UNIVERSITÄT DORTMUND

Fachbereich Informatik
Lehrstuhl für Software-Technologie

M E M O Nr. 128

Zwischenbericht der Projektgruppe Com42Bill (PG 411)

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

September 2002

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

Zwischenbericht der Projektgruppe

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

Version 1.0 vom 23.09.2002

Inhalt

1 Einleitung 6

2 Was ist eine Projektgruppe? 8

3 Seminarphase 9
3.1 Allgemein 9
3.2 Softwareentwicklungsprozesse 9
3.3 Applikationsserver 10
3.4 Konfigurationsverwaltung und Projektdokumentation 11
3.5 Telematik-, Mobilfunk- und Netzwerktechniken 12
3.6 XML-basierte Datenaustauschformate 12
3.7 Mobile Commerce 13
3.8 Sicherheit 14
3.9 E-Bill Anwendungen 14

4 Das Modell für Com42Bill 16

5 Anforderungen 17

6 Die Architektur 19
6.1 Allgemein 19
6.2 Komponente GUI 19
6.3 Komponente Sicherheit 22
6.4 Komponente Business Logic 24
6.5 Komponente Datenkonverter 28
6.6 Komponente Datenbank 30

7 Funktionalität 33

8 Der Prototyp 34
8.1 Allgemein 34
8.2 Szenario 1: Rechnungsdaten vom Rechnungssteller empfangen 34
8.3 Szenario 2: Rechnungsdaten eines Rechnungsempfängers darstellen 34
8.4 Szenario 3: Bezahlvorgang ausführen 35
8.5 Zielsetzung 35

9 Das Datenmodell 36

10 Hard- und Softwareinfrastruktur 54

11 Projektmanagement 55

12 Qualitätsmanagement 58
12.1 Allgemein 58
12.2 Prozessmodell 58
12.3 Symbolik 58
12.4 Entwicklungsprozess 59
12.5 Anforderungsanalyse 61
12.6 Anforderungen 62
12.7 Modelle 62
12.8 Technikentscheidung 63
12.9 Auswahl von Key-Features 64
12.10 GUI-Design 64
12.11 Klassenmodell 65
12.12 Testplan 65
12.13 Prototyping 66
12.14 Implementierung 67

13 Ausblick 69

Anhang A : Rollenmodell 70
A.1 : Einleitung 70
A.2 : Rechnungsempfänger 71
A.3 : Finanzdienstleister 71
A.4 : Rechnungssteller 72

Anhang B : Use Cases 73
B.1 : Rechnungssteller 73
B.2 : Rechnungsempfänger 73
B.3 : Betreiber 74
B.3.1: Allgemeine Verwaltung 74
B.3.2: Konfiguration Datenaustausch 75
B.4 : Benutzerverwaltung 75

Anhang C : Aktivitätsdiagramme 76
C.1 : Rechnungsempfänger 76
C.1.1: Registrierung 76
C.1.2: Anmeldung 77
C.1.3: Verwaltung 78
C.1.4: Kontenverwaltung 79

C.1.5: Rechnungspräsentation 80
C.1.6: Mahnungen 80
C.1.7: Mahnungen prüfen 81
C.1.8: Zahlung 82
C.1.9: Rechnungsstatus / Zahlungsstatus 83
C.2 : Rechnungssteller 84
C.2.1: Login 84
C.2.2: Registrierung 85
C.2.3: Finanzkonto anlegen 86
C.2.4: Finanzkonto editieren 86
C.2.5: Finanzkonto löschen 87
C.2.6: Rechnungsstellerkonto editieren 87
C.2.7: Kunden-ID überprüfen 88
C.3 Betreiber 89
C.3.1: Bericht abrufen 89
C.3.2: Statistik abrufen 90
C.3.3: Job anlegen 91
C.3.4: Job ausführen 92
C.3.5: Job löschen 93
C.3.6: Zeitgesteuerte Transaktionen ausführen 94
C.4 : Datenkonverter 95
C.4.1: Login 95
C.4.2: Neues CPA / CPP 96
C.4.3: Ändern 96
C.4.4: Hinzufügen 97
C.4.5: Löschen 97
C.4.6: Struktur eines CPA / CPP bearbeiten 98

Anhang D : Anforderungsliste 99
D.1 : GUI 99
D.2 : Sicherheit 114
D.3 : Business Logic 119
D.4 : Datenkonverter 125
D.5 : Datenbank 131

Anhang E : Projektplan 136
E.1 : Kennenlernfahrt & Seminarphase 136
E.2 : Komponente Datenkonverter 136
E.3 : Komponente GUI 136

E.4 : Komponente Sicherheit 137
E.5 : Komponente Business Logic 138
E.6 : Komponente Datenbank 139
E.7 : Integrationstest der Komponenten 140
E.8 : Querschnittsaufgaben 140

Anhang F : Sicherheitsrichtlinien 142

Anhang G : Verzeichnisse 143
G.1 : Abbildungsverzeichnis 143
G.2 : Literaturverzeichnis 145

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 grenzübergreifende 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.

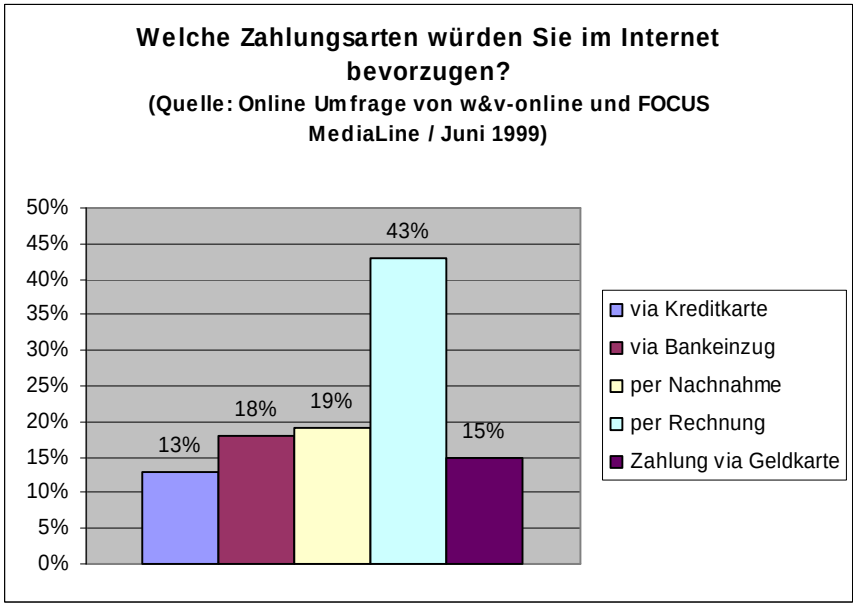

Abbildung 1: Welche Zahlungsarten würden Sie im Internet bevorzugen?

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. 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 Zwischenbericht beschreibt die bisher geleistete Arbeit der Projektgruppe Com42Bill, die die obigen Überlegungen in ein Softwaresystem umsetzt. Des weiteren wird der Aufbau und die Organisation dieser Projektgruppe beschrieben.

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 praktischen 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 alleine lösen zu können, muss das Probleme 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 sein 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 explorativ, 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).

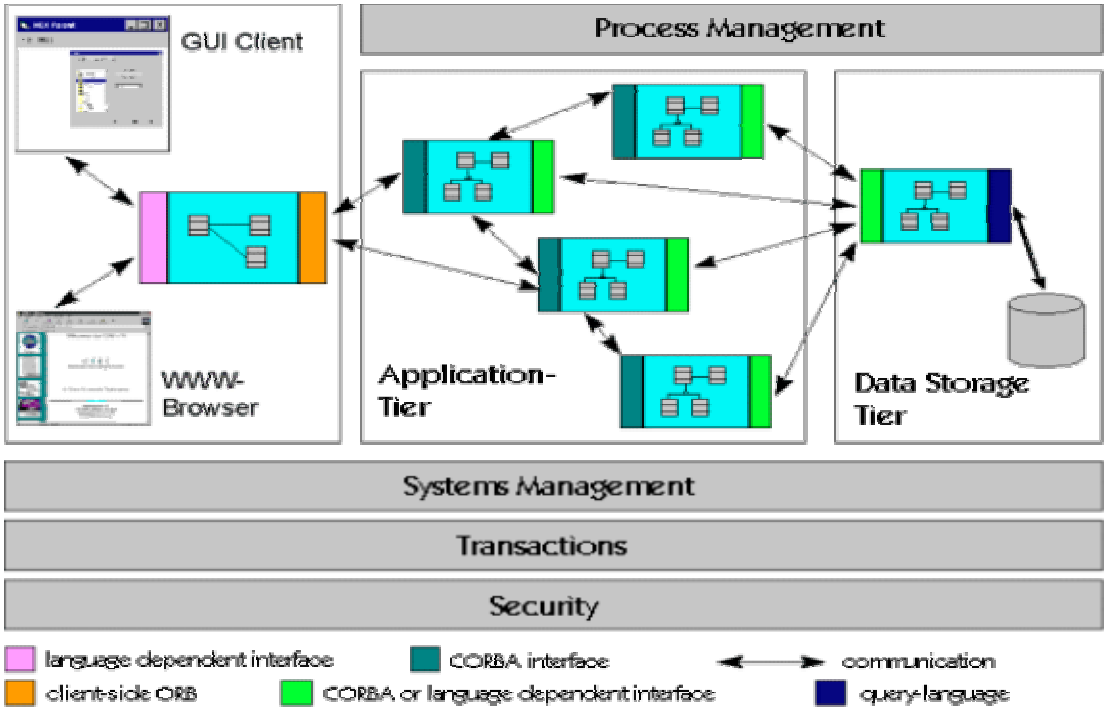

Abbildung 2: Aufbau eines 3- und mehrschichtigen Systems

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 gehört 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 E-Commerce 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 wurde 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 die Wahl auf den Thin Consolidator, der den Austausch und die Verarbeitung nur der notwendigsten Rechnungsdaten vorsieht. Im Vergleich zum Thick Consolidator fallen weniger Daten an, was die Datenhaltungs- und Entwicklungskosten senkt. Eine weitere Stärke des Thin Consolidator-Modells ist, dass der direkte Kontakt des Rechnungsstellers zum Rechnungsempfänger durch ein System dieser Art nicht unterbrochen wird. 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 Rechnungssteller übermitteln die Rechnungsdaten ihrer Kunden an den Betreiber von Com42Bill. Hierfür bietet Com42Bill eine Programmschnittstelle an. Über eine Weboberfläche kann der Rechnungssteller den Status seiner übertragenen Rechnungen einsehen und seine Daten wie z.B. die Rechnungen verwalten.

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

Die Finanzdienstleister führen die Zahlungsaufträge, die von den Rechnungsempfängern erteilt werden, aus. Durch die flexible Architektur von Com42Bill und den konsequenten Einsatz von offenen Standards wie z.B. ebXML (siehe Kapitel 3.6) kann das System ohne großen Aufwand an verschiedene Übertragungsprotokolle und Datenformate der Finanzdienstleister und Rechnungssteller angepasst werden.

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

5 Anforderungen

Die Anforderungsanalyse bildet das Fundament der Software-Entwicklung. In dieser Phase werden die Anforderungen an ein zu entwickelndes Softwaresystem beschrieben und hinsichtlich Korrektheit, Vollständigkeit, Sachgerechtigkeit, Konsistenz und Machbarkeit überprüft. Hierbei wird zwischen funktionalen und nicht-funktionalen Anforderungen unterschieden:

Funktionale Anforderungen beschreiben die Arbeitsweise und grundlegende Eigenschaften der problembezogenen Funktionalität.

Nicht-funktionale Anforderungen sind Anforderungen an die Umstände, unter denen die geforderte Funktionalität zu erbringen ist.

Die Ausgangsbasis der Anforderungsanalyse für das System Com42Bill war eine Einteilung in die fünf Komponenten *GUI*, *Business-Logic*, *Datenkonverter*, *Sicherheit* und *Datenbank*. Zunächst galt es, einen Überblick über die Funktionsweise eines EBPP-System und somit eine solide Basis für die Anforderungsdefinition zu bekommen. Grundlage hierfür war die Seminarausarbeitung über EBPP-Systeme [Has02] sowie weitere themenspezifische Dokumente, aber auch eigene Vorstellungen über die Funktionsweise eines EBPP-Systems. Jedes Komponententeam hat daraufhin ein Anforderungsdokument erstellt, in dem textuell die Funktionalitäten der eigenen Komponente sowie die Anforderungen an die Schnittstellen der anderen Komponenten beschrieben sind. Resultierend hieraus entstanden die ersten funktionalen Anforderungen der einzelnen Komponenten.

Es hat sich hierbei gezeigt, dass in den Anforderungsdokumenten Uneinigkeit über die Bezeichnung der am System beteiligten Akteure herrschte. In Folge dessen wurde ein Rollenmodell entwickelt, in dem die Akteure namentlich benannt und ihre Beziehungen untereinander abgebildet worden sind (siehe Anhang A). Von diesem Zeitpunkt an waren die im Rollenmodell verwendeten Bezeichnungen der Akteure für sämtliche Dokumente verpflichtend.

Nach mehreren Iterationen, in denen die Anforderungsdokumente der Komponenten erweitert und aufeinander abgestimmt wurden, entstand ein erstes einheitliches Gesamtbild eines EBPP-Systems. Schließlich wurden die in den Anforderungsdokumenten beschriebenen Anforderungen zu einer Anforderungsliste in Tabellenform zusammengefügt, um einen übersichtlicheren Gesamtüberblick zu gewinnen. Die Anforderungsliste wurde weiterhin um nicht-funktionale Anforderungen ergänzt.

Um den Prozess der Anforderungsdefinition zu unterstützen, wurden Use Case- und Aktivitätsdiagramme mit Hilfe der Unified Modeling Language (UML) erstellt (siehe Anhang B und Anhang C). Diese Diagramme dienten dazu, die anfallenden Geschäftsvorfälle zu beschreiben. Die Anforderungsliste wurde danach mit den UML-Diagrammen abgeglichen und um fehlende Anforderungen ergänzt. Das Erstellen der Anforderungsliste durchlief ebenfalls mehrere Iterationen, in denen die Anforderungen der Komponenten untereinander abgeglichen und vervollständigt wurden. Die vollständige Anforderungsliste ist in Anhang D abgebildet.

Die Komponente „Sicherheit“ hat Sicherheitsrichtlinien aufgestellt, die von jeder Komponente eingehalten werden müssen (siehe Anhang F). Diese Sicherheitsrichtlinien sind nicht-funktionale Anforderungen, die aber einen speziellen Status haben und daher nicht in die Anforderungsliste aufgenommen wurden.

Die Anforderungsliste ist nach Komponenten sortiert und besteht aus folgenden Spalten:

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.	
Autor	Gibt den Urheber an, um einen Ansprechpartner für Rückfragen zu haben	
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
Keyfeature (KF)	In dieser Spalte werden die Anforderungen mit einem „x“ markiert, bei denen es sich um ein Key-Feature für die Prototyp-Entwicklung handelt (vgl. Kapitel 8).	

Diese Anforderungsliste wird nun umgesetzt in eine verfeinerte Architektur. Sie ist weiterhin Richtlinie für die zu implementierenden Funktionen während der Implementierungsphase.

6 Die Architektur

6.1 Allgemein

Die Systemarchitektur gibt eine Übersicht über die einzelnen Komponenten, welche gemeinsam das E-Bill Presentment- & Paymentsystem 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 zufrieden stellendes 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 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. 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 dass Vorhandensein der benötigten Objekte sichergestellt werden.

Im Folgenden werden die Teilarchitekturen der einzelnen Komponenten vorgestellt.

6.2 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.

Die Idee ist, eine Oberfläche zu schaffen, 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 des Portals 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. Die feste Breite der Seiten unterstützt dabei dieses Ziel, indem dafür gesorgt wird, dass alle Elemente der Portalseiten unabhängig von der Bildschirmauflösung ihre feste Position behalten. Somit wird

einem versehentlichen Verrutschen der Textabsätze oder der Formularelemente vorgebeugt, wodurch eine Verwirrung des Benutzers vermieden wird.

Das Kennenlernen des Com42Bill-Portals 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.

Sollten dennoch Unklarheiten bestehen, können die Benutzer per Mausklick auf eine Hilfe-Datenbank zugreifen und diese nach bestimmten Schlagwörtern oder Phrasen durchsuchen.

Einen kleinen 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- sowie eine WML-Darstellung der Benutzungsoberfläche bereitgestellt. 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), XSLT (XML-Stylesheet-Language-Transformation) und XSL-FO (XML-Stylesheet-Language-Formatting-Objects). Mit diesen Mechanismen lassen sich auf einfachste Weise dynamisch Inhalte für die verschiedensten Ausgabemedien erzeugen. Die Komponente GUI erhält von der Business Logic fachliche Objekte, welche zur Anzeige ermittelt werden. Diese müssen nun zunächst in einen XML-Datenstrom transformiert werden. An dieser Stelle ist der evtl. 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. 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. Nur 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 bisher 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 würde der Einarbeitungsaufwand den gewonnenen Nutzen in diesem Falle übersteigen.

Die Komponente GUI hat folgende Architektur:

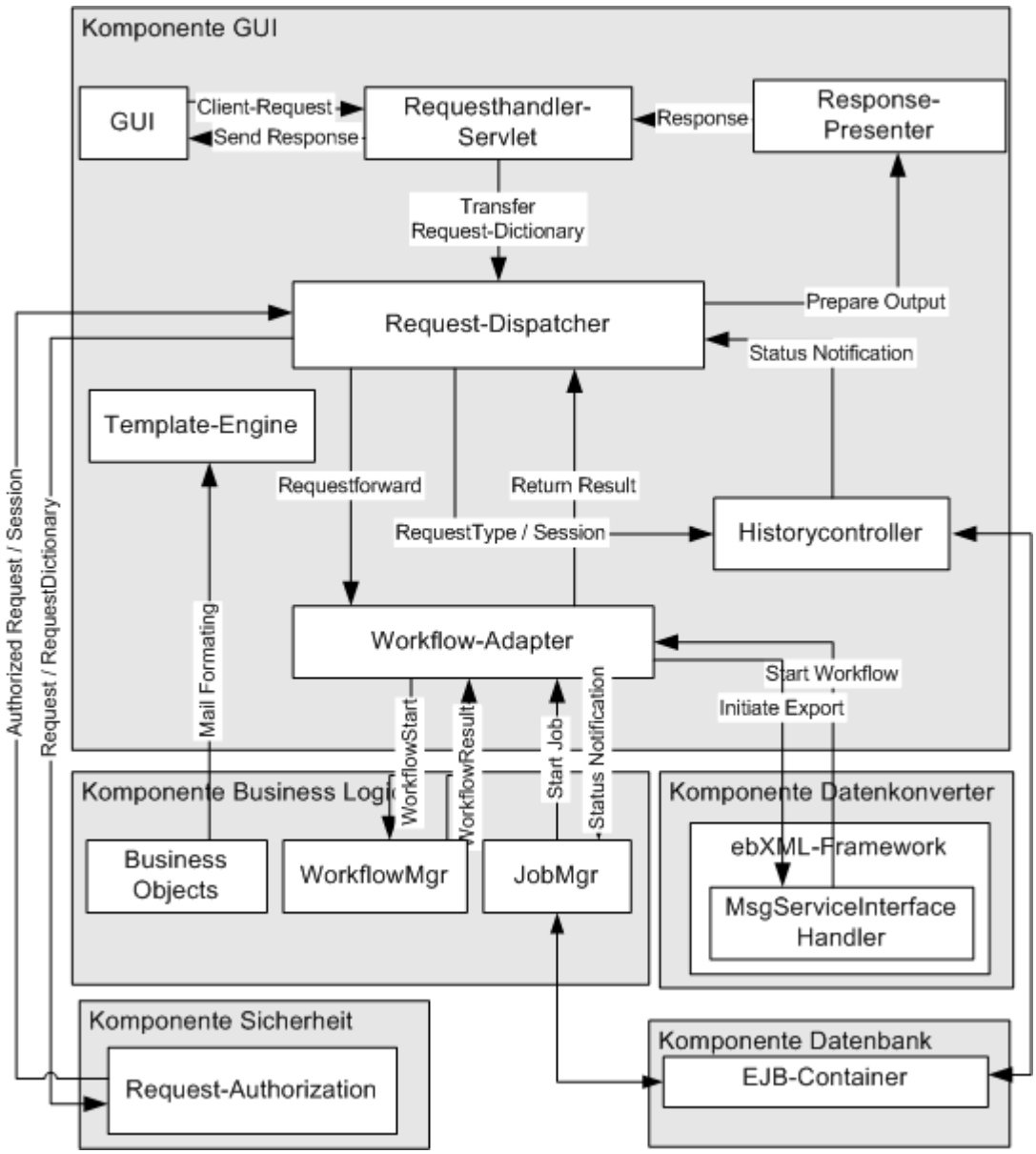

Abbildung 3: Teilarchitektur der Komponente GUI

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

Requesthandler-Servlet: 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 Request befinden, werden in einem RequestDictionary (Container-Objekt) hinterlegt. Je nach Art der Implementierung schickt das Servlet auch den Ausgabestrom mit den anzuzeigenden Daten zurück zum Client.

Request-Dispatcher: Diese Komponente ist die zentrale Verwaltungsstelle der ClientRequests. Der Dispatcher delegiert die Request-Autorisierung, das SessionManagement (und somit die Authentifizierung des Clients), die Gültigkeitsabfrage des Requests anhand der Navigationsstruktur, sowie die Initiierung des Geschäftsprozesses und die Rückgabe des Ergebnisses.

Historycontroller: Beim Historycontroller muss die ganze Navigationsstruktur der GUI hinterlegt werden. Hierdurch wird für jeden Request ersichtlich, ob dieser Request 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 System befindet. Der Controller arbeitet dabei ähnlich einem Zustandsautomaten, wobei durch das Ausführen von Geschäftsprozessen und dem anschließenden Anzeigen der Ergebnisse Zustände definiert werden. Mit Hilfe der Navigationsstruktur werden alle zulässigen Zustandsübergänge festgelegt. Bei jedem Request wird also der aktuelle Zustand der Session festgestellt und mit Hilfe der aktuellen Anfrage entschieden, ob ein entsprechender Zustandsübergang existiert, womit die Anfrage abgearbeitet werden kann. Andernfalls wird der Zustand nicht geändert und der Benutzer erhält eine entsprechende Meldung oder die vorhergehende Seite erneut angezeigt. Durch den Einsatz des Historycontrollers wird also dem mehrfachen Abschicken desselben Requests durch Betätigen des Back- oder Reload-Buttons vorgebeugt. Dies ist insbesondere beim Beauftragen von Transaktionen unerlässlich. Ein weiteres Problem stellt der Abbruchbutton der Browser dar. Entgegen einem weitverbreiteten Irrtum wird beim Betätigen dieses Buttons lediglich die http-Verbindung zum Server, über welche der Response empfangen wird, abgebaut. Der Request wird Serverseitig jedoch bis zum Ende durchgeführt. Somit besteht nach Klicken auf den Abbruchbutton die Möglichkeit, den Response, welchen der Client nun nicht mehr empfangen konnte, bei der nächsten Anfrage an das System anzuzeigen.

Workflow-Adapter: Diese Einheit stellt die einzige Schnittstelle von der Requestverwaltung zur BusinessLogic dar. Der Workflow-Adapter beauftragt den WorkflowMgr (siehe Kap. 6.4), welcher den Einstiegspunkt in die BusinessLogic darstellt, den Geschäftsprozess zu starten. Ebenso nimmt er die Ergebnisse der Geschäftsprozessausführung entgegen. Der Workflow-Adapter wird auch von der Komponente Datenkonverter und vom Job-Manager zum Starten von Geschäftsprozessen benutzt.

Response-Presenter: Der Response-Presenter arbeitet die Ergebnisse des Prozesses, welche sich in dem vom Requesthandler-Servlet erstellten RequestDictionary befinden, grafisch auf für die GUI-Anzeige und sendet einen Datenstrom entweder direkt oder über das Requesthandler-Servlet an diese.

Template-Engine: Die Template-Engine dient der Business Logic zur grafischen Aufbereitung von Inhalten, welche anschließend über einen Kommunikationsdienst verschickt werden. Hier ist evtl. der gleiche Mechanismus einsetzbar, wie er schon zu Aufbereitung von Client-Requests benutzt wird.

6.3 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 im Wesentlichen aus vier Untereinheiten. 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 4 zeigt die Architektur der Komponente Sicherheit:

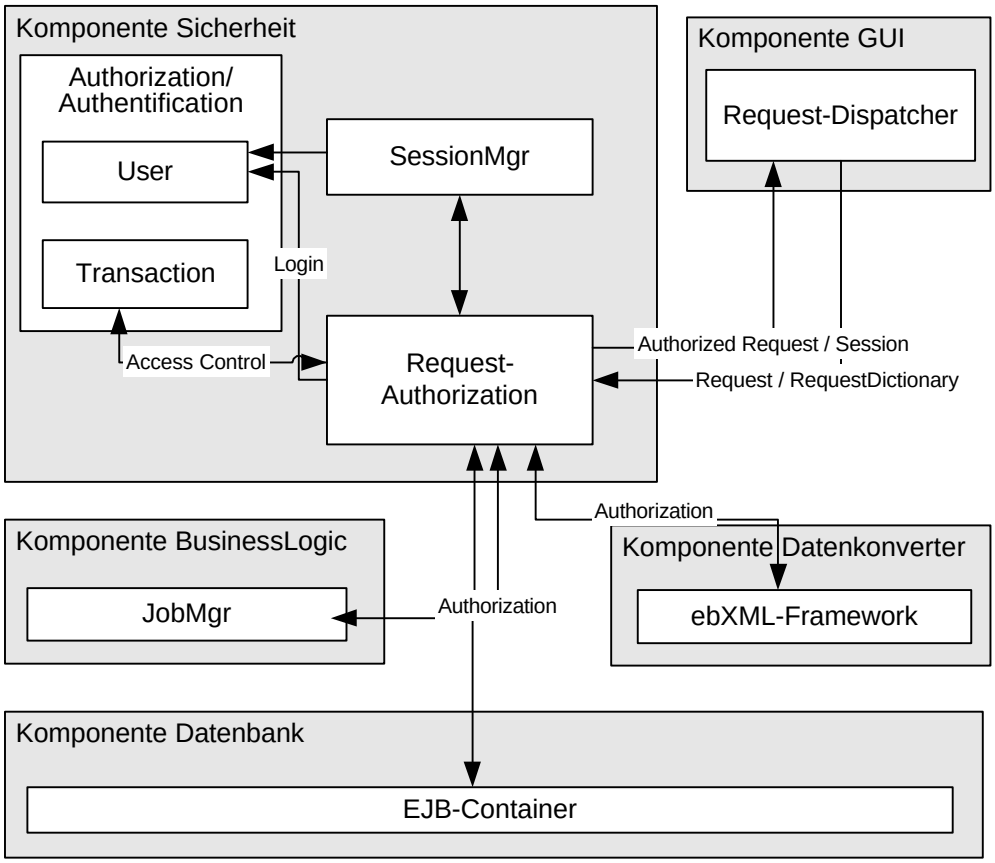

Abbildung 4: Teilarchitektur der Komponente Sicherheit

Request-Authorization: Die RequestAuthorization ist die erste Anlaufstelle bei der Bearbeitung eines jeden Requests. An dieser Stelle wird die Art des Requests identifiziert und im Anschluss die Durchführung des Prozesses autorisiert. Hierzu müssen einige Merkmale des zu initiierenden Geschäftsprozesses bekannt sein. Zunächst wird also die Art des Prozesses ermittelt. Im Anschluss wird evaluiert, ob der Prozess in einer Session ablaufen muss. Ist dies der Fall, wird geprüft, ob ein Benutzer angemeldet sein muss, oder der Request anonym ausgeführt werden darf.

SessionMgr: Der Session-Manager ist für die Verwaltung der Benutzersessions verantwortlich. Da http ein zustandsloses Protokoll ist, jedoch der Benutzer über mehrere Requests hinweg identifiziert werden soll, müssen bei jedem Request Daten mit übertragen werden, um dies sicherzustellen. Mit Hilfe von Sessions wird der Benutzer also auch davor bewahrt, sich bei jedem Request erneut authentifizieren zu müssen. Zur Zeit existieren zwei weitverbreitete

Technologien in diesem Bereich. Die erste Möglichkeit ist die Benutzung von Cookies. Hierbei unterscheidet man zwischen persistenten Cookies, welche für die Benutzung in mehreren Sessions auf der Festplatte abgelegt werden und Session-Cookies, welche beim Schließen des Browsers am Ende einer Session wieder entfernt werden. Da Cookies stets im System des Nutzers abgelegt werden müssen, sind diese jedoch eher zu vermeiden. Die zu bevorzugende Lösung im Bereich der Session-Verwaltung ist das URL-Rewriting, wobei jedem Request eine Session-ID angehängt wird, mit der der User beim System identifiziert werden kann.

User-Authorization/-Authentification: Hier findet die Verwaltung aller Benutzer des Systems statt. Hierzu gehören der Rechnungssteller (incl. dem Zugang über die Programmschnittstelle), der Rechnungsempfänger, sowie die Administratoren des Systems.

Transaction-Authorization/-Authentification: Mit Hilfe dieser Subkomponente 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. Hierbei unterscheidet man die verschiedenen RBAC-Level 0 bis 3. Beim einfachsten Level unterscheidet man zwischen verschiedenen Rollen, welchen Rechte zugeordnet werden. 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 führen komplexere Strukturen wie beispielsweise Hierarchien ein.

6.4 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 Abbildung der anfallenden Geschäftsprozesse erfolgt auf technischer Ebene durch Umsetzung einer Pipelinearchitektur. Mit Hilfe von Pipelines lassen sich Geschäftsprozesse auf eine sehr flexible und – im Falle einer grafischen Umsetzung – anschaulichen Weise darstellen. Pipelines bestehen aus mehreren Teilabschnitten, welche jeweils kleine Schritte von der Geschäftslogik ausführen. Abbildung 5 stellt dies am Beispiel einer Zahlungsanweisung dar.

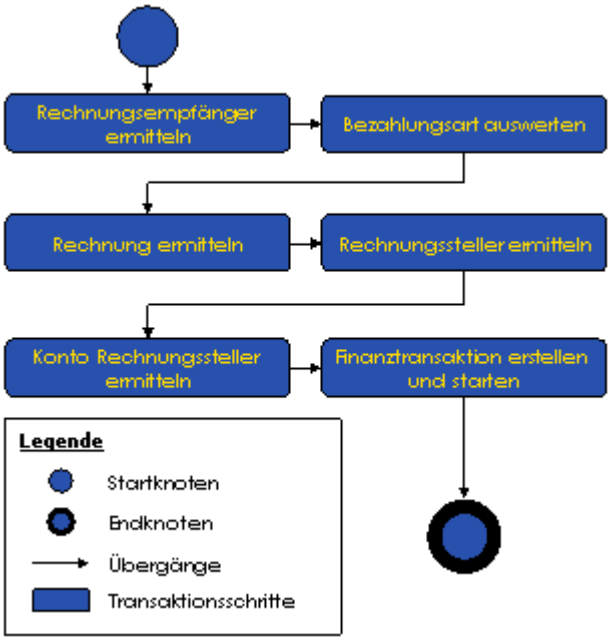

Abbildung 5: Pipeline-Architektur

Um eine größtmögliche Flexibilität zu erreichen, müssen weitergehende Funktionen, wie Verzweigungen zwischen Pipelines, sowie Entscheidungs- und Zusammenführungsknoten bereitgestellt werden. Aufgrund der Modellierung der Pipelines in XML unterliegt man bei der Umsetzung und Entwicklung der Infrastruktur keinen Einschränkungen. Um eine leichte Erweiterbarkeit zu gewährleisten, kann eine grafische Modellierungsoberfläche für die Pipelineerstellung eingesetzt werden.

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 Anweisung 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.

Die Komponente stellt fünf 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, zur Zahlung freigeben oder sperren. 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. Darüber hinaus werden Benachrichtigungs- und Personalisierungsdienste angeboten. Jeder Rechnungsempfänger kann individuell festlegen, ob und wie er bei neuen Rechnungen bzw.

Mahnungen benachrichtigt werden möchte. Weiterhin können Standardwerte für bestimmte Eingabefelder festgelegt werden. Umfangreiche Berichts- und Standardfunktionen runden das Funktionsangebot sowohl auf Betreiber- als auch auf Rechnungssteller- und Rechnungsempfängerseite ab.

Die Business Logic besteht aus zwei Bereichen. Zum einen werden Manager bereitgestellt, welche auf Entitäten operieren. Als zweites wird eine Workflow-Engine entwickelt, mit deren Hilfe sich Geschäftsprozesse modellieren und abarbeiten lassen. Die einzelnen Teile der Geschäftsprozesse laufen je nach Art der Implementierung im EJB-Applikationsserver.

Durch den Einsatz einer Pipelinearchitektur hat die Komponente sehr viel an Flexibilität gewonnen. 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. Aufgrund des Einsatzes der Manager wird der Benutzer der Entitäten in den Business Objects nahezu vom Einsatz von EntityBeans abgeschirmt und kommt in keinem Fall mehr mit SQL in Berührung. Die Unterteilung des Systems in mehrere Komponenten wird hier konsequent fortgeführt. Somit wird es möglich, im Falle einer Entwicklung von Com42Bill bis zur Marktreife, 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 (Abbildung 6) kurz erläutert.

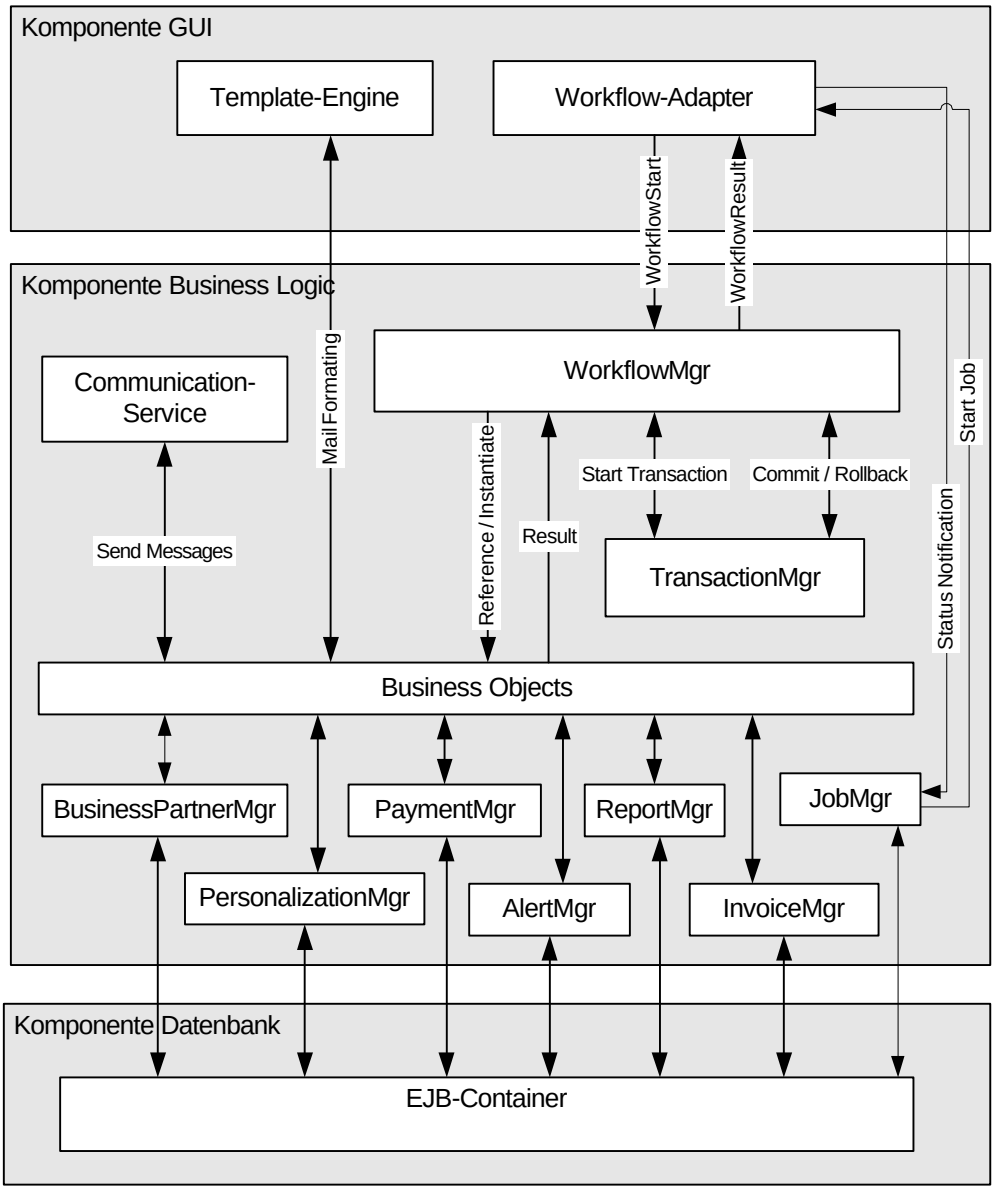

Abbildung 6: Teilarchitektur der Komponente Business Logic

WorkflowMgr: Der WorkflowMgr wird vom Workflow-Adapter aufgerufen. Dieser hat nun die Aufgabe den Geschäftsprozess zu starten und zu überwachen. Zu Beginn des Prozesses wird nach Bedarf eine Transaktion gestartet. Im Anschluss werden ein oder mehrere BusinessObjects erzeugt oder aus einem Pool referenziert.

Business-Objects: Die Business Objects sind die Komponenten der BusinessLogic, welche statusbehaftet sein können. Die Business Objects werden im Rahmen eines Geschäftsprozesses instanziiert und bilden einen Teil des Prozesses. 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 in der Regel durch den Applikationsserver zur Verfügung gestellt.

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

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

PaymentMgr: Der Paymentmanager 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.

PersonalisationMgr: Der Personalisierungsdienst bietet zum einen Funktionen für den Rechnungssteller, um dem Rechnungsempfänger gezielt Informationen zukommen zu lassen; zum anderen muss der Empfänger seine Benutzeroberfläche dauerhaft konfigurieren können.

ReportMgr: Diese Komponente vereint alle Operationen auf Entitäten, die für die Erstellung von Berichten und Statistiken notwendig sind.

JobMgr: Der Jobmanager initiiert alle zeitgesteuerten Geschäftsprozesse. Hierzu greift er auf den Workflow-Adapter zu. Zu Beginn der Ausführung eines Jobs wird ein JobRequest erzeugt, welcher zunächst von der Sicherheit zu autorisieren ist. Der Jobmanager muss auch die Möglichkeit erhalten, im Anschluss an die Durchführung eines Geschäftsprozesses einen Datenexport zu initiieren.

6.5 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.

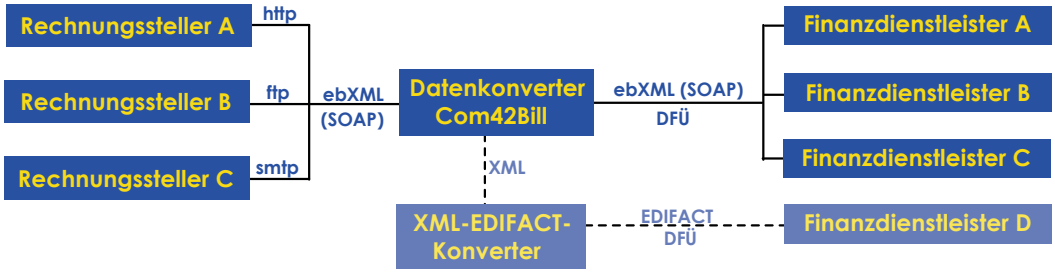

Abbildung 7: Kommunikation von Com42Bill mit anderen Systemen

Der Datenkonverter empfängt Rechnungsdaten von einem Rechnungssteller. Da diese jedoch für gewöhnlich nicht in einem standardisierten Format vorliegen, werden diese 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 von den Vereinten Nationen (UN/CEFACT) und der Organisation für die Förderung strukturierter Informationsstandards (OASIS) verabschiedet wurde [Cr02]. Das Ziel hierbei ist, den Austausch von elektronischen Geschäftsdokumenten zu standardisieren und somit einen einzigen globalen e-Business-Markt zu schaffen. Durch ebXML wird u. a. vorgegeben, wie Geschäftsdaten für den Austausch zwischen zwei Parteien strukturiert werden müssen.

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 7).

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, welches Mechanismen bereitstellt, unabhängig vom verwendeten Datenaustauschformat Dokumente mit Geschäftspartnern auszutauschen.

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 jedoch lediglich Datenträgeraustausch (DTAUS) und EDIFACT eingesetzt wird, woran sich in naher Zukunft auch nichts ändern dürfte. Eine interne Diskussion hat jedoch 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 handzuhaben, 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.

Die im Folgenden erläuterten Teilkomponenten sind in Abbildung 8 ersichtlich oder befinden sich außerhalb des Systems.

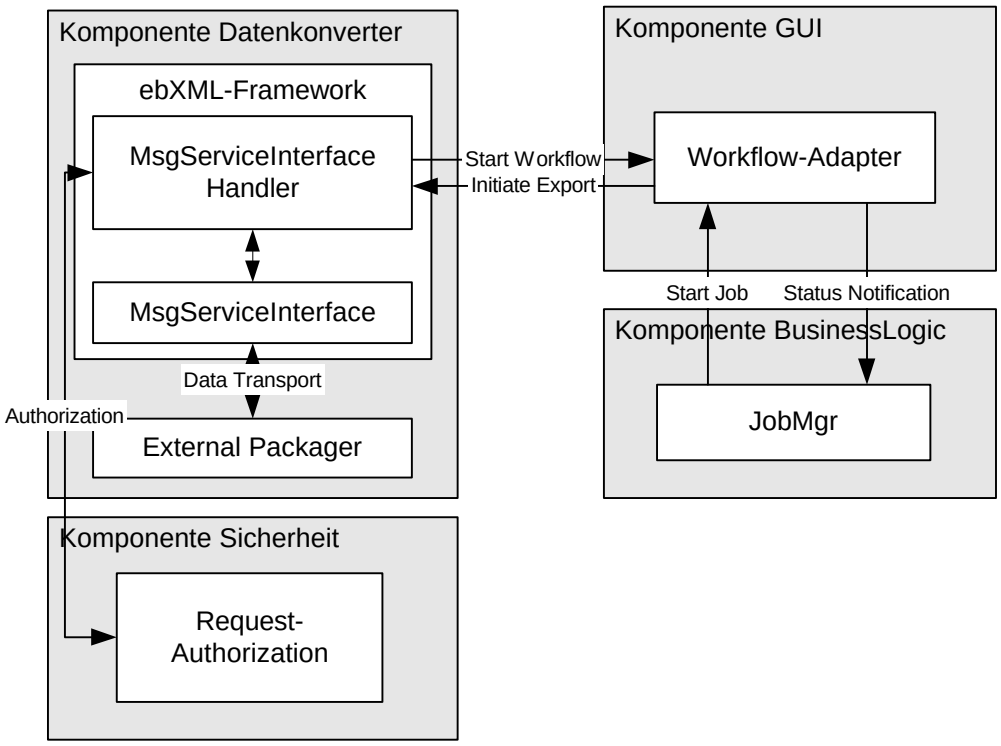

Abbildung 8: Teilarchitektur der Komponente Datenkonverter

ebXML-MsgServiceInterface: Das MsgServiceInterface ist die erste Anlaufstelle für im System ankommende Daten. Es stellt Funktionen für Transport, Routing und Packaging der Daten bereit. Die Nutzdaten werden dem MsgServiceInterfaceHandler zur Weiterverarbeitung übergeben. Ebenso werden Daten von diesem zum Versand übernommen.

ebXML-MsgServiceInterfaceHandler: Diese Teilkomponente des Frameworks stellt die Schnittstelle zum restlichen EBPP-Sytem bereit. Sie ist zunächst für die Authentifizierung/Autorisierung der ankommenden Datenimport-Requests mit Hilfe der Sicherheitskomponente verantwortlich. Bei erfolgreicher Durchführung kann der Geschäftsprozess in der Business Logic gestartet werden. Ebenso nimmt der Handler Datenexportanfragen vom Workflowadapter entgegen und initiiert die Datenübertragung.

External Packager: Der externe Packer wird Vertragspartnern zur Verfügung gestellt, welche keine Möglichkeit besitzen, Daten über das ebXML-Framework auszutauschen. Hierfür werden die eigentlichen Nutzdaten auf Seiten des Partners so verpackt, dass sie ebXML-konform sind und somit über die Systemkomponente Datenkonverter empfangen werden können. Der Packager hat ebenso die Möglichkeit, eine Datenverbindung zum System aufzubauen.

6.6 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. Die globalen Daten und die Daten einzelner Komponenten werden in jeweils voneinander getrennten Bereichen (z.B. Tablespaces) gehalten, um die Verteilbarkeit der Komponente Datenbank zu erleichtern.

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-Applikationsservers. 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 Applikationsservern zu begründen. Beim EJB-Server 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 einzelnen Bestandteile der Komponente, welche in Abbildung 9 schematisch dargestellt sind, werden abschließend erläutert.

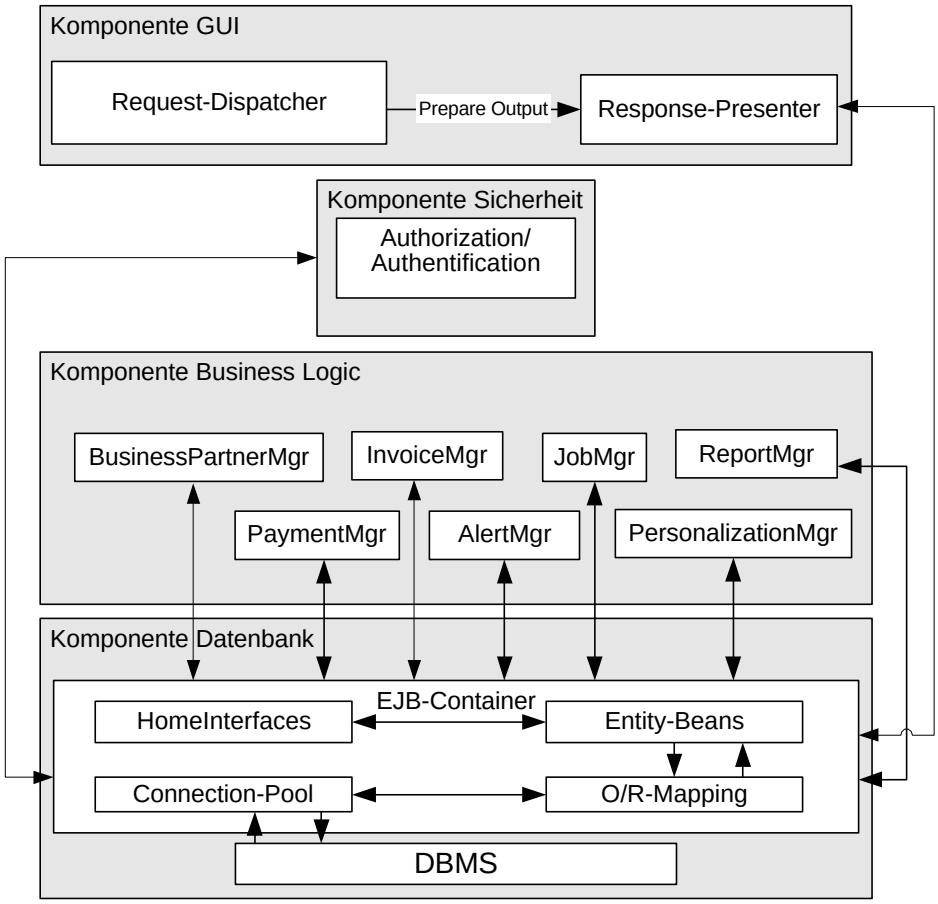

Abbildung 9: Teilarchitektur der Komponente Datenbank

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.

7 Funktionalität

Die Funktionalitäten des EBPP-Systems lassen sich insgesamt in Kernfunktionalitäten, Standardfunktionalitäten und Mehrwertdienste unterteilen.

Eine Kernfunktionalität des Systems ist die Verwaltung von Rechnungen. Hierbei besteht die Möglichkeit, nach bestimmten Rechnungen zu suchen und die Anzeige nach Kriterien, wie zum Beispiel dem Fälligkeitsdatum, zu filtern. Zusammen mit den Rechnungen wird stets der aktuelle Status, wie zum Beispiel offen, bezahlt oder gebucht, angezeigt. Da der Rechnungsempfänger immer über Hypertextdokumente auf das System zugreift, kann er über einen Hyperlink jederzeit die online verfügbare detaillierte Rechnung innerhalb des Internetauftritts des Rechnungsstellers einsehen. Ist der Rechnungsempfänger mit der Gesamtrechnung oder einzelnen Posten nicht einverstanden, besteht die Möglichkeit, sie beim Rechnungssteller zu reklamieren. Akzeptiert der Rechnungsempfänger die Rechnung, kann er eine Zahlungsanweisung entweder sofort initiieren oder für einen späteren Zeitpunkt planen. Hierzu kann jeder Rechnungsempfänger mehrere Finanzkonten im System pflegen und bei der Zahlungsbeauftragung das gewünschte Zahlungsmittel auswählen. Die Zahlungsanweisungen werden bei Fälligkeit an die entsprechenden Finanzinstitute weitergeleitet. Nach Zahlungseingang beim Rechnungssteller wird der Status der Rechnung entsprechend aktualisiert.

Der Rechnungssteller kann mit Hilfe von Datenaustauschformaten über eine Onlineverbindung die Rechnungsdaten zum System zu übertragen, welche im Anschluss in den Rechnungsbestand eingepflegt werden und daraufhin den jeweiligen Rechnungsempfängern präsentiert werden können.

Zusätzlich werden weitere für E-Commerce-Systeme typische Funktionen bereitgestellt. Zu diesen Standardfunktionen zählt vor allem die Verwaltung von Kundenkonten und den dabei anfallenden Daten. Bei Com42Bill existieren zwei Arten von Kunden: die Rechnungssteller und Rechnungsempfänger, welche gesondert behandelt werden, da sie einerseits über andere Schnittstellen mit dem System kommunizieren, andererseits auch ihre Daten auf unterschiedliche Weise präsentiert bekommen.

Unerlässlich für Com42Bill ist die Bereitstellung von Sicherheitsmechanismen, um die sensiblen Daten angemessen zu schützen und deren Integrität sicherzustellen. Hierzu werden Funktionen wie Authentifizierung, Autorisierung und sichere Verschlüsselung aller übertragenen Daten angeboten.

Das System bietet den Anwendern mehr als eine reine Rechnungsverwaltung. Der Rechnungssteller kann sich über verschiedene Kommunikationsmedien, wie zum Beispiel E-Mail, über neue Rechnungen und Fälligkeiten informieren lassen. Der Rechnungssteller hat die Möglichkeit, sein gesamtes Mahnwesen in Com42Bill auszulagern. Das Anwendungsspektrum wird ergänzt durch umfangreiche Mehrwertdienste wie Berichts- und Statistikfunktionen. Ein Personalisierungsdienst erlaubt es den Anwendern, ihren Systemzugang individuell zu konfigurieren, beispielsweise ist es möglich, Standardwerte für bestimmte Eingaben festzulegen.

8 Der Prototyp

8.1 Allgemein

Während des Prototypings werden die grundsätzlichen Eigenschaften des bisher entworfenen Softwaresystems auf ihre technische Machbarkeit überprüft. Die Klassen und Methoden werden nur mit reduziertem Funktionsumfang implementiert, um spezielle Features zu testen. Dies sind die für das System essentiellen Funktionen, auch Key-Features genannt. Das Prototyping wird in der Zeit vom 15.07.2002 bis zum 23.09.2002 durchgeführt.

Für das Com42Bill System wurde das Prototyping in zwei Phasen unterteilt. In der ersten Phase werden die technischen Rahmenbedingungen betrachtet. Diese Phase nimmt die ersten vier Wochen des Prototypings in Anspruch. In dieser Zeit beschäftigt sich jedes Komponententeam mit der eingesetzten Software. Dieses gilt insbesondere für den Application Server, da dieser die Grundlage für die Umsetzung des Systems darstellt. Weiterhin informiert sich jedes Komponententeam über Techniken, die zur Realisierung des Systems nötig sind.

Die zweite Phase beschäftigt sich mit fachlichen Problemen. Hierfür wurden drei Szenarien, welche typische Abläufe innerhalb des Systems darstellen, definiert. Diese wurden so gewählt, dass ein vollständiger Durchlauf durch alle Komponenten gewährleistet ist. Dadurch soll eine Interaktion aller Komponenten bereits im Prototyp realisiert werden. Bei der folgenden Beschreibung der ausgewählten Szenarien werden die im aktuellen Schritt beteiligten Komponenten zur besseren Lesbarkeit nur durch ihren Namen repräsentiert.

8.2 Szenario 1: Rechnungsdaten vom Rechnungssteller empfangen

- Der Datenkonverter empfängt Rechnungsdaten vom Rechnungssteller.
- Der Datenkonverter lässt den Rechnungssteller von der Komponente Sicherheit authentifizieren.
- Der Datenkonverter teilt der Sicherheit mit, welche Aktion (Rechnungsdaten speichern) ausgeführt werden soll.
- Die Sicherheit erteilt dem Datenkonverter die Erlaubnis, diese Aktion auszuführen.
- Der Datenkonverter übergibt der GUI (Workflow-Adapter) ein Container-Objekt, welches die Rechnungsdaten und die auszuführende Aktion enthält.
- Die GUI leitet das Container-Objekt an die Business-Logic weiter (Workflow-Manager).
- Die Business Logic startet einen Transaktionsablauf (Workflow) (Speichern der Rechnungsdaten in der Datenbank).
- Die Datenbank speichert die Daten.

8.3 Szenario 2: Rechnungsdaten eines Rechnungsempfängers darstellen

- Ein Rechnungsempfänger meldet sich über die GUI an.
- Die GUI lässt den Rechnungsempfänger von der Sicherheit authentifizieren.
- Die GUI teilt der Sicherheit mit, welche Aktion (Rechnungsdaten laden) ausgeführt werden soll.

- Die Sicherheit erteilt der GUI die Erlaubnis, diese Aktion auszuführen.
- Die GUI übergibt der Business-Logic ein Container-Object, welches die auszuführende Aktion enthält.
- Die Business-Logic startet einen Transaktionsablauf (Workflow) (Anfrage an Datenbank).
- Die Datenbank übergibt der Business-Logic die Daten.
- Die Business-Logic übergibt der GUI ein Container-Objekt, welches die Daten des Rechnungsempfängers enthält.
- Die GUI präsentiert dem Rechnungsempfänger die Rechnungsdaten.

8.4 Szenario 3: Bezahlvorgang ausführen

- Der Rechnungsempfänger wählt eine Rechung aus.
- Der Rechungsempfänger startet den Bezahlvorgang.
- Der Rechnungsempfänger wählt eine Zahlungsart aus.
- Der Rechnungsempfänger bestätigt die Zahlung.
- Die GUI teilt der Sicherheit mit, welche Aktion (Rechnung bezahlen) ausgeführt werden soll. Die Sicherheit erteilt der GUI die Erlaubnis, diese Aktion auszuführen.
- Die GUI übergibt der Business-Logic ein Container-Object, welches die auszuführende Aktion enthält.
- Die Business-Logic startet einen Transaktionsablauf (Workflow)(Überweisungsdaten aus der Datenbank laden, Zahlungsvorgang beim Datenkonverter initiieren).
- Die Busines-Logic übergibt dem Datenkonverter über den Workflow-Adapter ein Container-Objekt, welches die Überweisungsdaten und die auszuführende Aktion (Überweisung übermitteln) enthält.
- Der Datenkonverter führt die Überweisung aus und gibt der Business-Logic eine Status-Rückmeldung (Workflow beendet).
- Der Rechungsempfänger erhält anschließend eine Bestätigung über den Auftrag.

8.5 Zielsetzung

Am Ende des Prototypings existiert ein erster Teil des Systems, der die wichtigsten Funktionen und Eigenschaften des Softwaresystems enthält. Außerdem wird feststehen, welche Techniken die jeweiligen Komponenten verwenden werden und wie diese eingesetzt werden. Die gewonnenen Erkenntnisse bilden die Grundlage für die weitere Feinplanung und die Implementierung des Gesamtsystems.

9 Das Datenmodell

Um einen strukturierten und umfassenden Überblick über das Objektmodell von Com42Bill zu geben, wird zunächst die Package-Struktur des Systems dargestellt. Die Summe der Klassendiagramme aller Packages bildet das Objektmodell des Systems. Im folgenden werden die Packages des Softwaresystems Com42Bill beschrieben. Dieses Modell liegt jedoch noch nicht in seiner endgültigen Fassung vor, bis zum Endbericht werden sich noch einige Änderungen ergeben.

Ein Package kann weitere Packages oder Klassen enthalten. Packages bilden also Gruppierungen für funktional zusammengehörige Klassen. Die Packagestrukrur kann mit der 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.

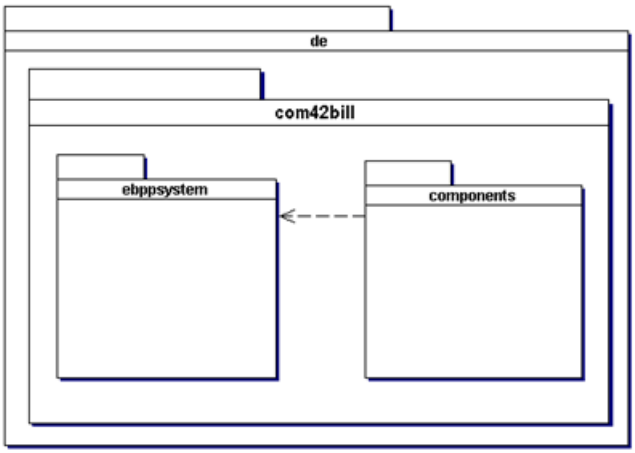

Abbildung 10: Grobe Package-Struktur des Systems

Abbildung 10 ist die grobe Darstellung der Package – 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 detailliert beschrieben. Um ein verteiltes System realisieren zu können, werden sämtliche Klassen des Systems, die als eine persistente Entität modelliert sind, einer speziellen Klasse *domain* zugeordnet. Diese Klasse ist im Package *de.com42bill.ebppsystem* enthalten. Die Relation zwischen den beiden Packages *ebppsystem* und *components* in Abbildung 10 repräsentiert u. a. diese Zuordnungen zwischen der Klasse *domain* und den persistenten Entitäten, die im Package *components* enthalten sind.

Das Package *ebppsystem* umfasst das zentrale Objektmodell des Systems. Dieses umfasst einen Teil der objektorientierten Modellierung der Komponente Business Logic sowie allgemeine Klassen, die allen Komponenten zur Verfügung stehen. Die Objektmodelle der übrigen Komponenten werden im Package *components* zusammengefasst.

Auf Vererbung muss verzichtet werden, da die Vererbung von Entity Beans von EJB 2.0-Spezifikation nicht unterstützt wird.

Klassen, die mit Klassen eines beschriebenen Packages verbunden und in anderen Packages enthalten sind, werden als Link modelliert. Ein Link wird durch ein Pfeil-Symbol an der unteren Kante des entsprechenden Klassen-Symbols gekennzeichnet.

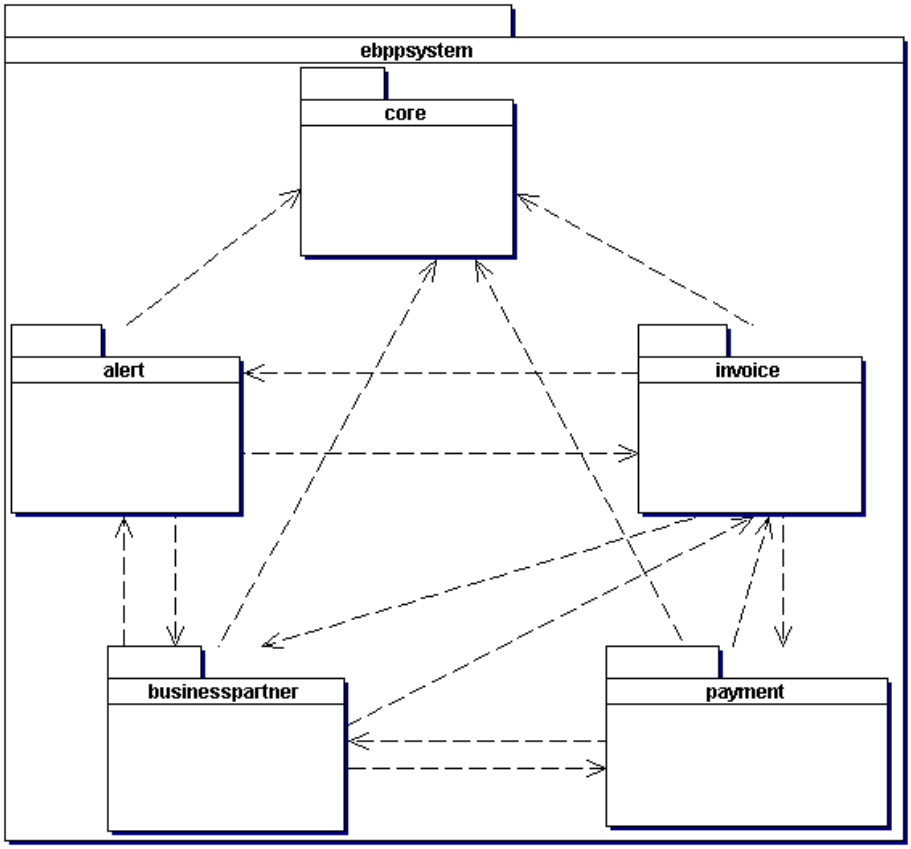

Abbildung 11: Struktur des Package ebppsystem

Wie Abbildung 11 zeigt, enthält das Package *ebppsystem* die fünf Sub Packages *businesspartner*, *invoice*, *payment*, alert und *core*. Diese sind in unterschiedlicher Weise voneinander abhängig. Die durch Pfeile dargestellten Abhängigkeiten korrespondieren mit den Assoziationen der Klassen, die in den unterschiedlichen Sub Packages von *ebppsystem* enthalten sind. Ungerichtete Assoziationen zwischen Klassen der Packages werden durch Doppelpfeile dargestellt. Die Doppelpfeile beschreiben die wechselseitige Beziehung zwischen den Klassen, die in den Packages enthalten sind.

Das Package *invoice* enthält alle Klassen, die zur Abbildung der Rechnung benötigt werden. Eine Rechnung und zugehörige Mahnungen, welche im Package alert modelliert sind, werden von einem Rechnungssteller an einen Rechnungsempfänger übermittelt. 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.

Eine zentrale Rolle spielt das Package *core*: Dieses Sub Package von *ebppsystem* enthält Klassen, die allen Komponenten bzw. Packages des Systems Informationen bereitstellen. Mit Hilfe von core wird auch die Konfiguration des gesamten Systems vorgenommen und Kernfunktionalitäten implementiert. Eine Übersicht zu diesem Package gibt Abbildung 12.

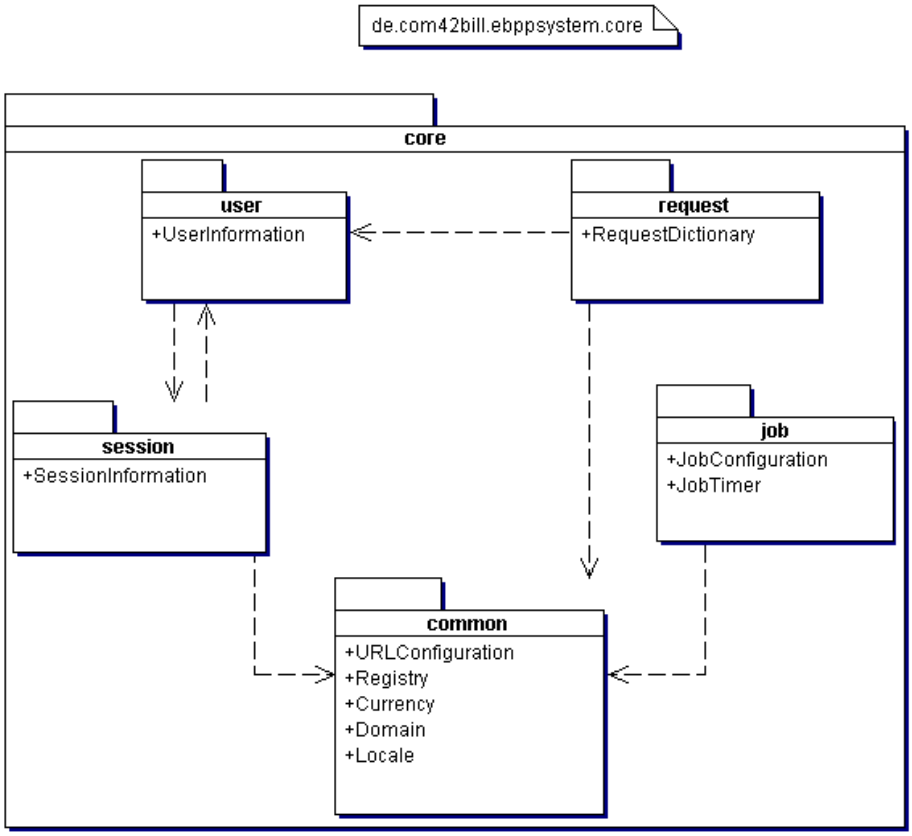

Abbildung 12: Package-Struktur des Sub Package core

Die Klassen des Sub Packages common stehen allen Komponenten des Systems zur Verfügung und stellen die systemweiten Informationen bereit. Die Klassenstruktur von *common* wird in Abbildung 13 dargestellt.

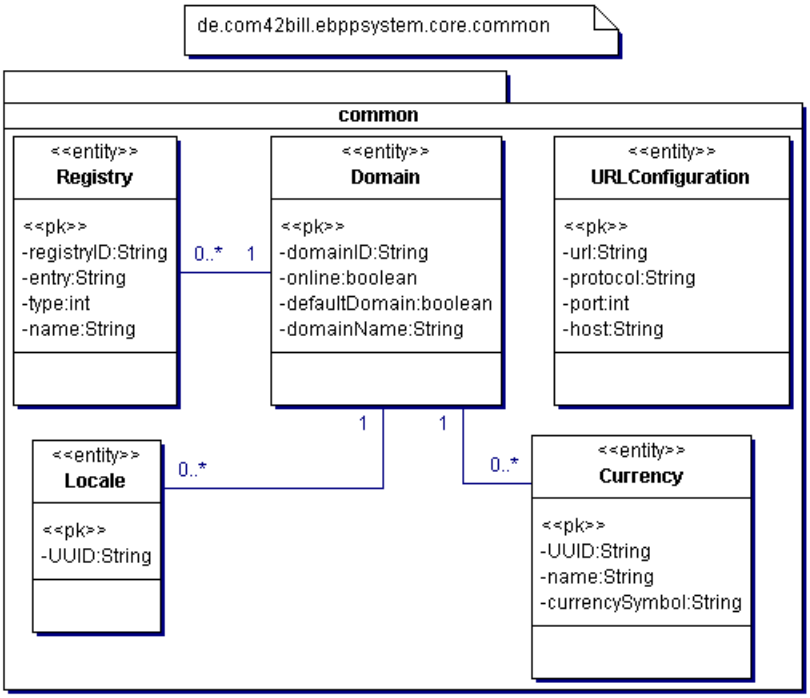

Abbildung 13: Klassen des Package common

Die Bedeutung der einzelnen Klassen lässt sich wie folgt beschreiben: Die Klasse *URLConfiguration* dient der Bereitstellung von Einstiegsadressen für Benutzergruppen des Systems. Diese Klasse dient der Angabe zusätzlicher URLs zur SSL-Verschlüsselung, welche durch das System selbst generiert werden können. Die Klasse *Locales* ist optional vorgesehen und unterstützt die Umsetzung der Länderanpassung des Systems. In der Klasse *Registry* werden Dienste und Funktionseinheiten des Systems zentral registriert. In der Klasse *Currency* werden Währungseinstellungen hinterlegt. Die Klasse Domain stellt die „Dachorganisation" des Systems dar und dient der Realisierung eines verteilten Systems. Ihr werden alle Entity-Klassen des Systems zugeordnet, die für den Aufbau einer Domäne eines verteilten Systems notwendig sind.

Das Sub Package *session* enthält die Klasse SessionInformation, die der Verwaltung von User-Verbindungen zum System dient. Hierfür werden innerhalb der Klasse *SessionInformation* die Daten aus der Klasse *UserInformation* des Package *user* benötigt (siehe Abbildung 14).

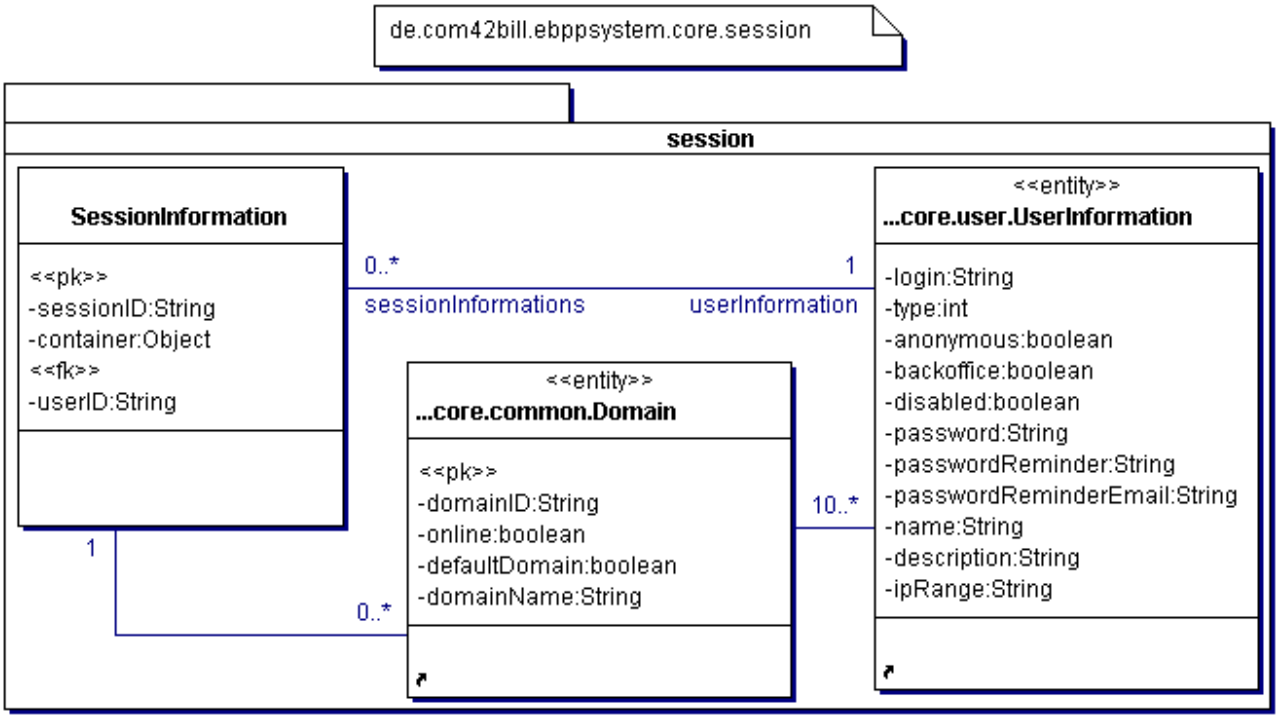

Abbildung 14: Klassen des Package session

Durch das Sub Package *user* bzw. durch die darin enthaltene Klasse *UserInformation* werden die Benutzer (User) abgebildet, die mit dem System in Online-Verbindung stehen. Diese Klasse wird von der Komponente *security* zur Unterstützung der Zugriffssteuerung benötigt. Die Klassenstruktur dieses Package wird in Abbildung 15 dargestellt.

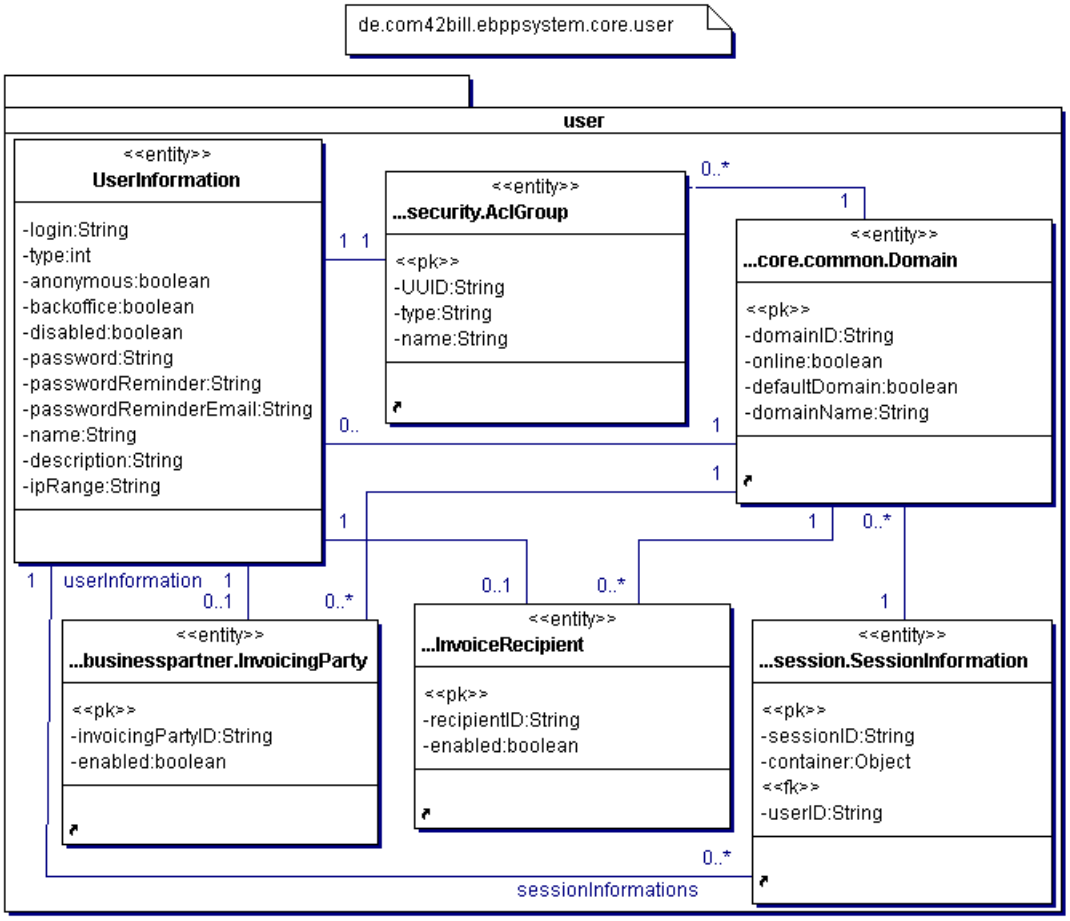

Abbildung 15: Klassen des Package user

Die Klassen, die zur Realisierung des Jobmanagers benötigt werden, sind im package *job* enthalten. Dieses wird in Abbildung 16 dargestellt.

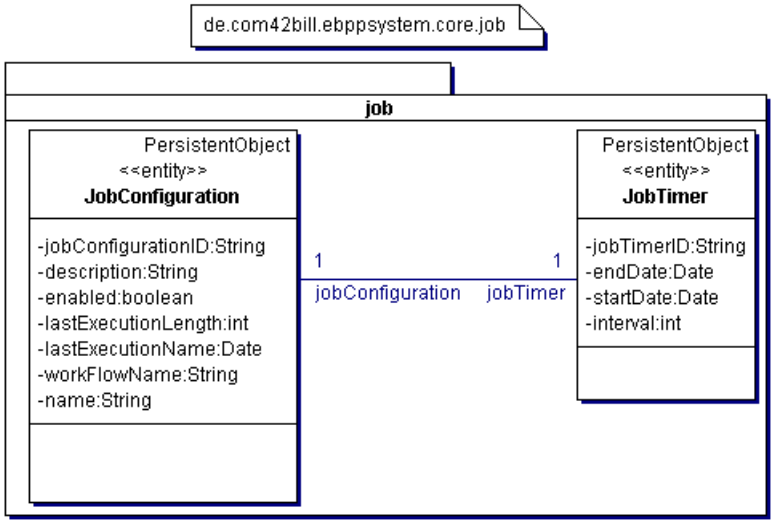

Abbildung 16: Klassen des Package job

In der Klasse JobConfiguration werden Einstellungen zur Ausführung eines Workflows hinterlegt. Die Klasse *JobTimer* dient der Zeitsteuerung der Workflow-Ausführung.

Die von den Komponenten benötigten Daten über die aktuell zu bearbeitende Sitzung werden mit Hilfe der Klasse *RequestDictionary* des Packages *request* zwischen den Komponenten übertragen. Die Klassenstruktur von *request* wird inAbbildung 17 dargestellt.

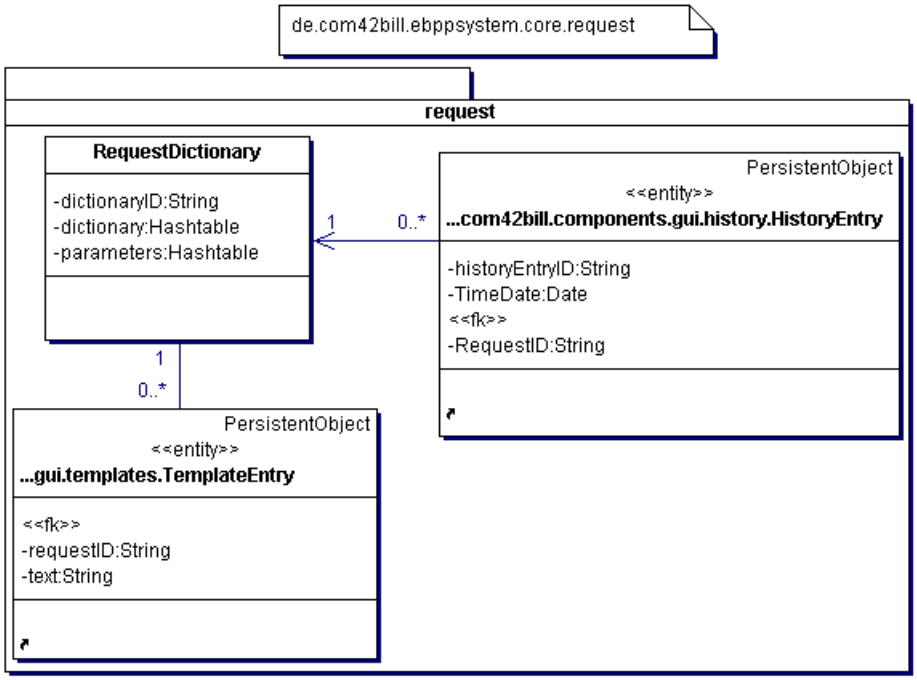

Abbildung 17: Klassen des Package request

Die Klasse *RequestDictionary* übernimmt die Bearbeitung von Online-Anfragen der User an das System. Die Bedeutung der übrigen Klassen, die in Relation zu *RequestDictionary* stehen, wird bei der Beschreibung der entsprechenden Packages erläutert.

Wie Abbildung 11 zeigt, ist das Package *alert* ein weiteres Sub Package von *ebppsystem*. Die Klassenstruktur von alert wird in Abbildung 18 dargestellt.

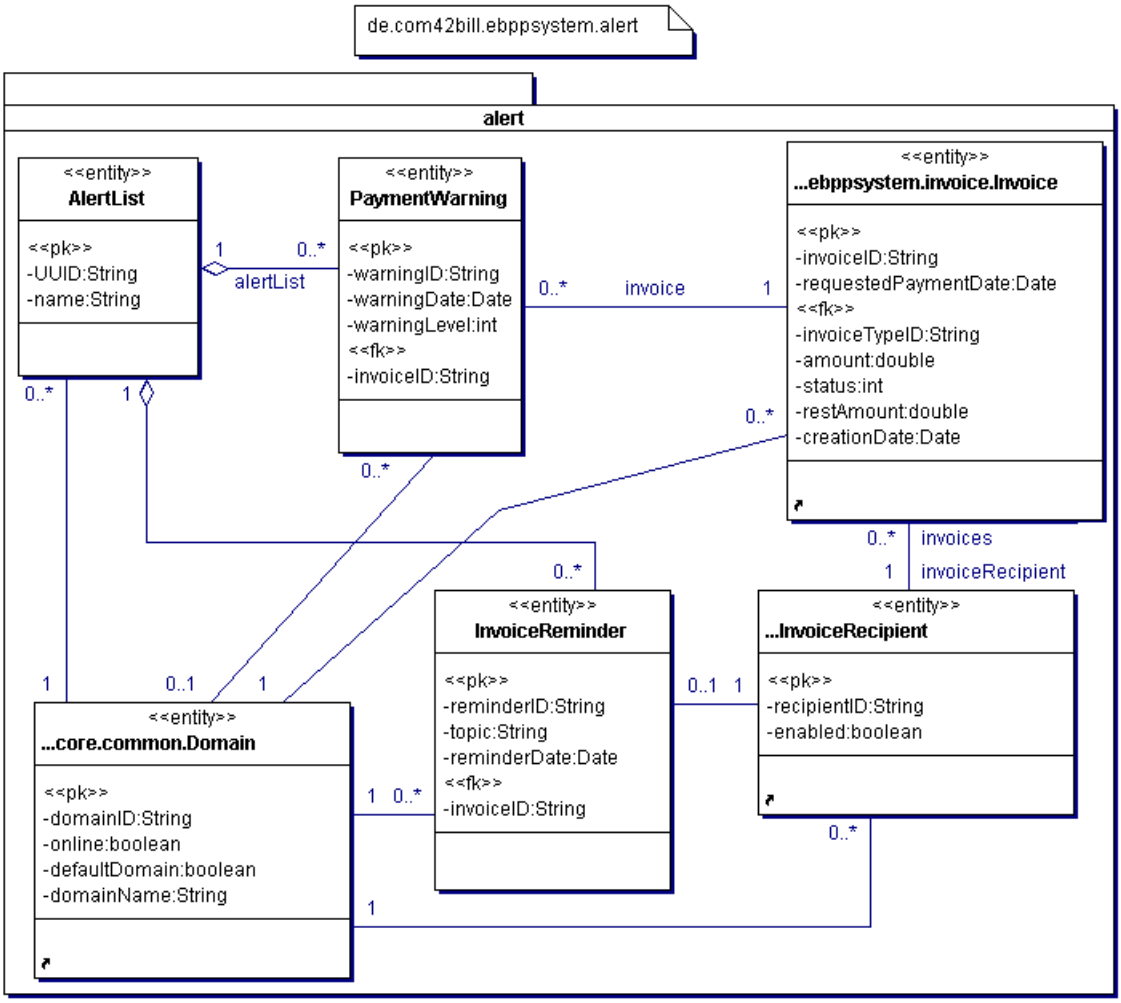

Abbildung 18: Klassen des Sub Package alert

Mehrere Zahlungshinweise können zu einer Liste zusammengefasst werden. Diese Liste wird durch *AlertList* modelliert. Es gibt zwei Ausprägungen des Zahlungshinweises: Erinnerung (*InvoiceReminder*) und Warnung (*PaymentWarning*), die je nach Dauer des Zahlungsverzugs eingesetzt werden können. Mehrere Zahlungshinweise sind einem Rechnungsempfänger und einer Rechnung zugeordnet.

Ein weiteres Sub Package von *ebppsystem* ist das Package *businesspartner*. Dessen Klassenstruktur wird in Abbildung 19 gezeigt.

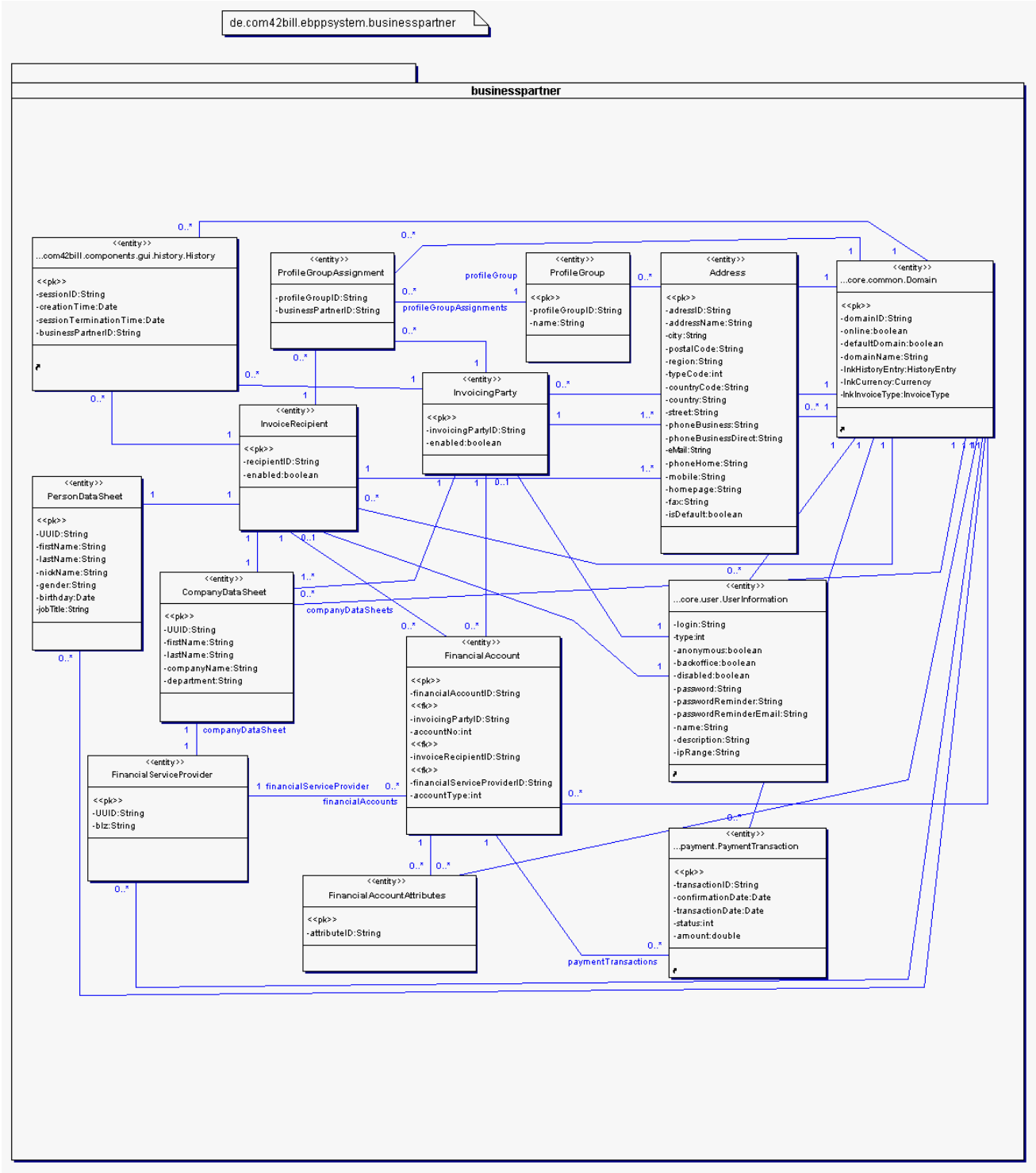

Abbildung 19: Klassen des Sub Package businesspartner

Das Package *businesspartner* enthält die Klasse, die die Teilnehmer an dem Softwaresystem repräsentieren (siehe Anhang A). Die Kontaktdaten werden durch die zwei Klassen *PersonDataSheet* und *CompanyDataSheet* dargestellt. Diese Klassen beinhalten Informationen zu Firmen und privaten Einzelpersonen. Die benötigten Adressinformationen

werden durch die Klasse *Address* repräsentiert. Die Klassen der Rechnungssteller und Rechnungsempfänger können über die Klasse *ProfileGroupAssignment* zu Profilgruppen (*ProfileGroup*) zusammengefasst werden, um z.B. die Rechtevergabe zu realisieren. Das Package *businesspartner* enthält ausserdem das Sub Package *financalaccount*, das in seiner Klassenstruktur ebenfalls in Abbildung 19 enthalten ist. Es repräsentiert die Finanzkonten der beteiligten Rollen, über die die Zahlungsvorgänge abgewickelt werden. Die Klasse *FinancialAccountAttributes* dient der weiteren Spezifizierung eines Objekts der Klasse *FinancialAccount*, z.B. Angabe von Kreditkarteninformation.

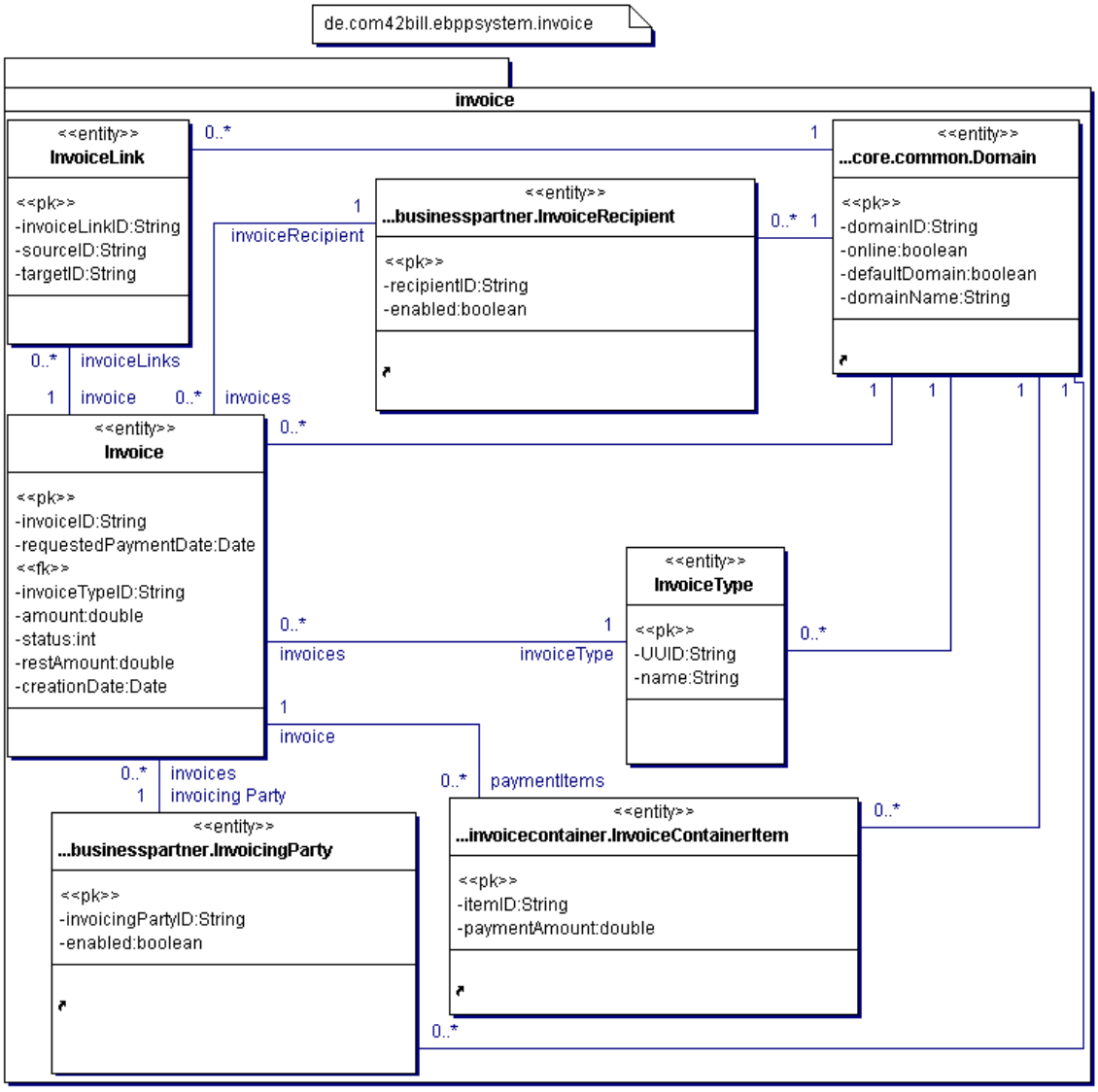

Abbildung 20: Klassen des Sub Package invoice

Mit den Klassen des Package *invoice* werden die Rechnungen modelliert. Seine Klassenstruktur ist in Abbildung 20 dargestellt. Rechnungen werden mit Hilfe der Klassen *InvoiceContainer* und *InvoiceContainerItem* bezüglich der Benutzer gruppiert. Eine thematische Gruppierung kann (optional) über die Klasse *InvoiceType* erfolgen. Mit *InvoiceLink* können Beziehungen zwischen mehreren Rechnungen hergestellt werden.

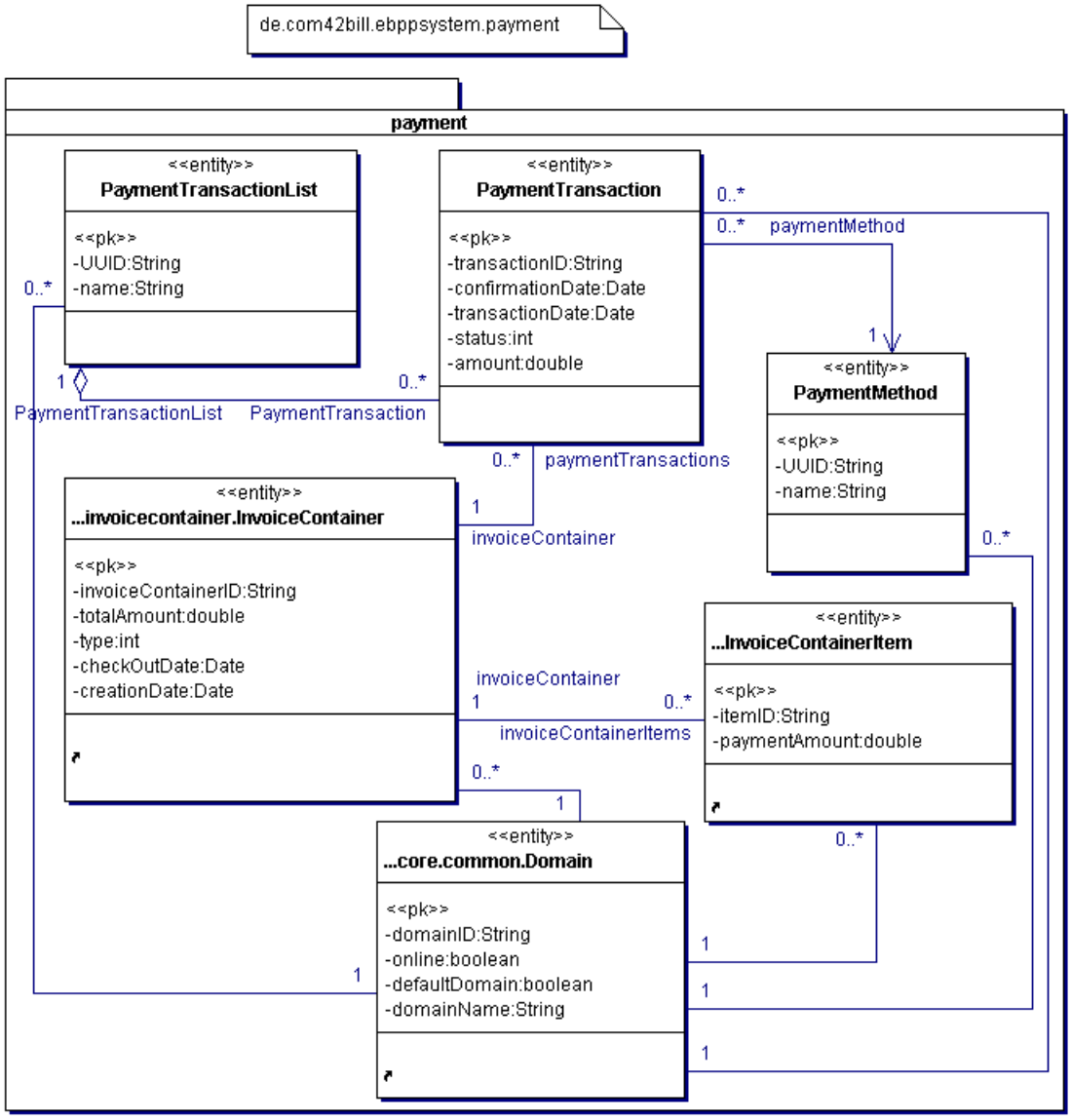

Abbildung 21: Klassen des Package payment

Der Zahlungsvorgang wird im System durch die Klassen des Package *payment* repräsentiert, hier dargestellt in Abbildung 21. Die Transaktionsdaten werden in der Klasse *PaymentTransaction* abgebildet, die Zahlungsweise in *PaymentMethod*. Zahlungen werden durch die Klasse *PaymentTransactionList* zusammengefasst.

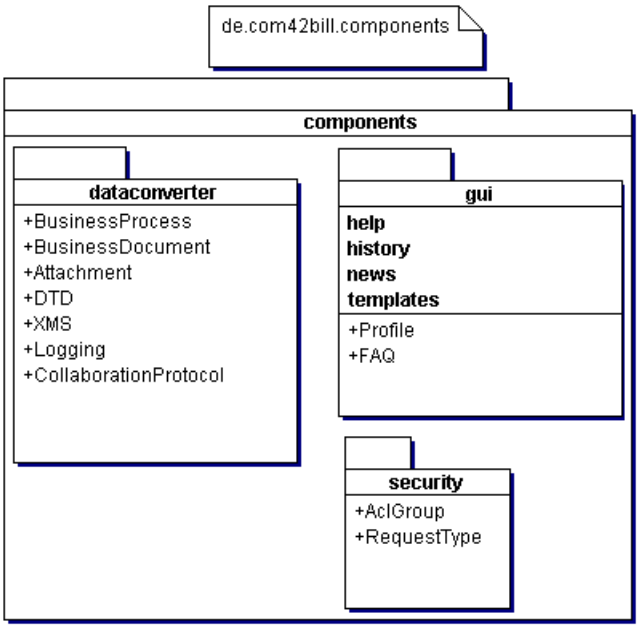

Abbildung 22: Package-Struktur des Package components

Wie Abbildung 10 zeigt, enthält das Objektmodell des Systems neben dem Package *ebppsystem* ein weiteres Package *components* mit zentraler Bedeutung. In dem Package *ebppsystem* wird die Komponente Business Logik modelliert, in dem Package *components* sind die Objektmodelle der übrigen Komponenten enthalten. Deren Klassenstrukturen werden in den jeweiligen Sub Packages *security*, *dataconverter* und *gui* beschrieben (siehe Abbildung 22). Die Klassen dieser Sub Packages hängen mit denen der Komponente *BusinessLogic* zusammen, jedoch existieren keine Relationen zwischen den Sub Packages von components.

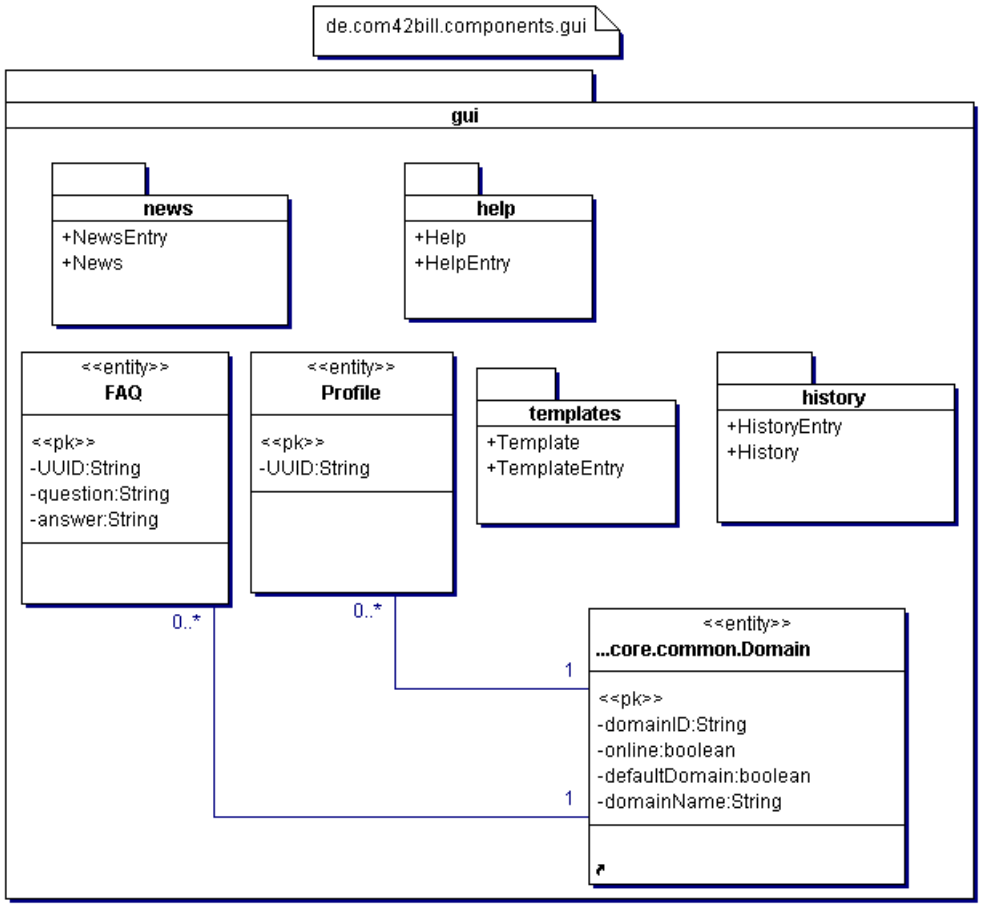

Abbildung 23: Klassen des Package gui

Die Klassen der Komponente GUI werden entsprechend Abbildung 23 im Package *gui* zusammengefasst. Innerhalb von *gui* werden die Klassen in weiteren Sub Packages organisiert.

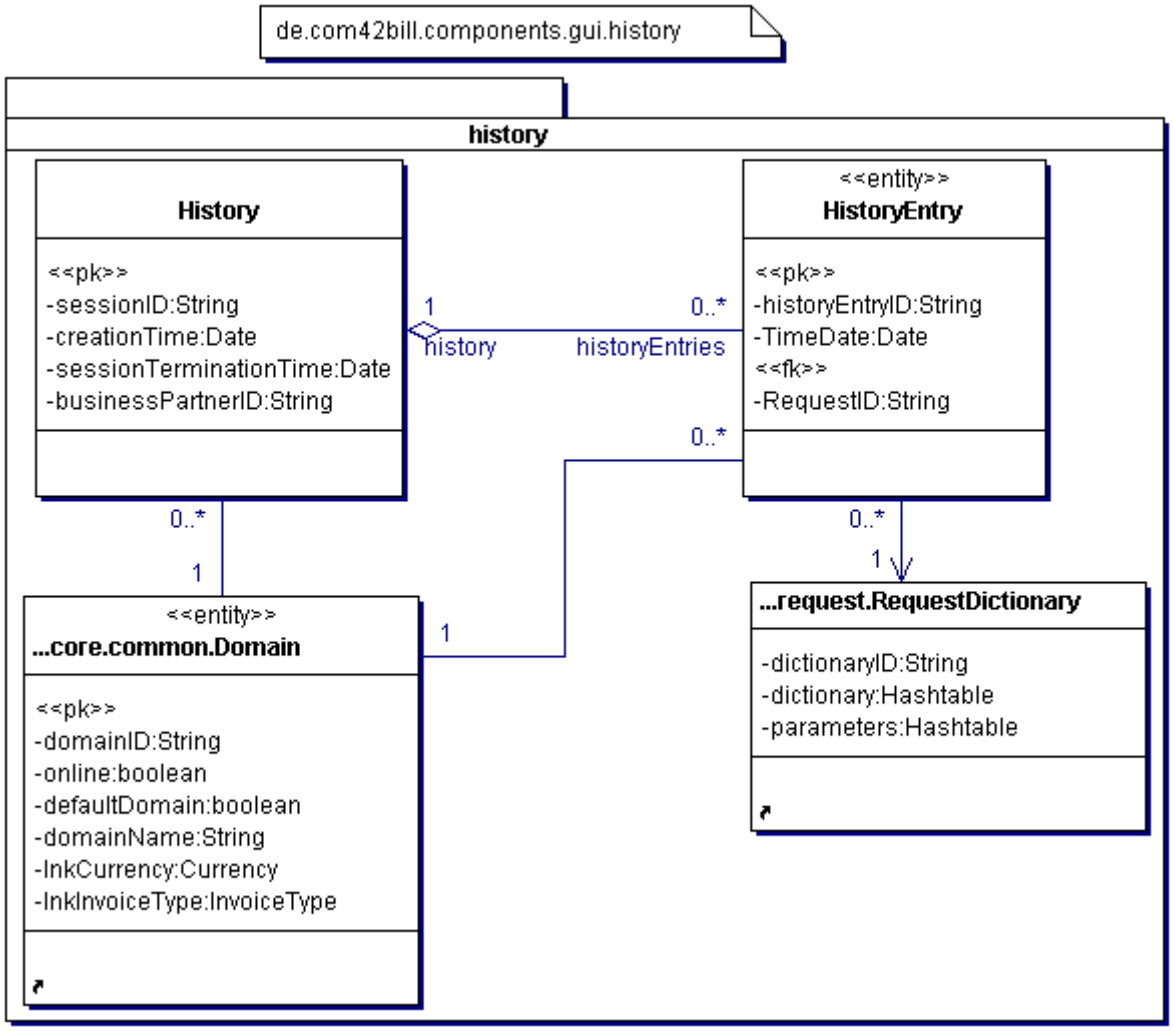

Abbildung 24: Klassen des Package history

Mit Hilfe der Klassen *History* und *HistoryEntry* des Sub Packages *history* wird die Navigation eines Benutzers im System protokolliert. Dieses ist sinnvoll, um bei Unterbrechungen einer Session zu einem späteren Zeitpunkt am aktuellen Navigationspunkt fortzufahren (siehe Abbildung 24).

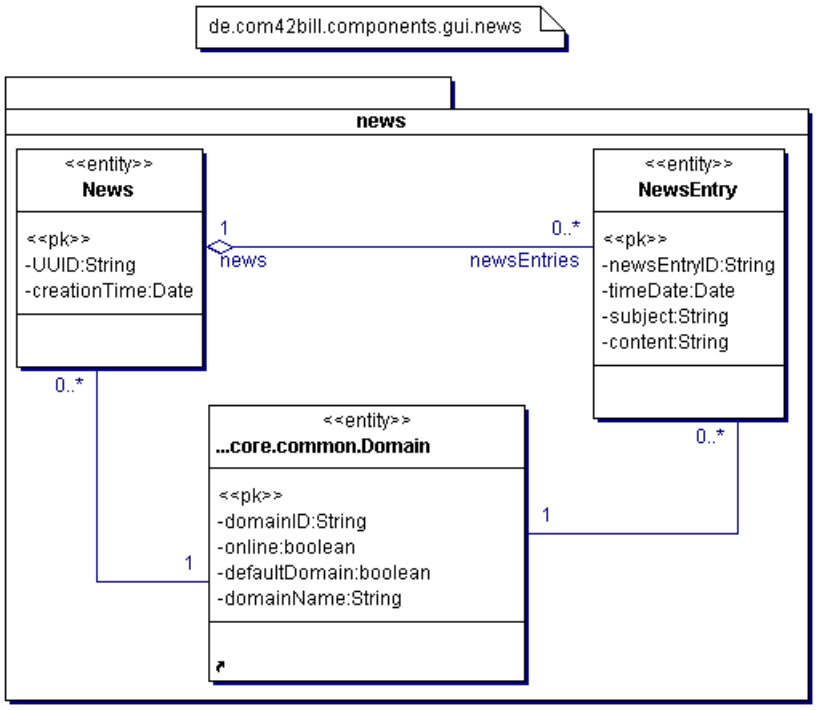

Abbildung 25: Klassen des Package news

Mit der Klasse News des Sub Package *news* ist es möglich, den User mit aktuellen Informationen zum System zu versorgen (siehe Abbildung 25).

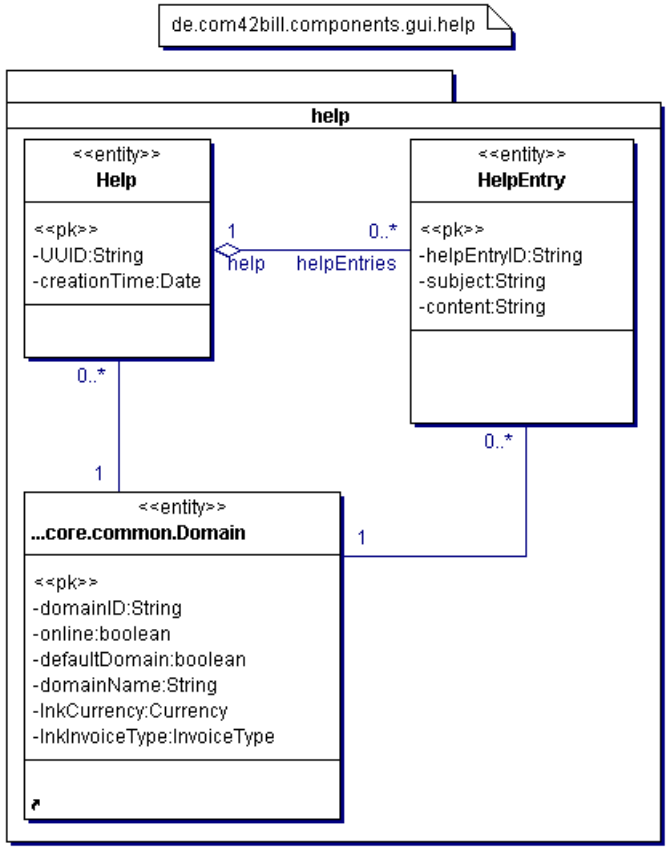

Abbildung 26: Klassen des Package help

Über die Klassen des Sub Packages *help* wird dem Benutzer eine Online-Hilfe zur Bedienung des Systems zur Verfügung gestellt (siehe Abbildung 26).

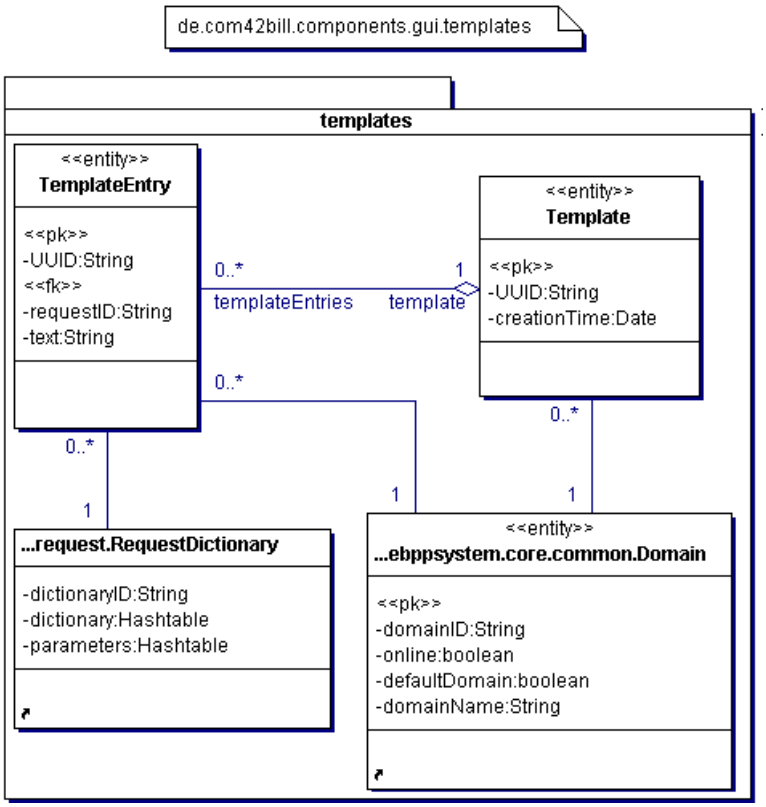

Abbildung 27: Klassen des Package templates

Die Klassen des Package *templates* ermöglichen es, Vorlagen zu hinterlegen (siehe Abbildung 27). Hierbei handelt es sich um XML-Vorlagen, die zur Darstellung der Benutzungsoberfläche benötigt werden.

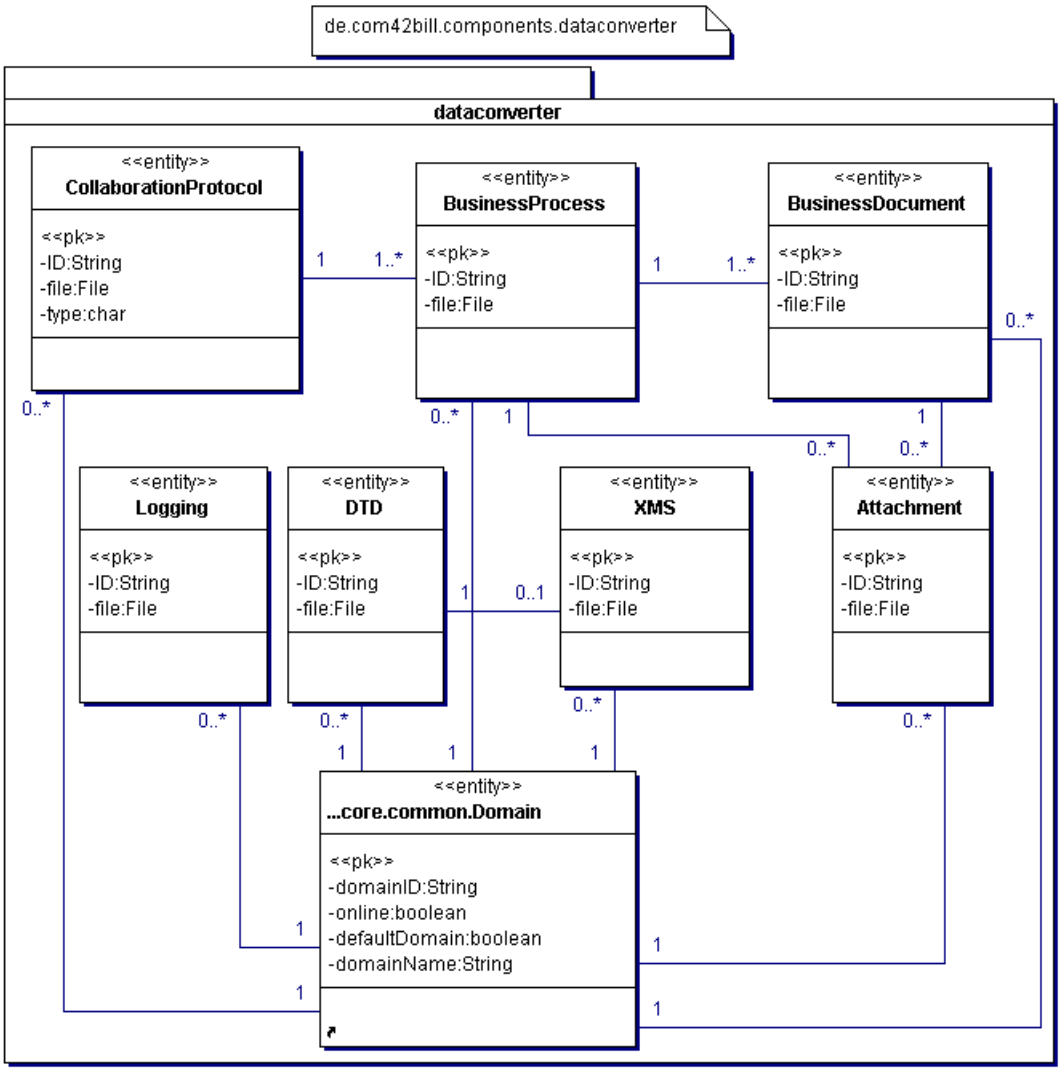

Abbildung 28: Klassen des Package dataconverter

Zur Speicherung von XML-Dateien in verschiedenen Ausprägungen verwendet die Komponente Datenkonverter die Klassen *CollaborationProtocol*, *BusinessProcess*, *BusinessDocument* und *Attachment*. Die XMS und DTD einer XML-Datei werden durch die entsprechenden Klassen repräsentiert und stellen ebenfalls XML-Dateien dar. Zur Speicherung der Protokollierung des Datenaustauschs mit dem Rechnungssteller werden Textdateien verwendet. Diese werden durch die Klasse *Logging* repräsentiert.

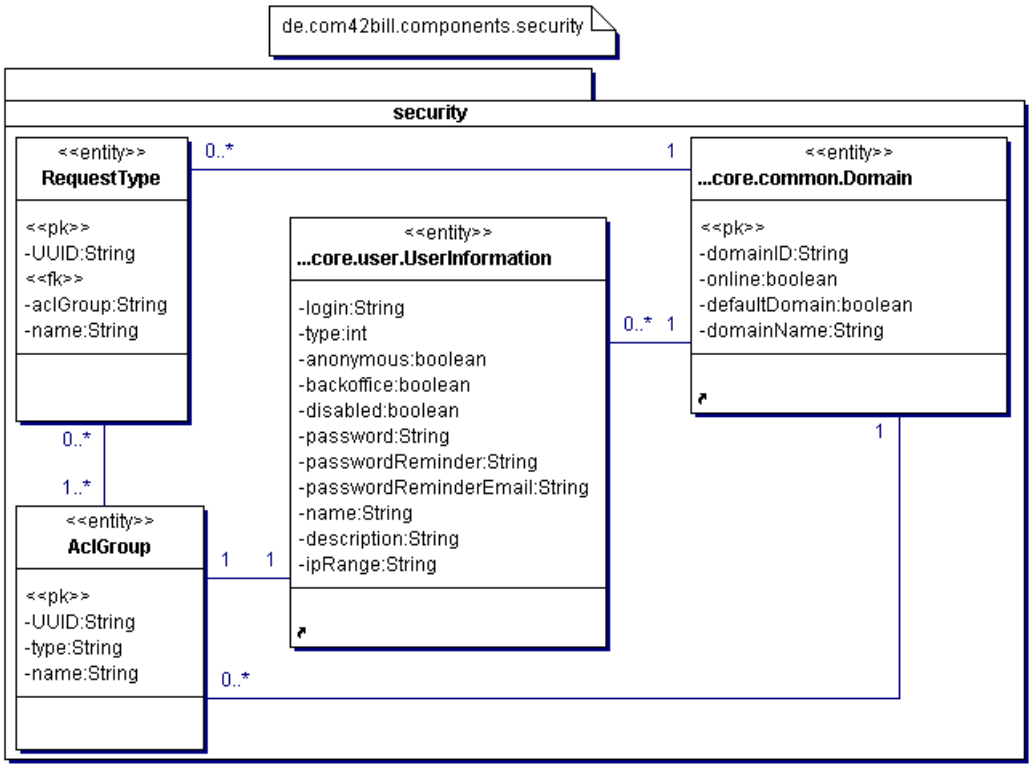

Abbildung 29: Klassen des Package security

Das Package *security* wird in Abbildung 29 dargestellt. Durch seine Klassen werden die Daten modelliert, welche zur Verwaltung der Zugriffsrechte benötigt werden. In *de.com42bill.ebppsystem.core.user.UserInformation* werden die nicht fachlichen Benutzerdaten gespeichert. Über die Klasse *UserInformationAclAssignment* wird eine n:m Kardinalität zwischen *AclGroup* und *core.user.UserInformation* hergestellt. Hierdurch können in einer *AclGroup* mehrere Benutzer enthalten sein. Außerdem kann somit ein Benutzer mehreren *AclGroups* zugeordnet werden. Eine *AclGroup* entspricht einer Benutzergruppe, die den darin enthaltenen Benutzern bestimmte *RequestTypes* erlaubt. Ein *RequestType* entspricht dem Aufruf eines bestimmten Workflows oder einer anderen Aktion. Zwischen *AclGroup* und *RequestType* besteht ebenfalls eine n:m Kardinalität. Jeder *AclGroup* können mehrere *RequestTypes* zugeordnet werden. In umgekehrter Richtung kann jeder *RequestType* mehreren *AclGroups* zugeordnet werden. Die Herstellung dieser Relation zwischen *AclGroup* und *RequestType* übernimmt die Klasse *RequestTypeAclGroupAssignment*. Dieser Aufwand ermöglicht eine sehr variable Rechteverwaltung.

10 Hard- und Softwareinfrastruktur

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.

11 Projektmanagement

In diesem Kapitel wird der Projektplan der Projektgruppe 411 (Com42Bill) beschrieben. Da eine Projektgruppe zwei Semester umfasst, werden sowohl die Aktivitäten des ersten, bereits vergangenen Semesters als auch die noch bevorstehenden Aktivitäten des zweiten Semesters in den Projektplan aufgenommen. Zunächst wird hier die grobe Struktur des Projektplans beschrieben, anschließend werden die wesentlichen Aktivitäten beispielhaft an der Komponente Datenkonverter näher erläutert. Da die Aktivitäten für alle Komponenten dieselben sind, wird auf zusätzliche Erläuterungen für die anderen Komponenten verzichtet.

Die Tabelle im Anhang E illustriert den Projektplan der Projektgruppe Com42Bill. Jede nummerierte Zeile stellt eine Aktivität und jede nicht nummerierte Zeile einen Meilenstein bzw. Tätigkeit einer Querschnittsaufgabe (s.u.) dar. Die erste Spalte dient der Identifizierung der Aktivitäten bzw. Querschnittsaufgaben, in der entsprechenden Identifikatoren eingetragen werden. Jeder Identifikator besteht aus Buchstaben gefolgt von Zahlen. Die Buchstaben DK stehen für die Komponente Datenkonverter, GUI für die Komponente Graphical User Interface, S für die Komponente Sicherheit, BL für die Komponente Business Logic, DB für die Komponente Datenbank, IT für die Integrationstest der Komponenten und QA für die Querschnittsaufgaben. Durch die Zahlen in den Identifikatoren werden die Aktivitäten und Subaktivitäten der Komponenten deutlich gemacht. Diese Art von Identifizierung ermöglicht einen effektiven Überblick über die Aktivitäten und Subaktivitäten einzelner Komponenten. In anderen Worten, der Projektplan wird in sieben Blöcke aufgegliedert. Durch die ersten fünf Blöcke der Tabelle werden die Komponenten, durch den sechsten Block der Integrationstest der Komponenten miteinander und durch den siebten Block die Querschnittsaugaben dargestellt. In der zweiten Spalte der Tabelle wird die Benennung bzw. Kurzbeschreibung der Aktivitäten, Meilensteine und Tätigkeiten der Querschnittsaufgaben vorgenommen. Die fett gedruckten Aktivitäten stellen die einzelnen Schritte der Entwicklung der Komponenten bzw. Namen der Querschnittsaufgaben dar, während die normal gedruckten die Subaktivitäten eines Schrittes wiedergeben. Die Meilensteine werden durch den kursiven Druck deutlich gemacht. Die dritte Spalte gibt die Dauer der Aktivitäten in Tagen an. Durch die vierte Spalte der Tabelle wird der Anfang und durch die fünfte Spalte das Ende einer Aktivität angezeigt. Die Tätigkeiten, die von den Querschnittsaufgaben ausgeführt werden und die, die während des Integrationstests notwendig sind, werden in den letzten zwei Blöcken näher erläutert.

Die Anforderungsanalyse (siehe E.2, DK-01) dauerte insgesamt 41 Tage. Ziel dieser Phase war es, nichtfunktionale Anforderungen für die Komponente Datenkonverter zu ermitteln. Hierbei wurden einige Eigenschaften, die der Datenkonverter besitzen muss, festgehalten. Für nähere Erläuterungen bzgl. Anforderungen siehe Kapitel 5.

In der Phase „Systemarchitektur“ (siehe E.2, DK-02) wurde erst durch die 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 angepasst.

Die Phase Klassenmodell (siehe E.2, DK-03) dient dazu, die nicht-persistenten Klassen der Komponenten in UML-Notation zu modellieren. Der vorgesehene Anfangstermin für diese Aktivität (02.07.2002) wurde nicht berücksichtigt. 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 können.

Die Phase Objektmodell (siehe E.2, DK-04) dient dazu, die persistenten Klassen der Komponente in UML-Notation zu modellieren.

Ziel der Phase Objektdesign (siehe E.2, DK-05) ist es, die Klassen der Komponente detailliert zu beschreiben, d.h. die notwendigen Attribute und Methoden hinzuzufügen.

Die Prototyping-Phase ist dafür vorgesehen, die Key-Features der Komponente mit reduziertem Funktionsumfang zu implementieren bzw. verschiedene Techniken zu testen, um die Machbarkeiten für die Komponente festzustellen. Hierbei sollte darauf geachtet werden, die Key-Features für die Komponente so auszuwählen, dass ein Gesamtdurchlauf des Systems, d.h. der Einbezug aller Komponenten, ermöglicht wird.

In der Implementierungs-Phase (siehe E.2, DK-07) findet die eigentliche Implementierung statt, wobei hier die Erkenntnisse und der Code aus der Prototyping-Phase einfließen sollten. Die Komponente wird in dieser Phase mit allen vorgesehenen Features vollständig implementiert und dokumentiert.

Die Subkomponententest-Phase (siehe E.2, DK-08) ist dafür vorgesehen, Subkomponenten jeder Komponente für sich bzgl. der verlangten Funktionalitäten zu testen. Nach dieser Phase sollten alle Subkomponenten der Komponente die gewünschten Funktionalitäten leisten.

Die Phase Integration (Subkomponenten) (siehe E.2, DK-09) ist für die Integration der Subkomponenten jeder Komponente vorgesehen. Hierbei werden lediglich die Subkomponenten einer Komponente miteinander integriert, ohne die Gesamtintegration aller Komponenten zu testen.

Jede Komponente wird nach Integration seiner Subkomponenten einem Test unterzogen (siehe E.2, DK-10). Hierbei wird die Komponente losgelöst vom Gesamtsystem auf ihre Funktionalitäten geprüft.

Die Integrationstest-Phase (siehe E.7) dient dazu, die Komponenten des Gesamtsystems miteinander zu integrieren und anschließend zu testen. Hierbei sollte darauf geachtet werden, dass die gewünschten Funktionalitäten einer Komponente im Gesamtsystem korrekt ausgeführt werden.

Da zu den Entwicklertätigkeiten auch weitere Aufgaben in einer Projektgruppe zu erledigen sind, haben alle Teilnehmer noch sogenannte Querschnittsaufgaben. Auch diese sind im Projektplan aufgeführt (siehe E.8).

Der Projektplan wurde während der Anfangsphase der Projektgruppe erstellt und muss kontinuierlich durch das Projektmanagement aktualisiert werden. In einem Rhythmus von ca. vier Wochen wird der Projektplan in der Projektgruppensitzung besprochen. Des weiteren wurde durch das Projektmanagement ein Anforderungsdokument erstellt, dessen Pflege an das Qualitätsmanagement übergeben wurde.

Zu den Tätigkeitsfeldern der Querschnittsaufgabe Marketing (siehe E.8, QA-02) gehört u. a. die Organisation. Im ersten Semester der Projektgruppe hat das Marketingteam die Bestellung von Projektgruppen T-Shirts organisiert und das Whitepaper entworfen. Z. Zt. beschäftigt sich das Marketingteam mit der redaktionellen Bearbeitung des Zwischenberichtes. Zudem ist das Marketingteam für die Kontaktaufnahme mit den Firmen verantwortlich sowie für die Erstellung eines Plakats. Das Marketingteam erstellt auch den Style Guide für Web und WAP.

Die Mitglieder der Querschnittsaufgabe Qualitätsmanagement (siehe E.8, QA-003) sind u. a. verantwortlich für den Entwurf eines Code Guide. Diese Aktivität wurde recht frühzeitig in den ersten Semesterwochen der Projektgruppe von ihnen erledigt. Sie stellten die Programmierrichtlinien von Sun vor; diese wurden von allen Projektgruppenmitgliedern als verbindlich für das Projekt akzeptiert. Desweiteren sind sie dafür verantwortlich, Dokumente

wie den Projektplan, Anforderungsdokument und Key-Features bzgl. ihrer Richtigkeit zu prüfen. Die Pflege und Wartung des Prozessmodells muss ebenfalls während der gesamten Projektgruppenzeit von dem Qualitätsmanagement erledigt werden. In dem zweiten Semester der Projektgruppe wird das Qualitätsmanagement jeweils einen Testplan für die Integrationstest und Systemtest entwickeln.

Die Querschnittsaufgabe GUI (siehe E.8, QA-004) wurde in der Mitte des ersten Projektgruppensemesters aufgelöst; das Team ging in den Querschnittsaufgaben Marketing und Qualitätsmanagement auf. Der Grund für die Auflösung der Querschnittsaufgabe GUI war der geringe Tätigkeitsumfang durch die weite Überschneidung der Aufgaben mit den Bereichen Marketing und GUI-Entwicklung.

Die Chef-Architekten (siehe E.8, QA-05) 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 wird von den Chef-Architekten gepflegt und aktualisiert. Zudem müssen 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.

Zu den Aktivitäten von Systemadministratoren (siehe E.8, QA-06) gehören u. a. die Einrichtung von Server, Workstations und CVS. Des weiteren sind sie für die Installation und Einrichtung der Entwicklungsumgebung bzw. neuer Software zuständig. Außer dieser Tätigkeiten haben die Systemadministratoren bisher die News-Group und FTP-Zugang für die Projektgruppe eingerichtet.

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
Sicherheitsserver	Bastian Schlich, Dennis Müller
GUI	Alexander Schmitz, Matthias Niggemeier, Dino Hasanbegovic
Querschnittsaufgabenzuständigkeiten	
Systemadministration	Stefan Pinschke, 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 Sommersemesters ihre Querschnitssaufgaben.

12 Qualitätsmanagement

12.1 Allgemein

Das Ziel der Projektgruppe 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 nicht nur die Kontrolle auf Fehler am Endprodukt, die dann beseitigt werden müssen, sondern auch die Untersuchung der Ursache solcher Fehler. 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 prozessorienierte Ansatz des Qualitätsmanagements geht von dem Grundsatz aus, dass ein qualitativ hochwertiger Prozess auch mit hoher Wahrscheinlichkeit ein qualitativ hochwertiges Produkt hervorbringt [Ste00].

Das Qualitätsmanagement der Projektgruppe Com42Bill legt den Schwerpunkt auf die Qualitätssicherung und die Optimierung der Softwareentwicklung. Für diese Aufgabe werden im weiteren Verlauf der Projektarbeit Tests und Reviews entwickelt und durchgeführt. Zusätzlich ist es nach dem prozessorientiertem Ansatz des Qualitätsmanagements notwendig, den Softwareentwicklungsprozess zu dokumentieren und zu analysieren. Ein wichtiges Mittel für diese Analyse des Entwicklungsprozesses ist das Aufstellen eines Prozessmodells.

12.2 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, so wie er bisher erfolgt ist, und gibt dazu noch einen Ausblick auf die geplanten Tätigkeiten im weiteren Verlauf des Projekts.

12.3 Symbolik

Für die Dokumentation des Softwareentwicklungsprozesses der Projektgruppe Com42Bill wurde die Darstellung gewählt, die an die Petri-Netz-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 bzw. mehrere Ergebnisse 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 30 zu sehen.

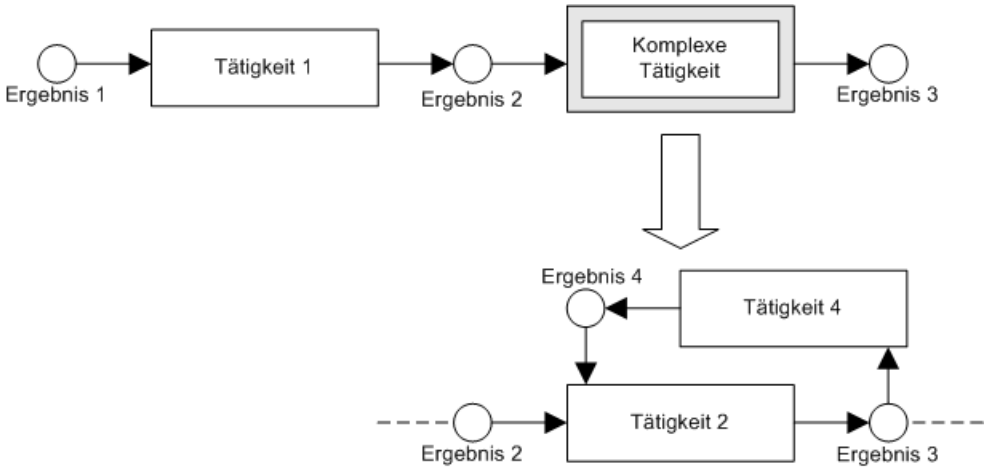

Abbildung 30: Notation

12.4 Entwicklungsprozess

Das Diagramm Entwicklungsprozess, das in Abbildung 31 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 komponentenbasierend 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 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. Auf die einzelnen Phasen so wie auf die unterstützenden Prozesse wird in den folgenden Abschnitten ausführlich eingegangen.

Zu dem Zeitpunkt der Erstellung dieses Zwischenberichts befinden wir uns in dem Entwicklungsprozess kurz vor dem Beginn des Prototypings.

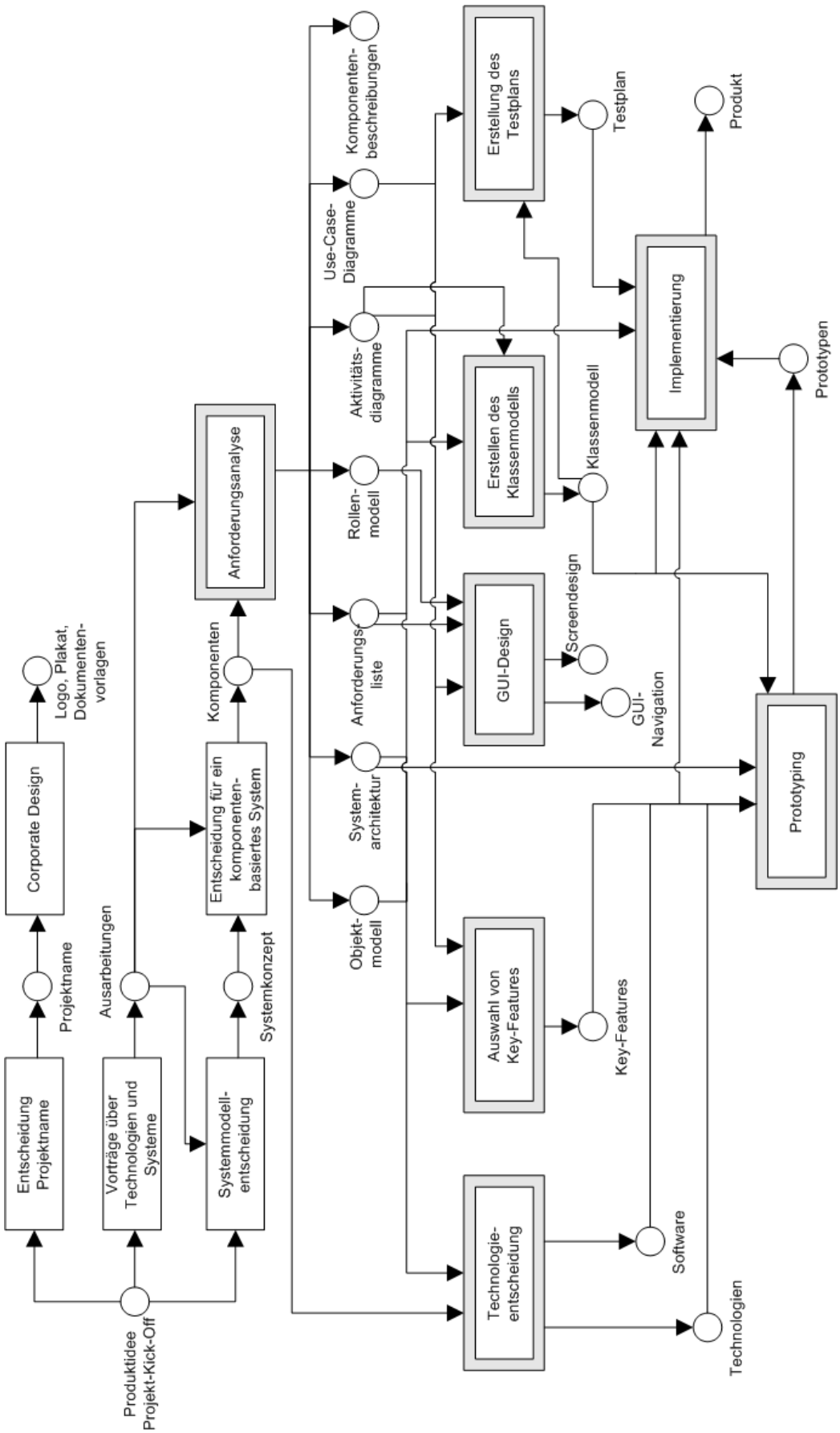
Produktidee
Projekt-Kick-Off
Entscheidung Projektname
Vorträge über Technologien und Systeme
Systemmodell-entscheidung
Projektname
Ausarbeitungen
Systemkonzept
Corporate Design
Logo, Plakat, Dokumenten-vorlagen
Entscheidung für ein komponenten-basiertes System
Komponenten
Anforderungsanalyse
Use-Case-Diagramme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

Abbildung 31: Entwicklungsprozess

12.5 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, wurde in weiteren Schritten 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 aus den Anforderungen wieder in die Modelle übernommen wurden, waren nötig, um die Anforderungsliste 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.

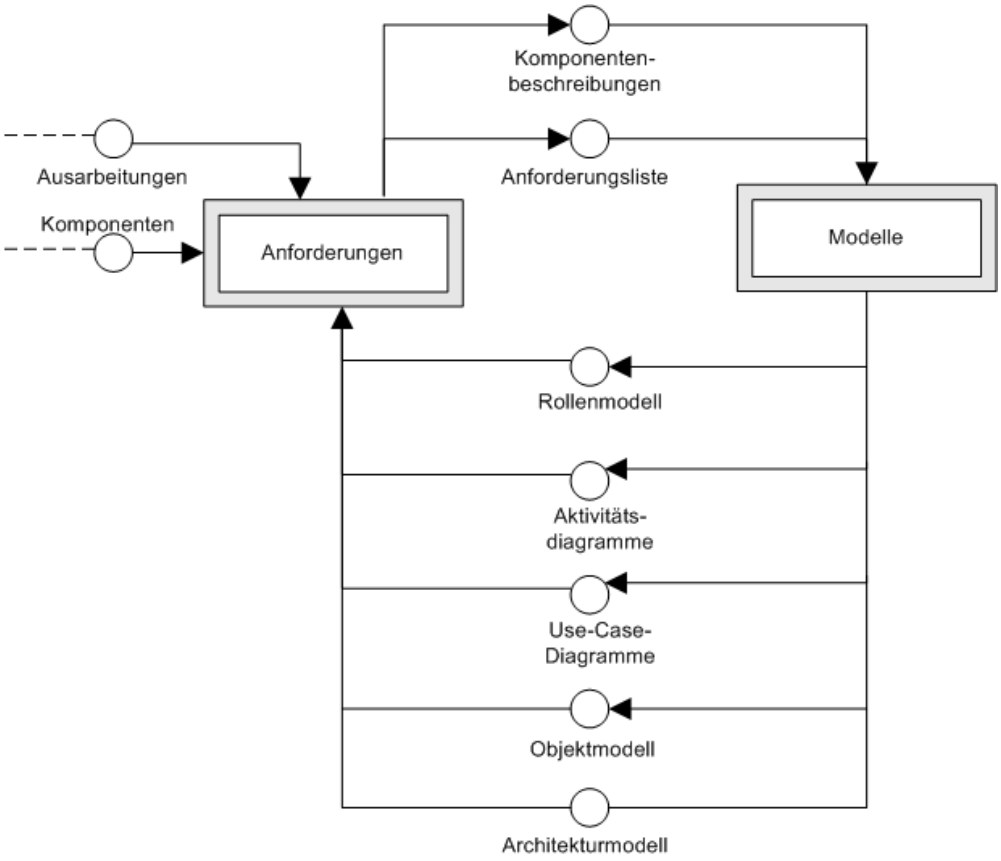

Abbildung 32: Anforderungsanalyse

12.6 Anforderungen

Ausgehend von der Entscheidung für Komponenten wurde versucht, Anforderungen an diese Komponenten aufzustellen. Für ein besseres Verständnis der Komponenten wurde zunächst eine textuelle Beschreibung der Komponenten erstellt. Diese Beschreibungen wurden durch Reviews bereits 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.

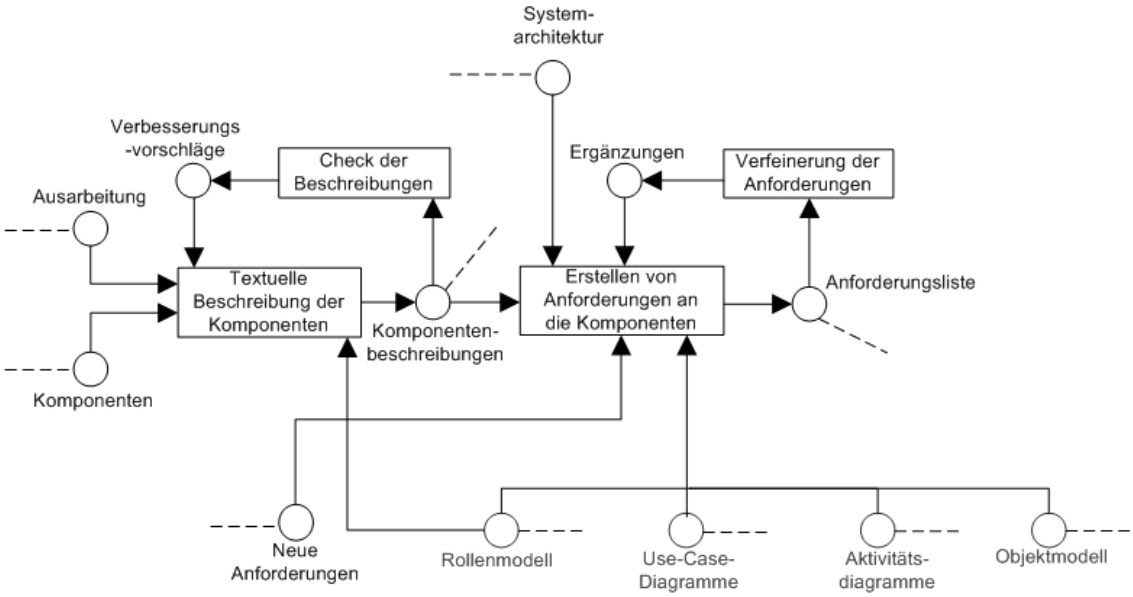

Abbildung 33: Anforderungen

12.7 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 und wird.

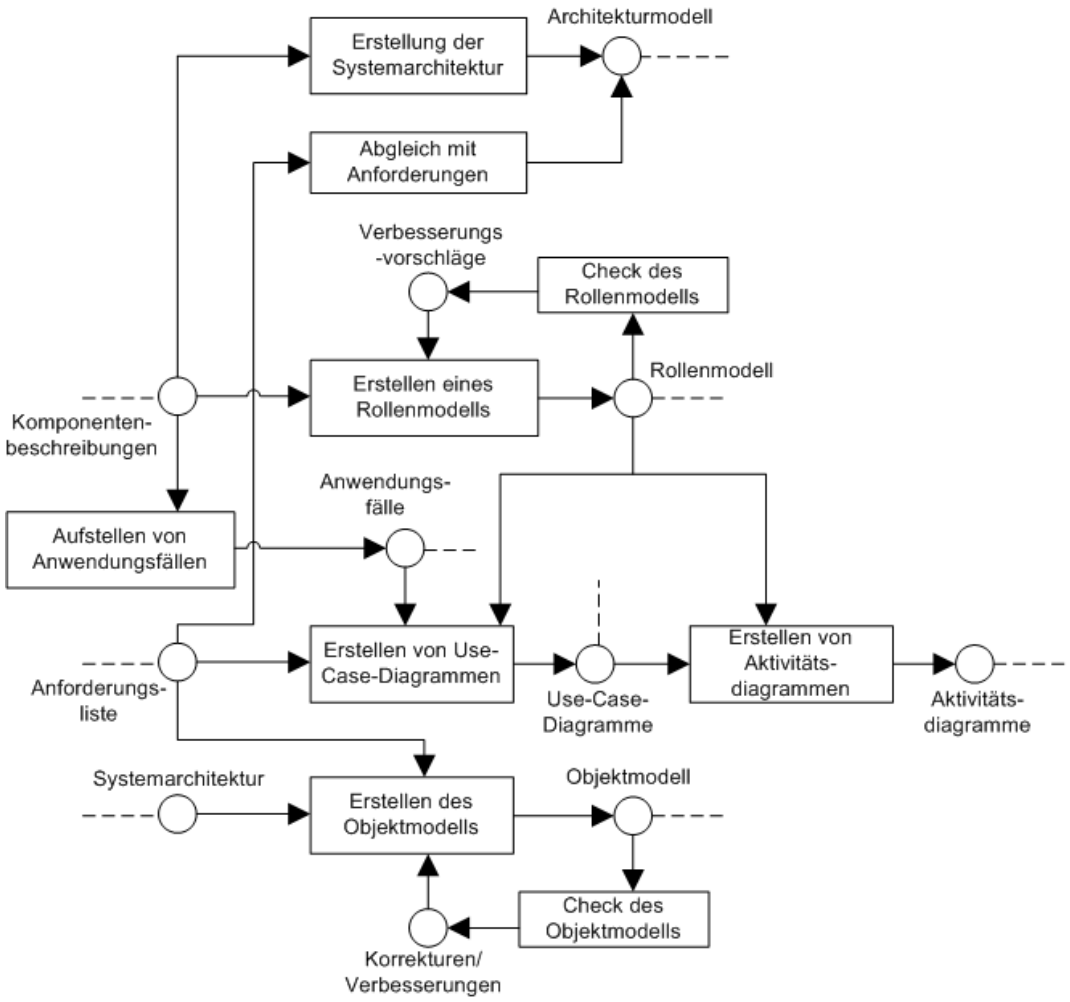

Abbildung 34: Modelle

12.8 Technikentscheidung

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 der Anforderungsliste ergab, getroffen werden. So wurde von der Komponente Business Logic beispielsweise eine Pipelinearchitektur für die Umsetzung der Geschäftsprozesse gewählt.

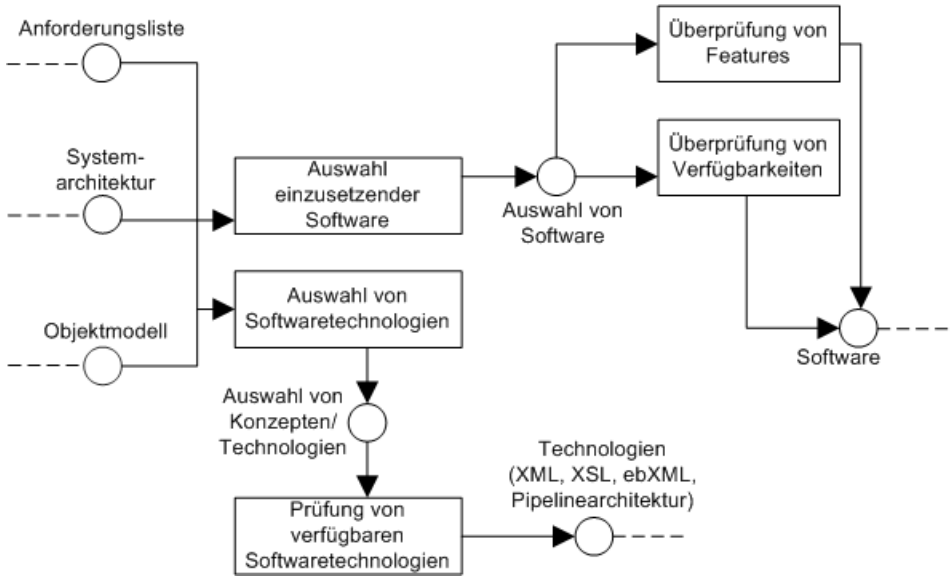

Abbildung 35: Technologieentscheidung

12.9 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.

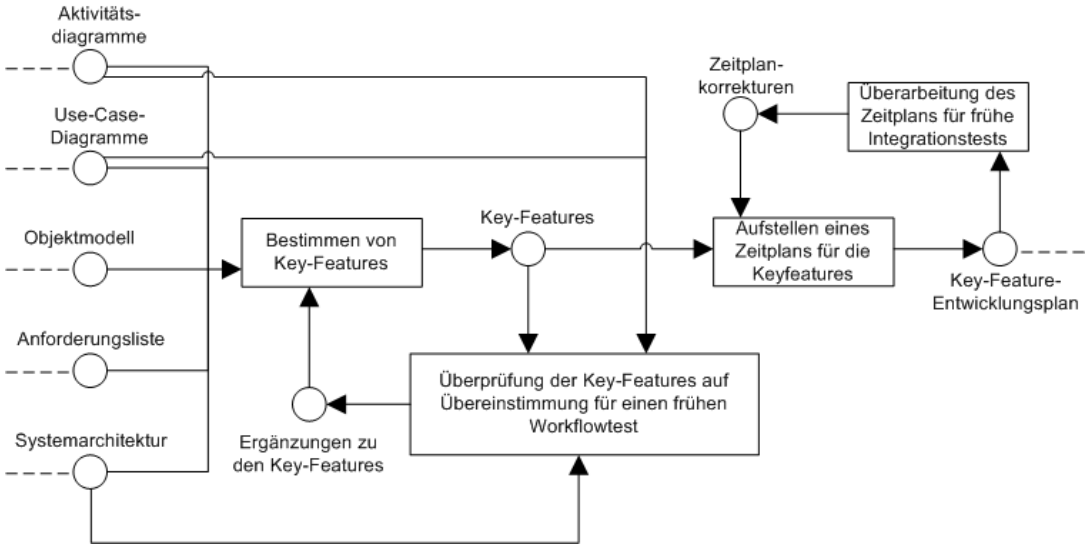

Abbildung 36: Auswahl von Key-Features

12.10 GUI-Design

Die Erstellung der Benutzungsoberflächen teilt 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 ergeben das entstandene Screendesign der Benutzeroberfläche.

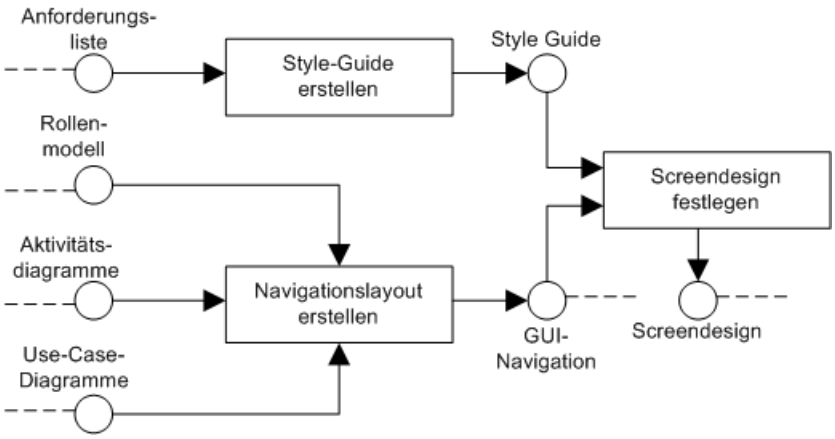

Abbildung 37: GUI-Design

12.11 Klassenmodell

Für das Prototyping und die Implementierung wird ein Klassenmodell benötigt. Dieses ergibt sich sehr leicht aus dem Objektmodell und der Systemarchitektur, die erforderlichen Schnittstellen und Übergabeparameter können aus den Anforderungen und den Aktivitätsdiagrammen ermittelt werden. Eine Kontrolle und Überarbeitung des Klassenmodells und besonders der Schnittstellen sind für eine spätere Zusammenarbeit der Komponenten unabdingbar, so dass an dieser Stelle ein evtl. mehrfaches durchzuführendes Review notwendig ist.

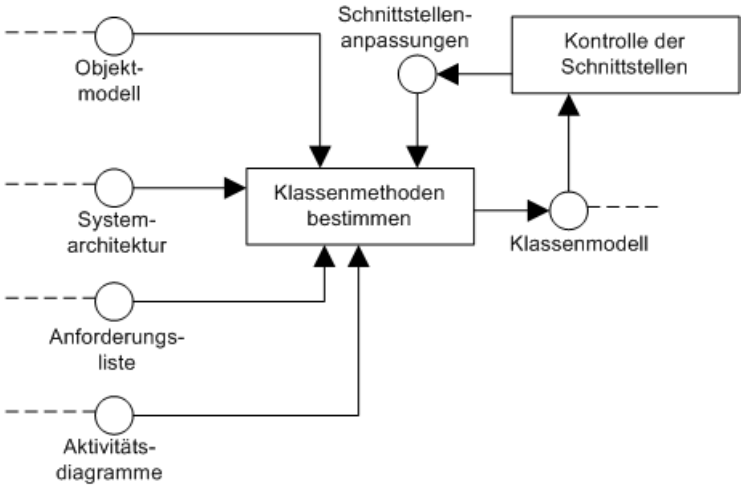

Abbildung 38: Klassenmodell

12.12 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. Aus dem Klassenmodell werden dazu Klassen für Tests und Reviews ausgewählt, für diese Klassen werden Testdaten bestimmt. Die Zusammenarbeit

der Klassen wird mittels Workflowtests geprüft, diese ergeben sich aus den Use-Case- und Aktivitätsdiagrammen. Für diese Tests werden durchführende Personen ausgewählt und die zeitliche Planung in einem Testplan festgehalten.

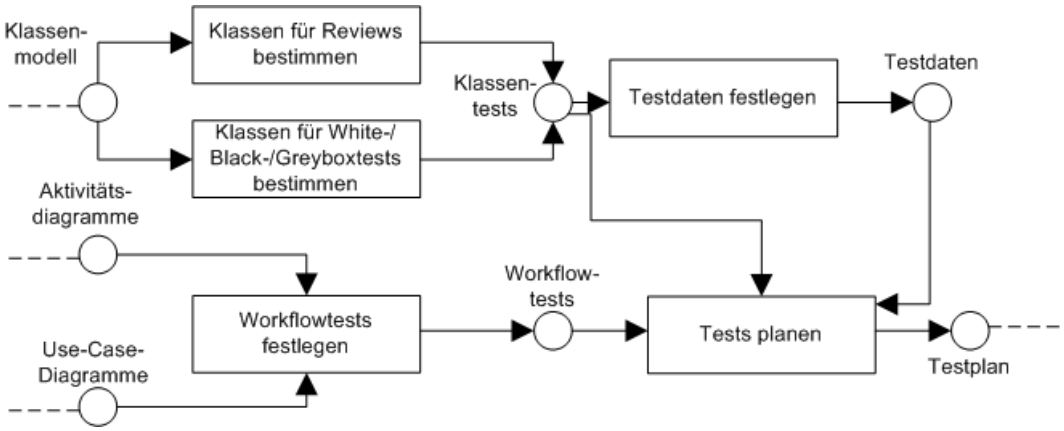

Abbildung 39: Testplan

12.13 Prototyping

Das Prototyping ist in zwei Stufen unterteilt. Die erste Stufe dient der Prüfung der verwendeten Technologie und der Machbarkeit. Das Ergebnis dieser Stufe sind bereits rudimentäre Prototypen. Erkenntnisse, die man aus der Implementierung dieser Prototypen gewonnen hat, können wiederum zu einer Veränderung dieser Prototypen führen, so dass hier mehrere Iterationen möglich sind.

Ist die technische Machbarkeit geklärt, so besteht die zweite Stufe aus der Implementierung eines funktionalen Prototypen. Auch die Erkenntnisse aus diesem Prototypen gehen in die Verbesserung ein. Die funktionalen Prototypen sollen 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.

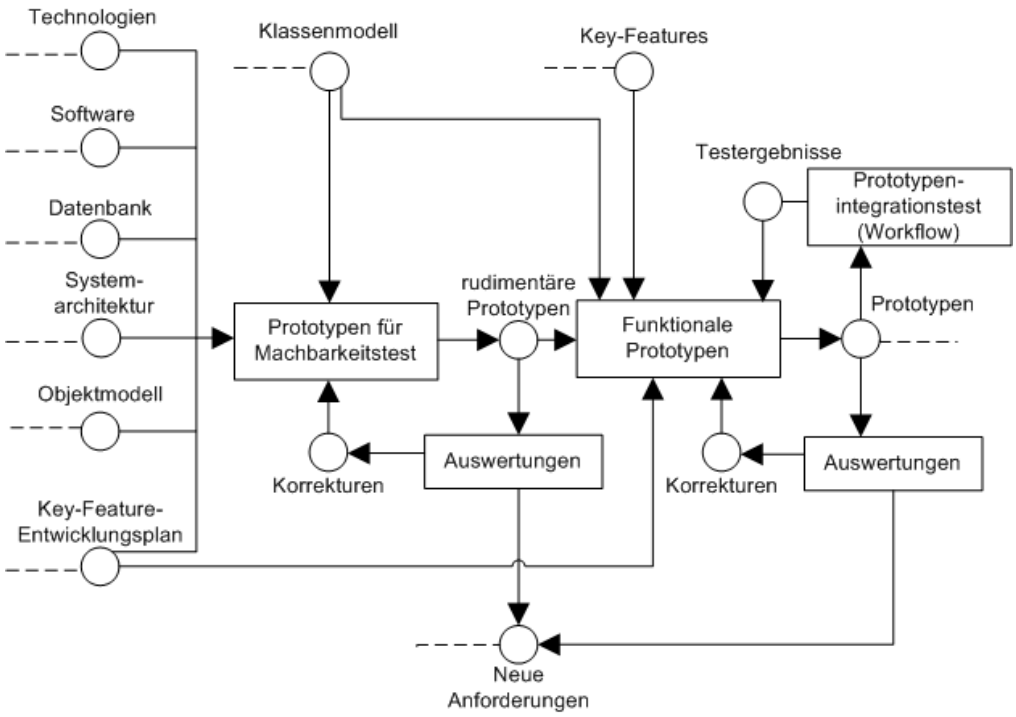

Abbildung 40: Prototyping

12.14 Implementierung

Die Implementierung stellt die vorletzte Phase des Entwicklungsprozesses dar. Mit den Ergebnissen des Prototyping sollte die Implementierung schnell zu funktionsfähigen Komponenten führen, diese werden laut dem Testplan Tests unterzogen, entdeckte Fehler werden sofort korrigiert und der Test wiederholt.

Die getesteten Komponenten werden nun schrittweise zusammengeführt, die entstehenden Teilsysteme werden wiederum getestet. Die bei den Integrationstests entdeckten Fehler müssen bei der Implementierung der Komponente, in der der Fehler entdeckt wurde, behoben werden. Bei diesen Tests geht es vor allem darum, die Schnittstellen zwischen den Komponenten zu testen.

Sind alle Teilsysteme getestet worden, so ist der nächste logische Schritt die Gesamtintegration des Systems. Nachdem alle Teilsysteme bereits getestet wurden, ergibt die Zusammenführung aller Komponenten ein erstes lauffähiges System. Dieses System wird einem Gesamtsystemtest unterzogen, Fehler dieses Tests müssen wieder selbstverständlich in den Komponenten selbst korrigiert werden. Auch bei diesem Test liegt ein besonderes Augenmerk auf der fehlerfreien Zusammenarbeit der Komponenten.

Konnten die Gesamtsystemstests fehlerfrei durchgeführt werden, kann das System als fertiges Produkt freigegeben werden.

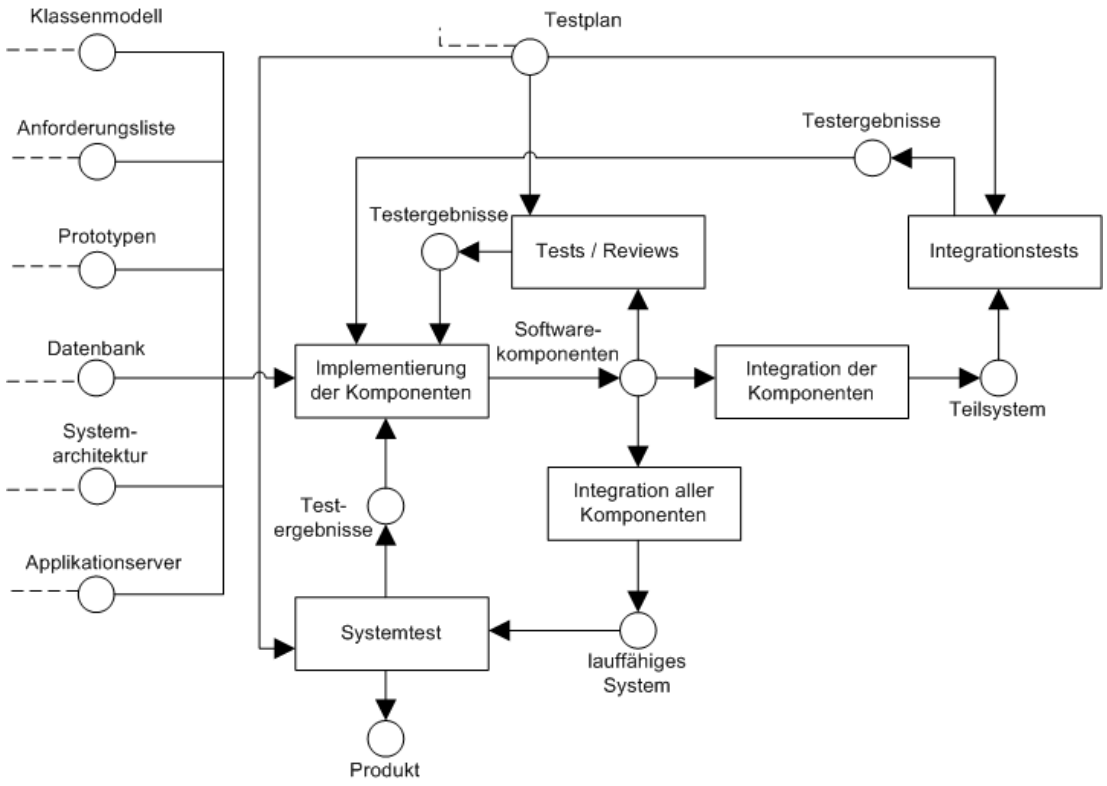

Abbildung 41: Implementierung

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. [Mül02]

13 Ausblick

Das Hauptziel der Projektgruppe COM42BILL ist die Fertigstellung des EBPP-Portals bis Ende Februar 2003. Dieser Zeitraum umfasst mehrere, in dem Projektplan bereits festgelegte Phasen der endgültigen Implementierung und der Analyse unserer E-Commerce-Lösung.

Die in der ersten Hälfte abgeschlossenen Phasen des objektorientierten Entwurfs mithilfe von UML werden als essenzielles Hilfsmittel für die bevorstehende Implementierung der Prototypen einzelner Komponenten des Portals vorausgesetzt. Basierend auf Ablaufszenarien können schnell und effizient Testläufe rekonstruiert und eingesetzt werden. Die Klassenhierarchie inklusive zugehöriger Methoden kann jederzeit den Klassendiagrammen entnommen werden.

Dem eigentlichen Prototyping geht eine kurze Testphase voraus, in der die neue Technologie zunächst analysiert und auf ihre Funktionalität und Nutzen hin geprüft wird. Dies wird durch die Implementierung einiger ausgewählter und insbesondere auf diese Testphase zugeschnittener Keyfeatures der jeweiligen Komponenten geschehen.

Die im Anschluss entworfenen Prototypen werden intensiven Tests unterzogen, bei welchen sie auf ihre Kompatibilität und Interaktion hin untersucht werden. Mit so gewonnenen Informationen werden entstandene Diskrepanzen und Fehler beseitigt, um eine optimale und reibungslose Integration der finalen Komponenten bereits vor deren eigentlicher Fertigstellung zu garantieren.

Zu guter letzt werden alle Komponenten endgültigen Tests unterzogen. Nach einem erfolgreichen Testabschluss verlassen die Komponenten ihre Beta-Phase und können als marktreif betrachtet werden. In einer letzten Präsentation wird das COM42BILL-Portal dem breiten Publikum, bzw. den potentiellen Interessenten vorgestellt.

Da während der gesamten Implementierungsphase Änderungen jeglicher Form nicht ausgeschlossen werden können, ist davon auszugehen, dass der Prozessleitfaden sowie der Projektplan den ggf. entstandenen Änderungen angeglichen und somit stets aktualisiert werden.

Parallel zum primären Ziel der Fertigstellung unseres EBPP-Portals sind zusätzliche Marketingstrategien notwendig, um den Bekanntheitsgrad und Attraktivität unseres Projektes in der Wirtschafts-, aber auch in der Verbraucherwelt zu erhöhen. Die ersten Kontakte mit der Commerzbank wurden bereits geknüpft, viele weitere sollen folgen. Es werden mehrere Besuche bei diversen Kreditinstituten stattfinden; dort soll eine Demonstration des COM42BILL-Portals erfolgen, in deren Rahmen auf enorme Vorteile der EBPP-Technologie gerade im Finanzwesen hingewiesen werden soll. Zusätzlich sollen Merchandising-Artikel den Bekanntheitsgrad des Portals unter den Endverbrauchern erhöhen und somit ihr Interesse für diese einfach zu handhabende und zukunftsorientierte Technologie wecken.

Anhang A: Rollenmodell

A.1: Einleitung

In einem Rollenmodell werden die Beteiligten eines Systems näher beschrieben. In diesem System gibt es vier verschiedene Rollen. Es gibt den Betreiber (BT), dem Rechnungssteller (RS), den Rechungsempfänger (RE) und den Finanzdienstleister (FD). Der Betreiber betreibt das EBill-Presentmentsystem. Der Rechnungssteller übermittelt an den BT die Rechnungen, welche von verschiedenen RE zu begleichen sind. Der RE benutzt das System, um die Bezahlung seiner Rechnungen zu veranlassen. Der FD nimmt die Aufforderungen zur Zahlung entgegen und führt diese durch. Der einzige indirekte Zugriff in diesem System ist wird benötigt, um dem RE genauere Daten seiner Rechnung zur Verfügung zu stellen. Dazu werden ihm Daten präsentiert, welche vom RS zur Verfügung gestellt werden. Wir verarbeiten diese Daten nicht sondern zeigen diese lediglich an. Dazu greifen wir auf die Dienste des RS zurück.

Ein RS stellt sich gegenüber dem BT als ein oder mehrere Benutzer da. Der RE ist immer genau ein Benutzer. Jeder Benutzer vollzieht den Datenaustausch über eine Software, die im weiteren Klient genannt wird. Dies ist für den RE z.B. ein WWW-Browser.

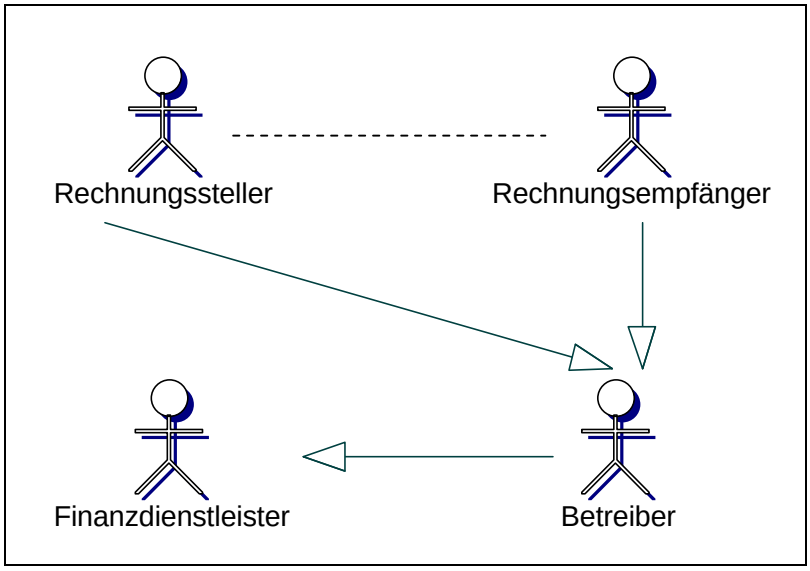

Abbildung 42: Übersicht

A.2: Rechnungsempfänger

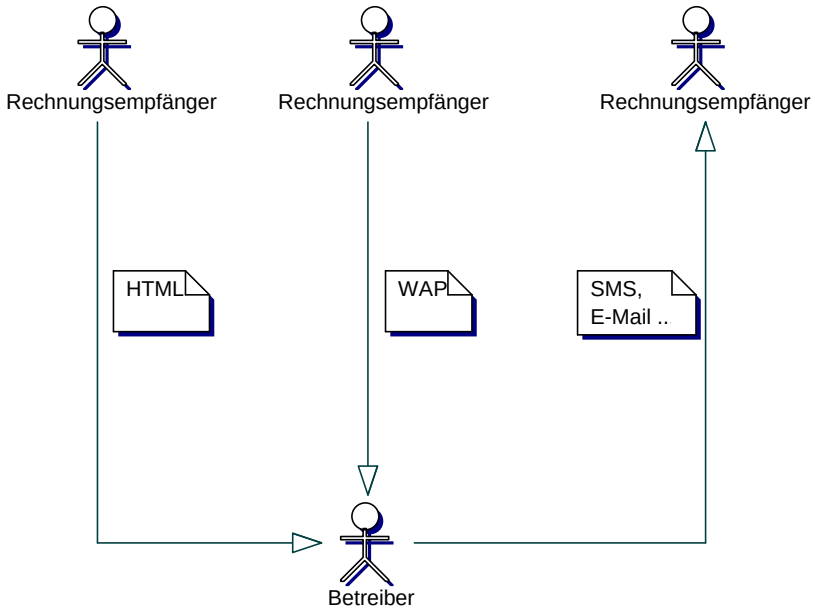

Abbildung 43: Rechnungsempfänger

Der RE kann auf die Dienste des BT über die Online-Schnittstelle zugreifen. Er hat zwei Möglichkeiten diese Schnittstelle zu verwenden. Die eine Möglichkeit ist der Zugriff per Webbrowser und die andere Möglichkeit ist der Zugriff über ein WAP-fähiges Gerät. Wenn der RE per Webbrowser auf die Dienste des Betreibers zugreift, bekommt er HTML Seiten angezeigt und kann diesen die gewünschten Informationen entnehmen. Er kann ebenfalls über verschiedene Dialoge Rechnungen bezahlen oder die Zahlung auf einen späteren Zeitpunkt verschieben. Per WAP-Zugriff hat der RE dieselben Möglichkeiten. Allerdings ist der Zugriff aufgrund der begrenzten Möglichkeiten von WAP nicht so komfortabel.

Des weiteren kann der RE vom BT Meldungen empfangen. Der BT versendet diese zu besonderen Ereignissen wie zum Beispiel dem Eintreffen einer neuen oder Mahnung einer fälligen Rechnung. Zum Versenden dieser Meldungen werden gebräuchliche Methoden wie z.B. SMS oder E-Mail verwendet.

A.3: Finanzdienstleister

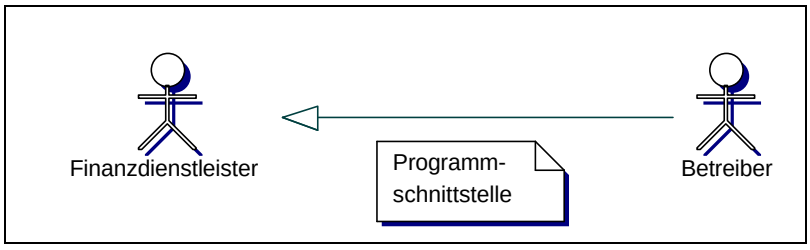

Abbildung 44: Finanzdienstleister

Der Betreiber übermittelt die verschiedenen Zahlungsanweisungen per Programmschnittstelle an den Finanzdienstleister. Diese Übermittlung soll ohne menschliche Interaktionen erfolgen. Da das System an dieser Stelle auf die Protokolle der FD einstellt werden muss, kann diese

Schnittstelle nicht präziser allgemein definiert werden. In wieweit der FD Informationen über den Status der Zahlungen übermitteln kann, muss noch eruiert werden.

A.4: Rechnungssteller

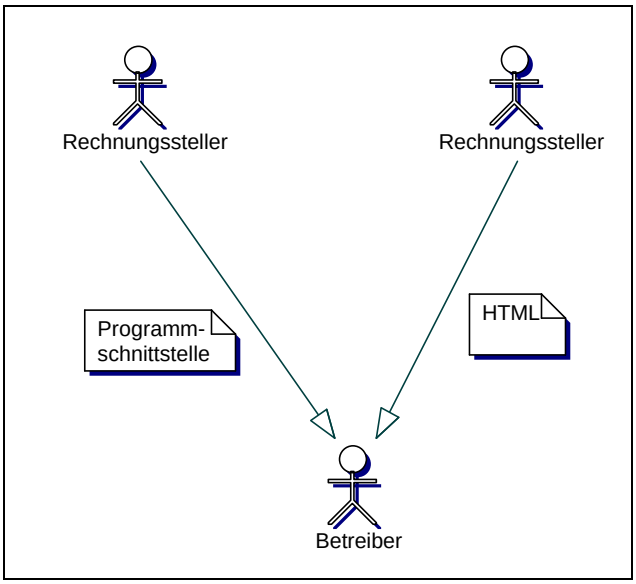

Abbildung 45: Rechnungssteller

Der Rechnungssteller kann über vier Wege auf die Dienste des Betreibers zugreifen. Diese vier Wege können in zwei Kategorien eingeteilt werden. Es gibt auf der einen Seite die Onlineschnittstellen, auf der anderen Seite steht die Programm Schnittstelle. Die Onlineschnittstellen teilen sich in die Möglichkeiten HTML, Applet und Applikation auf. Die Programm Schnittstelle wird nicht weiter aufgeteilt.

Anhang B: Use Cases

B.1: Rechnungssteller

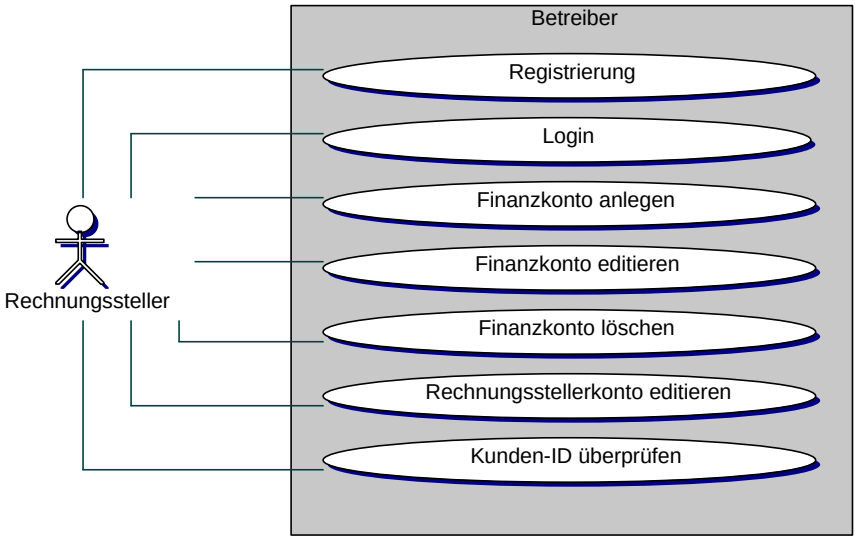

Abbildung 46: Use Case Rechnungssteller

B.2: Rechnungsempfänger

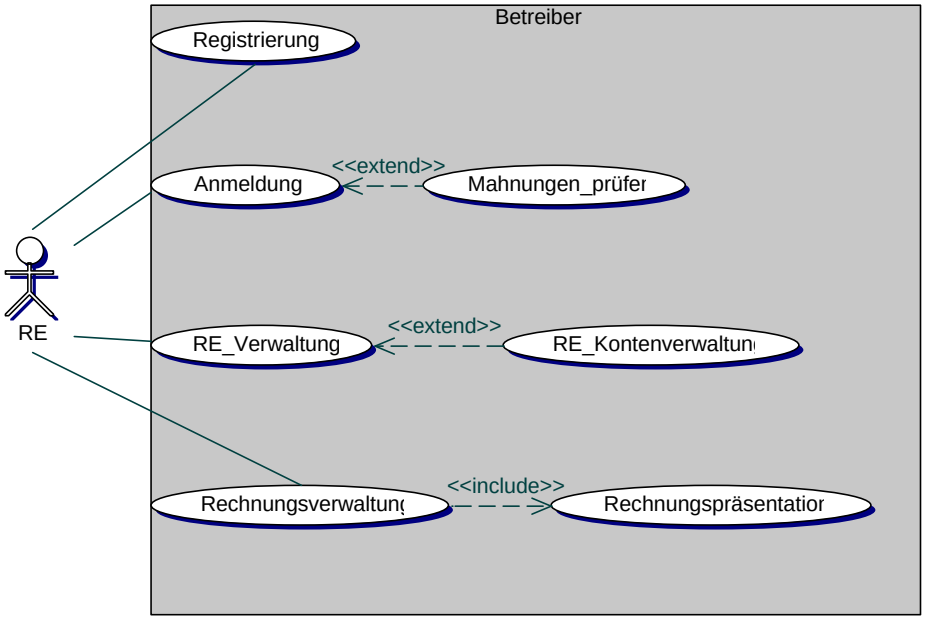

Abbildung 47: Use Case Rechnungsempfänger

B.3: Betreiber

B.3.1: Allgemeine Verwaltung

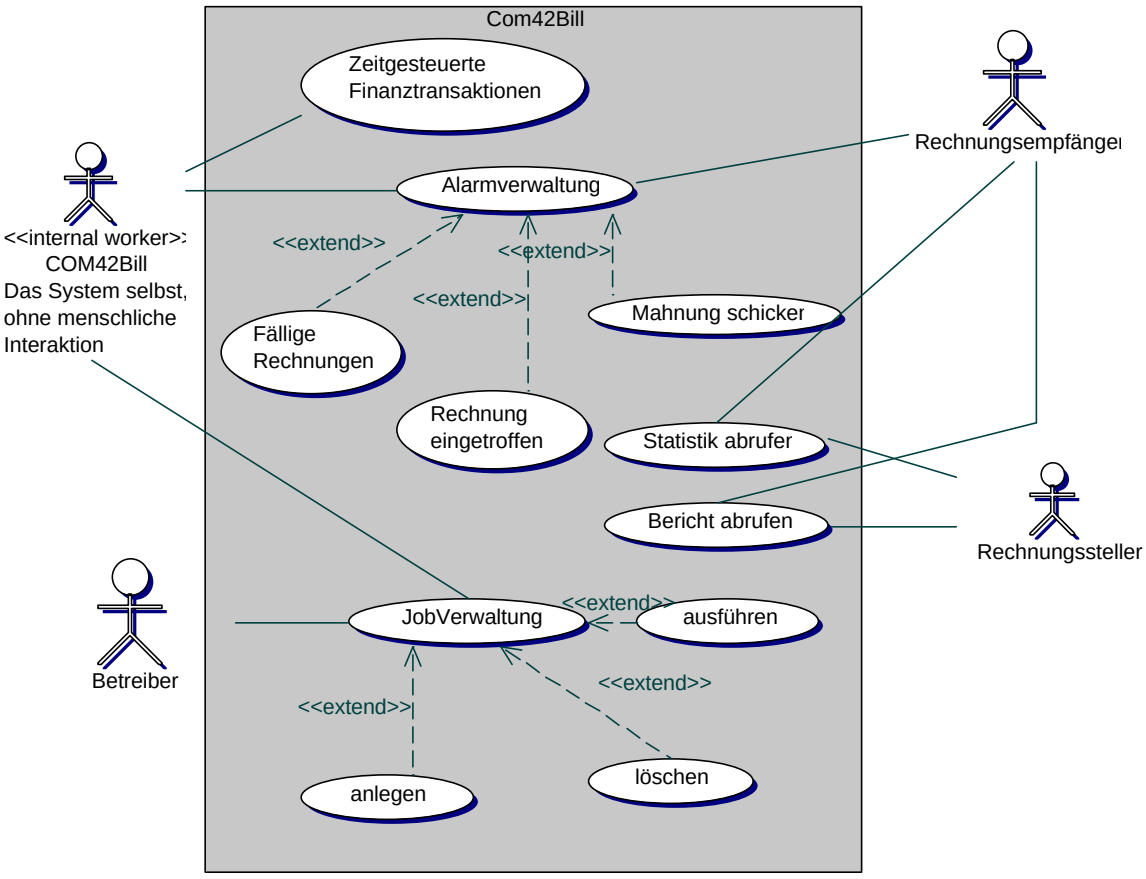

Abbildung 48: Use Case Verwaltung

B.3.2: Konfiguration Datenaustausch

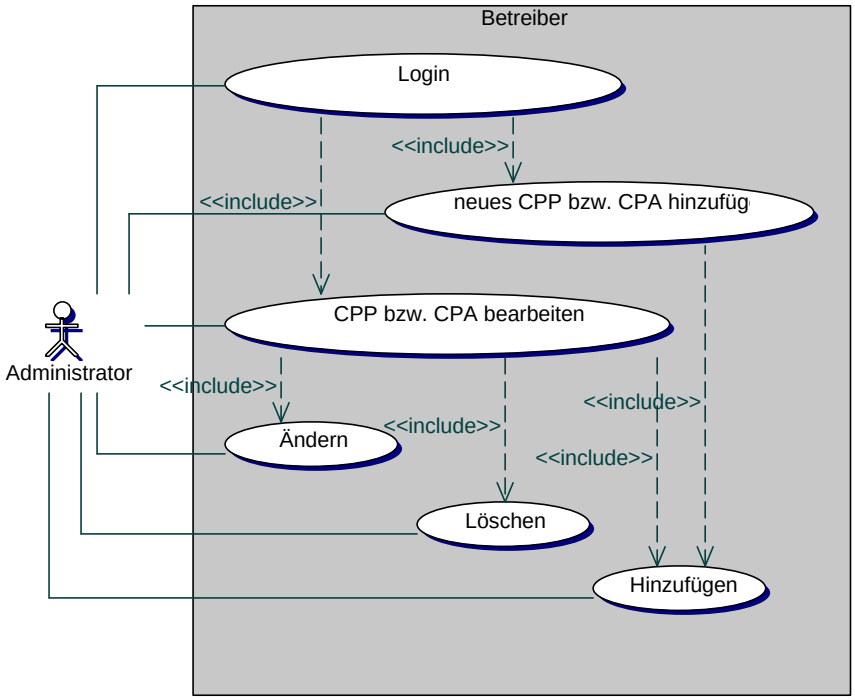

Abbildung 49: Use Case Konfiguration Datenaustausch

B.4: Benutzerverwaltung

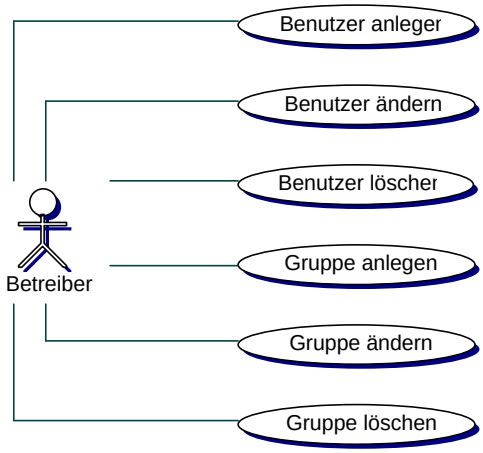

Abbildung 50: Use Case Benutzerverwaltung

Anhang C: Aktivitätsdiagramme

C.1: Rechnungsempfänger

C.1.1: Registrierung

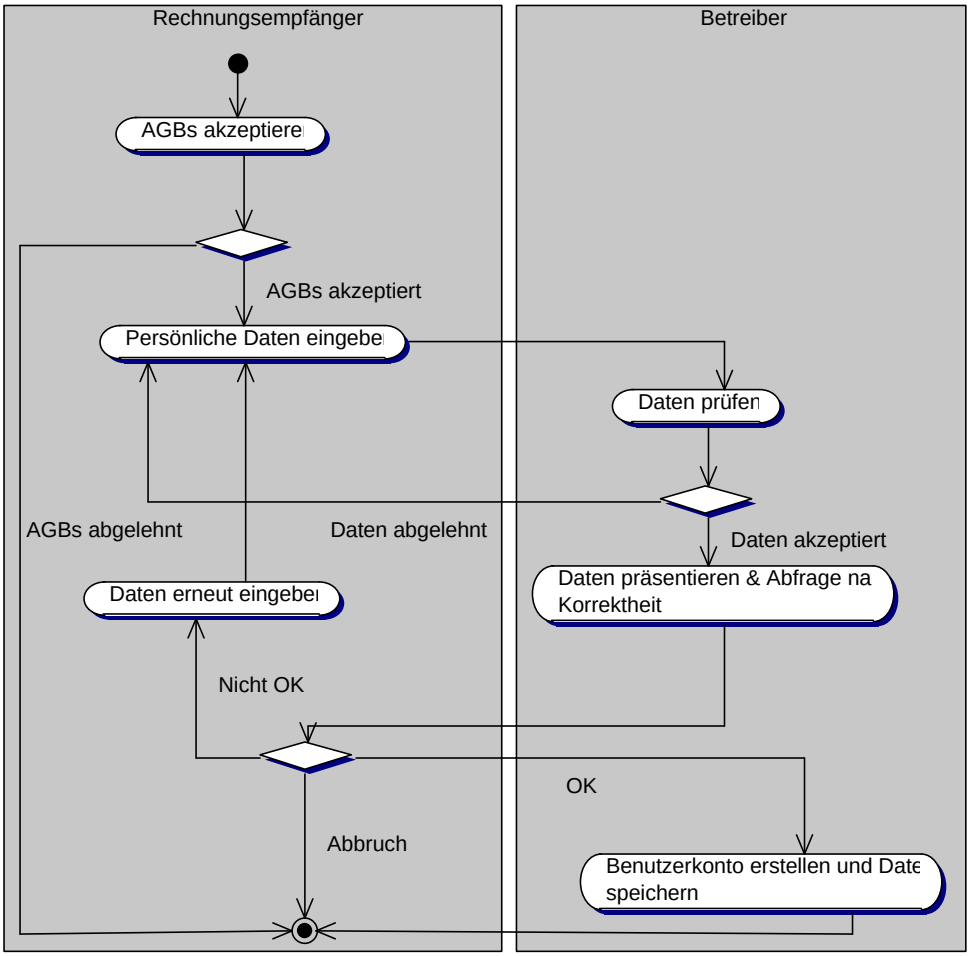

Abbildung 51: Aktivitätsdiagramm Registrierung RE

C.1.2: Anmeldung

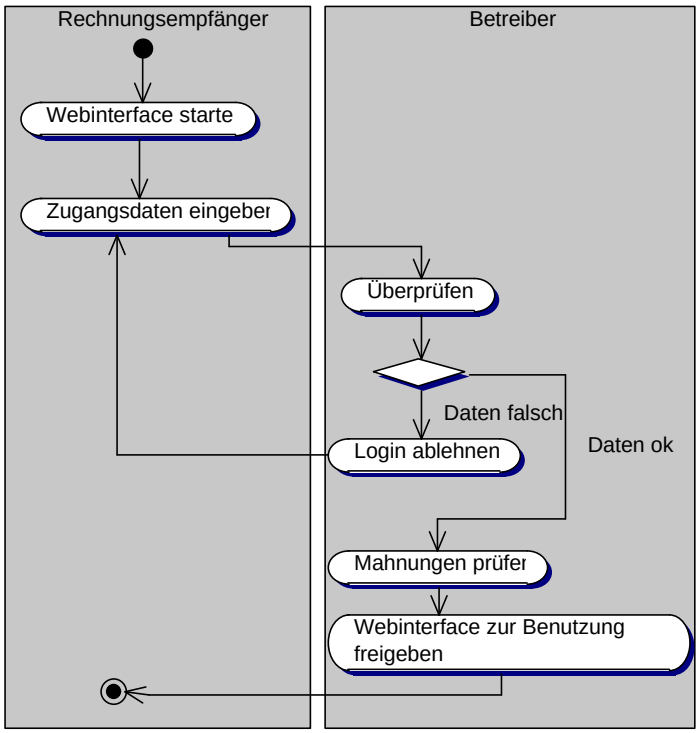

Abbildung 52: Aktivitätsdiagramm Anmeldung RE

C.1.3: Verwaltung

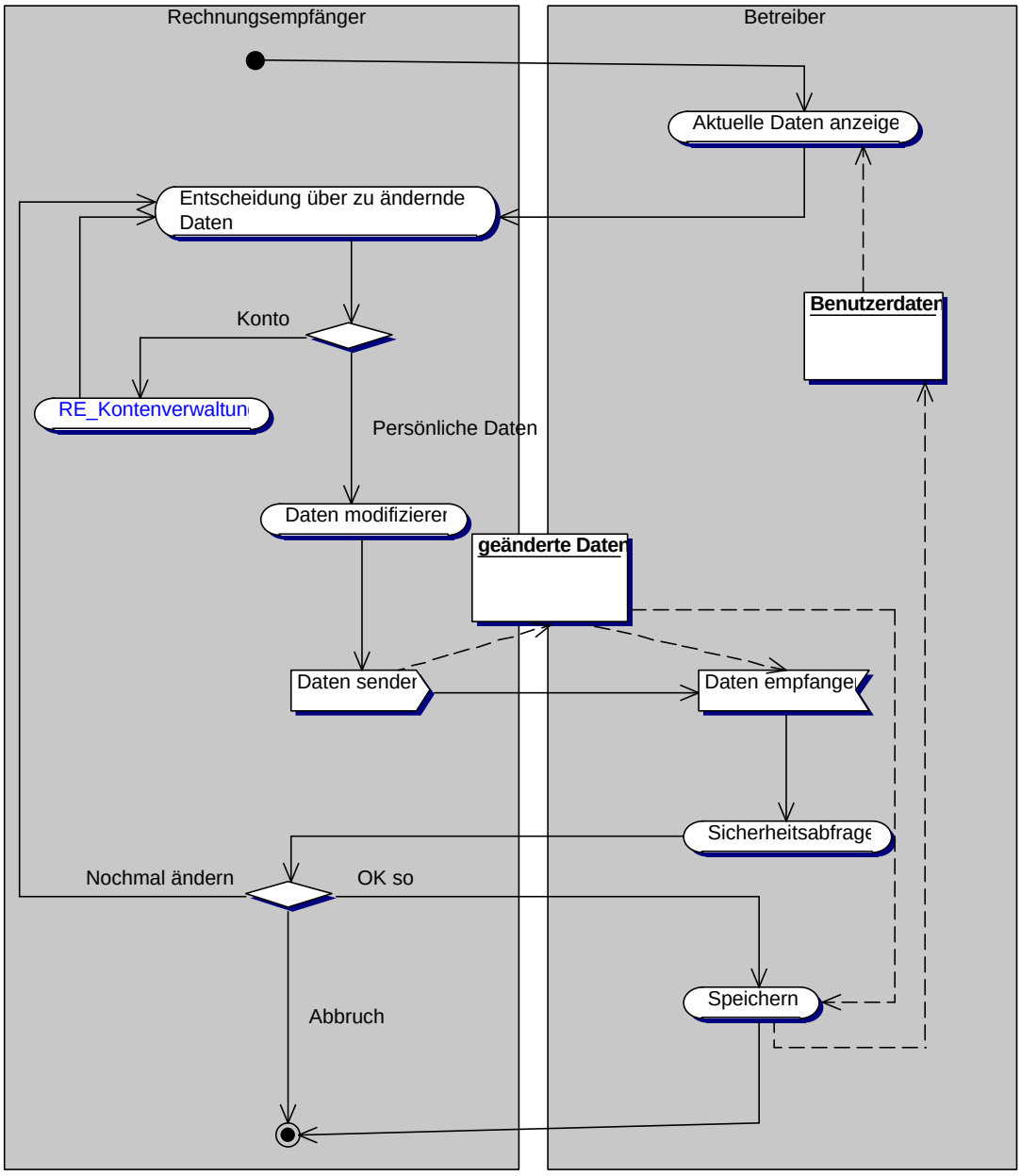

Abbildung 53: Aktivitätsdiagramm Verwaltung

C.1.4: Kontenverwaltung

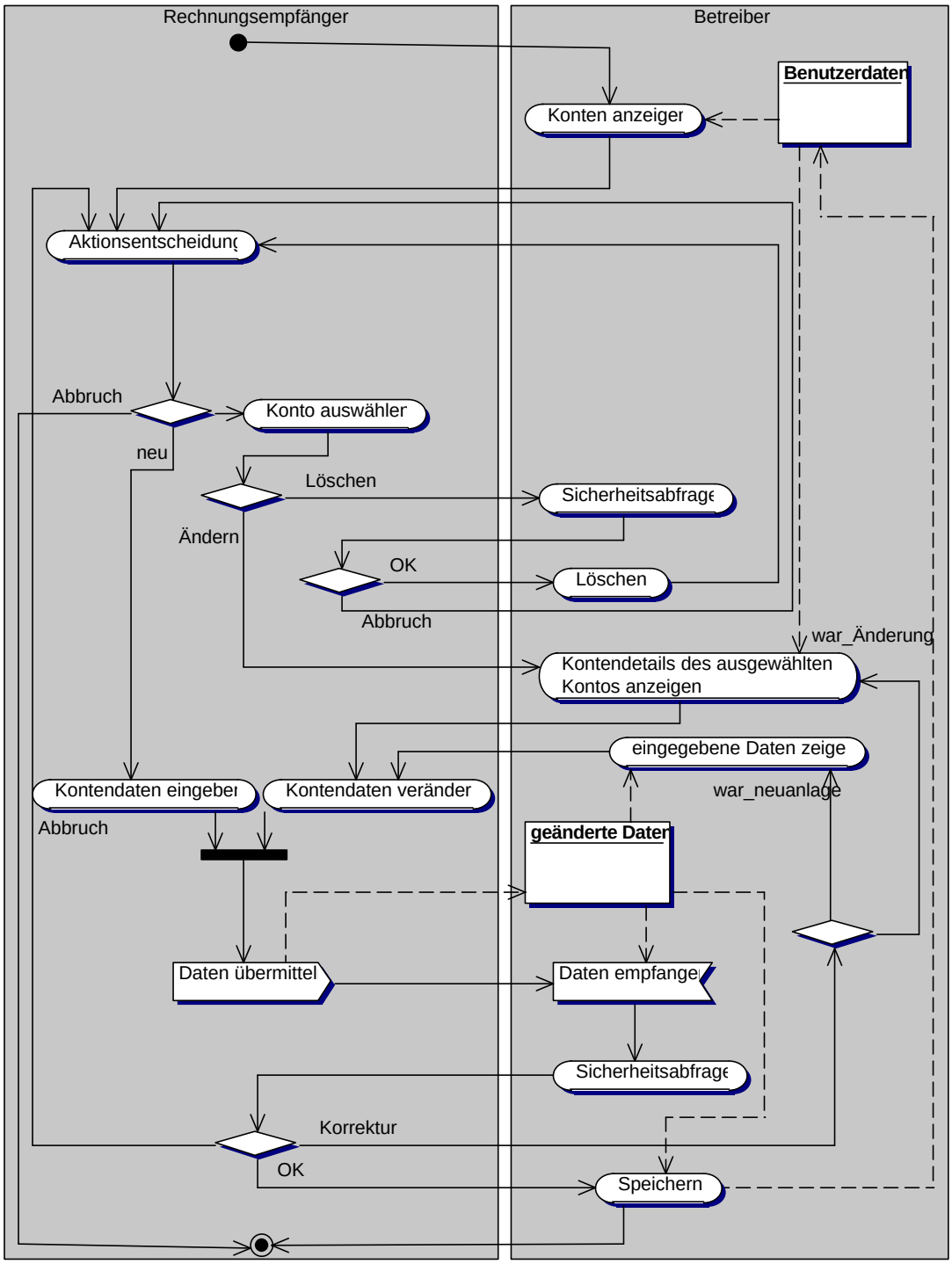

Abbildung 54: Aktivitätsdiagramm Kontenverwaltung RE

C.1.5: Rechnungspräsentation

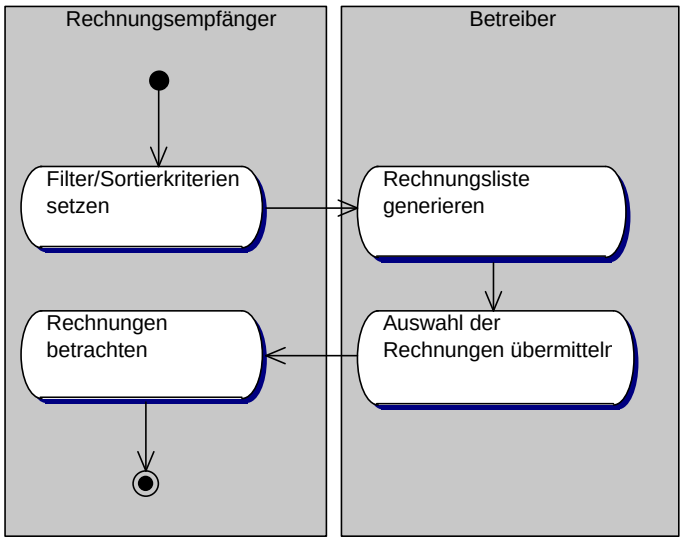

Abbildung 55: Aktivitätsdiagramm Rechnungspräsentation

C.1.6: Mahnungen

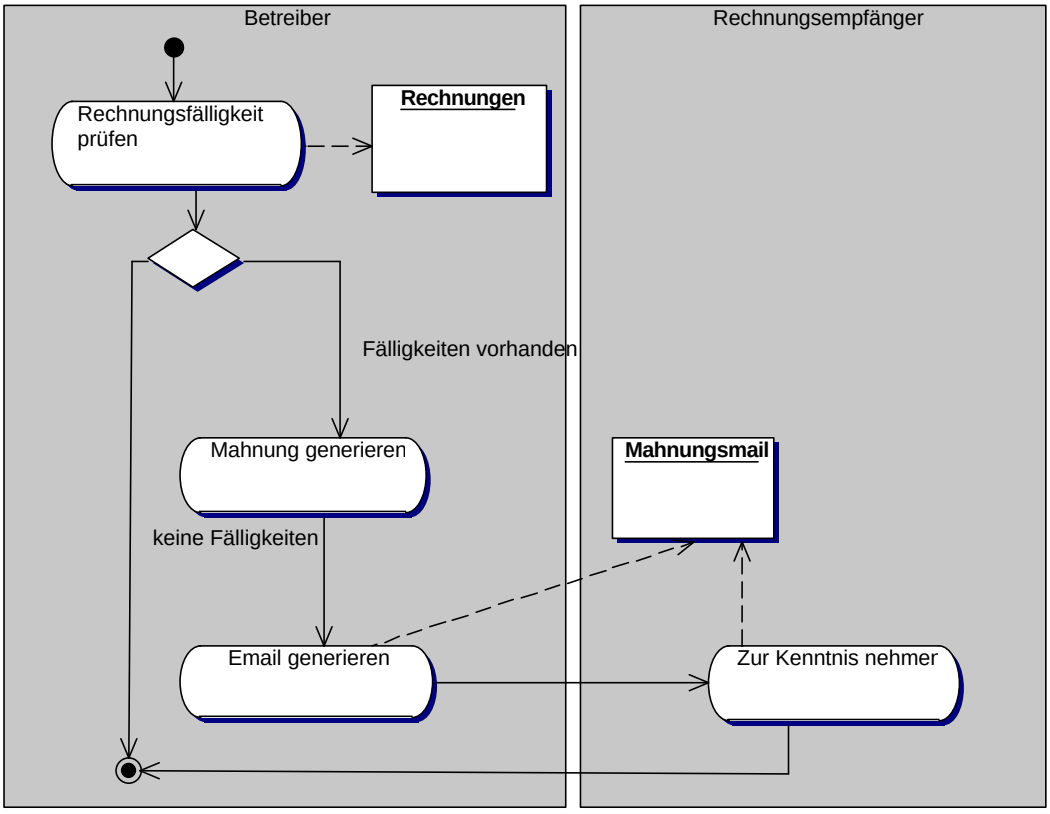

Abbildung 56: Aktivitätsdiagramm Mahnungen

C.1.7: Mahnungen prüfen

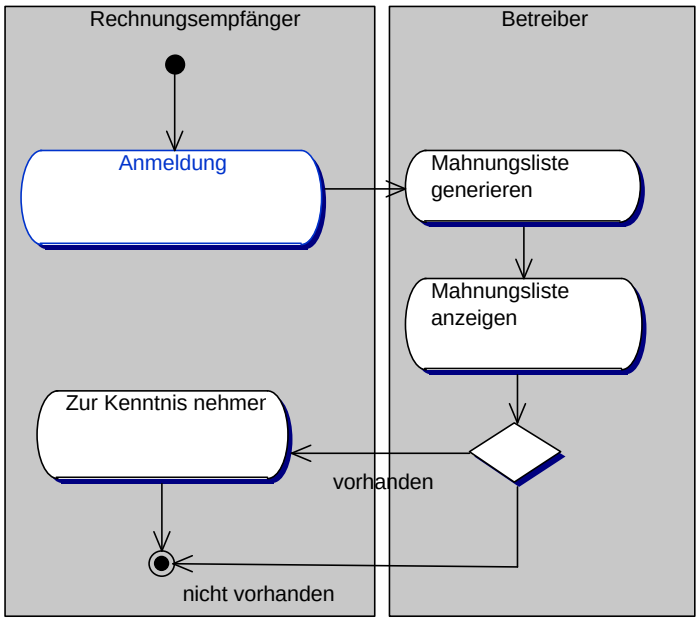

Abbildung 57: Aktivitätsdiagramm Mahnungen prüfen

C.1.8: Zahlung

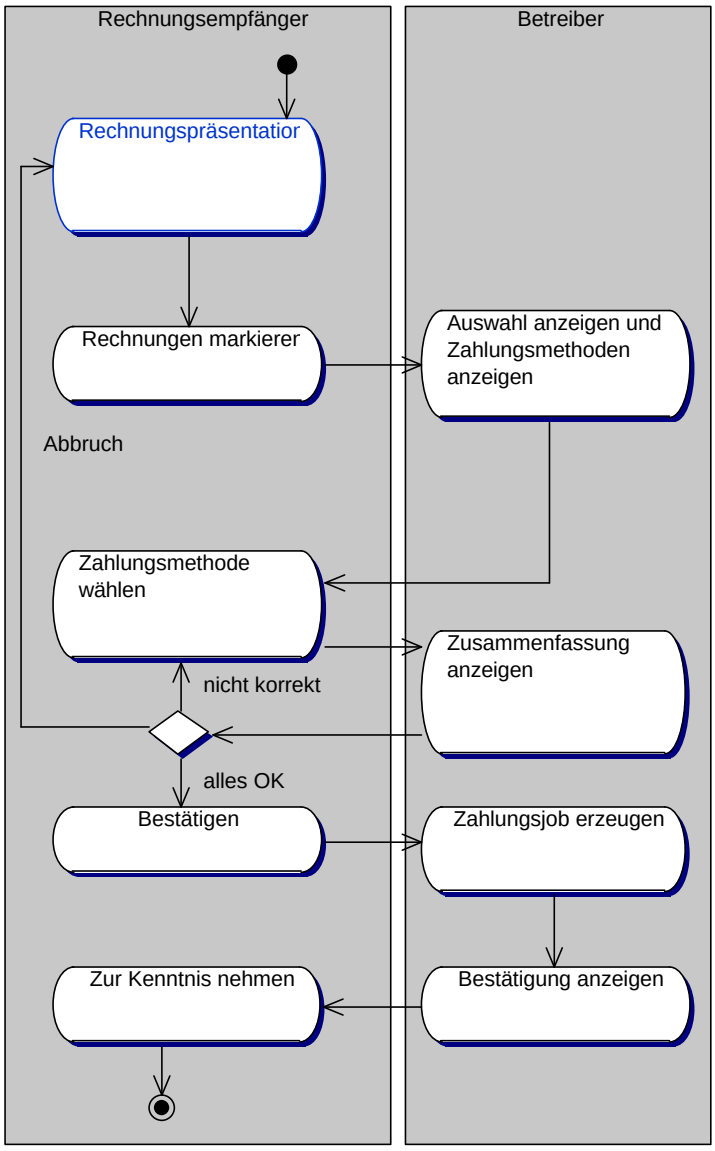

Abbildung 58: Aktivitätsdiagramm Zahlung

C.1.9: Rechnungsstatus / Zahlungsstatus

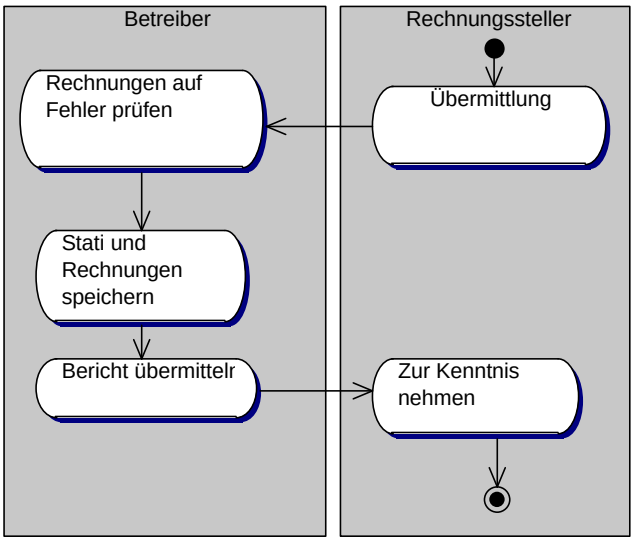

Abbildung 59: Aktivitätsdiagramm Rechnungs- / Zahlungsstatus

C.2: Rechnungssteller

C.2.1: Login

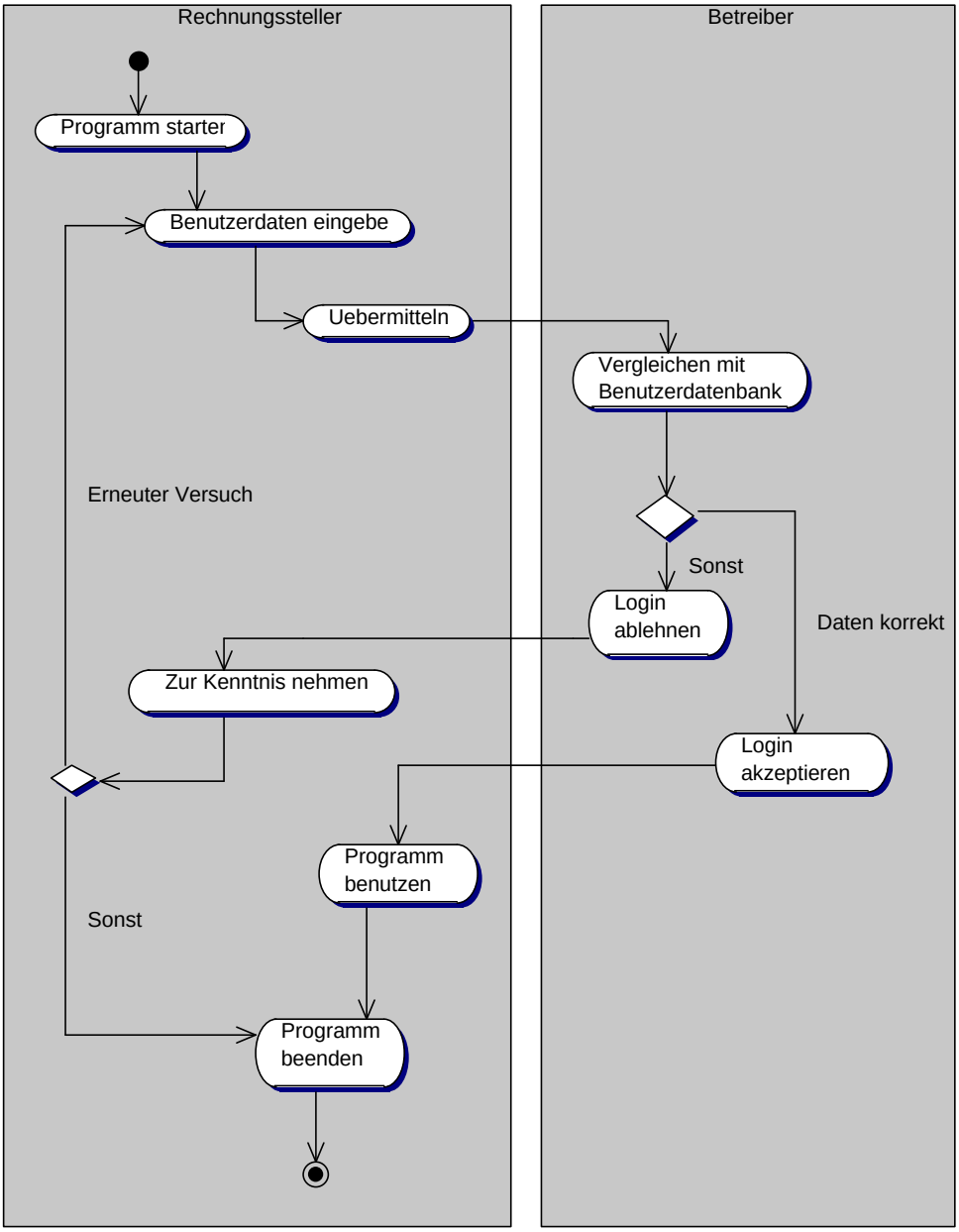

Abbildung 60: Aktivitätsdiagramm Anmelden RS

C.2.2: Registrierung

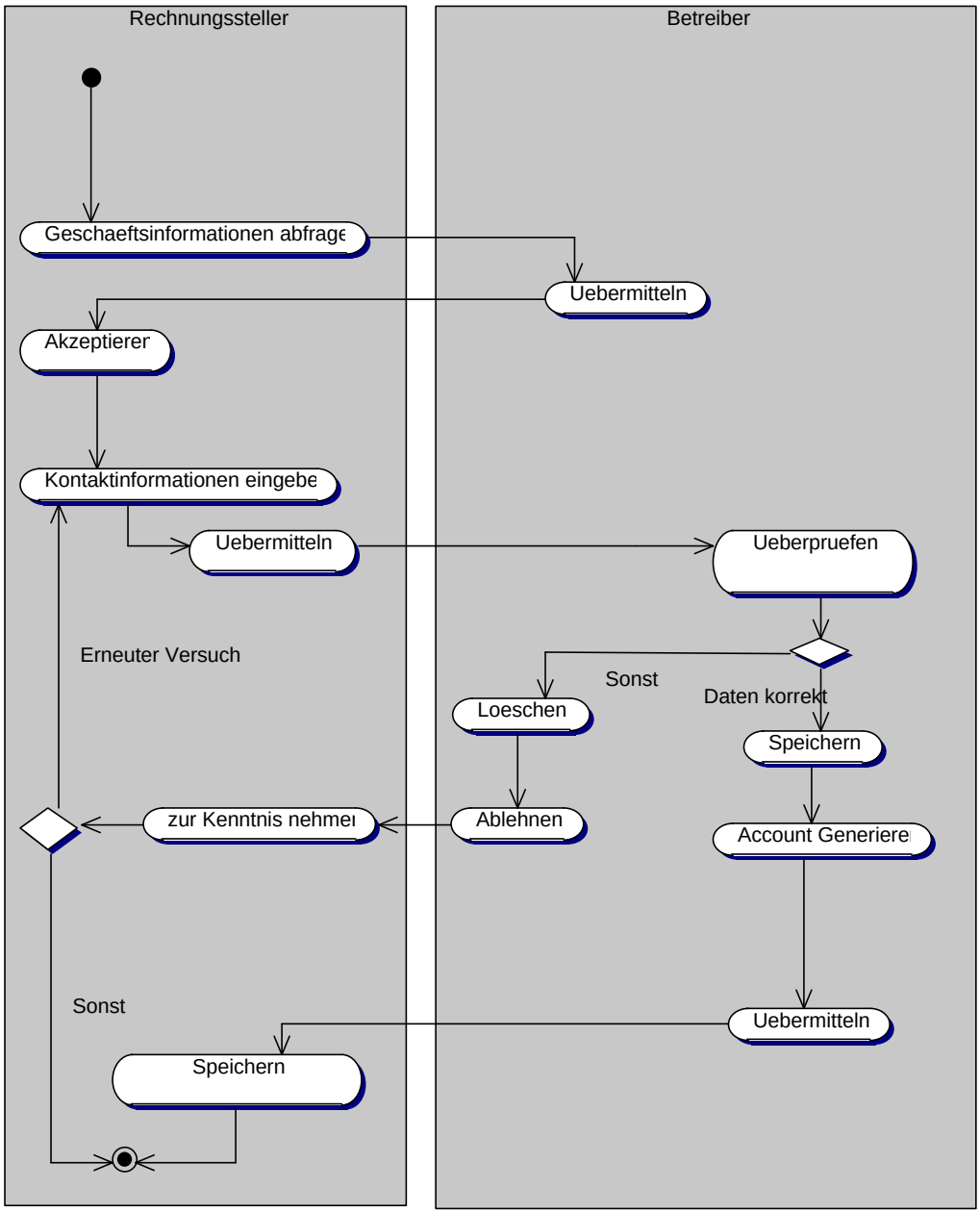

Abbildung 61: Aktivitätsdiagramm RS registrieren

C.2.3: Finanzkonto anlegen

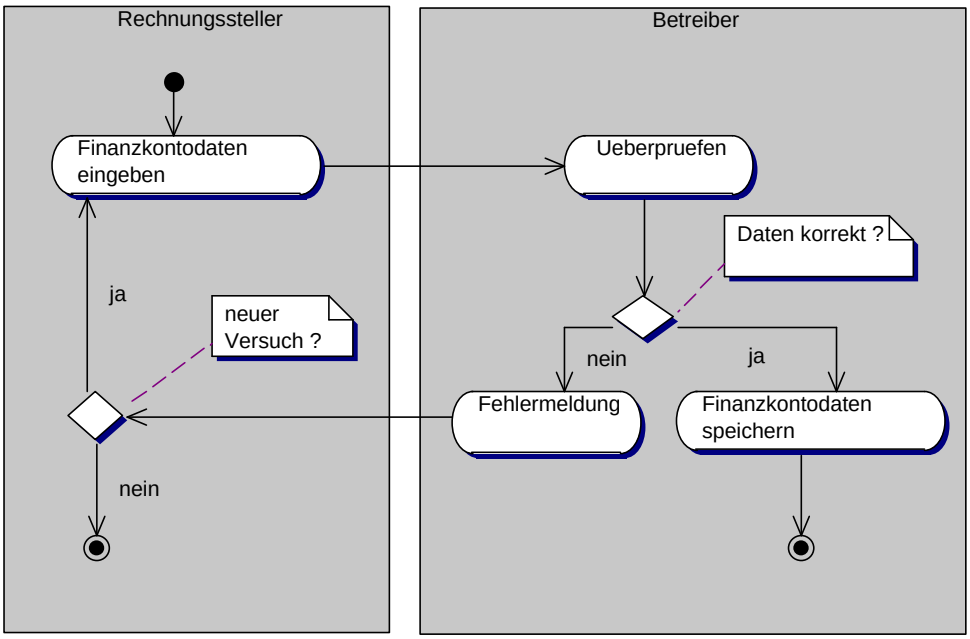

Abbildung 62: Aktivitätsdiagramm Finanzkonto anlegen

C.2.4: Finanzkonto editieren

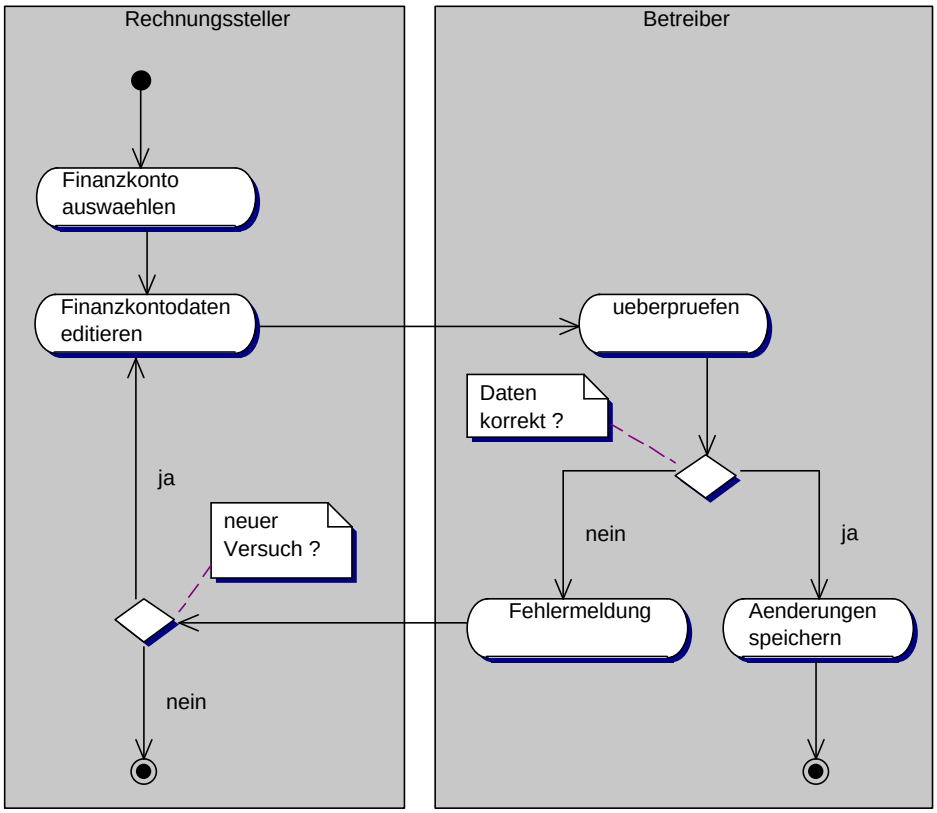

Abbildung 63: Aktivitätsdiagramm Finanzkonto editieren

C.2.5: Finanzkonto löschen

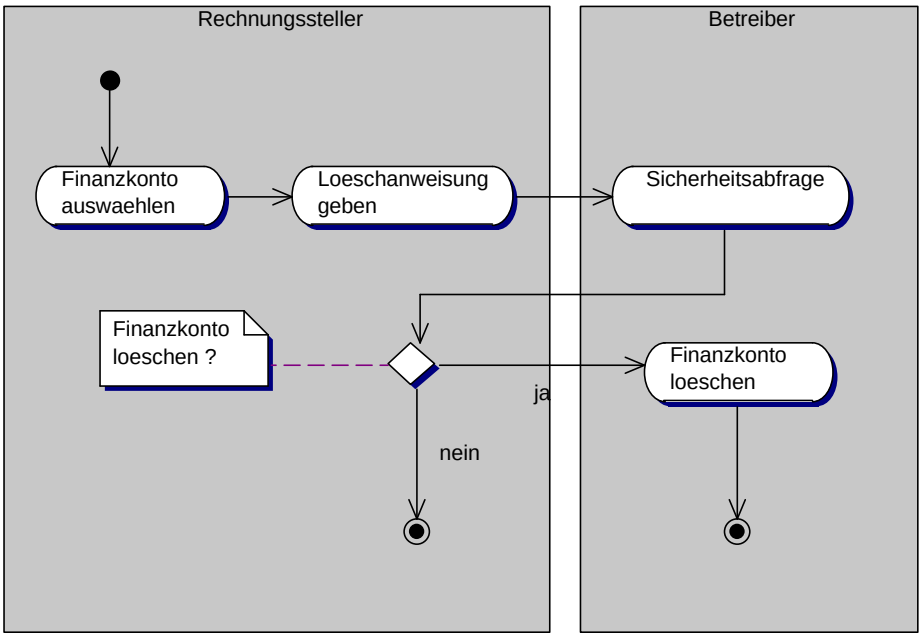

Abbildung 64: Aktivitätsdiagramm Finanzkonto löschen

C.2.6: Rechnungsstellerkonto editieren

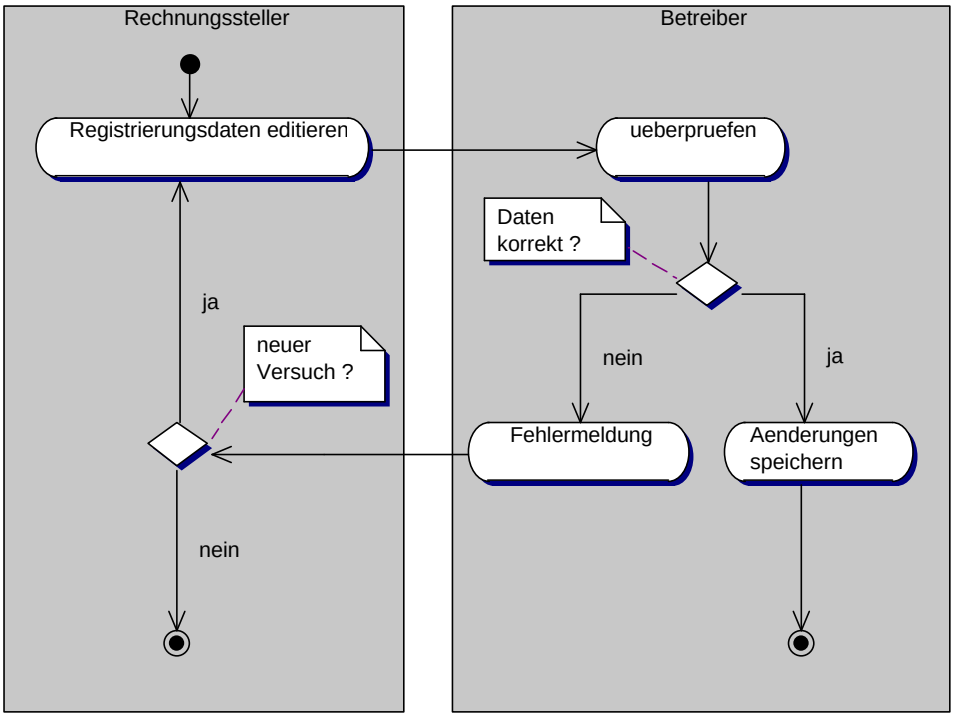

Abbildung 65: Aktivitätsdiagramm RS-Konto editieren

C.2.7: Kunden-ID überprüfen

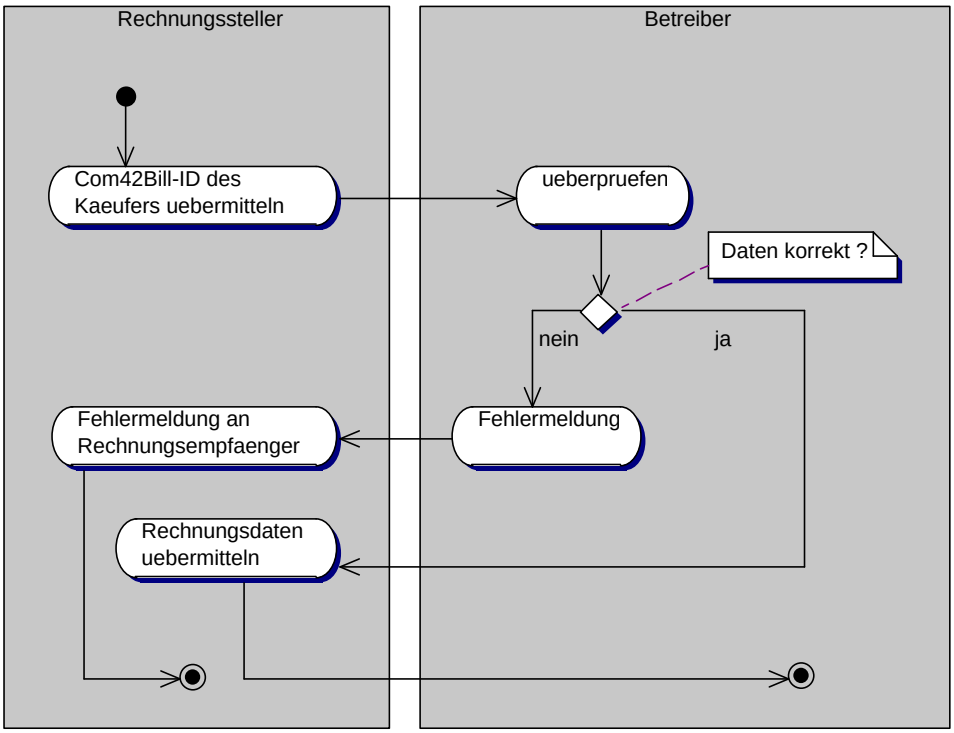

Abbildung 66: Aktivitätsdiagramm Kunden-ID prüfen

C.3Betreiber

C.3.1: Bericht abrufen

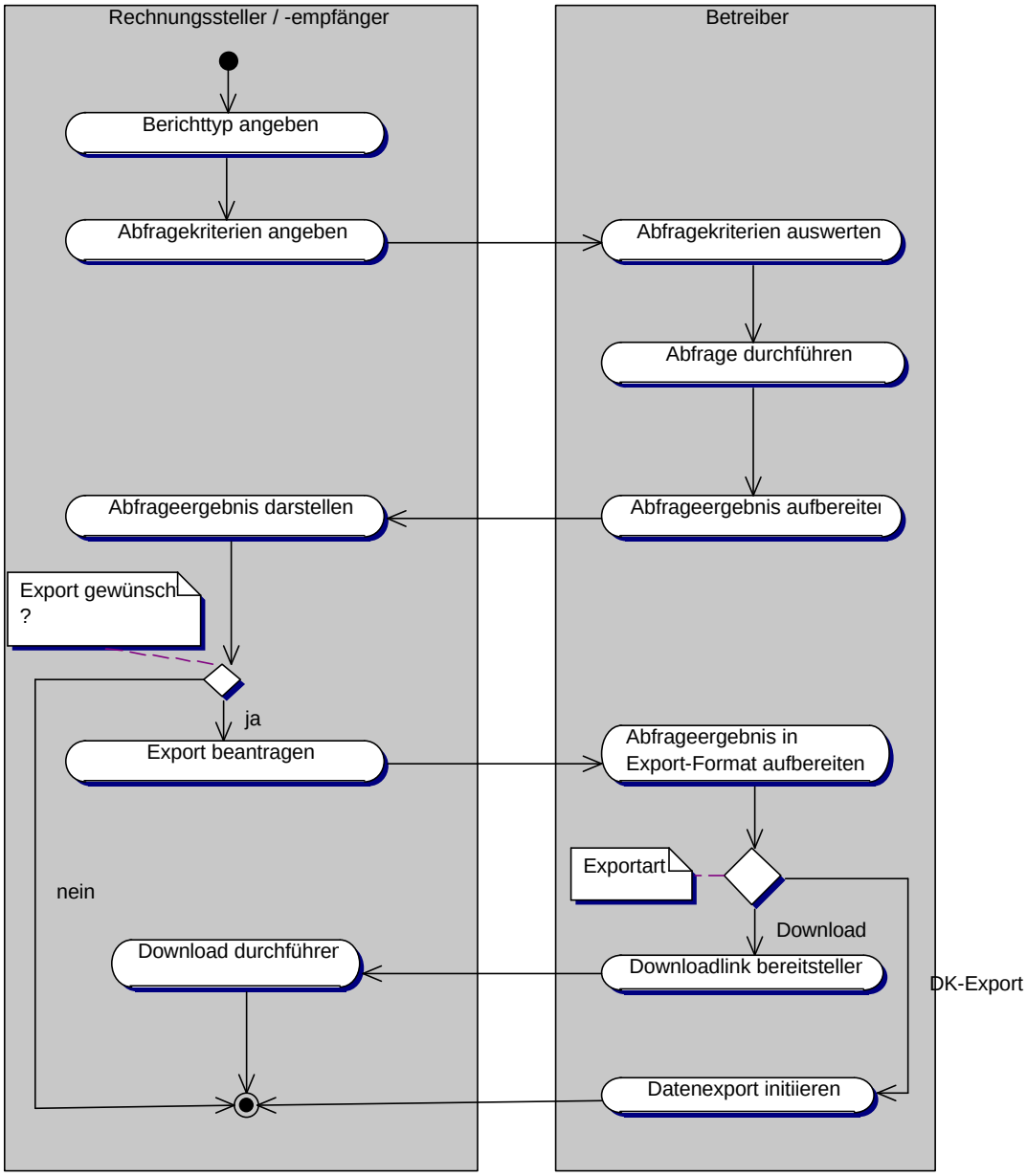

Abbildung 67: Aktivitätsdiagramm Bericht abrufen

C.3.2: Statistik abrufen

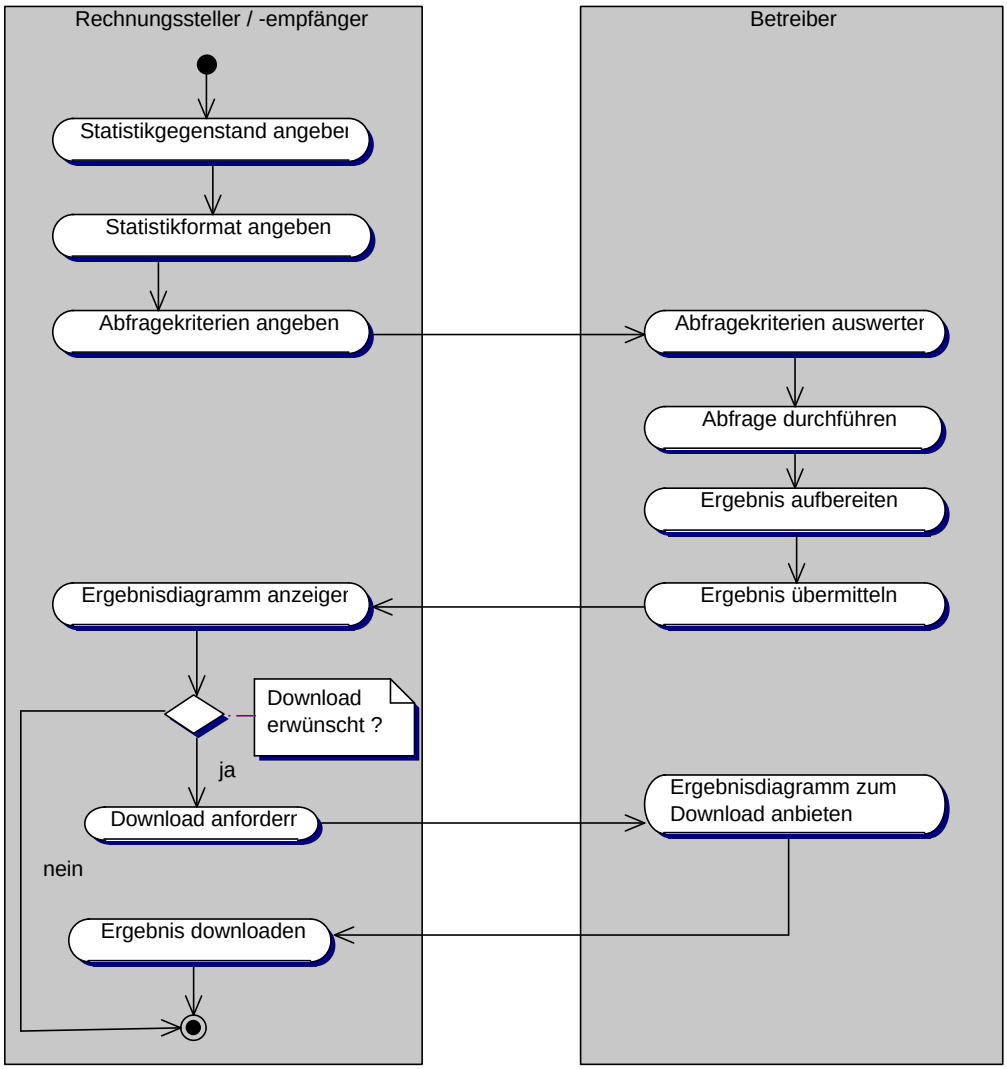

Abbildung 68: Aktivitätsdiagramm Statistik abrufen

C.3.3: Job anlegen

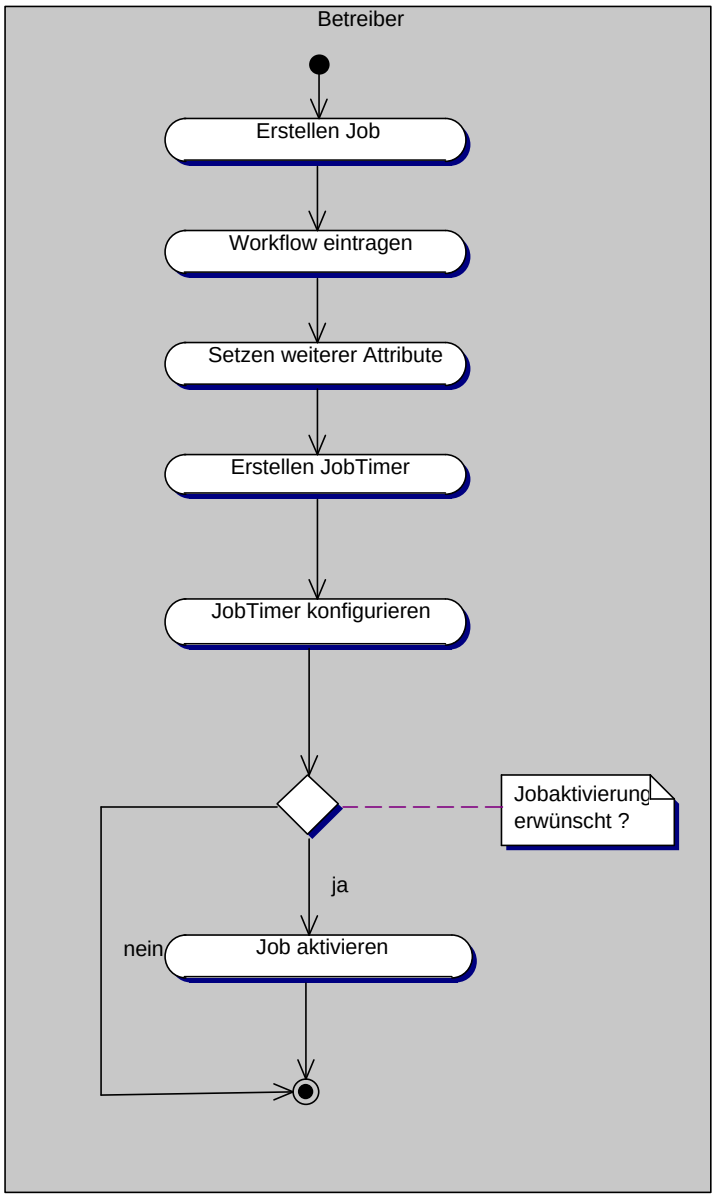

Abbildung 69: Aktivitätsdiagramm Job anlegen

C.3.4: Job ausführen

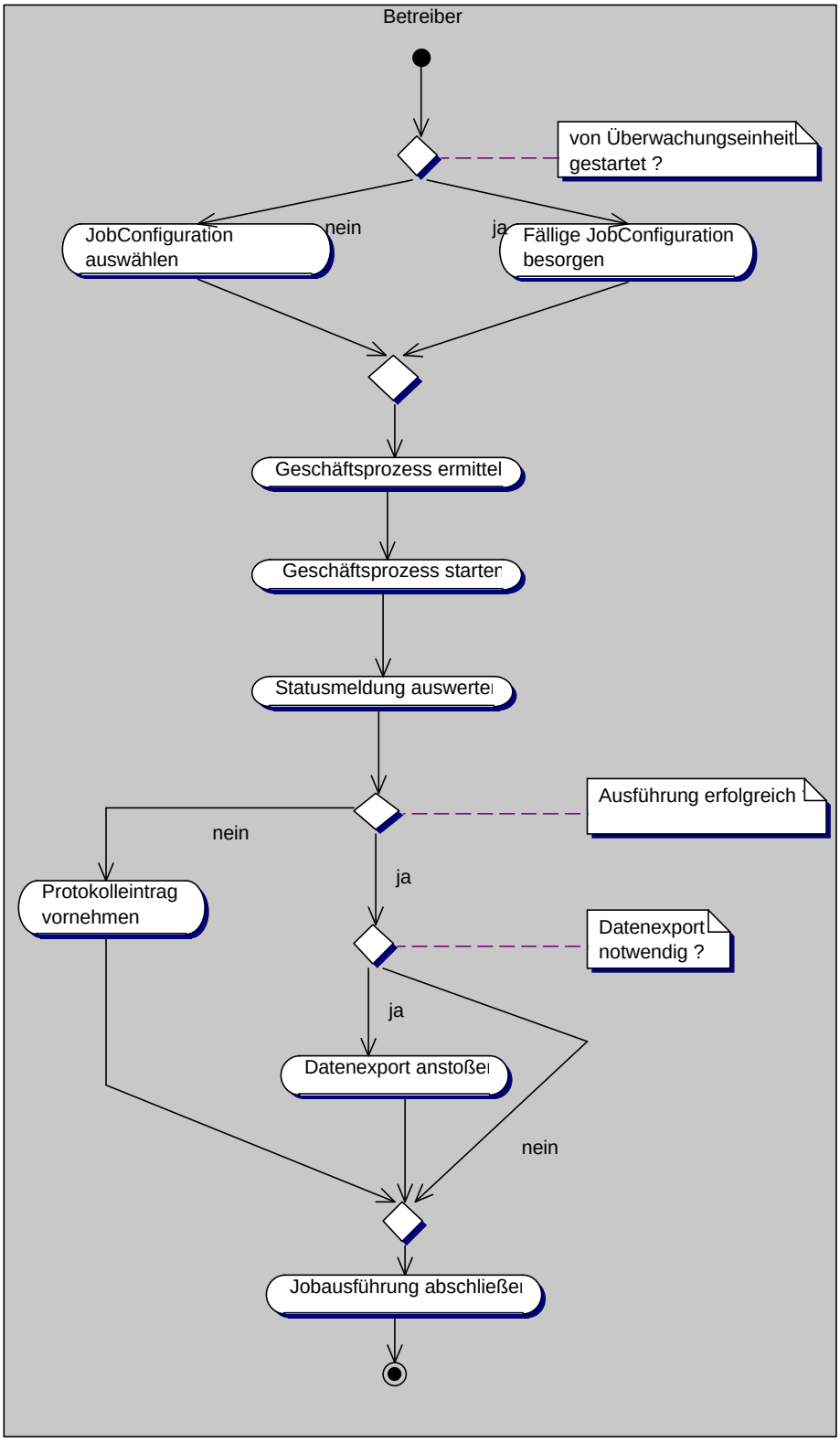

Abbildung 70: Aktivitätsdiagramm Job ausführen

C.3.5: Job löschen

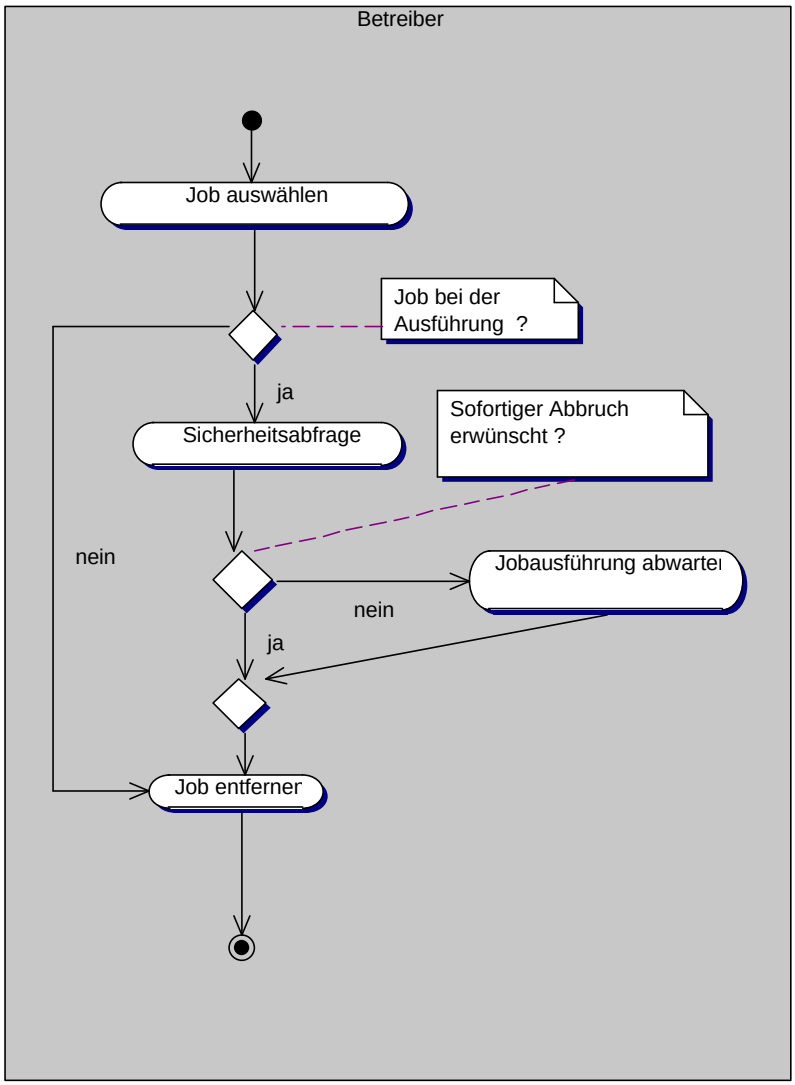

Abbildung 71: Aktivitätsdiagramm Job löschen

C.3.6: Zeitgesteuerte Transaktionen ausführen

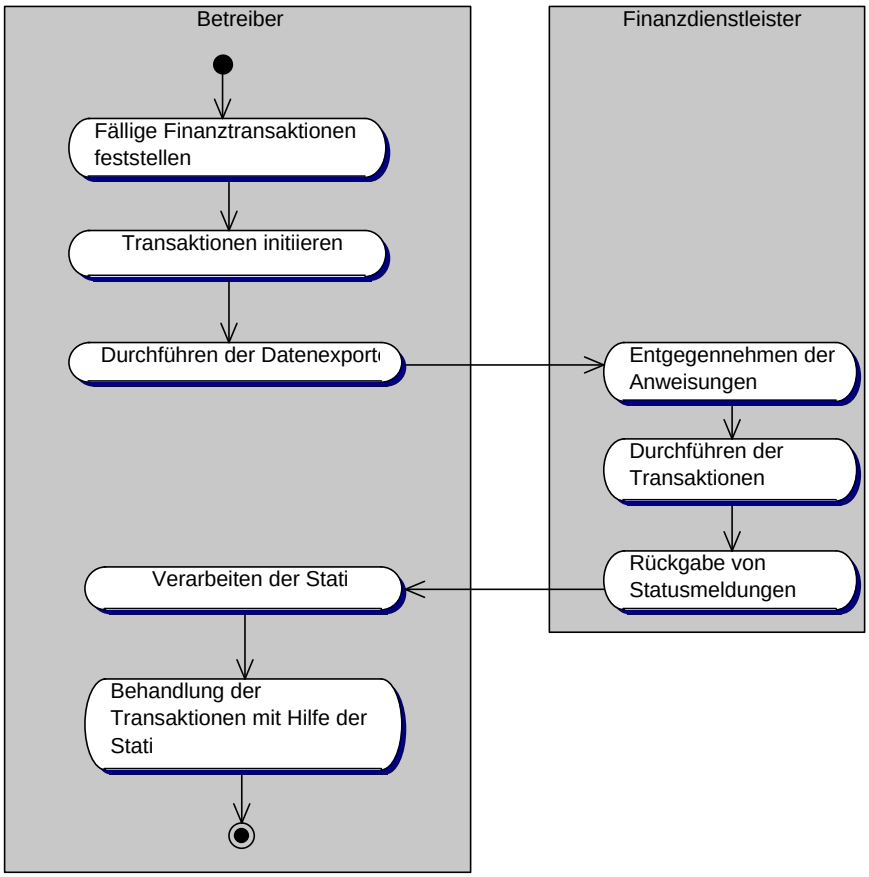

Abbildung 72: Aktivitätsdiagramm zeitgesteuerte Transaktionen

C.4: Datenkonverter

C.4.1: Login

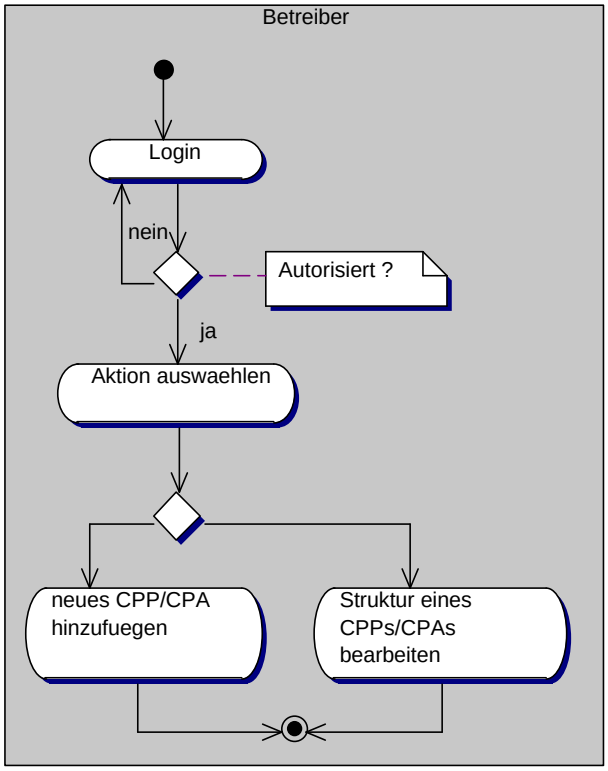

Abbildung 73: Aktivitätsdiagramm Login DK

C.4.2: Neues CPA / CPP

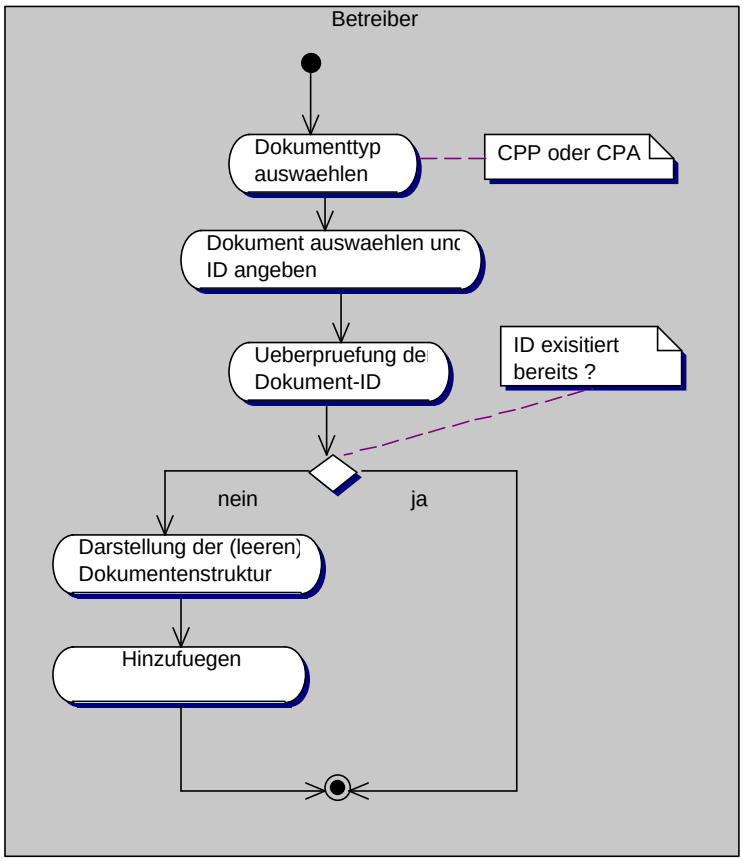

Abbildung 74: Aktivitätsdiagramm neues CPA / CPP anlegen

C.4.3: Ändern

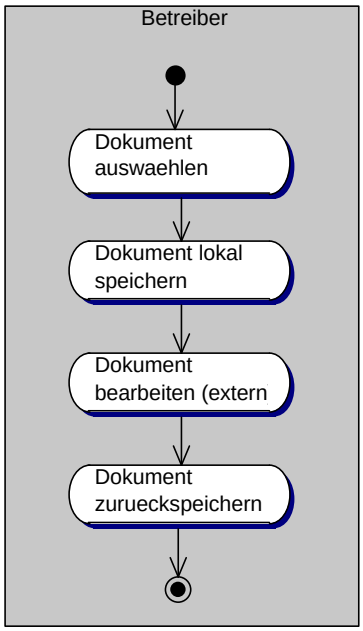

Abbildung 75: Aktivitätsdiagramm CPA / CPP ändern

C.4.4: Hinzufügen

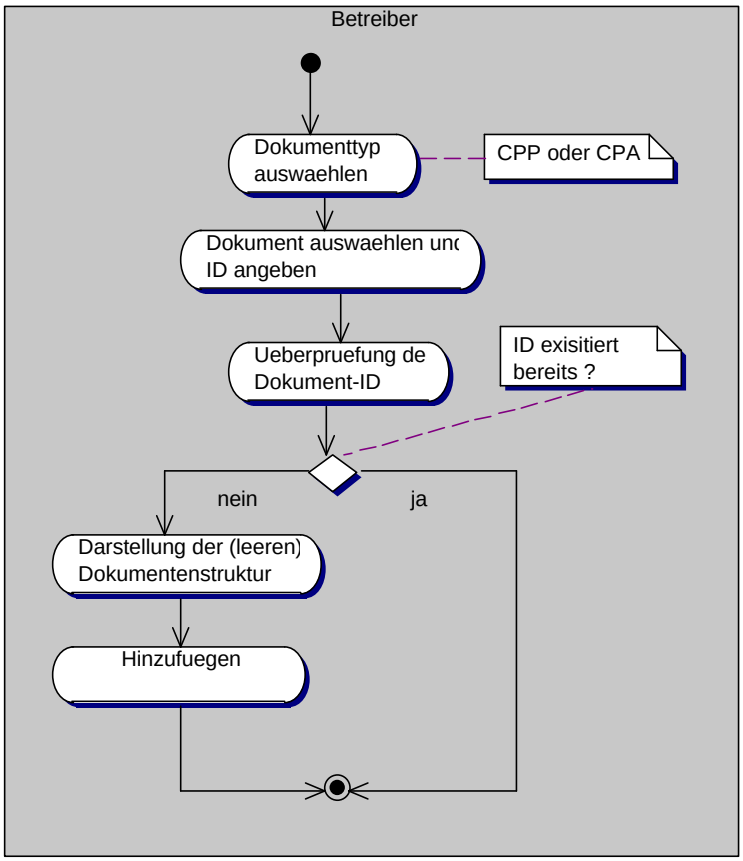

Abbildung 76: Aktivitätsdiagramm CPA / CPP hinzufügen

C.4.5: Löschen

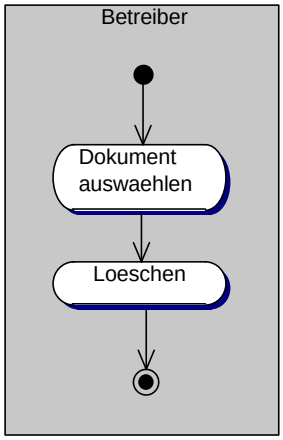

Abbildung 77: Aktivitätsdiagramm CPP / CPA löschen

C.4.6: Struktur eines CPA / CPP bearbeiten

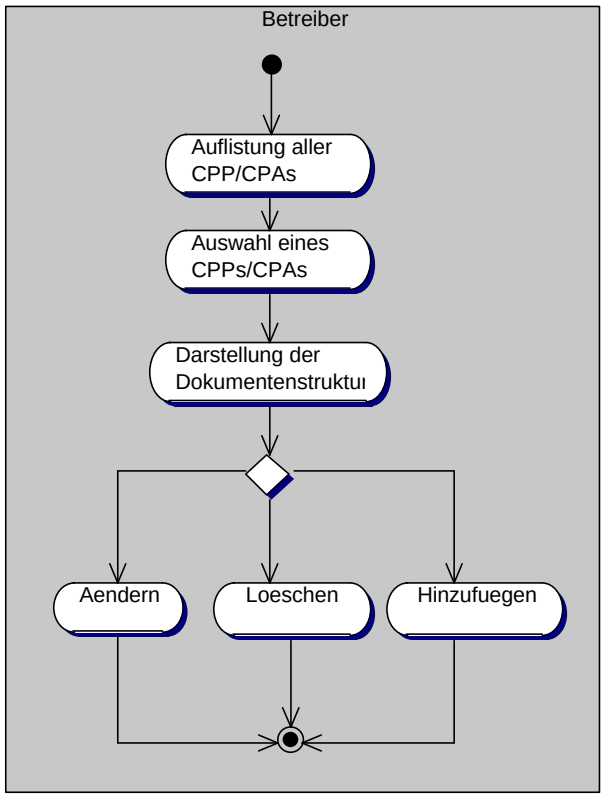

Abbildung 78: Aktivitätsdiagramm CPA / CPP bearbeiten

Anhang D: Anforderungsliste

D.1: GUI

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
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 ein zu einfaches Layout wird die Seite uninteressant.	alsc, mani, diha	28.4.02	3	muss		x		in Arbeit	
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.	alsc, mani, diha	28.4.02	2	muss		x		in Arbeit	
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.	alsc, mani, diha	28.4.02	3	muss		x		in Arbeit	
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	alsc, mani, diha	28.4.02	3	kann		x		in Arbeit	
G-005	GUI	selbst	Die Layoutgestaltung	Globale Änderungen im	alsc,	28.4.02	2	kann		x		in Arbeit	

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			beruht auf Templates.	Layout können leicht über die Templates durchgeführt werden.	mani, diha								
G-006	GUI	selbst	Dialoge sind intuitiv erschließbar sein, d.h. umittelbar verständlich sein.	Der Benutzer soll ohne Anleitung in der Lage sein, die Dialoge bedienen zu können.	alsc, mani, diha	28.4.02	2	kann		x		in Arbeit	
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.	alsc, mani, diha	28.4.02	1	kann		x		in Arbeit	
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.	alsc, mani, diha	28.4.02	3	kann		x		in Arbeit	
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 hat und ihm durch Ratschläge helfen,	Fehlerhafte Eingaben dürfen nicht toleriert werden und müssen vom Benutzer korrigiert werden, da sonst eine korrekte Abarbeitung der Eingaben nicht möglich ist.	alsc, mani, diha	28.4.02	1	muss	x			in Arbeit	

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			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.	alsc, mani, diha	28.4.02	2	kann		x		in Arbeit	
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.	alsc, mani, diha	28.4.02	4	kann		x		in Arbeit	
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.	alsc, mani, diha	28.4.02	2	muss		x		in Arbeit	
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.	alsc, mani, diha	28.4.02	3	kann		x		in Arbeit	
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.	alsc, mani, diha	28.4.02	3	muss		x		in Arbeit	

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
G-015	GUI	selbst	Die gesamte Webpräsenz muss hierarchisch strukturiert sein.	Mithilfe einer Hierarchie kann der Benutzer die Struktur der Seite leicht erfassen.	alsc, mani, diha	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.	alsc, mani, diha	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.	alsc, mani, diha	28.4.02	2	muss		x		in Arbeit	
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.	alsc, mani, diha	28.4.02	1	muss		x		in Arbeit	
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.	alsc, mani, diha	28.4.02	4	kann		x		in Arbeit	
G-020	GUI	selbst	Links beschreiben das verknüpfte Element möglichst exakt. Text-Links sind zu diesem Zweck	So ist eine Benutzung dieser Elemente ohne zusätzliche Erklärungen möglich.	alsc, mani, diha	28.4.02	2	kann		x		in Arbeit	

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			prägnant formuliert, grafische Links haben Symbole.										
G-021	GUI	selbst	Links müssen eindeutig gekennzeichnet sein: Text-Links durch farbige oder stilistische Hervorhebung (z.B. Unterstreichung), grafische Links durch optische Effekte (z.B. Highlighting).	Links ermöglichen die Navigation zwischen den Seiten und müssen deshalb klar erkennbar sein.	alsc, mani, diha	28.4.02	2	muss		x		in Arbeit	
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.	alsc, mani, diha	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.	alsc, mani, diha	28.4.02	2	kann		x		in Arbeit	
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.	alsc, mani, diha	28.4.02	2	muss	x			in Arbeit	x
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.	alsc, mani, diha	28.4.02	1	muss	x		x	entfällt	

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			anzumelden.										
G-026	GUI	selbst	Der Rechnungsempfänger wird in regelmäßigen Abständen daran erinnert sein Passwort zu ändern.	Eine regelmäßige Passwortänderung erhöht die Sicherheit.	alsc, mani, diha	28.4.02	4	kann	x			in Arbeit	
G-027	S	S-024	Die Sicherheit weist auf ein veraltetest Kennwort bei der Authentifizierung hin.	Diese Funktion löst G-026 aus.	alsc, mani, diha	11.6.02	3	kann	x		x	in Arbeit	
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.	alsc, mani, diha	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	alsc, mani, diha	11.6.02	2	muss	x		x	in Arbeit	
G-030	GUI	selbst	Der Rechnungsempfänger kann eine Rechnungsübersicht anfordern und einsehen.	Eine Rechnungsübersicht ist für die Überschaubarkeit der Rechnungen unabdingbar.	alsc, mani, diha	28.4.02	1	muss	x		x	in Arbeit	
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.	alsc, mani, diha	28.4.02	1	muss		x	x	in Arbeit	
G-032	GUI	selbst	In der Rechnungsübersicht können die Rechnungen nach Kriterien sortiert und gefilter werden.	Diese Maßnahme erleichtert die Übersicht über die Rechnungen.	alsc, mani, diha	28.4.02	3	kann	x		x	in Arbeit	

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
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	alsc, mani, diha	11.6.02	1	muss	x		x	in Arbeit	
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.	alsc, mani, diha	28.4.02	2	muss	x			in Arbeit	
G-035	GUI	selbst	Rechnungen aus der Rechnungsübersicht können für den Zahlungsvorgang markiert werden.	Auf diese Weise können mehrere Rechnungen auf einmal bezahlt werden.	alsc, mani, diha	28.4.02	1	muss	x			in Arbeit	
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.	alsc, mani, diha	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	alsc, mani, diha	11.6.02	1	muss	x		x	in Arbeit	
G-038	GUI	selbst	Nach dem Abschluss der	Der Zahlungsvorgang ist	alsc,	28.4.02	1	muss		x	x	in Arbeit	

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			Bezahlung und der Bestätigung, dass der Vorgang bei der Business Logic ist, wird eine Bestätigung des Zahlungsvorgangs präsentiert.	erst abgeschlossen, wenn er bei der Business Logic eingegangen ist. Danach erhält der Benutzer die notwendige Bestätigung.	mani, diha								
G-039	BL	nicht abgedeckt	Die BL gibt nach dem Eingang eines Bezahlvorgangs eine Bestätigung an die GUI.	Ermöglicht G-038	alsc, mani, diha	11.6.02	1	muss	x		x	in Arbeit	
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.	alsc, mani, diha	28.4.02	3	kann	x		x	in Arbeit	
G-041	DB	DB-009	Die Hilfen werden in der DB abgespeichert.		alsc, mani, diha	11.6.02	1	muss	x		x	in Arbeit	
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.	alsc, mani, diha	11.6.02	1	muss	x		x	in Arbeit	
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!?)	alsc, mani, diha	28.4.02	2	kann	x		x	entfällt	
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.	alsc, mani, diha	28.4.02	1	muss	x		x	in Arbeit	

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
G-045	GUI	selbst	Der Rechnungssteller kann eine Übersicht seiner Rechnungen einsehen.	Dadurch kann er seine Rechnungen bezahlt/unbezahlt übersehen.	alsc, mani, diha	28.4.02	1	muss	x			in Arbeit	
G-046	GUI	selbst	Der Rechnungesteller kann die Rechnungen in der Rechungsübersicht nach Kriterien sortieren und filtern	Eine Erleichterung bei der Übersicht der Rechnungen.	alsc, mani, diha	28.4.02	2	kann	x			in Arbeit	
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.	alsc, mani, diha	11.6.02	1	muss	x		x	entfällt	
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.	alsc, mani, diha	28.4.02	4	kann	x		x	entfällt	
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.	alsc, mani, diha	30.5.02	1	muss	x			in Arbeit	x
G-050	GUI	selbst	Das Requesthandler-Servlet kann die Eingaben bereits auf syntaktische Fehler kontrollieren.	Korrekte Eingaben sind für das System notwendig.	alsc, mani, diha	30.5.02	3	kann	x			in Arbeit	x
G-051	GUI	selbst	Der Request-Dispatcher unterscheidet zwischen Requests, die an die Sicherheit bzw. an die	Requests, die der Authentifizierung dienen werden nicht von der BL bedient und müssen an	alsc, mani, diha	30.5.02	1	muss	x			in Arbeit	

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			Business Logic weitergeleitet werden müssen.	die Sicherheit weitergeleitet werden.									
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.	alsc, mani, diha	30.5.02	3	muss	x		x	in Arbeit	
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.	alsc, mani, diha	30.5.02	2	muss	x			in Arbeit	x
G-054	GUI	selbst	Gültige Anfragen leitet der Request-Dispatcher an den Workflow-Adapter weiter.	Der Workflow-Adapter bietet die Schnittstelle zur Business-Logic.	alsc, mani, diha	30.5.02	1	muss	x			in Arbeit	x
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.	alsc, mani, diha	30.5.02	2	muss	x		x	in Arbeit	x
G-056	GUI	selbst	Der History-Controller	s. G-053	alsc,	30.5.02	2	muss	x			in Arbeit	x

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			kontrolliert ob eine gewünschte Aktion auf die vorhergehende(n) folgen darf.		mani, diha								
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.	alsc, mani, diha	30.5.02	1	muss	x			in Arbeit	x
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.	alsc, mani, diha	30.5.02	1	muss	x		x	in Arbeit	x
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.		alsc, mani, diha	11.6.02	1	muss	x		x	in Arbeit	
G-060	BL	BI-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	Statistiken bieten eine schnelle Möglichkeit sich einen Überblick über z.B. seine monatlichen ausgaben zu schaffen.	alsc, mani, diha	5.6.02	4	kann	x		x	in Arbeit	

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			verfügbar.										
G-061	GUI	selbst	Die GUI bietet dem Rechnungsempfänger Statistiken über seine Rechnungen an.	Eine mögliche Zusatzfunktion. (s. G-059)	alsc, mani, diha	11.6.02	4	kann	x			in Arbeit	
G-062	DB	DB-007, DB-009	Die internen Daten der Komponente GUI werden soweit wie möglich in der Datenbank gespeichert. (s. G-041)	Die Speicherung in der Datenbank ermöglicht eine Zugriffssicherheit.	alsc, mani, diha	5.6.02	1	muss			x	in Arbeit	x
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.	alsc, mani, diha	5.6.02	3	muss	x		x	entfällt	
G-064	GUI		Der DK bietet der GUI Funktionen für das Abfragen darzustellender Informationen der Registry und des Repositiory.	Ermöglicht G-062.	alsc, mani, diha	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)	alsc, mani, diha	11.6.02	1	muss		x	x	entfällt	
G-066	GUI		Es werden jederzeit nur notwendige Daten übertragen.	Zur Sicherheit werden keine unnötigen Informationen übertragen. (s. S-003)	alsc, mani, diha	11.6.02	1	muss		x	x	entfällt	

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
G-067	S		Die Übertragung zwischen dem jeweiligen Benutzer und der GUI verläuft spätestens ab der Authentifizierung verschlüsselt (SSL).	Sensible Daten müssen auch bei der Übertragung geschützt werden. (s. S-004)	alsc, mani, diha	11.6.02	1	muss		x	x	entfällt	
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.	alsc, mani, diha	11.6.02	3	kann	x		x	in Arbeit	
G-069	GUI		Es werden keine Daten auf dem Klienten gehalten. Ein eventueller Datenexport ist davon ausgenommen.	Daten könnten in falsche Hände geraten.	alsc, mani, diha	24.6.02	1	muss	x			entfällt	
G-070	GUI		Daten können nur von den hierzu berechtigten Benutzern eingesehen werden.	Daten könnten in falsche Hände geraten.	alsc, mani, diha	24.6.02	1	muss	x			entfällt	
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.	alsc, mani, diha	24.6.02	1	muss		x		entfällt	
G-072	S		Der Betreiber hat die Möglichkeit sich mittels Benutzername und Passwort am System	Ein Authentifizierung ist für den Zugriff auf geschützte Daten unbedingt nötig.	alsc, mani, diha	28.4.02	1	muss	x		x	entfällt	

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			anzumelden.										
G-073	GUI	selbst	Der Betreiber kann eine Liste der Rechnungsempfänger einsehen	Zur Kontrolle der Daten im System	alsc, mani, diha	28.6.02	2	muss	x			in Arbeit	
G-074	GUI	selbst	Der Betreiber kann Details der Rechnungsempfänger kontrollieren	Zur Kontrolle der Daten im System	alsc, mani, diha	28.6.02	2	muss	x		x	in Arbeit	
G-075	GUI	selbst	Der Betreiber kann Rechnungsempfängerkonten löschen, editieren und anlegen.	Zur Wartung der Daten im System.	alsc, mani, diha	28.6.02	3	muss	x		x	in Arbeit	
G-076	GUI	selbst	Der Betreiber kann eine Liste der Rechnungssteller einsehen	Zur Kontrolle der Daten im System	alsc, mani, diha	28.6.02	2	muss	x			in Arbeit	
G-077	GUI	selbst	Der Betreiber kann Details der Rechnungssteller kontrollieren	Zur Kontrolle der Daten im System	alsc, mani, diha	28.6.02	2	muss	x		x	in Arbeit	
G-078	GUI	selbst	Der Betreiber kann Rechnungsstellerkonten löschen, editieren und anlegen.	Zur Wartung der Daten im System.	alsc, mani, diha	28.6.02	3	muss	x		x	in Arbeit	
G-079	GUI	selbst	Der Betreiber kann eine Liste der Finanzdienstleister einsehen	Zur Kontrolle der Daten im System	alsc, mani, diha	28.6.02	4	muss	x			in Arbeit	
G-080	GUI	selbst	Der Betreiber kann Details der Finanzdienstleister kontrollieren	Zur Kontrolle der Daten im System	alsc, mani, diha	28.6.02	4	muss	x		x	in Arbeit	
G-081	GUI	selbst	Der Betreiber kann Finanzdiensteleisterkonten löschen, editieren und	Zur Wartung der Daten im System.	alsc, mani, diha	28.6.02	4	muss	x		x	in Arbeit	

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			anlegen.										
G-082	GUI	selbst	Der Betreiber kann eine Liste der Rechnungen einsehen	Zur Kontrolle der Daten im System	alsc, mani, diha	28.6.02	2	muss	x			in Arbeit	
G-083	GUI	selbst	Der Betreiber kann Details der Rechnungen kontrollieren	Zur Kontrolle der Daten im System	alsc, mani, diha	28.6.02	2	muss	x		x	in Arbeit	
G-084	GUI	selbst	Der Betreiber kann Rechungen löschen.	Zur Wartung der Daten im System.	alsc, mani, diha	28.6.02	3	muss	x		x	in Arbeit	
G-085	GUI	selbst	Einhalten der Sicherheitsrichtlinien	siehe S-021	alsc, mani, diha	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.	alsc, mani, diha	5.7.02	2	muss	x		x	in Arbeit	
G-087	GUI	selbst	Der Betreiber kann Sicherheitsgruppen anlegen, löschen und editieren.	Bearbeiten der Sicherheitsgruppen.	alsc, mani, diha	5.7.02	2	muss	x		x	in Arbeit	
G-088	GUI	selbst	Der Betreiber kann über die GUI Systemjobs anlegen und löschen.	?	alsc, mani, diha	5.7.02	2	muss	x		x	in Arbeit	
G-089	GUI	selbst	Der Betreiber kann über die GUI Systemjobs zur Ausführung anstossen.	?	alsc, mani, diha	5.7.02	2	muss	x		x	in Arbeit	
G-090	GUI	selbst	Der Betreiber stellt ein Interface für das	Zur Wartung des Datenkonverters.	alsc, mani,	5.7.02	2	muss	x		x	in Arbeit	

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			hinzufügen und löschen von CPP bzw. CPAs		diha								

D.2: Sicherheit

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
S-001	GUI		Es werden keine Daten auf dem Klienten gehalten. Ein eventueller Datenexport ist davon ausgenommen.	Daten könnten in falsche Hände geraten.	basc, demu	30.5.02	1	muss		x		entfällt	
S-002	GUI, BL		Die Daten werden in verarbeiteter, d.h. endgültiger Form, übermittelt.	Klienten müssen keine Verarbeitungslogik besitzen.	basc, demu	30.5.02	1	muss		x		entfällt	
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.	basc, demu	30.5.02	1	muss		x		entfällt	
S-004	GUI, DK		Ist eine Datenübertragung außerhalb des Betreibers nötig, wird nur über sichere Verbindungen kommuniziert.	Datensicherheit.	basc, demu	30.5.02	1	muss		x		entfällt	
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.	basc, demu	30.5.02	1	muss		x		entfällt	
S-006	GUI, DK, BL		Daten werden	Datensicherheit und	basc,	30.5.02	1	muss		x		entfällt	

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			ausschließlich in der Datenbank gespeichert.	Datenintegrität.	demu								
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.	basc, demu	30.5.02	1	muss		x		entfällt	
S-008	S		Benutzer des Systems müssen sich über die Komponente Sicherheit anmelden.	Anmeldung.	basc, demu	30.5.02	1	muss		x	x	entfällt	
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.	basc, demu	30.5.02	1	muss	x		x	in Arbeit	
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.	basc, demu	30.5.02	1	muss	x		x	in Arbeit	
S-011	S	selbst	Nachdem überprüft wurde ob die gewünschte Aktion zulässig ist, erteilt die	Authorisierung einer Aktion der Komponente Datenkonverter.	basc, demu	30.5.02	1	muss	x		x	in Arbeit	

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			Komponente Sicherheit der Komponente Datenkonverter die Erlaubnis bzw. ein Verbot.										
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.	basc, demu	30.5.02	1	muss	x		x	in Arbeit	
S-013	GUI	G-052	Bei einer „anonymen Anfrage“ übergibt die Komponente GUI der Komponente Sicherheit die gewünschte Aktion.	Authorisierung einer Aktion der Komponente GUI.	basc, demu	30.5.02	1	muss	x		x	in Arbeit	
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.	basc, demu	30.5.02	1	muss	x		x	in Arbeit	
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.	basc, demu	30.5.02	1	muss	x		x	in Arbeit	x
S-016	GUI	G-044	Die Komponente GUI muss, zur Anmeldung eines Benutzers an das System,	Anmeldung eines Benutzers der Komponente GUI.	basc, demu	30.5.02	1	muss	x		x	in Arbeit	

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			der Komponente Sicherheit den Benutzernamen, das Passwort und die benutzte Schnittstelle übermitteln.										
S-017	S	selbst	Die Komponente Sicherheit übergibt, nach erfolgter Anmeldung, der Komponente GUI eine Session-ID und den Benutzernamen, welcher zu dieser Session gehört.	Anmeldung eines Benutzers der Komponente GUI.	basc, demu	30.5.02	1	muss	x		x	in Arbeit	x
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.	basc, demu	30.5.02	1	muss	x		x	in Arbeit	
S-019	S	selbst	Die Komponente Sicherheit muss Methoden zur Verschlüsselung bereitstellen.	Datensicherheit.	basc, demu	14.6.02	1	muss	x			in Arbeit	x
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	Datensicherheit.	basc, demu	22.6.02	1	muss	x		x	in Arbeit	

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			Absender überprüft.										
S-021	Alle	DK-032, DB-024, BL-027, G-085	Die Komponenten müssen sich an die Sicherheitsrichtlinien halten.	Sicherheit.	basc, demu	22.6.02	1	muss		x		in Arbeit	
S-022	S	selbst	Die Komponente Sicherheit muss eine Public-Key Infrastruktur bereitstellen.	Datensicherheit.	basc, demu	27.6.02	1	muss	x		x	in Arbeit	
S-023	S	selbst	Die Komponente Sicherheit muss Methoden zur Signierung von Nachrichten bereitstellen.	Datensicherheit.	basc, demu	27.6.02	1	muss	x		x	in Arbeit	x
S-024	S	selbst	Die Komponente Sicherheit weißt den Benutzer auf ein veraltetes Kennwort hin.	Sicherheit.	basc, demu	27.6.02	1	muss	x			in Arbeit	x
S-025	S	selbst	Die Komponente Sicherheit stelle Methoden zur Verfügung, um Benutzer anzulegen, zu bearbeiten und zu löschen.	Benutzeradministration.	basc, demu	10.7.02	1	muss	x		x	in Arbeit	x
S-026	S	selbst	Die Komponente Sicherheit stelle Methoden zur Verfügung, um ACL-Groups anzulegen, zu bearbeiten und zu löschen.	Benutzeradministration.	basc, demu	10.7.02	1	muss	x		x	in Arbeit	x
S-027	S	selbst	Die Komponente Sicherheit stelle Methoden	Benutzeradministration.	basc, demu	10.7.02	1	muss	x		x	in Arbeit	x

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			zur Verfügung, um 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.	basc, demu	10.7.02	1	muss	x		x	in Arbeit	
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.	basc, demu	10.7.02	1	muss	x		x	in Arbeit	
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.	basc, demu	10.7.02	1	muss	x		x	in Arbeit	

D.3: Business Logic

ID	Betr. Komponente	Ab. durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
BL-001	BL	selbst	Abbilden aller im System anfallenden Geschäftsprozesse (sowohl für die Durchführung von	Hauptaufgabe der Komponente BL	alsa, tial, naka	4.6.02	1	muss		x		offen	

ID	Betr. Komponente	Ab. durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			GUI- und ImpEx-Requests, wie auch administrative Tätigkeiten des Betreibers)										
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	alsa, tial, naka	30.5.02	1	muss	x		x	offen	x
BL-003	BL	selbst	Die Abläufe einzelner Geschäftprozesse, welche von Pipelines realisiert werden, werden in Form von XML-Files festgehalten.	Die Geschäftsprozesse können mit Pipelines sehr anschaulich und flexibel modelliert werden. Die Verfügbarkeit von Tools ist zu klären	alsa, tial, naka	30.5.02	1	muss	x			offen	x
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	alsa, tial, naka	4.6.02	3	kann	x			offen	
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	alsa, tial, naka	30.5.02	1	muss	x			offen	x
BL-006	BL	selbst	Businesspartnermanager verwaltet alle Daten die	Die Rollen müssen im System repräsentiert	alsa, tial, naka	30.5.02	2	muss	x		x	offen	x

ID	Betr. Komponente	Ab. durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			für einen Businesspartner notwendig ist. Der Manager kann Kundenkonten anlegen, erfragen, löschen, bearbeiten usw.	werden									
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	alsa, tial, naka	30.5.02	2	muss	x		x	offen	
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	alsa, tial, naka	30.5.02	2	muss	x		x	offen	
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	alsa, tial, naka	4.6.02	3	muss	x		x	offen	

ID	Betr. Komponente	Ab. durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
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	alsa, tial, naka	30.5.02	4	muss	x		x	offen	
BL-011	BL	selbst	Personalisationmanager ist für die an die Kundenwünsche angepasste Umgebung zuständig	Value-Added-Service	alsa, tial, naka	30.5.02	4	kann	x		x	offen	
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	alsa, tial, naka	30.5.02	1	muss	x			offen	x
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	alsa, tial, naka	30.5.02	1	muss	x		x	offen	
BL-014	GUI	G-050	Überprüfen der Formulardaten, welche mit dem RequestDictionary übergeben werden auf Typkorrektheit	Die BL führt nur die inhaltliche Korrektheitsüberprüfung der Daten durch	alsa, tial, naka	4.6.02	2	muss		x		offen	
BL-015	GUI	G-068	Bereitstellen von Template-	Das Erscheinungsbild der	alsa, tial,	4.6.02	3	muss	x		x	offen	

ID	Betr. Komponente	Ab. durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			Mechanismen, welche es ermöglichen, aus einem Geschäftsprozess Benachrichtungen via Mail, SMS etc. zu verschicken.	zu verschickenden Benachrichtigungen sollte schnell anpassbar sein und von den Inhalten, welche im Geschäftsprozess ermittelt werden, abstrahiert werden	naka								
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	alsa, tial, naka	30.5.02	1	muss	x		x	offen	
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 EJB-Objekten abgerufen werden können	Die Datenbankkomponente ist der einzige Datenbehälter für fachliche Daten	alsa, tial, naka	30.5.02	1	muss	x		x	offen	
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	alsa, tial, naka	30.5.02	1	muss		x		offen	

ID	Betr. Komponente	Ab. durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
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	alsa, tial, naka	30.5.02	1	muss		x		entfällt	
BL-020	BL	selbst	Zurückgegeben eines Ergebnisses nach einem Workflowdurchlauf, weitere Statusmeldungen werden im RequestDictaionary abgelegt	GUI und DK müssen Antwort auf Request erhalten	alsa, tial, naka	25.6.02	1	muss	x		x	offen	x
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	alsa, tial, naka	25.6.02	2	muss	x			offen	
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	alsa, tial, naka	25.6.02	2	muss	x			offen	
BL-023	BL		So wenig Daten wie möglich übertragen	siehe S-003	alsa, tial, naka	25.6.02	1	muss		x		entfällt	
BL-024	BL		Verantwortlichkeit für übertragene Daten	siehe S-005	alsa, tial, naka	25.6.02	1	muss		x		entfällt	
BL-025	BL		Nur eigene Daten	siehe S-007	alsa, tial,	25.6.02	1	muss		x		entfällt	

ID	Betr. Komponente	Ab. durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			speichern		naka								
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	alsa, tial, naka	25.6.02	3	kann	x			offen	
BL-027	BL	selbst	Einhalten der Sicherheitsrichtlinien	siehe S-021	alsa, tial, naka	30.6.02	1	muss		x		offen	
BL-028	GUI	G-088, G-089	Die GUI muss eine Benutzeroberfläche bereitstellen, mit deren Hilfe man Systemjobs anlegen, löschen und anstossen kann.	Administration	alsa, tial, naka	10.7.02	2	muss	x			offen	

D.4: Datenkonverter

ID	Betr. Komponente	Ab. durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
DK-001	S	S-009	Die Sicherheitskomponente muss den Benutzer identifizieren und authentifizieren können (SSL).	Es dürfen nur berechtigte User Zugriff haben.	chko, zaam	23.6.02	1	muss	x		x	in Arbeit	x
DK-002	S	S-019	Die Sicherheitskomponente muss ausgehende Daten verschlüsseln und	Die Daten dürfen nicht durch Dritte gelesen werden.	chko, zaam	23.6.02	1	muss	x		x	in Arbeit	

ID	Betr. Komponente	Ab. durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			eingehende Daten entschlüsseln können (SSL).										
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.	chko, zaam	19.6.02	1	muss	x		x	in Arbeit	
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.	chko, zaam	11.6.02	1	muss	x		x	in Arbeit	x
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.	chko, zaam	11.6.02	1	muss	x		x	in Arbeit	
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.	chko, zaam	11.6.02	1	muss	x		x	in Arbeit	

ID	Betr. Komponente	Ab. durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
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.	chko, zaam	11.6.02	1	muss	x		x	in Arbeit	
DK-008	DB	DB-007, DB-009	Das DBMS speichert ebXML relevante Dateien (CPP von Com42Bill, CPAs, DTD's, Business Process Specification etc.)	Datenspeicherung	chko, zaam	11.6.02	1	muss	x		x	in Arbeit	
DK-009	DK		Der Datenkonverter stellt der Sicherheitskomponente die nötigen Informationen zur Verfügung, um Requests authentifizieren zu können.	Der User muss eindeutig identifiziert werden	chko, zaam	11.6.02	1	muss	x			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	chko, zaam	11.6.02	1	muss	x			entfällt	
DK-011	DK		Der Datenkonverter stellt der GUI (Workflow-Adapter) ein Container-Objekt bereit, welches die zu verarbeitenden Geschäftsdaten sowie den zugehörigen	Weiterleitung von Geschäftsprozessen und Daten an die Business-Logic.	chko, zaam	11.6.02	1	muss	x			entfällt	

ID	Betr. Komponente	Ab. durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			Geschäftsprozess enthält.										
DK-012	DK	selbst	Der Datenkonverter muss Daten abhängig vom Finanzdienstleister konvertieren und ihm bereitstellen können.	Überweisungsauftrag an Finanzdienstleister übermittelen	chko, zaam	11.6.02	1	muss	x			in Arbeit	x
DK-013	DK		Der Datenkonverter besitzt eine Subkomponente Registry	Verwaltung der ebXML-relevanten Dateien (CPAs etc.)	chko, zaam	11.6.02	1	muss	x			entfällt	
DK-014	DK	selbst	Der Datenkonverter besitzt eine Subkomponente MsgServiceInterface	Eingehende Daten empfangen und enpacken, ausgehende Daten verpacken und versenden (gemäß ebXML Protokoll)	chko, zaam	11.6.02	1	muss	x			in Arbeit	x
DK-015	DK	selbst	Der Datenkonverter besitzt eine Subkomponente MsgServiceInterfaceHandler	Konvertierung Kommunikation mit Sicherheitskomponente, DBMS über Registry, GUI (Workflow-Adapter)	chko, zaam	11.6.02	1	muss	x			in Arbeit	
DK-016	DK	selbst	Erstellung eines CPPs für das Betreiber-System und ein Dummy-CPP für Testzwecke.	notwendig für ebXML	chko, zaam	19.6.02	1	muss		x		in Arbeit	x
DK-017	DK	selbst	Erstellung eines CPAs anhand der beiden CPPs.	notwendig für ebXML	chko, zaam	19.6.02	1	muss		x		in Arbeit	x
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.	chko, zaam	19.6.02	1	muss		x		entfällt	

ID	Betr. Komponente	Ab. durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
DK-019	DK		Der Datenkonverter kommuniziert nur über sichere Verbindungen.	Datensicherheit.	chko, zaam	19.6.02	1	muss		x		entfällt	
DK-020	DK		Der Datenkonverter ist für die übertragenen Daten verantwortlich.	Datensicherheit.	chko, zaam	19.6.02	1	muss		x		entfällt	
DK-021	DK		Der Datenkonverter speichert nur seine eigenen Daten in der Datenbank	Datensicherheit.	chko, zaam	19.6.02	1	muss		x		entfällt	
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.	chko, zaam	19.6.02	1	muss	x			in Arbeit	x
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.	chko, zaam	11.6.02	4	kann	x			in Arbeit	
DK-024	DB	nicht abgedeckt	Die Datenbank soll die Möglichkeit bereitstellen, gezielt auf einzelne XML-Tags zuzugreifen.	Performance	chko, zaam	23.6.02	4	kann	x		x	in Arbeit	
DK-025	DK	selbst	Der Datenkonverter muss jede Datenübertragung protokollieren.	Nachverfolgung bei Problemen.	chko, zaam	23.6.02	1	muss	x			in Arbeit	x

ID	Betr. Komponente	Ab. durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
DK-026	S	S-022	Die Sicherheitskomponente muss eine Public-Key-Infrastruktur bereitstellen.	Datensicherheit.	chko, zaam	23.6.02	1	muss	x		x	in Arbeit	
DK-027	S	S-023	Die Sicherheitskomponente muss einen Mechanismus bereitstellen, um ausgehende Nachrichten digital zu signieren.	Datensicherheit.	chko, zaam	23.6.02	1	muss	x		x	in Arbeit	
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.	chko, zaam	16.6.02	1	muss	x			in Arbeit	
DK-029	DK	selbst	Der Datenkonverter muss Nachrichtenduplikate erkennen und ignorieren können.	Datensicherheit.	chko, zaam	17.6.02	1	muss	x			in Arbeit	
DK-030	DK	selbst	Die Gültigkeitsdauer einer Nachricht wird durch ein TTL-Attribut (Time To Life) festgesetzt.	Datensicherheit.	chko, zaam	18.6.02	1	muss	x			in Arbeit	
DK-031	DK	selbst	Der Datenkonverter benötigt ein Administrationstool, um interne Dateien verwalten	Es müssen Dateien manuell in der Datenbank abgelegt bzw. geändert werden.	chko, zaam	19.6.02	1	muss	x		x	in Arbeit	

ID	Betr. Komponente	Ab. durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
			zu können.										
DK-032	DK	selbst	Der Datenkonverter muss die Sicherheitsrichtlinien der Komponente Sicherheit berücksichtigen	Datensicherheit.	chko, zaam	26.6.02	1	muss	x		x	in Arbeit	
DK-033	DK	selbst	Der Datenkonverter muss der Datenbank ein Objektmodell zur Verfügung stellen.	Abbildung der Daten des DK in der DB	chko, zaam	26.6.02	1	muss		x	x	in Arbeit	
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	chko, zaam	10.7.02	2	muss	x		x	in Arbeit	

D.5: Datenbank

ID	Betr. Komponente	Ab. Durch	Beschreibung	Begründung	Autor	Datum	Priorität	Typ	F	NF	SR	Status	KF
DB-001	DK		Es muß eine Datenstruktur für das ebXML-Repository definiert werden		stpi, anpa	30.5.02	2	muss		x	x	entfällt	
DB-002	DK		Die Methoden für den Zugriff auf das ebXML-Repository müssen definiert werden		stpi, anpa	30.5.02	2	muss		x	x	entfällt	
DB-003	DB	selbst	Implementierung eines DB-User-basierten Zugriffsschutzes	Verhinderung von unberechtigten Zugriffen auf die gespeicherten	stpi, anpa	31.5.02	1	muss	x			in Arbeit	x

	Dokument Systemarchitektur	*1 Tag*	*21.06.02*	*21.06.02*
DK-03	**Klassenmodell**	**20 Tage**	**02.07.02**	**29.07.02**
	UML-Diagramm Klassenmodell	*1 Tag*	*29.07.02*	*29.07.02*
DK-04	**Objektmodell**	**20 Tage**	**28.06.02**	**25.07.02**
	UML-Diagramm Objektmodell	*1 Tag*	*05.07.02*	*05.07.02*
DK-05	**Objektdesign**	**20 Tage**	**05.07.02**	**01.08.02**
	Designdokument	*1 Tag*	*01.08.02*	*01.08.02*
DK-06	**Prototyp (Key Features)**	**45 Tage**	**16.07.02**	**13.09.02**
DK-06.1	CPP für Com42Bill	10 Tage	15.07.02	26.07.02
DK-06.2	CPAs definieren	5 Tage	29.07.02	02.08.02
DK-06.3	Austauschformat/Dokumentstru	5 Tage	05.08.02	09.08.02
DK-06.4	MsgServiceInterfaceHandler	10 Tage	12.08.02	23.08.02
DK-06.5	MsgServiceInterface	5 Tage	26.08.02	30.08.02
DK-06.6	Logging	5 Tage	02.09.02	06.09.02
	Lauffähiger Prototyp	*1 Tag*	*13.09.02*	*13.09.02*
DK-07	Implementierung	41 Tage	20.09.02	15.11.02
DK-07.1	MsgServiceInterface	41 Tage	20.09.02	15.11.02
DK-07.2	MsgServiceInterfaceHandler	41 Tage	20.09.02	15.11.02
DK-07.3	Registry	41 Tage	20.09.02	15.11.02
DK-07.4	Externe Packager	41 Tage	20.09.02	15.11.02
	Lauffähige Komponente	*1 Tag*	*15.11.02*	*15.11.02*
DK-08	Subkomponententest	18 Tage	18.11.02	11.12.02
DK-08.1	MsgServiceInterface	3 Tage	15.11.02	19.11.02
DK-08.2	MsgServiceInterfaceHandler	3 Tage	20.11.02	22.11.02
DK-08.3	Registry	3 Tage	28.11.02	02.12.02
DK-08.4	Externe Packager	3 Tage	03.12.02	05.12.02
	Testergebnis	*1 Tag*	*09.12.02*	*09.12.02*
DK-09	Integration (Subkomponenten)	6 Tage	10.12.02	17.12.02
	Ergebnis der Integration	*1 Tag*	*19.12.02*	*19.12.02*
DK-10	Test des Datenkonverters	16 Tage	20.12.02	10.01.03
	Testergebnis	*1 Tag*	*10.01.03*	*10.01.03*

E.3: Komponente GUI

Task-ID	Task Name	Dauer	Start	Ende
GUI-01	**Anforderungsanalyse**	**40 Tage**	**17.05.02**	**11.07.02**
	Anforderungsdokument	*1 Tag*	*11.07.02*	*11.07.02*
GUI-02	**Systemarchitektur**	**46 Tage**	**17.05.02**	**19.07.02**
	Dokument Systemarchitektur	*1 Tag*	*21.06.02*	*21.06.02*

Task-ID	Task Name	Dauer	Start	Ende
GUI-03	**Klassenmodell**	**61 Tage**	**17.05.02**	**09.08.02**
	UML-Diagramm Klassenmodell	*1 Tag*	*09.08.02*	*09.08.02*
GUI-04	**Objektmodell**	**50 Tage**	**17.05.02**	**25.07.02**
	UML-Diagramm Objektmodell	*1 Tag*	*25.07.02*	*25.07.02*
GUI-05	**Objektdesign**	**85 Tage**	**22.05.02**	**16.09.02**
	Designdokument	*1 Tag*	*16.09.02*	*16.09.02*
GUI-06	**Navigationsstruktur**	**12 Tage**	**08.09.02**	**23.09.02**
GUI-06.1	HTML	12 Tage	08.09.02	23.09.02
GUI-06.2	WAP	12 Tage	08.09.02	23.09.02
	Dokumentation	*1 Tag*	*26.09.02*	*26.09.02*
GUI-07	**Prototyp (Key-Features)**	**55 Tage**	**10.07.02**	**23.09.02**
GUI-07.1	Syntaxkontrolle	20 Tage	15.07.02	09.08.02
GUI-07.2	Datenbank-Templates	7 Tage	12.08.02	20.08.02
GUI-07.3	Presenter	8 Tage	21.08.02	01.09.02
GUI-07.4	Workflowtest	7 Tage	02.09.02	09.09.02
GUI-07.5	History	8 Tage	11.09.02	20.09.02
	Lauffähiger Prototyp	*1 Tag*	*23.09.02*	*23.09.02*
GUI-08	**Implementierung**	**50 Tage**	**23.09.02**	**29.11.02**
GUI-08.1	RequestServlet	50 Tage	23.09.02	29.11.02
GUI-08.2	RequestDispatcher	50 Tage	23.09.02	29.11.02
GUI-08.3	HistoryController	50 Tage	23.09.02	29.11.02
GUI-08.4	RequestPresenter	50 Tage	23.09.02	29.11.02
GUI-08.5	WorkflowAdapter	50 Tage	23.09.02	29.11.02
GUI-08.6	HTML/WAP	50 Tage	23.09.02	29.11.02
	Lauffähige Komponente	*1 Tag*	*29.11.02*	*29.11.02*
GUI-09	**Subkomponententest**	**30 Tage**	**11.11.02**	**20.12.02**
GUI-09.1	RequestServlet	30 Tage	11.11.02	20.12.02
GUI-09.2	RequestDispatcher	30 Tage	11.11.02	20.12.02
GUI-09.3	HistoryController	30 Tage	11.11.02	20.12.02
GUI-09.4	RequestPresenter	30 Tage	11.11.02	20.12.02
GUI-09.5	WorkflowAdapter	30 Tage	11.11.02	20.12.02
	Testergebnis	*1 Tag*	*20.12.02*	*20.12.02*
GUI-10	**Integration (Subkomponenten)**	**15 Tage**	**02.12.02**	**20.12.02**
	Ergebnis Integration	*1 Tag*	*20.12.02*	*20.12.02*
GUI-11	**Testen des GUIs**	**22 Tage**	**12.12.02**	**10.01.03**
	Testergebnis	*1 Tag*	*10.01.03*	*10.01.03*

E.4: Komponente Sicherheit

Task-ID	Task Name	Dauer	Start	Ende
S-01	**Anforderungsanalyse**	**55 Tage**	**26.04.02**	**11.07.02**
	Anforderungsdokument	*1 Tag*	*15.07.02*	*15.07.02*
S-02	**Systemarchitektur**	**55 Tage**	**06.05.02**	**19.07.02**
	Dokument Systemarchitektur	*1 Tag*	*19.07.02*	*19.07.02*
S-03	**Klassenmodell**	**47 Tage**	**16.05.02**	**19.07.02**
	UML-Diagramm Klassenmodell	*1 Tag*	*19.07.02*	*19.07.02*
S-04	**Objektmodell**	**35 Tage**	**07.06.02**	**25.07.02**
	UML-Diagramm Objektmodell	*1 Tag*	*26.07.02*	*26.07.02*
S-05	**Objektdesign**	**30 Tage**	**01.07.02**	**09.08.02**

Task-ID	Task Name	Dauer	Start	Ende
	Designdokument	*1 Tag*	*09.08.02*	*09.08.02*
S-06	**Prototyp (Key-Features)**	**51 Tage**	**16.07.02**	**23.09.02**
S-06.1	Autorisierung GUI	25 Tage	15.07.02	16.08.02
S-06.2	Authentifizierung GUI	6 Tage	19.08.02	26.08.02
S-06.3	Autorisierung DK	3 Tage	27.08.02	29.08.02
S-06.4	Authentifizierung DK	4 Tage	30.08.02	04.09.02
S-06.5	Administration	12 Tage	05.09.02	19.09.02
S-06.6	Kennwort	1 Tag	20.09.02	20.09.02
	Lauffähiger Prototyp	*1 Tag*	*23.09.02*	*23.09.02*
S-07	**Implementierung**	**40 Tage**	**23.09.02**	**15.11.02**
S-07.1	Request Authorization	42 Tage	03.10.02	29.11.02
S-07.2	Authorization/Authentification	42 Tage	03.10.02	29.11.02
S-07.2.1	Transaction	42 Tage	03.10.02	29.11.02
S-07.2.2	User	42 Tage	03.10.02	29.11.02
S-07.3	Session Manager	42 Tage	03.10.02	29.11.02
	Lauffähige Komponente	*1 Tag*	*19.11.02*	*19.11.02*
S-08	**Subkomponententest**	**22 Tage**	**20.11.02**	**19.12.02**
S-08.1	Request Authorization	22 Tage	02.12.02	31.12.02
S-08.2	Authorization/Authentification	22 Tage	02.12.02	31.12.02
S-08.2.1	Transaction	22 Tage	02.12.02	31.12.02
S-08.2.2	User	22 Tage	02.12.02	31.12.02
S-08.3	Session Manager	22 Tage	02.12.02	31.12.02
	Testergebnis	*1 Tag*	*19.12.02*	*19.12.02*
S-09	**Integration (Subkomponenten)**	**12 Tage**	**02.12.02**	**17.12.02**
	Ergebnis Integration	*1 Tag*	*17.12.02*	*17.12.02*
S-10	**Testen der Komponente Sicherheit**	**10 Tage**	**30.12.02**	**10.01.03**
	Testergebnis	*1 Tag*	*10.01.03*	*10.01.03*

E.5: Komponente Business Logic

Task-ID	Task Name	Dauer	Start	Ende
BL-01	**Anforderungsanalyse**	**60 Tage**	**15.04.02**	**05.07.02**
	Anforderungsdokument	*1 Tag*	*05.07.02*	*05.07.02*
BL-02	**Systemarchitektur**	**29 Tage**	**04.06.02**	**12.07.02**
	Dokument Systemarchitektur	*1 Tag*	*12.07.02*	*12.07.02*
BL-03	**Klassenmodell**	**23 Tage**	**01.07.02**	**31.07.02**
	UML-Diagramm Klassenmodell	*1 Tag*	*31.07.02*	*31.07.02*
BL-04	**Objektmodell**	**49 Tage**	**20.05.02**	**25.07.02**
	UML-Diagramm Objektmodell	*1 Tag*	*26.07.02*	*26.07.02*
BL-05	**Objektdesign**	**20 Tage**	**18.07.02**	**14.08.02**
	Designdokument	*1 Tag*	*14.08.02*	*14.08.02*
BL-06	**Prototyp (Key-Features)**	**56 Tage**	**15.07.02**	**27.09.02**
BL-06.1	Rechnungsempfängerverwaltung	15 Tage	15.07.02	04.08.02
BL-06.2	BusinessObjects für Beispielworkflow	5 Tage	05.08.02	11.08.02
BL-06.3	Modellierung eines Beispielworkflows	10 Tage	12.08.02	25.08.02
BL-06.4	Workflowverwaltung	10 Tage	27.08.02	08.09.02

Task-ID	Task Name	Dauer	Start	Ende
BL-06.5	Ergbnisrückgabe an GUI (Interaktion)	5 Tage	09.09.02	15.09.02
BL-06.6	Transaktionsverwaltung mit EJB-Container	9 Tage	16.09.02	26.09.02
	Lauffähiger Prototyp	*1 Tag*	*27.09.02*	*27.09.02*
BL-07	**Implementierung**	**47 Tage**	**12.09.02**	**15.11.02**
BL-07.1	Workflowmanager	47 Tage	26.09.02	29.11.02
BL-07.2	Business Objects	47 Tage	26.09.02	29.11.02
BL-07.3	Workflow modellieren	47 Tage	26.09.02	29.11.02
BL-07.4	Businesspartner Manager	47 Tage	26.09.02	29.11.02
BL-07.5	Invoice Manager	47 Tage	26.09.02	29.11.02
BL-07.6	Reporting Manager	47 Tage	26.09.02	29.11.02
BL-07.7	Personalisation Manager	47 Tage	26.09.02	29.11.02
	Lauffähige Komponente	*1 Tag*	*29.11.02*	*29.11.02*
BL-08	**Subkomponententest**	**17 Tage**	**20.11.02**	**12.12.02**
BL-08.1	Workflowmanager	17 Tage	02.12.02	24.12.02
BL-08.2	Business Objects	17 Tage	02.12.02	24.12.02
BL-08.3	Workflow	17 Tage	02.12.02	24.12.02
BL-08.4	Businesspartner Manager	17 Tage	02.12.02	24.12.02
BL-08.5	Invoice Manager	17 Tage	02.12.02	24.12.02
BL-08.6	Reporting Manager	17 Tage	02.12.02	24.12.02
BL-08.7	Personalisation Manager	17 Tage	02.12.02	24.12.02
	Testergebnis	*1 Tag*	*24.12.02*	*24.12.02*
BL-09	**Integration (Subkomponenten)**	**4 Tage**	**24.12.02**	**27.12.02**
	Ergebnis Integration	*1 Tag*	*27.12.02*	*27.12.02*
BL-10	**Testen der Komponente BL**	**5 Tage**	**06.01.03**	**10.01.03**
	Testergebnis	*1 Tag*	*10.01.03*	*10.01.03*

E.6: Komponente Datenbank

Task-ID	Task Name	Dauer	Start	Ende
DB-01	**Anforderungsanalyse**	**64 Tage**	**15.04.02**	**11.07.02**
	Anforderungsdokument	*1 Tag*	*11.07.02*	*11.07.02*
DB-02	**Objektmodell Com42Bill**	**54 Tage**	**29.04.02**	**11.07.02**
	UML-Diagramm Objektmodell	*1 Tag*	*11.07.02*	*11.07.02*
DB-03	**Klassenmodell DataBaseAccess.**	**66 Tage**	**15.07.02**	**11.10.02**
	UML-Diagramm Klassenmodell	*1 Tag*	*18.10.02*	*18.10.02*
DB-04	**Objektdesign DataBaseAccess**	**71 Tage**	**12.07.02**	**17.10.02**
	Designdokument	*1 Tag*	*18.10.02*	*18.10.02*
DB-05	**Installation DB-Server**	**5 Tage**	**01.07.02**	**05.07.02**
DB-06	**Konfiguration DB-Server**	**15 Tage**	**08.07.02**	**26.07.02**
DB-06.1	User	15 Tage?	08.07.02	26.07.02
DB-06.2	Netzwerkanbindung	15 Tage?	08.07.02	26.07.02
DB-06.3	Sicherheit	15 Tage?	08.07.02	26.07.02
	Ergebnis	*1 Tag*	*26.07.02*	*26.07.02*
DB-07	**Prototyp (Key-Features)**	**56 Tage**	**15.07.02**	**27.09.02**
DB-07.1	Persistenz Rechnungen	56 Tage	15.07.02	27.09.02
DB-07.2	Persistenz Zahlungsvorgang	56 Tage	15.07.02	27.09.02
DB-07.3	Persistenz Businesspartner	56 Tage	15.07.02	27.09.02

Task-ID	Task Name	Dauer	Start	Ende
DB-07.4	Persistenz Systemdaten	56 Tage	15.07.02	27.09.02
DB-07.5	Persistenz User	56 Tage	15.07.02	27.09.02
DB-07.6	Persistenz GUI	56 Tage	15.07.02	27.09.02
DB-07.7	Persistenz Datenkonverter	56 Tage	15.07.02	27.09.02
DB-07.8	Persistenz Sicherheit	56 Tage	15.07.02	27.09.02
	Lauffähiger Prototyp	*1 Tag*	*27.09.02*	*27.09.02*
DB-08	**Implementierung**	**35 Tage**	**21.10.02**	**06.12.02**
DB-08.1	Komponente DataBaseAccess	31 Tage	25.10.02	06.12.02
DB-08.2	Komponente DataBasePersistence	31 Tage	25.10.02	06.12.02
	Lauffähiger Prototyp	*1 Tag*	*06.12.02*	*06.12.02*
DB-09	**Subkomponententest**	**5 Tage**	**09.12.02**	**13.12.02**
DB-09.1	Komponente DataBaseAccess	5 Tage	09.12.02	13.12.02
DB-09.2	Komponente DataBasePersistence	5 Tage	09.12.02	13.12.02
	Testergebnis	*1 Tag*	*13.12.02*	*13.12.02*
DB-10	**Integration (Subkomponenten)**	**5 Tage**	**16.12.02**	**20.12.02**
	Ergebnis Integration	*1 Tag*	*20.12.02*	*20.12.02*
DB-11	**Testen der Komponente DB**	**20 Tage**	**23.12.02**	**17.01.03**
	Testergebnis	*1 Tag*	*10.01.03*	*10.01.03*

E.7: Integrationstest der Komponenten

Task-ID	Task Name	Dauer	Start	Ende
IT-01	**Integrationstest GUI**	**20 Tage?**	**13.01.03**	**07.02.03**
IT-02	**Integrationstest DK**	**20 Tage?**	**13.01.03**	**07.02.03**
IT-03	**Integrationstest BL**	**20 Tage?**	**13.01.03**	**07.02.03**
IT-04	**Integrationstest Sicherheit**	**20 Tage?**	**13.01.03**	**07.02.03**
IT-05	**Integrationstest DB**	**20 Tage?**	**13.01.03**	**07.02.03**
	Testergebnis	*1 Tag*	*24.02.03*	*24.02.03*

E.8: Querschnittsaufgaben

Task-ID	Task Name	Dauer	Start	Ende
QA-01	**Projektmanagement**	**225 Tage**	**15.04.02**	**20.02.03**
	Projektplan	*1 Tag*	*14.06.02*	*14.06.02*
	Pflege und Wartung des Projektplan	182 Tage	14.06.02	21.02.03
	Review Projektplan 1	*1 Tag*	*25.07.02*	*25.07.02*
	Review Projektplan 2	*1 Tag*	*29.08.02*	*29.08.02*
	Review Projektplan 3	*1 Tag*	*26.09.02*	*26.09.02*
	Review Projektplan 4	*1 Tag*	*24.10.02*	*24.10.02*
	Review Projektplan 5	*1 Tag*	*28.11.02*	*28.11.02*
	Review Projektplan 6	*1 Tag*	*19.12.02*	*19.12.02*
	Review Projektplan 7	*1 Tag*	*16.01.03*	*16.01.03*
	Review Projektplan 8	*1 Tag*	*30.01.03*	*30.01.03*
	Review Projektplan 9	*1 Tag*	*13.02.03*	*13.02.03*
	Review Projektplan 10	*1 Tag*	*27.02.03*	*27.02.03*
	Dokument Anforderungsmanagement	*1 Tag*	*20.06.02*	*20.06.02*

Task-ID	Task Name	Dauer	Start	Ende
	Abschlußbericht	*1 Tag*	*20.02.03*	*20.02.03*
QA-02	**Marketing**	**225 Tage**	**15.04.02**	**20.02.03**
	T-Shirts	*1 Tag*	*28.06.02*	*28.06.02*
	Whitepaper	33 d	16.05.02	01.07.02
	Zwischenbericht	*1 Tag*	*29.07.02*	*29.07.02*
	Plakat	*1 Tag*	*08.08.02*	*08.08.02*
	Kontakt_1 (Kontaktaufnahme)	*1 Tag*	*12.07.02*	*12.07.02*
	Kontakt_2 (Produkt Präsentation)	*1 Tag*	*13.02.03*	*13.02.03*
QA-03	**Qualitätsmanagement**	**225 Tage**	**15.04.02**	**20.02.03**
	Code Guide	*1 Tag*	*28.06.02*	*28.06.02*
	Testplan Integrationstest	*1 Tag*	*27.12.02*	*27.12.02*
	Testplan Systemtest	*1 Tag*	*31.01.03*	*31.01.03*
	Pflege und Wartung des Anforderungsdokuments	177 Tage	20.06.02	20.02.03
	Erstellung eines Prozessmodells	24 Tage	24.06.02	25.07.02
	Pflege und Wartung des Prozessmodells	152 Tage	25.07.02	20.02.03
	Pflege und Wartung der Key-Feature	152 Tage	25.07.02	20.02.03
QA-04	**GUI**	**225 Tage**	**15.04.02**	**20.02.03**
	Style Guide für Web, WAP	*1 Tag*	*26.07.02*	*26.07.02*
QA-05	**Chef-Architekten**	**225 Tage**	**15.04.02**	**20.02.03**
	Systemarchitektur	45 Tage	29.04.02	28.06.02
	Pflege und Wartung der Systemarchitektur	171 Tage	28.06.02	20.02.03
	Schnittstellenüberwachung	130 Tage	02.07.02	27.12.02
	Installation & Einrichtung Bea-Weblogic	21 Tage	04.07.02	01.08.02
QA-06	**Systemadministration**	**225 Tage**	**15.04.02**	**20.02.03**
	Einrichtung Server	1 Tag	31.05.02	31.05.02
	Einrichtung Workstations	1 Tag	31.05.02	31.05.02
	CVS Repository	1 Tag	31.05.02	31.05.02
	Pflege und Wartung der PG Homepage	224 Tage	16.04.02	20.02.03
	Einrichtung eines FTP-Zugangs	13 Tage	20.06.02	08.07.02
	Installation & Einrichtung Bea-Weblogic	21 Tag	04.07.02	01.08.02

Anhang F: Sicherheitsrichtlinien

- Alle Benutzer des Systems Com42Bill müssen sich über die Komponente Sicherheit anmelden.
- Es wird nur über sichere Verbindungen mit der Außenwelt kommuniziert.
- Alle Daten, welche mit der Außenwelt ausgetauscht werden, müssen verschlüsselt werden.
- Die übertragene Datenmenge ist möglich klein zu halten.
- Die Daten werden in verarbeiteter Form übertragen. Es findet keine Datenverarbeitung auf dem Klienten statt.
- Daten werden nur in der Datenbank gespeichert.
- Daten werden nur von der Komponente gespeichert, von der diese Daten auch verwaltet werden.

Anhang G: Verzeichnisse

G.1: Abbildungsverzeichnis

Sofern in den Grafiken nicht anders gekennzeichnet, wurden die Abbildungen durch die Autoren selbst erstellt.

Abbildung 1: Welche Zahlungsarten würden Sie im Internet bevorzugen? 6
Abbildung 2: Aufbau eines 3- und mehrschichtigen Systems 10
Abbildung 3: Teilarchitektur der Komponente GUI 21
Abbildung 4: Teilarchitektur der Komponente Sicherheit 23
Abbildung 5: Pipeline-Architektur 25
Abbildung 6: Teilarchitektur der Komponente Business Logic 27
Abbildung 7: Kommunikation von Com42Bill mit anderen Systemen 28
Abbildung 8: Teilarchitektur der Komponente Datenkonverter 30
Abbildung 9: Teilarchitektur der Komponente Datenbank 32
Abbildung 10: Grobe Package-Struktur des Systems 36
Abbildung 11: Struktur des Package ebppsystem 37
Abbildung 12: Package-Struktur des Sub Package core 38
Abbildung 13: Klassen des Package common 39
Abbildung 14: Klassen des Package session 40
Abbildung 15: Klassen des Package user 41
Abbildung 16: Klassen des Package job 41
Abbildung 17: Klassen des Package request 42
Abbildung 18: Klassen des Sub Package alert 43
Abbildung 19: Klassen des Sub Package businesspartner 44
Abbildung 20: Klassen des Sub Package invoice 45
Abbildung 21: Klassen des Package payment 46
Abbildung 22: Package-Struktur des Package components 47
Abbildung 23: Klassen des Package gui 48
Abbildung 24: Klassen des Package history 49
Abbildung 25: Klassen des Package news 50
Abbildung 26: Klassen des Package help 50
Abbildung 27: Klassen des Package templates 51
Abbildung 28: Klassen des Package dataconverter 52
Abbildung 29: Klassen des Package security 53
Abbildung 30: Notation 59
Abbildung 31: Entwicklungsprozess 61
Abbildung 32: Anforderungsanalyse 61
Abbildung 33: Anforderungen 62

Abbildung 34: Modelle 63
Abbildung 35: Technologieentscheidung 64
Abbildung 36: Auswahl von Key-Features 64
Abbildung 37: GUI-Design 65
Abbildung 38: Klassenmodell 65
Abbildung 39: Testplan 66
Abbildung 40: Prototyping 67
Abbildung 41: Implementierung 68
Abbildung 42: Übersicht 70
Abbildung 43: Rechnungsempfänger 71
Abbildung 44: Finanzdienstleister 71
Abbildung 45: Rechnungssteller 72
Abbildung 46: Use Case Rechnungssteller 73
Abbildung 47: Use Case Rechnungsempfänger 73
Abbildung 48: Use Case Verwaltung 74
Abbildung 49: Use Case Konfiguration Datenaustausch 75
Abbildung 50: Use Case Benutzerverwaltung 75
Abbildung 51: Aktivitätsdiagramm Registrierung RE 76
Abbildung 52: Aktivitätsdiagramm Anmeldung RE 77
Abbildung 53: Aktivitätsdiagramm Verwaltung 78
Abbildung 54: Aktivitätsdiagramm Kontenverwaltung RE 79
Abbildung 55: Aktivitätsdiagramm Rechnungspräsentation 80
Abbildung 56: Aktivitätsdiagramm Mahnungen 80
Abbildung 57: Aktivitätsdiagramm Mahnungen prüfen 81
Abbildung 58: Aktivitätsdiagramm Zahlung 82
Abbildung 59: Aktivitätsdiagramm Rechnungs- / Zahlungsstatus 83
Abbildung 60: Aktivitätsdiagramm Anmelden RS 84
Abbildung 61: Aktivitätsdiagramm RS registrieren 85
Abbildung 62: Aktivitätsdiagramm Finanzkonto anlegen 86
Abbildung 63: Aktivitätsdiagramm Finanzkonto editieren 87
Abbildung 64: Aktivitätsdiagramm Finanzkonto löschen 87
Abbildung 65: Aktivitätsdiagramm RS-Konto editieren 88
Abbildung 66: Aktivitätsdiagramm Kunden-ID prüfen 88
Abbildung 67: Aktivitätsdiagramm Bericht abrufen 89
Abbildung 68: Aktivitätsdiagramm Statistik abrufen 90
Abbildung 69: Aktivitätsdiagramm Job anlegen 91
Abbildung 70: Aktivitätsdiagramm Job ausführen 92

Abbildung 71: Aktivitätsdiagramm Job löschen 93
Abbildung 72: Aktivitätsdiagramm zeitgesteuerte Transaktionen 94
Abbildung 73: Aktivitätsdiagramm Login DK 95
Abbildung 74: Aktivitätsdiagramm neues CPA / CPP anlegen 96
Abbildung 75: Aktivitätsdiagramm CPA / CPP ändern 97
Abbildung 76: Aktivitätsdiagramm CPA / CPP hinzufügen 97
Abbildung 77: Aktivitätsdiagramm CPP / CPA löschen 97
Abbildung 78: Aktivitätsdiagramm CPA / CPP bearbeiten 98

G.2: Literaturverzeichnis

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

[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] 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

[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

[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

[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
June 2001

/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