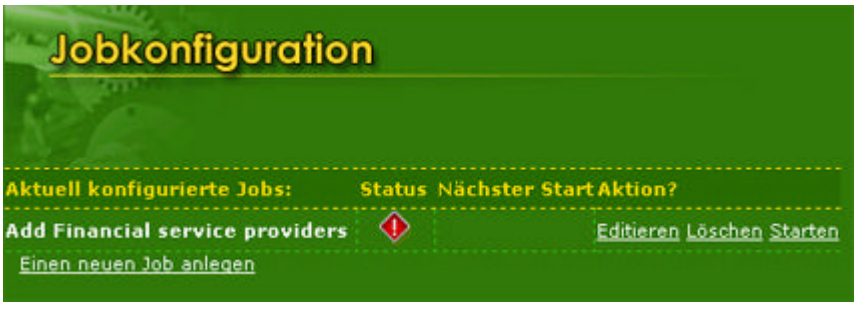


Abbildung 49: Jobübersicht

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 50: 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.

6.3.9 Berichte

Aus Zeitgründen wurde auf eine Implementierung von Statistik- und Berichtsfunktionen verzichtet.

Abbildung 51: Berichte

7 Das Datenmodell

Das Objektmodell umfasst alle persistenten Daten des Systems, die durch Entity Beans abgebildet werden. Das System enthält auch viele Klassen, die nicht auf der Datenbankebene abgebildet werden. Diese Klassen werden daher im Objektmodell nicht dargestellt.

Um einen strukturierten und umfassenden Überblick über das Objektmodell von Com42Bill zu geben, wird zunächst die Package-Struktur des Systems dargestellt. Sie entspricht dem Package-Aufbau des bestehenden Together-Projektes. Das Objektmodell setzt sich aus der Summe der Diagramme aller Packages zusammen.

Ein Package kann weitere Packages oder Klassen enthalten. Packages bilden also Gruppierungen für funktional zusammengehörige Klassen. Die Package-Struktur kann mit einer Baumstruktur verglichen werden, wobei jede Klasse ein Blatt ist. Die Komponenten des Systems werden zunächst durch Packages grob dargestellt. Eine verfeinerte Darstellung erfolgt dann auf Klassenebene.

Der Begriff Package wird in diesem Dokument folgenderweise verwendet: Ein Sub Package ist immer ein Bestandteil eines übergeordneten Packages. Ein Sub Package ist ebenfalls ein Package. Der Begriff Package ist der Oberbegriff und wird synonym für Sub Packages verwendet. Der Begriff Sub Package wird nur dann explizit verwendet, wenn auf die spezielle Eigenschaft des betroffenen Packages als Bestandteil eines übergeordneten Packages hingewiesen werden soll.

Zu jedem Package existiert ein Diagramm, welches die zu diesem Package gehörenden Entity Beans beinhaltet. Die Methoden der Entity Beans sind in den Diagrammen der Übersichtlichkeit halber ausgeblendet. Die ausgeblendeten Bereiche sind bei jedem Entity Bean-Symbol am unteren rechten Rand durch kleine Symbole gekennzeichnet (vgl. Abbildung 55).

Entity Beans haben häufig Relationen zu Entity Beans aus anderen Packages. Um die Funktion eines Packages im systemweiten Kontext erschliessen zu können, werden in einigen Diagrammen zur Beschreibung eines Packages auch Entity Beans aus fremden Packages abgebildet. Letztere werden dabei als Link dargestellt. Ein Link wird durch ein Pfeil-Symbol an der unteren Kante des entsprechenden Entity Bean-Symbols gekennzeichnet.

Zur Darstellung des Objektmodells wurden Teile der UML-Modellierung in Together 6.0 herangezogen, die insbesondere bei den Beschriftungen in der Package-Notation nicht mit UML 1.4 konform ist.

Ein wichtiges Merkmal des Objektmodells ist die fehlende Vererbung bei den Entity Beans: auf diese muss verzichtet werden, da Vererbung bei Entity Beans der EJB 2.0-Spezifikation nicht unterstützt wird.

Im Folgenden wird das Objektmodell für Com42Bill genauer erläutert:

Abbildung 52 zeigt die grobe Darstellung der hierarischen Together-Projekt-Struktur: Das Super Package de bzw. Com42Bill enthält die beiden Packages ebppsystem und components, die beide wiederum Sub Packages enthalten. Diese werden in den folgenden Kapiteln detailiert beschrieben. Die Relation zwischen den beiden Packages ebppsystem und components repräsentiert Beziehungen zwischen den persistenten Entitäten dieser beiden Packages.

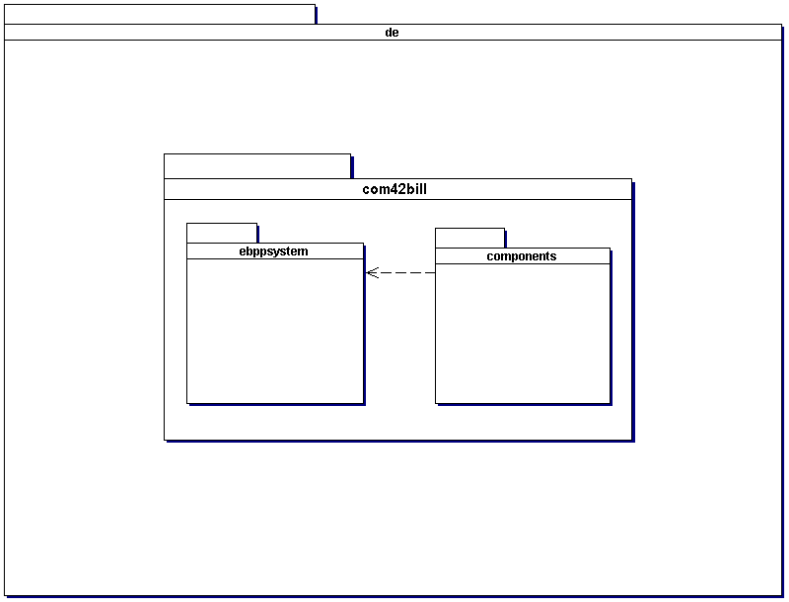


Abbildung 52: Grobe Package-Struktur des Systems

Das Package ebppsystem bildet den zentralen Teil des Objektmodells. Hier sind alle allgemeinen Entity Beans enthalten, auf denen alle Komponenten arbeiten, so wie speziell für die Komponente Business Logic bestimmte Entity Beans. Entity Beans, die jeweils nur von den übrigen Komponenten benutzt werden, sind im Package components zusammengefasst.

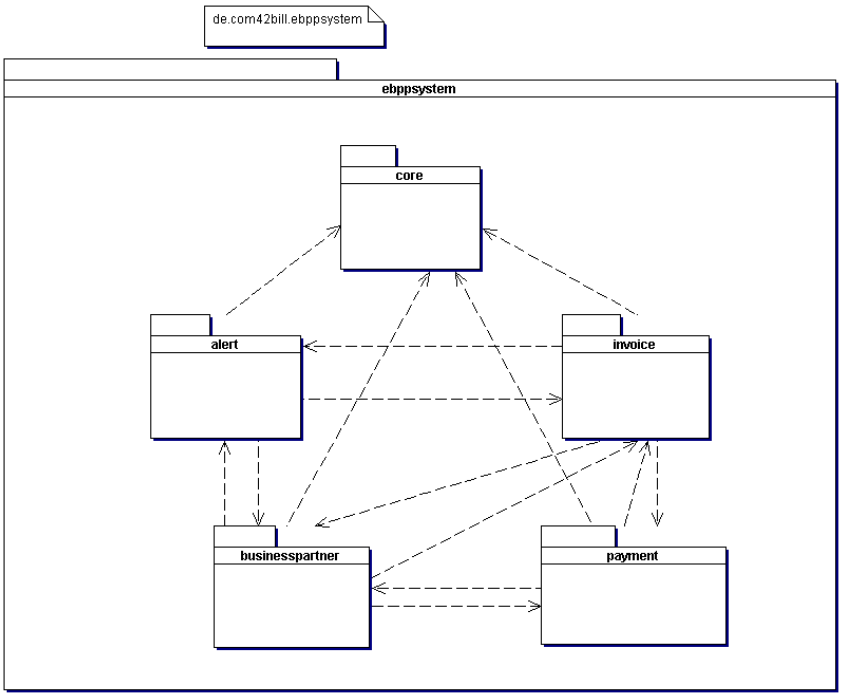


Abbildung 53: Struktur des Package ebppsystem

Wie Abbildung 53 zeigt, enthält das Package ebppsystem die fünf Sub Packages businesspartner, invoice, payment, alert und core. Diese sind an mehreren Punkten miteinander verknüpft. Die durch Pfeile dargestellten Abhängigkeiten korrespondieren mit

den Assoziationen der Entity Beans, die in den unterschiedlichen Sub Packages von ebppsystem enthalten sind.

Eine wichtige Rolle spielt das Package core: es enthält hauptsächlich nicht-fachliche Entity Beans, welche Funktionalitäten des Frameworks bereitstellen.

Das Package invoice enthält alle Entity Beans, die zur Abbildung einer Rechnung benötigt werden. Eine Rechnung wird von einem Rechnungssteller an Com42Bill übermittelt. Mahnungen (im Package alert abgebildet) werden vom System generiert. Rechnungssteller und Rechnungsempfänger werden innerhalb des Package businesspartner abgebildet. Daher bestehen zwischen diesen drei Packages direkte Beziehungen. Weiterhin ist die Zahlung, die im Package payment modelliert wird, mit der Modellierung der Rechnung verknüpft.

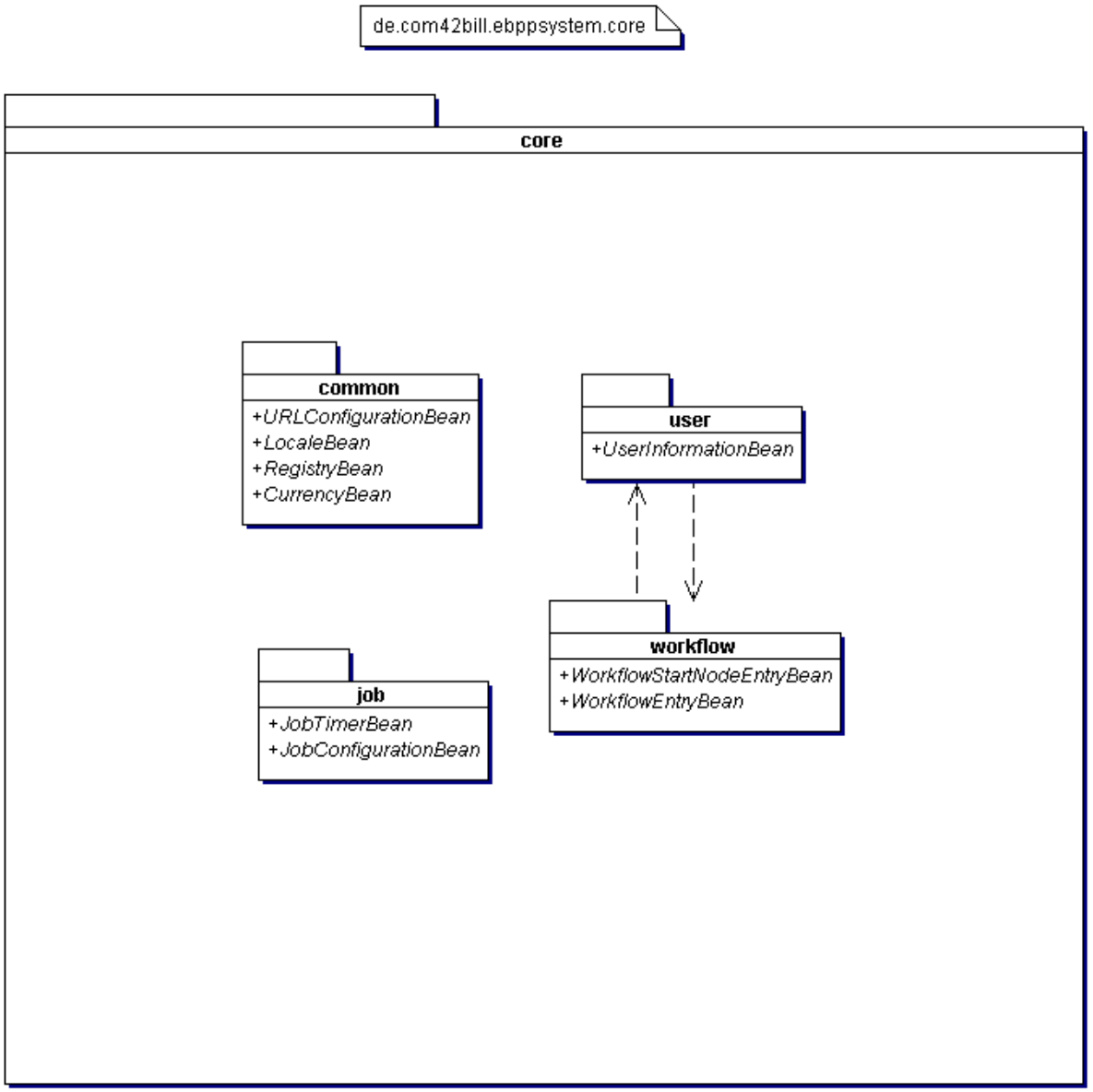


Abbildung 54: Package-Struktur des Sub Package core

Eine zentrale Rolle spielt das Package core: Dieses Sub Package von ebppsystem enthält Entity Beans, die von allen Komponenten des Systems benötigt werden. In den Entity Beans dieses Packages werden Konfigurationsdaten sowie nicht-fachliche Daten, die zum Betrieb des Systems notwendig sind, gespeichert. Eine Übersicht zu diesem Package gibt Abbildung 54. Zu beachten ist, dass außer den dargestellten Relationen keine sonstigen Relationen zwischen den Packages bestehen.

Durch das Entity Bean des Package user werden die Benutzer (User) abgebildet, die mit dem System in Online-Verbindung stehen.

Die auf den Workflow bezogene Job-Verwaltung wird durch die Entity Beans des Package job abgebildet. Hier werden Informationen über die auszuführenden Jobs sowie über den Ausführungszeitpunkt gespeichert.

Die Geschäftsprozesse der Komponente Business Logic werden durch das Package Workflow repräsentiert.

Die Entity Beans des Sub Packages common stehen allen Komponenten des Systems zur Verfügung und stellen die systemweiten Informationen bereit. Die Struktur von common wird in Abbildung 55 dargestellt.

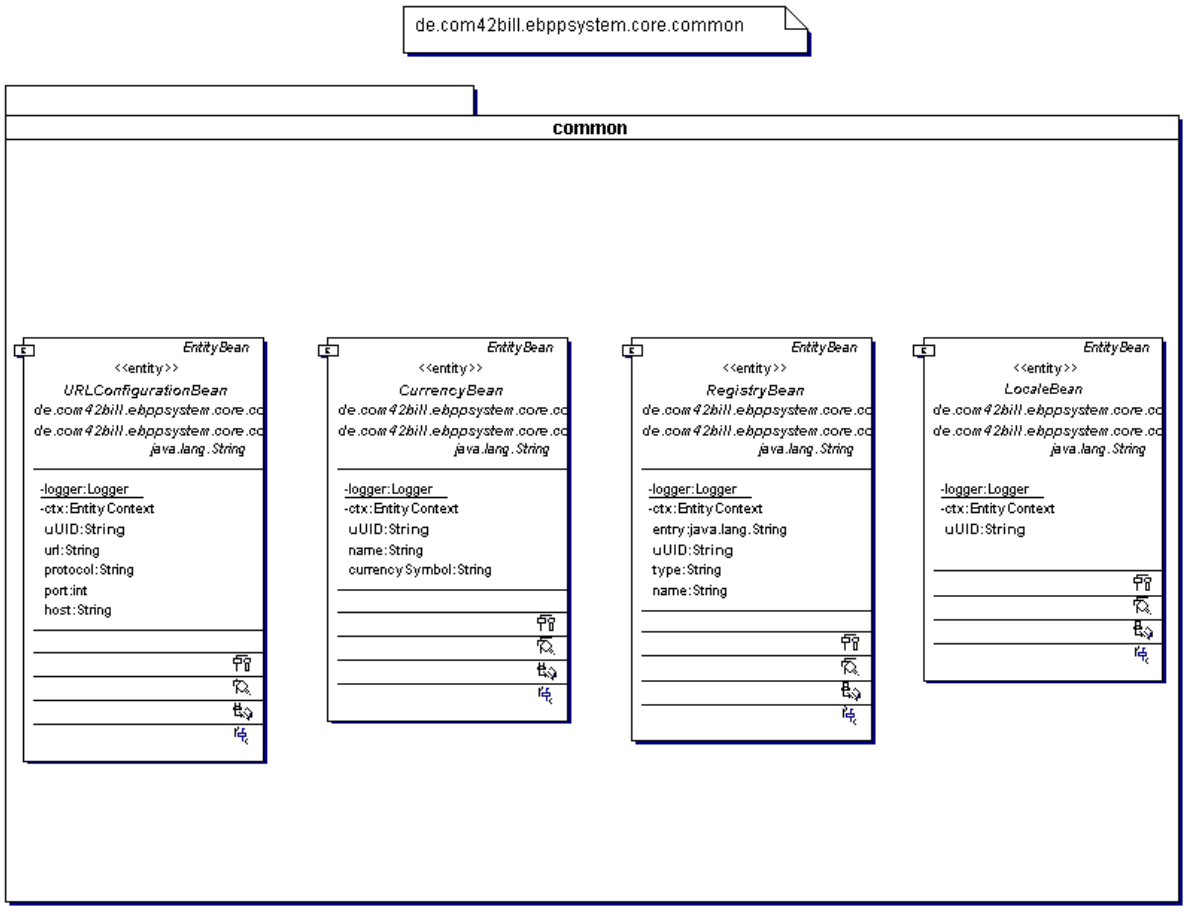


Abbildung 55: Struktur des Package common

Das Entity Bean URLConfigurationBean dient der Bereitstellung von Einstiegsadressen für Benutzergruppen des Systems. Dieses Entity Bean dient der Angabe zusätzlicher URLs zur SSL-

Verschlüsselung, welche durch das System selbst generiert werden können. Das Entity Bean LocaleBean unterstützt die Umsetzung der Länderanpassung des Systems. Diese Funktionalität wurde nicht implementiert. In dem Entity Bean RegistryBean werden Dienste und Funktionseinheiten des Systems zentral registriert. In dem Entity Bean CurrencyBean werden Währungseinstellungen hinterlegt.

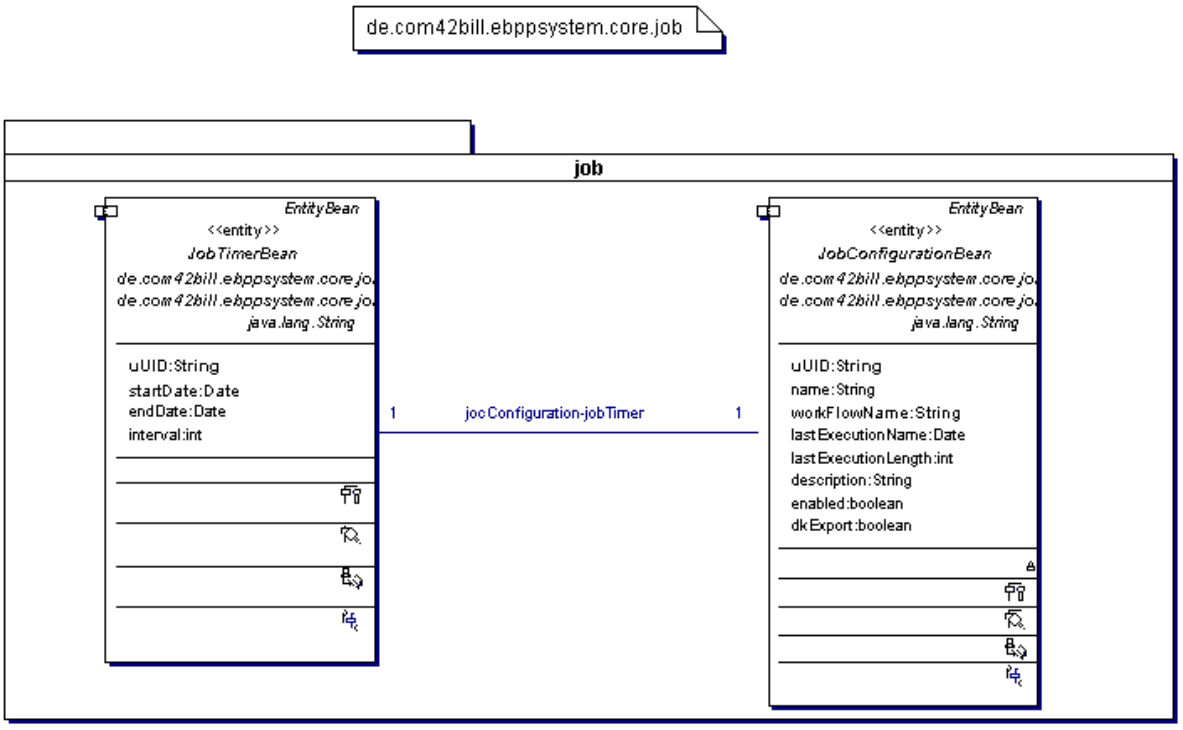


Abbildung 56: Struktur des Package job

Die auf den Workflow bezogene Job-Verwaltung wird durch die Entity Beans des Sub Package job modelliert.

In dem Entity Bean JobConfigurationBean werden Einstellungen zur Ausführung eines Workflows hinterlegt: Eine JobConfiguration stellt ein zeitgesteuertes Ereignis dar, mit dem ein Workflow gestartet wird. Das Entity Bean JobTimerBean dient der Zeitsteuerung der Workflow-Ausführung.

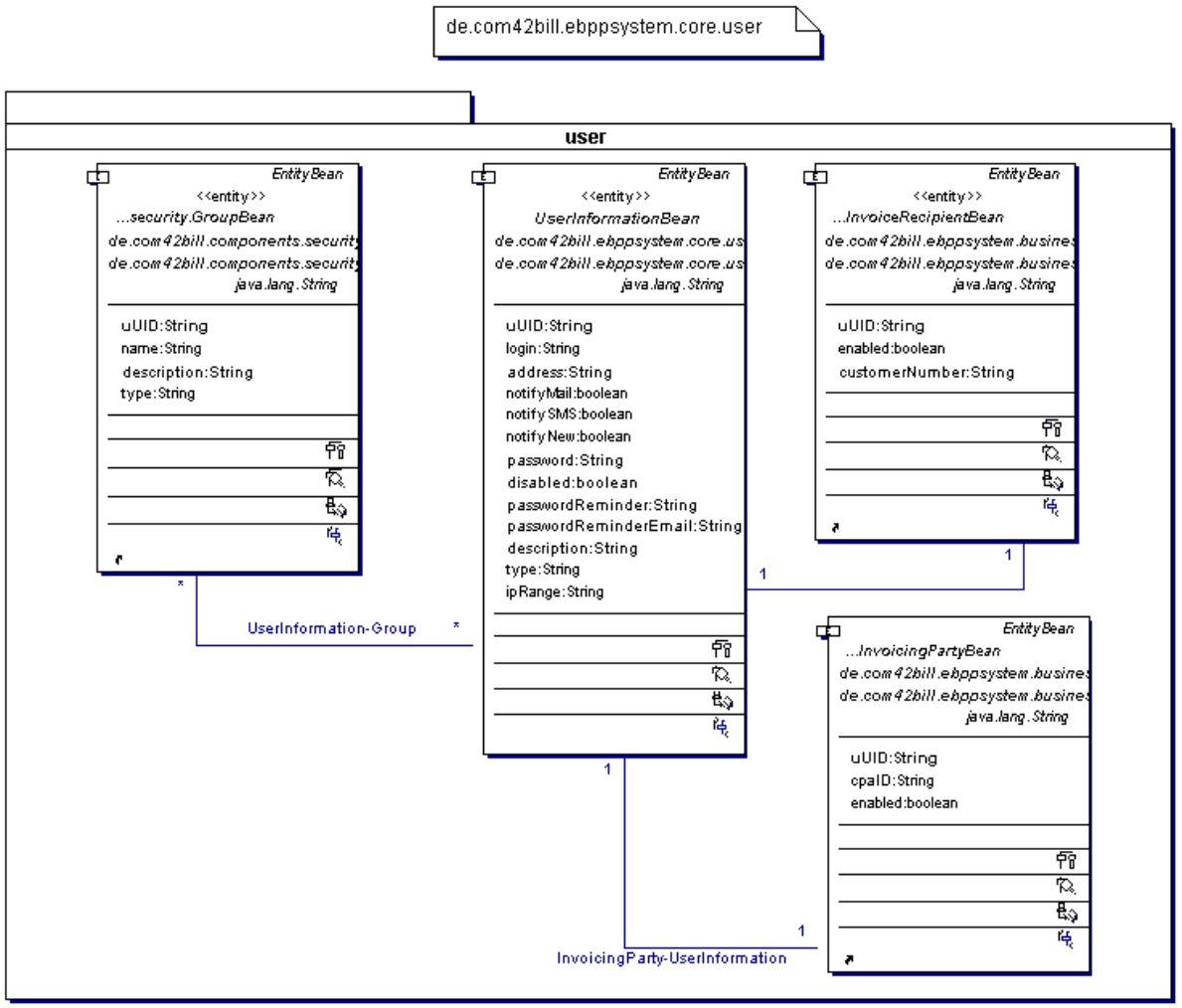


Abbildung 57: Struktur des Package user

Durch das im Sub Package user enthaltene Entity Bean UserInformationBean werden die Benutzer (User) abgebildet. UserInformationBean wird von der Komponente Sicherheit zur Unterstützung der Authentifizierung und Zugriffssteuerung benötigt. Die Struktur dieses Package wird in Abbildung 57 dargestellt.

Die n:m Beziehung zwischen GroupBean und UserInformationBean wird bei der Beschreibung des Package security näher erläutert (vgl. Abbildung 65).

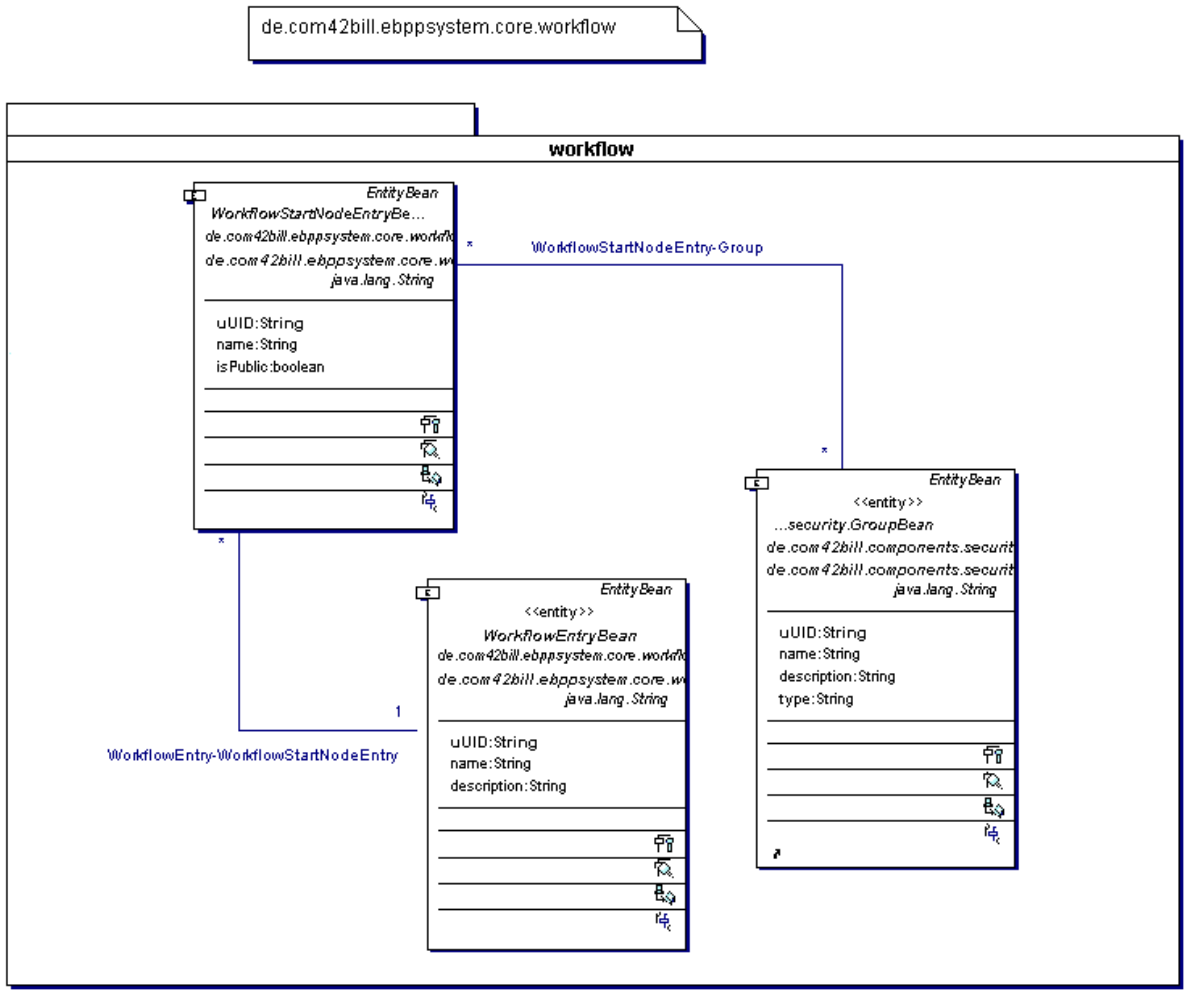


Abbildung 58: Struktur des Package workflow

Das Entity Bean WorkflowEntryBean repräsentiert die Geschäftsprozesse der Komponente Business Logic. Einem Geschäftsprozess können mehrere Startpunkte zugeordnet sein, hier repräsentiert durch das Entity Bean WorkflowStartNodeEntryBean. Über dieses Bean wird durch die Komponente Sicherheit festgelegt, ob dieser Geschäftsprozess durch den aktuell angemeldeten Benutzer ausgeführt werden darf.

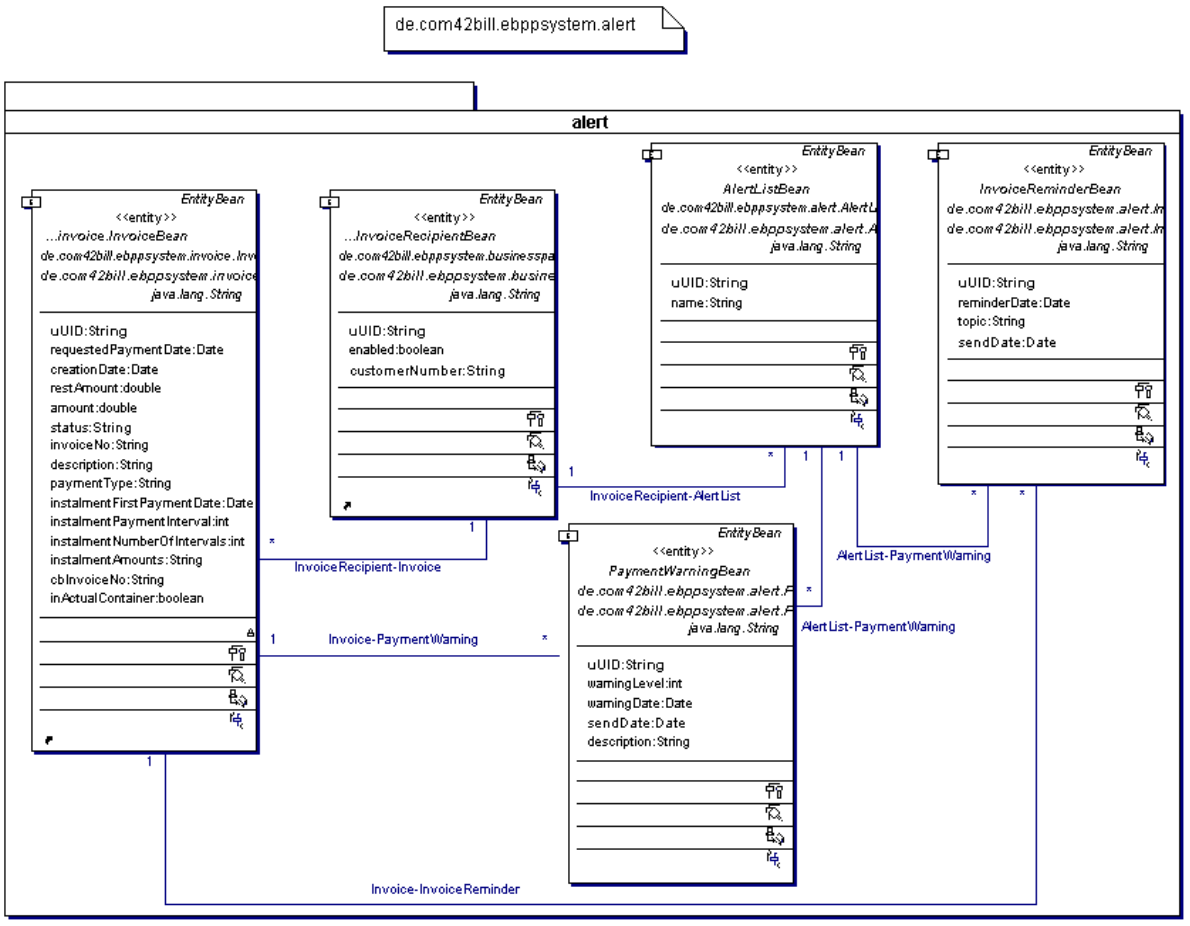


Abbildung 59: Struktur des Sub Package alert

Sind mehrere Rechnungen fällig bzw. sind mehrere Mahnungen vorhanden, werden diese in einer Liste zusammengefasst. Diese Liste wird durch AlertListBean dargestellt. Es gibt zwei Ausprägungen des Zahlungshinweises: Erinnerung (InvoiceReminderBean) und Mahnung (PaymentWarningBean), die je nach Dauer des Zahlungsverzugs eingesetzt werden können. Mehrere Zahlungshinweise sind einem Rechnungsempfänger und einer Rechnung zugeordnet.

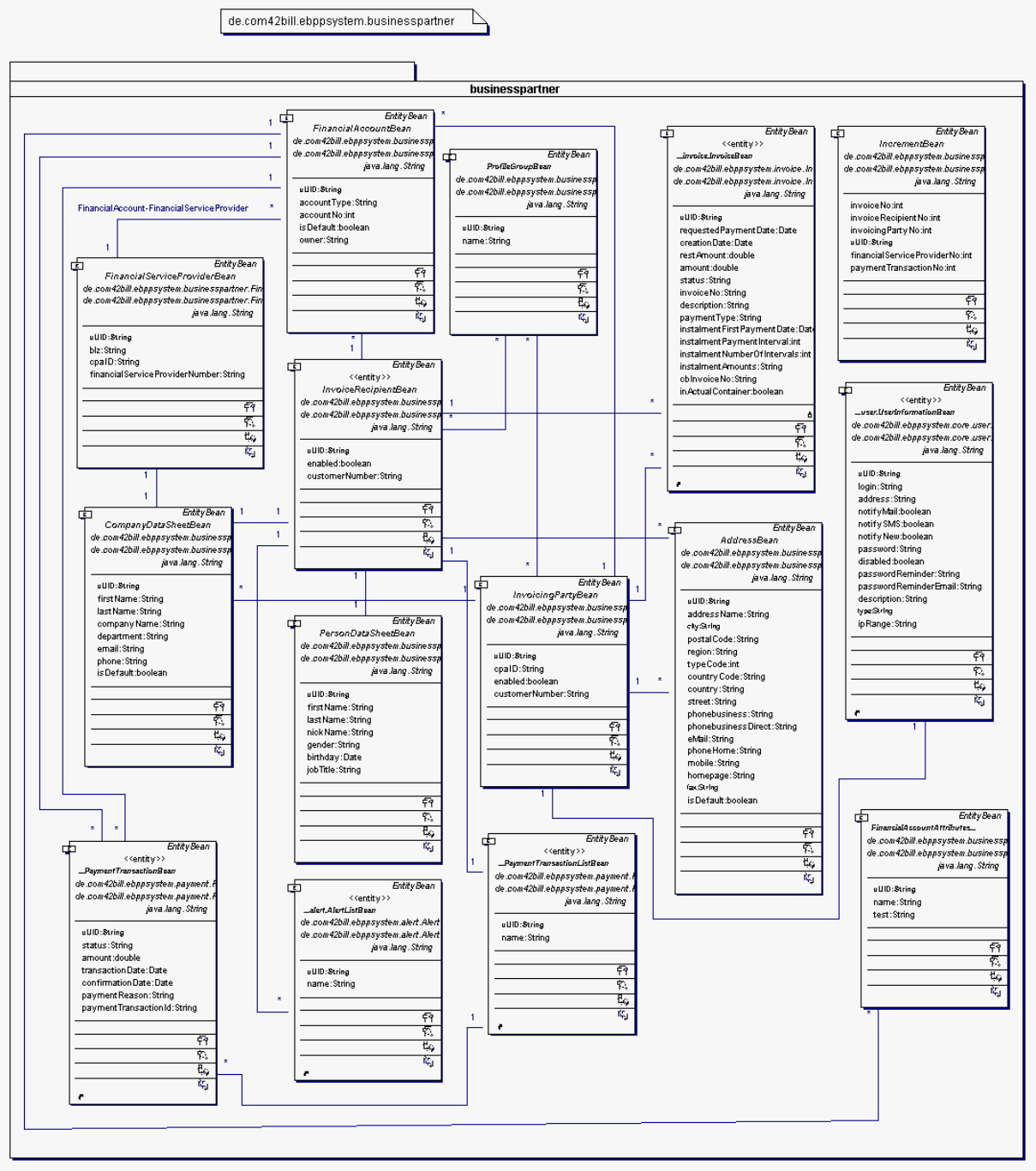


Abbildung 60: Struktur des Sub Package businesspartner

In dem Package businesspartner sind durch die Entity Beans InvoicingPartyBean und InvoiceRecipientBean die Rollen Rechnungssteller und Rechnungsempfänger modelliert. Zusammen mit dem Entity Bean FinancialServiceProviderBean repräsentieren sie alle beteiligten Rollen innerhalb eines Workflows der Business Logic (können aber auch von anderen Komponenten außerhalb eines Workflows verwendet werden). Ihre Kontaktdaten werden durch die beiden Entity Beans PersonDataSheetBean und CompanyDataSheetBean dargestellt. Diese Entity Beans entsprechen den unterschiedlichen Anforderungen, die Informationen zu Firmen und privaten Einzelpersonen mit sich bringen. Die benötigten Adressinformationen werden durch AddressBean repräsentiert. Die Entity Beans der Rechnungssteller und Rechnungsempfänger können zu Profilgruppen (ProfileGroupBean)

zusammengefasst werden. Die Profilgruppen können z.B. als Zugriffsrechte benutzt werden. Das Entity Bean FinancialAccountBean repräsentiert die Finanzkonten der beteiligten Rollen, über die die Zahlungsvorgänge abgewickelt werden. Das Entity Bean FinancialAccountAttributesBean dient der weiteren Spezifizierung des FianancialAccountBean, z.B. Angabe von Kreditkarteninformationen. Durch das IncrementBean lassen sich die laufenden Nummern für Rechnungssteller, Rechnungsempfänger, Rechnungen, Finanzdienstleister und Transaktionen generieren.

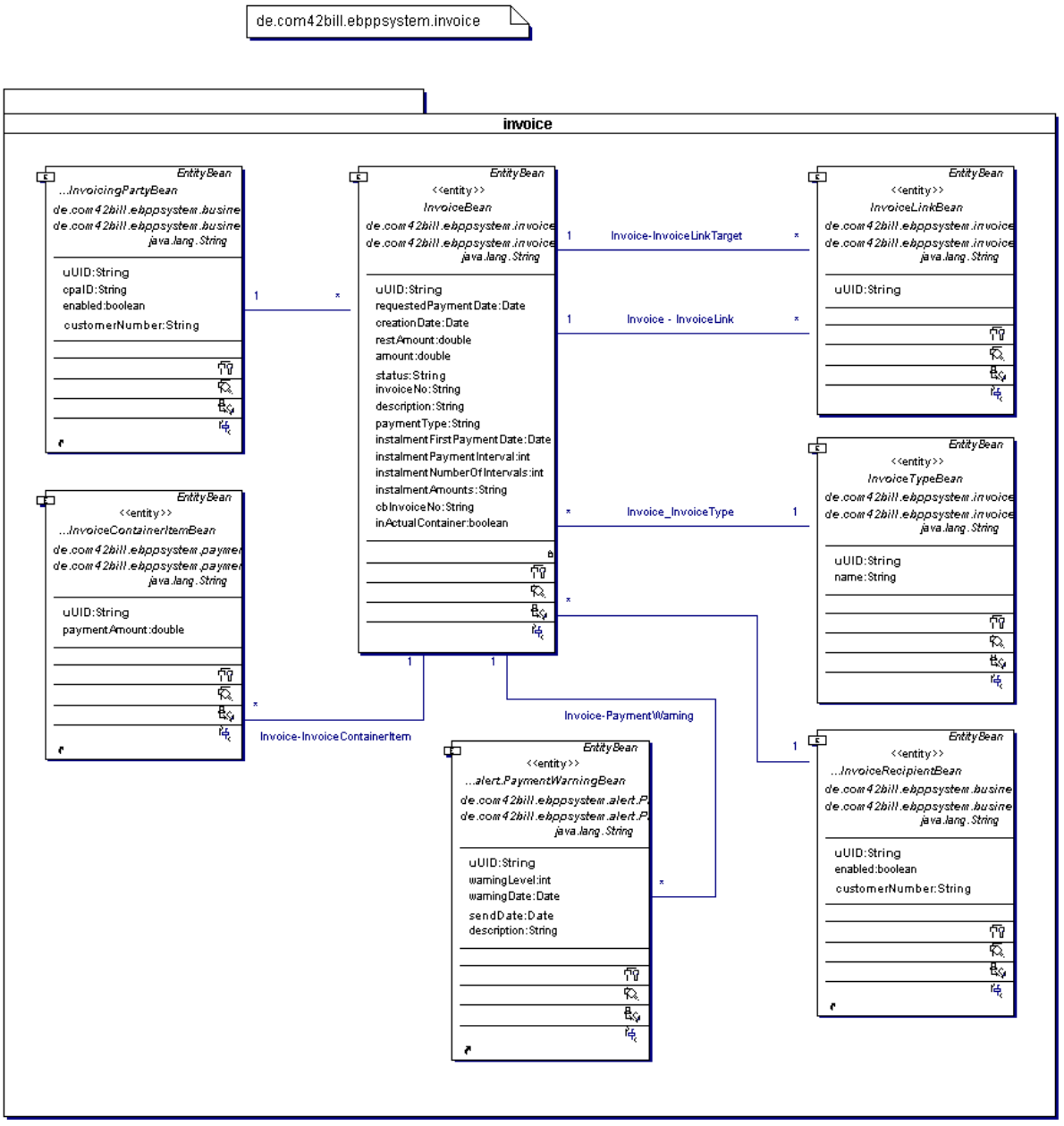


Abbildung 61: Struktur des Sub Package invoice

Mit den Entity Beans des Package invoice werden die Rechnungen modelliert. Seine Klassenstruktur wird in Abbildung 61 dargestellt. Das zentrale Entity Bean – InvoiceBean – repräsentiert Rechnungen. Mit dem InvoiceLinkBean können Beziehungen zwischen mehreren Rechnungen hergestellt werden. InvoiceTypeBean legt den Typ der Rechnungen fest.

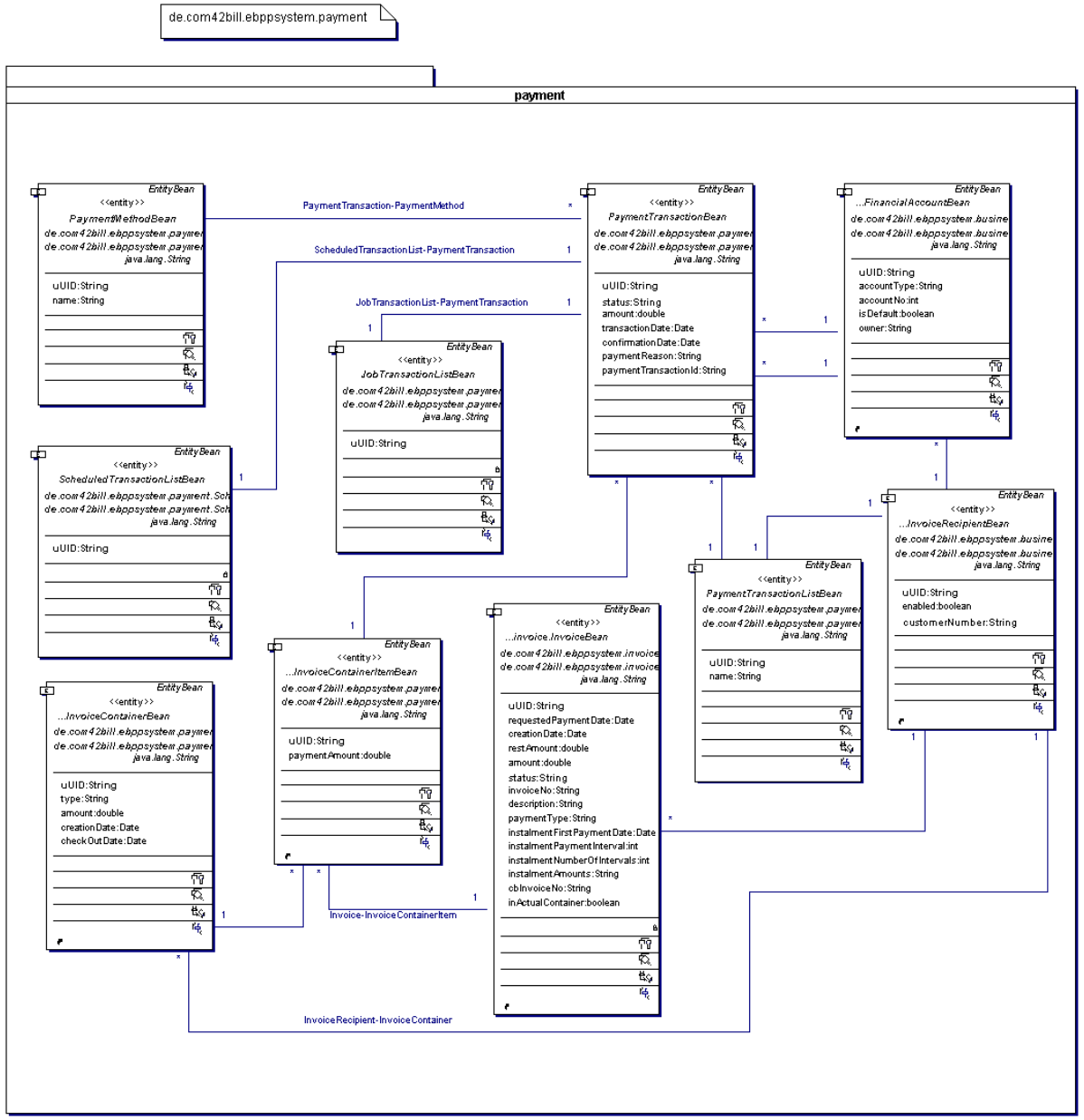


Abbildung 62: Struktur des Package payment

Der Zahlungsvorgang wird im Package payment modelliert, hier dargestellt in Abbildung 62.

Das InvoiceContainerBean dient ähnlich einem Warenkorb der Gruppierung von Rechnungen, welche zusammen bezahlt werden sollen. Eine thematische Gruppierung kann (optional) über InvoiceTypeBean erfolgen. Die Transaktionsdaten werden in dem Entity Bean PaymentTransactionBean abgebildet, die Zahlungsweise dagegen in PaymentMethodBean. Zahlungen werden durch das Entity Bean PaymentTransactionListBean zusammengefasst.

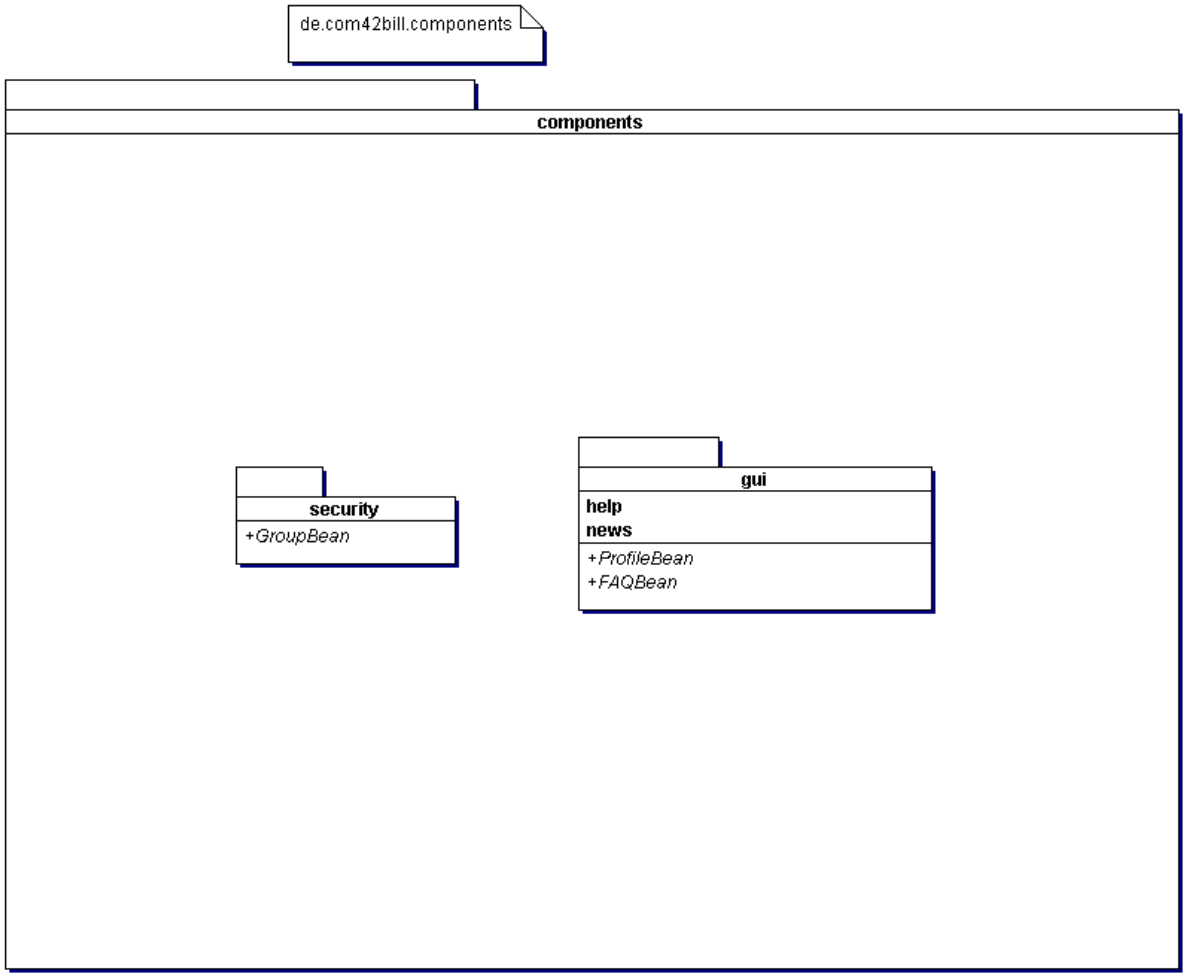


Abbildung 63: Package-Struktur des Package components

Wie in Abbildung 52 bereits dargestellt, enthält das Objektmodell des Systems neben dem Package ebppsystem ein weiteres Package components mit zentraler Bedeutung. In dem Package ebppsystem werden neben den allgemeinen auch die Entity Beans der Komponente Business Logic modelliert. In dem Package components sind die spezifischen Entity Beans der übrigen Komponenten enthalten (die Komponente Datenkonverter benötigt keine eigenen Entity Beans). Eine strikte Trennung der Komponenten wird dabei durch die weitere Unterteilung in die Sub Packages security und gui vorgenommen (siehe Abbildung 63). Die Entity Beans dieser Sub Packages hängen mit denen der Komponente Business Logic zusammen, jedoch existieren keine anderen Relationen zwischen den Sub Packages von components.

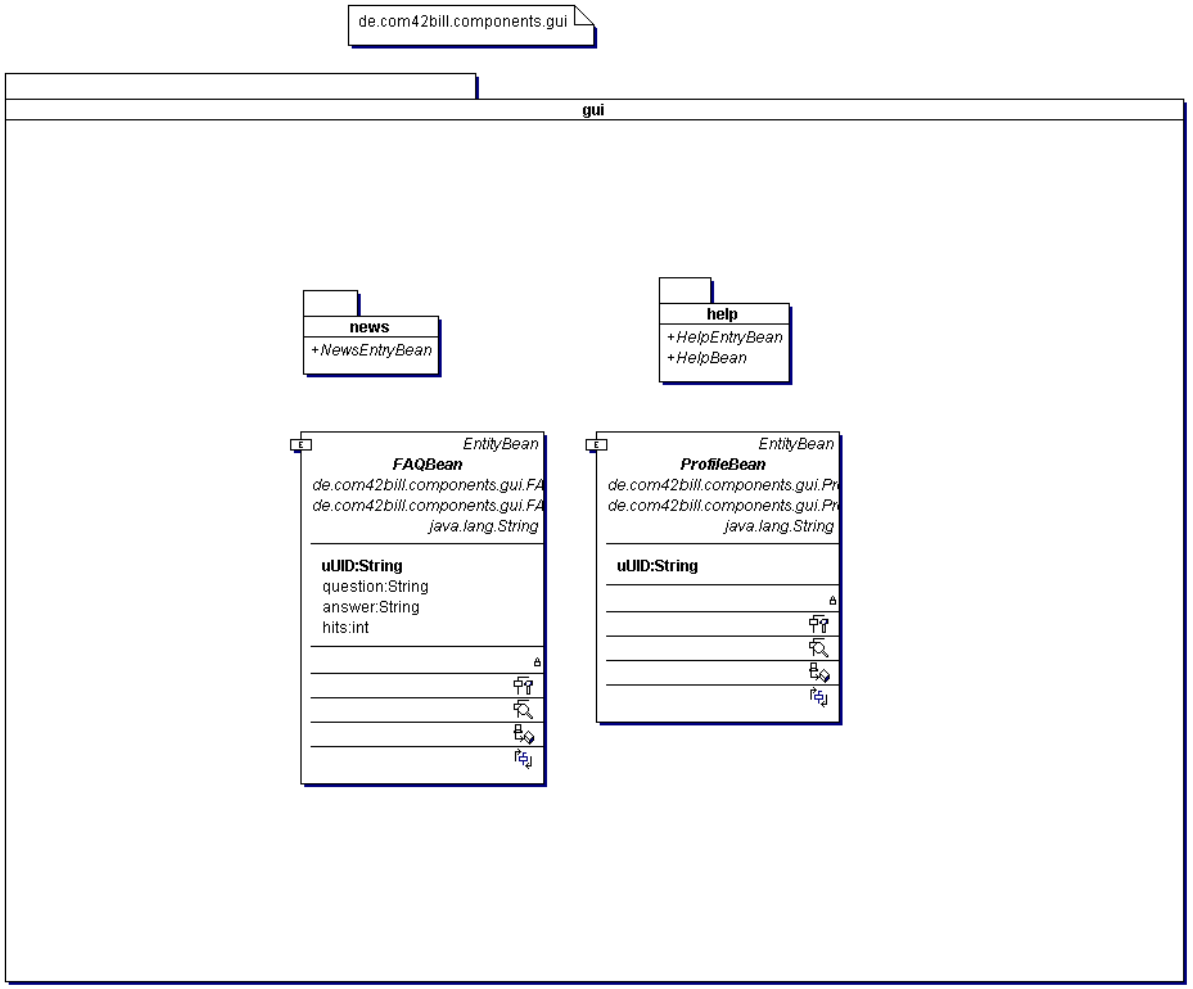


Abbildung 64: Struktur des Package gui

Die Entity Beans der Komponente GUI werden im Package gui zusammengefasst (siehe Abbildung 64). Gespeichert werden die Artikel für den Abschnitt „Mein Orakel“ auf der Hauptseite (News) (siehe Kapitel 6.2.1) sowie Hilfestellungen für den Umgang mit Com42Bill (FAQ). Die Speicherung von Hilfetexten ist vorgesehen, aber nicht implementiert.

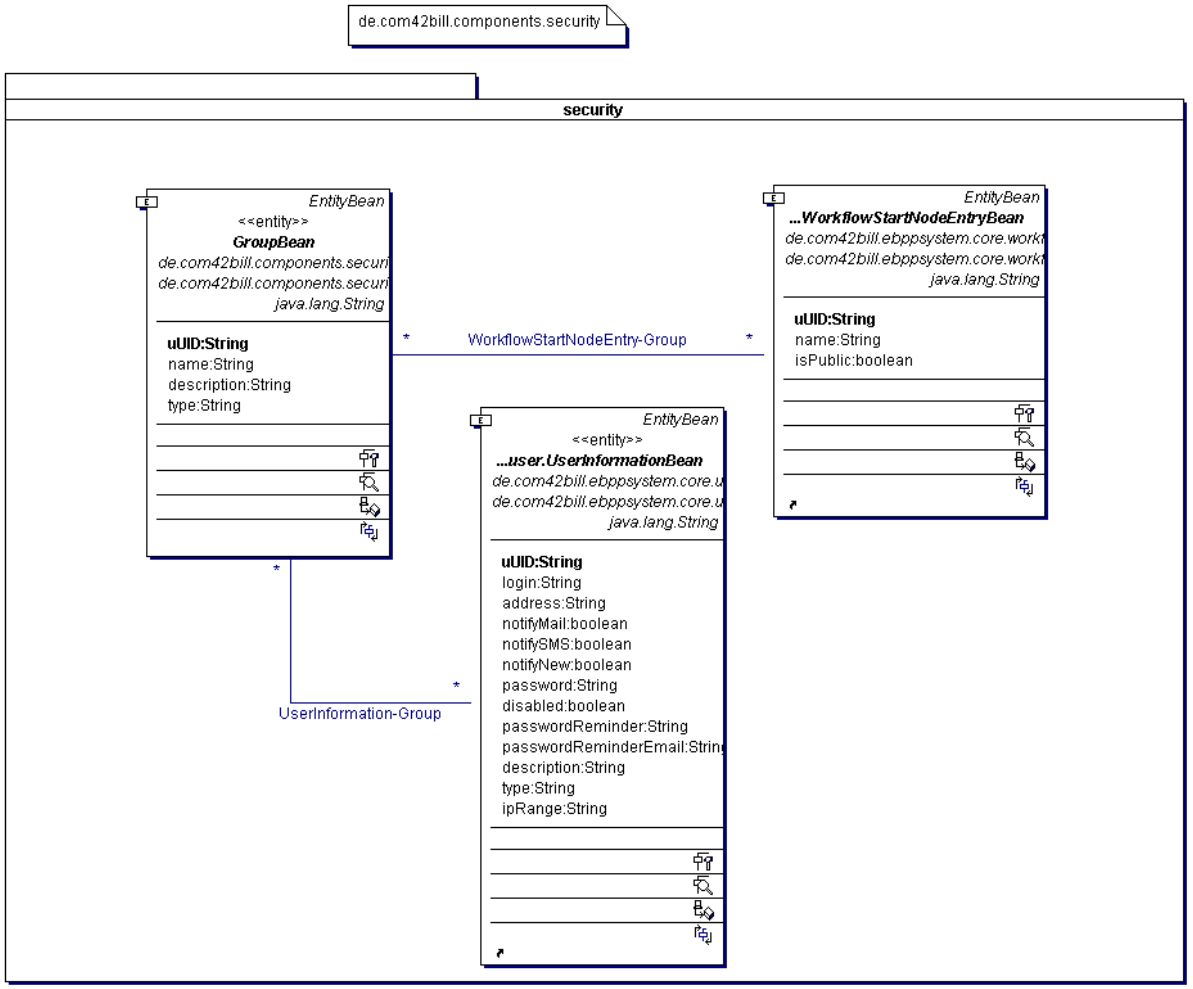


Abbildung 65: Struktur des Package security

Das Package security wird in Abbildung 65 dargestellt. Hier werden die Entity Beans modelliert, die zur Verwaltung der Zugriffrechte benötigt werden.

In UserInformationBean (Package de.Com42Bill.ebppsystem.core.user) werden die nicht-fachlichen Benutzerdaten gespeichert. Mehrere Benutzer können in einer Gruppe enthalten sein. Außerdem kann ein Benutzer mehreren Gruppen zugeordnet werden. Eine Gruppe entspricht einer Benutzergruppe, die den darin enthaltenen Benutzern die Ausführung bestimmter Workflows erlaubt. Zwischen GroupBean und WorkflowStartNodeEntryBean besteht ebenfalls eine n:m Beziehung. Jeder Gruppe können mehrere Workflows zugeordnet werden. In umgekehrter Richtung kann jeder Workflow mehreren Gruppen zugeordnet werden. Dieser Aufwand ermöglicht eine sehr variable Rechteverwaltung.

Bei der Modellierung von n:m Beziehungen zwischen Entity Beans müssen in Together 6.0 für beide betroffenen Entity Beans Assignment-Tabellen definiert werden. Eine Assignment-Tabelle besteht aus zwei Spalten, die für die Primärschlüssel der beiden Entity Beans vorgesehen sind. Das Eintragen der Primärschlüssel in die Assignment-Tabellen wird während des Deployments vom Applicationserver übernommen.

8 Hard- & Software

8.1 Infrastruktur

Das System Com42Bill wird in Java auf Basis der Spezifikation der Java 2 Enterprise Edition (J2EE) in der Version 1.3 entwickelt, da sich der Einsatz dieser Technik gerade im Bereich des e-Business bereits vielfach bewährt hat. Die Entwicklung findet in einer Windows-Umgebung statt. Der Domänencontroller ist ein Intel Celeron 1GHz Rechner, auf welchem das Betriebssystem Microsoft Windows 2000 Server installiert ist. Alle benötigten Serverdienste werden von diesem Rechner bereitgestellt. Die Klienten sind ebenfalls Intel Celeron 1 GHz Rechner und werden mit Microsoft Windows 2000 Professional betrieben. Damit stehen zur Entwicklung typische Arbeitsumgebungen zur Verfügung, da zu einem Großteil private PCs mit Microsoft Windows Betriebssystemen genutzt werden. Die Softwareentwicklungsumgebung ist das Together ControlCenter von TogetherSoft in der Version 6. Dadurch ist es möglich, das System vollständig in einer Entwicklungsumgebung zu entwickeln. Durch die Unterstützung von UML steht ein standardisiertes und effizientes Hilfsmittel bei der Entwicklung zur Verfügung, welches es erlaubt, die einzelnen Entwicklungsschritte grafisch und verständlich darzustellen. Die Dateien werden in einem CVS-System verwaltet. Hier kommt auf dem Server CVSNT zum Einsatz. Die Klienten benutzen WinCVS bzw. das Together ControlCenter zum Abgleich der Dateien mit den Versionen des Servers. Dies ermöglicht eine unkomplizierte Verwaltung sämtlicher Dokumente und aller anderen anfallenden Dateien, so dass jeder, der an der Entwicklung beteiligt ist, jederzeit die aktuellsten Informationen zur Hand hat. Zur Präsentation der Projektgruppe nach Außen wird der in Windows 2000 Server integrierte IIS-Dienst genutzt. Dieser unterstützt die von unserer Homepage geforderten Techniken. Darüber hinaus ist die Anmeldung an den internen Bereich der Homepage direkt mit der Anmeldung an der Domäne verbunden, so dass hier keine zusätzlichen Passworte notwendig werden und der Nutzungskomfort deutlich erhöht werden kann.

8.2 Installation

Welche Software für den Betrieb von Com42Bill benötigt wird, ist nicht vollständig vorgeschrieben. Als Grundlage für den Betrieb von Com42Bill dient der Applicationserver Weblogic 6.1 SP 3 der Firma Bea. Dadurch ist Com42Bill weitgehend unabhängig von der Wahl der Datenbank. Diese muss lediglich eine korrekte Anbindung an Weblogic bieten. Für die Entwicklung fiel die Entscheidung auf Oracle 8. Für eine möglichst einfache Konfiguration wird die Installation dieser Software in deren Standardverzeichnisse empfohlen.

Bevor mit der Installation von Com42Bil begonnen wird, muss die zu Grunde liegende Software installiert und konfiguriert werden. Eine neue Installation eines auf Windows NT basierenden Serverbetriebssystems ist als Grundlage zu empfehlen. Es sollte zuerst die Datenbank und anschließend der Applicationserver installiert werden.

Handelsübliche Datenbanken unterstützen eine Vielzahl von Parametern, um sie optimal auf die erwartete Auslastung einzustellen. Für die Entwicklung wurde Oracle auf minimale Ressourcennutzung konfiguriert, da nicht mehr als 10 - 20 gleichzeitige Verbindungen zu erwarten waren. Für den Produktiveinsatz muss die Konfiguration entsprechend der erwarteten Zugriffe angepasst werden. Genauere Informationen zur Konfiguration der Datenbank sind in deren Dokumentation verfügbar. Wichtig ist, dass ein Datenbankbenutzer angelegt wird, der anschließend von Weblogic für den Zugriff auf die Datenbank genutzt wird. Es ist davon abzuraten, für diesen Zweck einen Datenbankadministrator zu benutzen. Der Anfangsdatenbestand wird auf der Installations-CD von Com42Bill mitgeliefert und kann direkt in die Datenbank importiert werden.

Die Installation von Bea Weblogic sollte ohne die mitgelieferten Beispiele erfolgen. Die einzurichtende Domain soll cbdomain heißen, da die Konfigurationsskripte von Weblogic auf diese Domain zugeschnitten sind.

Nun sollte das Verzeichnis „Konfiguration“ von der Com42Bill-CD auf die Partition C: der Server-Festplatte kopiert werden. Wurden die Programme in die Standardpfade installiert, ist die Installation damit auch fast abgeschlossen. Ansonsten müssen die Pfade in den kopierten Dateien entsprechend angepasst werden. Die Dateien, die auf der CD im Verzeichnis y zu finden sind, müssen auf eine Partition oder Freigabe kopiert werden, die über den Laufwerksbuchstaben Y: eingebunden wird.

Bei der Konfiguration von Weblogic ist zu beachten, dass in den jeweiligen Einstellungen der aktuelle Server im Tab „Target“ ausgewählt wird, da erst dann die Einstellungen wirksam werden. Die Konfigurationskonsole ist über http://localhost:7001/console erreichbar. Zuerst sollte hier die Startklasse de.Com42Bill.ebppsystem.core.server.StartupCore im Bereich Startup & Shutdown eingetragen werden. Hierdurch werden grundlegende Initialisierungen bei jedem Serverstart durchgeführt. Über die Konfigurationskonsole müssen als Nächstes ein JDBC Connection Pool und eine JDBC Tx Data Source eingerichtet werden. Der Namensvorschlag für diese Einträge ist CbConnectionPool. Anschließend muss eine Mail-Session eingerichtet werden. Diese dient zum automatischen Versenden der Emails von Com42Bill. Als letzten Schritt vor der eigentlichen Installation von Com42Bill muss JMS konfiguriert werden. Es werden die Komponenten Connection Factory, Store und Server benötigt. Vorgesehene Namen hierfür sind CBJMSConnectionFactory, JMSFileStore und CBJMSServer.

Nun kann mit Hilfe der Konfigurationskonsole die Installation der Applikation beginnen. Als Applikation wird die Datei „Com42Bill.ear“ von der beiliegenden CD ausgewählt. Anschließend wird die Web Application GUIServlets als Standardservlet eingerichtet. Dies kann im Bereich „Servers“ vorgenommen werden.

Weblogic lässt neben der Kommunikation über HTTP auch die Kommunikation über HTTPS zu. Hier ist Port 7002 vorkonfiguriert. Für die Nutzung von Com42Bill sollte aus Sicherheitsgründen nur die Kommunikation über HTTPS erlaubt werden. Andere Ports sollten deshalb durch eine geeignete Firewall abgeschirmt werden.

Nach einer erfolgreichen Installation kann Com42Bill über folgende URLs erreicht werden:

Benutzerseiten: https://localhost:7002/doRequest

Administrationsseiten: https://localhost:7002/doRequest?reqid=adm

9 Projektmanagement

9.1 Einleitung

In diesem Kapitel wird der Projektplan der Projektgruppe 411 (Com42Bill) beschrieben. Dabei werden die Aktivitäten, die von den einzelnen Gruppen der Projektgruppe in den beiden Semestern durchgeführt wurden, vorgestellt. Bei diesen Gruppen handelt es sich um die Entwicklungs- und Querschnittsaufgabenteams. Somit wird der Projektplan auch in zwei wesentlichen Aufgabenbereichen aufgeteilt. Zuerst werden die Tätigkeiten, die in den Entwicklungsteams und anschließend die Tätigkeiten, die von den Querschnittsaufgabenteams vollzogen wurden, dargestellt. Insgesamt waren jeweils fünf verschiedene Entwicklungs- und Querschnittaufgabenteams im Einsatz. Jedes Entwicklungsteam war für die Entwicklung einer Komponente zuständig. Neben der Entwicklung wurden parallel durch die Querschnittsaufgabenteams weitere Tätigkeiten durchgeführt, die im Rahmen eines solchen Projektes typischerweise anfallen.

Die Vorgehensweise bei der Entwicklung des Com42Bill-Systems war für alle Entwicklungsteams dieselbe. Der Zeitaufwand bei der Entwicklung der einzelnen Komponenten variierte dabei leicht. Nachfolgend werden die Haupt-Aktivitäten, in denen jedes Entwicklungsteam involviert war, vorgestellt:

9.2 Abschnitte des Projekts

9.2.1 Anforderungsanalyse

Die Anforderungsanalyse zeichnete die Anfangsphase der Projektgruppe aus. Ziel dieser Phase war es, funktionale und nichtfunktionale Anforderungen für die einzelnen Komponenten zu ermitteln. Hierbei wurden die Eigenschaften, die die jeweilige Komponente besitzen muss, festgehalten. Für nähere Erläuterungen bzgl. Anforderungen der Komponenten siehe Anhang A. Für die Erstellung des Anforderungsdokuments wurden ca. 45 Tage, d.h. von Mitte Mai bis Mitte Juli 2002 Zeit benötigt.

9.2.2 Systemarchitektur

In der Phase Systemarchitektur wurde zunächst von den Chef-Architekten eine Architektur für das Gesamtsystem entworfen und vorgestellt. Die Komponententeams haben die vorgeschlagenen Architekturen für ihre Komponente übernommen und später gemäß eigenen Vorstellungen und Möglichkeiten nach und nach angepasst. Für die Erstellung der Systemarchitektur wurden ebenfalls ca. 45 Tage, d.h. von Mitte Mai bis Mitte Juli 2002 Zeit benötigt.

9.2.3 Klassenmodell

Die Phase Klassenmodell diente eigentlich dazu, die nicht-persistenten Klassen der Komponenten in UML-Notation zu modellieren. Der vorgesehene Anfangstermin für diese Aktivität (02.07.2002) wurde nicht eingehalten. Es wurde entschieden, diese Aktivität parallel zu der Prototyping-Phase durchzuführen, weil erst dort einige Fragen bzgl. der Machbarkeit einiger Funktionen und somit der Notwendigkeit einiger Klassen geklärt werden konnte. Trotz alledem ist diese Hauptaktivität von keiner Komponente genau durchgeführt und dokumentiert worden. Eigentlich sollte diese Phase Mitte Juli 2002 beginnen und Anfang November 2002 fertig sein.

9.2.4 Objektmodell

Die Phase Objektmodell diente dazu, die persistenten Klassen der Komponenten in UML-Notation zu modellieren. Insgesamt wurden für die erste Version des Objektmodells 95 Tage

benötigt, d.h. von Beginn der PG (April 2002) bis Anfang September 2002. Für die Verwaltung und Pflege des Objektmodells waren die Mitglieder der Komponente Datenbank zuständig.

9.2.5 Objektdesign

Ziel der Phase Objektdesign war es, die Klassen der Komponenten detailliert zu beschreiben, d.h. die notwendigen Attribute und Methoden hinzuzufügen. Am Ende dieser Phase sollte ein Designdokument vorliegen. Der vorgesehene Zeitrahmen für diese Phase war Mitte Juli 2002 bis Mitte Oktober 2002. Da das Designdokument erst am Ende der Implementierungsphase, d.h. als alle Komponenten lauffähig waren, fertig gestellt werden konnte, konnte der Endtermin für diese Phase nicht eingehalten werden.

9.2.6 Prototyp

Die Prototyping-Phase war dafür vorgesehen, die Key-Features der Komponenten mit reduziertem Funktionsumfang zu implementieren bzw. verschiedene Techniken zu testen, um die Machbarkeiten für die Komponente festzustellen. Hierbei sollten die Key-Features der Komponenten so ausgewählt werden, dass ein Gesamtdurchlauf des Systems, d.h. der Einbezug aller Komponenten, möglich war. Diese Phase ist in der vorlesungsfreien Zeit durchgeführt worden, d.h. von Anfang Juli bis Ende September waren die Teilnehmer mit Prototyping beschäftigt.

9.2.7 Implementierung

In der Implementierungs-Phase fand die eigentliche Implementierung statt, wobei hier in fast allen Komponenten die Erkenntnisse und der Code aus der Prototyping-Phase eingeflossen sind. Die Komponenten wurden in dieser Phase mit allen vorgesehenen Features vollständig implementiert und dokumentiert. Mit der Implementierung begannen die Komponententeams ungefähr Mitte Oktober. Einige benötigten mehr und andere weniger Zeit. Ingesamt dauerte dieser Phase bis Mitte Februar 2003.

Zu Beginn waren die Aktivitäten Klassenreviews, Klassentests und Integrationstests zusätzlich zu den anderen Aktivitäten jeder Komponente eingetragen. Da jedoch für die Planung und Beaufsichtigung dieser Aktivitäten die Querschnittsaufgabe Qualitätsmanagement im Laufe der zweiten Projektgruppensemester verantwortlich wurde, wurden diese zu den anderen Aktivitäten dieser Querschnittsaufgabe hinzugefügt. Für die Klassenreviews wurden insgesamt 34 Tage von 10.12.2002 bis 24.01.2003, für die Klassentests wurden insgesamt 9 Tage von 20. bis 30.01.2003 und für die Integrationstests insgesamt 15 Tage 30.01. bis 15.02.2003 benötigt.

9.3 Querschnitssaufgaben

Wie oben schon erwähnt, waren parallel zu Entwicklungsaktivitäten auch Querschnittsaktivitäten zu erfüllen, die nachstehend vorgestellt werden.

9.3.1 Projektmanagement

Der Projektplan wurde während der Anfangsphase der Projektgruppe erstellt und musste kontinuierlich durch das Projektmanagement aktualisiert werden. In einem Rhythmus von ca. zwei Wochen wurde der Projektplan in der Projektgruppensitzung besprochen. In den ersten Projektgruppensemester wurde durch das Projektmanagement ein Anforderungsdokument erstellt, dessen Pflege an das Qualitätsmanagement übergeben wurde. Des Weiteren pflegte das Projektmanagement die Abwesenheitsliste und organisierte die Verwaltung der Projektgruppen-Kasse. In der Abwesenheitsliste konnten sich die Teilnehmer, die während der Semesterferien für längere Zeit nicht anwesend waren, eintragen.

9.3.2 Marketing

Im ersten Semester der Projektgruppe hat das Marketingteam die Bestellung von Projektgruppen T-Shirts organisiert, das Whitepaper entworfen, die Erstellung des Zwischenberichtes koordiniert, das Projektgruppenplakat entworfen und sich um die Kontaktaufnahme mit den Firmen gekümmert. Im zweiten Projektgruppensemester hat das Marketingteam sich wieder um die Kontaktaufnahme mit den Firmen bemüht. Diese Bemühungen blieben jedoch ohne Erfolg. Eine weitere Tätigkeit war die Koordination der Erstellung des Abschlussberichts. Für die Präsentation des Com42Bill-Systems ist das Marketingteam ebenfalls zuständig.

9.3.3 Qualitätsmanagement

Das Qualitätsmanagement war u. a. verantwortlich für den Entwurf eines Code Guide. Diese Aktivität wurde recht frühzeitig in den ersten Semesterwochen der Projektgruppe erledigt. Des Weiteren war es dafür verantwortlich, Dokumente wie den Projektplan, Anforderungs-dokument und Key-Features bzgl. ihrer Richtigkeit zu prüfen. Die Erstellung, Pflege und Wartung des Prozessmodells wurde ebenfalls durch das Qualitätsmanagement vorgenommen. Im zweiten Semester der Projektgruppe hat das Qualitätsmanagement jeweils die Klassenreviews, die Klassentests und die Integrationstests der Komponenten organisiert und in Zusammenarbeit mit den jeweiligen Komponenten durchgeführt. Zum Abschluss wurde ein Systemtest durchgeführt.

9.3.4 GUI

Am Anfang der Projektgruppe existierte neben den anderen Querschnittsaufgaben auch die Querschnittsaufgabe GUI. Sie war für den Entwurf des Com42Bill-Logos zuständig und sollte auch das Plakat erstellen. Die Querschnittsaufgabe GUI wurde in der Mitte des ersten Projektgruppensemesters aufgelöst; das Team ging in den Querschnittsaufgaben Marketing und Qualitätsmanagement auf.

9.3.5 Chef-Architekten

Die Chef-Architekten haben am Anfang des ersten Semesters eine Gesamtsystemarchitektur, in der die Subkomponenten aller Komponenten ersichtlich werden, entwickelt. Diese wurde anschließend von den Komponententeams aufgenommen und erweitert. Die Systemarchitektur wurde von den Chef-Architekten während der gesamten Projektgruppe gepflegt und aktualisiert. Zudem mussten sie die Schnittstellen zwischen den Komponenten kontinuierlich überprüfen und Anpassungsvorschläge entwickeln. Neben diesen Tätigkeiten sind sie für die Installation und Einrichtung des Applikationsservers gemeinsam mit den Systemadministratoren zuständig gewesen.

9.3.6 Systemadministration

Zu den Aktivitäten von Systemadministratoren gehörten u. a. die Einrichtung von Server, Workstations und CVS. Des Weiteren sind sie für die Installation und Einrichtung der Entwicklungsumgebung bzw. neuer Software zuständig gewesen. Neben diesen Tätigkeiten haben sie die Newsgroup und den FTP-Zugang für die Projektgruppe eingerichtet.

9.4 Aufgabenübersicht

Zuordnung der Projektgruppenteilnehmer zu Komponenten / Querschnittsaufgaben:

Komponentenzuständigkeiten	
Business Logic	Alireza Salemi, Narcisse Kemogne Kamdem, Timo Albert
Datenkonverter	Christian Kotthoff, Zahir Amiri
Datenbankserver	Andre Pavlenko, Stefan Pinschke (1. PG-Semester)
Sicherheitsserver	Bastian Schlich, Dennis Müller
GUI	Alexander Schmitz, Matthias Niggemeier, Dino Hasanbegovic
Querschnittsaufgabenzuständigkeiten	
Systemadministration	Stefan Pinschke (1. PG-Semester), Dennis Müller
Marketing	Matthias Niggemeier, Dino Hasanbegovic
Qualitätsmanagement	Christian Kotthoff, Alexander Schmitz, Andre Pavlenko
Chef-Architekten	Alireza Salemi, Narcisse Kemogne Kamdem, Timo Albert
Projektmanagement	Bastian Schlich, Zahir Amiri

Stefan Pinschke und Bastian Schlich tauschten in der Mitte des ersten Projektgruppensemester ihre Querschnittssaufgabe. Im zweiten Projektgruppensemester war Stefan Pinschke nicht mehr Teilnehmer der Projektgruppe Com42Bill.

10 Qualitätsmanagement

10.1 Einleitung

Das Ziel bei der Software-Entwicklung ist es, eine Software von hoher Qualität zu erzeugen. Qualität wird von einigen Fachleuten als die Abwesenheit von Fehlern definiert [STE00]. Auf Softwareprodukte ist diese Definition nicht ohne weiteres übertragbar, denn auch Software mit Fehlern wird akzeptiert und genutzt. Die ISO-Norm definiert Qualität als die Gesamtheit von Merkmalen einer Einheit bezüglich ihrer Eignung, festgelegte und vorausgesetzte Erfordernisse zu erfüllen (ISO8402)[ISO00].

Die Aufgabe des Qualitätsmanagements bei der Softwareentwicklung ist es, sich nicht nur mit der Kontrolle auf Fehler am Endprodukt, die dann beseitigt werden müssen, sondern auch mit der Untersuchung der Ursache solcher Fehler zu beschäftigen. Auf diese Weise können Strategien entwickelt werden, diese Fehler zu vermeiden bzw. das Risiko ihres Auftretens zu verringern. Der Umschwung dieses Konzepts vom elementorientierten System der Qualitätssicherung zu dem prozessorientierten Qualitätsmanagement ist in der Softwareindustrie erst mit der Einführung der ISO9000-Normen geschehen. Dieser prozessorientierte Ansatz des Qualitätsmanagements geht von dem Grundsatz aus, dass ein qualitativ hochwertiger Prozess auch mit hoher Wahrscheinlichkeit ein qualitativ hochwertiges Produkt hervorbringt [Ste00].

In einer Projektgruppe ist das primäre Ziel nicht das Erreichen eines möglichst hohen Qualitätsstandards, sondern die Entwicklung eines funktionsfähigen Prototypen. Ein hohes Qualitätsniveau kann hier vor allem aus Zeitgründen nicht gewährleistet werden. Das bedeutet aber nicht, dass das Thema Qualität in einer Projektgruppe komplett vernachlässigt wird.

Das Qualitätsmanagement der Projektgruppe Com42Bill legt den Schwerpunkt auf die Qualitätssicherung und die Optimierung der Softwareentwicklung. Für die Qualitätssicherung war das Durchführen eines Testprojektes, das ausführlich in einem Testplan dokumentiert wurde, ein elementares Hilfsmittel. In dem Testprojekt wurden mehrere Testphasen angesetzt, die teilweise schon während der Implementierungsphase eingesetzt haben, um Fehler rechtzeitig finden und beheben zu können. Die Testphasen begannen auf Klassenebene mit Codeinspektionen und Klassentests ausgewählter Klassen. Besonders ergiebig war die darauf folgende Integrationstestphase, in der die Schnittstellen zwischen den einzelnen Komponenten getestet und in mehreren Iterationen anschließend korrigiert werden mussten. Abschließend folgte ein Gesamtsystemtest, bei dem die Funktionalität des Systems im Vordergrund stand.

Für die Optimierung der Softwareentwicklung als Ganzes wurde nach dem prozessorientierten Ansatz des Qualitätsmanagements ein Prozessmodell aufgestellt. Diese lehnt sich an das Wasserfallmodell mit Rückkopplung an. Mit Hilfe des Prozessmodells wurde der Softwareentwicklungsprozess dokumentiert und analysiert.

10.2 Prozessmodell

Das Prozessmodell bildet die für die Softwareentwicklung getätigten Aktionen ab und stellt einen Zusammenhang zwischen den Ergebnissen der Tätigkeiten und davon abhängenden Prozessen dar. Ein Prozess ist ein Satz von in Wechselbeziehung oder Wechselwirkung stehenden Tätigkeiten, der Eingaben in Ergebnisse umwandelt (DIN EN ISO 9000:2000) [ISO00]. Am Prozessmodell kann die Wertschöpfungskette des Entwicklungsprozesses nachvollzogen werden und Schwachpunkte analysiert werden, um den Prozess für zukünftige Projekte zu optimieren, es bietet somit eine Grundlage für die kontinuierliche Verbesserung der Softwareentwicklung. Das hier beschriebene Prozessmodell dokumentiert den Entwicklungsprozess der Projektgruppe.

10.2.1 Symbolik

Für die Dokumentation des Softwareentwicklungsprozesses der Projektgruppe Com42Bill wurde eine Darstellung gewählt, die an der Petrinetz-Notation angelehnt ist. Diese Notation folgt der Definition von Prozessen nach der ISO-Norm, aus einer oder mehreren Eingaben wird mittels einer Tätigkeit ein Ergebnis gewonnen.

Tätigkeiten werden mittels eines Rechtecks dargestellt, die Eingaben und die Ergebnisse dieser Tätigkeiten durch Kreise. Komplexe Tätigkeiten werden der Übersichtlichkeit halber in den Diagrammen zunächst durch ein Rechteck, das von einem grauen Rahmen umgeben ist, symbolisiert. Diese Tätigkeiten werden dann in weiteren Grafiken aufgelöst. Die Eingaben und Ausgaben, die außerhalb dieser Verfeinerung erstellt bzw. gebraucht werden, sind durch gestrichelte Linien gekennzeichnet. Ein Beispiel für die diese Darstellung ist in Abbildung 66 zu sehen.

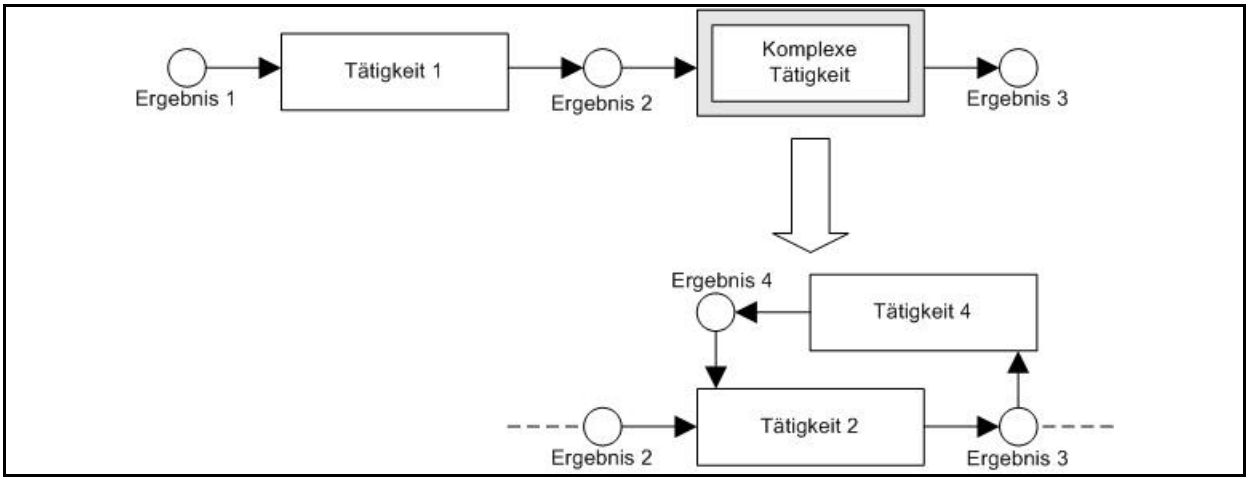


Abbildung 66: Notation

10.2.2 Entwicklungsprozess

Das Diagramm *Entwicklungsprozess*, das in Abbildung 67 gezeigt wird, stellt die Abläufe des gesamten Softwareentwicklungsprozesses dar. Ausgangspunkt der Softwareentwicklung war die in der Projektgruppenbeschreibung skizzierte Produktidee. Ausgehend von dieser Produktidee wurden Vorträge über möglicherweise benötigte Technologien und Konzepte gehalten; die Ergebnisse dieser Vorträge sind in den Seminarausarbeitungen nachzulesen. Ausgehend von diesen Ausarbeitungen und unter Berücksichtigung des zu erreichenden Endergebnisses wurde eine Entscheidung über das zu entwickelnde System gefällt. Die Wahl fiel auf ein Thin-Consolidator-System. Sehr früh in der Entwicklungsphase wurde die Entscheidung getroffen, das System komponentenbasiert zu erstellen. Das System wurde in benötigte Komponenten aufgeteilt und Teams mit der Verantwortlichkeit für die Komponenten betraut. Der weitere Verlauf des Softwareentwicklungsprozesses kann in folgende wichtige Phasen unterteilt werden: Anforderungsanalyse, Prototyping und Implementierung. Für den Beginn dieser Phasen waren noch unterstützende Prozesse, wie die Auswahl von Key-Features und das Erstellen eines Klassenmodells für das Prototyping, notwendig. Begleitend zur Phase der Implementierung setzen auch die bereits angesprochenen Tests und Reviews ein, um Fehler möglichst früh zu entdecken. Auf die einzelnen Phasen sowie auf die unterstützenden Prozesse wird in den folgenden Abschnitten ausführlich eingegangen.

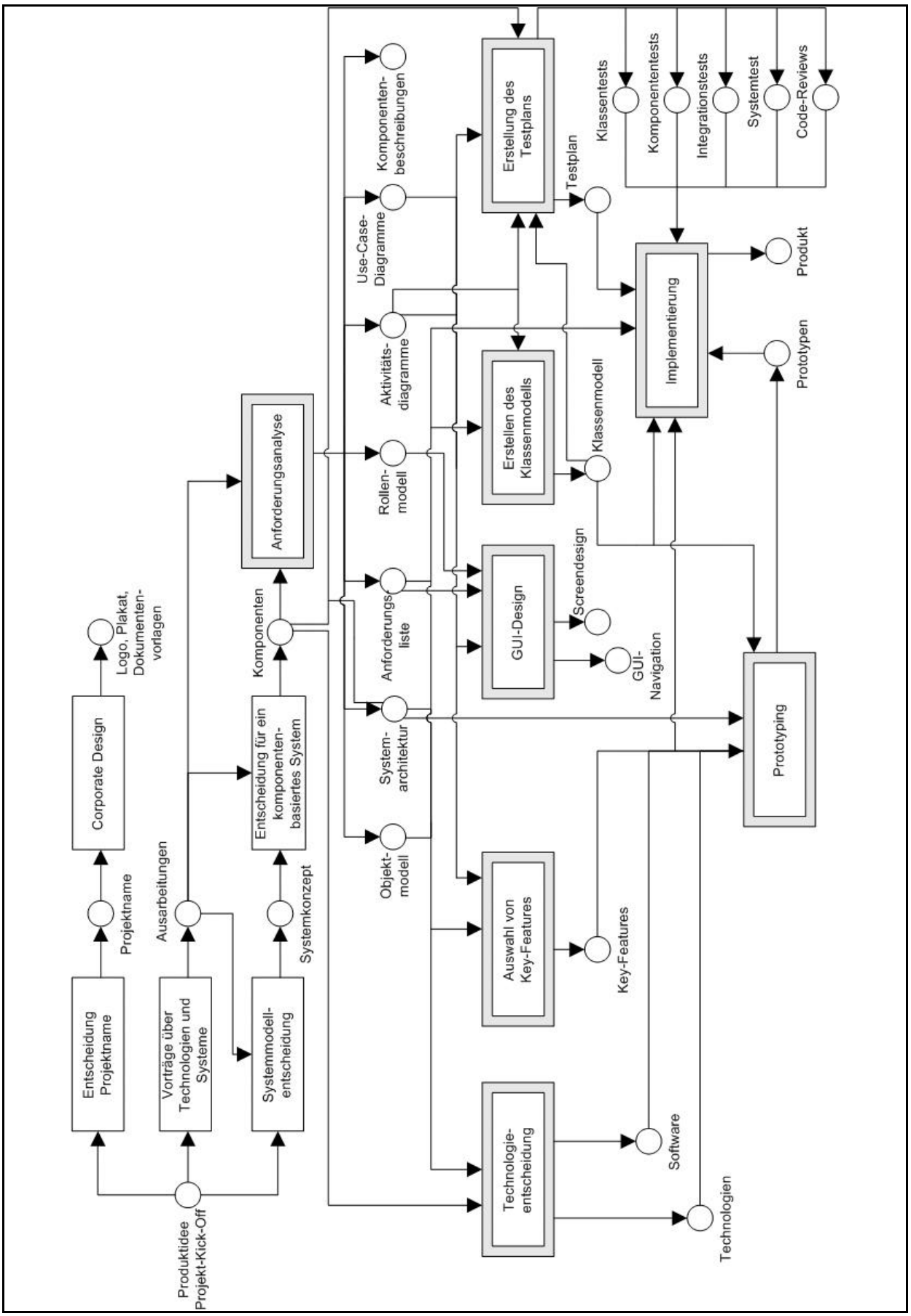

Produktidee
Projekt-Kick-Off
Entscheidung Projektname
Vorträge über Technologien und Systeme
Systemmodell-entscheidung
Projektname
Ausarbeitungen
Systemkonzept
Corporate Design
Logo, Plakat, Dokumenten-vorlagen
Entscheidung für ein komponenten-basiertes System
Komponenten
Anforderungsanalyse
Anforderungs-liste
Rollen-modell
Use-Case-Diagramme
Aktivitäts-diagramme
System-architektur
Objekt-modell
Komponenten-beschreibungen
Erstellung des Testplans
Testplan
Klassentests
Komponententests
Integrationstests
Systemtest
Code-Reviews
Erstellen des Klassenmodells
Klassenmodell
GUI-Design
Screendesign
GUI-Navigation
Implementierung
Produkt
Prototypen
Prototyping
Auswahl von Key-Features
Key-Features
Technologie-entscheidung
Software
Technologien

Abbildung 67: Entwicklungsprozess

10.2.3 Anforderungsanalyse

Als eine der aufwändigsten Aufgaben des Entwicklungsprozesses hat sich die Anforderungsanalyse herausgestellt. Ausgehend von der erfolgten Komponenteneinteilung und den Ausarbeitungen, aus denen erste Aufgaben des Systems und der einzelnen Komponenten herausgearbeitet werden konnten, wurden erste Anforderungen erstellt. Es wurde aber sehr schnell deutlich, dass Unklarheiten über die Zusammenarbeit der Komponenten, die späteren Anwendungsfälle und deren Ablauf eine Konkretisierung der Anforderungen erschwerte. Aus diesem Grund wurden für das bessere Verständnis notwendige Dokumente wie die Systemarchitektur und eine textuelle Beschreibung der Komponenten angefertigt. Um die Aufgaben des Endproduktes und damit auch die Aufgaben einzelner Komponenten besser verstehen zu können, wurden in einem weiteren Schritt ein Architekturmodell, Use-Case-Diagramme, Aktivitätsdiagramme, ein Rollenmodell und in späteren Iterationen auch bereits ein Objektmodell entworfen. Mehrere dieser Iterationen, in denen die jeweiligen Erkenntnisse aus den Modellen in die Anforderungen und wiederum weitere notwendige Ergänzungen in den Anforderungen wieder in die Modelle übernommen wurden, waren nötig, um die Anforderungslisten zu komplettieren.

Die Erstellung der Anforderungsliste und der Modelle waren aufgrund häufiger Reviews und Ergänzungen relativ komplexe Prozesse. Sie werden in den nächsten beiden Abschnitten noch genauer erklärt.

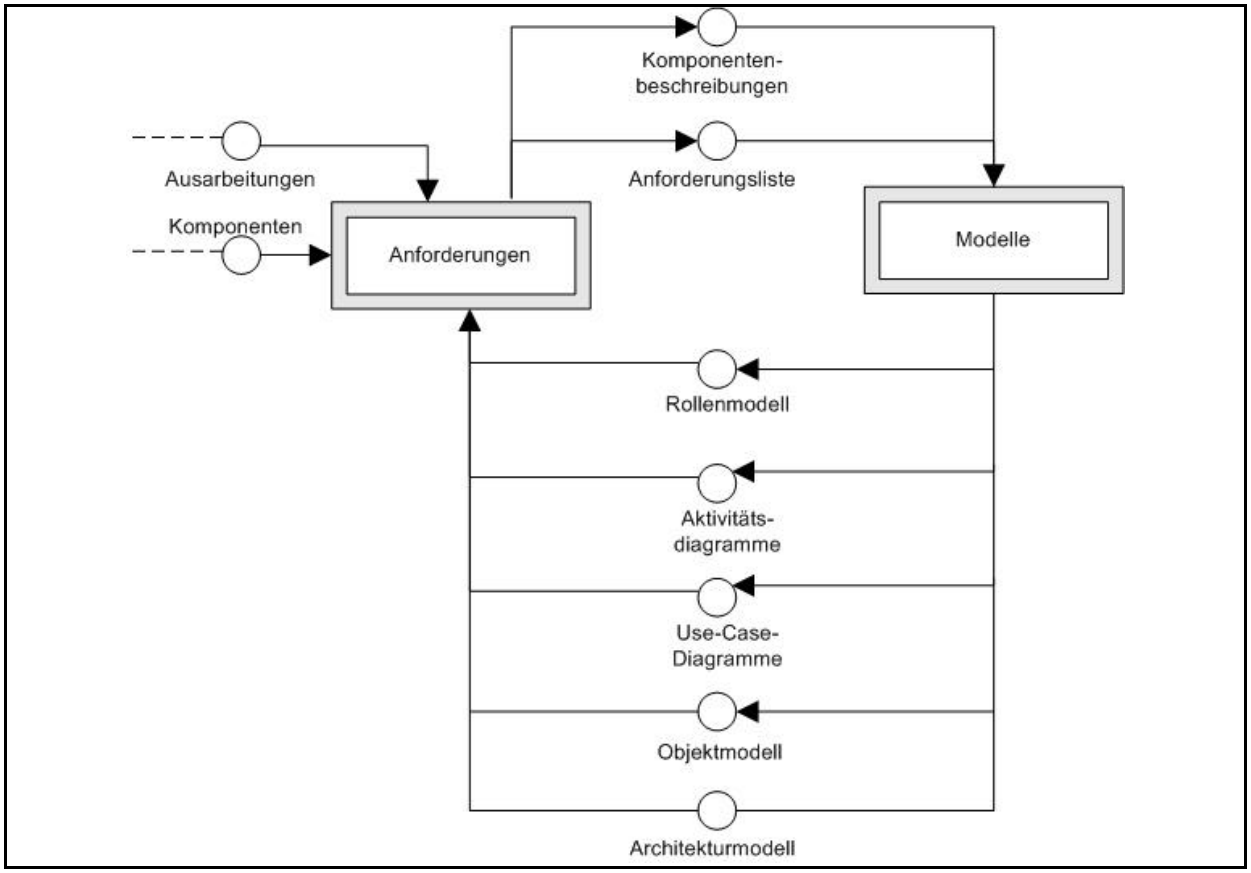


Abbildung 68: Anforderungsanalyse

10.2.4 Anforderungen

Ausgehend von der Entscheidung für Komponenten wurde versucht, Anforderungen an diese Komponenten aufzustellen. Für ein besseres Verständnis der Komponenten wurden zunächst textuelle Beschreibungen der Komponenten erstellt. Diese Beschreibungen wurden durch Reviews auf erste Fehler und auf Unstimmigkeiten überprüft und verbessert. Bereits bei der Erstellung dieser Beschreibungen wurde klar, dass man einheitliche Begrifflichkeiten für die vorkommenden Benutzerrollen benötigt, die durch das Rollenmodell eingeführt wurden. Die Komponentenbeschreibungen waren eine Grundlage für die Anforderungen an die Komponenten. Diese Anforderungsliste durchlief mehrere Iterationen, um die Anforderungen auf ein einheitliches Detailniveau zu bringen. In späteren Iterationen wurden die Anforderungen durch die Erkenntnisse aus den entwickelten Modellen ergänzt.

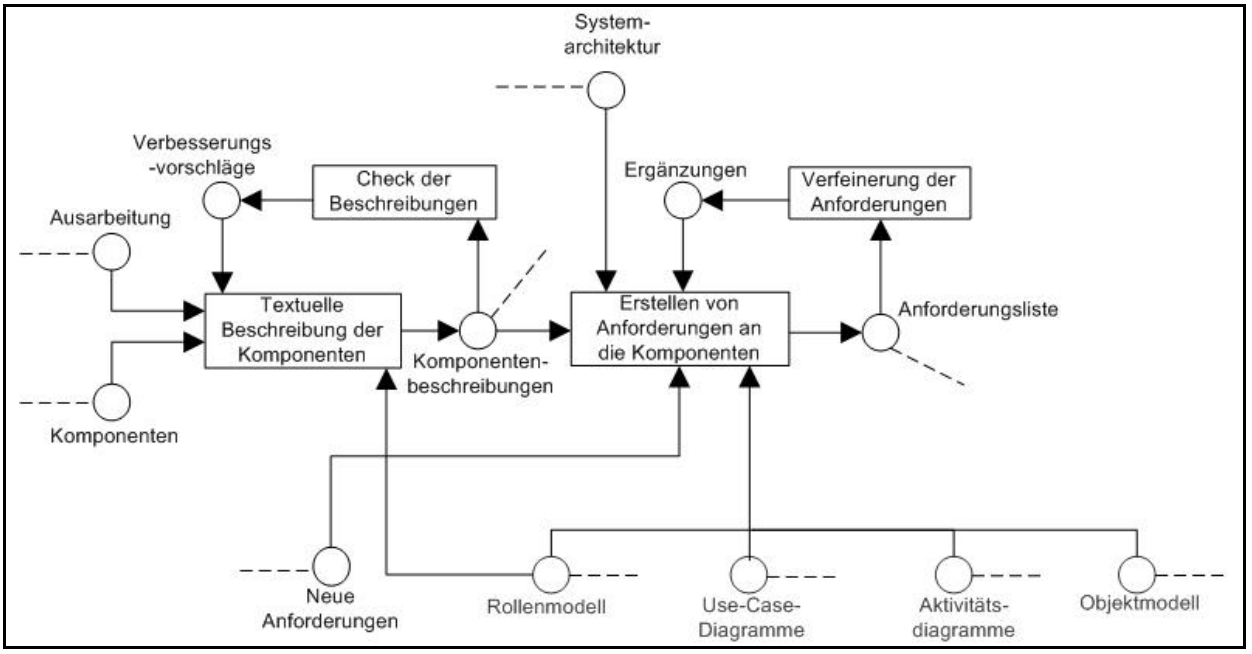


Abbildung 69: Anforderungen

10.2.5 Modelle

Ergänzend zu den Komponentenbeschreibungen und um für zukünftige Dokumente eine einheitliche Bezeichnung für die Benutzerrollen des Systems zu haben, wurde ein Rollenmodell erstellt.

Um einen Überblick über die Zusammenarbeit der Komponenten zu erhalten, wurde ein Architekturmodell erstellt. Durch neue Anforderungen wurden in weiteren Iterationen Anpassungen des Architekturmodells notwendig.

Für ein besseres Verständnis der Funktionen von Com42Bill wurden Use-Case-Diagramme entwickelt. Als weitere Verfeinerung wurden die Funktionen durch Aktivitätsdiagramme beschrieben. Die Erkenntnisse aus den Diagrammen über die Funktionsweise gingen in die weiteren Iterationsstufen zur Überarbeitung der Anforderungsliste ein.

Bereits bei den Anforderungen wurde versucht festzustellen, welche Daten die Komponenten dauerhaft speichern müssen. Diese Daten wurden in dem Objektmodell festgehalten, das durch ständige Reviews ergänzt und überarbeitet wurde.

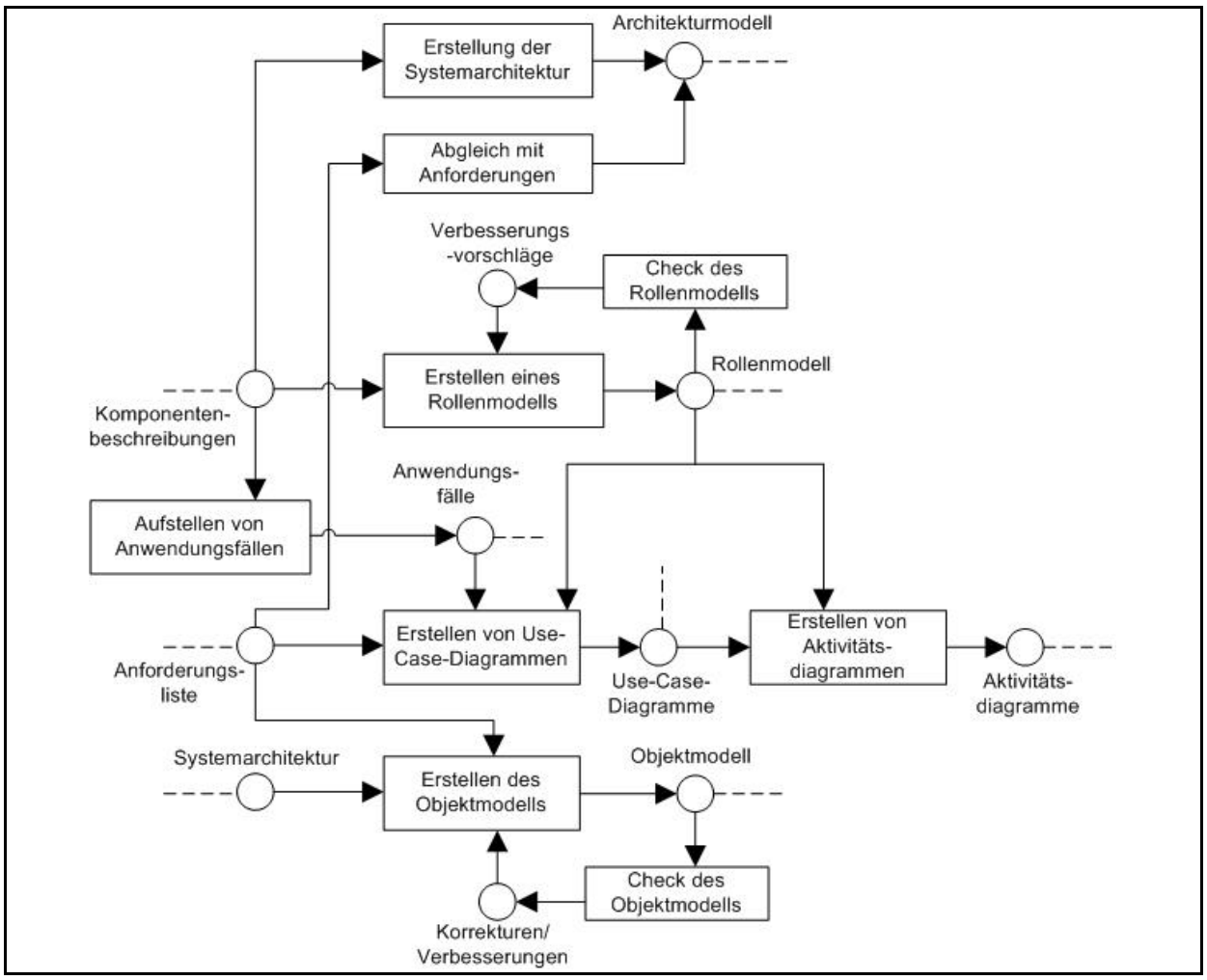


Abbildung 70: Modelle

10.2.6 Technologieentscheidung

Bereits in einer frühen Iterationsstufe der Erstellung der Anforderungsliste und der Modelle wurden Entscheidungen über die einzusetzende Software und die Softwaretechnologien getroffen. Aufgrund der Komplexität des Zusammenspiels der Komponenten und der zugehörigen Subkomponenten fiel die Entscheidung nach Auswertung der Features und der Verfügbarkeit der Produkte auf einen Applicationserver und einen Datenbankserver. Auch eine Entwicklungsumgebung wurde so bestimmt.

Die Auswahl von Softwaretechnologien konnte ebenfalls aufgrund der Aufgaben, die sich aus den Anforderungslisten ergaben, getroffen werden. So wurde von der Komponente Business Logic beispielsweise eine Pipelinearchitektur für die Umsetzung der Geschäftsprozesse gewählt.

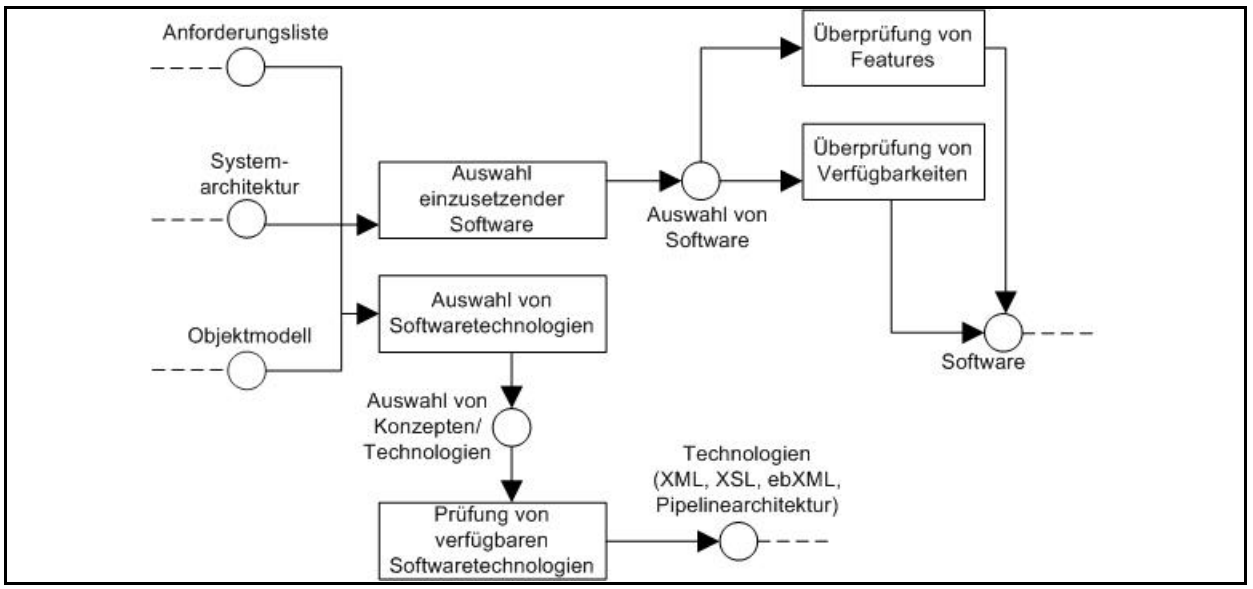


Abbildung 71: Technologieentscheidung

10.2.7 Auswahl von Key-Features

Aus den Ergebnissen der Anforderungsanalyse wurden für die Entwicklung von Prototypen Key-Features ausgewählt. Nachdem diese Key-Features zunächst unabhängig von der Zusammenarbeit der Komponenten untereinander bestimmt wurden, war eine Überarbeitung für einen wünschenswerten frühen Workflowtest nötig. Für ein effizientes Prototyping wurde ein Zeitplan für die Entwicklung der Prototypen, unter dem Gesichtspunkt, eine frühe Zusammenarbeit der Komponenten zu ermöglichen, erstellt.

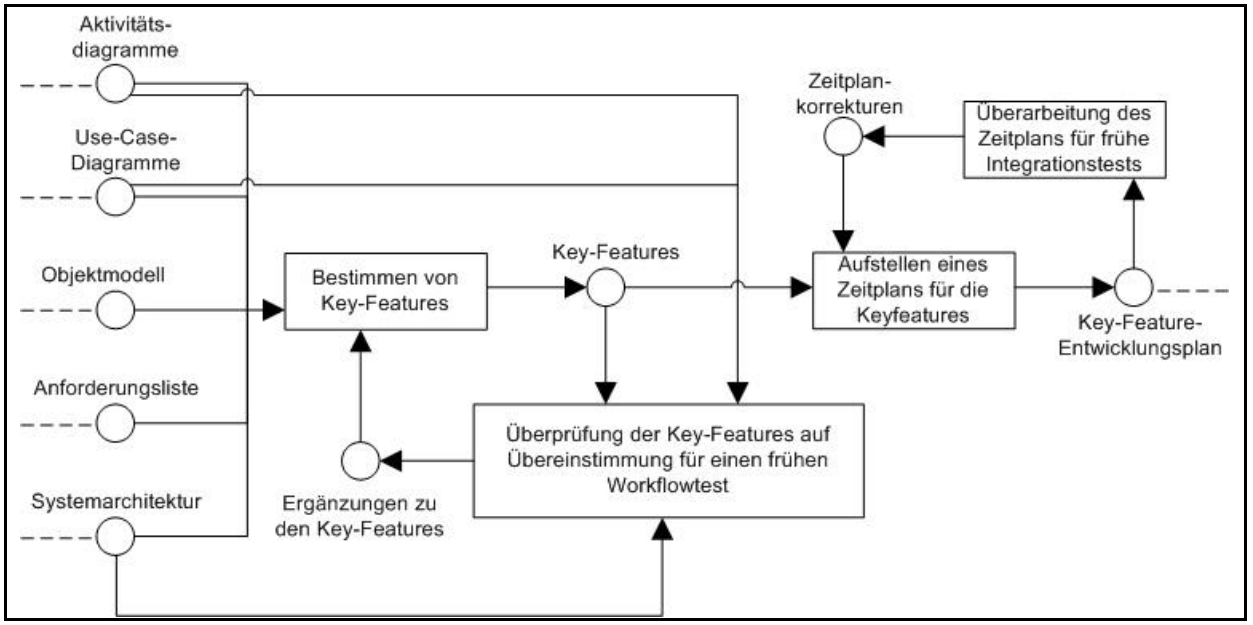


Abbildung 72: Auswahl von Key-Features

10.2.8 GUI-Design

Die Erstellung der Benutzeroberflächen teilte sich in zwei Bereiche, zum einen die Bestimmung eines einheitlichen Aussehens der Webseiten, einem Style-Guide, und dem Aufstellen eines

Navigationslayouts, an dem die Funktionen der Webseiten abgelesen werden können. Die Anforderungen an den Style Guide wurden bereits in der Anforderungsliste gestellt. Das Navigationslayout ergab sich aus den Benutzerrollen und den Funktionen, die durch die Use-Case-Diagramme und die Aktivitätsdiagramme beschrieben wurden. Die Zusammenführung des Style Guide und des Navigationslayout ergaben das endgültige Screendesign der Benutzeroberfläche.

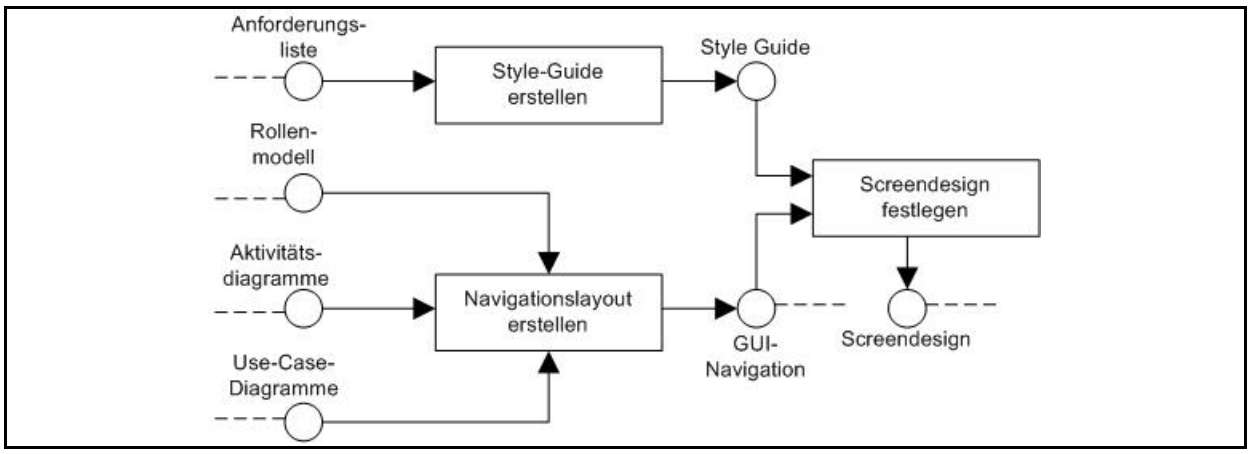


Abbildung 73: GUI-Design

10.2.9 Klassenmodell

Für das Prototyping und die Implementierung wurde ein Klassenmodell benötigt. Dieses ergab sich sehr leicht aus dem Objektmodell und der Systemarchitektur, die erforderlichen Schnittstellen und Übergabeparameter konnten aus den Anforderungen und den Aktivitätsdiagrammen ermittelt werden. Eine Kontrolle und Überarbeitung des Klassenmodells und besonders der Schnittstellen waren für eine spätere Zusammenarbeit der Komponenten unabdingbar.

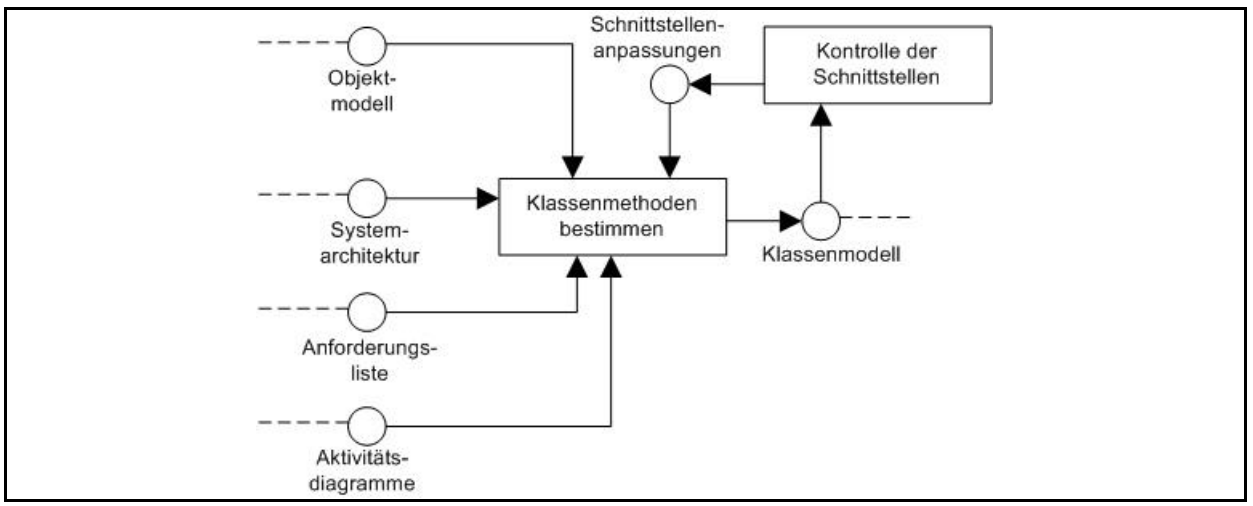


Abbildung 74: Klassenmodell

10.2.10 Testplan

Um Fehler in den Klassen und der Zusammenarbeit der Komponenten zu erkennen, sind Tests notwendig. Für systematische Tests, die die Qualität des Produkts Com42Billl sichern sollen, ist die Erstellung eines Testplans sinnvoll. Je früher ein Fehler in der Softwareentwicklung entdeckt

wird, desto leichter ist er zu beheben. Aus diesem Grund wurde entschieden, die Tests inkrementell aufeinander aufzubauen. Die Tests wurden deshalb in vier Testphasen eingeteilt. In der ersten Testphase wurden Whiteboxtests wie Code-Reviews und Klassentests geplant. Durch diese Tests soll die Fehlerfreiheit in den einzelnen Klassen der Komponenten gewährleistet werden. In der zweiten Phase sollten Komponententests, die als Blackboxtests geplant waren, durchgeführt werden. Diese Testphase wurde jedoch aufgrund von Problemen bei der Integration der Komponenten aus Zeitgründen verworfen. Die Zusammenarbeit der Komponenten wurde in der Integrationstestphase überprüft. Hierfür wurden Greyboxtests angesetzt, bei denen die Schnittstellen zwischen den Komponenten beobachtet werden. Abschliessend erfolgte in der letzten Phase ein Gesamtsystemtest. Für die einzelnen Tests wurden zunächst die Testlinge ausgewählt und danach Testszenarien ausgearbeitet. Die Verantwortlichen für die jeweiligen Tests und die zeitliche Planung wurden in einem Testplan festgehalten.

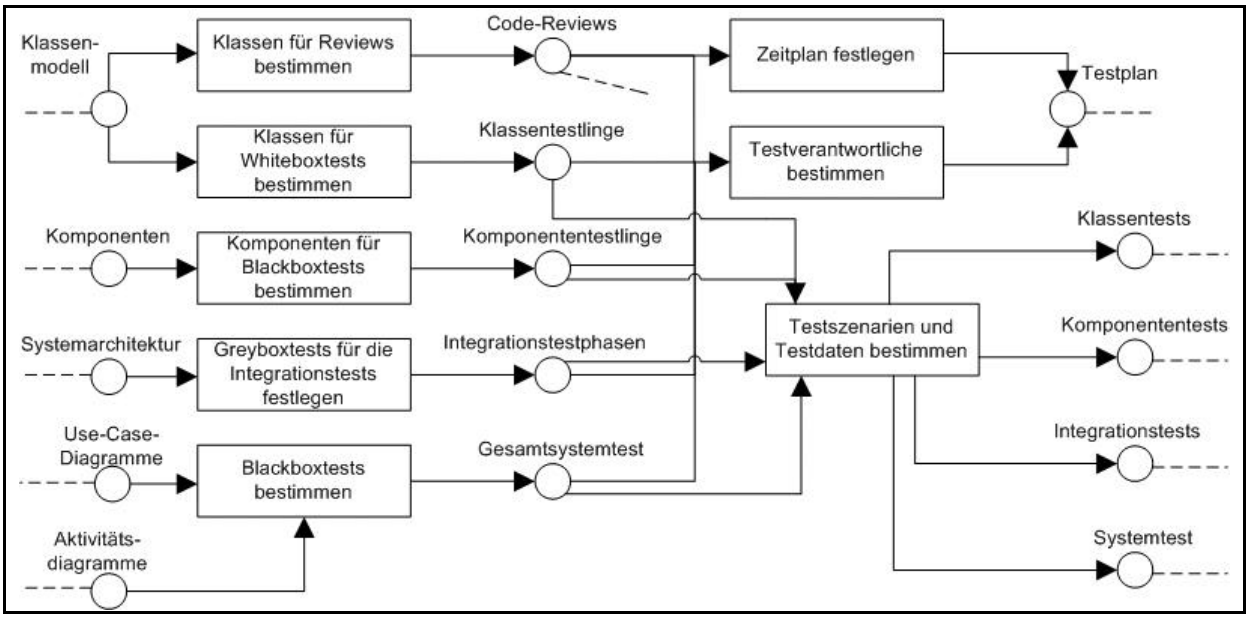


Abbildung 75: Testplan

10.2.11 Prototyping

Das Prototyping war in zwei Stufen unterteilt. Die erste Stufe diente der Prüfung der verwendeten Technologie und der Machbarkeit. Das Ergebnis dieser Stufe waren bereits rudimentäre Prototypen. Erkenntnisse, die man aus der Implementierung dieser Prototypen gewonnen hatte, konnten wiederum zu einer Veränderung dieser Prototypen führen, so dass hier mehrere Iterationen möglich waren.

War die technische Machbarkeit geklärt, so bestand die zweite Stufe aus einer Implementierung funktionaler Prototypen. Auch die Erkenntnisse aus diesen Prototypen gingen in die Verbesserung ein. Die funktionalen Prototypen sollten bereits in dieser Phase des Prototyping zu einem ersten Workflowtest zusammengeführt werden.

Die Auswertung der Prototypen kann in einem ungünstigen Fall einen Rückschritt zu den Anforderungen notwendig machen, wenn man erkennt, dass das System so nicht umsetzbar ist.

Aufgrund technischer Schwierigkeiten konnten nicht alle Komponententeams die Prototypenphase zum Abschluss bringen, so dass der geplante Workflowtest während des Prototypings nicht zustande kam.

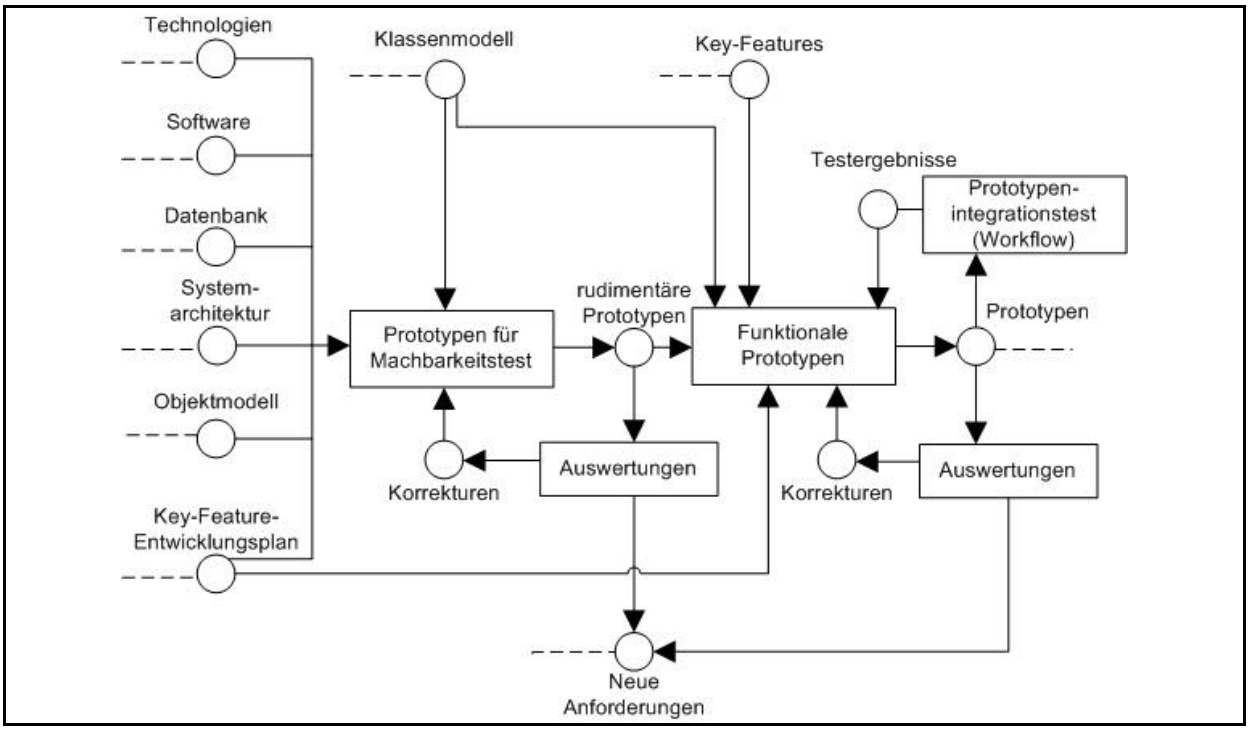


Abbildung 76: Prototyping

10.2.12 Implementierung

Die Implementierung und das Testen stellen die letzten Phasen des Entwicklungsprozesses dar. Mit den Ergebnissen des Prototyping sollte die Implementierung schnell zu funktionsfähigen Komponenten führen. Bereits während der Implementierung der Komponenten setzen die im Testplan festgehaltenen Code-Reviews und Klassentests an.

Eine Vorabintegration der Komponenten erfolgte schon während der Entwicklung, um die Schnittstellen der Komponenten aufeinander abzustimmen. Die Ergebnisse, die sich aus der Vorabintegration ergaben, gingen in die weitere Implementierung ein. Die vollständige Integration der Komponenten erfolgte danach. Das entstandene Gesamtsystem wurde nun mit Hilfe der geplanten Integrationstests überprüft. Die bei den Integrationstests entdeckten Fehler mussten bei der Implementierung der Komponente, in der die Fehler entdeckt wurden, behoben werden. Bei diesen Tests ging es vor allem darum, die Schnittstellen zwischen den Komponenten zu testen.

Nach dem Integrationstest folgte ein Gesamtsystemtest. Fehler, die bei diesem Test entdeckt wurden, mussten selbstverständlich wieder in den Komponenten selbst korrigiert werden. Auch bei diesem Test lag ein besonderes Augenmerk auf der fehlerfreien Zusammenarbeit der Komponenten.

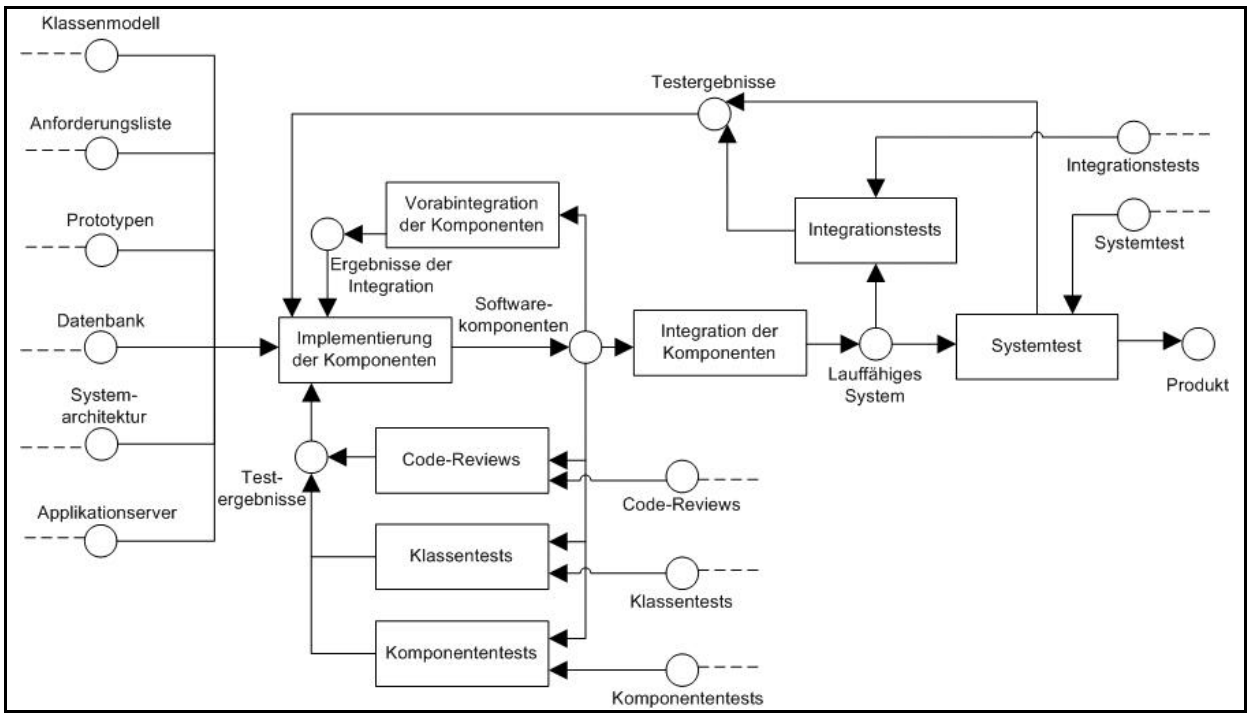


Abbildung 77: Implementierung

Betrachtet man den gesamten Entwicklungsprozess, so erkennt man, dass die einzelnen Phasen der Entwicklung aufeinander aufbauen. Rückschritte zu einer vorherigen Phase, also eine Iteration, sind erlaubt. Das Entwicklungsmodell der Projektgruppe lehnt sich an das allgemeinere Wasserfallmodell mit Rückkopplung an. Die Phasen des Modells sind dabei: Anforderungsanalyse, Prototyping, Implementierung, Test.

10.3 Qualitätssicherung

Für die Qualitätssicherung, also die Kontrolle der Software, wurde ein Testplan entwickelt. In diesem Testplan wurden Verantwortlichkeiten, Testmethoden, Testziele und Zeitpläne festgelegt. Die Tests wurden in mehreren Phasen, die aufeinander aufbauen, ausgeführt. Aufgrund der beschränkten Zeit der Projektgruppe konnte kein vollständiger Test in jeder Phase durchgeführt werden, so dass man sich auf exemplarische Tests an ausgewählten Testlingen beschränkte. Die einzelnen im Testplan angesetzten Phasen waren Code-Reviews, Klassentests, Komponententests, Integrationstests und ein Gesamtsystemtest.

10.3.1 Code-Inspektionen/ -Reviews

Ausgewählte Klassen wurden mittels Inspektionen überprüft und ihre Funktionen mit den Entwicklern diskutiert. Bei diesen Inspektionen erläuterte der Entwickler des jeweiligen Code-Stückes die Funktionen und die Art der Umsetzung. Durch Diskussionen und Vergleich mit den Anforderungen entdeckte Fehler wurden notiert, bewertet und später behoben. Die Auswahl der Klassen für die Inspektionen wurde nach den Kriterien getroffen, dass sie wichtig für die Gesamtfunktion des Systems sind und sich aufgrund ihres Zusammenspiels mit anderen Klassen oder ihrer Funktionsweise nicht für einen Klassentest eignen.

10.3.2 Klassentests

Als zweite Phase wurden Klassentests durchgeführt, bei denen ausgewählte Klassen der Komponenten mittels des Verfahrens der Zweigüberdeckung getestet wurden. Für die Tests wurden dementsprechend Eingabedaten für die Methoden der Klassen ausgewählt und beobachtet, ob das Verhalten den Erwartungen entspricht. Bei diesen Tests wurden Klassen

gewählt, die wichtige Entscheidungen im Ablauf des Systems treffen. Es konnten allerdings nicht von jeder Komponente Klassen getestet werden, denn der Klassentest verlangt, dass der Testling weitestgehend von anderen Klassen isoliert wird. Bei den Klassen der Business Logic war das nicht möglich, da die über 100 Beans erst durch die Workflowkonfiguration sinnvoll miteinander verknüpft werden und einzeln nur wenig Funktionen enthalten. Nur einige bei diesen Tests entdeckte Fehler waren kritisch und mussten behoben werden.

10.3.3 Komponententests

In einer folgenden Phase sollte ein Komponententest durchgeführt werden, der aber ausgesetzt wurde. Die Gründe dafür waren Probleme bei der Integration der Komponenten, die erforderten, dass die Komponenten während der Integration stetig auf die Zusammenarbeit mit anderen Komponenten angepasst werden mussten. Es bestand also kein abschließender Zustand, der zu diesem Zeitpunkt hätte getestet werden können. Aus Zeitgründen wurde diese Testphase komplett verworfen und der Integrationstest vorgezogen.

10.3.4 Integrationstest

Der Integrationstest stellte aufgrund der Komplexität der allgemeinen Schnittstelle zwischen den Komponenten eine besondere Herausforderung. Die Möglichkeit einen destruktiven Test durchzuführen, um Fehler aufzudecken, wurde schnell verworfen, da die Anzahl der möglichen Eingaben den Testrahmen sprengen würde. Man entschied sich also für einen demonstrativen Test, um die Funktionsfähigkeit der Komponenten zu zeigen. Wie im Testplan vorgesehen wurde ein nachrichtenbasierter Test angesetzt, bei dem der Inhalt des Requests, des Übergabeobjekts zwischen den Komponenten, an kritischen Stellen des Systems ausgegeben wurde.

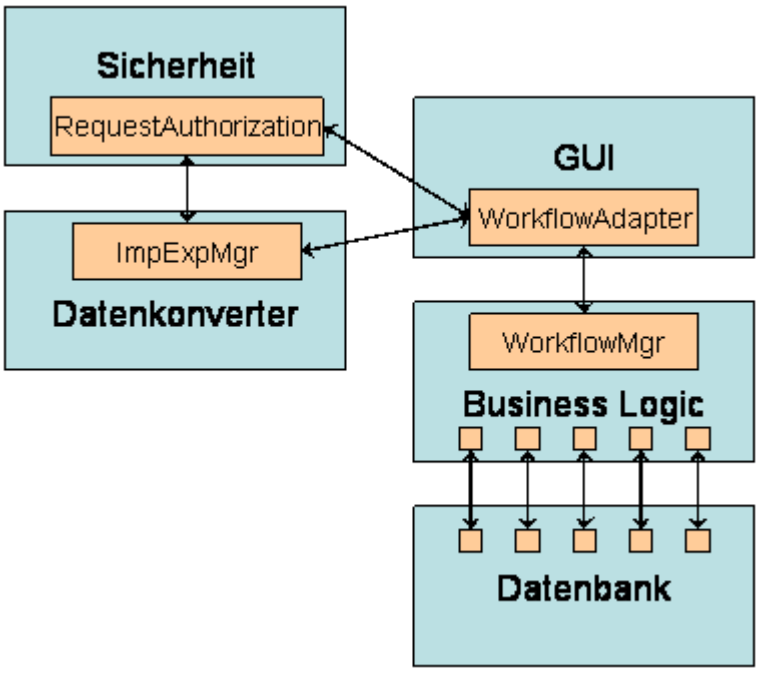


Abbildung 78: Komponentenschnittstellen

Diese Hotspots sind anhand der Systemarchitektur leicht zu erkennen, es sind die Schnittstellen zwischen den Komponenten. Auch bei diesem Test waren Einschränkungen zu machen, denn zwischen der Business Logik und der Datenbank gab es keine einheitliche Schnittstelle, an der Daten hätten abgegriffen werden können. Also Grundlage für den demonstrativen Test dienten die Use-Cases, die während des Systementwurfs herausgearbeitet wurden.

10.3.5 Gesamtsystemtest

In der letzten Testphase wurde der Gesamtsystemtest durchgeführt. Hierbei wurde der Schwerpunkt auf einen Funktionstest gelegt, von Performanz- und Belastungstests wurde Abstand gehalten. Der Funktionstest wurde als Black-Box-Test realisiert. Zunächst wurden alle Funktionen der Applikation, die in den Anwendungsfall- und Aktivitätsdiagrammen abgebildet worden sind, im Rahmen eines Funktionsflusstests überprüft. Dabei wurde über die beiden System-Schnittstellen, dem Web-Interface der GUI und dem Datenkonverter, für jeden spezifizierten Anwendungsfall eine entsprechende Oberflächenausprägung mit den benötigten Daten erzeugt. Anschließend wurden die aus dem jeweiligen Anwendungsfall resultierenden Ausgaben überprüft. Neben dem Funktionsflusstest wurde zusätzlich ein Syntaxtest durchgeführt. Hierbei wurden an den System-Schnittstellen absichtlich syntaktisch falsche Daten eingegeben, um zu kontrollieren, ob die Syntax richtig interpretiert wird und der Benutzer auf fehlerhafte Daten durch eine entsprechende Meldung hingewiesen wird.

11 Fazit

Die Teilnehmer der Projektgruppe Com42Bill konnten in ihrer einjährigen Arbeit viel Erfahrung sowohl im Umgang mit neuen Technologien, als auch im Umgang miteinander in einem Team sammeln. Die wohl wichtigsten Erfahrungen im Überblick:

Teamarbeit / Kommunikation:

Auch wenn die Trennung in Komponenten von der Sache und der eingesetzten Technik sinnvoll ist, sollte jeder immer daran denken, dass an einem Gesamtsystem gearbeitet wird. Wichtig ist der Blick „über den Tellerrand“, d.h. in die Arbeit und den Fortschritt der anderen Komponenten. Bei dem „Blick“ darf es jedoch auch nicht bleiben, eine permanente Absprache, welche Techniken eingesetzt werden und wie Schnittstellen aussehen ist dringend notwendig. Bei Schwierigkeiten kann so auf eine breitere Wissensbasis zurückgegriffen werden; abgesehen davon hilft ein konstruktives Gespräch oft mehr als gegenseitige Schuldzuweisungen. Die mangelnde Kommunikation wurde dadurch verstärkt, dass sehr selten zusammen entwickelt wurde. Jede Komponente hatte im privaten Bereich ihren eigenen Server aufgebaut; dies führte zu den oben erwähnten fehlenden Informationen über die anderen Komponenten. Der Hauptgrund für die Programmentwicklung auf privaten Rechnern dürfte bei der unzureichenden technischen Ausstattung und den unzumutbaren baulichen Gegebenheiten des Uni-Pools zu suchen sein.

Technik / Entwicklung:

Das eingesetzte Entwicklungswerkzeug, Together ControlCenter, ist auf den vorhandenen Rechnern nicht geeignet, ernsthaft Software zu entwickeln. Auf aktuellen Rechnern ist zwar eine fast flüssige Programmbedienung möglich, jedoch geraten auch hier einige Operationen, wie Übersetzen oder gar Deployment, zum Geduldsspiel.

Das Konzept des Applicationservers, wie er in deser PG eingesetzt wurde, fand breiten Anklang. Es wird ein umfangreiches Framework zur Verfügung gestellt, das dem Entwickler viel Arbeit abnimmt. Diese Arbeitserleichterung erkauft man sich allerdings mit zum Teil recht komplexen Konfigurationsvorgängen sowie zum Teil nicht nachvollziehbaren Verhaltensweisen des Servers beim Versuch, selbstgeschriebene Beans einzusetzen. Eine umfangreiche und tiefgreifende Einarbeitung in den eingesetzten Applicationserver ist vor dem Einsatz in einer Betriebsumgebung dringend nötig. Vor allem das Debugging gestaltet sich (noch) recht schwierig, da die eingesetzte Entwicklungsumgebung keinen Eingriff oder Einsicht in laufende Programme erlaubt.

Bevor man mit dem Programmieren beginnt, muss der zu programmierende Teil ordentlich spezifiziert werden. Alles, was hier eingespart wird, muss hinterher mehrfach angehängt werden. Um eine doch recht große Gruppe wie eine PG zu koordinieren, ist ein straffer Plan nötig, der allerdings auch von allen Teilnehmern ernst genommen werden muss. Natürlich kann es da zu Verschiebungen kommen; aber das Projektziel sollte nie aus den Augen verloren werden, sonst verschwendet man leicht zu viel Zeit.

Ausblick:

Die im Rahmen dieser PG erstellte prototypische Version von Com42Bill bietet an vielen Stellen Raum für Erweiterungen.

- Die Darstellung muss nicht auf PCs beschränkt bleiben. Eine Alternativdarstellung auf mobilen Endgeräten ist denkbar (und im Rahmen der umgesetzten Architektur auch leicht machbar).

- Der Datenkonverter kann leicht auf die verschiedensten Formate und Kommunikationskanäle von Rechnungsstellern und Finanzdienstleistern angepasst werden, ohne das ebXML-Rahmenwerk zugrunde legen zu müssen. Da innerhalb Deutschlands ebXML noch nicht verbreitet ist, wird dies vor einem „Echteinsatz“ auch nötig sein.

- Durch das Konzept der Business Logic sind Workflows recht einfach erweiterbar und anpassbar. Hier ist vor einem „Echteinsatz“ noch etwas Arbeit nötig, um wirklich alle Abläufe, die im täglichen Betrieb nötig sind, zu automatisieren.

- Sicherheit: Die Kommunikation zwischen dem Com42Bill-Server und den Benutzern geschieht derzeit über http. Diese Verbindung sollte, da ja sensible Daten übertragen werden, auf eine SSL-verschlüsselte Verbindung umgestellt werden. Die Möglichkeit hierzu ist im Applicationserver vorhanden; es muss jedoch noch ein öffentlicher Schlüssel bei einer der Zertifizierungsstellen (z.B. Thawte) bestellt werden.

Offene Fragen:

Die komplette Konzeption eines EBPP-Systems würde den Rahmen einer Projektgruppe sprengen. Wir haben uns bei unserer Arbeit hauptsächlich um die technische Umsetzung gekümmert, daher bleiben einige betriebswirtschaftliche Fragen offen:

- Kostenrechnung: Wie rechnet sich das System für den Betreiber? Wer trägt anfallende Kosten, und wie hoch sind diese?

- Zahlungssicherheit: Gibt es für den Rechnungssteller eine Zahlungsgarantie? Wie wird die für den Betreiber dargestellt?

- Zahlungen: Wer führt Zahlungen aus? Es gibt bislang keine ebXML-Schnittstelle zur deutschen Kreditwirtschaft; wie geht der Zahlungsverkehr vonstatten?

- Identität: Wie wird die Identität von Benutzern geprüft?

- Kredit: Wie werden Kreditlimits von Benutzern verwaltet?

Anhang A Anforderungsliste

A.1 Aufbau

ID	Identifiziert die Anforderung eindeutig, um auf sie referenzieren zu können (siehe Spalte „Abgedeckt durch“)	
Betroffene Komponente	Gibt an, welche Komponente diese Anforderung erfüllen muss.	
Abgedeckt durch	Gibt an, welche Anforderung einer Komponente diese Anforderung abdeckt. Mögliche Werte sind:	
	die *ID* einer Anforderung einer anderen Komponente,	falls diese Anforderung durch eine andere Komponente erfüllt wird.
	selbst	diese Anforderung muss von der eigenen Komponente erfüllt werden
	nicht abgedeckt	diese Anforderung ist an eine andere Komponente gerichtet ist, wird aber von ihr nicht erfüllt
Beschreibung	Beschreibt die Anforderung textuell, um Missverständnissen vorzubeugen.	
Begründung	Begründet, warum die Anforderung aufgestellt wurde.	
Datum	Gibt das Datum an, an dem die Anforderung aufgestellt wurde.	
Priorität	Gibt die Priorität der Anforderung an. Mögliche Werte sind Zahlen von 1 bis 4, wobei 1 die höchste Priorität beschreibt.	
Typ	Gibt an, ob eine Anforderung unbedingt erfüllt werden muss oder ob es sich um eine optionale Anforderung handelt. Mögliche Werte sind:	
	muss	die Anforderung muss unbedingt erfüllt werden
	kann	die Anforderung wird nur erfüllt, wenn zum PG-Ende hin noch genügend Zeit vorhanden ist.
Funktional (F)	Es wird ein „x“ eingetragen, falls die Anforderung funktional ist.	
Nicht-funktional (NF)	Es wird ein „x“ eingetragen, falls die Anforderung nicht-funktional ist.	
Schnittstellen-Relevanz (SR)	Es wird ein „x“ eingetragen, falls die Anforderung Auswirkungen auf eine Schnittstelle hat.	
Status	Gibt den aktuellen Status an, mögliche Werte sind:	
	in Arbeit	die Anforderung wurde noch nicht implementiert
	erledigt	die Anforderung wurde bereits implementiert

	entfällt	die Anforderung wird nicht mehr benötigt
Begründung für Status entfällt	In dieser Spalte steht die Begründung für ein eventuelles entfallen der Anforderung	

A.2 Datenkonverter (DK)

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ	F	NF	SR	Status	Begründung für Status entfällt.
DK-001	S	S-009	Die Sicherheitskomponente muss den Benutzer identifizieren und authentifizieren können (SSL).	Es dürfen nur berechtigte User Zugriff haben.	23.6.02	1	muss	x		x	erledigt	
DK-002	S	S-019	Die Sicherheitskomponente muss ausgehende Daten verschlüsseln und eingehende Daten entschlüsseln können (SSL).	Die Daten dürfen nicht durch Dritte gelesen werden.	23.6.02	1	muss	x		x	erledigt	wird von Weblogic automatisch erledigt, wurde aber noch nicht getestet
DK-003	S	S-020	Die Sicherheitskomponente muss anhand der digitalen Signatur die Unveränderheit (Integrität) der Daten und die Identität des Versenders (Authentizität) überprüfen.	Datensicherheit.	19.6.02	1	muss	x		x	entfällt	erfordert die Implementierung einer komplexen Infrastruktur, die den Rahmen der PG sprengen würde
DK-004	S	S-010	Die Sicherheitskomponente muss anhand einer ID einen Geschäftsprozess identifizieren, den der Datenkonverter dann über den Workflow-Adapter bzw. Workflow-Manager initiieren kann.	Der Datenkonverter muss entsprechende Geschäftsprozesse anstoßen können.	11.6.02	1	muss	x		x	erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ	F	NF	SR	Status	Begründung für Status entfällt.
DK-005	GUI	G-059	Die GUI muss im Workflow-Adapter eine Schnittstelle bereitstellen, die das Container-Objekt vom Datenkonverter empfangen und weiterleiten kann.	Der Datenkonverter muss entsprechende Geschäftsprozesse anstoßen können.	11.6.02	1	muss	x		x	erledigt	
DK-006	GUI	G-059	Die GUI muss dem Datenkonverter über den Workflow-Adapter ein Container-Objekt zur Verfügung zu stellen.	Der Datenkonverter ist verantwortlich, diese Daten zu konvertieren und zu versenden.	11.6.02	1	muss	x		x	erledigt	
DK-007	GUI	G-059	Die GUI muss dem Datenkonverter Statusmeldungen über den Verlauf der Geschäftsprozesse zur Verfügung stellen.	Der Datenkonverter muss den Status seines initiierten Geschäftsprozesses überprüfen können.	11.6.02	1	muss	x		x	erledigt	
DK-008	DB	DB-007; DB-009	Das DBMS speichert ebXML relevante Dateien (CPP von Com42Bill, CPAs, DTD's, Business Process Specification etc.)	Datenspeicherung	11.6.02	1	muss	x		x	entfällt	Die Dokumente werden nicht in der DB, sondern auf dem Server abgelegt, da sie der ebXML-Community zugänglich sein müssen (Registry-Ersatz)
DK-009	DK		Der Datenkonverter stellt der Sicherheitskomponente die nötigen Informationen zur Verfügung, um Requests authentifizieren zu können.	Der User muss eindeutig identifiziert werden	11.6.02	1	muss	x			entfällt	abgedeckt durch DK-032

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ	F	NF	SR	Status	Begründung für Status entfällt.
DK-010	DK		Der Datenkonverter stellt der Sicherheitskomponente die nötigen Informationen zur Verfügung (z. B. eine ID), um den entsprechenden Geschäftsprozess identifizieren zu können.	Der User muss eindeutig identifiziert werden	11.6.02	1	muss	x			entfällt	abgedeckt durch DK-032
DK-011	DK	selbst	Der Datenkonverter stellt der GUI (Workflow-Adapter) ein Container-Objekt bereit, welches die zu verarbeitenden Geschäftsdaten sowie den zugehörigen Geschäftsprozess enthält.	Weiterleitung von Geschäftsprozessen und Daten an die Business-Logic.	11.6.02	1	muss	x			erledigt	
DK-012	DK	selbst	Der Datenkonverter muss Daten abhängig vom Finanzdienstleister konvertieren und ihm bereitstellen können.	Überweisungsauftrag an Finanzdienstleister übermittelen	11.6.02	1	muss	x			erledigt	
DK-013	DK		Der Datenkonverter besitzt eine Subkomponente Registry	Verwaltung der ebXML-relevanten Dateien (CPAs etc.)	11.6.02	1	muss	x			entfällt	die Implmentierung einer ebXML Registry ist nicht das primäre PG-Ziel und hätte den Rahmen eindeutig gesprengt.
DK-014	DK	selbst	Der Datenkonverter besitzt eine Subkomponente MsgServiceInterface	Eingehende Daten empfangen und enpacken, ausgehende Daten verpacken und	11.6.02	1	muss	x			erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ	F	NF	SR	Status	Begründung für Status entfällt.
				versenden (gemäß ebXML Protokoll)								
DK-015	DK	selbst	Der Datenkonverter besitzt eine Subkomponente MsgServiceInterfaceHandler	Konvertierung; Kommunikation mit Sicherheitskomponente, DBMS über Registry, GUI (Workflow-Adapter)	11.6.02	1	muss	x			erledigt	
DK-016	DK	selbst	Erstellung eines CPPs für das Betreiber-System und ein Dummy-CPP für Testzwecke.	notwendig für ebXML	19.6.02	1	muss		x		erledigt	
DK-017	DK	selbst	Erstellung eines CPAs anhand der beiden CPPs.	notwendig für ebXML	19.6.02	1	muss		x		erledigt	
DK-018	DK		Es werden so wenig Daten wie möglich übertragen, d.h. die zu übertragenden Daten werden auf das nötigste beschränkt.	Performance und Datensicherheit.	19.6.02	1	muss		x		entfällt	abgedeckt durch DK-032
DK-019	DK		Der Datenkonverter kommuniziert nur über sichere Verbindungen.	Datensicherheit.	19.6.02	1	muss		x		entfällt	abgedeckt durch DK-032
DK-020	DK		Der Datenkonverter ist für die übertragenen Daten verantwortlich.	Datensicherheit.	19.6.02	1	muss		x		entfällt	abgedeckt durch DK-032
DK-021	DK		Der Datenkonverter speichert nur seine eigenen Daten in der Datenbank	Datensicherheit.	19.6.02	1	muss		x		entfällt	abgedeckt durch DK-032
DK-022	DK	selbst	Der Datenkonverter empfängt Rechnungsdaten vom Rechnungssteller, konvertiert und packt diese in ein Container-Objekt.	Weiterleitung von Geschäftsprozessen und Daten an die Business-Logic.	19.6.02	1	muss	x			erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ	F	NF	SR	Status	Begründung für Status entfällt.
DK-023	Externes ebXML-Modul	selbst	Eingehende Daten empfangen und entpacken, ausgehende Daten verpacken und versenden (gemäß ebXML Protokoll). Rechnungsstellr muss die Funktionen manuell ausführen.	Rechnungstellern, die kein ebXML-System besitzen, soll die Möglichkeit gegeben werden, Com42Bill zu nutzen.	11.6.02	4	kann	x			entfällt	war nur optional vorgesehen
DK-024	DB	nicht abgedeckt	Die Datenbank soll die Möglichkeit bereitstellen, gezielt auf einzelne XML-Tags zuzugreifen.	Performance	23.6.02	4	kann	x		x	entfällt	DK hält keine Daten in der DB, sondern nur im Filesystem des Webservers
DK-025	DK	selbst	Der Datenkonverter muss jede Datenübertragung protokollieren.	Nachverfolgung bei Problemen.	23.6.02	1	muss	x			erledigt	
DK-026	S	S-022	Die Sicherheitskomponente muss eine Public-Key-Infrastruktur bereitstellen.	Datensicherheit.	23.6.02	1	muss	x		x	entfällt	erfordert die Implementierung einer komplexen Infrastruktur, die den Rahmen der PG sprengen würde
DK-027	S	S-023	Die Sicherheitskomponente muss einen Mechanismus bereitstellen, um ausgehende Nachrichten digital zu signieren.	Datensicherheit.	23.6.02	1	muss	x		x	entfällt	erfordert die Implementierung einer komplexen Infrastruktur, die den Rahmen der PG sprengen würde

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ	F	NF	SR	Status	Begründung für Status entfällt.
DK-028	DK	selbst	Der Datenkonverter muss dem Sender den Empfang einer Nachricht durch eine Acknowledge Message bestätigen. Diese enthält Informationen (digitale Signatur), um die Identität und Integrität einer Nachricht überprüfen zu können.	Datensicherheit.	16.6.02	1	muss	x			erledigt	digitale Signatur fehlt
DK-029	DK	selbst	Der Datenkonverter muss Nachrichtenduplikate erkennen und ignorieren können.	Datensicherheit.	17.6.02	1	muss	x			entfällt	wurde aus Zeitgründen nicht implementiert, hat aber keine Auswirkung auf die Funktionalität
DK-030	DK	selbst	Die Gültigkeitsdauer einer Nachricht wird durch ein TTL-Attribut (Time To Life) festgesetzt.	Datensicherheit.	18.6.02	1	muss	x			entfällt	wurde aus Zeitgründen nicht implementiert, hat aber keine Auswirkung auf die Funktionalität
DK-031	DK	selbst	Der Datenkonverter benötigt ein Administrationstool, um interne Dateien verwalten zu können.	Es müssen Dateien manuell in der Datenbank abgelegt bzw. geändert werden.	19.6.02	1	muss	x		x	entfällt	Die Dokumente werden nicht in der DB, sondern auf dem Server abgelegt, da sie der ebXML-Community zugänglich sein

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ	F	NF	SR	Status	Begründung für Status entfällt.
												müssen (Registry-Ersatz)
DK-032	DK	selbst	Der Datenkonverter muss die Sicherheitsrichtlinien der Komponente Sicherheit berücksichtigen	Datensicherheit.	26.6.02	1	muss	x		x	erledigt	
DK-033	DK	selbst	Der Datenkonverter muss der Datenbank ein Objektmodell zur Verfügung stellen.	Abbildung der Daten des DK in der DB	26.6.02	1	muss		x	x	erledigt	
DK-034	GUI	G-090	Die GUI muss eine Benutzeroberfläche bereitstellen, mit deren Hilfe CPP/CPAs in die Datenbank eingefügt bzw. gelöscht und geändert werden können.	Administration	10.7.02	2	muss	x		x	entfällt	DK hält keine Daten in der DB, sondern nur im Filesystem des Webservers
DK-035	DK	selbst	Es müssen vom Rechnungssteller Statusinformationen über die erfolgte Bezahlung von Rechnungen importiert werden.	Rechnungspräsentation	#######	1	muss	x		x	erledigt	
DK-036	DK	selbst	Es müssen vom Rechnungssteller Informationen zu Stornierungen von Rechnungen importiert werden.	Rechnungspräsentation	#######	1	muss	x		x	erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ	F	NF	SR	Status	Begründung für Status entfällt.
DK-037	DK	selbst	Es müssen vom Finanzdienstleister Fehlermeldungen bei fehlgeschlagener Überweisungsausführung importiert werden.	Rechnungspräsentation	#######	1	muss	x		x	erledigt	
DK-038	DK	selbst	Versendete Nachrichten werden auf der Festplatte zwischengespeichert	Sicherheit.	#######	1	muss	x		x	erledigt	
DK-039	DK	selbst	Der Rechnungssteller erhält zu jeder importierten Rechnung, Storno sowie Statusinformation eine Statusmeldung über den Erfolg/Mißerfolg des jeweiligen Datenimports.	Sicherheit.	#######	1	muss	x		x	erledigt	
DK-040	DK	selbst	Der DK teilt der BL mit, wenn der Export einer Überweisung fehlgeschlagen ist.	Die BL kann des Bezahlungsstatus rückgängig machen.	#######	1	muss	x		x	erledigt	
DK-041	DK	selbst	Der Rechnungssteller erhält bei fehlgeschlagener Authentifizierung eine entspr. Fehlernachricht	Sicherheit.	#######	1	muss	x		x	erledigt	

A.3 GUI (G)

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
G-001	GUI	selbst	Informationen dürfen nicht durch das Layout der Seite in den Hintergrund gedrängt werden	Ein zu komplexes Layout verhindert eine einfache Wahrnehmung der dargestellten Informationen. Durch	28.4.02	3	muss		x		erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
				ein zu einfaches Layout wird die Seite uninteressant.								
G-002	GUI	selbst	Ein einheitliches Design auf allen Webseiten muss gewährleistet sein. (Corporate Design)	Ein einheitliches Layout hilft dem Benutzer, sich auch auf untergeordneten Seiten zurechtzufinden.	28.4.02	2	muss		x		erledigt	
G-003	GUI	selbst	Die Funktion und Bedienbarkeit von Web-Seiten darf nicht von Bildschirmauflösungen oder Farbtiefen abhängig sein.	Jeder Benutzer soll unabhängig von der verfügbaren Ausstattung die Seite nutzen können, wodurch einer Einschränkung der potentiellen Kunden vorgebeugt wird.	28.4.02	3	muss		x		erledigt	
G-004	GUI	selbst	Das Seitenlayout hat eine feste Breite.	Eine Konsistenz des Seiteninhalts wird gewährleistet, das Verschieben der einzelnen Elemente bei unterschiedlichen Auflösungen wird dadurch verhindert	28.4.02	3	kann		x		erledigt	
G-005	GUI	selbst	Die Layoutgestaltung beruht auf Templates.	Globale Änderungen im Layout können leicht über die Templates durchgeführt werden.	28.4.02	2	kann		x		erledigt	
G-006	GUI	selbst	Dialoge sind intuitiv erschließbar sein, d.h. umittelbar	Der Benutzer soll ohne Anleitung in der Lage sein, die Dialoge	28.4.02	2	kann		x		erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
			verständlich sein.	bedienen zu können.								
G-007	GUI	selbst	Dialoge unterstützen den Benutzer bei der Durchführung der Aufgabe ohne ihn zusätzlich zu belasten.	Effizientes Arbeiten mit den Dialogen muss möglich sein.	28.4.02	1	kann		x		erledigt	
G-008	GUI	selbst	Dialoge sind erwartungskonform sein, d.h. den Erwartungen des Benutzers entsprechen, die er aus Erfahrungen mit anderen Arbeitsabläufen bereits mitbringt und die er sich während der Benutzung des Systems angeeignet hat.	Ohne diese Voraussetzung kann der Benutzer die Dialoge nicht intuitiv bedienen.	28.4.02	3	kann		x		erledigt	
G-009	GUI	selbst	Dialoge müssen fehlerresistent sein, d.h. im Falle eines durch Benutzereingaben verursachten Fehlers muss das System stabil bleiben und dem Benutzer deutlich machen, welchen Fehler er begangen	Fehlerhafte Eingaben dürfen nicht toleriert werden und müssen vom Benutzer korrigiert werden, da sonst eine korrekte Abarbeitung der Eingaben nicht möglich ist.	28.4.02	1	muss	x			in Arbeit	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
			hat und ihm durch Ratschläge helfen, den aufgetretenen Fehler									
G-010	GUI	selbst	Der Text einer Seite ist leicht überschaubar durch z.B. aussagekräftige und hervorgehobene Überschriften und Unterteilung in Abschnitte.	Text enthält Informationen und sollte schnell einzuordnen und zu lesen sein.	28.4.02	2	kann		x		erledigt	
G-011	GUI	selbst	Informationen auf der Seite sind in der Reihenfolge ihrer Wichtigkeit angeordnet sein.	Die Erfassung wichtiger auf der Seite enthaltener Informationen soll zuerst erfolgen. Stößt der Benutzer zuerst auf Unwichtiges, so verlässt er die Seite evtl. vorzeitig.	28.4.02	4	kann		x		erledigt	
G-012	GUI	selbst	Überflüssige Animationen, Laufschriften und „top“-Technologien wie Flash dürfen nicht eingesetzt werden, wenn es alternative Standard-Darstellungsmöglichkeiten gibt.	Jeder Benutzer soll in der Lage sein, die Seite benutzen zu können - auch mit älteren Browsern oder auf anderen Betriebssystemen.	28.4.02	2	muss		x		erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
G-013	GUI	selbst	Grafische Symbole sind mit alternativen Textbeschreibungen versehen werden.	Wenn der Clientbrowser keine Bilder anzeigen kann, so soll der Benutzer immer noch in der Lage sein, sich auf der Seite zurechtzufinden.	28.4.02	3	kann		x		erledigt	
G-014	GUI	selbst	Grafiken und Multimedia-Objekte haben eine für jede Übertragungsart geeignete Dateigröße.	Die Ladezeit einer Seite muss in einem akzeptablen Rahmen liegen.	28.4.02	3	muss		x		erledigt	
G-015	GUI	selbst	Die gesamte Webpräsenz muss hierarchisch strukturiert sein.	Mithilfe einer Hierarchie kann der Benutzer die Struktur der Seite leicht erfassen.	28.4.02	1	muss		x		in Arbeit	
G-016	GUI	selbst	Die Navigation innerhalb der Hierarchie muss auf allen Seiten logisch zu erschließen und nachzuvollziehen sein.	Ohne einen logischen Aufbau kann der Benutzer den Aufbau der Seite nicht nachvollziehen.	28.4.02	1	muss		x		in Arbeit	
G-017	GUI	selbst	Verweise in der Hierarchie der Organisationsstruktur müssen deutlich erkennbar sein (z.B. Darstellung der gesamten Hierarchie in einem Navigationsbaum).	Der Benutzer muss schnell erkennen sein, wie er durch die Seiten navigieren kann.	28.4.02	2	muss		x		erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
G-018	GUI	selbst	Die aktuelle Position des Benutzers in der Hierarchie muss jederzeit erkennbar sein.	Nur so kann der Benutzer sich in der Organisation der Seiten orientieren.	28.4.02	1	muss		x		erledigt	
G-019	GUI	selbst	Im Fließtext werden Links nur sparsam eingesetzt.	Verknüpfungen im Text sind schwerer zu erkennen und tragen nicht zur Übersichtlichkeit bei. Durch häufige Links und damit Verzweigungsmöglichkeiten soll der Benutzer in seinem Lesefluss nicht gestört werden.	28.4.02	4	kann		x		erledigt	
G-020	GUI	selbst	Links beschreiben das verknüpfte Element möglichst exakt. Text-Links sind zu diesem Zweck prägnant formuliert, grafische Links haben Symbole.	So ist eine Benutzung dieser Elemente ohne zusätzliche Erklärungen möglich.	28.4.02	2	kann		x		erledigt	
G-021	GUI	selbst	Links müssen eindeutig gekennzeichnet sein: Text-Links durch farbige oder stilistische Hervorhebung (z.B. Unterstreichung), grafische Links	Links ermöglichen die Navigation zwischen den Seiten und müssen deshalb klar erkennbar sein.	28.4.02	2	muss		x		erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
			durch optische Effekte (z.B. Highlighting).									
G-022	GUI	selbst	Grafische Links müssen eine Alternative in Form von Text-Links haben.	Wenn der Clientbrowser keine Bilder anzeigen kann, so soll der Benutzer immer noch in der Lage sein, solche Links zu benutzen.	28.4.02	1	muss		x		in Arbeit	
G-023	GUI	selbst	Mit Ausnahme von Rechnungsdetails und evtl. Hilfen werden gelinkte Seiten nicht in einem neuen Fenster geöffnet.	Die Navigation mit Vor- und Zurückbuttons ist beim Öffnen neuer Seiten nicht möglich.	28.4.02	2	kann		x		erledigt	
G-024	GUI	selbst	Die Funktion des Back-Buttons muss gewährleistet sein, damit der Benutzer im Zweifelsfall jederzeit einen Schritt zurückgehen kann.	Bei Fehlern oder Unsicherheiten soll der Benutzer immer die Möglichkeit haben, zu einer vorhergehenden Seite zurückzukehren.	28.4.02	2	muss	x			erledigt	
G-025	S		Der Rechnungsempfänger hat die Möglichkeit sich mittels Benutzername und Passwort am System	Ein Authentifizierung ist für den Zugriff auf geschützte Daten unbedingt nötig.	28.4.02	1	muss	x		x	entfällt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
			anzumelden.									
G-026	GUI	selbst	Der Rechnungsempfänger wird in regelmäßigen Abständen daran erinnert sein Passwort zu ändern.	Eine regelmäßige Passwortänderung erhöht die Sicherheit.	28.4.02	4	kann	x			entfällt	
G-027	S	S-024	Die Sicherheit weist auf ein veraltetest Kennwort bei der Authentifizierung hin.	Diese Funktion löst G-026 aus.	11.6.02	3	kann	x		x	entfällt	
G-028	GUI	selbst	Über die GUI kann der Rechnungsempfänger Daten wie Adresse, Telefonnummer, Faxnummer, usw. ändern.	Eine Pflege der persönlichen Daten muss zur Kontaktaufnahme möglich sein.	28.4.02	3	muss	x			in Arbeit	
G-029	BL	BL-007	Die BL stellt Funktionen zum Ändern der Rechnungsempfängerdaten zur Verfügung	Ermöglicht G-028	11.6.02	2	muss	x		x	erledigt	
G-030	GUI	selbst	Der Rechnungsempfänger kann eine Rechnungsübersicht anfordern und einsehen.	Eine Rechnungsübersicht ist für die Überschaubarkeit der Rechnungen unabdingbar.	28.4.02	1	muss	x		x	erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
G-031	GUI/DB	selbst	Die dargestellten Rechnungen müssen einen eindeutigen Zustand haben, also „offen“, „fällig“, „bezahlt“ oder „in Bearbeitung“	Nur durch eindeutige Zustände kann der Status einer Rechnund schnell erkannt werden.	28.4.02	1	muss		x	x	erledigt	
G-032	GUI	selbst	In der Rechnungsübersicht können die Rechnungen nach Kriterien sortiert und gefilter werden.	Diese Maßnahme erleichtert die Übersicht über die Rechnungen.	28.4.02	3	kann	x		x	erledigt	
G-033	BL	BL-007	Die BL stellt eine Funktion für das Abfragen von Rechnungen nach Kriterien wie Zeitraum, Fälligkeit, Status, etc.	Diese Funktion ermöglicht G-030 und G-032	11.6.02	1	muss	x		x	erledigt	
G-034	GUI	selbst	Über Links der Rechnungen aus der Rechnungsübersicht wird man zu den bei dem Rechnungssteller bereit gestellten Details weitergeleitet.	Rechnungsdetails sind im lokalen System nicht verfügbar, deshalb muss zum Rechnungssteller weiterverlinkt werden.	28.4.02	2	muss	x			in Arbeit	
G-035	GUI	selbst	Rechnungen aus der Rechnungsübersicht können für den Zahlungsvorgang	Auf diese Weise können mehrere Rechnungen auf einmal bezahlt werden.	28.4.02	1	muss	x			in Arbeit	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
			markiert werden.									
G-036	GUI	selbst	Bei den Dialogen für die Bezahlung kann zwischen mehreren Zahlungsarten gewählt werden. Eine vorher gewählte Standardbezahlung ist die Grundeinstellung.	Die Wahl über die Art der Bezahlung ist für die Begleichung der Rechnung(en) notwendig.	28.4.02	2	muss	x			in Arbeit	
G-037	BL	BL-008	Eine Abfrage der möglichen Bezahlungsmöglichkeiten eines Rechnungsempfängers muss über die BL möglich sein.	Ermöglicht G-035	11.6.02	1	muss	x		x	in Arbeit	
G-038	GUI	selbst	Nach dem Abschluss der Bezahlung und der Bestätigung, dass der Vorgang bei der Business Logic ist, wird eine Bestätigung des Zahlungsvorgangs präsentiert.	Der Zahlungsvorgang ist erst abgeschlossen, wenn er bei der Business Logic eingegangen ist. Danach erhält der Benutzer die notwendige Bestätigung.	28.4.02	1	muss		x	x	in Arbeit	
G-039	BL	nicht abgedeckt	Die BL gibt nach dem Eingang eines Bezahlvorgangs eine Bestätigung an die GUI.	Ermöglicht G-038	11.6.02	1	muss	x		x	in Arbeit	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
G-040	GUI	selbst	Der Rechnungsempfänger kann zu jeder Funktion eine Hilfe aufrufen.	Eine notwendige Unterstützung des Benuters für den Fall, das er sich nicht mehr zurechtfindet.	28.4.02	3	kann	x		x	in Arbeit	
G-041	DB	DB-009	Die Hilfen werden in der DB abgespeichert.		11.6.02	1	muss	x		x	entfällt,	statische Texte wurden zur einfacheren Entwicklung ausserhalb der Datenbank gespeichert
G-042	DB	DB-007	Die DB bietet über ein entsprechendes Objekt einen Zugriff auf die Hilfen und Suchfunktionen in den Hilfstexten.	Über die Suchfunktionen kann der Rechnungsempfänger nach gewünschten Funktionen suchen.	11.6.02	1	muss	x		x	entfällt	alle Hilfen wurden durch Tooltip-Hilfen ersetzt
G-043	DK (Wieso Anforderung von GUI an DK?)		Der Rechnungssteller hat die Möglichkeit einen Datenaustausch ohne GUI durchzuführen.	Auf diese Weise wird eine Automatisierung des Übermittlungsvorgangs möglich. (DK!?)	28.4.02	2	kann	x		x	entfällt	für GUI, vom DK erledigt
G-044	S	S-017	Eine Authentifizierung ist für das Einsehen geschützer Daten notwendig.	Der Rechnungssteller kann sich am Webinterface mit einem Benutzernamen und Kennwort anmelden.	28.4.02	1	muss	x		x	in Arbeit	
G-045	GUI	selbst	Der Rechnungssteller kann eine Übersicht seiner Rechnungen	Dadurch kann er seine Rechnungen bezahlt/unbezahlt übersehen.	28.4.02	1	muss	x			erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
			einsehen.									
G-046	GUI	selbst	Der Rechnungesteller kann die Rechnungen in der Rechungsübersicht nach Kriterien sortieren und filtern	Eine Erleichterung bei der Übersicht der Rechnungen.	28.4.02	2	kann	x			erledigt	
G-047	BL		Die BL stellt eine Funktion für das Abfragen von Rechnungen nach Kriterien wie Zeitraum, Fälligkeit, Status, etc.	Diese Funktion erleichtert dem Rechnungssteller die Übersicht über seine Rechnungen.	11.6.02	1	muss	x		x	entfällt	der Rechnungsstell er hat keine eigene GUI (wurde aus Zeitgründen nicht mehr implementiert)
G-048	DK/BL		Informationen über die eigenen Rechnungen sollen in standardisierten Formen herunterladbar sein	Dies ist eine der Möglichkeiten Änderungen der Stati der eingespeisten Rechungen abzufragen.	28.4.02	4	kann	x		x	entfällt	optionales Feature aus Zeitmangel nicht umgesetzt
G-049	GUI	selbst	Das Requesthandler-Servlet packt die Eingaben der GUI in ein Containerobjekt.	Ein Containerobjekt beinhaltet alle Eingaben und Antworten der Komponenten, durch die es durchgereicht wurde.	30.5.02	1	muss	x			erledigt	
G-050	GUI	selbst	Das Requesthandler-Servlet kann die Eingaben bereits auf	Korrekte Eingaben sind für das System notwendig.	30.5.02	3	kann	x			in Arbeit	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
			syntaktische Fehler kontrollieren.									
G-051	GUI	selbst	Der Request-Dispatcher unterscheidet zwischen Requests, die an die Sicherheit bzw. an die Business Logic weitergeleitet werden müssen.	Requests, die der Authentifizierung dienen werden nicht von der BL bedient und müssen an die Sicherheit weitergeleitet werden.	30.5.02	1	muss	x			erledigt	
G-052	S	S-015	Der Request-Dispatcher überprüft beim Session-Manager vor der Bearbeitung einer Anfrage und vor der Ausgabe des Ergebnisses, ob eine Erlaubnis für diese Aktionen besteht.	Um sicherzustellen, dass in dem "Zeitloch" zwischen Start des Requests und der Ausgabe die Session nicht beendet wurde und so die Ausgabe evtl. eine unauthorisierte Person zu sehen bekommt, wird vor der Ausgabe nochmals mit der Sicherheit gegengecheckt.	30.5.02	3	muss	x		x	erledigt	
G-053	GUI	selbst	Der Request-Dispatcher überprüft mit dem History-Controller, ob die gewünschte Anfrage zulässig ist.	Ein Rückschritt, z.B. nach der Bestätigung einer Zahlung zu den Einstellungen für diese Zahlung, darf nicht möglich sein. Diesen Fall fängt der History-Controller ab.	30.5.02	2	muss	x			erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
G-054	GUI	selbst	Gültige Anfragen leitet der Request-Dispatcher an den Workflow-Adapter weiter.	Der Workflow-Adapter bietet die Schnittstelle zur Business-Logic.	30.5.02	1	muss	x			erledigt	
G-055	DB	DB-007; DB-009	Der History-Controller speichert alle Aktionen des Benutzers in der DB.	Um die gültige Reihenfolge der Anfragen eines Benutzers sicherzustellen, müssen diese Anfragen gespeichert werden.	30.5.02	2	muss	x		x	entfällt	logging wurde in Abstimmung mit der Gesamt-PG weggelassen, hauptsächlich aus Performance-Gründen
G-056	GUI	selbst	Der History-Controller kontrolliert ob eine gewünschte Aktion auf die vorhergehende(n) folgen darf.	s. G-053	30.5.02	2	muss	x			erledigt	
G-057	GUI	selbst	Der Response-Presenter bereitet die Ergebnisse einer Anfrage/Aktion für die Darstellung auf.	Die Ergebnisse einer Aktion/Abfrage in einem Containerobjekt müssen für die Ausgabe aufbereitet werden.	30.5.02	1	muss	x			erledigt	
G-058	GUI	selbst	Der Workflow-Adapter leitet sämtliche Anfragen an den Workflow-Manager zur Bearbeitung weiter. Spezifizierung!!!	Der Workflow-Adapter ist ein Wrapper des Workflow-Managers der Business Logic. Durch ihn wird eine strikte architektonische Trennung der GUI und der BL möglich.	30.5.02	1	muss	x		x	erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
G-059	GUI	selbst	Die GUI stellt die vom DK benötigten Funktionen im Workflow-Adapter bereit. Gemäss der noch zu definierenden Schnittstelle werden Containerobjekte und Statusmeldungen übergeben.		11.6.02	1	muss	x		x	erledigt	
G-060	BL	Bl-010	Die BL bietet Funktionen um statistische Daten über die Rechungen abzufrufen. Diese Statistiken sind sowohl für Rechnungsempfänger als auch für Rechnungssteller für eigene Rechnungen verfügbar.	Statistiken bieten eine schnelle Möglichkeit sich einen Überblick über z.B. seine monatlichen ausgaben zu schaffen.	5.6.02	4	kann	x		x	entfällt	optionales Feature aus Zeitmangel nicht umgesetzt
G-061	GUI	selbst	Die GUI bietet dem Rechnungsempfänger Statistiken über seine Rechnungen an.	Eine mögliche Zusatzfunktion. (s. G-059)	11.6.02	4	kann	x			entfällt	siehe G-060
G-062	DB	DB-007; DB-009	Die internen Daten der Komponente GUI werden soweit wie möglich in der Datenbank	Die Speicherung in der Datenbank ermöglicht eine Zugriffssicherheit.	5.6.02	1	muss			x	erledigt bis auf XSL und	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
			gespeichert. (s. G-041)								Grafiken	
G-063	GUI		Dem Rechnungssteller wird eine GUI für den Zugriff auf Daten der Registry und Repository des DK zur Verügung gestellt.	Über diese Schnittstelle kann der Rechnungssteller selbst das Format steuern, in dem er Rechnungen erhält und an den Betreiber verschickt.	5.6.02	3	muss	x		x	entfällt	umgesetzt durch DK (?)
G-064	GUI		Der DK bietet der GUI Funktionen für das Abfragen darzustellender Informationen der Registry und des Repositiory.	Ermöglicht G-062.	11.6.02	3	muss	x		x	entfällt	
G-065	GUI		Es wird keine Verarbeitungslogik auf den Clients sowohl beim Rechnungsempfänger als auch beim Rechnungssteller eingesetzt.	Dadurch wird die Benutzung von Applets zur Verarbeitung von Rechnungsdaten ausgeschlossen. (s. S-002)	11.6.02	1	muss		x	x	erledigt	
G-066	GUI		Es werden jederzeit nur notwendige Daten übertragen.	Zur Sicherheit werden keine unnötigen Informationen übertragen. (s. S-003)	11.6.02	1	muss		x	x	erledigt	
G-067	S		Die Übertragung zwischen dem jeweiligen Benutzer	Sensible Daten müssen auch bei der Übertragung geschützt	11.6.02	1	muss		x	x	entfällt	durch Komponente Sicherheit nicht

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
			und der GUI verläuft spätestens ab der Authentifizierung verschlüsselt (SSL).	werden. (s. S-004)								unterstützt
G-068	GUI	selbst	Die GUI stellt Funktionen zum Versenden von Emails, SMS, etc. zur Verfügung. (Softwareverfügbarkeit David?)	Mit dieser Funktion kann die BL Informationen über den Eingang neuer Rechungen oder Manungen an den Rechnungsempfänger versenden.	11.6.02	3	kann	x		x	entfällt	BL hat eigene Funktion zum Mailversand
G-069	GUI		Es werden keine Daten auf dem Klienten gehalten. Ein eventueller Datenexport ist davon ausgenommen.	Daten könnten in falsche Hände geraten.	24.6.02	1	muss	x			erledigt	
G-070	GUI		Daten können nur von den hierzu berechtigten Benutzern eingesehen werden.	Daten könnten in falsche Hände geraten.	24.6.02	1	muss	x			in Arbeit	
G-071	GUI		Es werden nur Daten, die von der jeweiligen Komponente verwaltet werden, gespeichert.	Komponenten speichern keine Daten, welche sie von anderen Komponenten erhalten haben. Aktualität der Daten und Datensicherheit.	24.6.02	1	muss		x		erledigt	
G-072	S		Der Betreiber hat die Möglichkeit sich mittels	Ein Authentifizierung ist für den Zugriff auf geschützte Daten	28.4.02	1	muss	x		x	erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
			Benutzername und Passwort am System anzumelden.	unbedingt nötig.								
G-073	GUI	selbst	Der Betreiber kann eine Liste der Rechnungsempfänger einsehen	Zur Kontrolle der Daten im System	28.6.02	2	muss	x			erledigt	
G-074	GUI	selbst	Der Betreiber kann Details der Rechnungsempfänger kontrollieren	Zur Kontrolle der Daten im System	28.6.02	2	muss	x		x	erledigt	
G-075	GUI	selbst	Der Betreiber kann Rechnungsempfängerkonten löschen, editieren und anlegen.	Zur Wartung der Daten im System.	28.6.02	3	muss	x		x	erledigt	
G-076	GUI	selbst	Der Betreiber kann eine Liste der Rechnungssteller einsehen	Zur Kontrolle der Daten im System	28.6.02	2	muss	x			erledigt	
G-077	GUI	selbst	Der Betreiber kann Details der Rechnungssteller kontrollieren	Zur Kontrolle der Daten im System	28.6.02	2	muss	x		x	erledigt	
G-078	GUI	selbst	Der Betreiber kann Rechnungsstellerkonten löschen, editieren und anlegen.	Zur Wartung der Daten im System.	28.6.02	3	muss	x		x	erledigt	
G-079	GUI	selbst	Der Betreiber kann eine Liste der Finanzdienstleister einsehen	Zur Kontrolle der Daten im System	28.6.02	4	muss	x			entfällt	kein Workflow

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
G-080	GUI	selbst	Der Betreiber kann Details der Finanzdienstleister kontrollieren	Zur Kontrolle der Daten im System	28.6.02	4	muss	x		x	entfällt	kein Workflow
G-081	GUI	selbst	Der Betreiber kann Finanzdiensteleister konten löschen, editieren und anlegen.	Zur Wartung der Daten im System.	28.6.02	4	muss	x		x	entfällt	kein Workflow; aktuell nur Banken unterstützt; die Daten der Banken werden importiert
G-082	GUI	selbst	Der Betreiber kann eine Liste der Rechnungen einsehen	Zur Kontrolle der Daten im System	28.6.02	2	muss	x			erledigt	
G-083	GUI	selbst	Der Betreiber kann Details der Rechnungen kontrollieren	Zur Kontrolle der Daten im System	28.6.02	2	muss	x		x	erledigt	
G-084	GUI	selbst	Der Betreiber kann Rechungen löschen.	Zur Wartung der Daten im System.	28.6.02	3	muss	x		x	erledigt	
G-085	GUI	selbst	Einhalten der Sicherheitsrichtlinien	siehe S-021	1.7.02	1	muss	x		x	in Arbeit	
G-086	GUI	selbst	Der Betreiber kann die Benutzer (Rechungsempfänger, Rechnungssteller, Finanzdienstleister) Gruppen zuordnen.	Die Sicherheitsgruppen werden für die Einordnung der Rechte der Benutzer benötigt.	5.7.02	2	muss	x		x	entfällt	bislang kein Workflow
G-087	GUI	selbst	Der Betreiber kann Sicherheitsgruppen anlegen, löschen und editieren.	Bearbeiten der Sicherheitsgruppen.	5.7.02	2	muss	x		x	entfällt,	die Zuordnung der Workflows zu Gruppen ist statisch

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
G-088	GUI	selbst	Der Betreiber kann über die GUI Systemjobs anlegen und löschen.	?	5.7.02	2	muss	x		x	erledigt	
G-089	GUI	selbst	Der Betreiber kann über die GUI Systemjobs zur Ausführung anstossen.	?	5.7.02	2	muss	x		x	entfällt,	von BL nicht vorgesehen
G-090	GUI	selbst	Der Betreiber stellt ein Interface für das hinzufügen und löschen von CPP bzw. CPAs	Zur Wartung des Datenkonverters.	5.7.02	2	muss	x		x	entfällt	bislang kein Workflow definiert
G-091	GUI	selbst	Die Kunden-ID des Rechnungsempfängers muss vom Rechnungssteller vor der Inanspruchnahme von Com42Bill als Zahlungsmethode überprüft werden, Dies soll realisiert werden durch eine Umleitung auf die Com42Bill-Login-Seite, wo der Rechnungsempfänger sich anmelden kann. Danach erfolgt der Rücksprung zur Rechnungssteller-Seite mit	Sicherheit.	5.7.02	2	kann	x		x	entfällt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
			entsprechender Rückmeldung.									

A.4 Business Logic (BL)

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
BL-001	BL	selbst	Abbilden aller im System anfallenden Geschäftsprozesse (sowohl für die Durchführung von GUI- und ImpEx-Requests, wie auch administrative Tätigkeiten des Betreibers)	Hauptaufgabe der Komponente BL	4.6.02	1	muss		x		erledigt	
BL-002	BL	selbst	Workflowmanager startet und steuert die Workflows, die einem vorher von BL festgelegten Ablauf folgen.	Die Ausführung der Geschäftsprozesse muss verwaltet werden	30.5.02	1	muss	x		x	erledigt	
BL-003	BL	selbst	Die Abläufe einzelner Geschäftprozesse, welche von Pipelines realisiert werden,	Die Geschäftsprozesse können mit Pipelines sehr anschaulich und	30.5.02	1	muss	x			erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
			werden in Form von XML-Files festgehalten.	flexibel modelliert werden. Die Verfügbarkeit von Tools ist zu klären								
BL-004	BL	selbst	Realisierung einer GUI für das Modellieren von Pipelines	Je nachdem, wie oft sich die Geschäftsprozesse um Laufe der Projektgruppe noch ändern und wie erweiterbar das System im Anschluss sein soll, ist das Handling der reinen XML-Dateien nach Möglichkeit abzulösen	4.6.02	3	kann	x			entfällt	Diese Funktionaltät war optional angedacht, um die Usabilty zu erleichert.
BL-005	BL	selbst	Workflowmanager arbeitet auf Businessobjects die vom Manager oder vom EJB-Container zur Verfügung gestellt werden (hängt vom evtl. Einsatz von SessionBeans ab)	Die BusinessObjects als Teile der Pipelines führen die eigentliche BusinessLogic aus	30.5.02	1	muss	x			erledigt	
BL-006	BL	selbst	Businesspartnerman ager verwaltet alle Daten die für einen Businesspartner notwendig ist. Der Manager kann Kundenkonten anlegen, erfragen,	Die Rollen müssen im System repräsentiert werden	30.5.02	2	muss	x		x	erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
			löschen, bearbeiten usw.									
BL-007	BL	selbst	Invoicemanager ist für die Rechnungsdatenverwaltung zuständig. Er bietet Funktionen zur Speicherung von Rechnungen und deren Zuordnung zu Businesspartnern. Da man Rechnungen nicht löschen können sollte, muss es möglich sein, sie zu stornieren. Ebenso muss	Logisch	30.5.02	2	muss	x		x	erledigt	
BL-008	BL	selbst	PaymentManager ist für das Bezahlungsmanagement der Rechnungen verantwortlich. Hiermit werden die Finanztransaktionen vorbereitet und verarbeitet	Ebenfalls essentiell für ein EBPP-System	30.5.02	2	muss	x		x	erledigt	
BL-009	BL	selbst	Alertmanager ist für alle anfallenden Benachrichtigungen im Rahmen der Rechnungsabhandlung zuständig	Benachrichtigungsdienste sind zwar unerlässlich, stellen jedoch nicht die Basisfunktionalität des Systems sicher	4.6.02	3	muss	x		x	erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
BL-010	BL	selbst	Reportmanger ist für das Erzeugen von Berichten, sowie zum Abrufen von Statistiken zuständig	Sehr nützliche Funktionalität sowohl für Rechnungssteller als auch -empfänger. Jedoch keine Basisfunktionalität	30.5.02	4	muss	x		x	entfällt	Diese ist ein illusionarische Anforderung. Es hatte auf "kann" stehen müssen.
BL-011	BL	selbst	Personalisationmanager ist für die an die Kundenwünsche angepasste Umgebung zuständig	Value-Added-Service	30.5.02	4	kann	x		x	entfällt	Diese Funktionalität wird nirgends gefordert. Aber die Entities sind voehanden.
BL-012	BL	selbst	Transaktionsmanagment sollte vom EJB-Container unterstützt werden. Gegebenenfalls müssen die Funktionalitäten erweitert werden.	Geschäftsprozesse müssen als Einheit ausgeführt werden können, um die Konsistenz der Daten zu wahren	30.5.02	1	muss	x			erledigt	
BL-013	GUI	G-058	Überreichen eines RequestDictionary, eines GUI-Requests, aus dem der zu startende Workflow ersichtlich ist und evtl. einer gültigen Session. Die Abarbeitung des Workflows wird vom Workflow-Adapter initiiert.	BL ist nur eine passive Komponente, d.h. wird immer von anderen Komponenten angestoßen. Wenn dies geschieht, muss aus dem Aufruf hervorgehen, welcher Geschäftsprozess unter welchen Umständen zu starten ist	30.5.02	1	muss	x		x	erledigt	
BL-014	GUI	G-050	Überprüfen der Formulardaten, welche mit dem	Die BL führt nur die inhaltliche Korrektheitsüberprüfu	4.6.02	2	muss		x		offen	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
			RequestDictionary übergeben werden auf Typkorrektheit	ng der Daten durch								
BL-015	GUI	G-068	Bereitstellen von Template-Mechanismen, welche es ermöglichen, aus einem Geschäftsprozess Benachrichtungen via Mail, SMS etc. zu verschicken.	Das Erscheinungsbild der zu verschickenden Benachrichtigungen sollte schnell anpassbar sein und von den Inhalten, welche im Geschäftsprozess ermittelt werden, abstrahiert werden	4.6.02	3	muss	x		x	entfällt	Siehe GUI
BL-016	GUI	G-058,G-059	Überreichen eines RequestDictionary, eines ImpEx-Requests, aus dem der zu startende Workflow ersichtlich ist, und ebenfalls einer Session. Die Abarbeitung des Workflows wird vom Workflow-Adapter initiiert.	Hier gilt das selbe wie unter BL-013, jedoch für Requests, welche vom Datenkonverter stammen	30.5.02	1	muss	x		x	erledigt	
BL-017	DB	DB-012	Datenbankkomponente stellt die EJB-Entity-Objekte für die Speicherung der in den Workflows anfallenden Daten bereit. Ebenso müssen gewünschte Daten in Form von	Die Datenbankkomponente ist der einzige Datenbehälter für fachliche Daten	30.5.02	1	muss	x		x	erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
			EJB-Objekten abgerufen werden können									
BL-018	GUI	G-054	Das Durchführen eines Workflows über den Workflow-Adapter muss bereits autorisiert sein.	Die Komponente BL führt selber keine Autorisierung / Authentifizierung durch. Daher muss diese Funktionalität von der Komponente GUI vorher sichergestellt werden	30.5.02	1	muss		x		erledigt	
BL-019	DK		Das Durchführen eines Workflows über den Workflow-Adapter muss bereits autorisiert sein.	Die Komponente BL führt selber keine Autorisierung / Authentifizierung durch. Daher muss diese Funktionalität von der Komponente Datenkonverter vorher sichergestellt werden	30.5.02	1	muss		x		erledigt	
BL-020	BL	selbst	Zurückgegeben eines Ergebnisses nach einem Workflowdurchlauf, weitere Statusmeldungen werden im RequestDictaionary abgelegt	GUI und DK müssen Antwort auf Request erhalten	25.6.02	1	muss	x		x	erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
BL-021	BL	selbst	Bereitstellen eines Communication Service, mit dessen Hilfe Mails und andere Nachrichten aus einem Workflow heraus verschickt werden können	Notwendig für Benachrichtigungsdienste	25.6.02	2	muss	x			erledigt	
BL-022	BL	selbst	Bereitstellen einer Jobverwaltung, welche zeitgesteuert oder auch manuell ausgelöste Workflows starten kann	Notwenig, um z.B. Finanztransaktionen zeitgesteuert durchführen zu können, oder Benachrichtigungen zu verschicken	25.6.02	2	muss	x			erledigt	
BL-023	BL		So wenig Daten wie möglich übertragen	siehe S-003	25.6.02	1	muss		x		entfällt	Siehe neue Sicherheits-anforderungen
BL-024	BL		Verantwortlichkeit für übertragene Daten	siehe S-005	25.6.02	1	muss		x		entfällt	Siehe neue Sicherheits-anforderungen
BL-025	BL		Nur eigene Daten speichern	siehe S-007	25.6.02	1	muss		x		entfällt	Siehe neue Sicherheits-anforderungen
BL-026	BL	selbst	Evtl. Bereistellen eines Query-Frameworks, mit dessen Hilfe dynamisch konfigurierte Suchabfragen durchgeführt werden können	Hilft bei Suchfunktionen und Auswertungen, jedoch hoher Implementierungsaufwand	25.6.02	3	kann	x			entfällt	Diese Komponete ist die Verbesserung der requestgeschwingkeit angedacht gewesen. Eine Searchfunktion ist auf Programmierebne implementiert.
BL-	BL	selbst	Einhalten der	siehe S-021	30.6.02	1	muss		x		erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
027			Sicherheitsrichtlinien									
BL-028	GUI	G-088, G-089	Die GUI muss eine Benutzeroberfläche bereitstellen, mit deren Hilfe man Systemjobs anlegen, löschen und anstossen kann.	Administration	10.7.02	2	muss	x			erledigt	

A.5 Sicherheit (S)

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
S-001	GUI		Es werden keine Daten auf dem Klienten gehalten. Ein eventueller Datenexport ist davon ausgenommen.	Daten könnten in falsche Hände geraten.	30.5.02	1	muss		x		erledigt	Auslagerung in die Sicherheitsrichlinien.
S-002	GUI, BL		Die Daten werden in verarbeiteter, d.h. endgültiger Form, übermittelt.	Klienten müssen keine Verarbeitungslogik besitzen.	30.5.02	1	muss		x		erledigt	Auslagerung in die Sicherheitsrichlinien.
S-003	GUI, DK, DB, BL		Es werden so wenig Daten wie möglich übertragen, d.h. die zu übertragenden Daten werden auf das nötigste beschränkt.	Performance und Datensicherheit.	30.5.02	1	muss		x		erledigt	Auslagerung in die Sicherheitsrichlinien.
S-004	GUI, DK		Ist eine Datenübertragung außerhalb des Betreibers nötig, wird nur über sichere	Datensicherheit.	30.5.02	1	muss		x		entfällt, von S nicht unterstützt	Verschlüsselung wird wegen nicht vorhandener PGP

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
			Verbindungen kommuniziert.									Infrastruktur nicht umgesetzt.
S-005	GUI, DK, DB, BL		Jede Komponente ist für die übertragenen Daten verantwortlich.	Jede Komponente muss wissen ob der Empfänger die Daten, welche er anfordert, wirklich bekommen darf.	30.5.02	1	muss		x		entfällt	Auslagerung in die Sicherheitsrichlinien.
S-006	GUI, DK, BL		Daten werden ausschließlich in der Datenbank gespeichert.	Datensicherheit und Datenintegrität.	30.5.02	1	muss		x		entfällt	Auslagerung in die Sicherheitsrichlinien.
S-007	GUI, DK, BL		Es werden nur Daten, die von der jeweiligen Komponente verwaltet werden, gespeichert.	Komponenten speichern keine Daten, welche sie von anderen Komponenten erhalten haben. Aktualität der Daten und Datensicherheit.	30.5.02	1	muss		x		entfällt	Auslagerung in die Sicherheitsrichlinien.
S-008	S		Benutzer des Systems müssen sich über die Komponente Sicherheit anmelden.	Anmeldung.	30.5.02	1	muss		x	x	entfällt	Auslagerung in die Sicherheitsrichlinien.
S-009	DK	DK-001	Die Komponente Datenkonverter muss, zur Anmeldung eines Benutzers an das System, der Komponente Sicherheit den Benutzernamen, das Passwort und die benutzte Schnittstelle übermitteln.	Anmeldung eines Benutzers der Komponente Datenkonverter.	30.5.02	1	muss	x		x	erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
S-010	DK	DK-004	Die Komponente Datenkonverter übergibt der Komponente Sicherheit, zur Überprüfung einer auszuführenden Aktion, die Session-ID und die gewünschte Aktion.	Authorisierung einer Aktion der Komponente Datenkonverter.	30.5.02	1	muss	x		x	erledigt	
S-011	S	selbst	Nachdem überprüft wurde ob die gewünschte Aktion zulässig ist, erteilt die Komponente Sicherheit der Komponente Datenkonverter die Erlaubnis bzw. ein Verbot.	Authorisierung einer Aktion der Komponente Datenkonverter.	30.5.02	1	muss	x		x	erledigt	
S-012	S	selbst	Die Komponente Sicherheit übergibt, nach erfolgter Anmeldung, der Komponente Datenkonverter eine Session-ID und den Benutzernamen, welcher zu dieser Session gehört.	Anmeldung eines Benutzers der Komponente Datenkonverter.	30.5.02	1	muss	x		x	erledigt	
S-013	GUI	G-052	Bei einer "anonymen Anfrage" übergibt die Komponente GUI der Komponente Sicherheit die	Authorisierung einer Aktion der Komponente GUI.	30.5.02	1	muss	x		x	erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
			gewünschte Aktion.									
S-014	GUI	G-052	Die Komponente GUI übergibt der Komponente Sicherheit, zur Überprüfung einer auszuführenden Aktion, die Session-ID und die gewünschte Aktion.	Authorisierung einer Aktion der Komponente GUI.	30.5.02	1	muss	x		x	erledigt	
S-015	S	selbst	Nachdem überprüft wurde ob die gewünschte Aktion zulässig ist, erteilt die Komponente Sicherheit der Komponente GUI die Erlaubnis bzw. ein Verbot.	Authorisierung einer Aktion der Komponente GUI.	30.5.02	1	muss	x		x	erledigt	
S-016	GUI	G-044	Die Komponente GUI muss, zur Anmeldung eines Benutzers an das System, der Komponente Sicherheit den Benutzernamen, das Passwort und die benutzte Schnittstelle übermitteln.	Anmeldung eines Benutzers der Komponente GUI.	30.5.02	1	muss	x		x	erledigt	
S-017	S	selbst	Die Komponente Sicherheit übergibt, nach erfolgter Anmeldung, der	Anmeldung eines Benutzers der Komponente GUI.	30.5.02	1	muss	x		x	erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
			Komponente GUI eine Session-ID und den Benutzernamen, welcher zu dieser Session gehört.									
S-018	DB	DB-007	Die Komponente Datenbank muss der Komponente Sicherheit Methoden zur Verfügung stellen, mit deren Hilfe die Komponente Sicherheit Daten ablegen kann.	Speicherung der Komponentendaten.	30.5.02	1	muss	x		x	erledigt	
S-019	S	selbst	Die Komponente Sicherheit muss Methoden zur Verschlüsselung bereitstellen.	Datensicherheit.	14.6.02	1	muss	x			erledigt	
S-020	S	selbst	Die Komponente Sicherheit überprüft anhand der digitalen Signatur, obe die Daten, welche dem Datenkonverter übertragen wurden unverändert sind. Es wird ebenfalls die Identität des Absender überprüft.	Datensicherheit.	22.6.02	1	muss	x		x	entfällt	Verschlüsselung wird wegen nicht vorhandener PGP Infrastruktur nich umgesetzt.
S-021	Alle	DK-032; DB-024; BL-027;G-085	Die Komponenten müssen sich an die Sicherheitsrichtlinien halten.	Sicherheit.	22.6.02	1	muss		x		entfällt	Auslagerung in die Sicherheitsrichlinien.

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
S-022	S	selbst	Die Komponente Sicherheit muss eine Public-Key Infrastruktur bereitstellen.	Datensicherheit.	27.6.02	1	muss	x		x	entfällt	Verschlüsselung wird wegen nicht vorhandener PGP Infrastruktur nich umgesetzt.
S-023	S	selbst	Die Komponente Sicherheit muss Methoden zur Signierung von Nachrichten bereitstellen.	Datensicherheit.	27.6.02	1	muss	x		x	entfällt	Verschlüsselung wird wegen nicht vorhandener PGP Infrastruktur nich umgesetzt.
S-024	S	selbst	Die Komponente Sicherheit weißt den Benutzer auf ein veraltetes Kennwort hin.	Sicherheit.	27.6.02	1	muss	x			entfällt	Feature wurde nicht mehr gewünscht.
S-025	S	selbst	Die Komponente Sicherheit stelle Methoden zur Verfügung, um Benutzer anzulegen, zu bearbeiten und zu löschen.	Benutzeradministration.	10.7.02	1	muss	x		x	in Arbeit	Durch DB.
S-026	S	selbst	Die Komponente Sicherheit stelle Methoden zur Verfügung, um ACL-Groups anzulegen, zu bearbeiten und zu löschen.	Benutzeradministration.	10.7.02	1	muss	x		x	in Arbeit	Durch DB.
S-027	S	selbst	Die Komponente Sicherheit stelle Methoden zur Verfügung, um	Benutzeradministration.	10.7.02	1	muss	x		x	in Arbeit	Wird nich benötigt.

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ		NF	SR	Status	Begründung für Status entfällt.
			Request-Types anzulegen, zu bearbeiten und zu löschen.									
S-028	GUI	G-075, G-078, G-081	Die Komponente GUI stellt eine Benutzeroberfläche zur Verfügung mit deren Hilfe man Benutzer anlegen, bearbeiten und löschen kann.	Benutzeradministration.	10.7.02	1	muss	x		x	in Arbeit	GUI nicht vorgesehen.
S-029	GUI	G-086, G-087	Die Komponente GUI stellt eine Benutzeroberfläche zur Verfügung mit deren Hilfe man ACL-Groups anlegen, bearbeiten und löschen kann.	Benutzeradministration.	10.7.02	1	muss	x		x	in Arbeit	GUI nicht vorgesehen.
S-030	GUI	nicht abgedeckt ??	Die Komponente GUI stellt eine Benutzeroberfläche zur Verfügung mit deren Hilfe man Request-Types anlegen, bearbeiten und löschen kann.	Benutzeradministration.	10.7.02	1	muss	x		x	in Arbeit	GUI nicht vorgesehen.

A.6 Datenbank (DB)

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ	F	NF	SR	Status	Begründung für Status entfällt.
DB-001	DK		Es muß eine Datenstruktur für das ebXML-Repository		30.5.02	2	muss		x	x	entfällt	Wurde von DK übernommen

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ	F	NF	SR	Status	Begründung für Status entfällt.
			definiert werden									
DB-002	DK		Die Methoden für den Zugriff auf das ebXML-Repository müssen definiert werden		30.5.02	2	muss		x	x	entfällt	Wurde von DK übernommen
DB-003	DB	selbst	Implementierung eines DB-User-basierten Zugriffsschutzes	Verhinderung von unberechtigten Zugriffen auf die gespeicherten Daten	31.5.02	1	muss	x			entfällt	von S übernommen
DB-004	DB		Konfiguration des Transaktions-Managements	Sicherstellung, dass Zugriffsvorgänge auf die DB vollständig durchgeführt werden	1.6.02	2	muss	x			erledigt	
DB-005	DB	selbst	Beschleunigung des Zugriffs auf Datenhaltung	Möglichst schneller Zugriff auf gespeicherte Daten	2.6.02	2	kann	x			entfällt	
DB-006	DB	selbst	Konfiguration der Serialisierung	Erfolgt automatisch durch AppServer	3.6.02	2	muss	x			erledigt	
DB-007	DB	selbst	Implementierung der Zugriffsfunktionalitäten Anlegen, Löschen, Ändern, Lesen, Suchen nach Vorgabe der entsprechenden Komponenten		4.6.02	2	muss	x		x	in Arbeit	
DB-008	GUI	selbst	Festlegung der Entitäten, die global für andere Komponenten verfügbar sein sollen		5.6.02	2	muss		x	x	erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ	F	NF	SR	Status	Begründung für Status entfällt.
DB-009	BL	selbst	Festlegung der Entitäten, die global für andere Komponenten verfügbar sein sollen		5.6.02	2	muss		x	x	erledigt	
DB-010	DK	selbst	Festlegung der Entitäten, die global für andere Komponenten verfügbar sein sollen		5.6.02	2	muss		x	x	erledigt	
DB-011	S	selbst	Festlegung der Entitäten, die global für andere Komponenten verfügbar sein sollen		5.6.02	2	muss		x	x	erledigt	
DB-012	DB	selbst	Alle Entitäten (auch vererbte) des Objektmodells sollen als Tabellen abgebildet werden, ihre Attribute werden als Spalten der entsprechenden Tabellen realsiert		6.6.02	2	muss	x			erledigt	
DB-013	BL	selbst	Festlegen der Methoden für Mgr-Subkomponenten	Für die Implementierung von HomeInterfaces wichtig	12.6.02	2	muss	x		x	erledigt	
DB-014	DB	selbst	Bereitstellen von Methoden für Aufbau der DB-Verbindungen (ConnectionPool)	Erfolgt automatisch durch AppServer	12.6.02	2	muss	x		x	erledigt	
DB-015	DB	selbst	Bereitstellen von get/set Methoden für Entity-Beans	Für den Zugriff auf die Attribute der Entity-Beans wichtig	12.6.02	1	muss	x		x	erledigt	
DB-016	DB	selbst	Ermöglichung von Container Management	Entlastung der BL	12.6.02	1	muss	x			erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ	F	NF	SR	Status	Begründung für Status entfällt.
			Persistenz									
DB-017	DB	selbst	Daten, die systemweit gültig sind (Entitäten des Package CORE im Objektmodell), werden in einem DB-Bereich gespeichert, der allen autorisierten Komponenten zugänglich ist	Trennung von Globalen und Komponenten-eigenen Daten zwecks Datenintegrität und Verteilbarkeit	18.6.02	1	muss		x		erledigt	
DB-018	DK	selbst	Alle Daten einer Komponente werden in einem exclusiven Bereich der DB gespeichert	Sicherung der Datenkonsistenz, Datenintegrität, leichte Austauschbarkeit der Daten	18.6.02	1	muss		x		erledigt	
DB-019	S	selbst	Alle Daten einer Komponente werden in einem exclusiven Bereich der DB gespeichert	Sicherung der Datenkonsistenz, Datenintegrität, leichte Austauschbarkeit der Daten	18.6.02	1	muss		x		erledigt	
DB-020	BL	selbst	Alle Daten einer Komponente werden in einem exclusiven Bereich der DB gespeichert	Sicherung der Datenkonsistenz, Datenintegrität, leichte Austauschbarkeit der Daten	18.6.02	1	muss		x		erledigt	
DB-021	GUI	selbst	Alle Daten einer Komponente werden in einem exclusiven Bereich der DB gespeichert	Sicherung der Datenkonsistenz, Datenintegrität, leichte Austauschbarkeit der Daten	18.6.02	1	muss		x		erledigt	

ID	Betroffene Komp.	Abgedeckt durch	Beschreibung	Begründung	Datum	Prio	Typ	F	NF	SR	Status	Begründung für Status entfällt.
DB-022	DB	selbst	Alle Daten einer Komponente werden in einem exclusiven Bereich der DB gespeichert	Sicherung der Datenkonsistenz, Verteilbarkeit der DB	18.6.02	1	muss		x		erledigt	
DB-023	DB	selbst	Der Zugriff auf die Daten einer Komponente erfolgt exclusiv und ausschliesslich durch sie selbst	Sicherung der Datenkonsistenz	18.6.02	1	muss		x		erledigt	
DB-024	DB	selbst	Die Sicherheitsrichtlinien müssen eingehalten werden.	S-021	27.6.02	1	muss		x		erledigt	

Anhang B Projektplan

B.1 Projektplan (tabellarisch)

Im Folgenden wird der Projektplan in tabellarischer Form dargestellt.

B.1.1: Kennenlernfahrt & Seminarphase

Task-ID	Task Name	Dauer	Start	Ende
	Kick-Off Nordhelle	**3 Tage**	**10.04.2002**	**12.04.2002**

B.1.2: Datenkonverter

Task-ID	Task Name	Dauer	Start	Ende
DK-01	Anforderungsanalyse	41 Tage	16.05.2002	11.07.2002
	Anforderungsdokument	1 Tage	15.07.2002	15.07.2002
DK-02	Systemarchitektur	119 Tage	06.05.2002	16.10.2002
	Dokument Systemarchitektur	1 Tage	16.10.2002	16.10.2002
DK-03	Klassenmodell	67 Tage	16.07.2002	15.10.2002
	UML-Diagramm Klassenmodell	1 Tage	15.10.2002	15.10.2002
DK-04	Objektmodell	51 Tage	28.06.2002	06.09.2002
	UML-Diagramm Objektmodell	1 Tage	06.09.2002	06.09.2002
DK-05	Objektdesign	79 Tage	16.07.2002	31.10.2002
	Designdokument	1 Tage	15.10.2002	15.10.2002
DK-06	Prototyp (Key Features)	45 Tage	16.07.2002	13.09.2002
DK-06.1	CPP für Com42Bill	10 Tage	15.07.2002	26.07.2002
DK-06.2	CPAs definieren	5 Tage	29.07.2002	02.08.2002
DK-06.3	Austauschformat/Dokumentstruktur	5 Tage	05.08.2002	09.08.2002
DK-06.4	MsgServiceInterfaceHandler	10 Tage	12.08.2002	23.08.2002
DK-06.5	MsgServiceInterface	25 Tage	23.08.2002	25.09.2002
	Lauffähiger Prototyp	1 Tage	01.10.2002	01.10.2002
DK-07	Implementierung	70 Tage	07.10.2002	10.01.2003
DK-07.1	Spezifikation des Request-Dictionarys	70 Tage	07.10.2002	10.01.2003
DK-07.2	Verfeinerung der Architektur	65 Tage	14.10.2002	10.01.2003
DK-07.3	Klassendiagramm (grob)	3 Tage	08.01.2003	10.01.2003
DK-07.4	Erstellung und Dokumentation des CPP & CPA	60 Tage	21.10.2002	10.01.2003
DK-07.5	BusinessServiceInterface	4 Tage	07.01.2003	10.01.2003
DK-07.6	MsgServiceHandler	5 Tage	06.01.2003	10.01.2003
DK-07.7	CPAParser	2 Tage	09.01.2003	10.01.2003
DK-07.8	ImpExpMgr	45 Tage	11.11.2002	10.01.2003
DK-07.9	Darstellung der Rechnung und Überweisung im Browser	10 Tage	13.01.2003	24.01.2003
	Lauffähige Komponente	27 Tage	10.01.2003	14.02.2003

B.1.3: GUI

Task-ID	Task Name	Dauer	Start	Ende
GUI-01	Anforderungsanalyse	40 Tage	17.05.2002	11.07.2002
	Anforderungsdokument	1 Tage	11.07.2002	11.07.2002
GUI-02	Systemarchitektur	46 Tage	17.05.2002	19.07.2002

	Dokument Systemarchitektur	1 Tage	21.06.2002	21.06.2002
GUI-03	Klassenmodell	84 Tage	16.07.2002	07.11.2002
	UML-Diagramm Klassenmodell	1 Tage	07.11.2002	07.11.2002
GUI-04	Objektmodell	98 Tage	17.05.2002	30.09.2002
	UML-Diagramm Objektmodell	1 Tage	30.09.2002	30.09.2002
GUI-05	Objektdesign	81 Tage	16.07.2002	04.11.2002
	Designdokument	1 Tage	04.11.2002	04.11.2002
GUI-06	Navigationsstruktur	12 Tage	23.10.2002	07.11.2002
GUI-06.1	HTML	12 Tage	23.10.2002	07.11.2002
GUI-06.2	WAP	12 Tage	23.10.2002	07.11.2002
	Dokumentation	1 Tage	07.11.2002	07.11.2002
GUI-07	Prototyp (Key-Features)	55 Tage	10.07.2002	23.09.2002
GUI-07.1	Syntaxkontrolle	55 Tage	10.07.2002	23.09.2002
GUI-07.2	Datenbank-Templates	55 Tage	10.07.2002	23.09.2002
GUI-07.3	Presenter	55 Tage	10.07.2002	23.09.2002
GUI-07.4	Workflowtest	55 Tage	10.07.2002	23.09.2002
GUI-07.5	History	55 Tage	10.07.2002	23.09.2002
	Lauffähiger Prototyp	1 Tage	23.09.2002	23.09.2002
GUI-08	Implementierung	96 Tage	01.10.2002	10.02.2003
GUI-08.1	RequestServlet	96 Tage	01.10.2002	10.02.2003
GUI-08.2	RequestDispatcher	96 Tage	01.10.2002	10.02.2003
GUI-08.3	HistoryController	96 Tage	01.10.2002	10.02.2003
GUI-08.4	RequestPresenter	96 Tage	01.10.2002	10.02.2003
GUI-08.5	WorkflowAdapter	96 Tage	01.10.2002	10.02.2003
GUI-08.6	HTML/WAP	96 Tage	01.10.2002	10.02.2003
	Lauffähige Komponente	1 Tage	28.02.2003	28.02.2003

B.1.4: Sicherheit

Task-ID	Task Name	Dauer	Start	Ende
S-01	Anforderungsanalyse	55 Tage	26.04.2002	11.07.2002
	Anforderungsdokument	1 Tage	15.07.2002	15.07.2002
S-02	Systemarchitektur	55 Tage	06.05.2002	19.07.2002
	Dokument Systemarchitektur	1 Tage	19.07.2002	19.07.2002
S-03	Klassenmodell	67 Tage	16.07.2002	15.10.2002
	UML-Diagramm Klassenmodell	1 Tage	15.10.2002	15.10.2002
S-04	Objektmodell	45 Tage	08.07.2002	06.09.2002
	UML-Diagramm Objektmodell	1 Tage	06.09.2002	06.09.2002
S-05	Objektdesign	67 Tage	16.07.2002	15.10.2002
	Designdokument	1 Tage	15.10.2002	15.10.2002
S-06	Prototyp (Key-Features)	51 Tage	09.07.2002	16.09.2002
S-06.1	Autorisierung GUI	25 Tage	15.07.2002	16.08.2002
S-06.2	Authentifizierung GUI	6 Tage	19.08.2002	26.08.2002
S-06.3	Autorisierung DK	3 Tage	27.08.2002	29.08.2002
S-06.4	Authentifizierung DK	4 Tage	30.08.2002	04.09.2002
S-06.5	Administration	9 Tage	05.09.2002	16.09.2002
S-06.6	Kennwort	1 Tage	16.09.2002	16.09.2002
S-07	Implementierung	69 Tage	17.09.2002	20.12.2002

S-07.1	Session Manager	29 Tage	17.09.2002	25.10.2002
S-07.2	Request Authorization	30 Tage	28.10.2002	06.12.2002
S-07.2.1	Authentification	25 Tage	28.10.2002	29.11.2002
S-07.2.2	Authorization	25 Tage	04.11.2002	06.12.2002
S-07.3	Administration	30 Tage	11.11.2002	20.12.2002
	Lauffähige Komponente	1 Tage	20.12.2002	20.12.2002

B.1.5: Business Logic

Task-ID	Task Name	Dauer	Start	Ende
BL-01	Anforderungsanalyse	60 Tage	15.04.2002	05.07.2002
	Anforderungsdokument	1 Tage	05.07.2002	05.07.2002
BL-02	Systemarchitektur	29 Tage	04.06.2002	12.07.2002
	Dokument Systemarchitektur	1 Tage	12.07.2002	12.07.2002
BL-03	Klassenmodell	67 Tage	16.07.2002	15.10.2002
	UML-Diagramm Klassenmodell	1 Tage	15.10.2002	15.10.2002
BL-04	Objektmodell	80 Tage	20.05.2002	06.09.2002
	UML-Diagramm Objektmodell	1 Tage	06.09.2002	06.09.2002
BL-05	Objektdesign	67 Tage	16.07.2002	15.10.2002
	Designdokument	1 Tage	15.10.2002	15.10.2002
BL-06	Prototyp (Key-Features)	56 Tage	15.07.2002	27.09.2002
BL-06.1	Rechnungsempfängerverwaltung	56 Tage	15.07.2002	27.09.2002
BL-06.2	BusinessObjects für Beispielworkflow	36 Tage	05.08.2002	20.09.2002
BL-06.3	Modellierung eines Beispielworkflows	36 Tage	12.08.2002	27.09.2002
BL-06.4	Workflowverwaltung	25 Tage	27.08.2002	27.09.2002
BL-06.5	Ergbnisrückgabe an GUI (Interaktion)	15 Tage	09.09.2002	27.09.2002
BL-07	Implementierung	71 Tage	14.10.2002	20.01.2003
BL-07.1	BPML Engine	48 Tage	14.10.2002	18.12.2002
BL-07.2	AlertMgr	46 Tage	14.10.2002	16.12.2002
BL-07.3	PamentMgr	46 Tage	14.10.2002	16.12.2002
BL-07.4	Business Objects	46 Tage	18.11.2002	20.01.2003
BL-07.5	Workflow modellieren	46 Tage	18.11.2002	20.01.2003
BL-07.5.1	Invoice BO	46 Tage	29.11.2002	31.01.2003
BL-07.5.2	BusinessPartner BO	46 Tage	29.11.2002	31.01.2003
BL-07.5.3	Payment BO	46 Tage	09.12.2002	07.02.2003
BL-07.5.4	Alert BO	46 Tage	09.12.2002	07.02.2003
BL-07.5.5	Core BO	46 Tage	09.12.2002	07.02.2003
BL-07.6	Businesspartner Manager	46 Tage	14.10.2002	16.12.2002
BL-07.7	Invoice Manager	46 Tage	14.10.2002	16.12.2002
BL-07.8	Job Manager	65 Tage	18.11.2002	13.02.2003
	Lauffähige Komponente	1 Tage	28.02.2003	28.02.2003

B.1.6: Datenbank

Task-ID	Task Name	Dauer	Start	Ende
DB-01	Anforderungsanalyse	64 Tage	15.04.2002	11.07.2002
	Anforderungsdokument	1 Tage	11.07.2002	11.07.2002

DB-02	Objektmodell Com42Bill	95 Tage	29.04.2002	06.09.2002
	UML-Diagramm Objektmodell	1 Tage	06.09.2002	06.09.2002
DB-03	Klassenmodell DataBaseAccess.	67 Tage	16.07.2002	15.10.2002
	UML-Diagramm Klassenmodell	1 Tage	15.10.2002	15.10.2002
DB-04	Objektdesign DataBaseAccess	67 Tage	16.07.2002	15.10.2002
	Designdokument	1 Tage	15.10.2002	15.10.2002
DB-05	Installation DB-Server	5 Tage	01.07.2002	05.07.2002
DB-06	Konfiguration DB-Server	15 Tage	08.07.2002	26.07.2002
DB-06.1	User	15 Tage?	08.07.2002	26.07.2002
DB-06.2	Netzwerkanbindung	15 Tage?	08.07.2002	26.07.2002
DB-06.3	Sicherheit	15 Tage?	08.07.2002	26.07.2002
	Ergebnis	1 Tage	26.07.2002	26.07.2002
DB-07	Prototyp (Key-Features)	56 Tage	15.07.2002	27.09.2002
DB-07.1	Persistenz Rechnungen	46 Tage	15.07.2002	15.09.2002
DB-07.2	Persistenz Zahlungsvorgang	46 Tage	15.07.2002	15.09.2002
DB-07.3	Persistenz Businesspartner	56 Tage	15.07.2002	27.09.2002
DB-07.4	Persistenz Systemdaten	56 Tage	15.07.2002	27.09.2002
DB-07.5	Persistenz User	56 Tage	15.07.2002	27.09.2002
DB-07.6	Persistenz GUI	56 Tage	15.07.2002	27.09.2002
DB-07.7	Persistenz Datenkonverter	56 Tage	15.07.2002	27.09.2002
DB-07.8	Persistenz Sicherheit	56 Tage	15.07.2002	27.09.2002
	Lauffähiger Prototyp	1 Tage	27.09.2002	27.09.2002
DB-08	Implementierung	75 Tage	21.10.2002	31.01.2003
DB-08.1	Businesspartner: Invoicing Party, InvoiceRecipient, ProfileGroup	9 Tage	30.10.2002	11.11.2002
DB-08.2	core.common	9 Tage	30.10.2002	11.11.2002
DB-08.3	core.job	9 Tage	30.10.2002	11.11.2002
DB-08.4	core.session	9 Tage	30.10.2002	11.11.2002
DB-08.5	core.user	9 Tage	30.10.2002	11.11.2002
DB-08.6	core.workflow	9 Tage	30.10.2002	11.11.2002
DB-08.7	Businesspartner: FinancialServiceProvider, PersonDataSheet, CompanyDataSheet	6 Tage	11.11.2002	18.11.2002
DB-08.8	UserInformation- InvoicingParty	6 Tage	11.11.2002	18.11.2002
DB-08.9	UserInformation- InvoiceRecipient	6 Tage	11.11.2002	18.11.2002
DB-08.10	security	6 Tage	11.11.2002	18.11.2002
DB-08.11	UserInformation- Group	6 Tage	11.11.2002	18.11.2002
DB-08.12	invoice	6 Tage	11.11.2002	18.11.2002
DB-08.13	Invoice- InvoiceContainer	6 Tage	11.11.2002	18.11.2002
DB-08.14	PaymentTransaction- InvoiceContainer	6 Tage	11.11.2002	18.11.2002
DB-08.15	Adress, ProfileGroupAssignment	6 Tage	18.11.2002	25.11.2002
DB-08.16	FinancialAccount, FinancialAccountAttributes	6 Tage	18.11.2002	25.11.2002
DB-08.17	alert	6 Tage	18.11.2002	25.11.2002
DB-08.18	PaymentWarning- Invoice	6 Tage	18.11.2002	25.11.2002
DB-08.19	InvoiceReminder- Invoice	6 Tage	18.11.2002	25.11.2002
DB-08.20	gui	6 Tage	25.11.2002	02.12.2002

DB-08.21	InvoiceRecipient- History	6 Tage	25.11.2002	02.12.2002
DB-08.22	InvoicingParty- History	6 Tage	25.11.2002	02.12.2002
DB-08.23	payment	6 Tage	25.11.2002	02.12.2002
DB-08.24	SessionInformation- UserInformation	6 Tage	25.11.2002	02.12.2002
DB-08.25	Invoice- InvoiceRecipient	6 Tage	25.11.2002	02.12.2002
DB-08.26	PaymentTransaction- FinancialAccount	6 Tage	25.11.2002	02.12.2002
DB-08.27	WorkflowStartNodeEntry- Group	6 Tage	25.11.2002	02.12.2002
	Lauffähiger Komponente	1 Tage	28.02.2003	28.02.2003

B.1.7: Projektmanagement

Task-ID	Task Name	Dauer	Start	Ende
	Projektplan	1 Tage	14.06.2002	14.06.2002
	Pflege und Wartung des Projektplans	182 Tage	17.06.2002	21.02.2003
	Review Projektplan 1	1 Tage	22.08.2002	22.08.2002
	Review Projektplan 2	1 Tage	19.09.2002	19.09.2002
	Review Projektplan 3	1 Tage	21.10.2002	21.10.2002
	Review Projektplan 4	1 Tage	28.10.2002	28.10.2002
	Review Projektplan 5	1 Tage	04.11.2002	04.11.2002
	Review Projektplan 6	1 Tage	18.11.2002	18.11.2002
	Review Projektplan 7	1 Tage	02.12.2002	02.12.2002
	Review Projektplan 8	1 Tage	16.12.2002	16.12.2002
	Review Projektplan 9	1 Tage	13.01.2003	13.01.2003
	Review Projektplan 10	1 Tage	20.01.2003	20.01.2003
	Review Projektplan 11	1 Tage	03.02.2003	03.02.2003
	Review Projektplan 12	1 Tage	17.02.2003	17.02.2003
	Abwesenheitszeiten organisieren	143 Tage	09.08.2002	21.02.2003

B.1.8: Marketing

Task-ID	Task Name	Dauer	Start	Ende
	T-Shirts	1 Tage	28.06.2002	28.06.2002
	Whitepaper	33 Tage	16.05.2002	01.07.2002
	Zwischenbericht	31 Tage	22.07.2002	02.09.2002
	Plakat	54 Tage	08.08.2002	21.10.2002
	Abschlußbericht	37 Tage	20.01.2003	10.03.2003
	Präsentation in Dortmund um 10 Uhr R318	1 Tage	28.03.2003	28.03.2003

B.1.9: Qualitätsmanagement

Task-ID	Task Name	Dauer	Start	Ende
	Code Guide	1 Tage	28.06.2002	28.06.2002
	Testplan Integrationstest	1 Tage	27.12.2002	27.12.2002
	Testplan Systemtest	1 Tage	31.01.2003	31.01.2003
	Pflege und Wartung des	177 Tage	21.06.2002	20.02.2003

	Anforderungsdokuments			
	Pflege und Wartung der Key-Feature	152 Tage	26.07.2002	20.02.2003
	Erstellung eines Prozessmodells	24 Tage	24.06.2002	25.07.2002
	Pflege und Wartung des Prozessmodells	152 Tage	26.07.2002	20.02.2003

B.1.10: Klassenreviews

Task-ID	Task Name	Dauer	Start	Ende
	Sicherheit	1 Tage	10.12.2002	10.12.2002
	GUI	1 Tage	12.12.2002	12.12.2002
	DK	1 Tage	16.01.2003	16.01.2003
	DB	1 Tage	22.01.2003	22.01.2003
	BL	1 Tage	24.01.2003	24.01.2003

B.1.11: Klassentests

Task-ID	Task Name	Dauer	Start	Ende
	Sicherheit	1 Tage	21.01.2003	21.01.2003
	DB	6 Tage	24.01.2003	31.01.2003
	GUI	1 Tage	24.01.2003	24.01.2003
	DK	1 Tage	27.01.2003	27.01.2003
	BL	1 Tage	30.01.2003	30.01.2003

B.1.12: Klassenreviews

Task-ID	Task Name	Dauer	Start	Ende
	Sicherheit	1 Tage	10.12.2002	10.12.2002
	GUI	1 Tage	12.12.2002	12.12.2002
	DK	1 Tage	16.01.2003	16.01.2003
	DB	1 Tage	22.01.2003	22.01.2003
	BL	1 Tage	24.01.2003	24.01.2003

B.1.13: Integrationstests

Task-ID	Task Name	Dauer	Start	Ende
	GUI - DK	1 Tage	30.01.2003	30.01.2003
	GUI - BL	1 Tage	31.01.2003	31.01.2003
	GUI - Sicherheit	1 Tage	31.01.2003	31.01.2003
	DK - Sicherheit	1 Tage	03.02.2003	03.02.2003
	DK -BL	13 Tage	30.01.2003	15.02.2003
	BL - DB	6 Tage	07.02.2003	15.02.2003

B.1.14: GUI

Task-ID	Task Name	Dauer	Start	Ende

	Style Guide für Web, WAP	67 Tage	16.07.2002	15.10.2002

B.1.15: Chefarchitekten

Task-ID	Task Name	Dauer	Start	Ende
	Systemarchitektur	45 Tage	29.04.2002	28.06.2002
	Pflege und Wartung der Systemarchitektur	171 Tage	01.07.2002	20.02.2003
	Schnittstellenüberwachung	130 Tage	02.07.2002	27.12.2002
	Installation & Einrichtung Bea-Weblogic	21 Tage	04.07.2002	01.08.2002

B.1.16: Systemadministration

Task-ID	Task Name	Dauer	Start	Ende
	Einrichtung Server	1 Tage	31.05.2002	31.05.2002
	Einrichtung Workstations	1 Tage	31.05.2002	31.05.2002
	CVS Repository	1 Tage	31.05.2002	31.05.2002
	Pflege und Wartung der PG Homepage	224 Tage	17.04.2002	20.02.2003
	Installation & Einrichtung Bea-Weblogic	21 Tage	04.07.2002	01.08.2002
	Einrichtung eines FTP-Zugangs	13 Tage	20.06.2002	08.07.2002

B.2 Projektplan (grafisch)

Nachfolgend ist die grafische Version des Projektplans abgebildet.

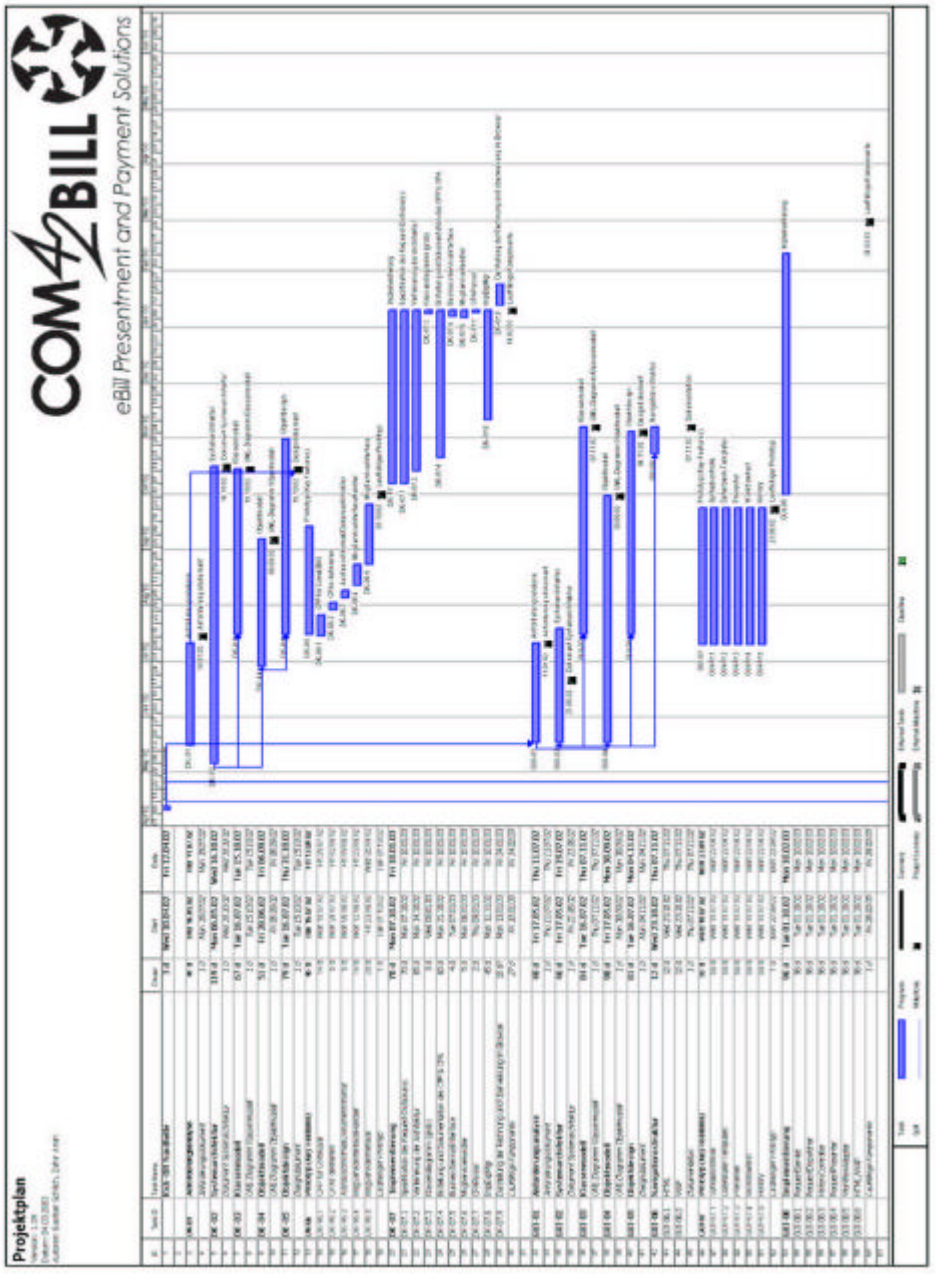


Abbildung 79: Projektplan, Teil 1

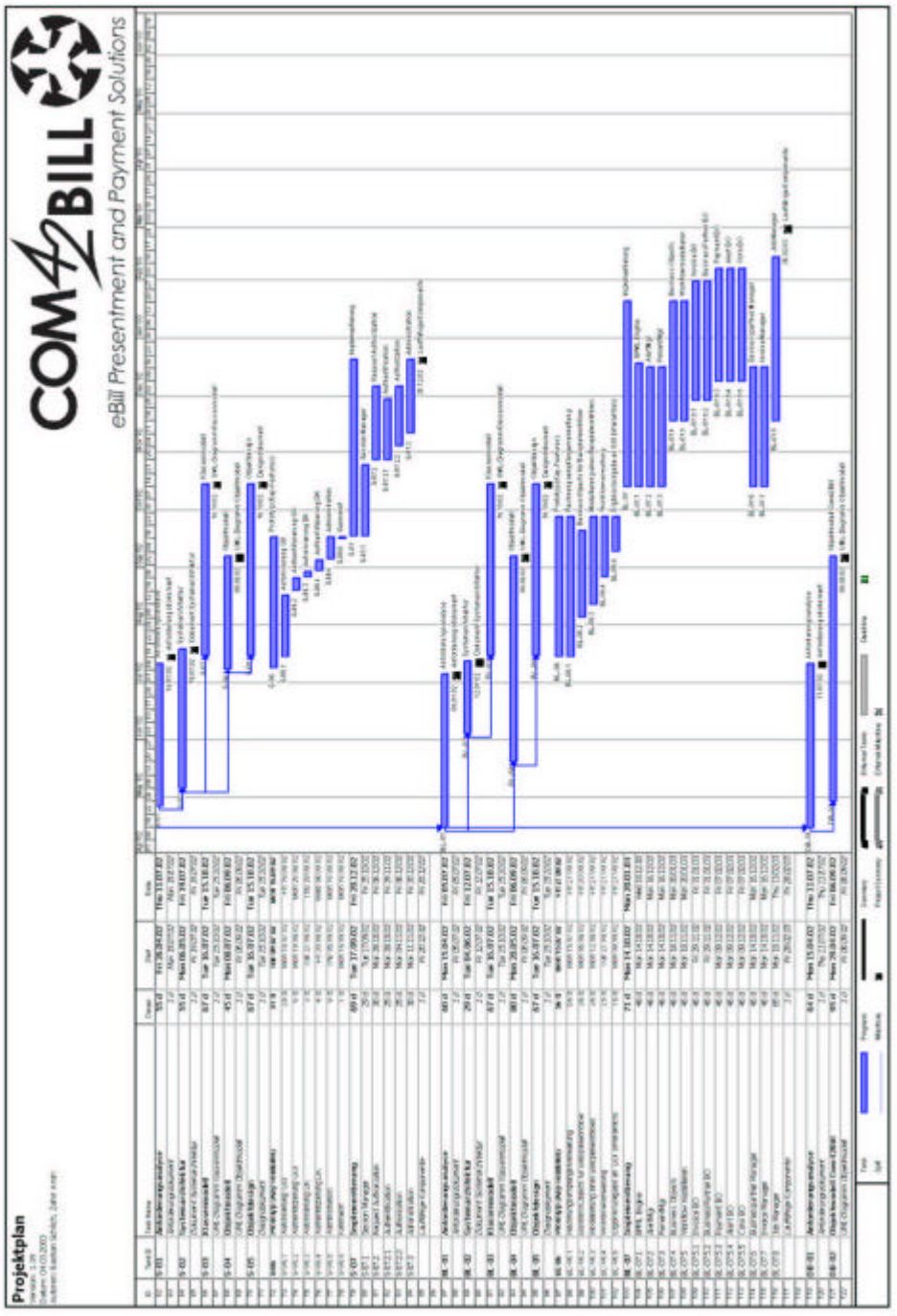


Abbildung 80: Projektplan, Teil 2

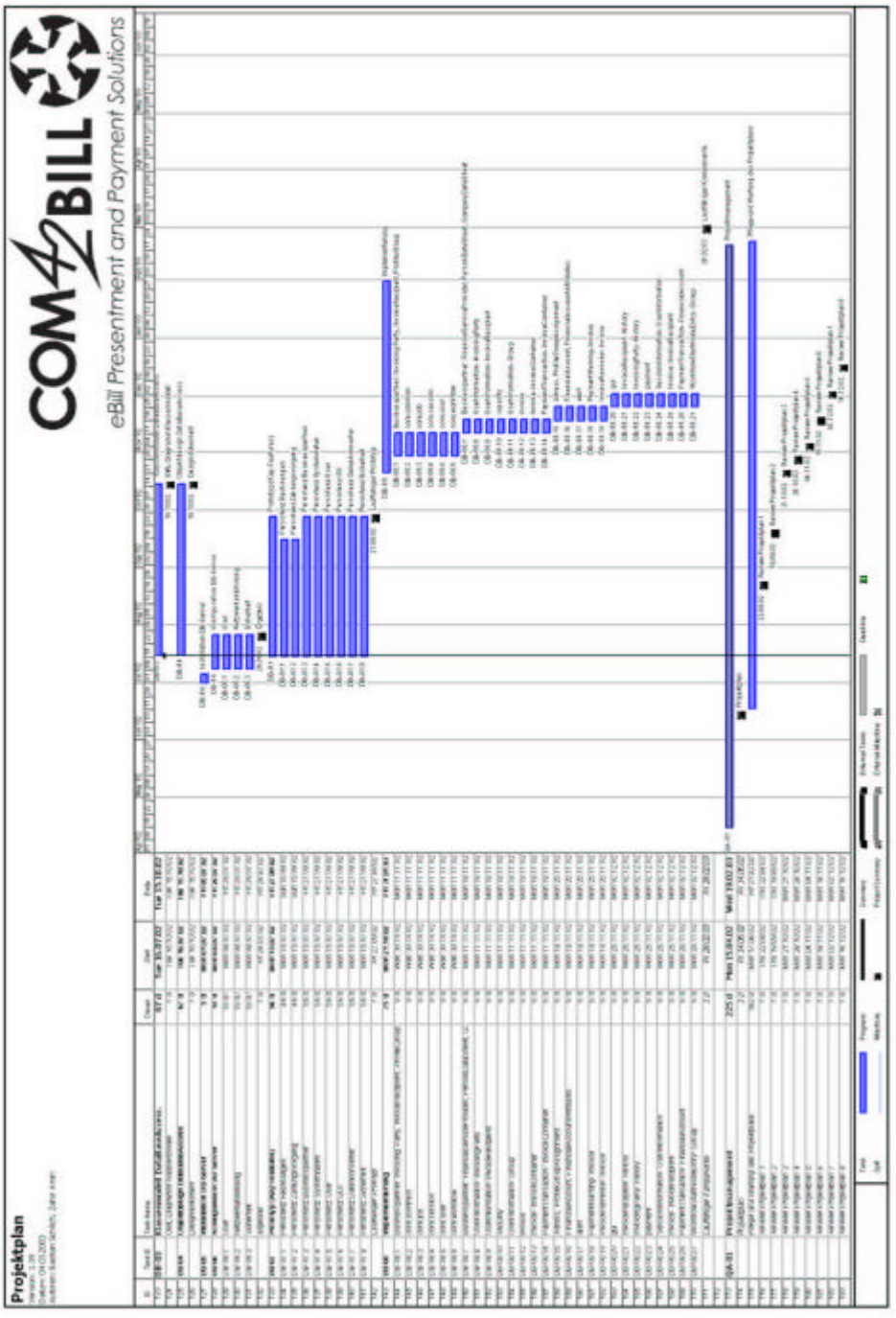


Abbildung 81: Projektplan, Teil 3

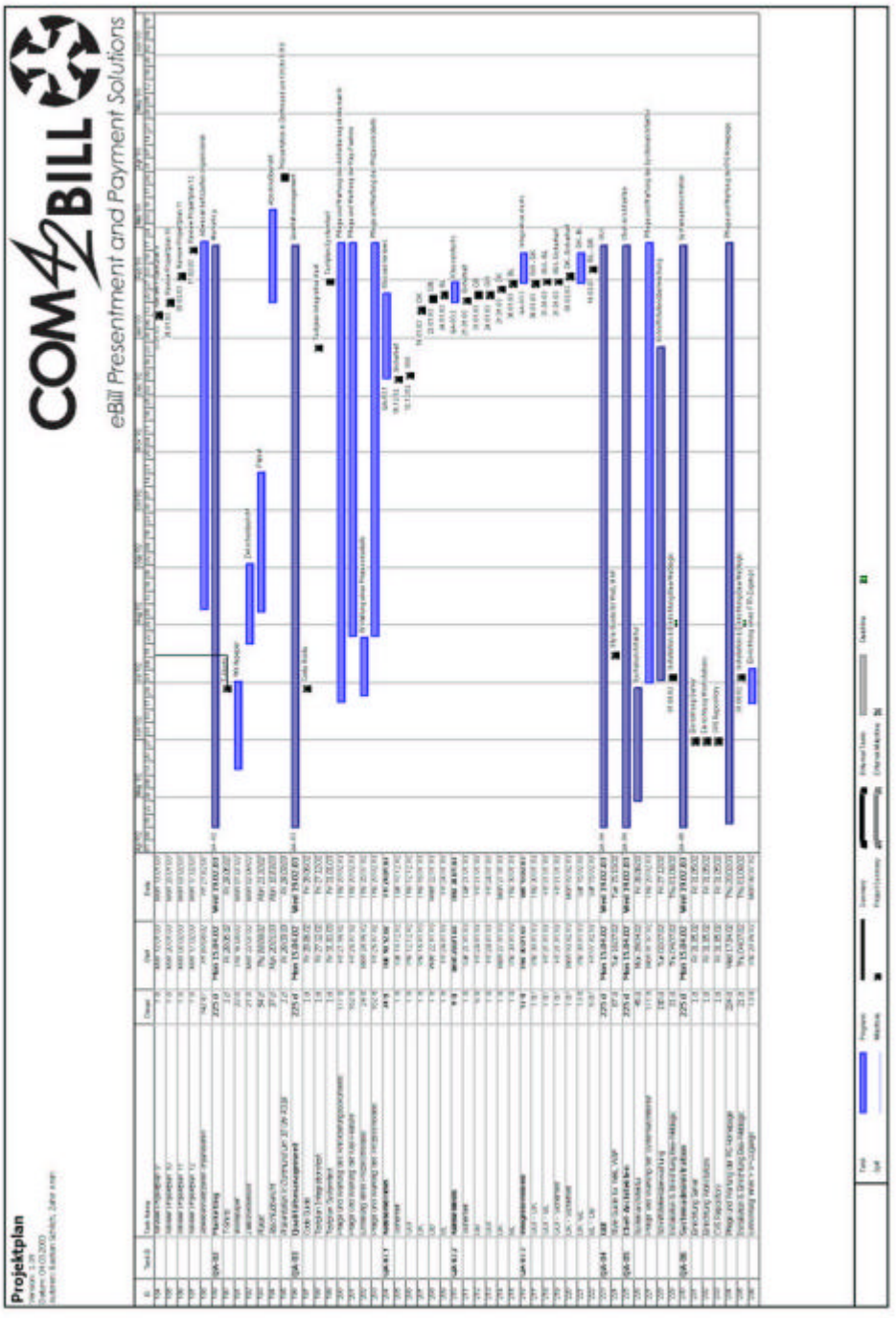


Abbildung 82: Projektplan, Teil 4

Anhang C ebXML-Dokumente

Im Folgenden werden die wichtigsten Dokumente, die für das ebXML-Rahmenwerk benötigt werden, dargestellt.

C.1 CPP für Com42Bill

Das CPP dient der öffentlichen Bekanntmachung von Com42Bill. Es beschreibt, welche Geschäftsprozesse, Nachrichtenformate, Kommunikationsschnittstellen etc. Com42Bill unterstützt.

```
<?xml version="1.0"?>
- <tp:CollaborationProtocolProfile xmlns:tp="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd cpp-cpa-2_0.xsd"
    tp:cppid="uri:http://www.Com42Bill.de/ebxml/cpp_Com42Bill.xml" tp:version="1.0">
  - <!--
    Party info for Com42Bill
    -->
- <tp:PartyInfo tp:partyName="Com42Bill" tp:defaultMshChannelId="Com42BillChannelHTTP_CH"
    tp:defaultMshPackageId="BillingDataImportRequest_PCK">
  <tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">4242424242</tp:PartyId>
  <tp:PartyRef xlink:href="http://www.Com42Bill.de/index.html" />
  - <!--
    Roles for Com42Bill within the business processes
    -->
- <tp:CollaborationRole>
  <tp:ProcessSpecification tp:version="1.0" tp:name="Com42Bill_BusinessProcess01" xlink:type="simple"
    xlink:href="http://www.Com42Bill.de/ebxml/bpss_Com42Bill.xml" tp:uuid="urn:icann:Com42Bill.de:bpid:bp01" />
  <tp:Role tp:name="BillingDataReceiver_RLE" xlink:type="simple"
    xlink:href="http://www.Com42Bill.de/ebxml/bpss_Com42Bill.xml#BillingDataReceiver_RLE" />
- <tp:ServiceBinding>
  <tp:Service>urn:icann:Com42Bill.de:bpid:bp01</tp:Service>
- <tp:CanReceive>
- <tp:ThisPartyActionBinding tp:id="BillingDataImportRequest_ABND" tp:action="BillingDataImportRequestAction"
    tp:packageId="BillingDataImportRequest_PCK">
  <tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="false" tp:isNonRepudiationReceiptRequired="false"
    tp:isConfidential="transient" tp:isAuthenticated="transient" tp:isTamperProof="transient"
    tp:isAuthorizationRequired="true" tp:timeToPerform="P1D" />
  <tp:ActionContext tp:binaryCollaboration="BillingDataImport" tp:businessTransactionActivity="BillingDataImport"
    tp:requestOrResponseAction="BillingDataImportRequestAction" />
  <tp:ChannelId>Com42BillChannelHTTP_CH</tp:ChannelId>
    </tp:ThisPartyActionBinding>
- <tp:CanSend>
- <tp:ThisPartyActionBinding tp:id="BillingDataImportConfirmation_ABND" tp:action="BillingDataImportConfirmationAction"
    tp:packageId="BillingDataImportConfirmation_PCK">
  <tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="false" tp:isNonRepudiationReceiptRequired="false"
    tp:isConfidential="transient" tp:isAuthenticated="transient" tp:isTamperProof="transient"
    tp:isAuthorizationRequired="true" tp:timeToPerform="P1D" />
  <tp:ActionContext tp:binaryCollaboration="BillingDataImport" tp:businessTransactionActivity="BillingDataImport"
    tp:requestOrResponseAction="BillingDataImportConfirmationAction" />
  <tp:ChannelId>Com42BillChannelHTTP_CH</tp:ChannelId>
    </tp:ThisPartyActionBinding>
    </tp:CanSend>
    </tp:CanReceive>
    </tp:ServiceBinding>
    </tp:CollaborationRole>
- <tp:CollaborationRole>
  <tp:ProcessSpecification tp:version="1.0" tp:name="Com42Bill_BusinessProcess01" xlink:type="simple"
    xlink:href="http://www.Com42Bill.de/ebxml/bpss_Com42Bill.xml" tp:uuid="urn:icann:Com42Bill.de:bpid:bp01" />
  <tp:Role tp:name="BillingDataStatusReceiver_RLE" xlink:type="simple"
    xlink:href="http://www.Com42Bill.de/ebxml/bpss_Com42Bill.xml#BillingDataStatusReceiver_RLE" />
```

```
- <tp:ServiceBinding>
  <tp:Service>urn:icann:Com42Bill.de:bpid:bp01</tp:Service>
- <tp:CanReceive>
- <tp:ThisPartyActionBinding tp:id="BillingDataStatusRequest_ABND" tp:action="BillingDataStatusRequestAction"
      tp:packageId="BillingDataStatusRequest_PCK">
  <tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="false" tp:isNonRepudiationReceiptRequired="false"
      tp:isConfidential="transient" tp:isAuthenticated="transient" tp:isTamperProof="transient"
      tp:isAuthorizationRequired="true" tp:timeToPerform="P1D" />
  <tp:ActionContext tp:binaryCollaboration="BillingDataStatus" tp:businessTransactionActivity="BillingDataStatus"
      tp:requestOrResponseAction="BillingDataStatusRequestAction" />
  <tp:ChannelId>Com42BillChannelHTTP_CH</tp:ChannelId>
    </tp:ThisPartyActionBinding>
- <tp:CanSend>
- <tp:ThisPartyActionBinding tp:id="BillingDataStatusConfirmation_ABND" tp:action="BillingDataStatusConfirmationAction"
      tp:packageId="BillingDataStatusConfirmation_PCK">
  <tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="false" tp:isNonRepudiationReceiptRequired="false"
      tp:isConfidential="transient" tp:isAuthenticated="transient" tp:isTamperProof="transient"
      tp:isAuthorizationRequired="true" tp:timeToPerform="P1D" />
  <tp:ActionContext tp:binaryCollaboration="BillingDataStatus" tp:businessTransactionActivity="BillingDataStatus"
      tp:requestOrResponseAction="BillingDataStatusConfirmationAction" />
  <tp:ChannelId>Com42BillChannelHTTP_CH</tp:ChannelId>
    </tp:ThisPartyActionBinding>
    </tp:CanSend>
    </tp:CanReceive>
    </tp:ServiceBinding>
    </tp:CollaborationRole>
- <tp:CollaborationRole>
  <tp:ProcessSpecification tp:version="1.0" tp:name="Com42Bill_BusinessProcess01" xlink:type="simple"
      xlink:href="http://www.Com42Bill.de/ebxml/bpss_Com42Bill.xml" tp:uuid="urn:icann:Com42Bill.de:bpid:bp01" />
  <tp:Role tp:name="BankTransferDataSender_RLE" xlink:type="simple"
      xlink:href="http://www.Com42Bill.de/ebxml/bpss_Com42Bill.xml#BankTransferDataSender_RLE" />
- <tp:ServiceBinding>
  <tp:Service>urn:icann:Com42Bill.de:bpid:bp01</tp:Service>
- <tp:CanSend>
- <tp:ThisPartyActionBinding tp:id="BankTransferDataExportRequest_ABND"
      tp:action="BankTransferDataExportRequestAction" tp:packageId="BankTransferDataExportRequest_PCK">
  <tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="false" tp:isNonRepudiationReceiptRequired="false"
      tp:isConfidential="transient" tp:isAuthenticated="transient" tp:isTamperProof="transient"
      tp:isAuthorizationRequired="true" tp:timeToPerform="P1D" />
  <tp:ActionContext tp:binaryCollaboration="BankTransferDataExport"
      tp:businessTransactionActivity="BankTransferDataExport"
      tp:requestOrResponseAction="BankTransferDataExportRequestAction" />
  <tp:ChannelId>Com42BillChannelHTTP_CH</tp:ChannelId>
    </tp:ThisPartyActionBinding>
- <tp:CanReceive>
- <tp:ThisPartyActionBinding tp:id="BankTransferDataFailureConfirmation_ABND"
      tp:action="BankTransferDataFailureConfirmationAction"
      tp:packageId="BankTransferDataFailureConfirmation_PCK">
  <tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="false" tp:isNonRepudiationReceiptRequired="false"
      tp:isConfidential="transient" tp:isAuthenticated="transient" tp:isTamperProof="transient"
      tp:isAuthorizationRequired="true" tp:timeToPerform="P1D" />
  <tp:ActionContext tp:binaryCollaboration="BankTransferDataExport"
      tp:businessTransactionActivity="BankTransferDataExport"
      tp:requestOrResponseAction="BankTransferDataFailureConfirmationAction" />
  <tp:ChannelId>Com42BillChannelHTTP_CH</tp:ChannelId>
    </tp:ThisPartyActionBinding>
    </tp:CanReceive>
    </tp:CanSend>
    </tp:ServiceBinding>
    </tp:CollaborationRole>
  - <!--
    Security certificates used for transport

    -->
- <tp:Certificate tp:certId="ClientCertificate_CER">
- <ds:KeyInfo>
  <ds:KeyName>ClientCertificateKey?????</ds:KeyName>
  <ds:KeyValue>ClientCertificateValue?????</ds:KeyValue>
    </ds:KeyInfo>
    </tp:Certificate>
- <tp:Certificate tp:certId="ServerCertificate_CER">
```

```
- <ds:KeyInfo>
  <ds:KeyName>???</ds:KeyName>
  <ds:KeyValue>???</ds:KeyValue>
     </ds:KeyInfo>
     </tp:Certificate>
- <tp:Certificate tp:certId="TransportCertificate_CER">
- <ds:KeyInfo>
  <ds:KeyName>???</ds:KeyName>
  <ds:KeyValue>???</ds:KeyValue>
     </ds:KeyInfo>
     </tp:Certificate>
- <tp:SecurityDetails tp:securityId="TransportSecurity_SD">
- <tp:TrustAnchors>
  <tp:AnchorCertificateRef tp:certId="TransportCertificate_CER" />
     </tp:TrustAnchors>
     </tp:SecurityDetails>
   - <!--
    Transport information
     -->
- <tp:DeliveryChannel tp:channelId="Com42BillChannelHTTP_CH" tp:transportId="Com42BillTransport_TP"
      tp:docExchangeId="docExchangeStandard_DEX">
  <tp:MessagingCharacteristics tp:syncReplyMode="mshSignalsOnly" tp:ackRequested="always"
      tp:ackSignatureRequested="never" tp:duplicateElimination="never" />
     </tp:DeliveryChannel>
- <tp:Transport tp:transportId="Com42BillTransport_TP">
- <tp:TransportSender>
  <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
  <tp:AccessAuthentication>basic</tp:AccessAuthentication>
- <tp:TransportClientSecurity>
  <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
  <tp:ClientCertificateRef tp:certId="ClientCertificate_CER" />
  <tp:ServerSecurityDetailsRef tp:securityId="TransportSecurity_SD" />
     </tp:TransportClientSecurity>
     </tp:TransportSender>
- <tp:TransportReceiver>
  <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
  <tp:AccessAuthentication>basic</tp:AccessAuthentication>
  <tp:Endpoint tp:uri="https://www.Com42Bill.de/MsgReceiver" tp:type="allPurpose" />
- <tp:TransportServerSecurity>
  <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
  <tp:ServerCertificateRef tp:certId="ServerCertificate_CER" />
  <tp:ClientSecurityDetailsRef tp:securityId="TransportSecurity_SD" />
     </tp:TransportServerSecurity>
     </tp:TransportReceiver>
     </tp:Transport>
- <tp:DocExchange tp:docExchangeId="docExchangeStandard_DEX">
- <tp:ebXMLSenderBinding tp:version="1.0">
- <tp:ReliableMessaging>
  <tp:Retries>3</tp:Retries>
  <tp:RetryInterval>PT2H</tp:RetryInterval>
  <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
     </tp:ReliableMessaging>
  <tp:PersistDuration>P1D</tp:PersistDuration>
     </tp:ebXMLSenderBinding>
- <tp:ebXMLReceiverBinding tp:version="1.0">
- <tp:ReliableMessaging>
  <tp:Retries>3</tp:Retries>
  <tp:RetryInterval>PT2H</tp:RetryInterval>
  <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
     </tp:ReliableMessaging>
  <tp:PersistDuration>P1D</tp:PersistDuration>
     </tp:ebXMLReceiverBinding>
     </tp:DocExchange>
     </tp:PartyInfo>
   - <!--
    Business Documents being exchanged within the business processes
     -->
   - <!--
    SimplePart corresponding to the SOAP Envelope
```

```
    -->
- <tp:SimplePart tp:id="SoapEnv_SP" tp:mimetype="text/xml">
  <tp:NamespaceSupported tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-
      2_0.xsd">http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-
      2_0.xsd</tp:NamespaceSupported>
    </tp:SimplePart>
  - <!--
    SimplePart corresponding to a BillingDataImportRequest
    -->
- <tp:SimplePart tp:id="BillingDataImportRequest_SP" tp:mimetype="text/xml">
  <tp:NamespaceSupported
      tp:location="www.Com42Bill.de/ebxml/BillingDataImportRequest.xsd">www.Com42Bill.de/ebxml/BillingDataImp
      ortRequest.xsd</tp:NamespaceSupported>
    </tp:SimplePart>
  - <!--
    SimplePart corresponding to a BillingDataImportConfirmation
    -->
- <tp:SimplePart tp:id="BillingDataImportConfirmation_SP" tp:mimetype="text/xml">
  <tp:NamespaceSupported
      tp:location="www.Com42Bill.de/ebxml/BillingDataImportConfirmation.xsd">www.Com42Bill.de/ebxml/BillingDat
      aImportConfirmation.xsd</tp:NamespaceSupported>
    </tp:SimplePart>
  - <!--
    SimplePart corresponding to a BillingDataStatusRequest
    -->
- <tp:SimplePart tp:id="BillingDataStatusRequest_SP" tp:mimetype="text/xml">
  <tp:NamespaceSupported
      tp:location="www.Com42Bill.de/ebxml/BillingDataStatusRequest.xsd">www.Com42Bill.de/ebxml/BillingDataStatu
      sRequest.xsd</tp:NamespaceSupported>
    </tp:SimplePart>
  - <!--
    SimplePart corresponding to a BankTransferDataExportRequest
    -->
- <tp:SimplePart tp:id="BankTransferDataExportRequest_SP" tp:mimetype="text/xml">
  <tp:NamespaceSupported
      tp:location="www.Com42Bill.de/ebxml/BankTransferDataExportRequest.xsd">www.Com42Bill.de/ebxml/BankTra
      nsferDataExportRequest.xsd</tp:NamespaceSupported>
    </tp:SimplePart>
  - <!--
    SimplePart corresponding to a BankTransferDataFailureConfirmation
    -->
- <tp:SimplePart tp:id="BankTransferDataFailureConfirmation_SP" tp:mimetype="text/xml">
  <tp:NamespaceSupported
      tp:location="www.Com42Bill.de/ebxml/BankTransferDataFailureConfirmationt.xsd">www.Com42Bill.de/ebxml/Ba
      nkTransferDataFailureConfirmationt.xsd</tp:NamespaceSupported>
    </tp:SimplePart>
  - <!--
    SimplePart corresponding to a BillingDataStatusConfirmation
    -->
- <tp:SimplePart tp:id="BillingDataStatusConfirmation_SP" tp:mimetype="text/xml">
  <tp:NamespaceSupported
      tp:location="www.Com42Bill.de/ebxml/BillingDataStatusConfirmation.xsd">www.Com42Bill.de/ebxml/BillingData
      StatusConfirmation.xsd</tp:NamespaceSupported>
    </tp:SimplePart>
  - <!--
    An ebXML message with a SOAP Envelope plus a BillingDataImportRequest
    -->
- <tp:Packaging tp:id="BillingDataImportRequest_PCK">
  <tp:ProcessingCapabilities tp:parse="true" tp:generate="true" />
- <tp:CompositeList>
- <tp:Composite tp:id="BillingDataImportRequest_CMP" tp:mimetype="multipart/related"
      tp:mimeparameters="type=text/xml">
  <tp:Constituent tp:idref="SoapEnv_SP" />
  <tp:Constituent tp:idref="BillingDataImportRequest_SP" />
    </tp:Composite>
    </tp:CompositeList>
    </tp:Packaging>
  - <!--
```

```
    An ebXML message with a SOAP Envelope plus a BillingDataImportConfirmation
    -->
- <tp:Packaging tp:id="BillingDataImportConfirmation_PCK">
  <tp:ProcessingCapabilities tp:parse="true" tp:generate="true" />
- <tp:CompositeList>
- <tp:Composite tp:id="BillingDataImportConfirmation_CMP" tp:mimetype="multipart/related"
      tp:mimeparameters="type=text/xml">
  <tp:Constituent tp:idref="SoapEnv_SP" />
  <tp:Constituent tp:idref="BillingDataImportConfirmation_SP" />
    </tp:Composite>
    </tp:CompositeList>
    </tp:Packaging>
  - <!--
    An ebXML message with a SOAP Envelope plus a BillingDataStatusRequest
    -->
- <tp:Packaging tp:id="BillingDataStatusRequest_PCK">
  <tp:ProcessingCapabilities tp:parse="true" tp:generate="true" />
- <tp:CompositeList>
- <tp:Composite tp:id="BillingDataStatusRequest_CMP" tp:mimetype="multipart/related"
      tp:mimeparameters="type=text/xml">
  <tp:Constituent tp:idref="SoapEnv_SP" />
  <tp:Constituent tp:idref="BillingDataStatusRequest_SP" />
    </tp:Composite>
    </tp:CompositeList>
    </tp:Packaging>
  - <!--
    An ebXML message with a SOAP Envelope plus a BankTransferDataExportRequest
    -->
- <tp:Packaging tp:id="BankTransferDataExportRequest_PCK">
  <tp:ProcessingCapabilities tp:parse="true" tp:generate="true" />
- <tp:CompositeList>
-         <tp:Composite          tp:id="BankTransferDataExportRequest_CMP"          tp:mimetype="multipart/related"
      tp:mimeparameters="type=text/xml">
  <tp:Constituent tp:idref="SoapEnv_SP" />
  <tp:Constituent tp:idref="BankTransferDataExportRequest_SP" />
    </tp:Composite>
    </tp:CompositeList>
    </tp:Packaging>
  - <!--
    An ebXML message with a SOAP Envelope plus a BankTransferDataFailureConfirmation
    -->
- <tp:Packaging tp:id="BankTransferDataFailureConfirmation_PCK">
  <tp:ProcessingCapabilities tp:parse="true" tp:generate="true" />
- <tp:CompositeList>
- <tp:Composite tp:id="BankTransferDataFailureConfirmation_CMP" tp:mimetype="multipart/related"
      tp:mimeparameters="type=text/xml">
  <tp:Constituent tp:idref="SoapEnv_SP" />
  <tp:Constituent tp:idref="BankTransferDataFailureConfirmation_SP" />
    </tp:Composite>
    </tp:CompositeList>
    </tp:Packaging>
  - <!--
    An ebXML message with a SOAP Envelope plus a BillingDataStatusConfirmation
    -->
- <tp:Packaging tp:id="BillingDataStatusConfirmation_PCK">
  <tp:ProcessingCapabilities tp:parse="true" tp:generate="true" />
- <tp:CompositeList>
- <tp:Composite tp:id="BillingDataStatusConfirmation_CMP" tp:mimetype="multipart/related"
      tp:mimeparameters="type=text/xml">
  <tp:Constituent tp:idref="SoapEnv_SP" />
  <tp:Constituent tp:idref="BillingDataStatusConfirmation_SP" />
    </tp:Composite>
    </tp:CompositeList>
    </tp:Packaging>
  <tp:Comment xml:lang="en-US">Com42Bill's Collaboration Protocol Profile</tp:Comment>
    </tp:CollaborationProtocolProfile>
```

C.2 CPA zwischen Com42Bill und einem Rechnungssteller

Das CPA stellt einen Vertrag zwischen Com42Bill und einem Rechnungsteller dar, anhand dessen die elektronischen Geschäftstransaktionen zwischen den beiden Parteien durchgeführt werden können.

```
<?xml version="1.0"?>
- <tp:CollaborationProtocolAgreement xmlns:tp="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd cpp-cpa-2_0.xsd"
    tp:cpaid="uri:http://www.Com42Bill.de/ebxml/cpa_Com42Bill_invoicingParty.xml" tp:version="1.0">
  <tp:Status tp:value="agreed" />
  <tp:Start>2002-10-01T07:21:00Z</tp:Start>
  <tp:End>2003-02-28T07:21:00Z</tp:End>
  - <!--
    Party info for Com42Bill
    -->
- <tp:PartyInfo tp:partyName="Com42Bill" tp:defaultMshChannelId="Com42BillChannelHTTP_CH"
    tp:defaultMshPackageId="BillingDataImportRequest_PCK">
  <tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">4242424242</tp:PartyId>
  <tp:PartyRef xlink:href="http://www.Com42Bill.de/index.html" />
  - <!--
    Roles for Com42Bill within the business processes
    -->
- <tp:CollaborationRole>
  <tp:ProcessSpecification tp:version="1.0" tp:name="Com42Bill_BusinessProcess01" xlink:type="simple"
    xlink:href="http://www.Com42Bill.de/ebxml/bpss_Com42Bill.xml" tp:uuid="urn:icann:Com42Bill.de:bpid:bp01" />
  <tp:Role tp:name="BillingDataReceiver_RLE" xlink:type="simple"
    xlink:href="http://www.Com42Bill.de/ebxml/bpss_Com42Bill.xml#BillingDataReceiver_RLE" />
- <tp:ServiceBinding>
  <tp:Service>urn:icann:Com42Bill.de:bpid:bp01</tp:Service>
- <tp:CanReceive>
- <tp:ThisPartyActionBinding tp:id="BillingDataImportRequest_ABND" tp:action="BillingDataImportRequestAction"
    tp:packageId="BillingDataImportRequest_PCK">
  <tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="false" tp:isNonRepudiationReceiptRequired="false"
    tp:isConfidential="transient" tp:isAuthenticated="transient" tp:isTamperProof="transient"
    tp:isAuthorizationRequired="true" tp:timeToPerform="P1D" />
  <tp:ActionContext tp:binaryCollaboration="BillingDataImport" tp:businessTransactionActivity="BillingDataImport"
    tp:requestOrResponseAction="BillingDataImportRequestAction" />
  <tp:ChannelId>Com42BillChannelHTTP_CH</tp:ChannelId>
    </tp:ThisPartyActionBinding>
- <tp:CanSend>
- <tp:ThisPartyActionBinding tp:id="BillingDataImportConfirmation_ABND" tp:action="BillingDataImportConfirmationAction"
    tp:packageId="BillingDataImportConfirmation_PCK">
  <tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="false" tp:isNonRepudiationReceiptRequired="false"
    tp:isConfidential="transient" tp:isAuthenticated="transient" tp:isTamperProof="transient"
    tp:isAuthorizationRequired="true" tp:timeToPerform="P1D" />
  <tp:ActionContext tp:binaryCollaboration="BillingDataImport" tp:businessTransactionActivity="BillingDataImport"
    tp:requestOrResponseAction="BillingDataImportConfirmationAction" />
  <tp:ChannelId>Com42BillChannelHTTP_CH</tp:ChannelId>
    </tp:ThisPartyActionBinding>
    </tp:CanSend>
    </tp:CanReceive>
    </tp:ServiceBinding>
    </tp:CollaborationRole>
- <tp:CollaborationRole>
  <tp:ProcessSpecification tp:version="1.0" tp:name="Com42Bill_BusinessProcess01" xlink:type="simple"
    xlink:href="http://www.Com42Bill.de/ebxml/bpss_Com42Bill.xml" tp:uuid="urn:icann:Com42Bill.de:bpid:bp01" />
  <tp:Role tp:name="BillingDataStatusReceiver_RLE" xlink:type="simple"
    xlink:href="http://www.Com42Bill.de/ebxml/bpss_Com42Bill.xml#BillingDataStatusReceiver_RLE" />
- <tp:ServiceBinding>
  <tp:Service>urn:icann:Com42Bill.de:bpid:bp01</tp:Service>
- <tp:CanReceive>
```

```
- <tp:ThisPartyActionBinding tp:id="BillingDataStatusRequest_ABND" tp:action="BillingDataStatusRequestAction"
      tp:packageId="BillingDataStatusRequest_PCK">
  <tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="false" tp:isNonRepudiationReceiptRequired="false"
      tp:isConfidential="transient" tp:isAuthenticated="transient" tp:isTamperProof="transient"
      tp:isAuthorizationRequired="true" tp:timeToPerform="P1D" />
  <tp:ActionContext tp:binaryCollaboration="BillingDataStatus" tp:businessTransactionActivity="BillingDataStatus"
      tp:requestOrResponseAction="BillingDataStatusRequestAction" />
  <tp:ChannelId>Com42BillChannelHTTP_CH</tp:ChannelId>
    </tp:ThisPartyActionBinding>
- <tp:CanSend>
- <tp:ThisPartyActionBinding tp:id="BillingDataStatusConfirmation_ABND" tp:action="BillingDataStatusConfirmationAction"
      tp:packageId="BillingDataStatusConfirmation_PCK">
  <tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="false" tp:isNonRepudiationReceiptRequired="false"
      tp:isConfidential="transient" tp:isAuthenticated="transient" tp:isTamperProof="transient"
      tp:isAuthorizationRequired="true" tp:timeToPerform="P1D" />
  <tp:ActionContext tp:binaryCollaboration="BillingDataStatus" tp:businessTransactionA ctivity="BillingDataStatus"
      tp:requestOrResponseAction="BillingDataStatusConfirmationAction" />
  <tp:ChannelId>Com42BillChannelHTTP_CH</tp:ChannelId>
    </tp:ThisPartyActionBinding>
    </tp:CanSend>
    </tp:CanReceive>
    </tp:ServiceBinding>
    </tp:CollaborationRole>
  - <!--
  Security certificates used for transport

    -->
- <tp:Certificate tp:certId="ClientCertificate_CER">
- <ds:KeyInfo>
  <ds:KeyName>ClientCertificateKey?????</ds:KeyName>
  <ds:KeyValue>ClientCertificateValue?????</ds:KeyValue>
    </ds:KeyInfo>
    </tp:Certificate>
- <tp:Certificate tp:certId="ServerCertificate_CER">
- <ds:KeyInfo>
  <ds:KeyName>???</ds:KeyName>
  <ds:KeyValue>???</ds:KeyValue>
    </ds:KeyInfo>
    </tp:Certificate>
- <tp:Certificate tp:certId="TransportCertificate_CER">
- <ds:KeyInfo>
  <ds:KeyName>???</ds:KeyName>
  <ds:KeyValue>???</ds:KeyValue>
    </ds:KeyInfo>
    </tp:Certificate>
- <tp:SecurityDetails tp:securityId="TransportSecurity_SD">
- <tp:TrustAnchors>
  <tp:AnchorCertificateRef tp:certId="TransportCertificate_CER" />
    </tp:TrustAnchors>
    </tp:SecurityDetails>
  - <!--
  Transport information

    -->
- <tp:DeliveryChannel tp:channelId="Com42BillChannelHTTP_CH" tp:transportId="Com42BillTransport_TP"
      tp:docExchangeId="docExchangeStandard_DEX">
  <tp:MessagingCharacteristics tp:syncReplyMode="mshSignalsOnly" tp:ackRequested="always"
      tp:ackSignatureRequested="never" tp:duplicateElimination="never" />
    </tp:DeliveryChannel>
- <tp:Transport tp:transportId="Com42BillTransport_TP">
- <tp:TransportSender>
  <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
  <tp:AccessAuthentication>basic</tp:AccessAuthentication>
- <tp:TransportClientSecurity>
  <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
  <tp:ClientCertificateRef tp:certId="ClientCertificate_CER" />
  <tp:ServerSecurityDetailsRef tp:securityId="TransportSecurity_SD" />
    </tp:TransportClientSecurity>
    </tp:TransportSender>
- <tp:TransportReceiver>
  <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
```

```
<tp:AccessAuthentication>basic</tp:AccessAuthentication>
<tp:Endpoint tp:uri="http://127.0.0.1:7001/DKServlets/MsgReceiver" tp:type="allPurpose" />
- <tp:TransportServerSecurity>
<tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
<tp:ServerCertificateRef tp:certId="ServerCertificate_CER" />
<tp:ClientSecurityDetailsRef tp:securityId="TransportSecurity_SD" />
  </tp:TransportServerSecurity>
  </tp:TransportReceiver>
  </tp:Transport>
- <tp:DocExchange tp:docExchangeId="docExchangeStandard_DEX">
- <tp:ebXMLSenderBinding tp:version="1.0">
- <tp:ReliableMessaging>
<tp:Retries>3</tp:Retries>
<tp:RetryInterval>PT2H</tp:RetryInterval>
<tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
  </tp:ReliableMessaging>
<tp:PersistDuration>P1D</tp:PersistDuration>
  </tp:ebXMLSenderBinding>
- <tp:ebXMLReceiverBinding tp:version="1.0">
- <tp:ReliableMessaging>
<tp:Retries>3</tp:Retries>
<tp:RetryInterval>PT2H</tp:RetryInterval>
<tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
  </tp:ReliableMessaging>
<tp:PersistDuration>P1D</tp:PersistDuration>
  </tp:ebXMLReceiverBinding>
  </tp:DocExchange>
  </tp:PartyInfo>
 - <!--
  Party info for InvoicingParty

  -->
- <tp:PartyInfo tp:partyName="InvoicingParty" tp:defaultMshChannelId="InvoicingPartyChannelHTTP_CH"
    tp:defaultMshPackageId="BillingDataImportRequest_PCK">
<tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">313131313131</tp:PartyId>
<tp:PartyRef xlink:href="http://www.Com42Bill.de/invoicingParty/index.html" />
 - <!--
 Roles for Invoicing Party within the business processes

  -->
- <tp:CollaborationRole>
<tp:ProcessSpecification tp:version="1.0" tp:name="Com42Bill_BusinessProcess01" xlink:type="simple"
    xlink:href="http://www.Com42Bill.de/ebxml/bpss_Com42Bill.xml" tp:uuid="urn:icann:Com42Bill.de:bpid:bp01" />
<tp:Role tp:name="BillingDataSender_RLE" xlink:type="simple"
    xlink:href="http://www.Com42Bill.de/ebxml/bpss_Com42Bill.xml#BillingDataSender_RLE" />
- <tp:ServiceBinding>
<tp:Service>urn:icann:Com42Bill.de:bpid:bp01</tp:Service>
- <tp:CanSend>
- <tp:ThisPartyActionBinding tp:id="BillingDataImportRequest_invoicingParty_ABND"
    tp:action="BillingDataImportRequestAction" tp:packageId="BillingDataImportRequest_PCK">
<tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="false" tp:isNonRepudiationReceiptRequired="false"
    tp:isConfidential="transient" tp:isAuthenticated="transient" tp:isTamperProof="transient"
    tp:isAuthorizationRequired="true" tp:timeToPerform="P1D" />
<tp:ActionContext tp:binaryCollaboration="BillingDataImport" tp:businessTransactionActivity="BillingDataImport"
    tp:requestOrResponseAction="BillingDataImportRequestAction" />
<tp:ChannelId>InvoicingPartyChannelHTTP_CH</tp:ChannelId>
  </tp:ThisPartyActionBinding>
- <tp:CanReceive>
- <tp:ThisPartyActionBinding tp:id="BillingDataImportConfirmation_invoicingParty_ABND"
    tp:action="BillingDataImportConfirmationAction" tp:packageId="BillingDataImportConfirmation_PCK">
<tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="false" tp:isNonRepudiationReceiptRequired="false"
    tp:isConfidential="transient" tp:isAuthenticated="transient" tp:isTamperProof="transient"
    tp:isAuthorizationRequired="true" tp:timeToPerform="P1D" />
<tp:ActionContext tp:binaryCollaboration="BillingDataImport" tp:businessTransactionActivity="BillingDataImport"
    tp:requestOrResponseAction="BillingDataImportConfirmationAction" />
<tp:ChannelId>InvoicingPartyChannelHTTP_CH</tp:ChannelId>
  </tp:ThisPartyActionBinding>
  </tp:CanReceive>
  </tp:CanSend>
  </tp:ServiceBinding>
  </tp:CollaborationRole>
```

```
- <tp:CollaborationRole>
  <tp:ProcessSpecification tp:version="1.0" tp:name="Com42Bill_BusinessProcess01" xlink:type="simple"
      xlink:href="http://www.Com42Bill.de/ebxml/bpss_Com42Bill.xml" tp:uuid="urn:icann:Com42Bill.de:bpid:bp01" />
  <tp:Role tp:name="BillingDataStatusSender_RLE" xlink:type="simple"
      xlink:href="http://www.Com42Bill.de/ebxml/bpss_Com42Bill.xml#BillingDataStatusSender_RLE" />
- <tp:ServiceBinding>
  <tp:Service>urn:icann:Com42Bill.de:bpid:bp01</tp:Service>
- <tp:CanSend>
- <tp:ThisPartyActionBinding tp:id="BillingDataStatusRequest_invoicingParty_ABND"
      tp:action="BillingDataStatusRequestAction" tp:packageId="BillingDataStatusRequest_PCK">
  <tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="false" tp:isNonRepudiationReceiptRequired="false"
      tp:isConfidential="transient" tp:isAuthenticated="transient" tp:isTamperProof="transient"
      tp:isAuthorizationRequired="true" tp:timeToPerform="P1D" />
  <tp:ActionContext tp:binaryCollaboration="BillingDataStatus" tp:businessTransactionActivity="BillingDataStatus"
      tp:requestOrResponseAction="BillingDataStatusRequestAction" />
  <tp:ChannelId>InvoicingPartyChannelHTTP_CH</tp:ChannelId>
    </tp:ThisPartyActionBinding>
- <tp:CanReceive>
- <tp:ThisPartyActionBinding tp:id="BillingDataStatusConfirmation_invoicingParty_ABND"
      tp:action="BillingDataStatusConfirmationAction" tp:packageId="BillingDataStatusConfirmation_PCK">
  <tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="false" tp:isNonRepudiationReceiptRequired="false"
      tp:isConfidential="transient" tp:isAuthenticated="transient" tp:isTamperProof="transient"
      tp:isAuthorizationRequired="true" tp:timeToPerform="P1D" />
  <tp:ActionContext tp:binaryCollaboration="BillingDataStatus" tp:businessTransactionActivity="BillingDataStatus"
      tp:requestOrResponseAction="BillingDataStatusConfirmationAction" />
  <tp:ChannelId>InvoicingPartyChannelHTTP_CH</tp:ChannelId>
    </tp:ThisPartyActionBinding>
    </tp:CanReceive>
    </tp:CanSend>
    </tp:ServiceBinding>
    </tp:CollaborationRole>
  - <!--
  Security certificates used for transport

    -->
- <tp:Certificate tp:certId="ClientCertificate_invoicingParty_CER">
- <ds:KeyInfo>
  <ds:KeyName>ClientCertificateKey?????</ds:KeyName>
  <ds:KeyValue>ClientCertificateValue?????</ds:KeyValue>
    </ds:KeyInfo>
    </tp:Certificate>
- <tp:Certificate tp:certId="ServerCertificate_invoicingParty_CER">
- <ds:KeyInfo>
  <ds:KeyName>???</ds:KeyName>
  <ds:KeyValue>???</ds:KeyValue>
    </ds:KeyInfo>
    </tp:Certificate>
- <tp:Certificate tp:certId="TransportCertificate_invoicingParty_CER">
- <ds:KeyInfo>
  <ds:KeyName>???</ds:KeyName>
  <ds:KeyValue>???</ds:KeyValue>
    </ds:KeyInfo>
    </tp:Certificate>
- <tp:SecurityDetails tp:securityId="TransportSecurity_invoicingParty_SD">
- <tp:TrustAnchors>
  <tp:AnchorCertificateRef tp:certId="TransportCertificate_invoicingParty_CER" />
    </tp:TrustAnchors>
    </tp:SecurityDetails>
  - <!--
  Transport information

    -->
- <tp:DeliveryChannel tp:channelId="InvoicingPartyChannelHTTP_CH" tp:transportId="InvoicingPartyTransport_TP"
      tp:docExchangeId="docExchangeStandard_DEX">
  <tp:MessagingCharacteristics tp:syncReplyMode="mshSignalsOnly" tp:ackRequested="always"
      tp:ackSignatureRequested="never" tp:duplicateElimination="never" />
    </tp:DeliveryChannel>
- <tp:Transport tp:transportId="InvoicingPartyTransport_TP">
- <tp:TransportSender>
  <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
  <tp:AccessAuthentication>basic</tp:AccessAuthentication>
```

```
- <tp:TransportClientSecurity>
  <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
  <tp:ClientCertificateRef tp:certId="ClientCertificate_CER" />
  <tp:ServerSecurityDetailsRef tp:securityId="TransportSecurity_SD" />
     </tp:TransportClientSecurity>
     </tp:TransportSender>
- <tp:TransportReceiver>
  <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
  <tp:AccessAuthentication>basic</tp:AccessAuthentication>
  <tp:Endpoint tp:uri="http://127.0.0.1:7001/DKServlets/InvoicingPartyServlet" tp:type="allPurpose" />
- <tp:TransportServerSecurity>
  <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
  <tp:ServerCertificateRef tp:certId="ServerCertificate_CER" />
  <tp:ClientSecurityDetailsRef tp:securityId="TransportSecurity_SD" />
     </tp:TransportServerSecurity>
     </tp:TransportReceiver>
     </tp:Transport>
- <tp:DocExchange tp:docExchangeId="docExchangeStandard_invoicingParty_DEX">
- <tp:ebXMLSenderBinding tp:version="1.0">
- <tp:ReliableMessaging>
  <tp:Retries>3</tp:Retries>
  <tp:RetryInterval>PT2H</tp:RetryInterval>
  <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
     </tp:ReliableMessaging>
  <tp:PersistDuration>P1D</tp:PersistDuration>
     </tp:ebXMLSenderBinding>
- <tp:ebXMLReceiverBinding tp:version="1.0">
- <tp:ReliableMessaging>
  <tp:Retries>3</tp:Retries>
  <tp:RetryInterval>PT2H</tp:RetryInterval>
  <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
     </tp:ReliableMessaging>
  <tp:PersistDuration>P1D</tp:PersistDuration>
     </tp:ebXMLReceiverBinding>
     </tp:DocExchange>
     </tp:PartyInfo>
   - <!--
    Business Documents being exchanged within the business processes
     -->
   - <!--
    SimplePart corresponding to the SOAP Envelope
     -->
- <tp:SimplePart tp:id="SoapEnv_SP" tp:mimetype="text/xml">
  <tp:NamespaceSupported tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-
     2_0.xsd">http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-
     2_0.xsd</tp:NamespaceSupported>
     </tp:SimplePart>
   - <!--
    SimplePart corresponding to a BillingDataImportRequest
     -->
- <tp:SimplePart tp:id="BillingDataImportRequest_SP" tp:mimetype="text/xml">
  <tp:NamespaceSupported
     tp:location="www.Com42Bill.de/ebxml/BillingDataImportRequest.xsd">www.Com42Bill.de/ebxml/BillingDataImp
     ortRequest.xsd</tp:NamespaceSupported>
     </tp:SimplePart>
   - <!--
    SimplePart corresponding to a BillingDataImportConfirmation
     -->
- <tp:SimplePart tp:id="BillingDataImportConfirmation_SP" tp:mimetype="text/xml">
  <tp:NamespaceSupported
     tp:location="www.Com42Bill.de/ebxml/BillingDataImportConfirmation.xsd">www.Com42Bill.de/ebxml/BillingDat
     aImportConfirmation.xsd</tp:NamespaceSupported>
     </tp:SimplePart>
   - <!--
    SimplePart corresponding to a BillingDataStatusRequest
     -->
- <tp:SimplePart tp:id="BillingDataStatusRequest_SP" tp:mimetype="text/xml">
```

```
<tp:NamespaceSupported
    tp:location="www.Com42Bill.de/ebxml/BillingDataStatusRequest.xsd">www.Com42Bill.de/ebxml/BillingDataStatu
    sRequest.xsd</tp:NamespaceSupported>
  </tp:SimplePart>
 - <!--
  SimplePart corresponding to a BillingDataStatusConfirmation
  -->
- <tp:SimplePart tp:id="BillingDataStatusConfirmation_SP" tp:mimetype="text/xml">
  <tp:NamespaceSupported
    tp:location="www.Com42Bill.de/ebxml/BillingDataStatusConfirmation.xsd">www.Com42Bill.de/ebxml/BillingData
    StatusConfirmation.xsd</tp:NamespaceSupported>
  </tp:SimplePart>
 - <!--
  An ebXML message with a SOAP Envelope plus a BillingDataImportRequest
  -->
- <tp:Packaging tp:id="BillingDataImportRequest_PCK">
  <tp:ProcessingCapabilities tp:parse="true" tp:generate="true" />
- <tp:CompositeList>
- <tp:Composite tp:id="BillingDataImportRequest_CMP" tp:mimetype="multipart/related"
    tp:mimeparameters="type=text/xml">
  <tp:Constituent tp:idref="SoapEnv_SP" />
  <tp:Constituent tp:idref="BillingDataImportRequest_SP" />
  </tp:Composite>
  </tp:CompositeList>
  </tp:Packaging>
 - <!--
  An ebXML message with a SOAP Envelope plus a BillingDataImportConfirmation
  -->
- <tp:Packaging tp:id="BillingDataImportConfirmation_PCK">
  <tp:ProcessingCapabilities tp:parse="true" tp:generate="true" />
- <tp:CompositeList>
- <tp:Composite tp:id="BillingDataImportConfirmation_CMP" tp:mimetype="multipart/related"
    tp:mimeparameters="type=text/xml">
  <tp:Constituent tp:idref="SoapEnv_SP" />
  <tp:Constituent tp:idref="BillingDataImportConfirmation_SP" />
  </tp:Composite>
  </tp:CompositeList>
  </tp:Packaging>
 - <!--
  An ebXML message with a SOAP Envelope plus a BillingDataStatusRequest
  -->
- <tp:Packaging tp:id="BillingDataStatusRequest_PCK">
  <tp:ProcessingCapabilities tp:parse="true" tp:generate="true" />
- <tp:CompositeList>
- <tp:Composite tp:id="BillingDataStatusRequest_CMP" tp:mimetype="multipart/related"
    tp:mimeparameters="type=text/xml">
  <tp:Constituent tp:idref="SoapEnv_SP" />
  <tp:Constituent tp:idref="BillingDataStatusRequest_SP" />
  </tp:Composite>
  </tp:CompositeList>
  </tp:Packaging>
 - <!--
  An ebXML message with a SOAP Envelope plus a BillingDataStatusConfirmation
  -->
- <tp:Packaging tp:id="BillingDataStatusConfirmation_PCK">
  <tp:ProcessingCapabilities tp:parse="true" tp:generate="true" />
- <tp:CompositeList>
- <tp:Composite tp:id="BillingDataStatusConfirmation_CMP" tp:mimetype="multipart/related"
    tp:mimeparameters="type=text/xml">
  <tp:Constituent tp:idref="SoapEnv_SP" />
  <tp:Constituent tp:idref="BillingDataStatusConfirmation_SP" />
  </tp:Composite>
  </tp:CompositeList>
  </tp:Packaging>
  <tp:Comment xml:lang="en-US"> agreement between Com42Bill and Invoicing Party</tp:Comment>
  </tp:CollaborationProtocolAgreement>
```

C.3 CPA zwischen Com42Bill und einem Finanzdienstleister

Das CPA stellt einen Vertrag zwischen Com42Bill und einem Finanzdienstleister dar, anhand dessen die elektronischen Geschäftstransaktionen zwischen den beiden Parteien durchgeführt werden können.

```
<?xml version="1.0"?>
- <tp:CollaborationProtocolAgreement xmlns:tp="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd cpp-cpa-2_0.xsd"
    tp:cpaid="uri:http://www.Com42Bill.de/ebxml/cpa_Com42Bill_bank.xml" tp:version="1.0">
  <tp:Status tp:value="agreed" />
  <tp:Start>2002-10-01T07:21:00Z</tp:Start>
  <tp:End>2003-02-28T07:21:00Z</tp:End>
  - <!--
    Party info for Com42Bill
    -->
- <tp:PartyInfo tp:partyName="Com42Bill" tp:defaultMshChannelId="Com42BillChannelHTTP_CH"
    tp:defaultMshPackageId="BankTransferDataExportRequest_PCK">
  <tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">4242424242</tp:PartyId>
  <tp:PartyRef xlink:href="http://www.Com42Bill.de/index.html" />
  - <!--
  Roles for Com42Bill within the business processes
    -->
- <tp:CollaborationRole>
  <tp:ProcessSpecification tp:version="1.0" tp:name="Com42Bill_BusinessProcess01" xlink:type="simple"
    xlink:href="http://www.Com42Bill.de/ebxml/bpss_Com42Bill.xml" tp:uuid="urn:icann:Com42Bill.de:bpid:bp01" />
  <tp:Role tp:name="BankTransferDataSender_RLE" xlink:type="simple"
    xlink:href="http://www.Com42Bill.de/ebxml/bpss_Com42Bill.xml#BankTransferDataSender_RLE" />
- <tp:ServiceBinding>
  <tp:Service>urn:icann:Com42Bill.de:bpid:bp01</tp:Service>
- <tp:CanSend>
- <tp:ThisPartyActionBinding tp:id="BankTransferDataExportRequest_ABND"
    tp:action="BankTransferDataExportRequestAction" tp:packageId="BankTransferDataExportRequest_PCK">
  <tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="false" tp:isNonRepudiationReceiptRequired="false"
    tp:isConfidential="transient" tp:isAuthenticated="transient" tp:isTamperProof="transient"
    tp:isAuthorizationRequired="true" tp:timeToPerform="P1D" />
  <tp:ActionContext tp:binaryCollaboration="BankTransferDataExport"
    tp:businessTransactionActivity="BankTransferDataExport"
    tp:requestOrResponseAction="BankTransferDataExportRequestAction" />
  <tp:ChannelId>Com42BillChannelHTTP_CH</tp:ChannelId>
    </tp:ThisPartyActionBinding>
- <tp:CanReceive>
- <tp:ThisPartyActionBinding tp:id="BankTransferDataFailureConfirmation_ABND"
    tp:action="BankTransferDataFailureConfirmationAction"
    tp:packageId="BankTransferDataFailureConfirmation_PCK">
  <tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="false" tp:isNonRepudiationReceiptRequired="false"
    tp:isConfidential="transient" tp:isAuthenticated="transient" tp:isTamperProof="transient"
    tp:isAuthorizationRequired="true" tp:timeToPerform="P1D" />
  <tp:ActionContext tp:binaryCollaboration="BankTransferDataExport"
    tp:businessTransactionActivity="BankTransferDataExport"
    tp:requestOrResponseAction="BankTransferDataFailureConfirmationAction" />
  <tp:ChannelId>Com42BillChannelHTTP_CH</tp:ChannelId>
    </tp:ThisPartyActionBinding>
    </tp:CanReceive>
    </tp:CanSend>
    </tp:ServiceBinding>
    </tp:CollaborationRole>
  - <!--
  Security certificates used for transport
    -->
- <tp:Certificate tp:certId="ClientCertificate_CER">
- <ds:KeyInfo>
  <ds:KeyName>ClientCertificateKey?????</ds:KeyName>
```

```
<ds:KeyValue>ClientCertificateValue?????</ds:KeyValue>
    </ds:KeyInfo>
    </tp:Certificate>
- <tp:Certificate tp:certId="ServerCertificate_CER">
- <ds:KeyInfo>
<ds:KeyName>???</ds:KeyName>
<ds:KeyValue>???</ds:KeyValue>
    </ds:KeyInfo>
    </tp:Certificate>
- <tp:Certificate tp:certId="TransportCertificate_CER">
- <ds:KeyInfo>
<ds:KeyName>???</ds:KeyName>
<ds:KeyValue>???</ds:KeyValue>
    </ds:KeyInfo>
    </tp:Certificate>
- <tp:SecurityDetails tp:securityId="TransportSecurity_SD">
- <tp:TrustAnchors>
<tp:AnchorCertificateRef tp:certId="TransportCertificate_CER" />
    </tp:TrustAnchors>
    </tp:SecurityDetails>
  - <!--
  Transport information

    -->
- <tp:DeliveryChannel tp:channelId="Com42BillChannelHTTP_CH" tp:transportId="Com42BillTransport_TP"
      tp:docExchangeId="docExchangeStandard_DEX">
<tp:MessagingCharacteristics tp:syncReplyMode="mshSignalsOnly" tp:ackRequested="always"
      tp:ackSignatureRequested="never" tp:duplicateElimination="never" />
    </tp:DeliveryChannel>
- <tp:Transport tp:transportId="Com42BillTransport_TP">
- <tp:TransportSender>
<tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
<tp:AccessAuthentication>basic</tp:AccessAuthentication>
- <tp:TransportClientSecurity>
<tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
<tp:ClientCertificateRef tp:certId="ClientCertificate_CER" />
<tp:ServerSecurityDetailsRef tp:securityId="TransportSecurity_SD" />
    </tp:TransportClientSecurity>
    </tp:TransportSender>
- <tp:TransportReceiver>
<tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
<tp:AccessAuthentication>basic</tp:AccessAuthentication>
<tp:Endpoint tp:uri="http://127.0.0.1:7001/DKServlets/MsgReceiver" tp:type="allPurpose" />
- <tp:TransportServerSecurity>
<tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
<tp:ServerCertificateRef tp:certId="ServerCertificate_CER" />
<tp:ClientSecurityDetailsRef tp:securityId="TransportSecurity_SD" />
    </tp:TransportServerSecurity>
    </tp:TransportReceiver>
    </tp:Transport>
- <tp:DocExchange tp:docExchangeId="docExchangeStandard_DEX">
- <tp:ebXMLSenderBinding tp:version="1.0">
- <tp:ReliableMessaging>
<tp:Retries>3</tp:Retries>
<tp:RetryInterval>PT2H</tp:RetryInterval>
<tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
    </tp:ReliableMessaging>
<tp:PersistDuration>P1D</tp:PersistDuration>
    </tp:ebXMLSenderBinding>
- <tp:ebXMLReceiverBinding tp:version="1.0">
- <tp:ReliableMessaging>
<tp:Retries>3</tp:Retries>
<tp:RetryInterval>PT2H</tp:RetryInterval>
<tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
    </tp:ReliableMessaging>
<tp:PersistDuration>P1D</tp:PersistDuration>
    </tp:ebXMLReceiverBinding>
    </tp:DocExchange>
    </tp:PartyInfo>
  - <!--
```

```
    Party info for Bank
    -->
- <tp:PartyInfo tp:partyName="Bank" tp:defaultMshChannelId="BankChannelHTTP_CH"
      tp:defaultMshPackageId="BankTransferDataExportRequest_PCK">
  <tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">212121212121</tp:PartyId>
  <tp:PartyRef xlink:href="http://www.Com42Bill.de/bank/index.html" />
   - <!--
   Roles for Bank within the business processes
    -->
- <tp:CollaborationRole>
  <tp:ProcessSpecification tp:version="1.0" tp:name="Com42Bill_BusinessProcess01" xlink:type="simple"
      xlink:href="http://www.Com42Bill.de/ebxml/bpss_Com42Bill.xml" tp:uuid="urn:icann:Com42Bill.de:bpid:bp01" />
  <tp:Role tp:name="BankTransferDataReceiver_RLE" xlink:type="simple"
      xlink:href="http://www.Com42Bill.de/ebxml/bpss_Com42Bill.xml#BankTransferDataReceiver_RLE" />
- <tp:ServiceBinding>
  <tp:Service>urn:icann:Com42Bill.de:bpid:bp01</tp:Service>
- <tp:CanReceive>
- <tp:ThisPartyActionBinding tp:id="BankTransferDataExportRequest_bank_ABND"
      tp:action="BankTransferDataExportRequestAction" tp:packageId="BankTransferDataExportRequest_PCK">
  <tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="false" tp:isNonRepudiationReceiptRequired="false"
      tp:isConfidential="transient" tp:isAuthenticated="transient" tp:isTamperProof="transient"
      tp:isAuthorizationRequired="true" tp:timeToPerform="P1D" />
  <tp:ActionContext tp:binaryCollaboration="BankTransferDataExport"
      tp:businessTransactionActivity="BankTransferDataExport"
      tp:requestOrResponseAction="BankTransferDataExportRequestAction" />
  <tp:ChannelId>BankChannelHTTP_CH</tp:ChannelId>
    </tp:ThisPartyActionBinding>
- <tp:CanSend>
- <tp:ThisPartyActionBinding tp:id="BankTransferDataFailureConfirmation_bank_ABND"
      tp:action="BankTransferDataFailureConfirmationAction"
      tp:packageId="BankTransferDataFailureConfirmation_PCK">
  <tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="false" tp:isNonRepudiationReceiptRequired="false"
      tp:isConfidential="transient" tp:isAuthenticated="transient" tp:isTamperProof="transient"
      tp:isAuthorizationRequired="true" tp:timeToPerform="P1D" />
  <tp:ActionContext tp:binaryCollaboration="BankTransferDataExport"
      tp:businessTransactionActivity="BankTransferDataExport"
      tp:requestOrResponseAction="BankTransferDataFailureConfirmationAction" />
  <tp:ChannelId>BankChannelHTTP_CH</tp:ChannelId>
    </tp:ThisPartyActionBinding>
    </tp:CanSend>
    </tp:CanReceive>
    </tp:ServiceBinding>
    </tp:CollaborationRole>
   - <!--
   Security certificates used for transport
    -->
- <tp:Certificate tp:certId="ClientCertificate_bank_CER">
- <ds:KeyInfo>
  <ds:KeyName>ClientCertificateKey?????</ds:KeyName>
  <ds:KeyValue>ClientCertificateValue?????</ds:KeyValue>
    </ds:KeyInfo>
    </tp:Certificate>
- <tp:Certificate tp:certId="ServerCertificate_bank_CER">
- <ds:KeyInfo>
  <ds:KeyName>???</ds:KeyName>
  <ds:KeyValue>???</ds:KeyValue>
    </ds:KeyInfo>
    </tp:Certificate>
- <tp:Certificate tp:certId="TransportCertificate_bank_CER">
- <ds:KeyInfo>
  <ds:KeyName>???</ds:KeyName>
  <ds:KeyValue>???</ds:KeyValue>
    </ds:KeyInfo>
    </tp:Certificate>
- <tp:SecurityDetails tp:securityId="TransportSecurity_bank_SD">
- <tp:TrustAnchors>
  <tp:AnchorCertificateRef tp:certId="TransportCertificate_CER" />
    </tp:TrustAnchors>
    </tp:SecurityDetails>
```

```
- <!--
  Transport information
  -->
- <tp:DeliveryChannel tp:channelId="BankChannelHTTP_CH" tp:transportId="BankTransport_TP"
    tp:docExchangeId="docExchangeStandard_DEX">
  <tp:MessagingCharacteris tics tp:syncReplyMode ="mshSignalsOnly" tp:ackRequested="always"
    tp:ackSignatureRequested ="never" tp:duplicateElimination="never" />
    </tp:DeliveryChannel>
- <tp:Transport tp:transportId="BankTransport_TP">
- <tp:TransportSender>
  <tp:TransportProtoc ol tp:version="1.1">HTTP</tp:TransportProtocol>
  <tp:AccessAuthentication>basic</tp:AccessAuthentication>
- <tp:TransportClientSecurity>
  <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
  <tp:ClientCertificateRef tp:certId="ClientCertificate_CER" />
  <tp:ServerSecurityDetailsRef tp:securityId="TransportSecurity_SD" />
    </tp:TransportClientSecurity>
    </tp:TransportSender>
- <tp:TransportReceiver>
  <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
  <tp:AccessAuthentication>basic</tp:AccessAuthentication>
  <tp:Endpoint tp:uri="http://127.0.0.1:7001/DKServlets/BankServlet" tp:type="allPurpose" />
- <tp:TransportServerSecurity>
  <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
  <tp:ServerCertificateRef tp:certId="ServerCertificate_CER" />
  <tp:ClientSecurityDetailsRef tp:securityI d="TransportSecurity_SD" />
    </tp:TransportServerSecurity>
    </tp:TransportReceiver>
    </tp:Transport>
- <tp:DocExchange tp:docExchangeId="docExchangeStandard_bank_DEX">
- <tp:ebXMLSenderBinding tp:version ="1.0">
- <tp:ReliableMessaging>
  <tp:Retries >3</tp:Retries >
  <tp:RetryInterval >PT2H</tp:RetryInterval >
  <tp:MessageOrderSemantics >Guaranteed</tp:MessageOrderSemantics >
    </tp:ReliableMessaging>
  <tp:PersistDuration>P1D</tp:PersistDuration>
    </tp:ebXMLSenderBinding>
- <tp:ebXMLReceiverBinding tp:version="1.0">
- <tp:ReliableMessaging>
  <tp:Retries >3</tp:Retries >
  <tp:RetryInterval >PT2H</tp:RetryInterval >
  <tp:MessageOrderSemantics >Guaranteed</tp:MessageOrderSemantics >
    </tp:ReliableMessaging>
  <tp:PersistDuration>P1D</tp:PersistDuration>
    </tp:ebXMLReceiverBinding >
    </tp:DocExchange >
    </tp:PartyInfo >
  - <!--
  Business Documents being exchanged within the business processes
    -->
  - <!--
   SimplePart corresponding to the SOAP Envelope
    -->
- <tp:SimplePart tp:id="SoapEnv_SP" tp:mimetype="text/xml">
  <tp:NamespaceSupported tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-
    2_0.xsd">http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-
    2_0.xsd</tp:NamespaceSupported>
    </tp:SimplePart>
  - <!--
   SimplePart corresponding to a BankTransferDataExportRequest
    -->
- <tp:SimplePart tp:id="BankTransferDataExportRequest_SP" tp:mimetype="text/xml">
  <tp:NamespaceSupported
    tp:location="www.Com42Bill.de/ebxml/BankTransferDataExportRequest.xsd">www.Com42Bill.de/ebxml/BankTra
    nsferDataExportRequest.xsd</tp:NamespaceSupported>
    </tp:SimplePart >
  - <!--
```

```
    SimplePart corresponding to a BankTransferDataFailureConfirmation
    -->
- <tp:SimplePart tp:id="BankTransferDataFailureConfirmation_SP" tp:mimetype="text/xml">
  <tp:NamespaceSupported
      tp:location="www.Com42Bill.de/ebxml/BankTransferDataFailureConfirmationt.xsd">www.Com42Bill.de/ebxml/BankTransferDataFailureConfirmationt.xsd</tp:NamespaceSupported>
    </tp:SimplePart>
  - <!--
    An ebXML message with a SOAP Envelope plus a BankTransferDataExportRequest
    -->
- <tp:Packaging tp:id="BankTransferDataExportRequest_PCK">
  <tp:ProcessingCapabilities tp:parse="true" tp:generate="true" />
- <tp:CompositeList>
- <tp:Composite tp:id="BankTransferDataExportRequest_CMP" tp:mimetype="multipart/related"
      tp:mimeparameters="type=text/xml">
  <tp:Constituent tp:idref="SoapEnv_SP" />
  <tp:Constituent tp:idref="BankTransferDataExportRequest_SP" />
    </tp:Composite>
    </tp:CompositeList>
    </tp:Packaging>
  - <!--
    An ebXML message with a SOAP Envelope plus a BankTransferDataFailureConfirmation
    -->
- <tp:Packaging tp:id="BankTransferDataFailureConfirmation_PCK">
  <tp:ProcessingCapabilities tp:parse="true" tp:generate="true" />
- <tp:CompositeList>
- <tp:Composite tp:id="BankTransferDataFailureConfirmation_CMP" tp:mimetype="multipart/related"
      tp:mimeparameters="type=text/xml">
  <tp:Constituent tp:idref="SoapEnv_SP" />
  <tp:Constituent tp:idref="BankTransferDataFailureConfirmation_SP" />
    </tp:Composite>
    </tp:CompositeList>
    </tp:Packaging>
  <tp:Comment xml:lang="en-US">agreement between Com42Bill and financial service provider</tp:Comment>
    </tp:CollaborationProtocolAgreement>
```

C.4 Business Process Specification Schema (BPSS)

In dem BPSS werden die Geschäftsprozesse, die Com42Bill durchführen kann, abgebildet.

```
    <?xml version="1.0" encoding="UTF-8" ?>
  - <!--
    edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by Krosty (Krosty Enterprises)
    -->
- <ProcessSpecification xmlns="http://www.ebxml.org/BusinessProcess"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.ebxml.org/BusinessProcess ebBPSS1.04.xsd"
      name="Com42Bill_BusinessProcess01" uuid="urn:icann:Com42Bill.de:bpid:bp01" version="1.0">
  <Documentation>doc</Documentation>
  <BusinessDocument name="BillingDataImportRequest" nameID="BillingDataImportRequest_BD"
      specificationLocation="www.Com42Bill.de/ebxml/BillingDataImportRequest.xsd" />
  <BusinessDocument name="BillingDataImportConfirmation" nameID="BillingDataImportConfirmation_BD"
      specificationLocation="www.Com42Bill.de/ebxml/BillingDataImportConfirmation.xsd" />
  <BusinessDocument name="BillingDataStatusRequest" nameID="BillingDataStatusRequest_BD"
      specificationLocation="www.Com42Bill.de/ebxml/BillingDataStatusRequest.xsd" />
  <BusinessDocument name="BankTransferDataExportRequest" nameID="BankTransferDataExportRequest_BD"
      specificationLocation="www.Com42Bill.de/ebxml/BankTransferDataExportRequest.xsd" />
  <BusinessDocument name="BankTransferDataFailureConfirmation" nameID="BankTransferDataFailureConfirmation_BD"
      specificationLocation="www.Com42Bill.de/ebxml/BankTransferDataFailureConfirmation.xsd" />
  <BusinessDocument name="BillingDataStatusConfirmation" nameID="BillingDataStatusConfirmation_BD"
      specificationLocation="www.Com42Bill.de/ebxml/BillingDataStatusConfirmation.xsd" />
- <BusinessTransaction name="BillingDataImport" nameID="BillingDataImport_BT">
```

```
- <RequestingBusinessActivity name="BillingDataImportRequestAction" nameID="BillingDataImportRequestAction_BA"
      isAuthorizationRequired="true" isIntelligibleCheckRequired="true" isNonRepudiationReceiptRequired="true"
      isNonRepudiationRequired="true" timeToAcknowledgeReceipt="P0Y0M0DT2H0M0S">
  <DocumentEnvelope businessDocument="BillingDataImportRequest"
      businessDocumentIDRef="BillingDataImportRequest_BD" isAuthenticated="persistent" isConfidential="transient"
      isTamperProof="persistent" />
    </RequestingBusinessActivity>
- <RespondingBusinessActivity name="BillingDataImportConfirmationAction"
      nameID="BillingDataImportConfirmationAction_BA" isAuthorizationRequired="true" isIntelligibleCheckRequired="true"
      isNonRepudiationRequired="true" timeToAcknowledgeReceipt="P0Y0M0DT2H0M0S">
  <DocumentEnvelope businessDocument="BillingDataImportConfirmation"
      businessDocumentIDRef="BillingDataImportConfirmation_BD" isAuthenticated="persistent" isConfidential="transient"
      isPositiveResponse="true" isTamperProof="persistent" />
    </RespondingBusinessActivity>
    </BusinessTransaction>
- <BusinessTransaction name="BillingDataStatus" nameID="BillingDataStatus_BT">
- <RequestingBusinessActivity name="BillingDataStatusRequestAction" nameID="BillingDataStatusRequestAction_BA"
      isAuthorizationRequired="true" isIntelligibleCheckRequired="true" isNonRepudiationReceiptRequired="true"
      isNonRepudiationRequired="true" timeToAcknowledgeReceipt="P0Y0M0DT2H0M0S">
  <DocumentEnvelope businessDocument="BillingDataStatusRequest"
      businessDocumentIDRef="BillingDataStatusRequest_BD" isAuthenticated="persistent" isConfidential="transient"
      isTamperProof="persistent" />
    </RequestingBusinessActivity>
- <RespondingBusinessActivity name="BillingDataStatusConfirmationAction" nameID="BillingDataStatusConfirmationAction"
      isAuthorizationRequired="true" isIntelligibleCheckRequired="true" isNonRepudiationRequired="true"
      timeToAcknowledgeReceipt="P0Y0M0DT2H0M0S">
  <DocumentEnvelope businessDocument="BillingDataStatusConfirmation"
      businessDocumentIDRef="BillingDataStatusConfirmation_BD" isAuthenticated="persistent" isConfidential="transient"
      isPositiveResponse="true" isTamperProof="persistent" />
    </RespondingBusinessActivity>
    </BusinessTransaction>
- <BusinessTransaction name="BankTransferDataExport" nameID="BankTransferDataExport_BT">
- <RequestingBusinessActivity name="BankTransferDataExportRequestAction"
      nameID="BankTransferDataExportRequestAction_BA" isAuthorizationRequired="true" isIntelligibleCheckRequired="true"
      isNonRepudiationReceiptRequired="true" isNonRepudiationRequired="true"
      timeToAcknowledgeReceipt="P0Y0M0DT2H0M0S">
  <DocumentEnvelope businessDocument="BankTransferDataExportRequest"
      businessDocumentIDRef="BankTransferDataExportRequest_BD" isAuthenticated="persistent" isConfidential="transient"
      isTamperProof="persistent" />
    </RequestingBusinessActivity>
- <RespondingBusinessActivity name="BankTransferDataFailureConfirmationAction"
      nameID="BankTransferDataFailureConfirmationAction_BA" isAuthorizationRequired="true"
      isIntelligibleCheckRequired="true" isNonRepudiationRequired="true" timeToAcknowledgeReceipt="P0Y0M0DT2H0M0S">
  <DocumentEnvelope businessDocument="BankTransferDataFailureConfirmation"
      businessDocumentIDRef="BankTransferDataFailureConfirmation_BD" isAuthenticated="persistent"
      isConfidential="transient" isPositiveResponse="true" isTamperProof="persistent" />
    </RespondingBusinessActivity>
    </BusinessTransaction>
- <BinaryCollaboration name="BillingDataImport" nameID="BillingDataImport_BC" initiatingRole="BillingDataSender_RLE">
  <Role name="BillingDataSender" nameID="BillingDataSender_RLE" />
  <Role name="BillingDataReceiver" nameID="BillingDataReceiver_RLE" />
  <Start toBusinessState="BillingDataImportState" />
  <BusinessTransactionActivity name="BillingDataImport" nameID="BillingDataImport_BTA"
      businessTransaction="BillingDataImport" businessTransactionIDRef="BillingDataImport_BT"
      fromRole="BillingDataSender" fromRoleIDRef="BillingDataSender_RLE" toRole="BillingDataReceiver"
      toRoleIDRef="BillingDataReceiver_RLE" isLegallyBinding="true" timeToPerform="P0Y0M0DT24H0M0S"
      isConcurrent="false" />
  <Success fromBusinessState="BillingDataImportState" conditionGuard="Success" />
  <Failure fromBusinessState="BillingDataImportState" conditionGuard="BusinessFailure" />
  <Transition fromBusinessState="BillingDataImportState" toBusinessState="BillingDataImportState" />
    </BinaryCollaboration>
- <BinaryCollaboration name="BillingDataStatus" nameID="BillingDataStatus_BC"
      initiatingRole="BillingDataStatusSender_RLE">
  <Role name="BillingDataStatusSender" nameID="BillingDataStatusSender_RLE" />
  <Role name="BillingDataStatusReceiver" nameID="BillingDataStatusReceiver_RLE" />
  <Start toBusinessState="BillingDataStatusState" />
  <BusinessTransactionActivity name="BillingDataStatus" nameID="BillingDataStatusState_BTA"
      businessTransaction="BillingDataStatus" businessTransactionIDRef="BillingDataStatus_BT"
      fromRole="BillingDataStatusSender" fromRoleIDRef="BillingDataStatusSender_RLE"
      toRole="BillingDataStatusReceiver" toRoleIDRef="BillingDataStatusReceiver_RLE" isLegallyBinding="true"
      timeToPerform="P0Y0M0DT24H0M0S" isConcurrent="false" />
```

```
<Success fromBusinessState="BillingDataStatusState" conditionGuard="Success" />
<Failure fromBusinessState="BillingDataStatusState" conditionGuard="BusinessFailure" />
<Transition fromBusinessState="BillingDataStatusState" toBusinessState="BillingDataStatusState" />
   </BinaryCollaboration>
- <BinaryCollaboration name="BankTransferDataExport" nameID="BankTransferDataExport_BC"
     initiatingRole="BankTransferDataSender_RLE">
<Role name="BankTransferSender" nameID="BankTransferDataSender_RLE" />
<Role name="BankTransferReceiver" nameID="BankTransferDataReceiver_RLE" />
<Start toBusinessState="BankTransferDataExportState" />
<BusinessTransactionActivity name="BankTransferDataExport" nameID="BankTransferDataExport_BTA"
     businessTransaction="BankTransferDataExport" businessTransactionIDRef="BankTransferDataExport_BT"
     fromRole="BankTransferDataSender" fromRoleIDRef="BankTransferDataSender_RLE"
     toRole="BankTransferDataReceiver" toRoleIDRef="BankTransferDataReceiver_RLE" isLegallyBinding="true"
     timeToPerform="P0Y0M0DT24H0M0S" isConcurrent="false" />
<Success fromBusinessState="BankTransferDataExportState" conditionGuard="Success" />
<Failure fromBusinessState="BankTransferDataExportState" conditionGuard="BusinessFailure" />
<Transition fromBusinessState="BankTransferDataExportState" toBusinessState="BankTransferDataExportState" />
   </BinaryCollaboration>
   </ProcessSpecification>
```

C.5 XML-Schema für den Import von Rechnungen/Stornos

Im Folgenden wird die Formatvorlage für den Austausch von Rechnungen bzw. Stornos dargestellt.

```
   <?xml version="1.0" encoding="UTF-8" ?>
 - <!--
   Attachment: Rechnungsdaten

   -->
- <schema targetNamespace="http://www.Com42Bill.de/ebXML/billingData"
     xmlns="http://www.w3.org/2001/XMLSchema" xmlns:bd="http://www.Com42Bill.de/ebXML/billingData">
 - <!--
   Invoicedata kapselt Rechnungsdaten unterteilt in drei Kategorien: a) ordinaere Rechnungen b) Stornos c)
     Updates

   -->
- <element name="invoiceDataTypes" minOccurs="1" maxOccurs="1">
- <complexType>
- <sequence>
 - <!--
   Der Invoicecontainer kapselt mehrere Rechnungen fuer verschiedene Rechnungsempfaenger

   -->
- <element name="invoiceContainer" minOccurs="0" maxOccurs="1">
- <complexType>
- <sequence>
 - <!--
   Daten einer Rechnung

   -->
- <element name="invoice" minOccurs="1" maxOccurs="unbounded">
- <complexType>
- <sequence>
 - <!--
   Bezahlungsart: Komplettzahlung oder Ratenzahlung

   -->
<element name="paymentType" type="string" minOccurs="1" maxOccurs="1" />
 - <!--
   Kundennummer der Rechnungsempaengers

   -->
<element name="invoiceRecipientNr" type="string" minOccurs="1" maxOccurs="1" />
 - <!--
   Kundennummer des Rechnungsstellers

   -->
<element name="invoicingPartyNr" type="string" minOccurs="1" maxOccurs="1" />
 - <!--
   eindeutige Rechnungsnummer
```

```
    -->
  <element name="invoiceNr" type="string" minOccurs ="1" maxOccurs ="1" />
    - <!--
    Ausstellungsdatum der Rechnung
    -->
  <element name="creationDate" type="string" minOccurs ="1" maxOccurs ="1" />
    - <!--
    Fälligkeitsdatum der Rechnung
    -->
  <element name="requestedPaymentDate" type ="string" minOccurs ="1" maxOccurs ="1" />
    - <!--
    Rechnungsbetrag
    -->
  <element name="amount" type ="string" minOccurs ="1" maxOccurs ="1" />
    - <!--
    Rechnungspositionen (optional)
    -->
- <element name="invoiceLineItemContainer" minOccurs ="0" maxOccurs ="1">
- <complexType>
- <sequence>
    - <!--
    Rechnungsposition
    -->
- <element name="invoiceLineItem" type="string" minOccurs ="0" maxOccurs ="unbounded">
- <complexType>
- <sequence>
    - <!--
    Beschreibung einer Rechnungsposition
    -->
  <element name="description" type="string" minOccurs ="1" maxOccurs ="1" />
    - <!--
    Betrag einer Rechnungsposition
    -->
  <element name="amount" type="string" minOccurs ="1" maxOccurs ="1" />
      </sequence>
      </complexType>
      </element>
      </sequence>
      </complexType>
      </element>
    - <!--
    Daten zur Ratenzahlung (noetig fuer entsprechenden Paymenttype)
    -->
- <element name="instalmentData" minOccurs ="0" maxOccurs ="1">
- <complexType>
- <sequence>
    - <!--
    Faelligkeit der ersten Rate
    -->
  <element name="firstRequestedPaymentDate" type="string" minOccurs ="1" maxOccurs ="1" />
    - <!--
    Zahlungsintervall in Tagen
    -->
  <element name="paymentInterval" type="string" minOccurs ="1" maxOccurs ="1" />
    - <!--
    Anzahl der Zahlungsintervalle
    -->
  <element name="numberOfIntervals" type="string" minOccurs ="1" maxOccurs ="1" />
      </sequence>
      </complexType>
      </element>
      </sequence>
      </complexType>
      </element>
      </sequence>
      </complexType>
```

```
    </element>
  - <!--
    Der Invoicecancellationcontainer kapselt mehrere Stornos fuer verschiedene Rechnungen
    -->
- <element name="invoiceCancellationContainer" minOccurs="0" maxOccurs="1">
- <complexType>
- <sequence>
  - <!--
    Daten eines Stornos
    -->
- <element name="invoiceCancellation" minOccurs="1" maxOccurs="unbounded">
- <complexType>
- <sequence>
  - <!--
    Kundennummer der Rechnungsempaengers
    -->
  <element name="invoiceRecipientNr" type="string" minOccurs="1" maxOccurs="1" />
  - <!--
    Kundennummer des Rechnungsstellers
    -->
  <element name="invoicingPartyNr" type="string" minOccurs="1" maxOccurs="1" />
  - <!--
    eindeutige Rechnungsnummer
    -->
  <element name="invoiceNr" type="string" minOccurs="1" maxOccurs="1" />
    </sequence>
    </complexType>
    </element>
    </sequence>
    </complexType>
    </element>
    </sequence>
    </complexType>
    </element>
    </schema>
```

C.6 XML-Schema für den Import von Rechnungsstati

Im Folgenden wird die Formatvorlage für den Austausch von Statusinformationen zu Rechnungen dargestellt.

```
    <?xml version="1.0" encoding="UTF-8" ?>
  - <!--
    Attachment: Status der Bezahlung der Rechnungen
    -->
- <schema targetNamespace="http://www.Com42Bill.de/ebXML/billingDataStatus"
    xmlns="http://www.w3.org/2001/XMLSchema" xmlns:bd="http://www.Com42Bill.de/ebXML/billingDataStatus">
  - <!--
    Der InvoiceStatuscontainer kapselt mehrere Bezahlungsstatusinformationen fuer Rechnungen verschiedener
      Rechnungsempfaenger
    -->
- <element name="invoiceStatusContainer" minOccurs="1" maxOccurs="1">
- <complexType>
- <sequence>
  - <!--
    Statusdaten einer Rechnung
    -->
- <element name="invoiceStatus" minOccurs="1" maxOccurs="unbounded">
- <complexType>
- <sequence>
  - <!--
    Kundennummer des Rechnungsempfaengers
    -->
```

```
<element name="invoiceRecipientNr" type="string" minOccurs ="1" maxOccurs ="1" />
 - <!--
 Kundennummer des Rechnungsstellers
   -->
<element name="invoicingPartyNr" type="string" minOccurs ="1" maxOccurs ="1" />
 - <!--
 eindeutige Rechnungsnummer
   -->
<element name="invoiceNr" type="string" minOccurs ="1" maxOccurs ="1" />
 - <!--
 Ratennr.
   -->
<element name="instalmentNr" type="string" minOccurs ="0" maxOccurs ="1" />
   </sequence>
   </complexType>
   </element>
   </sequence>
   </complexType>
   </element>
   </schema>
```

C.7 XML-Schema für die Rückmeldung eines Rechnungs- bzw. Stornoimports

Im Folgenden wird die Formatvorlage für die Rückmeldung zu einem Rechnungs- bzw. Stornoimport dargestellt.

```
   <?xml version="1.0" encoding="UTF-8" ?>
- <schema targetNamespace="http://www.Com42Bill.de/ebXML/confirmationData"
     xmlns ="http://www.w3.org/2001/XMLSchema" xmlns:cd="http://www.Com42Bill.de/ebXML/confirmationData">
- <element name="billingDataConfirmation" minOccurs ="1" maxOccurs ="1">
- <complexType>
- <sequence>
   - <!--
   Allgemeine Statusmeldung
     -->
- <element name="confirmationInvoiceMessage" minOccurs ="0" maxOccurs ="1">
- <complexType>
- <sequence>
  <element name="status" type="string" minOccurs ="1" maxOccurs ="1" />
   </sequence>
   </complexType>
   </element>
   - <!--
   Rueckmeldung ueber die interne Verabeitung der verschiedenen Rechnungsdaten
     -->
- <element name="confirmationInvoiceDataTypes" minOccurs ="1" maxOccurs ="1">
- <complexType>
- <sequence>
   - <!--
   Der confirmationInvoiceStatusContainer kapselt die Verarbeitungsstati der Rechnungen
     -->
- <element name="confirmationInvoiceContainer" minOccurs ="0" maxOccurs ="1">
- <complexType>
- <sequence>
   - <!--
   Statusdaten einer Rechnung
     -->
- <element name="confirmationInvoice" minOccurs ="1" maxOccurs ="unbounded">
- <complexType>
- <sequence>
   - <!--
   Kundennummer der Rechnungsempaengers
     -->
```

```
<element name="invoiceRecipientNr" type="string" minOccurs ="1" maxOccurs ="1" />
 - <!--
 Kundennummer des Rechnungsstellers
   -->
<element name="invoicingPartyNr" type="string" minOccurs ="1" maxOccurs ="1" />
 - <!--
 eindeutige Rechnungsnummer
   -->
<element name="invoiceNr" type="string" minOccurs ="1" maxOccurs ="1" />
 - <!--
 Status der internen Datenverarbeitung
   -->
<element name="status" type="string" minOccurs ="1" maxOccurs ="1" />
   </sequence>
   </complexType>
   </element>
   </sequence>
   </complexType>
   </element>
 - <!--
 Der confirmationInvoiceCancellationStatusContainer kapselt die Verarbeitungsstati der Stornos
   -->
- <element name="confirmationInvoiceCancellationContainer" minOccurs ="0" maxOccurs ="1">
- <complexType>
- <sequence>
 - <!--
 Statusdaten eines Stornos
   -->
- <element name="confirmationInvoiceCancellation" minOccurs ="1" maxOccurs ="unbounded">
- <complexType>
- <sequence>
 - <!--
 Kundennummer der Rechnungsempaengers
   -->
<element name="invoiceRecipientNr" type="string" minOccurs ="1" maxOccurs ="1" />
 - <!--
 Kundennummer des Rechnungsstellers
   -->
<element name="invoicingPartyNr" type="string" minOccurs ="1" maxOccurs ="1" />
 - <!--
 eindeutige Rechnungsnummer
   -->
<element name="invoiceNr" type="string" minOccurs ="1" maxOccurs ="1" />
 - <!--
 Status der internen Datenverarbeitung
   -->
<element name="status" type="string" minOccurs ="1" maxOccurs ="1" />
   </sequence>
   </complexType>
   </element>
   </sequence>
   </complexType>
   </element>
   </sequence>
   </complexType>
   </element>
   </sequence>
   </complexType>
   </element>
   </schema>
```

C.8 XML-Schema für die Rückmeldung eines Rechnungsstatusimports

Im Folgenden wird die Formatvorlage für die Rückmeldung zu einem Rechnungsstatusimport dargestellt.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
 edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by Krosty (Krosty Enterprises)
  -->
- <!--
Attachment: Status der Bezahlung der Rechnungen
  -->
- <schema targetNamespace="http://www.Com42Bill.de/ebXML/billingDataStatusConfirmation"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:bd="http://www.Com42Bill.de/ebXML/billingDataStatusConfirmation">
- <element name="billingDataStatusConfirmation" minOccurs="1" maxOccurs="1">
- <complexType>
- <sequence>
  - <!--
  Allgemeine Statusmeldung
    -->
- <element name="confirmationInvoiceStatusMessage" minOccurs="0" maxOccurs="1">
- <complexType>
- <sequence>
  <element name="status" type="string" minOccurs="1" maxOccurs="1" />
    </sequence>
    </complexType>
    </element>
  - <!--
  Der ConfirmationInvoiceStatuscontainer enthaelt die Stati zu der Verarbeitung der Daten
    -->
- <element name="confirmationInvoiceStatusContainer" minOccurs="1" maxOccurs="1">
- <complexType>
- <sequence>
  - <!--
  Statusdaten einer Rechnung
    -->
- <element name="confirmationInvoiceStatus" minOccurs="1" maxOccurs="unbounded">
- <complexType>
- <sequence>
  - <!--
  Kundennummer des Rechnungsempfaengers
    -->
  <element name="invoiceRecipientNr" type="string" minOccurs="1" maxOccurs="1" />
  - <!--
  Kundennummer des Rechnungsstellers
    -->
  <element name="invoicingPartyNr" type="string" minOccurs="1" maxOccurs="1" />
  - <!--
  eindeutige Rechnungsnummer
    -->
  <element name="invoiceNr" type="string" minOccurs="1" maxOccurs="1" />
  - <!--
  Status der internen Verarbeitung
    -->
  <element name="status" type="string" minOccurs="1" maxOccurs="1" />
    </sequence>
    </complexType>
    </element>
    </sequence>
    </complexType>
    </element>
    </sequence>
    </complexType>
```

```
</element>
</schema>
```

C.9 XML-Schema für eine Überweisung

Im Folgenden wird die Formatvorlage für eine Überweisung dargestellt.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
Attachment: Überweisungdaten
-->
- <schema targetNamespace="http://www.Com42Bill.de/ebXML/bankTransferData"
xmlns="http://www.w3.org/2001/XMLSchema" xmlns:bd="http://www.Com42Bill.de/ebXML/bankTransferData">
- <!--
Das Element "bankTransferData" beschreibt einen Ueberweisungsauftrag
-->
- <element name="bankTransferData" minOccurs="1" maxOccurs="1">
- <complexType>
- <sequence>
- <!--
Die Elemente, die mit dem Praefix "orderingParty" beginnen beschreiben, die Bankverbing des
Ueberweisungsauftraggebers und die die mit dem Praefix "recipient" beginnen beschreiben die
Bankverbindung der Beguenstigten.
-->
<element name="orderingParty_name" type="string" minOccurs="1" maxOccurs="1" />
<element name="orderingParty_accountNr" type="string" minOccurs="1" maxOccurs="1" />
<element name="orderingParty_bankCode" type="string" minOccurs="1" maxOccurs="1" />
<element name="orderingParty_bank" type="string" minOccurs="1" maxOccurs="1" />
<element name="recipient_name" type="string" minOccurs="1" maxOccurs="1" />
<element name="recipient_accountNr" type="string" minOccurs="1" maxOccurs="1" />
<element name="recipient_bankCode" type="string" minOccurs="1" maxOccurs="1" />
<element name="recipient_bank" type="string" minOccurs="1" maxOccurs="1" />
- <!--
Das Element "reasonForPayment" gibt den Verwendungszweck an
-->
<element name="reasonForPayment" type="string" minOccurs="1" maxOccurs="1" />
- <!--
Das Element "amount" gibt den Betrag an
-->
<element name="amount" type="string" minOccurs="1" maxOccurs="1" />
- <!--
eindeutige ID der Transaktion
-->
<element name="paymentTransactionId" type="string" minOccurs="1" maxOccurs="1" />
</sequence>
</complexType>
</element>
</schema>
```

C.10 XML-Schema für die Rückmeldung des Finanzdienstleisters

Im Folgenden wird die Formatvorlage für die Rückmeldung zu einer Überweisung dargestellt.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
Attachment: Status einer Ueberweisung (von Bank)
-->
```

```
- <schema targetNamespace="http://www.Com42Bill.de/ebXML/bankTransferDataStatus"
      xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:bd="http://www.Com42Bill.de/ebXML/bankTransferDataStatus">
   - <!--
    Das Element "bankTransferDataStatus" beschreibt den Status eines Ueberweisungsauftrags
    -->
- <element name="bankTransferDataStatus" minOccurs="1" maxOccurs="1">
- <complexType>
- <sequence>
   - <!--
   Statusmeldung
    -->
  <element name="status" type="string" minOccurs="1" maxOccurs="1" />
   - <!--
   eindeutige ID der Transaktion
    -->
  <element name="paymentTransactionId" type="string" minOccurs="1" maxOccurs="1" />
    </sequence>
    </complexType>
    </element>
    </schema>
```

C.11 Beispiel für eine ebXML Message (Rechnungsimport)

Im Folgenden wird am Beispiel eines Rechnungsimports eine ebXML Message dargestellt. Diese setzt sich aus einem SOAP Envelope und einem XML-Attachment zusammen. Das XML-Attachment basiert auf dem XML-Schema aus Anhang C.5.

C.11.1: SOAP Envelope

```
    <?xml version="1.0" encoding="UTF-8" ?>
-                <soap-env:Envelope                xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"                xmlns:eb="http://www.oasis-
    open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd" xmlns:xlink="http://www.w3.org/1999/xlink">
-    <soap-env:Header    xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-
    2_0.xsd http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd">
  <eb:AckRequested eb:signed="false" eb:version="2.0" soap-env:mustUnderstand="true" />
- <eb:MessageHeader eb:id="msgHeader" eb:version="2.0" soap-env:mustUnderstand="true">
- <eb:From>
  <eb:PartyId eb:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">InvoicingParty</eb:PartyId>
  <eb:Role>BillingDataSender_RLE</eb:Role>
    </eb:From>
- <eb:To>
  <eb:PartyId eb:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">Com42Bill</eb:PartyId>
  <eb:Role>BillingDataReceiver_RLE</eb:Role>
    </eb:To>
  <eb:CPAId>http://www.Com42Bill.de/ebXML/cpa_Com42Bill_invoicingParty.xml</eb:CPAId>
  <eb:ConversationId>Conversation</eb:ConversationId>
  <eb:Service eb:type="string">urn:icann:Com42Bill.de:bpid:bp01</eb:Service>
  <eb:Action>BillingDataImportRequestAction</eb:Action>
- <eb:MessageData>
  <eb:MessageId>BillingDataImportRequest</eb:MessageId>
  <eb:Timestamp>20030224_220628</eb:Timestamp>
    </eb:MessageData>
    </eb:MessageHeader>
    </soap-env:Header>
- <soap-env:Body xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
      http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd">
- <eb:Manifest eb:id="manifest01" eb:version="2.0">
- <eb:Reference eb:id="billingData01" xlink:href="cid:invoice_01">
  <eb:Schema eb:location="http://www.Com42Bill.de/ebXML/billingData.xsd" eb:version="2.0" />
  <eb:Description xml:lang="en-US">ebXML Message</eb:Description>
    </eb:Reference>
    </eb:Manifest>
    </soap-env:Body>
    </soap-env:Envelope>
```

C.11.2: Attachment

```
<?xml version="1.0" encoding="UTF-8" ?>
- <invoice-data-types valid="true">
  <invoice-cancellation-container invoice-cancellation-count="0" valid="true" />
- <invoice-container invoice-count="1" valid="true">
- <invoice valid="true">
  <invoicing-party-nr>1234</invoicing-party-nr>
  <invoice-recipient-nr>9876</invoice-recipient-nr>
  <creation-date>12.12.2002</creation-date>
  <amount>222</amount>
  <requested-payment-date>24.12.2002</requested-payment-date>
  <invoice-nr>1746342</invoice-nr>
  <payment-type>Komplettzahlung</payment-type>
    </invoice>
    </invoice-container>
    </invoice-data-types>
```

Anhang D Dokumentation der Workflow-Engine

D.1 Einleitung

In diesem Anhang werden die einzelnen Funktionen, sowie Eingabe- und Ausgabewerte der BusinessObjects beschrieben. Mit diesen ist eine effizientere Modellierung der Workflows möglich, wobei dieses Dokument als Referenz benutzt werden kann.

Die Beschreibung eines BusinessObjects besteht aus mehreren Teilen. Zunächst wird der vollständig qualifizierte Klassenname des SessionBeans angegeben. Darunter erfolgt die Angabe des aktuellen Status. Somit ist hier eine ständige Aktualisierung notwendig. Verwaltet werden der Spezifizierungsstatus und der Implementierungsstatus. Im Anschluss erfolgt eine Beschreibung der Funktionalität. Zusätzlich sollte hier angegeben werden, ob das BusinessObject einen „Error-Ausgang“ besitzt und wann dieser benutzt wird. Sollte das BusinessObject eine Transaktion benötigen, wo dies nicht anhand des Namens ersichtlich ist (Insert, Update ...), ist dies hier ebenfalls anzugeben. Im Folgenden sind noch die einzelnen Ein- und Ausgabewerte des BusinessObjects zu beschreiben. Die einzelnen FormValues müssen hier jedoch nicht angegeben werden, da dies zu unübersichtlich werden würde. Somit reicht es, sich auf komplexere Objekte und evtl. Primärschlüsselübergaben zu beschränken. Unter der Spalte Key sollte die entsprechende Variable aus den RequestConstants angegeben werden, welche einen bestimmten Dictionary-Wert kennzeichnet. Datentypen sollten selbsterklärend sein. Beim Typ unterscheidet man zwischen Dictionary-Eingangswerten (RD-IN), Dictionary-Ausgangswerten (RD-OUT) und Properties (PROP, werden im Workflow-Context gesetzt). Zusätzlich ist anzugeben, ob die Werte Pflichtangaben (mandatory) oder optional sind. Zuletzt sollte für jeden Dictionary-Key eine kurze Beschreibung abgegeben werden.

D.2 BusinessObjects

D.2.1: Package BusinessPartner

InsertAddress

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.InsertAddressBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Ermittelt, ob sich ein Rechnungssteller oder –empfänger im RD befindet. Erzeugt, wenn dies der Fall ist, ein neues Address-Objekt und ordnet dieses dem gefundenen BusinessPartner zu. Andernfalls wird eine BusinessObjectException geworfen

Key	Datentyp	Typ	Beschreibung
RD_InvoiceRecipient	InvoiceRecipient	RD-IN / optional	Rechnungsempfänger, für den eine neue Adresse erzeugt wird
RD_InvoicintParty	InvoicingParty	RD-IN / optional	Rechnungssteller, für den eine neue Adresse erzeugt wird
RD_Address	Address	RD-OUT / mandatory	Die neu erzeugte Adresse, welche entweder einem Rechnugssteller

			oder Rechnungsempfänger zugeordnet wurde.

UpdateAddress:

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.UpdateAddressBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Ermittelt eine Addresse aus dem RD und aktualisiert sie anhand der Input-Parameter im RD.

Key	Datentyp	Typ	Beschreibung
RD_Address	Address	RD-IN / mandatory	Die Adresse, welche aktualisiert werden soll

DeleteAddress

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.DeleteAddressBean

Status:
Spezifiziert / Implementiert

Beschreibung:
Ermittelt eine Addresse aus dem RD und löscht sie.

Key	Datentyp	Typ	Beschreibung
RD_Address	Address	RD-IN / mandatory	Die Address, welche gelöscht werden soll

DetermineAddress:

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.DetermineAddressBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Ermittelt eine oder mehrere Adressen. Wird eine UUID vorgefunden, ermittelt das BusinessObject nur diese eine Adresse. Wird andernfalls einer der beiden Businesspartner vorgefunden, wird entweder dessen DefaultAdresse (wenn gewünscht), oder alle seine Adressen zurückgegeben.

Key	Datentyp	Typ	Beschreibung
DefaultAddressOnly	boolean	PROP / mandatory	Gibt an, ob nur die DefaultAdresse zurückgegeben werden soll. Andernfalls wird bei Vorhandensein

			eines BusinessPartners eine Collection mit allen Adressen zurückgegeben.
FV_ADDRESSID	String	RD_IN / optional	Die UUID der Adresse, welche ermittelt werden soll
RD_ADDRESS	Address	RD_OUT / optional	Einzelne ermittelte Adresse (entweder über UUID oder Default)
RD_INVOICERECIPIENT	InvoiceRecipient	RD_IN / optional	Der Optionale Rechnungsempfänger, von welchem eine oder mehrere Adressen besorgt werden.
RD_INVOICINGPARTY	InvoicingParty	RD_IN / optional	Der optionale Rechnungssteller, von welchem eine oder mehrere Adressen besorgt werden.
RD_ADDRESSES	Collection	RD_OUT / optional	Collection der ermittlten Adressen eine BusinessPartner's

InsertPersonDataSheet

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.InsertPersonDataSheetBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Erzeugt ein neues PersonDataSheet und setzt es zu dem vorher ermittelten InvoiceRecipient in Beziehung.

Key	Datentyp	Typ	Beschreibung
RD_INVOICERECIPIENT	InvoiceRecipient	RD_IN / mandatory	Der InvoiceRecipient, zu welchem der neue DataSheet in Beziehung gesetzt wird.
RD_PERSONDATASHEET	PersonDataSheet	RD_OUT / mandatory	Der neu erzeugte PersonDataSheet

InsertCompanyDataSheet

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.InsertCompanyDataSheetBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Erzeugt ein neues CompanyDataSheet und setzt es zu dem vorher ermittelten FinancialServiceProvider oder InvoingParty in Beziehung. Wird keiner von den beiden im Dictionary gefunden, wirft das BusinessObject eine BusinessObjectException.

Key	Datentyp	Typ	Beschreibung

RD_INVOICINGPARTY	InvoicingParty	RD_IN / optional	Die optionale InvoicingParty, welche zum neuen DataSheet in Beziehung gesetzt wird.
RD_FINANCIALSERVICEPROVIDER	FinancialServiceProvider	RD_IN / optional	Der optionale FinancialServiceProvider, welcher zum neuen DataSheet in Beziehung gesetzt wird.
RD_COMPANYDATASHEET	CompanyDataSheet	RD_OUT / mandatory	Der neu erzeugte CompannyDataSheet

DeleteCompanyDataSheet

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.DeleteCompanyDataSheetBean

Status:
Spezifiziert / Implementiert

Beschreibung:
Ermittelt ein CompanyDataSheet im Dictionary. Wird keines gefunden, wirft das BusinessObject eine BusinessObjectException, andernfalls wird das gefundene CompanyDataSheet gelöscht.

Key	Datentyp	Typ	Beschreibung
RD_CompanyDataSheet	CompanyDataSheet	RD_IN / mandatory	Der zu löschende CompanyDataSheet

DeterminePersonDataSheet

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.DeterminePersonDataSheetBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Ermittelt ein PersonDataSheet anhand des Primärschlüssels oder anhand eines vorhandenen InvoiceRecipients. Falls nichts Passendes gefunden wird, wird eine BusinessObjectException ausgelöst.

Key	Datentyp	Typ	Beschreibung
RD_INVOICERECIPIENT	InvoiceRecipient	RD_IN / optional	Der InvoiceRecipient, dessen PersonDataSheet ermittelt wird.
FV_PERSONDATASHEET_UUID	String	RD_IN / optional	Die UUID des zu ermittelnden PersonDataSheet
RD_PERSONDATASHEET	PersonDataSheet	RD_OUT / mandatory	Der ermittelte PersonDatSheet

UpdatePersonDataSheet

Bean:

de.Com42Bill.ebppsystem.businesspartner.businessobjects.UpdatePersonDataSheetBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Ermittelt einen InvoiceRecipient oder ein PersonDataSheet aus dem Dictionary. Wird keines der beiden gefunden, wird eine BusinessObjectException ausgelöst. Andernfalls werden die Werte des PersonDataSheet aktualisiert.

Key	Datentyp	Typ	Beschreibung
RD_INVOICERECIPIENT	InvoiceRecipient	RD_IN / optional	Der optionale InvoiceRecipient, dessen PersonDataSheet aktualisert werden soll.
RD_PERSONDATASHEET	PersonDataSheet	RD_IN / optional	Das PersonDataSheet, das aktualisiert werden soll.

UpdateCompanyDataSheet

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.UpdateCompanyDataSheetBean

Status:
Spezifiziert / Rumpf

Beschreibung:
Aktualisiert ein CompanyDataSheet, welches sich im Dictionary befindet. Falls ein Fehler auftritt, wird eine BusinessObjectException ausgelöst.

Key	Datentyp	Typ	Beschreibung
RD_COMPANYDATASHEET	CompanyDataSheet	RD_IN / mandatory	Der zu aktualisierende CompanyDataSheet

DetermineCompanyDataSheet

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.DetermineCompanyDataSheetBean

Status:
Spezifiziert / Rumpf

Beschreibung:
Ermittelt ein CompanyDataSheet anhand des Primärschlüssels oder anhand einer vorhandenen InvoicingParty. Wird nichts passendes gefunden, so wird eine BusinessObjectException geworfen.

Key	Datentyp	Typ	Beschreibung

FV_CompanyDataSheet_UUID	CompanyDataSheet	RD_IN / mandatory	Ermittelt den CompanyDataSheet nach UUID

InsertFinancialAccount

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.InsertFinancialAccountBean

Status:
Spezifiziert / Implementiert

Beschreibung:
Erzeugt einen neuen FinancialAccount und setzt die notwendigen Beziehungen. Zunächst müssen entweder eine InvoicingParty oder ein InvoiceRecipient im Dictionary sein. Ebenfalls sollte der FinancialServiceProvider vorhanden sein. Wenn ein Fehler auftritt, wird eine BusinessObjectException ausgelöst. In diesem BusinessObject sollten ebenfalls alle FinancialAccountAttributes angelegt werden.

Key	Datentyp	Typ	Beschreibung
RD_INVOICINGPARTY	InvoicingParty	RD_IN / optional	InvoingParty, welche dem neuen Account zugeodnet werden soll.
RD_INVOICERECIPIENT	InvoiceRecipient	RD_IN / optional	InvoiceRecepient, welcher dem neuen Account zugeordnet werden soll
RD_FINANCIALSERVICEPROVIDER	FinancialServiceProvider	RD_IN / mandatory	FinancialServiceProvider, welcher dem neuen Account zugeordnet werden soll
RD_FINANCIALACCOUNT	FinancialAccount	RD_OUT / mandatory	Der neu erzeugte FimenancialAccount

UpdateFinancialAccount

de.Com42Bill.ebppsystem.businesspartner.businessobjects.UpdateFinancialAccountBean

Status: Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Aktualisiert einen FinancialAccount mit Hilfe von FormValues. Wenn kein Account im Dictionary gefunden wird, wird eine BusinessObjectException ausgelöst.

Key	Datentyp	Typ	Beschreibung
RD_FINANCIALACCOUNT	FinancialAccount	RD_IN / mandatory	Der zu aktualisierende FinancialAccount

DeleteFinancialAccount

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.DeleteFinancialAccountBean

Status:
Spezifiziert / Implementiert

Beschreibung:
Löscht einen FinancialAccount. Falls kein FinancialAccount gefunden wird, wird eine BusinessObjectException ausgelöst.

Key	Datentyp	Typ	Beschreibung
RD_FINANCIALACCOUNT	FinancialAccount	RD_IN / mandatory	Der zu löschende FinancialAccount

DetermineFinancialAccount

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.DetermineFinancialAccount

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Enthält das Dictionary eine UUID für einen FinancialAccount, wird dieser ermittelt. Wird eine AccountNumber gefunden, wird der FinancialAccount mit dem gegebenen FinancialServiceProvider ermittelt. Wird kein Provider gefunden, wird eine BusinessObjectException ausgelöst. Sind weder UUID noch AccountNo vorhanden, wird nach und nach auf Vorhandensein von InvoiceRecipient, InvoicingParty und FinancialServiceProvider geprüft. Für den ersten gefundenen der drei werden die Accounts (unter Auswertung des DefaultFlags bei InvoicingParty und InvoiceRecipient) ermittelt. Sollte überhaupt kein Input gefunden werden, wird eine BusinessObjectException ausgelöst.

Key	Datentyp	Typ	Beschreibung
FV_FINANCIALACCOUNT_UUID	String	RD_IN / optional	Die UUID des zu ermittelnden FinancialAccounts
RD_INVOICERECIPIENT	InvoiceRecipient	RD_IN / optional	InvoiceRecipient, dessen FinancialAccounts ermittelt werden (Default-Flag wird ausgewertet)
RD_INVOICINGPARTY	InvoicingParty	RD_IN / optional	InvoicingParty, derern FinancialAccounts ermittelt werden (Default-Flag wird ausgewertet)
RD_FINANCIALSERVICEPROVIDER	FinancialServiceProvider	RD_IN / optional	FinancialServiceProvider, dessen FincialAccounts ermittelt werden. Wird ein Account anahnd der AccountNumber ermittelt, wird hierfür der ServiceProvider zwingend benötigt
FV_FINANCIALACCOUNT_ACCOUNTNO	String	RD_IN / optional	Der zugehörige Account wird für den gegebenen FinancialServiceProvider ermittelt
RD_FINANCIALACCOUNT	FinancialAccount	RD_OUT / optional	Ein ermittelter FinancialAccount
RD_FINANCIALACCOUNT_COLLECTION	Collection	RD_OUT / optional	Ein Liste von ermittelten FinancialAccounts.
Default	Boolean	PROP / optional	Dieses Flag bestimmt, ob nur der Default-FinancialAccount ermittelt,

			wenn mehrere existieren. (Standard: false)

DetermineInvoiceRecipient

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.DetermineInvoiceRecipientBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Zunächst werden die beiden Config-Flags abgefragt. Sind beide auf ‚true' gesetzt, wird eine WorkflowInitException ausgelöst. Sonst wird im Falle von „DetermineByUser" der aktuelle Benutzer aus dem Dictionary geholt und anhand dessen der BusinessPartner ermittelt und im Dictionary abgelegt. Bei „DetermineAll" werden alle vorhandenen InvoiceRecipients ermittelt. Dies sollte aus Performance-Gründen vermieden werden und später durch ordentliche Abfragen ersetzt werden. Sind beide Flags auf false gesetzt, wird der InvoiceRecipient entweder anhand seiner UUID oder seiner Kundennummer ermittelt.

Key	Datentyp	Typ	Beschreibung
FV_INVOICERECIPIENT_UUID	String	RD_IN / optional	Die UUID des zu suchenden InvoiceRecipients
FV_INVOICERECIPIENT_CUSTOMERNUMBER	String	RD_IN / optional	Die Kundennummer des zu suchenden InvoiceRecipients.
DetermineByUser	Boolean	PROP / mandatory	Mit Hilfe dieses Flags wird festgesetzt, ob der InvoiceRecipient anhand des eingeloggten Users bestimmt werden soll. (Standard: false)
DetermineAll	Boolean	PROP / optional	Mit Hilfe dieses Flags wird festgesetzt, ob alle existierenden Benutzer ermittelt werden sollen. (Standard: false)
RD_INVOICERECIPIENT	InvoiceRecipient	RD_OUT / optional	Der ermittelte InvoiceRecipient
RD_INVOICERECIPIENT_COLLECTION	Collection	RD_OUT / optional	Eine Liste von ermittelten InvoiceRecipients
RD_USERINFORMATION	UserInformation	RD_IN / optional	Der optionale User, von welchen der BusinessPartner ermittelt werden kann

InsertInvoiceRecipient

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.InsertInvoiceRecipientBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:

Erstellt einen neuen InvoiceRecipient und füllt ihn mit FormValues.

Key	Datentyp	Typ	Beschreibung
RD_INVOICERECIPIENT	InvoiceRecipient	RD_OUT / mandatory	Der angelegte InvoiceRecipient

UpdateInvoiceRecipient

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.UpdateInvoiceRecipientBean

Status:
Spezifiziert / Rumpf

Beschreibung:
Aktualisiert einen vorher ermittelten InvoiceRecipient anhand der FormValues

Key	Datentyp	Typ	Beschreibung
RD_INVOICERECIPIENT	InvoiceRecipient	RD_IN / Mandatory	Der InvoiceRecipient, welcher aktualisiert werden soll.

DetermineInvoicingParty

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.DetermineInvoicingPartyBean

Status:
Rumpf

Beschreibung:
Zunächst werden die beiden Config-Flags abgefragt. Sind beide auf ‚true' gesetzt, wird eine WorkflowInitException ausgelöst. Sonst wird im Falle von „DetermineByUser" der aktuelle Benutzer aus dem Dictionary geholt und anhand dessen der BusinessPartner ermittelt und im Dictionary abgelegt. Bei „DetermineAll" werden alle vorhandenen InvoicingParties ermittelt. Dies sollte aus Performance-Gründen vermieden werden und später durch ordentliche Abfragen ersetzt werden. Sind beide Flags auf false gesetzt, wird die InvoicingParty entweder anhand seiner UUID oder seiner Kundennummer ermittelt.

Key	Datentyp	Typ	Beschreibung
FV_INVOICINGPARTY_UUID	String	RD_IN / optional	Die UUID der zu suchenden InvoicingParty
FV_INVOICINGPARTY_CUSTOMERNUMBER	String	RD_IN / optional	Die Kundennummer der zu suchenden InvoicingParty.
DetermineByUser	Boolean	PROP / mandatory	Mit Hilfe dieses Flags wird festgesetzt, ob die InvoicingParty anhand des eingeloggten Users bestimmt werden soll. (Standard: false)
DetermineAll	Boolean	PROP / optional	Mit Hilfe dieses Flags wird festgesetzt, ob alle existierenden Benutzer ermittelt werden sollen.

			(Standard: false)
RD_INVOICINGPARTY	InvoicingParty	RD_OUT / optional	Die ermittelte InvoicingParty
RD_INVOICINGPARTY_COLLECTION	Collection	RD_OUT / optional	Eine Liste von ermittelten InvoicingParties
RD_USERINFORMATION	UserInformation	RD_IN / optional	Der optionale User, von welchen der BusinessPartner ermittelt werden kann

InsertInvoicingParty

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.InsertInvoicingPartyBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Legt eine neue InvoicingParty an und füllt sie mit FormValues.

Key	Datentyp	Typ	Beschreibung
RD_INVOICINGPARTY	InvoicingParty	RD_OUT / mandatory	Die neu erzeugte InvoicingParty

UpdateInvoicingParty

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.UpdateInvoicingPartyBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Aktualisiert eine vorher ermittelte InvoingParty mit FormValues

Key	Datentyp	Typ	Beschreibung
RD_INVOICINGPARTY	InvoicingParty	RD_IN / mandatory	Die zu aktualisierende InvoicingParty

InsertProfileGroup

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.InsertProfileGroupBean

Status:
Spezifiziert / Implementiert

Beschreibung:
Legt eine neue ProfileGroup an.

Key	Datentyp	Typ	Beschreibung
RD_PROFILEGRO	ProfileGroup	RD_OUT /	Die neu erzeugte ProfileGroup

UP		mandatory	

UpdateProfileGroup

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.UpdateProfileGroupBean

Status:
Spezifiziert / Rumpf

Beschreibung:
Aktualisiert eine vorher ermittelte ProfileGroup. Tritt dabein ein Fehler auf, wird eine BusinessObjectException ausgelöst.

Key	Datentyp	Typ	Beschreibung
RD_PROFILEGROUP	ProfileGroup	RD_IN / mandatory	Die zu aktualisierende ProfileGroup

DeleteProfileGroup

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.DeleteProfileGroupBean

Status:
Spezifiziert / Rumpf

Beschreibung:
Löscht eine vorher ermittelte ProfileGroup. Tritt dabei ein Fehler auf, wird eine BusinessObjectException ausgelöst.

Key	Datentyp	Typ	Beschreibung
RD_PROFILEGROUP	ProfileGroup	RD_IN / mandatory	Die zu löschende ProfileGroup

AssignProfileGroupsToBusinessPartner

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.AssignProfileGroupsToBusinessPartnerBean

Status:
Spezifiziert / Rumpf

Beschreibung:
Erstellt Beziehungen zwischen einem BusinessPartner (InvoicingParty/InvoiceRecipient) und ProfileGroups. Es werden sowohl neue Beziehung gesetzt als auch nicht mehr benötigte entfernt. Wird keine InvoicingParty gefunden, wird eine BusinessObjectException ausgelöst. Wird andernfalls keine ProfileGroup-Collection gefunden, wird ebenfalls eine BusinessObjectException ausgelöst.

Key	Datentyp	Typ	Beschreibung
RD_INVOICERECIPIENT	InvoiceRecipient	RD_IN / optional	Der optionale erste Teil für die Beziehung

RD_INVOICINGPARTY	InvoicingParty	RD_IN / optional	Der optional erste Teil für sie Beziehung
RD_PROFILEGROUP_COLLECTION	Collection	RD_IN / mandatory	Die ProfileGroups, mit denen eine Beziehung hergestellt werden soll. Beziehung zu ProfileGroups, die nicht in der Collection sind, müssen entfernt werden.

DetermineProfileGroup

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.DetermineProfileGroupBean

Status:
Spezifiziert / Rumpf

Beschreibung:
Ermittelt eine ProfileGroup anhand ihrer UUID oder ihres Namens. Andernfalls werden alle ermittelt.

Key	Datentyp	Typ	Beschreibung
FV_PROFILEGROUP_UUID	String	RD_IN / optional	UUID der zu ermittelnden ProfileGroup
FV_PROFILEGROUP_NAME	String	RD_IN / optional	Name der zu ermittelnden ProfileGroup
RD_PROFILEGROUP	ProfileGroup	RD_OUT / optional	Einzelne ermittelte ProfileGroup
RD_PROFILEGROUP_COLLECTION	Collection	RD_OUT / optional	Collection von ermittelten ProfileGroups

InsertFinancialServiceProvider

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.InsertFinancialServiceProviderBean

Status:
Spezifiziert / Implementiert

Beschreibung:
Legt einen neuen FinancialServiceProvider an.

Key	Datentyp	Typ	Beschreibung
RD_FINANCIALSERVICEPROVIDER	FinancialServiceProvider	RD_OUT / mandatory	Der neu erzeugte FinancialServiceProvider.

UpdateFinancialServiceProvider

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.UpdateFinancialServiceProviderBean

Status:
Spezifiziert / Rumpf

Beschreibung:
Aktualisiert einen vorher ermittelten FinancialServiceProvider anhand der gegebenen FormValues. Falls ein Fehler auftritt, wird eine BusinessObjectException ausgelöst.

Key	Datentyp	Typ	Beschreibung
RD_FINANCIALSERVICEPROVIDER	FinancialServiceProvider	RD_IN / mandatory	Der zu aktualisierende FinancialServiceProvider

DeleteFinancialServiceProvider

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.DeleteFinancialServiceProviderBean

Status:
Spezifiziert / Rumpf

Beschreibung:
Löscht einen vorher ermittelten FinancialServiceProvider. Falls keiner gefunden wird, wird eine BusinessObjectException ausgelöst.

Key	Datentyp	Typ	Beschreibung
RD_FINANCIALSERVICEPROVIDER	FinancialServiceProvider	RD_IN / Mandatory	Der zu löschende FinancialServiceProvider

DetermineFinancialServiceProvider

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.DetermineFinancialServiceProviderBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Da es vorerst keine GUI zur Pflege von FinancialServiceProvidern geben wird, muss dieser automatisch angelegt werden, sofern noch nicht im System vorhanden. Daher wird der Error-Ausgang benutzt, sofern weder unter der UUID noch der BLZ oder der CPAID ein FinancialServiceProvider gefunden wurde. Sollte keiner dieser drei FormValues im Dictionary vorhanden sein, wird das Flag ‚DetermineAll' ausgewertet. Standardmässig wird eine BusinessObjectException geworfen. Steht das Flag auf true, wird eine Collection mit allen FinancialServiceProvidern ermittelt.

Key	Datentyp	Typ	Beschreibung
FV_FINANCIALSERVICEPROVIDER_UUID	String	RD_IN / optional	UUID des zu ermittelnden ServiceProviders
FV_FINANCIALSERVICEPROVIDER_BLZ	String	RD_IN / optional	BLZ der zu ermittelnden ServiceProviders
FV_FINANCIALSERVICEPROVIDER_C	String	RD_IN / optional	CPAID des zu ermittelnden ServiceProviders

PAID			
RD_FINANCIALSERVICEPROVIDER	FinancialServiceProvider	RD_OUT / optional	Ein einzelner ermittelter ServiceProvider
RD_FINANCIALSERVICEPROVIDER_COLLECTION	Collection	RD_OUT / optional	Eine Collection von ermittelten ServiceProvidern
DetermineAll	Boolean	PROP / optional	Ist das Flag auf true gesetzt, werden alle existierenden FinancialServiceProvider zurückgegeben, sofern keine einschränkenden Parameter angeben sind. Falls ein Fehler auftritt, wird eine BusinessObjectException ausgelöst. (Standard:false)

ImportFinancialServiceProvider

Bean:
de.Com42Bill.ebppsystem.businesspartner.businessobjects.ImportFinancialServiceProviderBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Liest eine CSV-Datei ein und legt die dort spezifizierten FinancialServiceProvicer an. Ist bereits ein Datensatz mit der BLZ vorhanden, wird der zu importierende Datensatz zu Testzwecken zunächst ignoriert.

Key	Datentyp	Typ	Beschreibung
AbsolulteFilePath	String	PROP / mandatory	Der absolute Pfad zur CSV-Datei

D.2.2: Package Invoice

InsertInvoiceType

Bean:
de.Com42Bill.ebppsystem.invoice.businessobjects.InsertInvoiceTypeBean

Status:
Rumpf

Beschreibung:
Legt einen neuen InvoiceType an.

Key	Datentyp	Typ	Beschreibung
FV_FINANCIALACCOUNT_UUID	String	RD_OUT / mandatory	Der neu angelegte InvoiceType.

UpdateInvoiceType

Bean:

de.Com42Bill.ebppsystem.invoice.businessobjects.UpdateInvoiceTypeBean

Status:
Spezifiziert / Rumpf

Beschreibung:
Aktualisiert einen vorher ermittelten InvoiceType. Wird keiner gefunden, wird eine BusinessObjectException ausgelöst.

Key	Datentyp	Typ	Beschreibung
RD_INVOICETYPE	InvoiceType	RD_IN / mandatory	Der zu aktualisierende InvoiceType

DeleteInvoiceType

Bean:
de.Com42Bill.ebppsystem.invoice.businessobjects.DeleteInvoiceTypeBean

Status:
Spezifiziert / Rumpf

Beschreibung:
Löscht einen vorher ermittelten InvoiceType. Wird keiner gefunden, wird eine BusinessObjectException ausgelöst. Hängen an dem InvoiceType noch Invoice dran, darf der InvoiceType nicht entfernt werden. Sollte dies der Fall sein, wird der ERROR-Ausgang des BusinessObjects benutzt und ein Fehler-Hinweis ins Dictionary eingetragen.

Key	Datentyp	Typ	Beschreibung
RD_INVOICETYPE	InvoiceType	RD_IN / mandatory	Der zu löschende InvoiceType
ER_INVOICETYPE_BOUND	String	RD_OUT / optional	Dieser Wert wird im Dictionary eingetragen, wenn dem InvoiceType noch Invoices zugeordnet sind.

DetermineInvoiceType

Bean:
de.Com42Bill.ebppsystem.invoice.businessobjects.DetermineInvoiceTypeBean

Status:
Spezifiziert / Rumpf

Beschreibung:
Ermittelt ein oder mehrere InvoiceTypes. Zunächst wird auf das Vorhandensein einer UUID geprüft. Wird keine vorgefunden, ist der Versuch mit einem Namen für den InvoiceType durchzuführen. Sind beide Versuche erfolglos, werden alle InvoiceTypes ermittelt.

Key	Datentyp	Typ	Beschreibung
FV_INVOICETYPE_UUID	String	RD_IN / optional	Die UUID des zu ermittelnden InvoiceTypes
FV_INVOICETYPE_NAME	String	RD_IN / optional	Der Names des zu ermittelnden InvoiceTypes

RD_INVOICETYPE_COLLECTION	Collection	RD_OUT / optional	Collection mit ermittelten InvoiceTypes
RD_INVOICETYPE	InvoiceType	RD_OUT / optional	Ein ermittelter InvoiceType (anhand seiner UUID oder seines Namens)

InsertInvoiceLink

Bean:
de.Com42Bill.ebppsystem.invoice.businessobjects.InsertInvoiceLinkBean

Status:
Spezifiziert / Rumpf

Beschreibung:
Legt einen InvoiceLink zwischen zwei vorher ermittelten Invoices an. Wird eine der beiden nicht gefunden, wird eine BusinessObjectException ausgelöst.

Key	Datentyp	Typ	Beschreibung
RD_INVOICE_SOURCE	Invoice	RD_IN / mandatory	Der erste Teil des zu erstellenden InvoiceLinks
RD_INVOICE_TARGET	Invoice	RD_IN / mandatory	Der zweite Teil des zu erstellenden InvoiceLinks
RD_INVOICELINK	InvoiceLink	RD_OUT / mandatory	Der neu erstellte InvoiceLink

DeleteInvoiceLink

Bean:
de.Com42Bill.ebppsystem.invoice.businessobjects.DeleteInvoiceLinkBean

Status:
Spezifiziert / Rumpf

Beschreibung:
Entfernt ein oder mehrere InvoiceLinks. Wird ein voher ermittelter InvoiceLink aufgefunden, wird dieser einfach gelöscht. Das gleiche passiert mit einer InvoiceLink-Collection. Werden zwei Invoices aufgefunden, wird der InvoiceLink zwischen diesen beiden entfernt. Wird nur eine Invoice gefunden, werden alle InvoiceLinks, die von dieser Invoice abgehen entfernt. Wird nichts dergleichen aufgefunden, wird eine BusinessObjectException ausgelöst.

Key	Datentyp	Typ	Beschreibung
RD_INVOICELINK	InvoiceLink	RD_IN / optional	Ein einzelner zu löschender InvoiceLink
RD_INVOICELINK_COLLECTION	Collection	RD_IN / optional	Eine Collection von zu löschenden InvoiceLinks
RD_INVOICE_SOURCE	Invoice	RD_IN / optional	Der erste Teil, der zwei Invoices, zwischen denen der InvoiceLink gelöscht werden soll
RD_INVOICE_TARGET	Invoice	RD_IN / optional	Der zweite Teil der zwei Invoices, zwischen denen der InvoiceLink gelöscht werden soll.
RD_Invoice	Invoice	RD_IN / optional	Die Invoice, von der alle

			abgehenden InvoiceLinks entfernt werden sollen.

InsertInvoice

Bean:
de.Com42Bill.ebppsystem.invoice.businessobjects.InsertInvoiceBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Erzeugt eine neue Rechnung und setzt sie zu den beiden zugehörigen BusinessPartnern in Beziehung. Sollte einer der beiden fehlen, wird eine BusinessObjectException ausgelöst.

Key	Datentyp	Typ	Beschreibung
RD_INVOICERECIPIENT	InvoiceRecipient	RD_IN / mandatory	Der zuzuordnende Rechnungsempfänger
RD_INVOICINGPARTY	InvoicingParty	RD_IN / mandatory	Der zuzuordnende Rechnungssteller
RD_INVOICEBEAN	Invoice	RD_OUT / mandatory	Die neu erzeugte Rechnung

DetermineInvoice

Bean:
de.Com42Bill.ebppsystem.invoice.businessobjects.DetermineInvoiceBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Ermittelt Rechnungen entweder für den Datenkonverter anhand der InvoiceNumber oder zur GUI-Anzeige anhand der übergebenen Suchkriterien.

Key	Datentyp	Typ	Beschreibung
RD_INVOICEBEAN_COLLECTION	Collection	RD_OUT / optional	Die ermittelten Rechnungen.

D.2.3: Package Payment/InvoiceContainer

AddInvoiceContainerItems

Bean:
de.Com42Bill.ebppsystem.payment.invoicecontainer.businessobjects.AddInvoiceContainerItemsBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:

Dieses BusinessObject fügt neue Invoices in den aktuellen InvoiceContainer ein. Vorher muss der aktuelle InvoiceContainer ermittelt werden. Falls ein Fehler auftritt, wird eine BusinessObjectException ausgelöst. Das BusinessObject erhält als Input ein Invoice, zu den dann InvoiceContainerItem angelegt wird, welche wiederum mit dem InvoiceContainer in Beziehung gesetzt wird. Wird keine Rechnung vorgefunden, wird eine BusinessObjectException ausgelöst.

Key	Datentyp	Typ	Beschreibung
RD_INVOICEBEAN	Collection	RD_IN / mandatory	Collection mit Invoices, für die ein InvoiceContainerItem erstellt werden soll, welches dann zum aktuellen InvoiceConatiner in Beziehung gesetzt wird.
RD_INVOICECONTAINER	InvoiceContainer	RD_IN / mandatory	The actual InvoiceContainer.

UpdateInvoiceContainerItems

Bean:
de.Com42Bill.ebppsystem.payment.invoicecontainer.businessobjects.UpdateInvoiceContainerItemsBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Dieses BusinessObject aktualisiert die Attribute der InvoiceContainerItems.

Key	Datentyp	Typ	Beschreibung
FV_INVOICECONTAINERITEM_UUID	String	RD_IN / mandatory	Aktualisiert die Attribute des InvoiceContainerItems.

DeleteInvoiceContainerItems

Bean:
de.Com42Bill.ebppsystem.payment.invoicecontainer.businessobjects.DeleteInvoiceContainerItemsBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Dieses Businessobject löscht ein InvoiceContainerItem aus dem aktuellen InvoiceContainer Wird jedoch kein aktueller InvoiceContainer im Dictionary vorgefunden, wird eine BusinessObjectException ausgelöst.

Key	Datentyp	Typ	Beschreibung
RD_INVOICEBEAN	Collection	RD_IN / optional	Eine Collection mit Invoices, für die die entsprechenden InvoicerContainerItems gelöscht werden müssen. Diese befinden sich am aktuellen InvoiceContainer.
RD_INVOICECON	InvoiceContainer	RD_IN /	The actual InvoiceContainer.

TAINER		mandatory	

DetermineInvoiceContainer

Bean:
de.Com42Bill.ebppsystem.payment.invoicecontainer.businessobjects.DetermineInvoiceContainerBean

Status:
Spezifiziert / Teilimplementiert / In Betrieb

Beschreibung:
Ermittelt ein oder mehrere InvoiceContainer für den aktuellen InvoiceRecipient. Je nachdem, wie die Properties gesetzt wurden, wird nur der aktuelle, nur die History-Container oder alle ermittelt. Sollte kein aktueller existieren, jedoch angefordert worden sein, muss er angelegt werden. In der aktuellen Implementierung wird nur der aktive InvoiceContainer zurückgegeben. Ist kein InvoiceRecipient vorhanden, wird eine BusinessObjectException ausgelöst.

Key	Datentyp	Typ	Beschreibung
RD_INVOICERECIPIENT	InvoiceRecipient	RD_IN / mandatory	Der InvoiceRecipient, dessen InvoicerContainer ermittelt werden soll(en).
DetermineActual	boolean	PROP / optional	Set to ‚false', if you don't want the actual InvoiceContainer to be determined (default is ‚true').
DetermineHistory	Boolean	PROP / optional	Set to 'true', if you want the History-InvoiceContainers to be determined (default is 'false').
RD_INVOICECONTAINER	InvoiceContainer	RD_OUT / optional	The actual InvoiceContainer.
RD_INVOICECONTAINER_COLLECTION	Collection	RD_OUT / optional	Collection with History-InvoiceContainer for the current InvoiceRecipient

CheckoutInvoiceContainer

Bean:
de.Com42Bill.ebppsystem.payment.invoicecontainer.businessobjects.CheckoutInvoiceContainerBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Beim Container-Checkout wird der Typ geändert und das Checkout-Date gesetzt.

Key	Datentyp	Typ	Beschreibung
RD_INVOICECONTAINER	InvoiceContainer	RD_IN / mandatory	The actual InvoiceContainer, that has to be checked out.
RD_INVOICECONTAINERITEM_COLL	Collection	RD_Out / mandatory	The collection of invoices, that were in InvoiceContainer.

ECTION			

D.2.4: Package Payment

ExecuteScheduledPaymentTransactions

Bean:
de.Com42Bill.ebppsystem.payment.businessobjects.ExecuteScheduledPaymentTransactionsBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Ermittelt die ScheduledTransactionLists die fällig sind und erzeugt JobTransactionList.

Key	Datentyp	Typ	Beschreibung

DetermineExportPaymentTransactions

Bean:
de.Com42Bill.ebppsystem.payment.businessobjects.DetermineExportPaymentTransactionsBean

Status: Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Ermittelt alle JobTransactionLists für die Ausführung von Jobs.

Key	Datentyp	Typ	Beschreibung
RD_PAYMENTTRANSACTION_COLLECTION	Collection	RD_Out / mandatory	Der übergebene Paymenttransactionliste.

InsertPaymentTransaction

Bean:
de.Com42Bill.ebppsystem.payment.businessobjects.InsertPaymentTransactionBean

Status:
Spezifiziert / Implementiert / In Betrieb

Beschreibung:
Dieses Businessobject holt sich eine Rechnung aus der InvoiceCollection und erzeugt eine neue Überweisung.

Key	Datentyp	Typ	Beschreibung
RD_INVOICECONTAINERITEM_COLLECTION	Collection	RD_IN / mandatory	Der InvoiceCollection mit den zu überweisende Rechnungen.
RD_FINANCIALACCOUNT	FinancialAccountLocal	RD_IN / mandatory	FinancialAcount von InvoiceRecipient
RD_PAYMENTTRA	PaymentTransact	RD_IN / optional	Wenn eine PaymentTransactionlist

NSACTIONLIST	ionListLocal		angegeben ist, wird es in Beziehung zu PaymentTransaction gesetzt.
RD_PAYMENT_TRANSACTION_LOCAL	PaymentTransactionLocal	RD_Out / mandatory	

UpdatePaymentTransaction

Bean:
de.Com42Bill.ebppsystem.payment.businessobjects.UpdatePaymentTransactionBean

Status:
Rumpf

Beschreibung:
Aktualisiert die Attributte von PaymentTransaction.

Key	Datentyp	Typ	Beschreibung

InsertPaymentMethod

Bean:
de.Com42Bill.ebppsystem.payment.businessobjects.InsertPaymentMethodBean

Status:
Rumpf

Beschreibung:
Erzeugt eine Zahlungmethode für ein Konto.

Key	Datentyp	Typ	Beschreibung

UpdatePaymentMethod

Bean:
de.Com42Bill.ebppsystem.payment.businessobjects.UpdatePaymentMethodBean

Status:
Rumpf

Beschreibung:
Aktualisiert die Attribute von der Zahlungmethode.

Key	Datentyp	Typ	Beschreibung

DeletePaymentMethod

Bean:
de.Com42Bill.ebppsystem.payment.businessobjects.DeletePaymentMethodBean

Status:
Rumpf

Beschreibung:
Löscht eine Zahlungmethode

Key	Datentyp	Typ	Beschreibung

DeterminePaymentMethod

Bean:
de.Com42Bill.ebppsystem.payment.businessobjects.DeterminePaymentMethodBean

Status:
Rumpf

Beschreibung:
Ermittelt die Zahlungsmethode.

Key	Datentyp	Typ	Beschreibung

InsertPaymentTransactionList

Bean:
de.Com42Bill.ebppsystem.payment.businessobjects.InsertPaymentTransactionListBean

Status:
Spezifiziert / Implementiert

Beschreibung:
Dieses erzeugt ein PaymentTransactionList Object und setzt es in Beziehung mit der PaymentTransaction.

Key	Datentyp	Typ	Beschreibung
RD_PAYMENT_TRANSACTION_LOCAL	PaymenttransactionLocal	RD_IN / mandatory	Der PaymentTranscaction zu dem eine Beziehung erstellt werden soll.

DeterminePaymentTransactionList

Bean:
de.Com42Bill.ebppsystem.payment.businessobjects.DeterminePaymentTransactionListBean

Status:
Spezifiziert / Implementiert

Beschreibung:
Dieses Business Objekt ermittelt alle PaymentTransactions.

Key	Datentyp	Typ	Beschreibung
RD_INVOICERECIPIENT	String	RD_IN / mandatory	Der Rechnungsempfänger, für den alle PaymenttansactionList ermittelt werden sollen.

InsertJobTransaction

Bean:
de.Com42Bill.ebppsystem.payment.businessobjects.InsertJobTransactionBean

Status:
Spezifiziert / Implementiert

Beschreibung:
Dieses Business Objekt erzeugt eine neue JobTransaction.

Key	Datentyp	Typ	Beschreibung
RD_PAYMENT_TRANSACTION_LOCAL	PaymenttransactionLocal	RD_IN / mandatory	Der PaymentTranscaction zu dem eine Beziehung erstellt werden soll.

D.2.5: Package Core

InsertJob

Bean:
de.Com42Bill.ebppsystem.core.businessobjects.InsertJobBean

Status:
Spezifiziert / Rumpf / In Betrieb

Beschreibung:
Legt eine neue JobConfiguration und den dazugehörigen JobTimer an.

Key	Datentyp	Typ	Beschreibung
RD_JOBCONFIGURATION	JobConfiguration	RD_OUT / mandatory	Die neu erzeugte JobConfiguration
RD_JOBTIMER	JobTime	RD_OUT / mandatory	Der neu erzeugte JobTimer, der mit der JobConfiguration in Bezihung gesetzt wurde.

DeleteJob

Bean:
de.Com42Bill.ebppsystem.core.businessobjects.DeleteJobBean

Status:
Spezifziert / Rumpf / In Betrieb

Beschreibung:
Löscht eine JobConfiguration und den zugehörigen Timer. Es wird z. Zt. Nicht berücksichtigt, ob der Job in der Ausführung ist. Anschliessend muss der Job aus dem Dictionary entfernt werden, damit die GUI nicht auf die gelöschte Instanz zugreifen kann.

Key	Datentyp	Typ	Beschreibung
RD_JOBCONFIGURATION	JobConfiguration	RD_IN / mandatory	Die zu löschende JobConfiguration

DetermineJobs

Bean:
de.Com42Bill.ebppsystem.core.businessobjects.DetermineJobsBean

Status:
Spezifziert / Rumpf /In Betrieb

Beschreibung:
Ermittelt ein oder mehrere JobConfigurations. Wird die UUID einer JobConfiguration vorgefunden, wird eine einzelne ermittelt, ansonsten alle.

Key	Datentyp	Typ	Beschreibung
FV_JOBCONFIGURATION_UUID	String	RD_IN / optional	Die UUID der zu ermittelnden JobConfiguration.
RD_JOBCONFIGURATION	JobConfiguration	RD_OUT / optional	Die anhand der UUID ermittelte JobConfiguration
RD_JOBCONFIGURATION_COLLECTION	Collection	RD_OUT / optional	Alle ermittelten JobConfigurations

UpdateJob

Bean:
de.Com42Bill.ebppsystem.core.businessobjects.UpdateJobBean

Status:
Spezifiziert / Rumpf / In Betrieb

Beschreibung:
Aktualisiert eine vorhandene JobConfiguration oder den zugehörigen JobTimer. Falls keine JobConfiguration vorhanden íst, wird eine BusinessObjectException ausgelöst.

Key	Datentyp	Typ	Beschreibung
RD_JOBCONFIGURATION	JobConfiguration	RD_IN / mandatory	Die zu aktualisierende JobConfiguration

InsertUserInformartion

Bean:
de.Com42Bill.ebppsystem.core.businessobjects.InsertUserInformation

Status: Spezifiziert / Rumpf / In Betrieb

Beschreibung:
Legt einen neuen User an. Das Passwort wird verschlüsselt. Der Passwortvergleich muss bei der Eingabe durch die GUI erfolgen, da das Kontrollpasswort nicht bis hierhin durchgereicht wird.

Key	Datentyp	Typ	Beschreibung
RD_USERINFORMATION	UserInformation	RD_OUT / mandatory	Die neu erzeugte Userinformation

DeleteUserInformation

Bean:

de.Com42Bill.ebppsystem.core.businessobjects.DeleteUserInformationBean

Status: Rumpf

Beschreibung:
Löscht die UserInformation.

Key	Datentyp	Typ	Beschreibung

D.3 Workflows

D.3.1: BusinessPartner

StartNode	Funktionalität	Typ
InsertAddress	Pflegt eine neue die Adresse ein.	Private
UpdateAddress	Aktualisiert die Adresse	GUI
UpdateDataSheet/CompanySheet	Aktualisiert persönliche Daten.	GUI
InsertFinancialAccount	Fügt eine neue Bankverbindung ein.	GUI
UpdateFinancialAccount	Aktualisiert die Bankverbindung, fügt neue hinzu, wenn keine UID mitgegeben wurde	GUI
RemoveFincialAccount	Entfernen der Bankverbindung	GUI
UpdateInformation	UserInformation aktualisieren	GUI
DetermineInformation	UserInformation ermitteln	GUI
DetermineFinancialAccounts	FinancialAccounts ermitteln	GUI
DetermineFinancialAccount	FinancialAccount ermitteln	GUI

D.3.2: InvoiceRecipient:

StartNode	Funktionalität	Typ
Login	Holt InvoiceRecipient anhand von Userinformation und übermittelt evtl. neu eingetroffene, fällige Rechnungen und Mahnungen an die Komponente GUI.	GUI
Register	Legt eine neue InvoiceRecipient an. Danach wird diese mit eine neuen erzeugten PersonalDatasheet, Adresse in Beziehung gesetzt.	GUI
DetermineAll	Ermittelt alle InvoiceRecipients	Admin
DetermineFinancialAccounts	Ermittelt alle FinancialAccounts des eingeloggten Benutzer	GUI

D.3.3: InvoicingParty:

StartNode	Funktionalität	Typ
Login	Holt InvoicingParty anhand von Userinformation und übermittelt evtl. neu eingetroffene, fällige Rechnungen und Mahnungen an die Komponente GUI.	GUI
Register	Legt eine neue InvoicingParty an. Danach wird diese mit eine neuen erzeugten PersonalDatasheet, Adresse in Beziehung gesetzt.	GUI

D.3.4: InvoiceForInvoicingParty:

StartNode	Funktionalität	Typ
Summary	Ermittelt alle Rechnungen	GUI
DetailView	Ermittelt eine einzelne Rechnung	GUI
Insert	Ermittelt den InvoicingParty und InvoiceRecipient. Danach wird eine neue Rechnung erzeugt, die in Beziehung zu beteiligten gesetzt wird.	DataConverter
Cancellation	Ermittelt eine Rechnung und storniert sie.	DataConverter
Update	???	DataConverter
Search	Suche alle Rechnungen nach bestimmten Suchkriterien	GUI

D.3.5: InvoiceForInvoiceRecipient:

StartNode	Funktionalität	Typ
Summary	Ermittelt alle offenen Rechnungen	GUI
DetailView	Ermittelt eine einzelne Rechnung	GUI
Search	Suche alle Rechnungen nach bestimmten Suchkriterien	GUI

D.3.6: InvoiceContainer:

StartNode	Funktionalität	Typ
CreateContainer	Erzeugt einen neuen Container für den Rechnungsempfänger	Private
AddItems	Erzeugt für übergebenen Invoices Items, die in Container eingefügt werden.	GUI
RemoveItems	Entfernt für übergebenen Invoices Items aus dem Container.	GUI
CheckoutItems	Erzeugt für jede Item in den Container einen PaymentTransaktion und fügt diese in den JobList.	GUI
UpdateItems	Setzt weitere Attribute des ContainerItems, wie höhe des Betrages beispielweise bei Rechnungskurzungen, Ttransaktionsdatum, Kontenauswahl.	GUI
DetermineContainer	Ermitteln von Container eines bestimmten Typs	GUI
UpdateItems	AddItems + RemoveItems	GUI
DetermineItems	Invoices im Container	GUI

D.3.7: Payment:

Startnode	Funktionalität	Typ
ScheduledTransaction	Exportieren von PaymentTransaction aus der ScheduledTransactionList zu einem angebebenen Zeitpunkt in die ExportTransactionList	Job
DetermineExportTransactions	Ermitteln der zu exportierende Transaction.	Job
InsertPaymentTransaction	Erzeugt die PaymentTransaction und setzt die Zahlungsart	Private

D.3.8: JobManagment:

Startnode	Funktionalität:	Typ

Insert	Einfügen eines Jobs	Admin
Remove	Löschen eines Jobs	Admin
DetermineAll	Ermittelt alle JobConfigurations	Admin
Update	Äderungen der Parameters eines bestehen den Jobs	Admin
Detail	Ermittelt einen einzelnen Job für die Detailansicht	Admin

D.3.9: Alert:

Startnode	Funktionalität	Typ
SendInvoiceReminders	Benachrichtigung der fälligen Rechnungen	Job
SendNewInvoices	Benachrichtigung über neu eingetroffene Rechnung	DataConverter
SendPaymentWarnings	Mahnung per Email verschicken	Job
CreatePaymentWarnings	Sucht überfällige Rechnungen und erzeugt Mahnungen.	Job

D.3.10: UserManagement:

Startnode	Funktionalität	Typ
Insert	Anlegen eines neuen Users	Admin
Remove	Löschen eines neuen bestehenden Users	Admin
Update	Äderung der Parameter eines Users	Admin
InsertUsergroup	Anlegen einer neuen Gruppe	Admin
RemoveUsergroup	Löschen einer neuen bestehenden Gruppe	Admin
UpdateUsergroup	Äderung der Parameter einer Gruppe	Admin
AssignUserGroup	Erstellung einer Beziehung zwischen den Users und einer Gruppe	Admin
AssignWorklfow	Erstellung einer Beziehung zwischen mehreren Gruppen und einem Workflow	Admin
RemoveUserGroup-Assignment	Entfernen einer Beziehung zwischen den Users und einer Gruppe	Admin
RemoveWorklfow-Assignment	Entfernen einer Beziehung zwischen einer Gruppe und einem Workflow	Admin

Anhang E Qualitätsmanagement: Testplan

Das Testen sollte als eigenes Projekt neben der Entwicklung des Programms angesehen werden. Der Testplan legt fest, wie bei dem Testprojekt „Com42Bill-Test“ vorgegangen wird. Zunächst wird in der Testprojektbeschreibung kurz das Ziel der Tests beschrieben und eine Unterteilung der Tests in Testphasen nach Art und Umfang der Tests vorgenommen. Der Testhintergrund beschreibt die Besonderheiten des zu testenden Systems, wie die eingesetzte Serversoftware und die Programmiersprache. In dem Abschnitt Testgegenstände werden die Testlinge und die Testfälle in Abhängigkeit zu den einzelnen Testphasen aufgeführt. Eine Teststrategie ist bereits durch die jeweilige Phase grob festgelegt, das Testszenario und die genauen Ziele des speziellen Tests werden erst bei der Einzeltestplanung bestimmt. Es werden danach konkrete Testziele aufgeführt und festgelegt, wo der Schwerpunkt der Tests liegen soll. Im Allgemeinen sind Testprojekte immer wirtschaftlichen Einschränkungen unterworfen. Die Einschränkungen für diesen Testplan werden in dem Abschnitt Testeinschränkungen konkretisiert. Da Tests nicht unendlich fortgesetzt werden können, werden im nächsten Abschnitt deshalb Kriterien definiert, die für das Ende der Tests erfüllt werden müssen. Die Tests müssen nachweisbare Ergebnisse liefern, die in den Testergebnisdokumenten festgehalten werden. Diese sind im entsprechenden Abschnitt aufgeführt. Die Aufgaben, die bei den Tests anfallen und die Verantwortlichkeiten für diese Aufgaben werden in den folgenden Abschnitten aufgezählt. Abschließend wird ein Zeitplan für die Durchführung der Tests festgelegt.

E.1 Beschreibung des Testplans

Das durch diesen Testplan beschriebene Testprojekt „Com42Bill-Test“ wird von der Projektgruppe 411 durchgeführt, um zu überprüfen, ob die implementierte Anwendung mit den Anforderungen an das Projekt übereinstimmt und ob die Anwendung die geforderten Aufgaben fehlerfrei ausführt.

E.2 Testprojektbeschreibung

Mit dem Testprojekt soll nachgewiesen werden, dass alle Aktivitäten, die in den entsprechenden Aktivitätsdiagrammen beschrieben sind, fehlerfrei vom System ausgeführt werden.

Das Testprojekt wird vom Qualitätsmanagement geleitet. Für die Tests werden abhängig von der jeweiligen Teststrategie Endanwender, Entwickler oder zusammengestellte Testgruppen unter der Aufsicht des Qualitätsmanagement verantwortlich sein.

Ein abschließender Systemtest ist als Gesamttest nicht ausreichend, da zum einen nicht alle Anwendungsfälle getestet werden können und zum anderen bei einem Gesamtsystemtest entdeckte Fehler schwierig zu lokalisieren sind. Deswegen wird das System nach einem inkrementellen, induktiven Testansatz in mehreren Phasen von Grund auf geprüft werden. In der ersten Testphase werden White-Box-Tests als Code-Reviews oder Build-In-Tests bei einzelnen Klassen durchgeführt, aufbauend auf diesen Tests werden in der nächsten Phase Black-Box-Tests bei den Komponenten durchgeführt. In der anschließenden Phase der Integrationstests werden die Komponenten durch nachrichtenbasierte Grey-Box-Tests geprüft. Dabei werden die Nachrichten an den Schnittstellen zwischen den zu testenden Komponenten kontrolliert. Die abschließende Phase des Systemtests stellt auch den Abnahmetest dar. Hierbei werden die durch die Aktivitätsdiagramme beschriebenen Funktionen der Applikation in einem Black-Box-Test getestet.

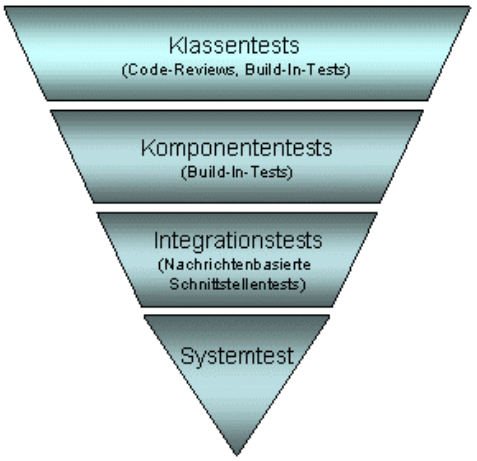


Abbildung 83: Testphasen

E.2.1: Testhintergrund

Das Software-System Com42Bill ist als Client/Server-System realisiert und setzt sich aus fünf Komponenten zusammen. Die Implementierung erfolgt in der Programmiersprache Java (Enterprise-Edition, J2EE 1.3). Als Middleware wird der Application Server BEA Weblogic 6.1 eingesetzt. Für die Persistierung der Daten wird auf das relationale Datenbanksystem Oracle 8i zurückgegriffen.

Der Testprozess soll bereits parallel zu der Entwicklung beginnen. Während der Implementierungsphase können bereits fertig gestellte Klassen getestet werden. Komponenten- und Integrationstests können erfolgen, sobald die ersten Komponenten vollständig implementiert sind. Der Systemtest kann dagegen erst erfolgen, wenn die Entwicklungsphase abgeschlossen ist.

E.2.2: Testumfang

- nur elementare Klassen; ca. 30% aller im System existierenden Klassen
- Fünf Komponenten
- 100% aller im System definierten Schnittstellen zwischen den einzelnen Komponenten
- Gesamtsystemtest

E.3 Testgegenstände

Bei den Klassentests werden folgende Klassen getestet:

a) Code-Review

- GUI: ausgewählte Klassen
- Sicherheit: ausgewählte Klassen
- Datenkonverter: ausgewählte Klassen
- Business Logic: ausgewählte Klassen
- Datenbank: ausgewählte Klassen

b) Build-In Test:

- GUI: Response-Presenter
- Sicherheit: SessionManager
- Datenkonverter: MsgReceiver, MsgExtractor
- Business Logic: InvoiceMgr
- Datenbank: PaymentTransaction

Bei den Komponententests werden alle Komponenten getestet:

- GUI
- Sicherheit
- Datenkonverter
- Business Logic
- Datenbank

Folgende Schnittstellen werden bei dem Top-Down-Integrationstest getestet:

- GUI – Business Logic
- GUI – Sicherheit
- Datenkonverter – Busines Logic
- Datenkonverter – Sicherheit
- GUI – Datenkonverter
- Business Logic – Datenbank

Im Systemtest wird das gesamte System Com42Bill getestet. Hierbei werden folgende Anwendungsfälle, die durch die Aktivitätsdiagramme beschrieben werden, untersucht:

a) Rechnungssteller:

- Registrieren
- Login
- Finanzkonto anlegen
- Finanzkonto löschen
- Finanzkonto editieren
- Rechnungsstellerkonto editieren
- Rechnungsdaten importieren

b) Rechnungsempfänger:

- Registrierung
- Login
- Verwaltung der persönlichen Daten
- Kontenverwaltung
- Rechnungsverwaltung
- Rechnungspräsentation

c) Betreiber (Sicherheit):

- Benutzer anlegen
- Benutzer editieren
- Benutzer löschen
- Gruppe anlegen
- Gruppe editieren
- Gruppe löschen

E.4 Testziele

Das allgemeine Testziel ist es, zu verifizieren, ob die an das System gestellten Erwartungen erfüllt werden. Ein Schwerpunkt sind hierbei die funktionalen Anforderungen an das System. Der Test soll sicherstellen, dass die zuvor definierten Anwendungsfälle vollständig und korrekt abgearbeitet werden. Desweiteren muss überprüft werden, ob die nicht-funktionalen Anforderungen einen bestimmten Erfüllungsgrad erreichen. Ein wichtiger Faktor ist an dieser Stelle die Zuverlässigkeit bzw. Robustheit des Systems.

Das Software System Com42Bill hat den Test bestanden, wenn die einzelnen Testzielkriterien den jeweils vorgegebenen Prozentsatz erfüllt haben:

- Funktionalität: >90%
- Korrektheit: >95%
- Robustheit: >90%

E.5 Testeinschränkungen

Die Leistungsbeschränkung des Servers und die maximal mögliche Anzahl von acht Clients im Versuchsaufbau ermöglichen keinen Belastungstest des Systems.

Aufgrund der geringen Zeit, die für das Testen zur Verfügung steht, wird auf einen vollständigen Klassentest verzichtet und stattdessen ein Klassentest nur exemplarisch an einigen elementaren Klassen durchgeführt werden. Aus dem gleichen Grund wird bei den Build-In-Tests auf eine vollständige Isolierung des Testlings, also ein Ersetzen aller von dem Testling benötigten Klassen durch Dummyklassen, verzichtet. Auf Klassen, die fast ausschließlich zur Speicherung von Daten dienen, darf in diesen Tests zugegriffen werden.

E.6 Teststrategie

Folgende Teststrategien werden bei den Tests verwendet:

- Klassentest: Statische Programmanalyse, Build-In-Tests
- Komponententest: Build-In-Tests
- Integrationstest: Top-Down-Strategie (nachrichtenbasiert)
- Systemtest: Anwendungsfallstrategie

E.7 Kriterien für das Ende des Tests

Um die Kriterien für das Testende zu beschreiben, sind folgende Begriffe wichtig:

- C_0-Überdeckung: Quotient aus der Anzahl der durchlaufenen Anweisungen und der Anzahl aller Anweisungen des Testlings
- C_1-Überdeckung: Quotient aus der Anzahl der durchlaufenen Zweige und der Anzahl aller Zweige des Testlings

Damit der Test als erfolgreich absolviert gelten kann, müssen die einzelnen Testphasen folgende Kriterien erfüllen.

- Klassentest: Aus Zeitgründen kann eine vollumfängliche Überprüfung der C_0-Überdeckung bzw. der C_1-Überdeckung in den Klassen nicht erfolgen. Stattdessen können die C_0-Überdeckung und C_1-Überdeckung nur stichprobenartig überprüft werden.
- Komponententest: Alle definierten Schnittstellen einer Komponente müssen mit korrekten und fehlerhaften Nachrichten angestoßen werden. Die Schnittstellen und die dafür definierten Nachrichtentypen ergeben sich aus der Spezifikation des Request Dictionaries der betroffenen Komponente.
- Integrationstest: Es müssen alle Nachrichten, die für die jeweilige Schnittstelle spezifiziert worden sind, getestet werden. Die Schnittstellen und die dafür definierten Nachrichtentypen ergeben sich aus der Spezifikation des Request Dictionaries der betroffenen Komponenten.
- Systemtest: Alle für das System spezifizierten Anwendungsfälle müssen durchlaufen werden (vgl. Anhang E.3).

E.8 Testergebnisse

Das Testprojekt liefert die folgenden Testdokumente. Sollte bei einem Test ein Fehler nachgewiesen werden, der eine weitere Behandlung des Testlings erfordern, so müssen die entsprechenden Korrekturen durch die betroffenen Entwickler durchgeführt werden. Der Test muss danach wiederholt werden.

- Testplan (dieses Dokument)
- Testentwurf
- Testfallspezifikation
- Testprozeduren
- Testobjektverzeichnis
- Testlog
- Testvorfallbericht
- Testabschlussbericht

E.9 Testaufgaben

In dem Testprojekt ergeben sich die im Folgenden beschriebenen Testaufgaben, die sich in die drei Bereiche „vorbereitende Aufgaben“, „Testausführung“ und „Testauswertung“ unterteilen lassen.

E.9.1: Testvorbereitungsaufgaben

- Testanforderungen ermitteln
- Testansatz festlegen
- Testszenarien ausarbeiten
- Kriterien für das Testende setzen
- Testfälle spezifizieren
- Testumgebung aufbauen

E.9.2: Testausführungsaufgaben

- Klassen-Code-Reviews durchführen
- Build-In-Klassentests durchführen
- Klassen abnehmen
- Build-In-Komponententests durchführen
- Komponenten abnehmen
- Integrations-/Schnittstellentest durchführen
- Komponentenintegration abnehmen
- Gesamtsystemtest durchführen
- Gesamtsystem abnehmen

E.9.3: Testauswertungsaufgaben

- Testprotokolle auswerten
- Testüberdeckung bestimmen
- Mängelbericht verfassen

- Testbericht schreiben

E.10 Testumgebungsanforderungen

An die Testumgebung werden folgende Anforderungen gestellt:

- Hardware: Zwei PCs (Client & Server), LAN
- Software: Betriebssystem Windows 2000, Java Entwicklungsumgebung Together Control Center 6.0, Application Server BEA Weblogic 6.1, Datenbank Oracle 8i
- Personell: Für die einzelnen Testphasen muss jeweils mind. ein Vertreter des Qualitätsmanagements und ein Vertreter des betroffenden Testlings anwesend sein.

E.11 Testverantwortlichkeiten

Im Folgenden wird die Aufgabenverantwortung festgelegt. Bei den Testaufgaben steht das Qualitätsmanagement, wie in der Testaufgabenverteilung zu sehen, den Entwicklern beratend zur Seite. Die testenden Entwickler können zwar ihre selbstgeschriebene Komponente Testen, jedoch sollte ein Entwickler seinen eigenen Code testen.

- Teststrategie ? Qualitätsmanagement
- Testumgebung ? Systemadministration
- Klassentests ? Entwickler
- Komponententests ? Entwickler
- Integrationstest ? Entwickler (Testgruppe)
- Systemtest ? Testgruppe
- Systemabnahme ? Projektbetreuer

E.12 Testaufgabenverteilung

Die Einzelaufgaben werden unter den Gruppen wie folgt verteilt:

- Testanforderungen ermitteln ? Qualitätsmanagement
- Testansatz festlegen ? Qualitätsmanagement
- Testszenarien ausarbeiten ? Entwickler, Qualitätsmanagement

- Testendekriterien setzen ? Qualitätsmanagement
- Testfälle spezifizieren ? Entwickler
- Testumgebung aufbauen ? Systemadministration
- Klassen-Code-Reviews durchführen ? Entwickler, QM, Beisitzer
- Build-In-Klassentests durchführen ? Entwickler
- Klassen abnehmen ? Qualitätsmanagement
- Build-In-Komponententests durchführen ? Entwickler
- Komponenten abnehmen ? Qualitätsmanagement
- Integrations-/Schnittstellentest ? Entwickler
- Komponentenintegration abnehmen ? Qualitätsmanagement
- Gesamtsystemtest durchführen ? Testgruppe
- Gesamtsystem abnehmen ? Projektbetreuer
- Testprotokolle auswerten ? Qualitätsmanagement
- Testüberdeckung bestimmen ? Qualitätsmanagement
- Mängelbericht verfassen ? Qualitätsmanagement
- Testbericht schreiben ? Qualitätsmanagement
- Fehlerbehebung ? Entwickler

E.13 Testzeitplan

Die Spezifikation der Testfälle und die Vorbereitung der Tests müssen spätestens bis zur Fertigstellung der ersten Klassen erfolgen, damit die ersten Klassentests umgehend durchgeführt werden können.

Die Klassentests erfolgen also parallel zur Implementierungsphase. Eine Klasse wird zwar schon während der Entwicklungsphase durch den Entwickler Tests unterzogen, jedoch wird sie erst als korrekt getestet angesehen, wenn die vollständig implementierte Klasse getestet wurde. Nach der vollständigen Implementierung einer Komponente erfolgt anschließend der Komponententest. Danach kann der Integrationstest durchgeführt. Zum Abschluss erfolgt der Systemtest.

Der folgende Zeitplan gibt einen groben Zeitrahmen vor, innerhalb dessen die einzelnen Testphasen spätestens abgeschlossen worden seien müssen. Einzelne Tests können je nach Fortschritt der Implementierung auch vorgezogen werden.

Nr.	Name	Zeitraum	Art des Tests	Testziel	Verantwortlichkeiten
1	DK-Review I	28.11.2002-02.12.2002	Code-Review (White-Box-Test)	Überprüfen des Codes einer ausgewählten Klasse auf Logikfehler auch unter besonderer Begutachtung von Sonderfällen. Check, ob der Codeguide eingehalten wurde	Zahir, Alexander
2	GUI-Review I	28.11.2002-02.12.2002	Code-Review	s. Nr. 1	Dino, Christian
3	Sicherheit-Review I	02.12.2002-09.12.2002	Code-Review	s. Nr. 1	Dennis, Alexander, Christian
4	BL-Review I	02.12.2002-09.12.2002	Code-Review	s. Nr. 1	Narcisse, Alexander, Christian
5	DB-Review I	28.11.2002-02.12.2002	Code-Review	s. Nr. 1	Andre, Alexander, Christian
6	DK-Klassentest I	02.12.2002-09.12.2002	Build-In-Klassentest (White-Box-Test)	Möglichst vollständiger Test der Funktionen einer ausgewählten Klasse durch Überdeckung möglichst aller Entscheidungsteilbäume der Funktion (C_1-Überdeckung)	Zahir, Christian
7	GUI-Klassentest I	02.12.2002-09.12.2002	Build-In-Klassentest	s. Nr. 6	Matthias
8	Sicherheit-Klassentest I	09.12.2002- 16.12.2002	Build-In-Klassentest	s. Nr. 6	Bastian
9	BL-Klassentest I	09.12.2002-16.12.2002	Build-In-Klassentest	s. Nr. 6	Timo, Alireza
10	DB-Klassentest I	02.12.2002- 09.12.2002	Build-In-Klassentest	s. Nr. 6	Andre
11	DK-Komponententest I	09.12.2002- 16.12.2002	Build-In-Komponententest (Black-Box-Test)	Test der Funktionen einer Komponente durch Aufruf der Komponente mit allen spezifizierten Nachrichten. Dabei sollen sowohl richtige als auch fehlerhafte Nachrichten verarbeitet werden. (Äquivalenzklassenabdeckung)	DK
12	GUI-Komponententest	09.12.2002-16.12.2002	Build-In-Komponententest	s. Nr. 11	GUI

	I				
13	Sicherheit-Komponententest I	16.12.2002- 23.12.2002	Build-In-Komponententest	s. Nr. 11	Sicherheit
14	BL-Komponententest I	16.12.2002- 23.12.2002	Build-In-Komponententest	s. Nr. 11	BL
15	DB-Komponententest I	09.12.2002- 16.12.2002	Build-In-Komponententest	s. Nr. 11	DB
16	Integrationstest I DK-Sicherheit	06.01.2003- 13.01.2003	Build-In-Integrationstest (Nachrichten-basierend) (Grey-Box-Test)	Check der Nachrichten, die zwischen den Komponenten ausgetauscht werden.	DK, Sicherheit
17	Integrationstest II GUI-Sicherheit	06.01.2003- 13.01.2003	Build-In-Integrationstest	s. Nr. 16	GUI, Sicherheit
18	Integrationstest III Sicherheit-DB	13.01.2003- 20.01.2003	Build-In-Integrationstest	s. Nr. 16	Sicherheit, DB
19	Integrationstest IV BL-DB	13.01.2003- 20.01.2003	Build-In-Integrationstest	s. Nr. 16	BL, DB
20	Integrationstest V GUI-BL	20.01.2003- 27.01.2003	Build-In-Integrationstest	s. Nr. 16	DK, BL
21	Integrationstest VI DK-BL	20.01.2003- 27.01.2003	Build-In-Integrationstest	s. Nr. 16	DK, BL
22	Gesamtsystemtest	27.01.2003- 03.02.2003	Black-Box-Test	Die durch die Aktivitäts-diagramme geforderten Anwendungsfälle werden getestet.	Testgruppe

E.14 Testrisiken und Notfallplan

Risiken bei den Tests sind ein unerwartet hoher Zeitaufwand bei der Planung und Durchführung der Tests zu Lasten der Entwicklung. Sollte sich ein solches Szenario abzeichnen, so müssen die Tests auf notwendige Bereiche gekürzt werden oder aber einige Tests ausgesetzt werden. Ein weiteres Risiko bei diesem Vorgehen ist es allerdings, dass der Aufwand, spät entdeckte Fehler zu beheben, sehr gross sein kann.

Das gleiche Vorgehen der Kürzung bzw. Aussetzung von Tests bietet sich an, wenn die Entwickler den Implementatierungszeitplan nicht einhalten können, so dass auch die geplanten Tests nicht rechtzeitig durchgeführt werden können.

E.15 Fazit

Aufgrund von Problemen bei der Vorabintegration der Komponenten in der Implementierungsphase wurde die zuvor festgelegte Teststrategie geändert. Wie geplant, wurden Klassenreviews, Klassentests, Integrationstests und ein abschliessender Systemtest durchgeführt. Die Komponententests, die zwischen den Klassen- und den Integrationstests angesetzt waren, wurden nicht durchgeführt. Zu dem dafür vorgesehenen Zeitpunkt bestand das Problem, dass die Komponenten im Rahmen von Vorabintegrationen ständig angepasst werden mussten und somit kein abschließender Zustand vorherrschte, der hätte getestet werden können.

Die Durchführung der Klassentests konnte bei der Komponente Business Logic nicht vollzogen werden. Bei einem Klassentest muss der Testling weitestgehend von anderen Klassen isoliert werden. Die Klassen der Business Logic können erst im Rahmen einer Workflowkonfiguration sinnvoll miteinander verknüpft werden. Separat betrachtet enthalten sie zu wenig Funktionalität, um getestet werden zu können.

Die Integrationstests waren ebenfalls Einschränkungen unterlegen. Zwischen den Komponenten Business Logic und Datenbank gab es keine einheitlichen Schnittstellen, an der Daten hätten abgegriffen werden können.

Aus Zeitgründen konnten die Tests in den einzelnen Testphasen nicht vollständig durchgeführt werden. Bei den Klassenreviews und Klassentests wurde jeweils exemplarisch von jeder Komponente nur eine Klasse getestet. Die Integrationstests wurden im Rahmen eines demonstrativen Tests durchgeführt, bei dem die im Systementwurf erstellten Anwendungsfälle zugrunde gelegt wurden. Ein destruktiver Test wäre zu aufwendig gewesen.

Bei der Durchführung des Testprojektes konnten in jeder Testphase Fehler gefunden werden. Aufgrund der erwähnten Einschränkungen hätte ein Großteil der Fehler aber wahrscheinlich früher aufgedeckt werden können. Da die Komponententests nicht durchgeführt wurden, konnten Fehler in den einzelnen Komponenten erst während der Integrationstests gefunden werden. Das hat die Integration erheblich verkompliziert. Ein Komponententeam musste auf ein anderes warten, während dieses die Fehler in ihrer Komponente beheben. Dadurch hat sich diese Phase stark in die Länge gezogen.

Auch wenn die einzelnen Testphasen eingeschränkt wurden, war das Durchführen des Testprojektes mit einem nicht zu unterschätzenden Aufwand verbunden. Es hat sich aber in jedem Fall gelohnt, da eine Vielzahl von Fehlern verschiedenster Schwierigkeitsgrade aufgedeckt und rechtzeitig behoben werden konnte.

Anhang F: Qualitätsmanagement: Reviews

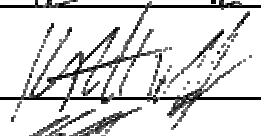
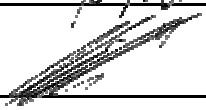
Im Folgenden werden die Berichte, die sich aus den durchgeführten Klassenreviews ergeben haben, dargestellt.

F.1 de.Com42Bill.components.security.Authentification.java

Review-Bericht	Review-Name: _Authentification_ Review-Nr.: _1_____ Seite _1___ von _2____ Datum: 13.12.2002 Zeit von: _9:50_ bis _10:20_
Testling Name / Version	Authentification.java
Referenzunterlagen Name / Version	Anforderungen
Bewertung / Maßnahmen	¦ akzeptiert ? keine Änderungen notwendig ? nicht akzeptiert ¦ kleine Änderungen

	? Review nicht beendet	? grosse Änderungen ? Überarbeitung

Review-Team

Name	Funktion	Unterschrift
Alexander Schmitz	Moderator	
Christian Kotthoff	Entwurfskontrolle	
Dennis Müller	Codierung	

Freigabe

Name	Datum	Unterschrift
Alexander Schmitz	13.12.2002	

Review-Nr.: _1____ Seite _2___ von _2____

Liste der Befunde

? im Testling

¦ in den Referenzunterlagen

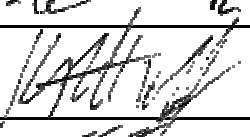
Nr.	Ort	Befund	Bewertung

1	S20	Anforderung entfällt	min/e
2	S23	Anforderung entfällt	min/e
3	S24	Anforderung entfällt	min/e
4	S25, S27	Wurden an die BL abgegeben	min/w
Bewertung: Fehlerart: **w**rong / **m**issing / **e**xtra Fehlertyp: **maj**or / **min**or			

F.2 de.Com42Bill.components.security.RequestAuthorizationBean.java

Review-Bericht	Review-Name: _RequestAuthorizationBean_ Review-Nr.: _1_____ Seite _1___ von _2____ Datum: 13.12.2002 Zeit von: _10:20_ bis _10:45_
Testling Name / Version	RequestAuthorizationBean.java
Referenzunterlagen Name / Version	Anforderungen
Bewertung / Maßnahmen	¦ akzeptiert ? nicht akzeptiert ? Review nicht beendet ¦ keine Änderungen notwendig ? kleine Änderungen ? grosse Änderungen ? Überarbeitung

Review-Team

Name	Funktion	Unterschrift
Alexander Schmitz	Moderator	
Christian Kotthoff	Entwurfskontrolle	
Dennis Müller	Codierung	

Freigabe

Name	Datum	Unterschrift
Alexander Schmitz	13.12.2002	

Review-Nr.: _1____ Seite _2___ von _2____

Liste der Befunde

? im Testling

? in den Referenzunterlagen

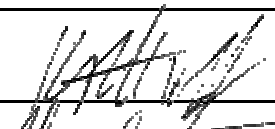
Nr.	Ort	Befund	Bewertung
1	GUI	Muss SessionID speichern und bei jedem folgenden Aufruf mitgeben	maj/m

Bewertung: Fehlerart: **w**rong / **m**issing / **e**xtra Fehlertyp: **maj**or / **min**or			

F.3 de.Com42Bill.components.gui.HistoryControllerBean.java

Review-Bericht	Review-Name: HistoryControllerBean__ Review-Nr.: _1____ Seite __1__ von __2___ Datum: 06.12.2003 Zeit von: _13:30_ bis _14:15_
Testling Name / Version	HistoryControllerBean.java Version 1.8
Referenzunterlagen Name / Version	Anforderungen
Bewertung / Maßnahmen	? akzeptiert ¦ nicht akzeptiert ? Review nicht beendet ? keine Änderungen notwendig ¦ kleine Änderungen ¦ grosse Änderungen ? Überarbeitung

Review-Team

Name	Funktion	Unterschrift
Alexander Schmitz	Moderator	
Christian Kotthoff	Entwurfskontrolle	
Dino Hasanbegovic	Codierung	

Freigabe

Name	Datum	Unterschrift
Alexander Schmitz	06.12.2002	

Review-Nr.: __1___ Seite __2__ von __2___

Liste der Befunde

¦	im Testling
?	in den Referenzunterlagen

Nr.	Ort	Befund	Bewertung
1	Zeile 235	Widerspricht G-062 (DB-Speicherung)	min/m
2	267-269	ExceptionHandling unzureichend	maj/m

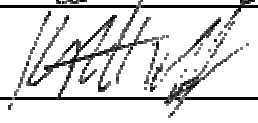
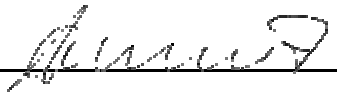
3	84-89	Dokumentation fehlt	min/m
4	198-199	Nicht alle Keys vorhanden, die ignoriert werden sollen	maj/m
5	207	Ignorieren bestimmter Keys fehlt	maj/m
6	219	SessionContext für den Lookup bei Passivate fehlt	maj/m
7	?	Backbuttenbehandlung (getLastPage) fehlt	maj/m
8	G-055	Anforderung entfällt, Logging als Extra vorgesehen	min/e
9	G-062	Anfoderung entspricht Gh, s. 1, wird nicht vollständig eingehalten	min/e

Bewertung: Fehlerart: **w**rong / **m**issing / **e**xtra Fehlertyp: **maj**or / **min**or

F.4 de.Com42Bill.components.dataconverter.impExpMgr.ImpExpMgrBean.java

Review-Bericht	Review-Name: ImpExpMgrBean__ Review-Nr.: _1____ Seite __1__ von __2___ Datum: 09.01.2003 Zeit von: _16:20_ bis _17:00_
Testling Name / Version	ImpExpMgrBean.java Version 1.13
Referenzunterlagen Name / Version	Anforderungen
Bewertung / Maßnahmen	? akzeptiert ¦ nicht akzeptiert ? Review nicht beendet ? keine Änderungen notwendig ¦ kleine Änderungen ? grosse Änderungen ? Überarbeitung

Review-Team

Name	Funktion	Unterschrift
Alexander Schmitz	Moderator	
Christian Kotthoff	Entwurfskontrolle	
Zahir Amiri	Codierung	

Freigabe

Name	Datum	Unterschrift
Alexander Schmitz	09.01.2003	

Review-Nr.: __1___ Seite __2__ von __2___

Liste der Befunde

¦ im Testling

? in den Referenzunterlagen

Nr.	Ort	Befund	Bewertung
1	Zeile 90-94	WorkflowIdentifier mit Konstanten arbeiten, die komponentenübergreifend genutzt werden können s. 1	min/mi

2	134-136	s. 1	min/mi
3	178-182	Aufruf des Workflowadapters fehlt (aus Testgründen auskommentiert)	min/mi
4	216-220	„Ratenzahlung“ als Konstante definieren, s. 1	maj/mi
5	274	Parameterübergabe an BL ungeklärt, über Formvalues? Zu klären!	min/mi
6	287-290	s. 1	maj/mi
7	294-296	s. 1	min/mi
8	328-331	s. 1	min/mi
9	362-364	s. 1	min/mi
10	452-453	s. 1	min/mi
11	509-511		min/mi

Bewertung: Fehlerart: **w**rong / **m**issing / **e**xtra Fehlertyp: **maj**or / **min**or

Anhang G Qualitätsmanagement: Klassentests

Im Folgenden werden die Ergebnisdokumente der durchgeführten Klassentests dargestellt. Zu jeder Methode werden alle Kombinationen von Eingabeparametern, die sich aus der Inspektion des Quellcodes der Methode ergeben haben, getestet. Für jede Eingabe wird die erwartete Ausgabe mit der Ausgabe des Testprogramms verglichen. Wenn diese nicht übereinstimmen, enthält die Methode einen Fehler.

G.1 de.Com42Bill.components.security.Authentification.java

Methode: isAuthentification(String startNode)

Eingabe:

Parameter	Wert
startNode	null

Erwartete Ausgabe:

Parameter	Wert
Com42BillSecurityException	RESULT_NULL_ARGUMENT

Beobachtete Log-Ausgabe:

```
### TEST ############################################################

### TEST ###: Calling Authentification.isAuthentification(null)

### TEST ### Result: Com42BillSecurityException:'RESULT_NULL_ARGUMENT'
```

Eingabe:

Parameter	Wert
startNode	„login“

Erwartete Ausgabe:

Parameter	Wert
Return-Wert	true

Beobachtete Log-Ausgabe:

```
### TEST ############################################################

### TEST ###: Calling Authentification.isAuthentification("login")
```

```
### TEST ###: Result: true
```

Eingabe:

Parameter	Wert
startNode	„logout“

Erwartete Ausgabe:

Parameter	Wert
Return-Wert	true

Beobachtete Log-Ausgabe:

```
### TEST ############################################################
### TEST ###: Calling Authentification.isAuthentification("logout")
### TEST ###: Result: true
```

Eingabe:

Parameter	Wert
startNode	„XXX“

Erwartete Ausgabe:

Parameter	Wert
Return-Wert	false

Beobachtete Log-Ausgabe:

```
### TEST ############################################################
### TEST ###: Calling Authentification.isAuthentification("XXX")
### TEST ###: Result: false
```

Methode: isLogin(String startNode)

Eingabe:

Parameter	Wert
startNode	null

Erwartete Ausgabe:

Parameter	Wert
Com42BillSecurityException	RESULT_NULL_ARGUMENT

Beobachtete Log-Ausgabe:

```
### TEST ############################################################

### TEST ###: Calling Authentification.isLogin(null)

### TEST ### Result: Com42BillSecurityException:'RESULT_NULL_ARGUMENT'
```

Eingabe:

Parameter	Wert
startNode	„login“

Erwartete Ausgabe:

Parameter	Wert
Return-Wert	true

Beobachtete Log-Ausgabe:

```
### TEST ############################################################

### TEST ###: Calling Authentification.isLogin("login")

### TEST ### Result: true
```

Eingabe:

Parameter	Wert

startNode	„XXX“

Erwartete Ausgabe:

Parameter	Wert
Return-Wert	false

Beobachtete Log-Ausgabe:

```
### TEST ############################################################

### TEST ###: Calling Authentification.isLogin("XXX")

### TEST ### Result: false
```

Methode: isLogout(String startNode)

Eingabe:

Parameter	Wert
startNode	null

Erwartete Ausgabe:

Parameter	Wert
Com42BillSecurityException	RESULT_NULL_ARGUMENT

Beobachtete Log-Ausgabe:

```
### TEST ############################################################

### TEST ###: Calling Authentification.isLogout(null)

### TEST ### Result: Com42BillSecurityException:'RESULT_NULL_ARGUMENT'
```

Eingabe:

Parameter	Wert
startNode	„logout“

Erwartete Ausgabe:

Parameter	Wert
Return-Wert	true

Beobachtete Log-Ausgabe:

```
### TEST ############################################################

### TEST ###: Calling Authentification.isLogout("logout")

### TEST ### Result: true
```

Eingabe:

Parameter	Wert
startNode	„XXX“

Erwartete Ausgabe:

Parameter	Wert
Return-Wert	false

Beobachtete Log-Ausgabe:

```
### TEST ############################################################

### TEST ###: Calling Authentification.isLogout("XXX")

### TEST ### Result: false
```

Methode: authenticate(Request request)

Eingabe:

Parameter	Wert
request	null

Erwartete Ausgabe:

Parameter	Wert

Com42BillSecurityException	RESULT_NULL_ARGUMENT

Beobachtete Log-Ausgabe:

```
### TEST ############################################################

### TEST ###: Calling Authentification.authenticate(null)

java.lang.NullPointerException

at de.Com42Bill.components.security.Authentification.authenticate (Authentification.java:106)

at de.Com42Bill.components.gui.RequestDispatcherBean.test (RequestDispatcherBean.java:344)

    at                              de.Com42Bill.components.gui.RequestDispatcherBean.doRequest
(RequestDispatcherBean.java:661)

    at                de.Com42Bill.components.gui.RequestDispatcherBean_v6pmbq_EOImpl.doRequest
(RequestDispatcherBean_v6pmbq_EOImpl.java:37)

    at de.Com42Bill.components.gui.RequestServlet.doPost (RequestServlet.java:211)

    at de.Com42Bill.components.gui.RequestServlet.doGet (RequestServlet.java:242)

    at javax.servlet.http.HttpServlet.service (HttpServlet.java:740)

    at javax.servlet.http.HttpServlet.service (HttpServlet.java:853)

    at weblogic.servlet.internal.ServletStubImpl.invokeServlet (ServletStubImpl.java:265)

    at weblogic.servlet.internal.ServletStubImpl.invokeServlet (ServletStubImpl.java:200)

    at                              weblogic.servlet.internal.WebAppServletContext.invokeServlet
(WebAppServletContext.java:2546)

    at weblogic.servlet.internal.ServletRequestImpl.execute (ServletRequestImpl.java:2260)

    at weblogic.kernel.ExecuteThread.execute (ExecuteThread.java:139)

    at weblogic.kernel.ExecuteThread.run (ExecuteThread.java:120)
```

Eingabe:

Parameter	Wert
request	
request.requestDictionary	null

Erwartete Ausgabe:

Parameter	Wert

Com42BillSecurityException	RESULT_NULL_ARGUMENT

Beobachtete Log-Ausgabe:

```
### TEST ############################################################

### TEST ###: Calling Authentification.authenticate(request)

request.requestDictionary = null

java.lang.NullPointerException

        at de.Com42Bill.components.security.Authentification.authenticate
(Authentification.java:109)

        at de.Com42Bill.components.gui.RequestDispatcherBean.test
(RequestDispatcherBean.java:393)

        at de.Com42Bill.components.gui.RequestDispatcherBean.doRequest
(RequestDispatcherBean.java:661)

        at de.Com42Bill.components.gui.RequestDispatcherBean_v6pmbq_EOImpl.doRequest
(RequestDispatcherBean_v6pmbq_EOImpl.java:37)

        at de.Com42Bill.components.gui.RequestServlet.doPost(RequestServlet.java:211)

        at de.Com42Bill.components.gui.RequestServlet.doGet(RequestServlet.java:242)

        at javax.servlet.http.HttpServlet.service(HttpServlet.java:740)

        at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)

        at weblogic.servlet.internal.ServletStubImpl.invokeServlet (ServletStubImpl.java:265)

        at weblogic.servlet.internal.ServletStubImpl.invokeServlet (ServletStubImpl.java:200)

        at weblogic.servlet.internal.WebAppServletContext.invokeServlet
(WebAppServletContext.java:2546)

        at weblogic.servlet.internal.ServletRequestImpl.execute (ServletRequestImpl.java:2260)

        at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139)

        at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120)
```

Eingabe:

Parameter	Wert
request	
request.requestDictionary(FV_USERINFORMATION_LOGIN)	null

Erwartete Ausgabe:

Parameter	Wert
Com42BillSecurityException	RESULT_NULL_ARGUMENT

Beobachtete Log-Ausgabe:

```
### TEST ############################################################

### TEST ###: Calling Authentification.authenticate(request)

request.requestDictionary("FV_USERINFORMATION_LOGIN"=null)

java.lang.NullPointerException

        at java.util.Hashtable.put(Hashtable.java:375)

        at de.Com42Bill.ebppsystem.core.request.RequestDictionary.putFormValue
(RequestDictionary.java:93)

        at de.Com42Bill.components.gui.RequestDispatcherBean.test
(RequestDispatcherBean.java:443)

        at de.Com42Bill.components.gui.RequestDispatcherBean.doRequest
(RequestDispatcherBean.java:661)

        at de.Com42Bill.components.gui.RequestDispatcherBean_v6pmbq_EOImpl.doRequest
(RequestDispatcherBean_v6pmbq_EOImpl.java:37)

        at de.Com42Bill.components.gui.RequestServlet.doPost(RequestServlet.java:211)

        at de.Com42Bill.components.gui.RequestServlet.doGet(RequestServlet.java:242)

        at javax.servlet.http.HttpServlet.service(HttpServlet.java:740)

        at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)

        at weblogic.servlet.internal.ServletStubImpl.invokeServlet (ServletStubImpl.java:265)

        at weblogic.servlet.internal.ServletStubImpl.invokeServlet (ServletStubImpl.java:200)

        at weblogic.servlet.internal.WebAppServletContext.invokeServlet
(WebAppServletContext.java:2546)

        at weblogic.servlet.internal.ServletRequestImpl.execute (ServletRequestImpl.java:2260)

        at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139)

        at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120)
```

Eingabe:

Parameter	Wert
request	

request.requestDictionary(FV_USERINFORMATION_LOGIN)	„XXX“ (ungültiger Userlogin)

Erwartete Ausgabe:

Parameter	Wert
Com42BillSecurityException	RESULT_WRONG_LOGIN

Beobachtete Log-Ausgabe:

```
### TEST ############################################################
### TEST ###: Calling Authentification.authenticate(request)
request.requestDictionary("FV_USERINFORMATION_LOGIN"="XXX")
### TEST ### Result: Com42BillSecurityException:'RESULT_WRONG_LOGIN'
```

Eingabe:

Parameter	Wert
request	
request.requestDictionary(FV_USERINFORMATION_LOGIN)	„Admin“ (gültiger Userlogin)
request.requestDictionary(FV_USERINFORMATION_PASSWORD)	„XXX“ (ungültiges Passwort)

Erwartete Ausgabe:

Parameter	Wert
Com42BillSecurityException	RESULT_WRONG_LOGIN

Beobachtete Log-Ausgabe:

```
### TEST ############################################################
### TEST ###: Calling Authentification.authenticate(request)
request.requestDictionary("FV_USERINFORMATION_LOGIN"="Admin")
request.requestDictionary("FV_USERINFORMATION_PASSWORD"="XXX")
### TEST ### Result: Com42BillSecurityException:'RESULT_WRONG_LOGIN'
```

Eingabe:

Parameter	Wert
request	
request.requestDictionary(FV_USERINFORMATION_LOGIN)	„Admin“ (gültiger Userlogin)
request.requestDictionary(FV_USERINFORMATION_PASSWORD)	„Com42Bill“ (gültiges Passwort)

Erwartete Ausgabe:

Parameter	Wert
Return-Wert	UserLocalHome

Beobachtete Log-Ausgabe:

```
### TEST ### Result: Com42BillSe

### TEST ############################################################

### TEST ###: Calling Authentification.authenticate(request)

request.requestDictionary("FV_USERINFORMATION_LOGIN"="Admin")

request.requestDictionary("FV_USERINFORMATION_PASSWORD"="Com42Bill")

### TEST ### Result: return UserInformationLocal
```

Bewertung der Beobachtungen:

Die als nicht schwerwiegend betrachteten Fehler des ersten Klassentests wurden ignoriert. Der Fehler, dass selbst gültige Benutzer nicht authentifiziert wurden, konnte auf die Encrypt-Methode zurückgeführt werden, mit der die Passworte der Benutzer verschlüsselt werden. Die Verschlüsselung erzeugte auch nicht druckbare Zeichen, die beim Abspeichern in der Datenbank zu einer Verkürzung der Strings führte, so dass der Vergleich der gespeicherten Passworte mit den übergebenen Fehlschlug. Die Encrypt-Methode wurde so modifiziert, dass jetzt nur noch druckbare Zeichen erzeugt werden.

G.2 de.Com42Bill.components.gui.RequestDispatcherBean.java

Methode: doRequest(Request request)

Eingabe:

Parameter	Wert
request	Null

Erwartete Ausgabe:

Parameter	Wert
Log-Ausgabe Exception	Stacktrace

Beobachtete Log-Ausgabe:

```
##########################################

##### Test doRequest START...

##### Eingabe:

##### request = NULL

#####

##### Ausgabe:

java.lang.NullPointerException

    at
de.Com42Bill.components.gui.RequestDispatcherBean.doRequest(RequestDispatcherBean.java:88)

##### Test doRequest ENDE...

##########################################
```

Eingabe:

Parameter	Wert
request.requestDicitionary	null

Erwartete Ausgabe:

Parameter	Wert
Log-Ausgabe Exception	Stacktrace

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary = NULL

#####

Ausgabe:

java.lang.NullPointerException

at de.Com42Bill.components.gui.RequestDispatcherBean.doRequest(RequestDispatcherBean.java:89)

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert
request.requestDicitionary(„SecurityFlag“)	RESULT_NULL_ARGUMENT

Erwartete Ausgabe:

Parameter	Wert
Log-Ausgabe Com42BillSecurityException	RESULT_NULL_ARGUMENT

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "RESULT_NULL_ARGUMENT"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): securityException:'RESULT_NULL_ARGUMENT'

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert
request.requestDicitionary(„SecurityFlag“)	RESULT_NO_WORKFLOWIDENTIFIER

Erwartete Ausgabe:

Parameter	Wert
Log-Ausgabe Com42BillSecurityException	RESULT_NO_WORKFLOWIDENTIFIER

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "RESULT_NO_WORKFLOWIDENTIFIER"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): securityException:'RESULT_NO_WORKFLOWIDENTIFIER'

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert
request.requestDicitionary(„SecurityFlag“)	RESULT_NO_STARTNODE

Erwartete Ausgabe:

Parameter	Wert

Log-Ausgabe Com42BillSecurityException	RESULT_NO_STARTNODE

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "RESULT_NO_STARTNODE"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): securityException:'RESULT_NO_STARTNODE'

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert
request.requestDicitionary(„SecurityFlag“)	RESULT_WRONG_LOGIN

Erwartete Ausgabe:

Parameter	Wert
Log-Ausgabe Com42BillSecurityException	RESULT_WRONG_LOGIN

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "RESULT_WRONG_LOGIN"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): securityException:'RESULT_WRONG_LOGIN'

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert
request.requestDicitionary(„SecurityFlag“)	RESULT_SESSION_NOT_IN_LIST

Erwartete Ausgabe:

Parameter	Wert
Log-Ausgabe Com42BillSecurityException	RESULT_SESSION_NOT_IN_LIST

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "RESULT_SESSION_NOT_IN_LIST"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): securityException:'RESULT_SESSION_NOT_IN_LIST'

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert
request.requestDicitionary(„SecurityFlag“)	RESULT_UNKNOWN_REQUESTSOURCE

Erwartete Ausgabe:

Parameter	Wert

Log-Ausgabe Com42BillSecurityException	RESULT_UNKNOWN_REQUESTSOURCE

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "RESULT_UNKNOWN_REQUESTSOURCE"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): securityException:'RESULT_UNKNOWN_REQUESTSOURCE'

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert
request.requestDicitionary(„SecurityFlag“)	RESULT_NO_INITIAL_CONTEXT

Erwartete Ausgabe:

Parameter	Wert
Log-Ausgabe Com42BillSecurityException	RESULT_NO_INITIAL_CONTEXT

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "RESULT_NO_INITIAL_CONTEXT"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): securityException:'RESULT_NO_INITAL_CONTEXT'

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert
request.requestDicitionary(„SecurityFlag“)	RESULT_NO_SESSION

Erwartete Ausgabe:

Parameter	Wert
Log-Ausgabe Com42BillSecurityException	RESULT_NO_SESSION

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "RESULT_NO_SESSION"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): securityException:'RESULT_NO_SESSION'

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert
request.requestDicitionary(„SecurityFlag“)	RESULT_WRONG_CLASS

Erwartete Ausgabe:

Parameter	Wert

Log-Ausgabe Com42BillSecurityException	RESULT_WRONG_CLASS

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "RESULT_WRONG_CLASS"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): securityException:'RESULT_WRONG_CLASS'

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert
request.requestDicitionary(„SecurityFlag“)	RESULT_CREATE_EXCEPTION

Erwartete Ausgabe:

Parameter	Wert
Log-Ausgabe Com42BillSecurityException	RESULT_CREATE_EXCEPTION

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "RESULT_CREATE_EXCEPTION"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): securityException:'RESULT_CREATE_EXCEPTION'

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert
request.requestDicitionary(„SecurityFlag“)	RESULT_UNKNOWN_REQUEST

Erwartete Ausgabe:

Parameter	Wert
Log-Ausgabe Com42BillSecurityException	RESULT_UNKNOWN_REQUEST

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "RESULT_UNKNOWN_REQUEST"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): securityException:'RESULT_UNKOWN_REQUEST'

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert
request.requestDicitionary(„SecurityFlag“)	RESULT_SHA_NOT_AVAILABLE

Erwartete Ausgabe:

Parameter	Wert
Log-Ausgabe Com42BillSecurityException	RESULT_SHA_NOT_AVAILABLE

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "RESULT_SHA_NOT_AVAILABLE"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): securityException:'RESULT_SHA_NOT_AVAILABLE'

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert
request.requestDicitionary(„ErrorList“)	ArrayList
request.requestDicitionary(„SecurityFlag“)	NO_SECURITY_EXCEPTION
request.Com42BillSessionLocal	Nicht NULL

Erwartete Ausgabe:

Parameter	Wert
Log-Ausgabe:	request.requestDicitionary.put(„FV_USERINFORMATION“) ausgeführt
Log-Ausgabe:	RequestDispatcherBean.doRequest(): errorList!=null -> Syntaxfehler vorhanden

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "NO_SECURITY_EXCEPTION"

requestDictionary("ErrorList") = "ArrayList"

request("Com42BillSessionBean") = "nicht NULL"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): requestDictionary.put("FV_USERINFORMATION", userInformation)

RequestDispatcherBean.doRequest(): errorList!=null -> Syntaxfehler vorhanden

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert

request.requestDicitionary(„ErrorList“)	ArrayList
request.requestDicitionary(„SecurityFlag“)	NO_SECURITY_EXCEPTION
request.Com42BillSessionLocal	NULL

Erwartete Ausgabe:

Parameter	Wert
Log-Ausgabe:	RequestDispatcherBean.doRequest(): errorList!=null -> Syntaxfehler vorhanden

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "NO_SECURITY_EXCEPTION"

requestDictionary("ErrorList") = "ArrayList"

request("Com42BillSessionBean") = "NULL"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): errorList!=null -> Syntaxfehler vorhanden

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert
request.requestDicitionary(„ErrorList“)	NULL
request.requestDicitionary(„SecurityFlag“)	NO_SECURITY_EXCEPTION
request.Com42BillSessionLocal	Nicht NULL
request.requestDicitionary(„RequestStatusFlag“)	GRANTED

Erwartete Ausgabe:

Parameter	Wert
Log-Ausgabe:	request.requestDicitionary.put(„FV_USERINFORMATION“) ausgeführt
Log-Ausgabe:	RequestDispatcherBean.doRequest(): checkRequest - HistoryController.GRANTED

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "NO_SECURITY_EXCEPTION"

requestDictionary("ErrorList") = "NULL"

requestDictionary("RequestStatusFlag") = "GRANTED"

request("Com42BillSessionBean") = "nicht NULL"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): requestDictionary.put("FV_USERINFORMATION", userInformation)

RequestDispatcherBean.doRequest(): checkRequest - HistoryController.GRANTED

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert
request.requestDicitionary(„ErrorList“)	NULL
request.requestDicitionary(„SecurityFlag“)	NO_SECURITY_EXCEPTION
request.Com42BillSessionLocal	Nicht NULL

request.requestDicitionary(„RequestStatusFlag“)	CACHED_GRANTED

Erwartete Ausgabe:

Parameter	Wert
Log-Ausgabe:	request.requestDicitionary.put(„FV_USERINFORMATION“) ausgeführt
Log-Ausgabe:	RequestDispatcherBean.doRequest(): checkRequest - HistoryController.CACHED_GRANTED

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "NO_SECURITY_EXCEPTION"

requestDictionary("ErrorList") = "NULL"

requestDictionary("RequestStatusFlag") = "CACHED_GRANTED"

request("Com42BillSessionBean") = "nicht NULL"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): requestDictionary.put("FV_USERINFORMATION", userInformation)

RequestDispatcherBean.doRequest(): checkRequest - HistoryController.CACHED_GRANTED

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert
request.requestDicitionary(„ErrorList“)	NULL
request.requestDicitionary(„SecurityFlag“)	NO_SECURITY_EXCEPTION

request.Com42BillSessionLocal	Nicht NULL
request.requestDicitionary(„RequestStatusFlag“)	FORBIDDEN

Erwartete Ausgabe:

Parameter	Wert
Log-Ausgabe:	request.requestDicitionary.put(„FV_USERINFORMATION“) ausgeführt
Log-Ausgabe:	RequestDispatcherBean.doRequest(): checkRequest - HistoryController.FORBIDDEN

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "NO_SECURITY_EXCEPTION"

requestDictionary("ErrorList") = "NULL"

requestDictionary("RequestStatusFlag") = "FORBIDDEN"

request("Com42BillSessionBean") = "nicht NULL"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): requestDictionary.put("FV_USERINFORMATION", userInformation)

RequestDispatcherBean.doRequest(): checkRequest - HistoryController.FORBIDDEN

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert
request.requestDicitionary(„ErrorList“)	NULL

request.requestDicitionary(„SecurityFlag“)	NO_SECURITY_EXCEPTION
request.Com42BillSessionLocal	NULL
request.requestDicitionary(„RequestStatusFlag“)	GRANTED

Erwartete Ausgabe:

Parameter	Wert
Log-Ausgabe:	RequestDispatcherBean.doRequest(): checkRequest - HistoryController.GRANTED

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "NO_SECURITY_EXCEPTION"

requestDictionary("ErrorList") = "NULL"

requestDictionary("RequestStatusFlag") = "GRANTED"

request("Com42BillSessionBean") = "NULL"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): checkRequest - HistoryController.GRANTED

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert
request.requestDicitionary(„ErrorList“)	NULL
request.requestDicitionary(„SecurityFlag“)	NO_SECURITY_EXCEPTION

request.Com42BillSessionLocal	NULL
request.requestDicitionary(„RequestStatusFlag“)	CACHED_GRANTED

Erwartete Ausgabe:

Parameter	Wert
Log-Ausgabe:	RequestDispatcherBean.doRequest(): checkRequest - HistoryController.CACHED_GRANTED

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "NO_SECURITY_EXCEPTION"

requestDictionary("ErrorList") = "NULL"

requestDictionary("RequestStatusFlag") = "CACHED_GRANTED"

request("Com42BillSessionBean") = "NULL"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): checkRequest - HistoryController.CACHED_GRANTED

Test doRequest ENDE...

##

Eingabe:

Parameter	Wert
request.requestDicitionary(„ErrorList“)	NULL
request.requestDicitionary(„SecurityFlag“)	NO_SECURITY_EXCEPTION
request.Com42BillSessionLocal	NULL

request.requestDicitionary(„RequestStatusFlag“)	FORBIDDEN

Erwartete Ausgabe:

Parameter	Wert
Log-Ausgabe:	RequestDispatcherBean.doRequest(): checkRequest - HistoryController.FORBIDDEN

Beobachtete Log-Ausgabe:

##

Test doRequest START...

Eingabe:

requestDictionary("SecurityFlag") = "NO_SECURITY_EXCEPTION"

requestDictionary("ErrorList") = "NULL"

requestDictionary("RequestStatusFlag") = "FORBIDDEN"

request("Com42BillSessionBean") = "NULL"

#####

Ausgabe:

RequestDispatcherBean.doRequest(): checkRequest - HistoryController.FORBIDDEN

Test doRequest ENDE...

##

Bewertung und Beobachtung:

Die geprüften Methoden funktionieren in allen untersuchten Fällen fehlerfrei. Die Ausgaben stimmen stets mit den erwarteten Ausgaben überein. Die erzeugten *NullpointerExceptions* können aufgrund der Interaktion mit den anderen Komponenten nicht auftreten, da niemals Null-Referenzen, sondern stets belegte Parameter übergeben werden.

G.3 de.Com42Bill.components.dataconverter.businessServiceInterface.AttachmentConstructorBean.java

Methode: createDefaultAttachmentpart()

Eingabe:

Parameter	Wert
keine	

Erwartete Ausgabe:

Parameter	Wert
AttachmentPart	Leere Objektinstanz

Beobachtete Log-Ausgabe:

##

Test createDefaultAttachmentpart START...

Eingabe:

keine

#####

Ausgabe:

Instanz von AttachmentPart wurde erzeugt.

Test createDefaultAttachmentpart ENDE...

Methode: createBankTransferData(PaymentTransactionLocal paymentTransaction)

Eingabe:

Parameter	Wert
paymentTransaction	Siehe unten
paymentTransaction.amount	999.99
paymentTransaction.paymentTransactionId	03020401
paymentTransaction.paymentReason	Com42Bill

	3020401
Daten des Auftraggebers:	
contractor.setAccountNo:	30204901
contractor_financialServiceProvider.setBlz	03020401
contractor_financialServiceProviderCompanyDataSheet.setCompanyName	Volksbank
contractor_PersonDataSheet.setFirstName	Peter
contractor_PersonDataSheet.setLastName	Meier
Daten des Empfängers:	
recipient.setAccountNo	30204701
r_financialServiceProvider.setBlz	03020401
r_invoicingPartyCompanyDataSheet.setCompanyName	Karstadt Sport
r_financialServiceProviderPartyCompanyDataSheet.setCompanyName	Sparkasse Dortmund

Erwartete Ausgabe:

Parameter	Wert
BankTransferData btd	Siehe unten
btd.paymentTransactionId	03020401
btd.orderingParty_name	Meier, Peter
btd.orderingParty_bankCode	30204901
btd.orderingParty_bank	Volksbank
btd.orderingParty_accountno	30204901
btd.recipient_name	Karstadt Sport
btd.recipient_bankCode	03020401
btd.recipient_bank	Sparkasse Dortmund
btd.recipient_accountno	30204701
btd.reasonForPayment	Com42Bill 3020401

btd.amount	999.99

Beobachtete Log-Ausgabe:

##

Ausgabe:

AttachmentConstructorBean.createBankTransferData(): start

AttachmentConstructorBean.createBankTransferData(): end

BankTransferData btd = attachmentConstructor.createBankTransferData(this.paymentTransaction);

Betrag = 999.99

Auftraggeber Kontonr.30204901

Auftraggeber Bank = Volksbank

Auftraggeber Bank BLZ = 03020401

Auftraggeber Name = Meier, Peter

Zahlungsgrund = Com42Bill 3020401

Empfõnger Kontonr. = 30204701

Empfõnger Bank = Sparkasse Dortmund

Empfõnger BLZ= 03020401

Empfõnger Name = Karstadt Sport

PaymentTransaction Id = 03020401

Eingabe:

Parameter	Wert
paymentTransaction	NULL

Erwartete Ausgabe:

Parameter	Wert
Exception	Stacktrace

Beobachtete Log-Ausgabe:

##

Test createBankTransferData START...

Eingabe:

PaymentTransactionLocal paymentTransaction = NULL

#####

Ausgabe:

java.lang.NullPointerException

at de.Com42Bill.components.dataconverter.businessServiceInterface.AttachmentConstructorBean.createBankTransferData(AttachmentConstructorBean.java:91)

Test createBankTransferData ENDE...

Methode: createAttachmentPart(Object obj)

Eingabe:

Parameter	Wert
obj	NULL

Erwartete Ausgabe:

Parameter	Wert
AttachmentPart	Leere Instanz

Beobachtete Log-Ausgabe:

##

Test createAttachmentPart START...

Eingabe:

Object obj = NULL

#####

Ausgabe:

AttachmentPart als XML Stream:

Test createAttachmentPart ENDE...

Eingabe:

Parameter	Wert
obj	Instanceof PaymentTransactionLocal = paymentTransaction
paymentTransaction.amount	999.99
paymentTransaction.paymentTransactionId	03020401
paymentTransaction.paymentReason	Com42Bill 3020401
Daten des Auftraggebers:	
contractor.setAccountNo:	30204901
contractor_financialServiceProvider.setBlz	03020401
contractor_financialServiceProviderCompanyDataSheet.setCompanyName	Volksbank
contractor_PersonDataSheet.setFirstName	Peter
contractor_PersonDataSheet.setLastName	Meier
Daten des Empfängers:	
recipient.setAccountNo	30204701
r_financialServiceProvider.setBlz	03020401
r_invoicingPartyCompanyDataSheet.setCompanyName	Karstadt Sport
r_financialServiceProviderPartyCompanyDataSheet.setCompanyName	Sparkasse Dortmund

Erwartete Ausgabe:

Parameter	Wert
AttachmentPart	Enthält die im Objekt „btd“ enthaltenen Daten in XML konvertiert
BankTransferData btd	Zwischenergebnis

btd.paymentTransactionId	03020401
btd.orderingParty_name	Meier, Peter
btd.orderingParty_bankCode	30204901
btd.orderingParty_bank	Volksbank
btd.orderingParty_accountno	30204901
btd.recipient_name	Karstadt Sport
btd.recipient_bankCode	03020401
btd.recipient_bank	Sparkasse Dortmund
btd.recipient_accountno	30204701
btd.reasonForPayment	Com42Bill 3020401
btd.amount	999.99

Beobachtete Log-Ausgabe:

##

Test createAttachmentPart START...

Eingabe:

Object obj = instanceof PaymentTransactionLocal (= paymentTransaction)

#####

Ausgabe:

<?xml version="1.0" encoding="UTF-8"?>

<bank-transfer-data valid="true"><ordering-party_name>Meier, Peter</ordering-party_name><ordering-party_account-nr>30204901</ordering-party_account-nr><reason-for-payment>Com42Bill 3020401</reason-for-payment><recipient_bank-code>03020401</recipient_bank-code><paym ent-transaction-id>03020401</paymen

t-transaction-id><recipient_account-nr>30204701</recipient_account-nr><recipient_name>Karstadt Sport</recipient_name><amount>999.99</amount><ordering-party_bank>Volksbank</ordering-party_bank><ordering-party_bank-code>03020401</ordering-party_bank-code><recipient_bank>Sparkasse Dortmund</recipient_b

ank></bank-transfer-data>

Test createAttachmentPart ENDE...

##

Eingabe:

Parameter	Wert
obj	Instanceof BillingDataWorkflowResultList = resultList
resultList.isSecurityException	FALSE
resultList.invoiceDataTypes.invoiceContainer.invoice(0). invoiceNr	123444
resultList.invoiceDataTypes.invoiceContainer.invoice(0). invoiceRecipientNr	123555
resultList.invoiceDataTypes.invoiceContainer.invoice(0). invoicingPartyNr	123666
resultList.status(invoice(0).invoiceNr)	ok
resultList.invoiceDataTypes.invoiceContainer.invoice(1). invoiceNr	321444
resultList.invoiceDataTypes.invoiceContainer.invoice(1). invoiceRecipientNr	321555
resultList.invoiceDataTypes.invoiceContainer.invoice(1). invoicingPartyNr	321666
resultList.status(invoice(1).invoiceNr)	ok
resultList.invoiceDataTypes.incoiceCancellationContainer. invoiceCancellation(0).invoiceNr	456777
resultList.invoiceDataTypes.incoiceCancellationContainer. invoiceCancellation(0).invoiceRecipientNr	456888

resultList.invoiceDataTypes.incoiceCancellationContainer. invoiceCancellation(0).invoicingPartyNr	456999
resultList.status(invoiceCancellation(0).invoiceNr	ok
resultList.invoiceDataTypes.incoiceCancellationContainer. invoiceCancellation(1).invoiceNr	654777
resultList.invoiceDataTypes.incoiceCancellationContainer. invoiceCancellation(1).invoiceRecipientNr	654888
resultList.invoiceDataTypes.incoiceCancellationContainer. invoiceCancellation(1).invoicingPartyNr	654999
resultList.status(invoiceCancellation(1).invoiceNr	ok

Erwartete Ausgabe:

Parameter	Wert
AttachmentPart	Enthält die im Objekt „bdc“ enthaltenen Daten in XML konvertiert
BillingDataConfirmation bdc	Zwischenergebnis
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer. confirmationInvoice(0).invoiceNr	123444
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer. confirmationInvoice(0).invoiceRecipientNr	123555
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer. confirmationInvoice(0).invoicingPartyNr	123666
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer. confirmationInvoice(0).status	ok
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer.	321444

confirmationInvoice(1).invoiceNr	
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer. confirmationInvoice(1).invoiceRecipientNr	321555
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer. confirmationInvoice(1).invoicingPartyNr	321666
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer. confirmationInvoice(1).status	ok
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer. confirmationInvoiceCancellation (0).invoiceNr	456777
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer. confirmationInvoiceCancellation (0).invoiceRecipientNr	456888
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer. confirmationInvoiceCancellation (0).invoicingPartyNr	456999
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer. confirmationInvoiceCancellation (0).status	ok
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer. confirmationInvoiceCancellation (1).invoiceNr	654777
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer. confirmationInvoiceCancellation (1).invoiceRecipientNr	654888
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer. confirmationInvoiceCancellation (1).invoicingPartyNr	654999
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer. confirmationInvoiceCancellation (1).status	ok

Beobachtete Log-Ausgabe:

##

Test createAttachmentPart START...

Ausgabe:

AttachmentPart als XML Stream:

<?xml version="1.0" encoding="UTF-8"?>

<billing-data-confirmation valid="true"><confirmation-invoice-data-types valid="true"><confirmation-invoice-cancellation-container valid="true" confirmation-invoice-cancellation-count="2"><c

onfirmation-invoice-cancellation valid="true"><status>ok</status><invoice-recipient-nr>456888</invoice-recipient-nr><invoice-nr>456777</invoice-nr><invoicing-party-nr>456999</invoicing-party

-nr></confirmation-invoice-cancellation><confirmation-invoice-cancellation valid="true"><status>ok</status><invoice-recipient-nr>654888</invoice-recipient-nr><invoice-nr>654777</invoice-nr><

invoicing-party-nr>654999</invoicing-party-nr></confirmation-invoice-cancellation></confirmation-invoice-cancellation-container><confirmation-invoice-container confirmation-invoice-count="2"

valid="true"><confirmation-invoice valid="true"><status>ok</status><invoice-recipient-nr>123555</invoice-recipient-nr><invoice-nr>123444</invoice-nr><invoicing-party-nr>123666</invoicing-pa

rty-nr></confirmation-invoice><confirmation-invoice valid="true"><status>ok</status><invoice-recipient-nr>321555</invoice-recipient-nr><invoice-nr>321444</invoice-nr><invoicing-party-nr>3216

66</invoicing-party-nr></confirmation-invoice></confirmation-invoice-container></confirmation-invoice-data-types><confirmation-invoice-message valid="true"/></billing-data-confirmation>

Test createAttachmentPart ENDE...

Eingabe:

Parameter	Wert
obj	Instanceof BillingDataWorkflowResultList = resultList
resultList.isSecurityException	TRUE

Erwartete Ausgabe:

Parameter	Wert

AttachmentPart	Enthält die im Objekt „bdc“ enthaltenen Daten in XML konvertiert
BillingDataConfirmation bdc	Zwischenergebnis
bdc.confirmationInvoiceMessage	Authentifizierung fehlgeschlagen !!!

Beobachtete Log-Ausgabe:

##

Test createAttachmentPart START...

Eingabe:

Object obj = instanceof BillingDataWorkflowResultList (= resultList)

resultList.setIsSecurityException(TRUE)

#####

Ausgabe:

BillingDataConfirmation bdc

bdc.confirmationInvoiceMessage = "Fehlerhafte Authentifizierung !!!"

AttachmentPart als XML Stream:

<?xml version="1.0" encoding="UTF-8"?>

<billing-data-confirmation valid="true"><confirmation-invoice-message valid="true"><status>Fehlerhafte Authentifizierung !!!</status></confirmation-invoice-message></billing-data-confirmatio

n>

Test createAttachmentPart ENDE...

Eingabe:

Parameter	Wert
obj	Instanceof InvoiceStatusWorkflowResultList = resultList

resultList.isSecurityException	FALSE
resultList.invoiceStatusContainer.invoiceStatus(0). invoiceNr	123444
resultList.invoiceStatusContainer.invoiceStatus (0). invoiceRecipientNr	123555
resultList.invoiceStatusContainer.invoiceStatus(0). invoicingPartyNr	123666
resultList.status(invoiceStatus(0).invoiceNr)	bezahlt
resultList.invoiceDataTypes.invoiceContainer.invoiceStatus(1). invoiceNr	321444
resultList.invoiceDataTypes.invoiceContainer.invoiceStatus(1). invoiceRecipientNr	321555
resultList.invoiceDataTypes.invoiceContainer.invoiceStatus(1). invoicingPartyNr	321666
resultList.status(invoiceStatus(1).invoiceNr)	1. Rate bezahlt

Erwartete Ausgabe:

Parameter	Wert
AttachmentPart	Enthält die im Objekt „bdsc“ enthaltenen Daten in XML konvertiert
BillingDataStatusConfirmation bdsc	Zwischenergebnis
bdsc.confirmationInvoiceStatusContainer. confirmationInvoiceStatus(0).invoiceNr	123444
bdsc.confirmationInvoiceStatusContainer. confirmationInvoiceStatus(0).invoiceRecipientNr	123555
bdsc.confirmationInvoiceStatusContainer. confirmationInvoiceStatus(0).invoicingPartyNr	123666

bdsc.confirmationInvoiceStatusContainer. confirmationInvoiceStatus(0).status	bezahlt
bdsc.confirmationInvoiceStatusContainer. confirmationInvoiceStatus(1).invoiceNr	321444
bdsc.confirmationInvoiceStatusContainer. confirmationInvoiceStatus(1).invoiceRecipientNr	321555
bdsc.confirmationInvoiceStatusContainer. confirmationInvoiceStatus(1).invoicingPartyNr	321666
bdsc.confirmationInvoiceStatusContainer. confirmationInvoiceStatus(1).status	1. Rate bezahlt

Beobachtete Log-Ausgabe:

##

Test createAttachmentPart START...

Ausgabe:

AttachmentPart als XML Stream:

<?xml version="1.0" encoding="UTF-8"?>

<billing-data-status-confirmation valid="true"><confirmation-invoice-status-container confirmation-invoice-status-count="2" valid="true"><confirmation-invoice-status valid="true"><status>ok<

/status><invoice-recipient-nr>123555</invoice-recipient-nr><invoice-nr>123444</invoice-nr><invoicing-party-nr>123666</invoicing-party-nr></confirmation-invoice-status><confirmation-invoice-s

tatus valid="true"><status>ok</status><invoice-recipient-nr>321555</invoice-recipient-nr><invoice-nr>321444</invoice-nr><invoicing-party-nr>321666</invoicing-party-nr></confirmation-invoice-

status></confirmation-invoice-status-container><confirmation-invoice-status-message valid="true"/></billing-data-status-confirmation>

Test createAttachmentPart ENDE...

Eingabe:

Parameter	Wert
obj	Instanceof InvoiceStatusWorkflowResultList = resultList
resultList.isSecurityException	TRUE

Erwartete Ausgabe:

Parameter	Wert
AttachmentPart	Enthält die im Objekt „bdsc“ enthaltenen Daten in XML konvertiert
BillingDataStatusConfirmation bdsc	Zwischenergebnis
bdsc.confirmationInvoiceMessage	Authentifizierung fehlgeschlagen !!!

Beobachtete Log-Ausgabe:

##

Test createAttachmentPart START...

Eingabe:

Object obj = instanceof InvoiceStatusWorkflowResultList (= resultList)

resultList.setIsSecurityException(TRUE)

#####

Ausgabe:

BillingDataStatusConfirmation bdsc

bdsc.confirmationInvoiceStatusMessage = "Fehlerhafte Authentifizierung !!!"

AttachmentPart als XML Stream:

<?xml version="1.0" encoding="UTF-8"?>

<billing-data-status-confirmation valid="true"><confirmation-invoice-status-message valid="true"><status>Fehlerhafte Authentifizierung !!!</status></confirmation-invoice-status-message></bil

ling-data-status-confirmation>

Test createAttachmentPart ENDE...

Methode: createBillingDataConfirmation(BillingDataWorkflowResultList resultlist)

Eingabe:

Parameter	Wert
resultList	Siehe unten
resultList.invoiceDataTypes.invoiceContainer.invoice(0). invoiceNr	123444
resultList.invoiceDataTypes.invoiceContainer.invoice(0). invoiceRecipientNr	123555
resultList.invoiceDataTypes.invoiceContainer.invoice(0). invoicingPartyNr	123666
resultList.status(invoice(0).invoiceNr)	ok
resultList.invoiceDataTypes.invoiceContainer.invoice(1). invoiceNr	321444
resultList.invoiceDataTypes.invoiceContainer.invoice(1). invoiceRecipientNr	321555
resultList.invoiceDataTypes.invoiceContainer.invoice(1). invoicingPartyNr	321666
resultList.status(invoice(1).invoiceNr)	ok
resultList.invoiceDataTypes.incoiceCancellationContainer. invoiceCancellation(0).invoiceNr	456777
resultList.invoiceDataTypes.incoiceCancellationContainer. invoiceCancellation(0).invoiceRecipientNr	456888
resultList.invoiceDataTypes.incoiceCancellationContainer. invoiceCancellation(0).invoicingPartyNr	456999

resultList.status(invoiceCancellation(0).invoiceNr	ok
resultList.invoiceDataTypes.incoiceCancellationContainer. invoiceCancellation(1).invoiceNr	654777
resultList.invoiceDataTypes.incoiceCancellationContainer. invoiceCancellation(1).invoiceRecipientNr	654888
resultList.invoiceDataTypes.incoiceCancellationContainer. invoiceCancellation(1).invoicingPartyNr	654999
resultList.status(invoiceCancellation(1).invoiceNr	ok

Erwartete Ausgabe:

Parameter	Wert
BillingDataConfirmation bdc	Siehe unten
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer. confirmationInvoice(0).invoiceNr	123444
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer. confirmationInvoice(0).invoiceRecipientNr	123555
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer. confirmationInvoice(0).invoicingPartyNr	123666
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer. confirmationInvoice(0).status	ok
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer. confirmationInvoice(1).invoiceNr	321444
bdc.confirmationInvoiceDataTyp es.confirmationInvoiceContainer. confirmationInvoice(1).invoiceRecipientNr	321555
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer. confirmationInvoice(1).invoicingPartyNr	321666

bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer. confirmationInvoice(1).status	ok
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer. confirmationInvoiceCancellation (0).invoiceNr	456777
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer. confirmationInvoiceCancellation (0).invoiceRecipientNr	456888
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer. confirmationInvoiceCancellation (0).invoicingPartyNr	456999
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer. confirmationInvoiceCancellation (0).status	ok
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer. confirmationInvoiceCancellation (1).invoiceNr	654777
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer. confirmationInvoiceCancellation (1).invoiceRecipientNr	654888
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer. confirmationInvoiceCancellation (1).invoicingPartyNr	654999
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer. confirmationInvoiceCancellation (1).status	ok

Beobachtete Log-Ausgabe:

##

Test createBillingDataConfirmation START...

Ausgabe:

BillingDataConfirmation bdc

#####
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer.confirmationInvoice(0).invoiceNr = "123444"

#####
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer.confirmationInvoice(0).invoiceRecipientNr = "123555"

#####
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer.confirmationInvoice(0).invoicingPartyNr = "123666"

#####
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer.confirmationInvoice(0).status = "ok"

#####
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer.confirmationInvoice(1).invoiceNr = "321444"

#####
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer.confirmationInvoice(1).invoiceRecipientNr = "321555"

#####
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer.confirmationInvoice(1).invoicingPartyNr = "321666"

#####
bdc.confirmationInvoiceDataTypes.confirmationInvoiceContainer.confirmationInvoice(1).status = "ok"

#####
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer.confirmationInvoiceCancellation(0).invoiceNr = "456777"

#####
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer.confirmationInvoiceCancellation(0).invoiceRecipientNr = "456888"

#####
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer.confirmationInvoiceCancellation(0).invoicingPartyNr = "456999"

#####
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer.confirmationInvoiceCancellation(0).status = "ok"

#####
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer.confirmationInvoiceCancellation(1).invoiceNr = "654777"

#####
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer.confirmationInvoiceCancellation(1).invoiceRecipientNr = "654888"

#####
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer.confirmationInvoiceCancellation(1).invoicingPartyNr = "654999"

#####
bdc.confirmationInvoiceDataTypes.confirmationInvoiceCancellationContainer.confirmationInvoiceCancellation(1).status = "ok"

Test createBillingDataConfirmation ENDE...

Eingabe:

Parameter	Wert
resultList	NULL

Erwartete Ausgabe:

Parameter	Wert
Exception	stacktrace

Beobachtete Log-Ausgabe:

##

Test createBillingDataConfirmation START...

Eingabe:

BillingDataWorkflowResultList resultList = NULL

#####

Ausgabe:

java.lang.NullPointerException

at
de.Com42Bill.components.dataconverter.businessServiceInterface.AttachmentConstructorBean.createBillingDataConfirmation(AttachmentConstructorBean.java:242)

Methode: createBillingDataStatusConfirmation(InvoiceStatusWorkflowResultList resultlist)

Eingabe:

Parameter	Wert
resultList	Siehe unten
resultList.invoiceStatusContainer.invoiceStatus(0). invoiceNr	123444
resultList.invoiceStatusContainer.invoiceStatus (0). invoiceRecipientNr	123555
resultList.invoiceStatusContainer.invoiceStatus(0). invoicingPartyNr	123666
resultList.status(invoiceStatus(0).invoiceNr)	bezahlt
resultList.invoiceDataTypes.invoiceContainer.invoiceStatus(1). invoiceNr	321444
resultList.invoiceDataTypes.invoiceContainer.invoiceStatus(1). invoiceRecipientNr	321555
resultList.invoiceDataTypes.invoiceContainer.invoiceStatus(1). invoicingPartyNr	321666
resultList.status(invoiceStatus(1).invoiceNr)	1. Rate bezahlt

Erwartete Ausgabe:

Parameter	Wert
BillingDataStatusConfirmation bdsc	Siehe unten
bdsc.confirmationInvoiceStatusContainer. confirmationInvoiceStatus(0).invoiceNr	123444
bdsc.confirmationInvoiceStatusContainer. confirmationInvoiceStatus(0).invoiceRecipientNr	123555

bdsc.confirmationInvoiceStatusContainer. confirmationInvoiceStatus(0).invoicingPartyNr	123666
bdsc.confirmationInvoiceStatusContainer. confirmationInvoiceStatus(0).status	bezahlt
bdsc.confirmationInvoiceStatusContainer. confirmationInvoiceStatus(1).invoiceNr	321444
bdsc.confirmationInvoiceStatusContainer. confirmationInvoiceStatus(1).invoiceRecipientNr	321555
bdsc.confirmationInvoiceStatusContainer. confirmationInvoiceStatus(1).invoicingPartyNr	321666
bdsc.confirmationInvoiceStatusContainer. confirmationInvoiceStatus(1).status	1. Rate bezahlt

Beobachtete Log-Ausgabe:

##

Test createBillingDataStatusConfirmation START...

Eingabe:

InvoiceStatusWorkflowResultList resultList)

resultList.invoiceStatusContainer.invoiceStatus0.setInvoiceNr("123444"

resultList.invoiceStatusContainer.invoiceStatus0.setInvoiceRecipientNr("123555"

resultList.invoiceStatusContainer.invoiceStatus0.setInvoicingPartyNr("123666"

resultList.put(invoiceStatus0.getInvoiceNr(), "ok"

resultList.invoiceStatusContainer.invoiceStatus1.setInvoiceNr("321444"

resultList.invoiceStatusContainer.invoiceStatus1.setInvoiceRecipientNr("321555"

resultList.invoiceStatusContainer.invoiceStatus1.setInvoicingPartyNr("321666"

resultList.put(invoiceStatus1.getInvoiceNr(), "ok"

#####

Ausgabe:

BillingDataStatusConfirmation bdsc

bdsc.confirmationInvoiceContainer.confirmationInvoiceStatus(0).invoiceNr = "123444"

bdsc.confirmationInvoiceContainer.confirmationInvoiceStatus(0).invoiceRecipientNr = "123555"

bdsc.confirmationInvoiceContainer.confirmationInvoiceStatus(0).invoicingPartyNr = "123666"

bdsc.confirmationInvoiceContainer.confirmationInvoiceStatus(0).status = "ok"

bdsc.confirmationInvoiceContainer.confirmationInvoiceStatus(1).invoiceNr = "321444"

bdsc.confirmationInvoiceContainer.confirmationInvoiceStatus(1).invoiceRecipientNr = "321555"

bdsc.confirmationInvoiceContainer.confirmationInvoiceStatus(1).invoicingPartyNr = "321666"

bdsc.confirmationInvoiceContainer.confirmationInvoiceStatus(1).status = "ok"

Test createBillingDataStatusConfirmation ENDE...

Eingabe:

Parameter	Wert
resultList	NULL

Erwartete Ausgabe:

Parameter	Wert
Exception	stacktrace

Beobachtete Log-Ausgabe:

##

Test createBillingDataStatusConfirmation START...

Eingabe:

InvoiceStatusWorkflowResultList resultList = NULL

Ausgabe:

java.lang.NullPointerException

at
de.Com42Bill.components.dataconverter.businessServiceInterface.AttachmentConstructorBean.createBillingDataStatusConfirmation(AttachmentConstructorBean.java:341)

Methode: createBillingDataSecurityFailureConfirmation()

Eingabe:

Parameter	Wert
keine	

Erwartete Ausgabe:

Parameter	Wert
BillingDataConfirmation bdc	Siehe unten
bdc.confirmationInvoiceMessage	Authentifizierung fehlgeschlagen !!!

Beobachtete Log-Ausgabe:

##

Test createBillingDataSecurityFailureConfirmation START...

Eingabe:

keine

#####

Ausgabe:

BillingDataConfirmation bdc

bdc.confirmationInvoiceMessage = "Fehlerhafte Authentifizierung !!!"

Test createBillingDataSecurityFailureConfirmation ENDE...

Methode: createBillingDataStatusSecurityFailureConfirmation()

Eingabe:

Parameter	Wert
keine	

Erwartete Ausgabe:

Parameter	Wert
BillingDataStatusConfirmation bdsc	Siehe unten
bdsc.confirmationInvoiceMessage	Authentifizierung fehlgeschlagen !!!

Beobachtete Log-Ausgabe:

##

Test createBillingDataStatusSecurityFailureConfirmation START...

Eingabe:

keine

#####

Ausgabe:

BillingDataStatusConfirmation bdsc

bdsc.confirmationInvoiceStatusMessage = "Fehlerhafte Authentifizierung !!!"

Test createBillingDataStatusSecurityFailureConfirmation ENDE...

##

ENDE TEST AttachmentConstructorBean.....

Bewertung und Beobachtung:

Die geprüften Methoden funktionieren in allen untersuchten Fällen fehlerfrei. Die Ausgaben stimmen stets mit den erwarteten Ausgaben überein. Die erzeugten *NullpointerExceptions* werden in jeder Methode abgefangen und an eine andere Klasse zur Behandlung durchgereicht. Aufgrund der Interaktion mit den anderen Komponenten werden aber keine Null-Referenzen, sondern stets belegte Parameter übergeben.

G.4 de.Com42Bill.ebppsystem.core.UserInformation.java

Anmerkung: *Die zum großen Teil durch Together automatisch generierten Funktionen zu testen ist sinnlos, daher wurde kein strikter Klassentest durchgeführt, sondern auch die Relationen zweier Entity-Beans.*

Bean-Generierung

(mit Relation zu einer Entity-Bean InvoiceRecipient)

Beobachtete Log-Ausgabe:

```
### TEST ############################################################

### TEST ###: Creating UserInformation

### TEST ###: setAddress("Mr.")

### TEST ###: setDescription("First user JohnDoe")

### TEST ###: setDisabled(false)

### TEST ###: setLogin("JohnDoe")

### TEST ###: setPassword("12345)

### TEST ###: setPasswordReminder("Count Numbers")

### TEST ###: setPasswordReminderEmail("JohnDoe@test.invalid")

### TEST ###: Creating InvoiceRecipient

### TEST ###: InvoiceRecipient.setCustomerNumber("123456");

### TEST ###: Setting Relation between UserInformation and

                  InvoiceRecipient
```

Methode: findByLogin

(Test mit Kontrolle der generierten Bean und der Relation)

Eingabe:

Parameter	Wert
login	„JohnDoe“

Erwartete Ausgabe:

Parameter	Wert
Return-Wert	UserInformationLocal

Beobachtete Log-Ausgabe:

```
### TEST ############################################################

### TEST ###: Find Userinformation by Login - findByLogin("JohnDoe")

### TEST ### Result: Found UserInformationLocal

### TEST ### Result: UserInformationLocal.getAddress() = Mr.

### TEST ### Result: UserInformationLocal.getDescription() = First user
                  JohnDoe

### TEST ### Result: UserInformationLocal.getDisabled() = false

### TEST ### Result: UserInformationLocal.getLogin() = JohnDoe

### TEST ### Result: UserInformationLocal.getPassword() = 123456

### TEST ### Result: UserInformationLocal.getPasswordReminder() = Count

### TEST ### Result: UserInformationLocal.getPasswordReminderEmail() =
                  JohnDoe@test.invalid

### TEST ### Checking if a relation to InvoiceRecipient exists:

### TEST ### Result: found InvoiceRecipient with CustomerNumber: 123456

### TEST ### Checking if the revert relation from UserInformation to
                  InvoiceRecipient exists:

### TEST ### Result: found UserInformation with Login: JohnDoe
```

Eingabe:

Parameter	Wert
login	„JOHNDOE“ (ungültiger Login)

Erwartete Ausgabe:

Parameter	Wert

Return-Wert	Null

Beobachtete Log-Ausgabe:

```
### TEST ############################################################

### TEST ###: Find Userinformation by Login - findByLogin("JOHNDOE")

javax.ejb.ObjectNotFoundException: Bean not found in 'findByLogin'.

at de.Com42Bill.ebppsystem.core.user.UserInformationBean_4n1tv3_
_WebLogic_CMP_RDBMS.ejbFindByLogin(UserInformationBean_4n1tv3__WebLo

gic_CMP_RDBMS.java:1338)

at java.lang.reflect.Method.invoke(Native Method)

at weblogic.ejb20.cmp.rdbms.RDBMSPersistenceManager.scalarFinder

(RDBMSPersistenceManager.java:212)

at weblogic.ejb20.manager.BaseEntityManager.scalarFinder

(BaseEntityManager.java:539)

at weblogic.ejb20.manager.BaseEntityManager.localScalarFinder

(BaseEntityManager.java:489)

at weblogic.ejb20.internal.EntityEJBLocalHome.finder

(EntityEJBLocalHome.java:377)

at de.Com42Bill.ebppsystem.core.user.UserInformationBean_4n1tv3_

LocalHomeImpl.findByLogin (UserInformationBean_4n1tv3_LocalHomeImpl.j

ava:157)
```

Eingabe:

(Test bei dem zunächst eine zweite Bean mit gleichem Login angelegt wird)

Parameter	Wert
login	„JohnDoe“

Erwartete Ausgabe:

Parameter	Wert

Return-Wert	Null

Beobachtete Log-Ausgabe:

```
### TEST ############################################################

### TEST ###: Creating UserInformation with Login "JohnDoe" again

### TEST ###: setAddress("Mr.")

### TEST ###: setDescription("Second user JohnDoe")

### TEST ###: setDisabled(false)

### TEST ###: setLogin("JohnDoe")

### TEST ###: setPassword("654321")

### TEST ###: setPasswordReminder("Count Numbers backwards")

### TEST ###: setPasswordReminderEmail("JohnDoe@test.invalid")

### TEST ############################################################

### TEST ###: Find Userinformation by Login - findByLogin("JohnDoe")

javax.ejb.FinderException: finder/ejbSelect 'findByLogin'has returned more
than one value.  We were expecting only ONE value.  See EJB Spec

10.5.6.1, 10.5.7.1

 at de.Com42Bill.ebppsystem.core.user.UserInformationBean_4n1tv3_

      _WebLogic_CMP_RDBMS.ejbFindByLogin(UserInformationBean_4n1tv3__WebLo

gic_CMP_RDBMS.java:1341)

 at java.lang.reflect.Method.invoke(Native Method)

 at weblogic.ejb20.cmp.rdbms.RDBMSPersistenceManager.scalarFinder

(RDBMSPersistenceManager.java:212)

 at weblogic.ejb20.manager.BaseEntityManager.scalarFinder

(BaseEntityManager.java:539)

 at weblogic.ejb20.manager.BaseEntityManager.localScalarFinder

(BaseEntityManager.java:489)

 at weblogic.ejb20.internal.EntityEJBLocalHome.finder
```

```
(EntityEJBLocalHome.java:377)
 at de.Com42Bill.ebppsystem.core.user.UserInformationBean_4n1tv3_Local
HomeImpl.findByLogin(UserInformationBean_4n1tv3_LocalHomeImpl.j
ava:157)
```

Löschen der Bean

(Es befindet sich nur eine Bean mit Login „John.Doe“ in der DB)

Beobachtete Log-Ausgabe:

```
### TEST ############################################################
### TEST ###: Find Userinformation by Login - findByLogin("JohnDoe")
### TEST ### Result: Found UserInformationLocal
### TEST ### Deleting User "John.Doe"
### TEST ###: Find Userinformation by Login - findByLogin("JohnDoe")
javax.ejb.ObjectNotFoundException: Bean not found in 'findByLogin'.
  at de.Com42Bill.ebppsystem.core.user.UserInformationBean_4n1tv3_
_WebLogic_CMP_RDBMS.ejbFindByLogin(UserInformationBean_4n1tv3__WebLo
gic_CMP_RDBMS.java:1338)
  at java.lang.reflect.Method.invoke(Native Method)
  at weblogic.ejb20.cmp.rdbms.RDBMSPersistenceManager.scalarFinder
(RDBMSPersistenceManager.java:212)
  at weblogic.ejb20.manager.BaseEntityManager.scalarFinder
(BaseEntityManager.java:539)
  at weblogic.ejb20.manager.BaseEntityManager.localScalarFinder
(BaseEntityManager.java:489)
  at weblogic.ejb20.internal.EntityEJBLocalHome.finder
(EntityEJBLocalHome.java:377)
  at de.Com42Bill.ebppsystem.core.user.UserInformationBean_4n1tv3_Local
```

```
HomeImpl.findByLogin(UserInformationBean_4n1tv3_LocalHomeImpl.j

ava:157)
```

Bewertung der Beobachtungen:

Die Methoden der getesteten Entity Beans funktionieren wie erwartet, bis auf eine Ausnahme: die ausgelösten FinderExceptions müssen bei Verwendung der Beans abgefangen und behandelt werden, besonders die vermutlich oft vorkommene „Bean not found“-Exception. Die „returned more than one value“-Exception bei einer N:1 oder 1:1-Relation hat zur Folge, dass bei Generierung von Beans überprüft werden muss, ob die neu zu generierende Bean solche Bedingungen verletzt. Es sollte noch geprüft werden, ob solche Fälle nicht durch Regeln in der DB abgefangen werden können. Der Test der Relation, in diesem Fall eine 1:1-Relation, war erfolgreich, der Vollständigkeit wegen wird in einem weiteren QM-Test noch eine M:N-Relation getestet werden.

Anhang H Qualitätsmanagement: Systemtest

Nachfolgend werden die Ergebnisse des Systemtests dargestellt.

Ausgeführte Aktion	Ergebnis
Rechnungsempfänger registrieren	
Einleitungsseite anzeigen	ok
Registrierungsformular anzeigen	ok
Benutzerdaten eingeben (stichprobenartiger Check der Syntaxkontrolle)	ok
Logindaten eingeben	ok
Kontoninformationen eingeben	ok
Kontrolle der Daten (Übersicht)	ok
Bestätigungsmeldung der Registrierung	ok
Rechnungsempfänger anmelden	
Logindaten eingeben (Quick Login)	ok
Logindaten eingeben (Normales Login)	ok
Anzeige der Übersichtsseite „MyCom42Bill“	ok
Persönliche Daten editieren	
Persönliche Daten anzeigen	ok
Änderungen speichern	ok
Kontodaten editieren	
Liste der Konten anzeigen	ok
Konto editieren	ok

Editiertes Konto speichern	ok
Neues Konto anlegen	ok
Neues Konto speichern	ok
Rechnungen importieren	
Rechnung importieren	ok
Rechnungsübersicht	
Sortieren nach Fälligkeitskriterien	ok, außer Sortieren nach Rechnungsstellern
Rechnungen zur Bezahlung auswählen	ok
Bezahlvorgang starten	
Anzeige des Rechnungskorbs	ok
Entfernen einer Rechnung aus dem Rechnungskorb	ok
Kontoauswahl	ok
Bestätigung des Zahlungsauftrages	ok
Rechnungsstatus importieren	
Rechnungsstatus importieren	ok
Rechnung stornieren	
Storno importieren	ok

Anhang I Verzeichnisse

I.1 Abbildungsverzeichnis

Sofern in den Grafiken nicht anders gekennzeichnet, wurden die Abbildungen durch die Autoren selbst erstellt.

I.2 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

/99/ T. Bühren, M. Cakir, E. Can, A. Dombrowski, G. Geist, V. Gruhn, M. Gürgrn, S. Handschumacher, M. Heller, C. Lüer, D. Peters, G. Vollmer, U. Wellen, J. von Werne
Endbericht der Projektgruppe eCCo (PG 315)
Electronic Commerce in der Versicherungsbranche
Beispielhafte Unterstützung verteilter Geschäftsprozesse
Februar 1999

/100/ A. Fronk, J. Pleumann,
Der DoDL-Compiler
August 1999

/101/ K. Alfert, E.-E. Doberkat, C. Kopka
Towards Constructing a Flexible Multimedia Environment for Teaching the History of Art
September 1999

/102/ E.-E. Doberkat
An Note on a Categorial Semantics for ER-Models
November 1999

/103/ Christoph Begall, Matthias Dorka, Adil Kassabi, Wilhelm Leibel, Sebastian Linz, Sascha Lüdecke, Andreas Schröder, Jens Schröder, Sebastian Schütte, Thomas Sparenberg, Christian Stücke, Martin Uebing, Klaus Alfert, Alexander Fronk, Ernst-Erich Doberkat
Abschlußbericht der Projektgruppe PG-HEU (326)
Oktober 1999

/104/ Corina Kopka
Ein Vorgehensmodell für die Entwicklung multimedialer Lernsysteme
März 2000

/105/ Stefan Austen, Wahid Bashirazad, Matthais Book, Traugott Dittmann, Bernhard Flechtker, Hassan Ghane, Stefan Göbel, Chris Haase, Christian Leifkes, Martin Mocker, Stefan Puls, Carsten Seidel, Volker Gruhn, Lothar Schöpe, Ursula Wellen
Zwischenbericht der Projektgruppe IPSI
April 2000

/106/ Ernst-Erich Doberkat
Die Hofzwerge — Ein kurzes Tutorium zur objektorientierten Modellierung
September 2000

/107/ Leonid Abelev, Carsten Brockmann, Pedro Calado, Michael Damatow, Michael Heinrichs, Oliver Kowalke, Daniel Link, Holger Lümkemann, Thorsten Niedzwetzki, Martin Otten, Michael Rittinghaus, Gerrit Rothmaier
Volker Gruhn, Ursula Wellen
Zwischenbericht der Projektgruppe Palermo
November 2000

/108/ Stefan Austen, Wahid Bashirazad, Matthais Book, Traugott Dittmann, Bernhard Flechtker, Hassan Ghane, Stefan Göbel, Chris Haase, Christian Leifkes, Martin Mocker, Stefan Puls, Carsten Seidel, Volker Gruhn, Lothar Schöpe, Ursula Wellen
Endbericht der Projektgruppe IPSI
Februar 2001

/109/ Leonid Abelev, Carsten Brockmann, Pedro Calado, Michael Damatow, Michael Heinrichs, Oliver Kowalke, Daniel Link, Holger Lümkemann, Thorsten Niedzwetzki, Martin Otten, Michael Rittinghaus, Gerrit Rothmaier
Volker Gruhn, Ursula Wellen
Zwischenbericht der Projektgruppe Palermo
Februar 2001

/110/ Eugenio G. Omodeo, Ernst-Erich Doberkat
Algebraic semantics of ER-models from the standpoint of map calculus.
Part I: Static view
März 2001

/111/ Ernst-Erich Doberkat
An Architecture for a System of Mobile Agents
März 2001

/112/ Corina Kopka, Ursula Wellen
Development of a Software Production Process Model for Multimedia CAL Systems by Applying Process Landscaping
April 2001

/113/ Ernst-Erich Doberkat
The Converse of a Probabilistic Relation
Oktober 2002

/114/ Ernst-Erich Doberkat, Eugenio G. Omodeo
Algebraic semantics of ER-models in the context of the calculus of relations.
Part II: Dynamic view
Juli 2001

/115/ Volker Gruhn, Lothar Schöpe (Eds.)
Unterstützung von verteilten Softwareentwicklungsprozessen durch integrierte Planungs-, Workflow- und Groupware-Ansätze
September 2001

/116/ Ernst-Erich Doberkat
The Demonic Product of Probabilistic Relations
September 2001

/117/ Klaus Alfert, Alexander Fronk, Frank Engelen
Experiences in 3-Dimensional Visualization of Java Class Relations
September 2001

/118/ Ernst-Erich Doberkat
The Hierarchical Refinement of Probabilistic Relations
November 2001

/119/ Markus Alvermann, Martin Ernst, Tamara Flatt, Urs Helmig, Thorsten Langer, Ingo Röpling,
Clemens Schäfer, Nikolai Schreier, Olga Shtern
Ursula Wellen, Dirk Peters, Volker Gruhn
Project Group Chairware Intermediate Report
November 2001

/120/ Volker Gruhn, Ursula Wellen
Autonomies in a Software Process Landscape
Januar 2002

/121/ Ernst-Erich Doberkat, Gregor Engels (Hrsg.)
Ergebnisbericht des Jahres 2001
des Projektes “MuSofT – Multimedia in der SoftwareTechnik”
Februrar 2002

/122/ Ernst-Erich Doberkat, Gregor Engels, Jan Hendrik Hausmann, Mark Lohmann, Christof Veltmann
Anforderungen an eine eLearning-Plattform — Innovation und Integration —
April 2002

/123/ Ernst-Erich Doberkat
Pipes and Filters: Modelling a Software Architecture Through Relations
Juni 2002

/124/ Volker Gruhn, Lothar Schöpe
Integration von Legacy-Systemen mit Eletronic Commerce Anwendungen
Juni 2002

/125/ Ernst-Erich Doberkat
A Remark on A. Edalat’s Paper *Semi-Pullbacks and Bisimulations in Categories of Markov-Processes*
Juli 2002

/126/ Alexander Fronk
Towards the algebraic analysis of hyperlink structures
August 2002

/127/ Markus Alvermann, Martin Ernst, Tamara Flatt, Urs Helmig, Thorsten Langer
Ingo Röpling, Clemens Schäfer, Nikolai Schreier, Olga Shtern
Ursula Wellen, Dirk Peters, Volker Gruhn
Project Group Chairware Final Report
August 2002

/128/ Timo Albert, Zahir Amiri, Dino Hasanbegovic, Narcisse Kemogne Kamdem,
Christian Kotthoff, Dennis Müller, Matthias Niggemeier, Andre Pavlenko, Stefan Pinschke,
Alireza Salemi, Bastian Schlich, Alexander Schmitz
Volker Gruhn, Lothar Schöpe, Ursula Wellen
Zwischenbericht der Projektgruppe Com42Bill (PG 411)
September 2002

/129/ Alexander Fronk
An Approach to Algebraic Semantics of Object-Oriented Languages
Oktober 2002

/130/ Ernst-Erich Doberkat
Semi-Pullbacks and Bisimulations in Categories of Stochastic Relations
November 2002

/131 Yalda Ariana, Oliver Effner, Marcel Gleis, Martin Krzysiak,
Jens Lauert, Thomas Louis, Carsten Röttgers, Kai Schwaighofer,
Martin Testrot, Uwe Ulrich, Xingguang Yuan
Prof. Dr. Volker Gruhn, Sami Beydeda
Endbericht der PG nightshift:
Dokumentation der verteilten Geschäftsprozesse im FBI und Umsetzung von Teilen dieser Prozesse im Rahmen eines FBI-Intranets basierend auf WAP- und Java-Technologie
Februar 2003

/132/ Ernst-Erich Doberkat, Eugenio G. Omodeo
ER Modelling from First Relational Principles
Februar 2003

/133/ Klaus Alfert, Ernst-Erich Doberkat, Gregor Engels (Hrsg.)
Ergebnisbericht des Jahres 2002 des Projektes “MuSofT – Multimedia in der SoftwareTechnik”
März 2003

/134/ Ernst-Erich Doberkat
Tracing Relations Probabilistically
März 2003

/135/ Timo Albert, Zahir Amiri, Dino Hasanbegovic, Narcisse Kemogne Kamdem,
Christian Kotthoff, Dennis Müller, Matthias Niggemeier,
Andre Pavlenko, Alireza Salemi,Bastian Schlich, Alexander Schmitz,
Volker Gruhn, Lothar Schöpe, Ursula Wellen
Endbericht der Projektgruppe Com42Bill (PG 411)
März 2003