Hochschule für Technik und Wirtschaft Dresden(FH)

Fachbereich Informatik/Mathematik

# Diplomarbeit

im Studiengang Wirtschaftsinformatik

**Thema: Entwurf & prototyp. Implementierung eines Buchungssystems für den Zahlungsverkehr zu elektron. Verträgen auf der Grundlage einer verteilten Java-Applikation.**

eingereicht von: Andreas Schöbel
Matrikelnummer: 3620

eingereicht am: 13.08.2001

Betreuer: Prof. Dr.-Ing. Arnold Beck (HTW)
Dip.-Ing. Mehrdad Jalali (IGD-FhG)

# Inhaltsverzeichnis

Nach §24 Abs. 14 ADPO hat die Hochschule ein einfaches Nutzungsrecht an der Diplomarbeit für Lehre und Forschung. Zum Beginn der Bearbeitung werden daher eventuelle schutzrechtliche Voraussetzungen zur anderweitigen Nutzung der zu erarbeitenden Ergebnisse geklärt.

Dresden, August 2001

Hiermit versichere ich, dass ich die Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Andreas Schöbel

# Abbildungsverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| ACTS | Advanced Communications Technologies and Services |
| AGBG | Gesetz für Allgemeine Geschäftsbedingungen |
| ANSI | American National Standardization Institute |
| API | Application Programming Interface |
| ASF | Apache Software Foundation |
| ASN.1 | Abstract Syntax Notation One |
| B2B | Busines to Business |
| B2C | Business to Consumer |
| CA | Certification Authority |
| DBMS | Database Management Systeme |
| DES | Data Encryption Standard |
| DSA | Digital Signature Algorithm |
| DTD | Document Type Definition |
| EDI | Electronic Data Interchange |
| EDIFACT | EDI for Administration Commerce and Transport |
| ERD | Entity Relationship Diagram |
| FCA | Fee Collecting Agency |
| FhG | Fraunhofer Gesellschaft |
| Filigrane | Flexible IPR for Software Agent Reliance |
| IDE | Integrated Developement Enviroment |
| IGD | Institut für Grafische Datenverarbeitung |
| HTML | Hyper Text Markup Language |
| HTTP | Hypertext Transfer Protokoll |
| IP | Internet Protocol |
| ITU-T | ITU Telecommunication Standardization Sector |

| | |
|---|---|
| JAR | Java Archive |
| JCE | Java Cryptography Extension |
| JDBC | Java Database Connectivity |
| JDK | Java Developer Kit |
| JSP | JavaServer Pages |
| JSSE | Java Secure Socket Extension |
| OSI | Open Systems Interconnection |
| RCH | Rights Clearing House |
| RFC | Request for Comment |
| RSA | Public-Key Algorithmus nach Rivest, Shamir und Adleman |
| SDK | Software Developer Kit |
| SEMPER | Secure Electronic Market Place for Europe |
| SET | Secure Electronic Transaction |
| SGML | Standard Generalized Markup Language |
| SRDL | Software Rights Description Language |
| SQL | Structured Query Language |
| SSL | Secure Socket Layer |
| TCP | Transmission Control Protocol |
| TLS | Transport Layer Security |
| TTP | Trusted Third Party |
| UML | Unified Model Language |
| W3C | World Wide Web Consortium |
| WWW | World Wide Web |
| XML | Extensible Markup Language |
| XSL | Extensible Stylesheet |

# Kapitel 1

# Einleitung

## 1.1 Motivation

Die Entwicklung des Internets und die daraus resultierende Entstehung des WWW galt und gilt, gerade in den letzten Jahren, als Motor und Impulsgeber für immer neue Technologien und Handelsmöglichkeiten. Begriffe wie E-Business oder E-Commerce spielen dabei immer öfter eine Rolle. Besonders E-Commerce, der Handel im Internet, gewinnt in letzter Zeit immer mehr an Bedeutung. Dienstleister vertreiben beispielsweise Bücher, CDs und andere Güter über das weltumfassende Netzwerk. Aus diesem Grund werden Anwendungen benötigt, die eine einfache Handhabung der Handelsabläufe, speziell für Software und mediale Inhalte, ermöglichen. Die Entwicklung solcher Applikationen bedingt die Sicherung der Rechte der Hersteller mit technischen Mitteln und die automatische Regelung des Zahlungsflusses, der vom Konsumenten bis zum Hersteller entsteht.

## 1.2 Aufgabenstellung

Ziel dieser Diplomarbeit ist es, ein Buchungssystem für das bereits bestehende Teilsystem zu entwerfen und prototypisch in Java zu implementieren. Die Aufgabe des geplanten Buchungssystems besteht darin, den Geldverkehr

zwischen den einzelnen Nutzern dieser verteilten Applikation zu erfassen und entsprechend zu regeln. Auf Basis der beim Rights Clearing House (RCH) gespeicherten Verträge soll das Buchungssystem die Einnahmen des Kaufes an die beteiligten Vertragsparteien verteilen. Dabei soll sichergestellt werden, daß alle Parteien, die von dem entsprechenden Vertrag direkt und indirekt betroffen sind, über den Umsatz des Providers in Kenntnis gesetzt werden.

## 1.3 Umfeld der Aufgabenstellung

Das Umfeld der Aufgabenstellung ist eine, im Institut für Grafische Datenverarbeitung (IGD) Darmstadt der Fraunhofer Gesellschaft (FhG) entworfene, verteilte Architektur und wurde bis zum Zeitpunkt dieser Diplomarbeit teilweise in Java prototypisch implementiert.
Die Anforderungen für dieses Rahmenwerk stammen aus einem Szenario, in dem es mehrere Anbieter gibt, die mit Verbrauchern, Produzenten aber auch untereinander Handel treiben. Aus diesem Grund gibt es mehrere zentrale Komponenten, die den Handel überwachen und sichern.

## 1.4 Implementierungsumgebung

Die Implementierung des Prototyps wurde in Java mit dem Java Software Developer Kit (SDK) 1.3 von Sun Microsystems durchgeführt. Dabei diente Kawa 4.01 für Windows NT von Tek-Tools als Integrated Developement Enviroment (IDE). Zur Visualisierung der UML-Diagramme wurde Rational Rose verwendet.

## 1.5 Struktur der Arbeit

Im Anschluß an die hier erläuterte Aufgabenstellung und deren Randbedingungen, werden im 2. Kapitel die Grundlagen und Anforderungen für

E-Commerce-Applikationen erläutert, auf denen die Arbeit aufbaut. In Kapitel 3 wird das Modell entwickelt, welches letztendlich die Funktionalitäten des FCA nach den Anforderungen der Aufgabenstellung erfüllen soll. Weiterführend beschreibt Kapitel 4 die für die Implementierung entwickelten Objekte und Datenstrukturen. Abschließend werden in Kapitel 5 die Ergebnisse zusammengefaßt und ein Ausblick auf weitere Entwicklungen gegeben.

# Kapitel 2

# Anforderungen und Grundlagen

In diesem Kapitel werden die Grundlagen für den zu entwickelnden Anwendungsteil geschaffen. Dabei wird eine kurze Einführung in die für diese Applikation genutzten Techniken und Technologien gegeben. Desweiteren wird der Begriff E-Commerce definiert, dessen Bedeutung erläutert sowie das Filigrane-Konzept und dessen Funktionsweise erklärt. Darauf aufbauend werden anschließend die Anforderungen an die geplante Anwendung formuliert.

## 2.1 Grundlagen

### 2.1.1 Kryptographie & Datensicherheit

Kryptographie bezeichnet die Kunst des Geheimschreibens und ist in dieser Bedeutung bis heute aktuell geblieben. Sie stellt dazu Methoden und Algorithmen zur Verfügung, die Kommunikation oder Dokumente vor unbefugtem Zugriff schützen und dadurch die Vertraulichkeit von Informationen sicherstellen sollen.[guth]

Moderne Kryptographieverfahren verwenden Schlüssel, die sich der Anwender selbst erzeugt, um die entsprechenden Daten zu kodieren bzw. zu dekodieren. So können nur Personen mit dem passenden Schlüssel darauf zugreifen. Ein Schlüssel besteht dabei aus einer Reihe von aufeinander folgenden Bits. Die Anzahl der Bits ist die Schlüssellänge, welche die Stärke bzw. Sicherheit der Verschlüsselung bestimmt. Als Faustregel für das Maß der Sicherheit gilt der Aufwand, der betrieben werden muß, um die Information zu entschlüsseln. Ist dieser Aufwand größer als der Nutzen, den ein potentieller Angreifer daraus ziehen könnte, gilt das Verfahren als sicher.
Algorithmen moderner Verfahren sind allgemein bekannt, was die Transparenz erhöht, die Fehlerwahrscheinlichkeit verringert und dadurch die Sicherheit erhöht.
Verschlüsselungsverfahren können der Methode nach in symmetrische und in asymmetrische unterschieden werden, wobei symmetrische nur einen Schlüssel und asymmetrische ein Schlüsselpaar besitzen.
Außerdem bietet die Kryptographie Methoden, die sicherstellen sollen, daß kein Angreifer eine falsche Identität vortäuscht und das Informationen nicht verändert wurden.

### Symmetrische Verschlüsselungsverfahren

Symmetrische Kryptographieverfahren, auch Secret-Key-Verfahren genannt, verwenden zur Ver- und Entschlüsselung ein und denselben Schlüssel, im Idealfall sogar denselben Algorithmus. Ihr Vorteil ist die schnelle Chiffrierung und Dechiffrierung. Aus diesem Grund wird versucht, den Großteil der zu übertragenden Daten mit symmetrischen Verfahren zu verschlüsseln. Das Problem dieser Methode besteht in der Weitergabe des Schlüssels, was nur über einen sicheren Transportkanal geschehen darf (siehe Abb. 2.1).
Alice und Bob wollen zum Beispiel einen Text verschlüsselt über das Internet austauschen. Dazu erzeugt Alice einen Schlüssel und kodiert mit diesem den Text. Alice gibt den Secret-Key nun über einen abhörsicheren Weg an Bob weiter, womit er den empfangenen Text jetzt entschlüsseln kann. [lehn]

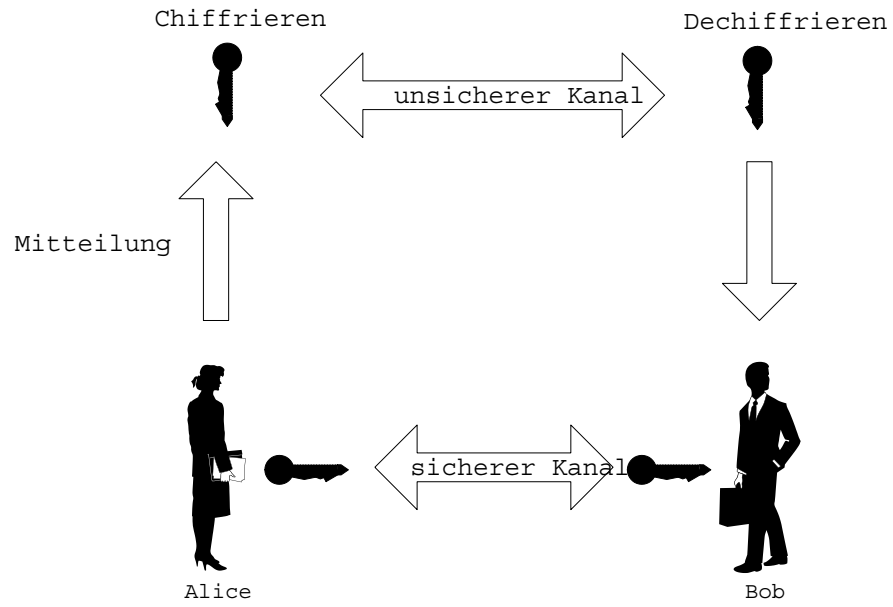


Abbildung 2.1: Symmetrische Verschlüsselung

### Asymmetrische Verschlüsselungsverfahren

Asymmetrische Kryptographieverfahren, auch als Public-Key-Verfahren bekannt, verwenden, im Gegensatz zu den symmetrischen, ein Schlüsselpaar, bestehend aus einem öffentlichen und einem privaten Schlüssel. Die zu schützenden Daten werden mit einem Schlüssel kodiert und mit dem dazugehörigen dekodiert (siehe Abb. 2.2).
Ein Beispiel: Alice will Bob eine Nachricht verschlüsselt zukommen lassen. Dazu übergibt Bob seinen öffentlichen Schlüssel, den sogenannten Public-Key, an Alice. Hat sie die entsprechende Nachricht kodiert, kann allein Bob die Nachricht mit seinem Private-Key dekodieren. [lehn]

Dieses Verfahren eröffnet noch eine weitere Möglichkeit: die Kommunikationspartner können sich gegenseitig identifizieren. Dazu muß Alice ihre Nachricht an Bob nochmals mit ihrem privaten Schlüssel chiffrieren. Kann Bob die Nachricht, nach dem Dechiffrieren mit seinem Private-Key, mit Hilfe von Alice' öffentlichen Schlüssels dekodieren, ist die Nachricht eindeutig von ihr.

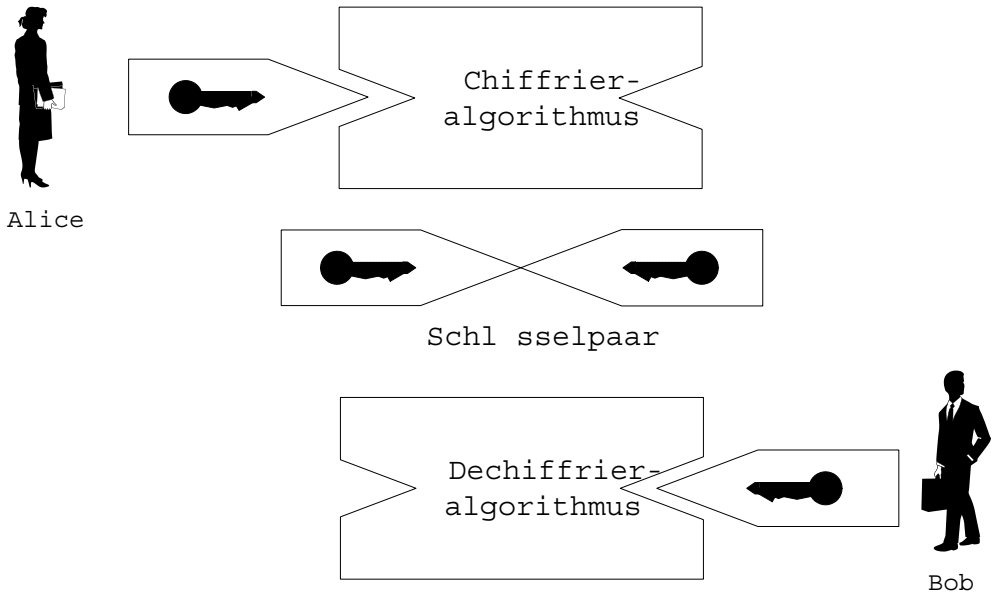


Abbildung 2.2: Asymmetrische Verschlüsselung

Der Nachteil solcher erfahren liegt im hohen Rechenaufwand beim Ver- und Entschlüsseln. Vorteilhaft ist die einfache Methode des Schlüsseltausches, da zum Dekodieren ein anderer Schlüssel des Schlüsselpaares benötigt wird, als zum Kodieren.

### Authentifizierung mit digitalen Signaturen

Wie schon im vorangegangenen Abschnitt erwähnt, eröffnen Public-Key-Verfahren einen Weg, sich gegenüber einem Partner mittels einer digitalen Signatur auszuweisen. Die eigentliche Identifikation funktioniert nach dem schon oben beschriebenen Prinzip. Digitale Signaturen gehen aber noch einen Schritt weiter. Sie identifizieren nicht nur den jeweiligen Kommunikationspartner, zusätzlich sichern sie die Integrität der zu übertragenden Informationen.
Dazu wird mittels einer sogenannten Hashfunktion ein digitaler Fingerabdruck (Message Digest oder Hashwert) der Information erzeugt. Dieser Wert wird nun mit dem Private-Key des Senders verschlüsselt und an die Nach-

richt angehängt. Der Empfänger kann den Hashwert mit dem Public-Key des Senders entschlüsseln und weiß dadurch, daß die Nachricht von ihm ist. Hat er die Information entschlüsselt, erzeugt er ebenfalls einen Hashwert davon und vergleicht diesen mit dem soeben entschlüsselten. Sind beide identisch, ist die Information nicht manipuliert worden.[ct95]

### Digitale Zertifikate

Wie digitale Signaturen, sind digitale Zertifikate ebenfalls eine Art der elektronischen Unterschrift. Anders als bei digitalen Signaturen wird der öffentliche Schlüssel eines Schlüsselpaares eindeutig einer Person oder Instanz zugeordnet. Das geschieht durch Zertifizierungsstellen, die dem Signaturgesetz entsprechend zugelassen sind. Diese sogenannte Certification Authorities (CA) erzeugt aus den Daten des Schlüsseleigners eine Signatur, wodurch sie den Inhaber eindeutig identifiziert. Dadurch wird ein rechtsverbindlicher Abschluss von Verträgen mit dieser Art der digitalen Unterschrift möglich. Ein weitverbreiteter Standard ist das X.509-Zertifikat, was auf der Empfehlung ITU Telecommunication Standardization Sector (ITU-T) beruht und sich allgemein durchgesetzt hat. Die Kodierung des Zertifikats ist eindeutig, wodurch z.B. der Name oder der Public-Key des Zertifikatinhabers eindeutig identifiziert werden kann. Ein X.509-Zertifikat besteht aus den folgenden Elementen:

- Version (version): unterscheidet die Versionen des Zertifikats
- Seriennummer (serial number): eine Integerzahl, die das Zertifikat bei der Zertifizierungsinstanz eindeutig kennzeichnet
- Algorithmus Identifikator (algorithm identifier): beschreibt den Algorithmus und weitere Parameter, die zur Verifikation der Signatur notwendig sind
- Herausgeber (issuer): verweist auf die CA, welche das Zertifikat herausgegeben hat

- Gültigkeitsdauer (period of validity): definiert den Gültigkeitszeitraum durch Anfangs- und Enddatum
- Zugehöriger Anwender (subject): kennzeichnet den Besitzer des Zertifikats durch Namesattribute, wie z.B. Vor- und Nachname oder Anschrift
- Public-Key (public key information): beschreibt den Public-Key selbst sowie für welchen Algorithmus dieser eingesetzt werden kann
- Signatur (signature): Einbeziehung aller vorangegangener Elemente unter Verwendung einer Hash-Funktion

### Secure Socket Layer (SSL)

Das von Netscape entwickelte Netzwerkprotokoll basiert auf dem TCP/IP-Protokoll und dient zur Sicherung des Übertragungskanals zwischen zwei Kommunikationspartnern. Dafür nutzt SSL symmetrische und asymmetrische Verschlüsselungsverfahren gleichermaßen.

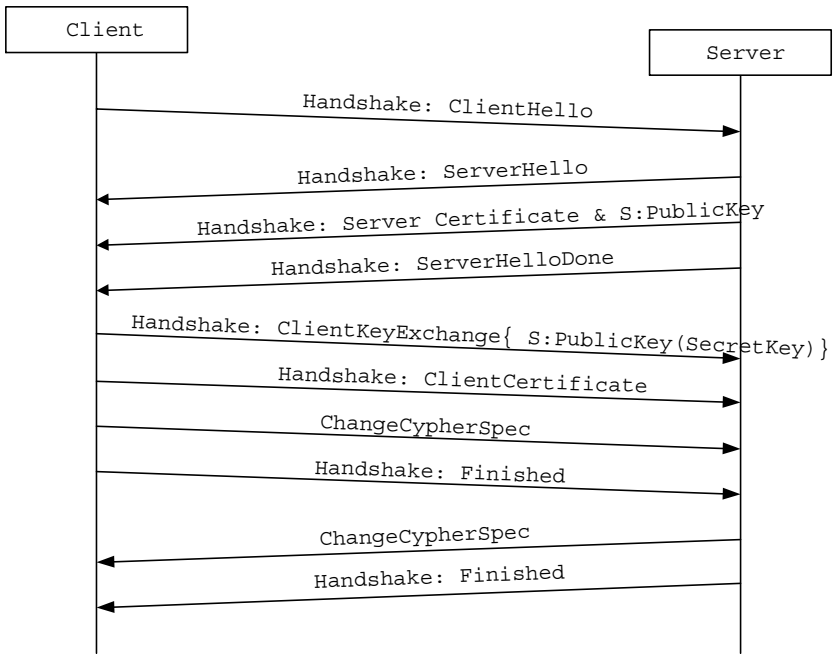


Abbildung 2.3: SSL Verbindungsaufbau mit Client-Authentication

Wie in Abb. 2.3 dargestellt, schickt der Server sein X.509-Zertifikat und seinen öffentlichen Schlüssel, auf die Anfrage des Clients. Das Zertifikat kann der Client nun bei der CA, die das Zertifikat erstellt hat, überprüfen. Optional kann der Server die Authentifizierung des Clients verlangen. Dazu muß sich der Client, wie der Server, mit einem gültigen Zertifikat ausweisen. Bei Richtigkeit der Daten erzeugt der Client einen symmetrischen Schlüssel mit einem bestimmten Verschlüsselungsverfahren. Dazu hat er vom Server eine Liste mit den unterstützten Algorithmen erhalten. Dieser Schlüssel, der sogenannte Session-Key, wird nun mit dem vorher erhaltenen Public-Key des Servers chiffriert und an ihn zurückgesendet. Nun kommunizieren beide Partner auf Basis dieses symmetrischen Schlüssels. Unter Angabe der Session-Number können spätere Verbindungen mit dem gleichen Schlüssel wieder aufgebaut werden, was den Verbindungsaufbau erheblich verkürzt.[guth]
SSL nutzt unter anderem die symmetrischen Algorithmen Triple DES, DES oder RC4 bzw. die asymmetrischen Verfahren RSA oder DSA.

### 2.1.2 Elektronische Zahlungsmittel im Einzelhandel

Es gibt eine Vielzahl von Zahlungsmethoden, die zur unmittelbaren Abwicklung von Käufen der Konsumenten im WWW geeignet sind. Diese Verfahren wurden größtenteils speziell für den Interneteinsatz entwickelt, wodurch schnelle, einfache und kostengünstige Methoden geschaffen wurden, die auch noch für Zahlungen von Kleinstbeträgen rentabel sind. Die Klassifizierung der einzelnen Verfahren ist je nach Anspruch aus verschiedenen Blickwinkeln möglich, wie Abb. 2.4 verdeutlicht.[ukai]
Traditionelle Zahlungsverfahren, wie Nachnahme, Rechnung oder Vorauskasse, sind bereits durch den Versandhandel gut bekannt und erprobt. Wurde Nachnahme als Zahlungsart vereinbart, findet die Zahlung zum Lieferzeitpunkt statt. Wird die Leistung gegen Rechnung erbracht, erfolgt die Zahlung innerhalb einer Frist. Im Gegensatz dazu zahlt der Kunde bei Vorauskasse schon vor der Leistungserbringung.

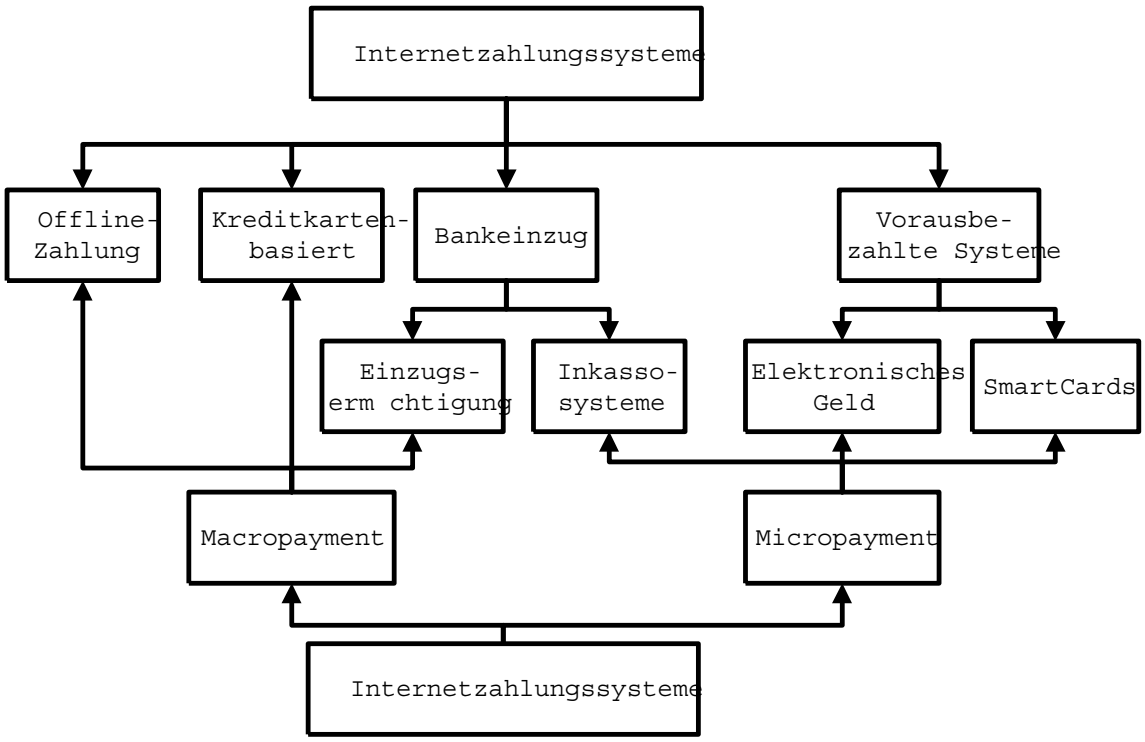


Abbildung 2.4: Klassifizierung elektronischer Zahlunsverfahren

Die im Internet eingesetzten Verfahren unterscheiden sich in Bezug auf Zahlungszeitpunkt und Zeitpunkt der Leistungserbringung nur unerheblich von den traditionellen. Dementsprechend gibt es Methoden, die die Zahlung zum Zeitpunkt der Leistungserbringung, davor oder danach ermöglichen.

### Vorausbezahlte Verfahren

Bei diesen Verfahren muß der Kunde schon vor dem eigentlichen Kauf eine finanzielle Vorleistung bei einem Händler erbringen. Das stellt aber kein allgemeines Zahlungsverfahren für das Internet dar, da der Kunde nur ein Guthaben bei dem entsprechenden Händler hat und damit nur dort einkaufen kann. Dadurch sichert der Händler die Bonität des Kunden, allerdings auf dessen Risiko, da das Geld vor der eigentlichen Leistungserbringung auf dem Händlerkonto gutgeschrieben wird.

### Digitales Geld

Der Kunde muß, wie bei vorausbezahlten Verfahren, eine finanzielle Vorleistung erbringen, jedoch nicht beim Händler. Wie beim Abheben von Bargeld, tauscht der Nutzer Buchgeld von seinem vom Girokonto gegen digitales Geld ein und lädt damit seine elektronische Geldbörse auf. Das daraus entstandene digitale Guthaben kann er für Zahlungen im Internet nutzen, vorausgesetzt der Händler unterstützt dieses Zahlungssystem. Dadurch kann der Kunde nie mehr Geld ausgeben, als in der Geldbörse gerade vorhanden ist und die Zahlungen finden immer beim Abschluß des Vertrags statt. Die bekanntesten Systeme sind eCash von eCash Technologies und die Geldkarte von Giesecke & Devrient.

### Zahlungen via Bankeinzug

Im Gegensatz zu den bisher genannten Verfahren, wird hier die Zahlung erst nach Abschluss des Vertrags durchgeführt. Dazu gibt der Kunde dem Händler die Berechtigung, sein Konto mit dem vereinbarten Betrag zu belasten, indem er ihm seine Bankverbindung nennt. Unsicherheiten dieser Verfahren, sind die mögliche mangelnde Liquidität sowie falsche Kontoangaben des Kunden.
Eine sicherere Art sind Zahlungen über einen Mittler, wie Inkassostellen oder Kreditkartengesellschaften. Der Mittler steht zwischen Händler und Kunde, indem er erst die ausgegebenen Beträge sammelt, zu einem bestimmten Zeitpunkt vom Kundenkonto abbucht und, nach Abzug einer Provision, an den Händler weiterleitet. Dadurch wird das Risiko falscher Angaben gemindert und das Risiko der Zahlungsunfähigkeit des Kunden geht auf den Mittler über.
Um die bei diesen Verfahren entstehenden Risiken auf ein Minimum zu senken, wurde Secure Electronic Transaction (SET) geschaffen. Nutzt der Kunde SET zur Zahlung, wird sein Konto entweder direkt oder indirekt durch die Kreditkarte belastet. Es ermöglicht eine sichere Identifizierung der Handelspartner durch spezielle digitale Zertifikate und sichert zusätzlich die Liquidität des Kunden. Dadurch sind beide Seiten gegen Mißbrauch geschützt.

#### SEMPER

SEMPER steht für **S**ecure **E**lectronic **M**arket**P**lace for **E**u**R**ope und ist ein Teilprojekt des ACTS-Programms der Europäischen Kommission. Dessen Ziel ist die Entwicklung eines Rahmenwerks zur Sicherung des elektronischen Handels. Ein Teilergebnis dieses Projekts ist eine Klassenbibliothek zur sicheren Abwicklung elektronischer Zahlungen für E-Commerce-Anwendungen. Sie ermöglicht eine einfache Implementation und Integration digitaler Zahlungsverfahren in elektronischen Geschäftsanwendungen.[sdfr]

### 2.1.3 EDI(FACT)

Electronic Data Interchange (EDI) ist ein seit über 20 Jahren existierender Kommunikationsstandard, bei dem kommerzielle und technische Daten plattformunabhängig zwischen Computern bzw. Applikationen räumlich getrennter Geschäftspartner ausgetauscht werden. Dieser internationale Standard soll die Weiterverarbeitung strukturierter Geschäftsdaten ohne Medienbruch automatisieren und damit beschleunigen. [ecin]
Strukturierte Geschäftsdaten sind alle Informationen, die sich in Form von Formularen abbilden lassen. So werden unter anderem Rechnungen, Bestellungen, Lieferscheine, Zolldokumente oder Zahlungsaufträge zwischen Geschäftspartnern, Banken und Behörden elektronisch ausgetauscht und ohne menschliche Eingriffe automatisch verarbeitet.
Zur Gewährleistung dieses Datenaustausches existiert eine Art Regelwerk, bestehend aus standardisierten Datenformaten, welches die Abbildung der Informationen darstellt. **E**lectronic **D**ata **I**nterchange **F**or **A**dministration, **C**ommerce and **T**ransport (EDIFACT) stellt dabei den weltweit gültigen und branchenübergreifenden Standard dar. Neben EDIFACT existieren noch sogenannte Subsets, welche exakt definierte Untermengen darstellen und lediglich nationale oder branchenweite Bedeutung besitzen.
Wird EDI implementiert, und damit im Unternehmen integriert, können vollautomatische Geschäftsprozesse modelliert werden, die dem traditionellen Geschäftsdatenaustausch auf Papier, in Bezug auf Geschwindigkeit, Kosten

und Fehleranfälligkeit, weit überlegen sind.

EDI-Dokumente bestehen aus drei Teilen:

- dem Transaktionsset,
- den EDI-Segmenten und
- den EDI-Elementen.

In einem ANSI-X12-konformen Transaktionsset müssen Segmente und Elemente so voneinander getrennt sein, daß ersichtlich ist, wann ein Teil endet. Segmente werden meistens durch Carriage Return und Elemente durch das Asterisk-Symbol ("*") voneinander getrennt. Transaktionsets bestehen aus Segmenten, die wiederum aus Elementen bestehen. Das erste Element jedes Segments ist der Segmentcode. Durch das logische Zusammenfügen von Segmenten und Elementen entsteht ein Transaktionsset nach X12.[xedi]

### Transaktionsset

Das Transaktionsset ist eine Sammlung von Segmenten, die ein Geschäftsdokument ausmachen. Die meisten sind, wie das Vorbild im Papierformat, dreiteilig bestehend aus Kopf, Details und Zusammenfassung.

### EDI-Segmente

Ein Segment ist eine Gruppe logisch verwandter Informationen, die durch zwei- oder dreistellige alphanumerische Zeichen identifiziert werden. Dabei können verschiedene Transaktionssets den gleichen Segmenttyp nutzen. Weiterhin können sie obligatorisch oder optional sein und, abhängig von den Syntaxregeln des entsprechenden Transaktionssets, ein oder mehrere Male vorkommen.
Segmente können auch Teil einer Schleife sein. Eine Schleife ist eine Gruppe gleichartiger Segmente mit einer vorgeschriebenen Reihenfolge. Syntaxregeln jedes Transaktionssets legen fest, wie oft die Schleife durchlaufen wird.

**EDI-Elemente**

Elemente sind logisch zusammengehörige Daten, die ein Segment ausmachen. Jedes Element wird durch einen ein- bis vierstelligen numerischen Code definiert. Mehrere verschiedene Segmente können die gleichen Elementcodetypen nutzen.

### 2.1.4 eXtensible Markup Language (XML)

Diese, vom World Wide Web Konsortium (W3C) entwickelte, Metasprache ist ein Dokumentenverarbeitungsstandard, der eine vereinfachte Form der Standard Generalized Markup Language (SGML) darstellt. Im Gegensatz zu HTML ist XML nicht statisch, d.h. es existieren keine statisch vordefinierten Elemente, sogenannte Tags. Weiterhin können eigene Formatierungselemente, sogenannte Markups, geschaffen und das Dokument damit strukturiert werden. [oxml]
XML-Dokumente bestehen in der Regel aus 3 Teilen respektive Dateien:

1. einem Stylesheet, basierend auf der eXtensible Stylesheet Language (XSL), welcher die letztendliche Präsentation der Daten beschreibt.
2. dem Document Type Definition (DTD), der die inhaltliche Struktur des Dokuments beschreibt. Hier werden die Tags für das XML-Dokument definiert. Zur Vereinfachung hat das W3C inzwischen das XML-Schema für diese Aufgabe eingeführt, welches die bisherigen DTDs ablösen soll.
3. dem eigentlichen XML-Dokument, das die nach dem DTD oder XML-Schema beschriebenen Inhalte enhält.

Derzeit können die Browser von Microsoft (InternetExplorer) und Operasoftware (Opera) diese Dokumente verarbeiten und teilweise darstellen.

### 2.1.5 EDI in Verbindung mit XML

EDI ist der Standard für den Dokumentenaustausch zwischen Unternehmen. Durch den langjährigen Einsatz existiert eine große Basis an Anwendern,

bei denen sich dieses Format bewährt hat. Doch EDI-Dokumente haben ein kryptisches Dokumentenformat und erzeugen bei einer Implementierung hohe Kosten. Im Gegensatz dazu ermöglicht XML eine gut lesbare Darstellungsform von Daten und die Umsetzung einer XML-Implementierung ist nicht so kostspielig wie bei EDI. Aus diesem Grund wurde damit begonnen EDI-Dokumente in XML darzustellen. Mit Hilfe von XML/EDI-Translatoren kann auf bereits bestehende EDI-Infrastrukturen aufgebaut werden und die Vorteile von XML genutzt werden. [xedi]

## 2.2 E-Commerce

Trotz der derzeitigen Repression in der New Economy, wird der Handel bzw. Einkauf im World Wide Web immer beliebter. In den vergangenen 12 Monaten ist die Zahl der „Online-Shopper“ deutlich angestiegen. Nach aktuellen Studien nutzt bereits jeder Zehnte die Einkaufsangebote im Internet, wobei es im Jahr davor noch 5% waren. Derzeit sind Bücher und CDs die begehrtesten im Internet erhältlichen Artikel, doch andere Bereiche holen bereits stark auf.[fsnw]
Der Verkauf von materiellen Gütern ist dabei nur ein Zweig des Internethandels. Wie Napster und andere Dienstleister beweisen, gewinnt auch der Handel mit medialen Inhalten, wie z.B. Bildern oder Musik, und die Vermietung von Software durch einen entsprechenden Anbieter immer mehr an Bedeutung.
Doch E-Commerce ist nicht nur der Handel in diesem Bereich. E-Commerce steht für alle wirtschaftlichen Prozesse, an denen Verbraucher, Hersteller, Händler sowie weitere Dienstleister beteiligt sind, die mit Hilfe von elektronischen Netzwerken, wie z.B. dem Internet, durchgeführt werden. E-Commerce ist der Oberbegriff für alle wirtschaftlichen Aktivitäten die mit dem Handel in Computernetzwerken verbunden sind. [ec99]

Die Vorteile des elektronischen Handels sind :

- die Erhöhung der Geschwindigkeit und Effektivität wirtschaftlicher Prozesse und Transaktionen,
- die Verbesserung des Kontakts und Services zwischen Geschäftspartnern sowie
- die Verbesserung des Wettbewerbs.

Vorallem kleine und mittelständische Unternehmen erhalten dadurch die Chance, ihre Güter einfach und preiswert auf der ganzen Welt zu präsentieren und zu offerieren. Für den Verbraucher hingegen ergeben sich ebenfalls neue Möglichkeiten. Er steht einer größeren Auswahl an Produkten und Dienstleistungen gegenüber, was infolgedessen zu reduzierten Preisen und verbessertem Service führen wird. [us00]
E-Commerce unterteilt sich nach Art der Geschäftspartner in zwei Bereiche:

- Busines-to-Business-Bereich (B2B), bezieht sich auf den Güteraustausch zwischen Unternehmen
- Business-to-Consumer-Bereich (B2C), bezeichnet den Einzelhandel

Im B2B-Bereich waren schon vor der Verbreitung des Internets Lösungen für die Kommunikation und den Datenaustausch vorhanden. Dabei spielt EDI die bedeutendste Rolle.
Der B2C-Bereich verbreitete sich erst mit der Expansion des Internets, wobei er auch die populärsten E-Commerce-Projekte in sich beherbergt. Der klassische Versandhandel, angereichert mit den Möglichkeiten des Internets und den Vorteilen elektronischer Datenverarbeitung, hat Anbieter wie Amazon bekannt gemacht.

## 2.3 Filigrane

Im Rahmen eines europäischen Projektes Namens **F**lex**I**b**L**e **I**PR for Software A**G**ent **R**eli**AN**c**E** (Filigrane) wurde am IGD der FhG in Darmstadt eine Sicherheitsarchitektur entwickelt. Das Ziel des Projekts war die Entwicklung eines frei konfigurierbaren Sicherheitsrahmens für den Handel mit mobilem Code, unter der Vorgabe, die Rechte des Urhebers, vorrangig technisch, zu sichern.
Innerhalb des Projekts kristallisierten sich nach [fil12] die folgenden Komponenten heraus:

- **Producer**: ist der Produzent von Software oder medialen Inhalten. Er stellt den Urheber, dessen Rechte gewahrt werden sollen, dar.
- **Provider**: ist der Dienstleister, der die Produkte des Producers für Customer auf seiner Filigrane-Plattform anbietet.
- **Customer**: stellt den Konsumenten dar, der die entsprechenden Produkte beim Provider nachfragt und diese nutzt.
- **Fee Collecting Agency**: hat die Aufgabe, den Geldfluß zwischen den einzelnen Akteuren (Producer, Provider und Customer) automatisch anhand der geschlossenen Verträge zu verwalten.
- **Rights Clearing House**: ist eine zentrale Schiedsstelle, die die Rechte des Producers durch Speicherung und Registrierung der elektronischen Verträge wahrt.
- **Certification Authority**: stellt eine Trusted Third Party (TTP) dar, die den Filigrane-Akteuren elektronische Zertifikate ausstellt.
- **Quality Label Service**: testet und überprüft die entwickelten Produkte auf Funktion und Qualität und garantiert dafür mit seiner elektronischen Signatur.
- **E-Notary**: ist ein sogenannter Monitoring Service, der sämtliche Ereignisse und Transaktionen innerhalb des Filigrane-Systems protokolliert.

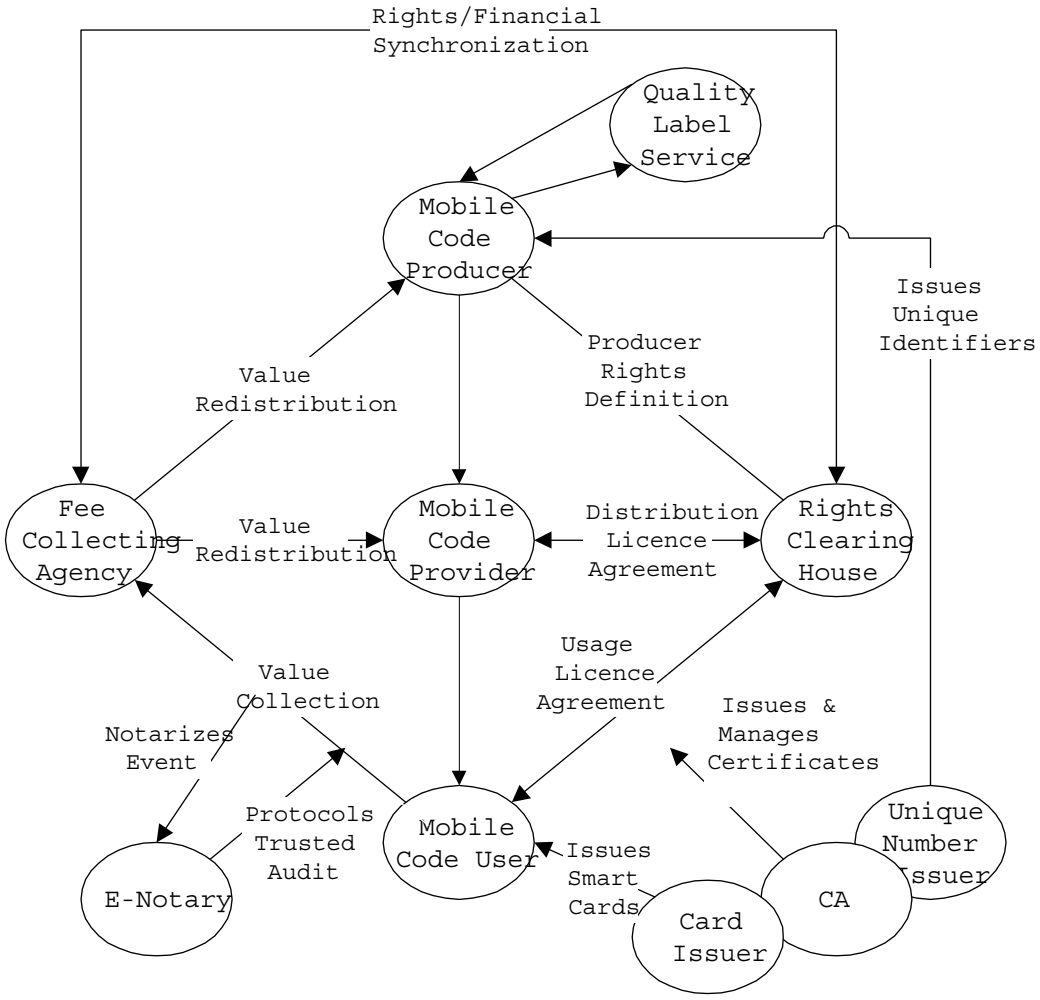


Abbildung 2.5: Filigrane Architektur

Im Gegensatz zu dem in Abb. 2.5 dargestellten Ansatz, soll der Entwurf einen dezentralen Ansatz verfolgen. Zum Zeitpunkt der Entstehung dieser Arbeit ist der Einkauf des Customers beim Provider implementiert, wozu auch ein Demonstrator existiert.
Das Filigrane-Konzept basiert auf elektronischen Verträgen, welche durch die XML-basierende Software Rights Description Language (SRDL) dargestellt werden. Sie beschreiben die zu einem Produkt gehörigen Lizenzierungs- und Nutzungsbedingungen. Die Lizenzverträge werden dem Zweck nach in zwei Arten unterschieden. Die erste Art sind Vertragslizenzen, welche die

vom Lizenzgeber beanspruchten Rechte für die Weiterlizenzierung beschreiben. Sie existieren nur zwischen Providern und Developern bzw. zwischen Providern untereinander. Die zweite Art versteht sich als Nutzungslizenz und beinhaltet die Nutzungsbedingungen, die an ein Produkt durch die entsprechende Vertragslizenz gebunden sind. Diese Lizenzart kann nur vom Customer erworben werden.

## 2.4 Anforderungen an E-Commerce

Verfolgt man ein Produkt von der Herstellung bis zur Nutzung, lassen sich die Handelspartner in drei verschiedene Gruppen unterteilen und daraus die späteren Nutzertypen für E-Commerce-Applikationen ableiten. Diese Typen sind Produzenten (Producer), Anbieter (Provider) und Konsumenten (Customer). Dementsprechend gibt es aus Sicht jeder einzelnen Nutzergruppe verschiedene Anforderungen an eine solche Applikation. Hinzu kommen weitere Anforderungen, die unabhängig von den einzelnen Anwendergruppen formuliert werden können.
Die angestrebte Anwendung soll in erster Linie einen zuverlässigen Betrieb, in einer akzeptablen Geschwindigkeit und entsprechender Sicherheit gewährleisten. Das ist einerseits von der eingesetzten Hardware, in Bezug auf das Netzwerk und die Rechenleistung, und andererseits von der entworfenen Softwarearchitektur abhängig.
Die Auswahl der Hardware und die Planung der Infrastruktur ist immer vom jeweiligen Anwendungsziel abhängig und wird aus diesem Grund nicht näher erläutert.
Ein wichtiger Aspekt innerhalb solcher Applikation ist die Datensicherheit. Die vorrangigen Angriffspunkte sind die eingesetzten Server sowie die Kommunikation zwischen den einzelnen Komponenten der geplanten Anwendung. Der Schutz der Server erfordert ein Sicherungskonzept, das unbefugte Zugriffe von vornherein abwehrt. Beispielsweise können sämtliche Zugriffe durch Authentifizierung authorisiert werden.

Die Sicherung der Kommunikationsdaten wird über :

- die Verschlüsselung der Übertragungskanäle oder der Kommunikationsdaten,
- die eindeutige Identifizierung der Kommunikationspartner und
- die Versiegelung der zu übertragenden Informationen

vorgenommen.
Weiterhin spielen auch juristische Aspekte eine Rolle. Ein Handel im Internet unterliegt den gleichen Gesetzen, wie im Supermarkt oder bei einer Katalogbestellung. In der Essenz kommt dabei ein Vertrag zwischen 2 Parteien zustande. Voraussetzung hierfür ist wiederum ein verbindliches Angebot einer Partei an die andere.
Der Abschluß eines Vertrages besteht im grundlegenden aus folgenden Schritten:

1. dem Angebot, d.h. Leistungen werden zu bestimmten Konditionen angeboten
2. dem Vertragsschluss, d.h. das Angebot wird angenommen, wodurch ein Vertrag zustande kommt
3. der Leistungserbringung, d.h. es findet der vereinbarte Austausch von Leistung und Gegenleistung zwischen beiden Parteien statt

Ein Vertrag bedarf im Regelfall keiner schriftlichen Form, wodurch sich prinzipiell eine rechtliche Basis für Geschäfte im Internet ergibt. Allerdings sind Abmachungen ohne deren Dokumentation juristisch schwer nachweisbar, weshalb darauf nicht verzichtet werden sollte. Weiterhin ist die Beweiskraft elektronischer Dokumente aufgrund ihrer leichten Veränderbarkeit sehr gering.

Deshalb müssen elektronische Verträge die folgenden Punkte erfüllen, damit sie rechtlich durchsetzbar sind:

- Integrität der elektronischen Dokumente
- Sicherstellung der Identität der Vertragsparteien

Beide Anforderungen werden durch den Einsatz digitaler Signaturen sichergestellt, deren Verbindlichkeit durch das Signaturgesetz geregelt wird. [dedig] Nach Abschluß des Vertrags, erbringt eine der Parteien ihre Leistung meist als Zahlung. Im Internet gibt es die Möglichkeit, dies mit Hilfe elektronischer Zahlungsmittel zu tun. In dieser Hinsicht müssen elektronische Zahlungsverfahren absichern, daß

- der korrekte Betrag gesendet wird,
- die digitalen Werte den richtigen Empfänger erreichen und
- der Zahlungseingang quittiert wird.

Prinzipiell müssen elektronische Zahlungssysteme genug Sicherheit bieten, um sich gegen Angriffe und Missbrauch zu schützen.
Aus Sicht des Konsumenten muß das gewünschte Verfahren transparent, verständlich und einfach bedienbar sein. Die Anonymität der Zahlung, zur Gewährleistung des Datenschutzes, ist eine weitere, aber nicht zwingende, Anforderung.
Für Produzenten und Anbieter spielen die Transaktionskosten ein große Rolle. Sie sollen so gering sein, daß auch sehr niedrige Zahlungsbeträge noch rentabel sind. Bei einem höheren Betrag spielen die Transaktionskosten eine untergeordnete Rolle, dafür tritt die Absicherung der Zahlung in den Vordergrund.
Zusätzlich müssen noch ergonomische Gesichtspunkte bei der Implementierung beachtet werden, um den Nutzern die Arbeit mit der geplanten Applikation einfach, logisch und transparent zu gestalten. Daten, beispielsweise für den elektronischen Katalog, müssen einfach eingegeben, geändert oder gelöscht werden können. Stammen sie aus anderen Quellen, wie z.B. Bestandsdaten, sollen sie automatisiert übergeben werden.

## 2.5 Anforderungen an Filigrane

Die im vorangegangenen Teil behandelten Anforderungen an E-Commerce-Anwendungen betreffen natürlich auch Filigrane und werden in diesem Abschnitt für den geplanten Anwendungsteil verfeinert.
Der Zugriff auf den Providerteil von Filigrane erfordert grundsätzlich die Authentifizierung des Nutzers. Das setzt wiederum die Registrierung mit einem gültigen X.509-Zertifikat voraus.
Für die Kommunikation ist dieses Zertifikat ebenfalls erforderlich, da für die Sicherung der Übertragungskanäle SSL mit Client-Authentication eingesetzt wird. Zusätzlich werden die übertragenen Informationen mit der Signatur des jeweiligen Absenders versehen, was die Sicherheit und Integrität der Daten gewährleistet. Die zu übertragenden Information sollen dabei im XML-Format dargestellt werden.
Die technische Umsetzung der reglementierten Nutzung durch elektronische Verträge erfolgt mit Hilfe von Verschlüsselungsverfahren und SmartCards.
Die Aufgabe des FCA ist die Regelung des Zahlungsverkehrs in Filigrane. Dies soll automatisch und ohne Medienbruch realisiert werden, was nur durch elektronische Zahlungsverfahren möglich ist. Beim Entwurf ist zu beachten, daß neue Verfahren einfach implementiert und integriert werden können.
Die Auswahl der zu implementierenden Verfahren soll eine Liquiditätsprüfung des Customers ermöglichen und dessen Anonymität, wenn möglich, sichern. Da das Transaktionsvolumen auch Kleinstbeträge annehmen kann, spielen die Transaktionskosten bei der Auswahl eines Zahlungssystems eine maßgebende Rolle.
Um die Gesamtkosten für finanzielle Transaktionen niedrig zu halten, soll deren Zahl reduziert werden. Dazu sollen die eingehenden Beträge erst gebucht und addiert werden. Zu einem vom Nutzer festgelegten Zeitpunkt soll der FCA die eingegangenen Buchungen ausgleichen. Die Einnahmen sollen dann bei der Bank des Nutzers gutgeschrieben und die Empfangsbestätigung dem Nutzer übergeben werden. Die Kommunikation hierfür erfordert einen Datenaustausch im EDI- oder EDI-XML-Format.
Die Verteilung der Lizenzeinnahmen an die direkt und indirekt betroffenen Parteien soll durch ein Zusammenspiel von FCA und RCH gesichert werden.

Das RCH sichert die Einhaltung der geschlossenen Verträge ab, indem es die Gültigkeit der weitergegebenen Lizenzen überprüft. Deshalb muß der FCA jede Lizenz beim RCH prüfen lassen, was dem RCH ermöglicht, alle am Erlös beteiligten Parteien zu registrieren und auf Anfrage zu informieren. Zusätzlich soll der FCA den entsprechenden Lizenzgeber, falls dieser ebenfalls ein Provider ist, über seine Einnahmen informieren. Um eine Umgehung des RCH und nachträgliche Änderungen der Nutzungslizenz zu verhindern, benötigt der Customer zur Nutzung eines gekauften Produkts eine vom RCH signierte Lizenz.

# Kapitel 3

# Modellentwicklung

Die Modellentwicklung beinhaltet den Modellentwurf nach der Analyse des Ist-Zustands. Das Vorgehen gestaltet sich dabei unter den Gesichtspunkten der vorher erläuterten Anforderungen. Mit dem aus der Ist-Analyse gewonnenen Wissen, wurden die geplanten Funktionalitäten in Objekte formuliert und die dazugehörigen Daten strukturiert.

## 3.1 Ist-Zustand

Bisher wurde die Architektur für einen Demonstrator teilweise prototypisch implementiert. Die für diese Anwendung nötigen Daten wurden durch Konfigurationsdateien vorinitialisiert bzw. manuell in die Datenbank eingetragen. Der Demonstrator zeigt den Ablauf eines Nutzungskaufs durch den Customer beim Provider. Der Ablauf stellt sich dabei wie folgt dar.
Zuerst meldet sich der Customer am Filigrane-Server an, wodurch er den elektronischen Katalog erreicht. Dort werden verschiedene Programme angeboten. Nach der Auswahl eines Produkts und der Übergabe seines X.509-Zertifikats kann er die Nutzungsdaten bestimmen. Dazu stellt er Nutzungszeitraum, Nutzungsdauer und Anzahl der Starts für das gewünschte Programm ein. Daraufhin berechnet und erstellt der Server ein Angebot für den Customer. Bestätigt er das Angebot, wird der zu zahlende Betrag von seinem

Konto abgebucht sowie eine verschlüsselte Jar-Datei erstellt und zum Download zur Verfügung gestellt. In der Jar-Datei befindet sich der erworbene Artikel mit dessen Nutzungsbedingungen. Der dazugehörige Schlüssel wird mit einem Teil der Nutzungsbedingungen auf die SmartCard des Customers geschrieben, wodurch der Kauf abgeschlossen ist.
Zur Nutzung des über Filigrane erworbenen Programms, braucht der Customer den Filigrane-Client. Dieser entpackt die heruntergeladene Jar-Datei mit dem entsprechenden Schlüssel von der SmartCard und führt das Programm aus. Nach Ablauf der Lizenz wird der Schlüssel ungültig und der Customer kann das Produkt nicht mehr nutzen.
Der Demonstrator teilt sich in mehrere Komponenten auf. Auf der Clientseite existieren der Filigrane-Client mit einer Schnittstelle zur SmartCard und der Webbrowser.

**Filigrane-Client**

Der Filigrane-Client ermöglicht dem Filigrane-Server das Schreiben auf die SmartCard des Nutzers. Weiterhin reglementiert er die Einhaltung der Nutzungsbedingungen. Aus diesem Grund wird er auch bei der Nutzung der Produkte benötigt.

Die Serverseite unterteilt sich in drei Schichten, bestehend aus :

- dem Webserver,
- dem Filigrane-Server und
- dem Datenbank-Server.

Der Webserver stellt, zusammen mit dem Browser auf der Clientseite, das User-Interface dar. Dazu bedient er den Browser mit dynamisch erzeugten HTML-Seiten. Diese werden durch eine Kombination von Java Servlets und JavaServer Pages (JSP) auf dem Webserver erstellt.

Filigrane selbst ist der Application-Server, der den gesamten Ablauf der Anwendung auf der Providerseite steuert, wobei er die benötigten und anfallenden Daten aus der Datenbank bezieht und dort ablegt.

Der Filigrane-Server gliedert sich intern in verschiedene Services, die bestimmte Teilaufgaben der Server-Applikation übernehmen. Für den Demonstrator wurden :

- der Catalogue,
- der OfferService und
- der PackagingService

implementiert.

### Catalogue

Zur Auswahl des gewünschten Artikels existiert ein elektronischer Katalog, dessen Funktionalitäten durch die Catalogue-Komponente bereitgestellt wird.

### OfferService

Hat der Customer ein Produkt im Katalog ausgewählt, dient der OfferService zur Angebotsberechnung und -erstellung.

### PackagingService

Seine Aufgabe ist, das Produkt mit den Nutzungsbedingungen zu verpacken, zu verschlüsseln und den Schlüssel auf die SmartCard des Customers zu schreiben.

### 3.1.1 Datenstruktur

Die für den Demonstrator verwendete Datenbank nutzt die folgenden Tabellendefinitionen zur Darstellung seiner Daten. Die Darstellung ist verkürzt und beschränkt sich auf die erforderlichen Spalten:

- **User**, enthält die Nutzerinformationen

| Spalte | Datentyp | Bemerkung |
|---|---|---|
| user_ID | integer | not null, Primärschlüssel |
| name | varchar | not null, Name des Nutzers |
| credit | integer | not null, Guthaben des Nutzers |

- **Contract**, enthält die geschlossenen Verträge

| Spalte | Datentyp | Bemerkung |
|---|---|---|
| contract_ID | integer | not null, Primärschlüssel |
| product_ID | integer | not null, Fremdschlüssel(Product) |
| user_ID | integer | not null, Fremdschlüssel(User), Vertragspartner |
| contractText | varchar | not null, Vertragstext in SRDL |

- **Product**, enthält die Produktinformationen

| Spalte | Datentyp | Bemerkung |
|---|---|---|
| product_ID | integer | not null, Primärschlüssel |
| version | varchar | not null, Version des Produkts |
| description | varchar | not null Produktbeschreibung |

- **Licence**, enthält die abgegebenen Nutzungslizenzen

| Spalte | Datentyp | Bemerkung |
|---|---|---|
| licence_ID | integer | not null, Primärschlüssel |
| contract_ID | integer | not null Fremdschlüssel(Contract), zugrunde liegender Vertrag |
| user_ID | integer | not null, Fremdschlüssel(Käufer) |
| licenceText | varchar | not null, Lizenztext in SRDL |

## 3.2 Soll-Zustand

Wie schon in Abschnitt 2.4 erläutert, gibt es drei verschiedene Nutzergruppen:

- Producer
- Provider
- Customer

Der Producer produziert Güter und gibt diese, nach einer Qualitätskontrolle beim Quality Label Service, zum Weiterverkauf an den Provider ab. Dabei kommt zwischen beiden Nutzern ein Lizenzvertrag zu dem jeweiligen Produkt zustande. Dieser wird vom RCH registriert und gespeichert. Der Provider bietet diesen Artikel entsprechend den Lizenzarten auf seiner Filigrane-Plattform an. Ein anderer Provider kann für die angebotenen Güter eine Vertragslizenz schließen, wobei wieder ein Vertrag entsteht, der beim RCH gespeichert werden muß. Im Gegensatz dazu kann der Customer eine Nutzungslizenz für die angebotenen Produkte erwerben, was ebenfalls durch das RCH registriert wird. Im Hinblick auf die im vorangegangenen Kapitel gestellten Anforderungen unterteilen sich die Aufgaben des FCA in drei Anwendungsfälle (siehe UseCase-Diagramm in Abb. 3.1):

1. Verkaufen : Finanzielle Transaktion beim Kauf eines Produkts
2. Abrechnen : Abrechnung der angefallenen Lizenzeinnahmen
3. Konto laden : Aufladen des internen User-Kontos mit Hilfe elektronischer Zahlungsmittel

### 3.2.1 UseCase: Verkaufen

Dieser Anwendungsfall bezieht sich auf den Nutzungskauf durch den Customer. Dafür muß der FCA die Lizenzeinnahmen und -ausgaben buchen bzw.

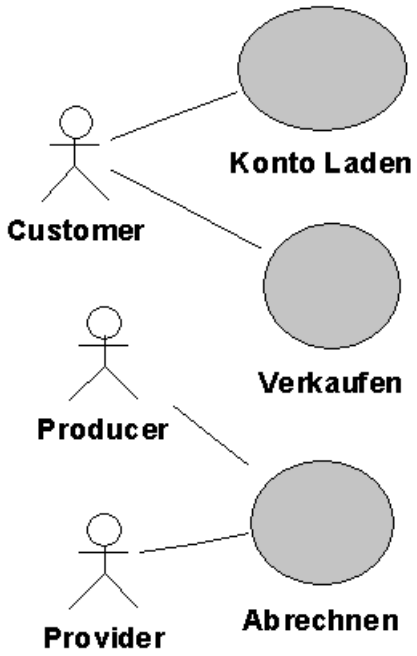


Abbildung 3.1: UseCases des FCA

das RCH und alle direkt und indirekt beteiligten Vertragsparteien über diesen Nutzungskauf informieren.
Der Ablauf wird teilweise im Demonstrator dargestellt. Der Nutzer wählt ein Produkt zum Kauf aus, läßt sich dafür ein Angebot erstellen und nimmt dieses Angebot an. Daraufhin muß die Nutzungslizenz, den Anforderungen nach, vom RCH signiert werden. Danach wird der Kaufbetrag vom Customerkonto auf das Providerkonto und die Lizenzabgabe vom Providerkonto auf das Konto des Vertragspartners gebucht. Anschließend wird der Download für den Customer erzeugt und der entsprechende Vertragspartner, sofern es ein Provider ist, über seinen Lizenzerlös informiert. Dieser Vorgang wird im Sequenzdiagramm in der Abb. 3.2 nach Abschluß des Vertrags zwischen Customer und Provider dargestellt.

Daraus ergibt sich ein weiterer untergeordneter Anwendungsfall für den FCA. Wird ein Provider über einen Lizenzerlös benachrichtigt, muß er ebenfalls die entsprechenden Beträge buchen und seinem direkten Vertragspart-

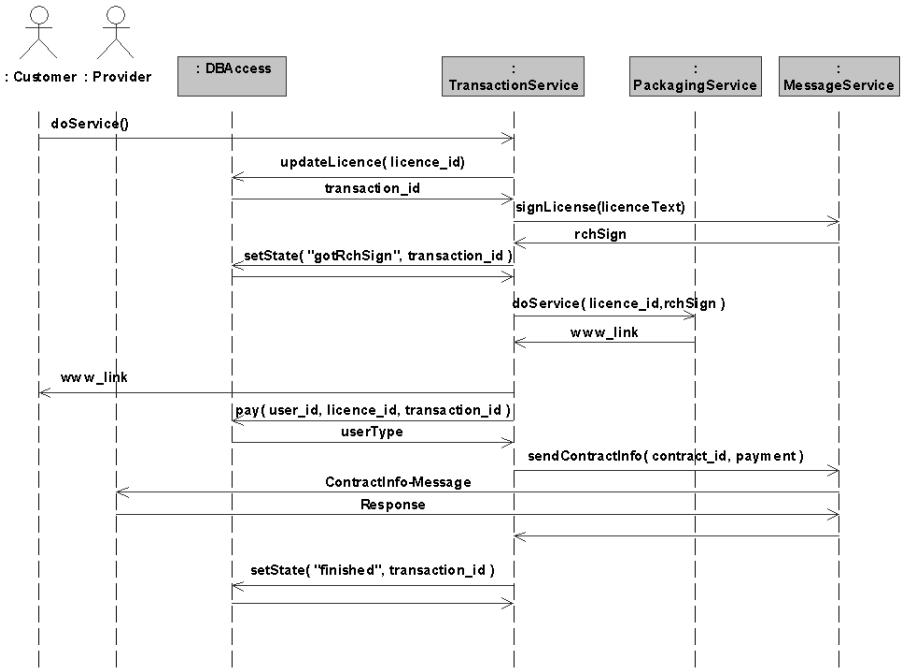


Abbildung 3.2: Transaktion nach Vertragsschluß

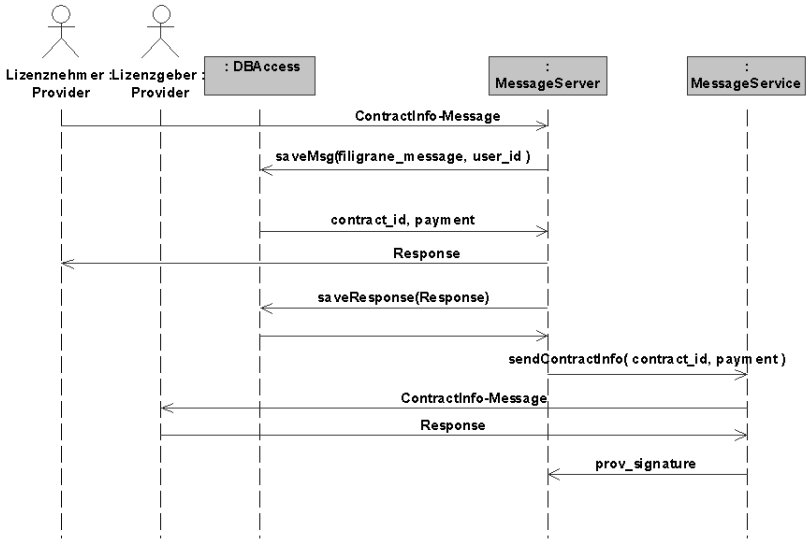


Abbildung 3.3: Empfang einer ContractInfo-Message

ner, bei entsprechendem Nutzertyp, darüber informieren. Die Abb. 3.3 zeigt das dazugehörige Sequenzdiagramm. Für dieses Szenario sind der Catalogue, der OfferService und der PackagingService bereits durch den Demonstrator implementiert. Als neue Komponente kommt der TransactionService hinzu, der die noch im Demonstrator fehlenden Funktionalitäten und den PackagingService in sich vereinigt.

### 3.2.2 UseCase: Abrechnen

Der Erlös eines Nutzungskaufs wird unter den Betroffenen entsprechend der geschlossenen Verträge aufgeteilt, d.h., daß nach dem Kauf eine finanzielle Transaktion zur Überweisung der Lizenzabgabe nötig ist. Da jede finanzielle Transaktion Kosten verursacht und die Preise oft Minimalbeträge annehmen, werden sie vorerst intern gebucht. Dazu haben Developer und Provider ein Konto, dessen Stand zum Zeitpunkt der Abrechnung auch die Rechnungssumme darstellt. Die jeweiligen Nutzer können ihre Einnahmen, auf direktem Wunsch, zu einem festgelegten Zeitpunkt oder ab Erreichen eines bestimmten Betrags abrechnen lassen. Der Nutzer erhält nach dem Abschluß der Transaktion eine Bestätigung von der Bank. Das dazugehörige Sequenzdiagramm findet sich in der Abb. 3.4.

Der Vorgang beginnt mit der Buchung des Rechnungsbetrages vom Konto des Nutzers auf das Providerkonto. Danach wird ein XML-basiertes EDI-Dokument erzeugt und an die Bank des Händlers gesendet. Die Bank schickt ihrerseits eine Empfangsbestätigung an den Provider zurück, der sie dann dem entsprechenden Vertragspartner zukommen läßt. Auch hier entsteht ein neuer untergeordneter Anwendungsfall. Wird für einen Provider abgerechnet, erhält dieser eine Kopie der Empfangsbestätigung der Bank, wie im Sequenzdiagramm der Abb. 3.5 veranschaulicht.

### 3.2.3 UseCase : Konto laden

Der Zahlungsvorgang soll, den Anforderungen nach, mit Hilfe elektronischer Zahlungsmittel stattfinden. Dabei wurde beim Entwurf auf der bereits be-

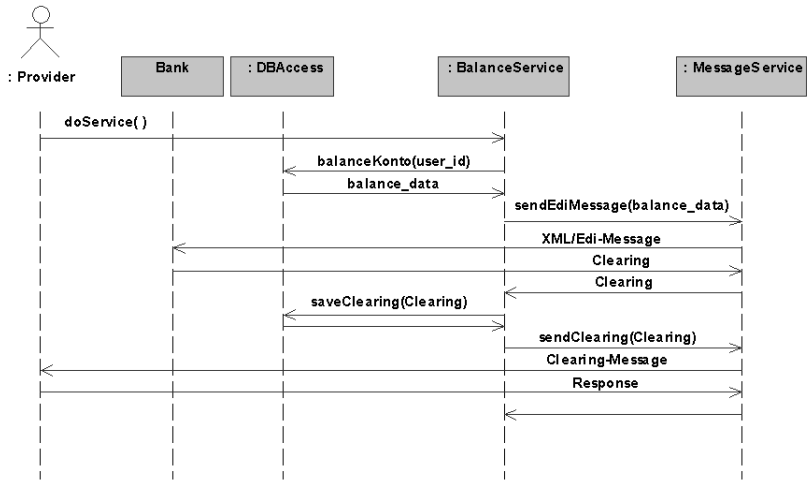


Abbildung 3.4: Abrechnen

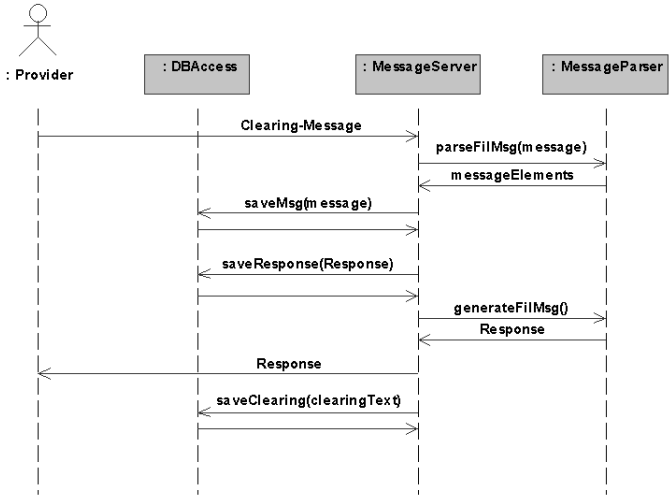


Abbildung 3.5: Empfang einer Clearing-Message

stehenden Struktur des Demonstrators aufgebaut. Dort hat der Customer intern ein Guthabenkonto, was bei jedem Kauf belastet wird. Ziel dieses Anwendungsfall ist es, das Guthabenkonto des Customers durch elektronische Zahlungsmittel aufzufüllen.
Der Customer übergibt Zahlungsmittel, Währung und den gewünschten Betrag. Nach nochmaliger Ausgabe dieser Werte bestätigt der Nutzer die Transaktion, wodurch die digitalen Werte zum Provider übertragen werden. Danach wird der Betrag in der Datenbank auf dessen Guthabenkonto gutgeschrieben und vom Providerkonto abgebucht. Dieser Vorgang wird im Sequenzdiagramm in der Abb. 3.6 dargestellt.

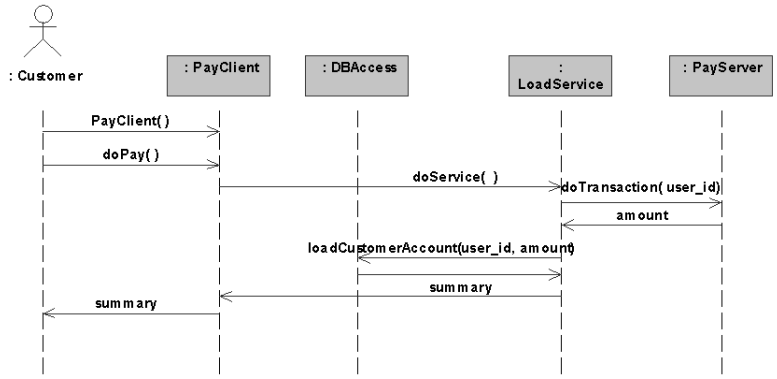


Abbildung 3.6: Konto Laden

### 3.2.4 Verfeinerung

Aus den hier dargelegten Abläufen innerhalb der einzelnen Anwendungsfälle resultiert das in Abb. 3.7 dargestellte verfeinerte UseCase-Diagramm. Daraus ergeben sich wiederum die später zu implementierenden Komponenten des FCA.

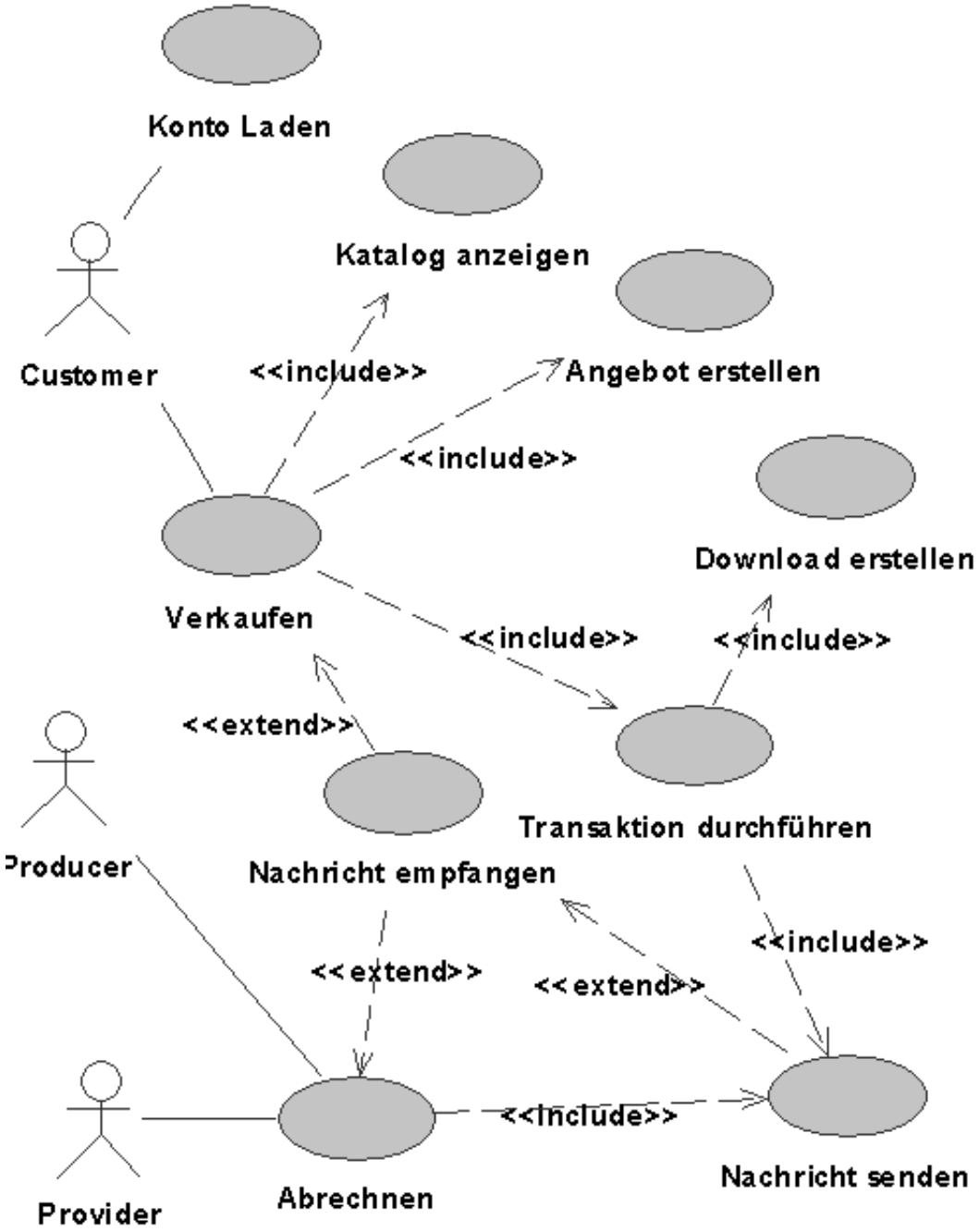


Abbildung 3.7: Verfeinertes UseCase-Diagramm

## 3.3 Datenstrukturen

Die folgenden Strukturen wurden anhand der vorher entworfenen Komponenten strukturiert. Dabei war zu beachten, daß diese konsistent sind und ihre Integrität den Anforderungen nach gewährleistet ist.

### 3.3.1 Kommunikation

Der FCA hat drei verschiedene Kommunikationspartner. In Abhängigkeit vom Kommunikationspartner werden verschiedene Nachrichtentypen verwendet. Da die Kommunikationsdaten in XML strukturiert werden, gibt es für jeden Typ einen eigenen DTD. Werden Informationen innerhalb von Filigrane ausgetauscht, wird die Filigrane-Message verwendet. Für die Kommunikation mit der Bank wird die EDI-Message genutzt.

#### XML/EDI-Message

Als Kommunikationsschnittstelle zur Abwicklung der finanziellen Transaktionen zwischen Provider und Bank dient EDI. Die entsprechenden Formulare werden in drei Gruppen unterschieden. Nach [cedi] gibt es Formulare für:

- Transaktionen,
- Informationen sowie
- Service und Sicherheit.

Aus diesem Spektrum sind aber nur die Formulare für Überweisungen, für Lastschriften und zur Informationsgewinnung relevant :

- PAYEXT (Extended Payment Order) = Überweisung
- DIRDEB (Direct Debit) = Lastschrift
- FINSTA (Finance Status) = Tagesauszug

Von den hier aufgezählten Dokumenttypen spielt nur die „Überweisung“ eine Rolle für die Implementierung. Die übrigen Formulare können aber für Weiterentwicklungen der EDI-Kommunikation in Filigrane genutzt werden. Für den Datenaustausch wird die EDI-Nachricht als XML-Dokument dargestellt und in dieser Form an die Bank gesendet. Die Daten werden entsprechend des DTDs der XEDI.ORG strukturiert. Der DTD verfolgt dabei die gleiche 3-teilige Strukturierung, wie sie in einem EDI-Dokument verwendet wird.
Der DTD gestaltet sich wie folgt:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!ELEMENT transactionSet (name,desc?,version*,(loop|segment)*)>
<!ATTLIST transactionSet
code CDATA #REQUIRED
>
<!ELEMENT loop ((loop|segment)*)>
<!ATTLIST loop
code CDATA #REQUIRED
>
<!ELEMENT segment (name,desc?,version*,element*)>
<!ATTLIST segment
code CDATA #REQUIRED
>
<!ELEMENT element (name,desc?,version*,value*)>
<!ATTLIST element
code CDATA #REQUIRED
req (M,O) "O"
width CDATA #IMPLIED
ref CDATA #IMPLIED
>
<!ELEMENT name #PCDATA>

<!ELEMENT desc #PCDATA>
```

```
<!ELEMENT version #PCDATA>

<!ELEMENT value #PCDATA>
<!ATTLIST value
code CDATA #REQUIRED
>
```

### Filigrane-Message

Die Filigrane-Message dient für den verteilten Objekten innerhalb des Rahmenwerks als Protokoll. Sie besteht aus vier Elementen :

- id, dient zur Identifikation der Nachricht
- type, definiert den Typ der Nachricht
- data, beinhaltet den eigentlichen Nachrichteninhalt
- signature, enthält die Signatur der Nachricht die über data erzeugt wurde

Der DTD gestaltet sich wie folgt:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!ELEMENT filigrane_message (id, type, data, signature+)>

<!ELEMENT id EMPTY>
<!ATTLIST id
  value CDATA #REQUIRED
>
<!ELEMENT type EMPTY>
<!ATTLIST type value
(rch_participant | Pwrk | Uwrk | contract_info | clearing | response)
```

```
#REQUIRED
>
<!ELEMENT data (#PCDATA)>

<!ELEMENT signature (#PCDATA)>
<!ATTLIST signature
  role (none | licensor | licensee) #REQUIRED
>
```

Die für den FCA relevanten Typen der Filigrane-Message sind:

- SignLicence-Message
- ContractInfo-Message
- Clearing-Message
- Response

### SignLicence-Message

Die SignLicence-Message entspricht dem Typ „Uwrk“ in der Filigrane-Message. Dieser Nachrichtentyp bezeichnet Nachrichten, die eine Nutzungslizenz zur Signierung beim RCH als Inhalt haben. Er besteht aus:

- Ver, definiert die verwendete SRDL-Version
- Lic, bezeichnet die Nutzungslizenz
- Soft, enthält das Produkt
- Condition, beschreibt die Nutzungsbedingungen

Der DTD gestaltet sich (in verkürzter Form) wie folgt:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!ELEMENT Uwrk (Ver, Lic, Soft, Condition)>

<!-- SRDL-VERSION -->
<!ELEMENT Ver EMPTY>
<!ATTLIST Ver
  ver_id CDATA #REQUIRED
>

<!-- LICENSE -->
<!ELEMENT Lic (Pwrk, Licor, Licee, Lid)>

<!-- MEDIUM -->
<!ELEMENT Soft EMPTY>
<!ATTLIST Soft
  sid CDATA #REQUIRED
  desc CDATA #IMPLIED
>

<!-- CONDITION -->
<!ELEMENT Condition (((Time, Lch) | Time | Lch), Fee?)>
```

### ContractInfo-Message

Die ContractInfo-Message entspricht dem Typ „contract_info“ in der Filigrane-Message. Dieser Nachrichtentyp bezeichnet Nachrichten, die einen Provider über seinen Lizenzerlös informieren sollen. Er besteht aus :

- contract_id, identifiziert den Vertrag
- payment, enthält den Zahlungsbetrag

Der DTD gestaltet sich wie folgt:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!ELEMENT contract_info (contract_id, payment)>

<!ELEMENT contract_id EMPTY>
<!ATTLIST contract_id
  value CDATA #REQUIRED
>

<!ELEMENT payment EMPTY>
<!ATTLIST payment
  value CDATA #REQUIRED
>
```

### Clearing-Message

entspricht dem Typ „clearing“ in der Filigrane-Message. Dieser Nachrichtentyp bezeichnet Nachrichten, die eine Abrechnung der Bank für einen Provider zum Inhalt haben. Sie besteht aus:

- RCH_id, identifiziert den Absender anhand seiner ID beim RCH
- Clearing_text, enthält die Empfangsbestätigung der Bank

Der DTD gestaltet sich wie folgt:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!ELEMENT clearing (RCH_id, Clearing_text)>

<!ELEMENT RCH_id EMPTY>
<!ATTLIST RCH_id
```

```
  value CDATA #REQUIRED
>

<!ELEMENT Clearing_text EMPTY>
```

**Response**

entspricht dem Typ „response“ in der Filigrane-Message und bezeichnet die Antwort des Nachrichtenempfängers. Sie besteht aus: Signature, was die Signatur der vorangegangenen Nachricht beinhaltet.
Der DTD gestaltet sich wie folgt:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!ELEMENT response (signature)>

<!ELEMENT signature EMPTY>
<!ATTLIST signature
  value CDATA #REQUIRED
>
```

### 3.3.2 Datenbank

Die Tabellenstruktur wurde anhand der bestehenden Daten des Demonstrators aufgebaut. Für die Aufgabe des FCA wurden einige Tabellen geändert bzw. hinzugefügt. Im folgenden Entity Relationship Diagramm (ERD) werden die Beziehungen der Tabellen untereinander veranschaulicht. Die nachfolgenden Tabellendefinitionen beschränken sich auf die Spalten, die für das Datenmodell notwendig sind. Für die finale Implementierung bietet es sich an diese Tabellen um weitere informative Datenfalder zu erweitern. Die Tabelle Product ist von den Änderungen am Demonstrator nicht betroffen und aus diesem Grund nicht mit aufgeführt.

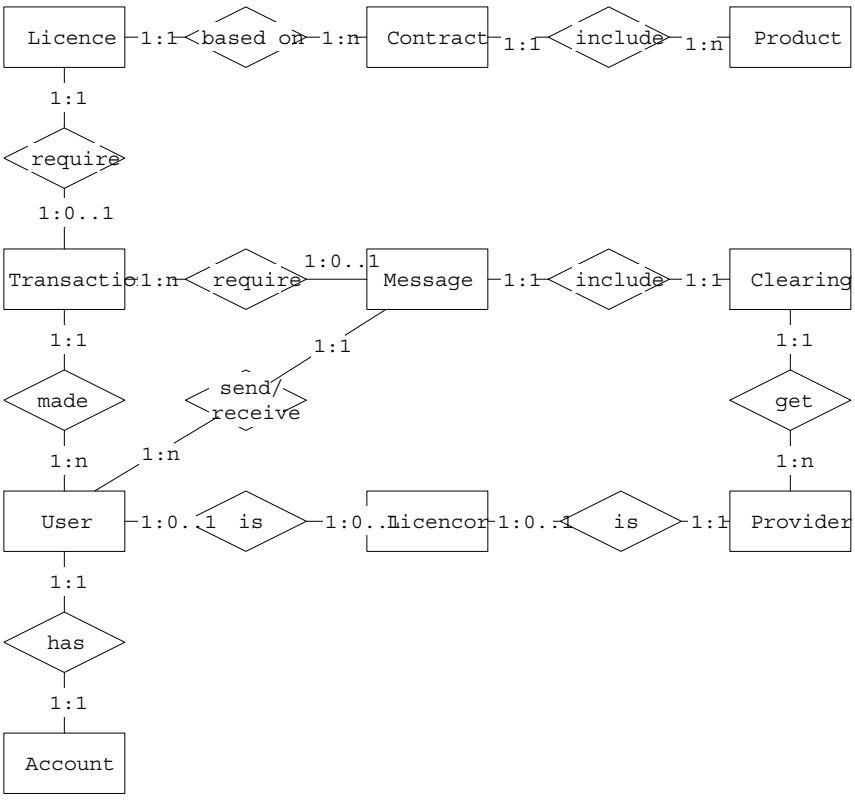


Abbildung 3.8: ERD nach Chen

- **Contract**, enthält die abgeschlossenen Verträge

| Spalte | Datentyp | Bemerkung |
|---|---|---|
| contract_ID | integer | not null, Primärschlüssel, wird vom RCH vergeben |
| product_ID | integer | not null, Fremdschlüssel(Product) |
| user_ID | integer | not null, Fremdschlüssel(User), ist der Vertragspartner |
| type | varchar | not null, Vertragstyp (angenommen/abgegeben) |
| contractText | varchar | not null, Vertragstext in SRDL |
| date | timestamp | default 'now' not null, Zeitstempel |

- **Licence**, enthält die abgegebenen Lizenzen

| Spalte | Datentyp | Bemerkung |
|---|---|---|
| licence_ID | integer | not null, Primärschlüssel |
| contract_ID | integer | not null, Fremdschlüssel(Contract), zugrunde liegender Vertrag |
| transact_ID | integer | Fremdschlüssel(Transaction), optional zugehörige Transaktion |
| licenceText | varchar | not null, Lizenztext in SRDL |
| price | integer | not null, Preis für die Lizenz |
| date | timestamp | default 'now' not null, Zeitstempel |

- **Transaction**, enthält die gestarteten Transaktionen

| Spalte | Datentyp | Bemerkung |
|---|---|---|
| transact_ID | integer | not null, Primärschlüssel |
| user_ID | integer | not null, Fremdschlüssel(User), Auslöser der Transaktion |
| type | varchar | not null, Transaktionstyp (Abrechnung oder Lizenzkauf) |
| state | varchar | not null |
| date | timestamp | default 'now' not null, Zeitstempel |

- **Message**, enthält die gesendeten und empfangenen Nachrichten

| Spalte | Datentyp | Bemerkung |
|---|---|---|
| message_ID | integer | not null, Primärschlüssel |
| user_ID | integer | not null, Fremdschlüssel(User) |
| type | varchar | not null, ein- oder abgehende Nachricht |
| request | varchar | not null, Nachrichtentext |
| response | varchar | not null, Antworttext der Nachricht |
| transact_ID | integer | Fremdschlüssel(Transaction), optional zugehörige Transaktion |
| date | timestamp | default 'now' not null, Zeitstempel |

- **User**, enthält die Nutzerinformationen

| Spalte | Datentyp | Bemerkung |
|---|---|---|
| user_ID | integer | not null, Primärschlüssel |
| name | varchar | not null |
| certificate | varchar | not null |
| type | varchar | not null, |

- **Licencor**, enthält Informationen über die Lizenzgeber

| Spalte | Datentyp | Bemerkung |
|---|---|---|
| user_ID | integer | not null, Primärschlüssel |
| rch_ID | integer | not null, ID beim RCH |
| bank | varchar | not null, Bankverbindung |

- **Provider**, enthält Adreßinformationen der Provider

| Spalte | Datentyp | Bemerkung |
|---|---|---|
| user_ID | integer | not null, Primärschlüssel |
| msgSrv | varchar | not null, Adresse des MessageServers |

- **Account**, enthält die Beträge der Nutzerkonten

| Spalte | Datentyp | Bemerkung |
|---|---|---|
| user_ID | integer | not null, Primärschlüssel |
| credit | float | not null, Kontostand |

- **Clearing**, enthält die Abrechnungen der Lizenzeinnahmen

| Spalte | Datentyp | Bemerkung |
|---|---|---|
| user_ID | integer | not null, Primärschlüssel |
| message_ID | integer | not null, Fremdschlüssel(Message) |
| clearing | varchar | not null, Abrechnung |
| date | timestamp | default 'now' not null, Zeitstempel |

# Kapitel 4

# Implementierung des Prototypen

Die aus der Modellierung resultierenden Komponenten werden in diesem Kapitel behandelt. Dabei wird ihr struktureller Aufbau und ihre Funktion innerhalb der Anwendung erläutert. Desweiteren werden Entscheidungen für die Implementierung begründet und die eingesetzten Techniken und Technologien kurz erklärt.

## 4.1 Überblick über die Struktur

Das verfeinerte UseCase-Diagramm und die in im vorangegangenen Abschnitt erläuterten Abläufe bilden die Basis für das in Abb. 4.1 dargestellte Klassendiagramm. Es enthält die für die Aufgabe des FCA zu implementierenden Klassen und zeigt deren Beziehungen untereinander.

#### TransactionService

Der TransactionService hat die Aufgabe, die dem Verkauf von Software folgende Transaktion durchzuführen. Dazu registriert er als erstes eine neue Transaktion in der Datenbank. Der Status der Transaktion wird mit jedem

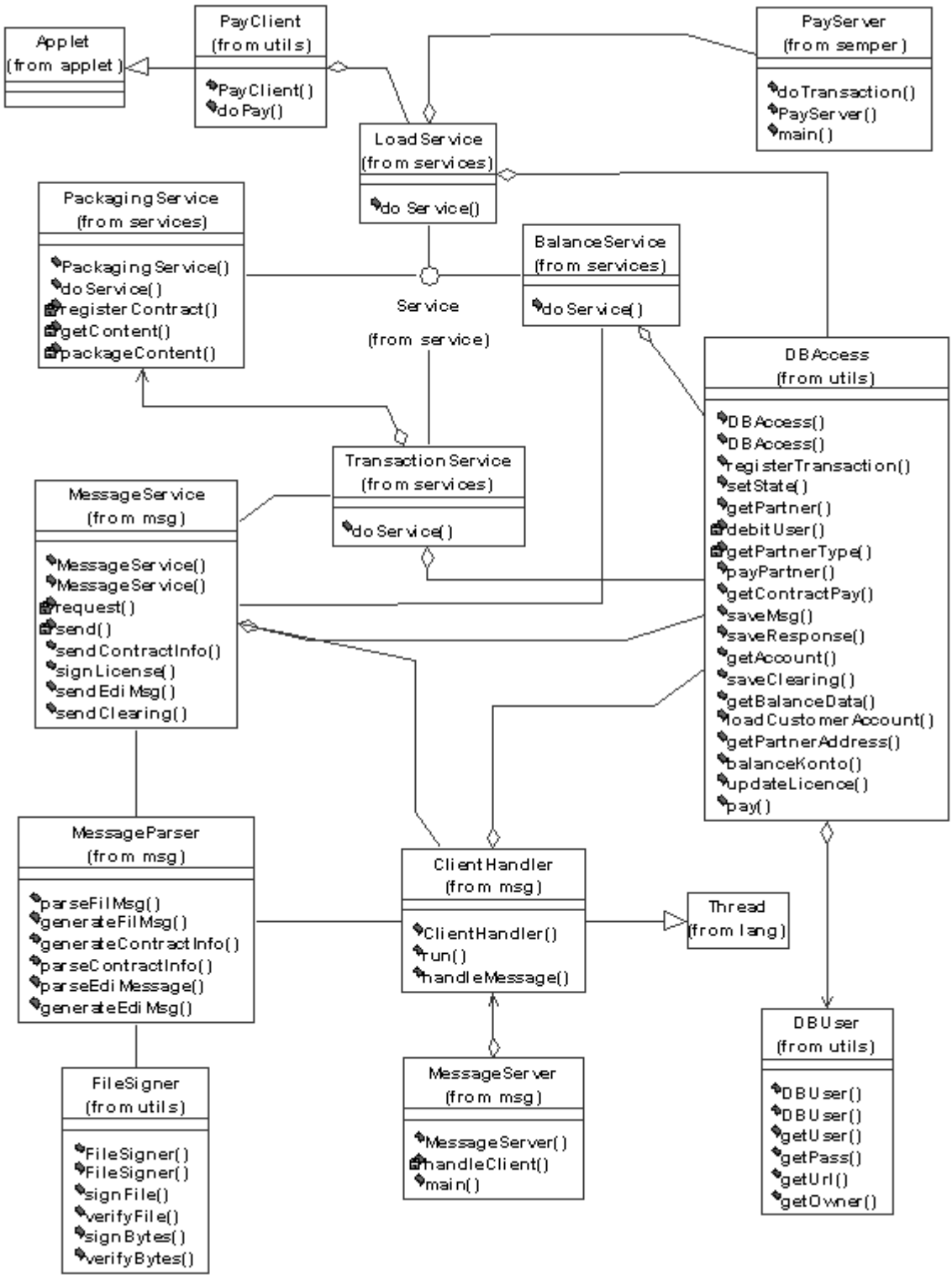


Abbildung 4.1: Klassendiagramm des FCA

Schritt aktualisiert, um die Transaktion nach einem Abbruch an der entsprechenden Stelle fortsetzen zu können. Nach dem Auslesen der Nutzungslizenz aus der Datenbank, wird sie zum signieren an das RCH gesendet. Nach Eingang der Signatur erstellt der PackagingService den Download. Anschließend werden der Kaufbetrag und die Lizenzabgabe gebucht. Ist der Lizenzgeber des Produkts ein Provider, wird eine ContractInfo-Message an den Vertragspartner gesendet. Da der TransactionService ein Service des Filigrane-Servers ist, implementiert er das Service-Interface.

### PackagingService

Da der PackagingService bereits implementiert wurde, mußte er nur an die Anforderungen angepasst werden, sodaß er die Signatur des RCH zusätzlich mit in die Jar-Datei verpackt.

### BalanceService

Seine Aufgabe besteht in der Abrechnung der Einnahmen von Developern und Providern. Nach dem Auslesen eines Events, zeitlich oder direkt durch den Nutzer, wird der BalanceService gestartet. Nach Buchung der Beträge in der Datenbank wird eine EDI-Message erstellt und an die Bank des Providers gesendet. Da der BalanceService ein Service von Filigrane ist, implementiert dieser das Service-Interface.

### LoadService

Der LoadService ermöglicht das Aufladen des Customerkontos mit Kreditheiten. Dazu nutzt er den PayServer als Zahlungsempfänger und den PayClient als zahlende Instanz. Ist der Transfer der Werte abgeschlossen, wird der Betrag auf dem Guthabenkonto des Customers gutgeschrieben.

### MessageCenter

Das MessageCenter beschreibt das Package „msg“. Seine Aufgabe besteht in der Abwicklung der gesamten Kommunikation mit verteilten Objekten innerhalb der Filigrane-Architektur bzw. außerhalb davon. Die Komponenten MessageService, MessageServer + ClientHandler, MessageParser und FileSigner sind entsprechend ihrem Aufgabengebiet in diesem Package zu finden.

### MessageService & MessageServer + ClientHandler

Die Aufgabe des MessageService ist es, alle Nachrichtenzu versenden. Beim Versenden speichert er die Nachricht zusammen mit der dazugehörigen Antwort in der Datenbank ab. Die entsprechende Gegenstelle des MessageService ist der MessageServer. Er übernimmt die Aufgabe des Nachrichtenempfängers. Der ClientHandler nimmt die Clientanfragen beim MessageServer entgegen und wird dadurch instanziiert. So existiert für jede Verbindung zwischen MessageService (Client) und MessageServer (Server) eine eigene Instanz des ClientHandler. Er speichert, wie der MessageService, alle eingehenden Nachrichten mit ihrer Antwort in der Datenbank ab.
Als Übertragungsprotokoll für die Nachrichtenübertragung dient SSL mit Client-Authentication.

### MessageParser

Diese Komponente des MessageCenters dient zum Auslesen der Informationen aus einer Nachricht. Dazu wird die Nachricht als String übergeben, woraufhin die Inhalte in einer Hashtable zurückgegeben werden. Ferner übernimmt diese Komponente die Erstellung der Nachricht. Dies erfolgt durch Übergabe des Inhalts in einer Hashtable, woraus der Parser eine Nachricht als String erzeugt.

#### FileSigner

Der FileSigner hat die Funktion, alle abgehenden Nachrichten zu signieren bzw. die Signaturen eingehender Nachrichten zu validieren.

#### DBAccess

Dieses Objekt repräsentiert die Schnittstelle zur Datenbank. Es stellt Methoden zur Verfügung, die die benötigten Daten im entsprechenden Format zurückgeben. DBAccess nutzt dazu den InterbaseClient-Treiber von Borland.

#### PayServer

Der PayServer wird zum Start von Filigrane initialisiert. Er ist der Empfänger der digitalen Werte. Nach Feststellung der zahlenden Gegenstelle und der Zahlungsoptionen wird ein Zahlungsverfahren gewählt und der Transaktionsbeginn abgewartet.

#### PayClient

Der PayClient stellt den zahlenden Endpunkt einer Zahlungstransaktion dar. Im Gegensatz zum PayServer wird er zur Laufzeit initialisiert. Ist das Zahlungsverfahren, durch Ermitteln der Gegenstelle und der Zahlungsoptionen gewählt, wird die Transaktion gestartet.

## 4.2 Implementierungsentscheidungen

### 4.2.1 Java

Als zugrundeliegende Programmiersprache wurde Java gewählt, da das vorhandene Teilsystem bereits in Java implementiert wurde. Außerdem bietet es eine mächtige Klassenbibliothek, welche die Implementierung und Anpassung

stark erleichtert. Im folgenden werden die für die Implementierung eingesetzten Teilbibliotheken des JavaTM Developer Kit (JDK) kurz erläutert.

### Java Secure Socket Extension (JSSE)

Diese Erweiterung des JDK ist eine Gruppe von Packages, die sichere Internetverbindungen auf Basis der Transportprotokolle SSL und TLS ermöglichen und wurde zur Sicherung der Kommunikation bei der Implementierung eingesetzt.

### Java Cryptography Extension (JCE)

Die Packages der JCE stellen ein Rahmenwerk und Implementationen zur Verschlüsselung, Schlüsselverwaltung und Message Authentication zur Verfügung. Es unterstützt symmetrische, asymmetrische, Block- & Streamverschlüsselung sowie sichere Streams und versiegelte Objekte.

### Java Database Connectivity (JDBC)

Die API des JDBC ist ein Teil der Klassenbibliothek des JDK. Diese Packages stellen Klassen und Methoden für den Zugriff auf SQL-Datenbanken und andere tabellarische Datenquellen zur Verfügung.

### JavaServer Pages und Java Servlets

JavaServer Pages (JSP) und Servlet sind Technologien des JDK, die auf der Serverseite die vom Client angeforderten HTML-Seiten dynamisch erzeugen. Dadurch können Inhalte bei bestehendem Layout zur Laufzeit in die Webseiten eingebunden werden.

### 4.2.2 Apache-Webserver

Für die Darstellung der dynamisch erzeugten Webseiten wird der Apache-Webserver der Apache Software Foundation (ASF) eingesetzt, da er die aktuellen Protokolle, inklusive HTTP/1.1 (RFC2616), unterstützt, für Windows NT/9x, Netware 5.x, OS/2 sowie die meisten Unix-Betriebssysteme verfügbar ist und seine Nutzung unentgeltlich ist. [apch] Er stellt außerdem eine API zur Integration von Modulen zur Verfügung. So können Drittanbieter durch die Programmierung und Integration eigener Module die Funktionalität des Servers erweitern.

#### Jakarta-Tomcat

Eine Projektgruppe der ASF ist Jakarta, welche das Modul Tomcat für den Apache-Webserver entwickelt und umgesetzt hat. Tomcat ist die offizielle Referenzimplementierung der Java-Technologien JavaServer Pages und Servlets, deren Spezifikationen von Sun erarbeitet wurden, weshalb es auch für Filigrane eingesetzt wird.

### 4.2.3 XML/EDI

Der Datenaustausch mit der Bank erfolgt auf Basis von EDI. Um eine leicht lesbare Form, der auszutauschenden Daten zu erreichen, werden sie in XML strukturiert. Dies geschieht mit Hilfe des DTDs der XEDI.ORG. Seine Struktur ermöglicht die Definition nahezu aller EDI-Transaktionssets mit wenigen XML-Elementen. Ein solches Dokument beschreibt das Transaktionsset und die dazugehörigen Daten.

# Kapitel 5

# Zusammenfassung

Abschließend wird in diesem Kapitel ein Überblick über die erreichten Ergebnisse gegeben. Weiterhin werden noch fehlende Funktionalitäten und mögliche Verbesserungen für zukünftige Weiterentwicklungen diskutiert.

## 5.1 Resümee

Im Rahmen dieser Diplomarbeit wurde der Aufgabenteil des FCA modelliert. Dadurch wird eine automatische Aufteilung der Lizenzeinnahmen möglich. Die beteiligten Parteien erhalten dabei ihren Teil der Einnahmen, anhand der vorher geschlossenen Verträge. Bis zum Zeitpunkt der Abgabe der Diplomarbeit konnte nur der Anwendungsfall „Verkaufen“ getestet werden, da die Implementierung noch nicht abgeschlossen ist.
Die Prüfung der weiteren Komponenten erfordert zusätzlich Partner aus dem Kreditwesen, die sich bis zum Abschluß der Dimplomarbeit noch nicht gefunden haben. Nur so kann sichergestellt werden, daß der EDI-Datenaustausch mit der Bank korrekt funktioniert. Idealerweise könnte dieser Partner digitale Werte zum Test des eingesetzten Zahlungsystems zur Verfügung stellen.
Bei der Entwicklung des Modells wurden zusätzlich Komponenten, wie das MessageCenter, geschaffen, deren Funktionalitäten auch bei anderen Abläufen innerhalb der Filigrane-Architektur benötigt werden, beispielsweise zur Registrierung eines neuen Vertrags beim RCH.

Die Entscheidung für ein Zahlungssystem, wurde vorerst außer acht gelassen. Die Struktur der SEMPER-Klassen erlaubt eine Implementierung der Zahlungskomponente unabhängig vom eingesetzten Verfahren.

## 5.2 Ausblick

Ein Teil der hier erarbeiteten Lösung läßt sich auch für andere Anwendungen nutzen. Die gekapselten Funktionsteile, wie z.B. die EDI-Funktionalitäten des MessageParsers, lassen sich anhand der definierten Schnittstellen leicht in neue Umgebungen integrieren.
Eine mögliche Verbesserung des entstandenen Entwurfs ist, die indirekte Zahlung über das Guthabenkonto durch direkte Zahlung zu ersetzen. Dadurch wird der Anwendungsfall „Konto laden“ überflüssig und der Customer zahlt zum Leistungszeitpunkt. Zusätzlich können spezielle Eigenschaften und Funktionen der eingesetzten Zahlungssysteme für den Nutzer verfügbar gemacht werden.
Weiterhin können die Möglichkeiten von EDI besser genutzt werden. Durch die Integration weiterer EDI-Nachrichten wären dann z.B. Kontoinformationen abrufbar. Werden die EDI-Funktionalitäten des MessageParser mit in das RCH integriert, ergeben sich weitere Möglichkeiten, die Einhaltung der Verträge zu überwachen.
Desweiteren ist zu überlegen, inwieweit auf Angriffe bei der Kommunikation zwischen den Komponenten zu reagieren ist. Dazu müssen Strategien entwickelt werden, die solche Konflikte, unter Beachtung der Server- und Netzlast, der Performance und der Datensicherheit, lösen.
Diese Arbeit vervollständigt die Umsetzung des Filigrane-Rahmenwerks um einen weiteren Baustein. Nach der endgültigen Fertigstellung, wird Filigrane seinen Teil zur Vereinfachung und Sicherung des Handels, vorallem mit Software und medialen Inhalten, im Internet beitragen.

# Anhang A

# Glossar

**ANSI-X12** - Beschreibt die Geschäftsdokumente für EDI.

**Application-Server** - Ist ein Server, der die Aufgabe der Anwendung auf der Serverseite im Internet übernimmt.

**Certification Authority** - siehe Zertifizierungsinstanz

**Hashtable** - Ist ein Container, in welchem Objekte ueber eine Schlüssel identifiziert werden.

**Hashwert** - Ist ein Wert fester Länge, der durch einen Algorithmus (Hash-Funktion) für jede beliebige Information einen eindeutigen, reproduzierbaren Wert liefert.

**Komponente** - Besteht aus ein oder mehreren Objekten, die zusammen eine bestimmte Aufgabe erfüllen.

**Markup** - Bezeichnet die Formatierungs- oder Strukturierungssprachen, die mit der computergestützten Textverarbeitung entstanden sind.

**Message Digest** - siehe Hashwert

**Private-Key** - Der private Schlüssel eines Schlüsselpaares, das bei asymmetrischen Verschlüsselungsalgorithmen Anwendung findet.

**Public-Key** - Der öffentliche Schlüssel eines Schlüsselpaares, das bei asymmetrischen Verschlüsselungsalgorithmen Anwendung findet.

**RFC2616** - Definiert das HTTP-Protokoll in der Version 1.1.

**Secret-Key** - Der geheime Schlüssel, der bei symmetrischen Verfahren Anwendung findet.

**Session-Key** - Ist ein Secret-Key, der für die Dauer einer Kommunikationsverbindung zur Sicherung der Daten verwendet wird.

**SmartCard** - Eine Plastikkarte mit einem elektronischen Chip, der Daten verarbeitet und speichert.

**SQL** - Ist eine Sprache zur strukturierten Abfrage von Datenbanken.

**Tag** - (dtsch. Etikett) Beschreibt die Elemente eines Dokuments.

**XEDI** - Umsetzung der Darstellung von EDI-Dokumenten mit XML

**XML/EDI-Translator** - Übersetzer, für EDI nach XML und umgekehrt

**Zeitstempel** - Eine Zeitangabe, die mit hoher Genauigkeit den Zeitpunkt der letzten Änderung einer Datenstruktur beschreibt.

**Zertifizierungsinstanz** - Genehmigung und Vergabe von Zertifikaten und Zertifzierungsstellen.

# Literaturverzeichnis


[guth] http://www.mathematik.uni-kl.de/ guthmann/was.htm

[lehn] http://www.ssw.uni-linz.ac.at/Teaching/Lectures/Sem/2000/Lehner/

[ec99] Nabil R. Adam, Oktay Dogramaci Aryya Gangopadhyay, Yelana Yesha. Electronic Commerce - Technical, Business and Legal Issues. Prentice Hall, Upper SaddleRiver, NJ, 1999

[dedig] http://www.dedig.de/Dedig/01-info-angebot/05-ec_und_recht/ec_edi-gesetze.asp

[tu-b] http://ig.cs.tu-berlin.de/da/041/Kapitel1.htm

[ct95] Dirk Fox, Digitale Signaturen, Heise Verlag, c't 10/95, S. 278

[fsnw] http://firstsurf.de/news/07-02-09_f.htm

[xedi] http://www.xedi.org/pdf/XEDI_Tech_Paper.pdf

[us00] Uwe Sandner, Architektur und Implementierung von Internetdatenbanken - Ausarbeitung, TU-München

[oxml] Robert Eckstein (Übersetzung: Nik Klever), XML - kurz & gut, 1. Auflage O'Reilly Köln 2000

[ecin] http://www.ecin.de/edi/

[ukai] http://www.iww.uni-karlsruhe.de/IZV4/Infoseiten/

[cedi] http://www.commerzbank.de/firmen/service/edifact/prozess.html

[sdfr] Gérard Lacoste, Birgit Pfitzmann, Michael Steiner, Michael Waidner; Final report of Project SEMPER, Springer-Verlag, Heidelberg, 2000

[apch] http://www.apache.org

[fil12] M. Jalali, L. Den Hollander, G. Hachez, P. Vannel, C. Vasserot; Deliverable 1.2 - Filigrane System Specification, Darmstadt, 1999