JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis

e-Billing
Entwicklung und Einführung in der Praxis

MAGISTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

in der Studienrichtung

INFORMATIK

Eingereicht von:
Wolfgang Schlapschy Bakk.techn., 0155555

Angefertigt am:
Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM)

Betreuung:
Priv.-Doz. Mag. iur. Dipl.Ing. Dr. Michael Sonntag

Wien, Mai 2008

Danksagung

An dieser Stelle möchte ich mich bei einigen Personen bedanken, welche mich bei dieser Magisterarbeit, beziehungsweise auch während meines gesamten Studiums unterstützt und begleitet haben.

Zuerst möchte ich mich bei den Mitarbeitern des Instituts für Informationsverarbeitung und Mikroprozessortechnik (FIM) für die hervorragende Betreuung während des Studiums und insbesondere während meiner Auslandsaufenthalte bedanken. Besonderer Dank gebührt hier natürlich meinem Diplomarbeitsbetreuer Herrn Priv.-Doz. Mag. iur. Dipl.Ing. Dr. Michael Sonntag, welcher vor allem in juristischen Belangen immer erste Anlaufstelle für Fragen war.

Großer Dank gilt auch dem Verein AustriaPro, und hier insbesondere Frau Alexandra Sladek und Herrn Mag. Lothar Winkelbauer, welche mir die Teilnahme am e-Billing-Arbeitskreis ermöglicht haben. Darüber hinaus möchte ich mich bei allen Interviewpartnern bedanken, besonders auch bei Herrn Franz Gepp von der Firma A-Trust, der mir zum Testen einen e-Billing-Kartenleser zur Verfügung stellte.

In erster Linie möchte ich mich aber bei meiner Familie und vor allem bei meinen Eltern bedanken, die mir durch Ihre Unterstützung ein sorgenfreies Studium ermöglichten. Ihnen und ihrem Betrieb ist es auch zu verdanken, dass diese Diplomarbeit einen sehr hohen Praxisbezug aufweist, und ich bin mir sicher, dass die entwickelten Applikationen dort sinnvoll eingesetzt werden können. Des Weiteren bedanke ich mich bei meiner Freundin Mag. (FH) Edith Putschögl für ihre Unterstützung und bei meinem Cousin David Selig für das Korrekturlesen der Arbeit.

Nicht vergessen möchte ich auch meine engsten Studienkollegen, Dipl.Ing. Gernot Inthaler, Dipl.Ing. Georg Möstl, Dipl.Ing. Klaus Pesendorfer, Dipl.Ing. Stefan Preuer und Manuel Wallnöfer, denen ich für die ausgesprochen gute Kameradschaft, Ihre Hilfe und Ihren Ansporn danken möchte.

Kurzfassung

In einer von Automatisierung und elektronischem Datenaustausch geprägten Wirtschaft ist die Rechnungsstellung eine der letzten Bastionen des Papiers. Obwohl durch einen großflächigen Einsatz der e-Rechnung, vor allem im B2B-Bereich, beeindruckende Einsparungen erzielt werden könnten, ist die elektronische Rechnung in Österreich immer noch eine Ausnahmeerscheinung. Dies mag vielleicht auch daran liegen, dass der Gesetzgeber an eine auf elektronischem Weg übermittelte Rechnung ganz besondere Anforderungen stellt.

Diese Diplomarbeit versucht die elektronische Rechnung ganzheitlich zu beleuchten. Zuerst wird ein Überblick über die geltenden Gesetze sowie mögliche zukünftige juristische Entwicklungen gegeben.

Danach wird gezeigt, wie sowohl Rechnungssteller als auch Rechnungsempfänger durch den Einsatz der e-Rechnung profitieren können.

Eine Fallstudie zeigt die Einführung der elektronischen Rechnung in einem kleinen Unternehmen. Dabei wird auch ein einfaches im Zuge der Diplomarbeit entwickeltes e-Billing-System sowie die dabei verwendeten Technologien vorgestellt.

Abstract

Although our economy is dominated by automation and electronic data exchange, there are still some workflows based on the exchange of paper documents. One of these is the billing process. Although the whole economy, and especially the B2B sector, could enormously benefit from a wide adoption of the electronic invoice, todays daily work is still dominated by the paper bill. This may be due to the fact that Austrian law requires an electronic invoice to meet very stringent requirements.

This diploma thesis aims at taking a holistic view at the e-billing processes. In the first part the legal background is examined and possible future developments are presented.

The second part shows how companies can benefit from the adoption of the electronic invoice.

A case study finally illustrates the adoption of e-billing in a small business. In the course of this case study a tailor-made e-billing system along with the technologies used is presented.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Listings

Abkürzungsverzeichnis

Abs	Absatz
AK	Arbeitskreis
API	Application Programming Interface
Art	Artikel
ASN.1	Abstract Syntax Notation One
B2B	Business to Business
B2C	Business to Consumer
B2G	Business to Government
BMF	Bundesministerium für Finanzen
BMWA	Bundesministerium für Wirtschaft und Arbeit
CN	Common Name
DBMS	Datenbankmanagementsystem
DER	Distinguished Encoding Rules
DN	Distinguished Name
e-Rechnung	elektronische Rechnung

EDI	Electronic Data Interchange
EJB	Enterprise Java Beans
ERP	Enterprise Resource Planning
FIBU	Finanzbuchhaltung
FTP	File Transfer Protocol
GUI	Graphical User Interface
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union
JEE	Java Enterprise Edition
JNI	Java Native Interface
KMU	Kleine und mittlere Unternehmen
lit	Litera (Ziffer)
NIST	National Institute of Standards and Technology
OOP	Objektorientierte Programmierung
OR-Mapping	Objektrelationales Mapping
PEM	Privacy Enhanced Email
PIN	Personal Identification Number
RDBMS	Relationales Datenbankmanagementsystem
RTR GmbH	Rundfunk und Telekom Regulierungs-GmbH
SigG	Signaturgesetz

SigRL	Signaturrichtlinie
SigV	Signaturverordnung
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SQL	Structured Query Language
SSCD	Secure Signature Creation Device
SSL	Secure Socket Layer
TLS	Transport Layer Security
USt	Umsatzsteuer
UStG	Umsatzsteuergesetz
VPN	Virtual Private Network
WKÖ	Wirtschaftskammer Österreich
XSLT	Extensible Stylesheet Language Transformation
Z	Ziffer
ZDA	Zertifizierungsdiensteanbieter

1 Einleitung

1.1. Motivation

Die Informationstechnologie hat die Wirtschaft in den letzten Jahrzehnten nachhaltig verändert. Geschäftsprozesse wurden umgestaltet und werden nun computerunterstützt und automatisiert bearbeitet. Effizienz ist das Schlüsselwort für die globalisierte Wirtschaft.

Zeit- und damit kosteneffizient kann heute aber nur mehr gearbeitet werden, wenn wiederkehrende Geschäftsprozesse möglichst gut automatisiert werden. Den im Zuge dieser Prozesse ausgetauschten Daten kommt dabei besondere Bedeutung zu. Eine Grundvoraussetzung für erfolgreiche Automatisierung ist, dass möglichst viele dieser Daten auf elektronischem Weg ausgetauscht werden. Sind Daten in einem System in elektronischer Form gespeichert und sollen diese in ein anderes System übertragen werden, so liegt es nahe, sie auch auf elektronischem Weg zu verschicken. Niemand würde z. B. auf die Idee kommen, ein digital gespeichertes Produktfoto auszudrucken, um dieses per Post an Geschäftspartner zu verschicken, damit diese es einscannen und auf ihrer Web-Präsenz präsentieren können. Dies würde einen Medienbruch darstellen, welcher Zeit und Geld kostet und zweifelsohne die Qualität der Daten vermindert.

Umso erstaunlicher ist es, dass ein Großteil der Rechnungen heute noch auf genau diese Art ausgetauscht wird. Abbildung 1.1 zeigt diesen Vorgang: Die Daten sind im System A gespeichert, welches daraus die Rechnung generiert. Danach wird diese ausgedruckt und per Post versendet, was natürlich zu Kosten führt. Bei diesem Vorgang geht semantische Information verloren, die dann beim Empfänger mühsam wieder rekonstruiert werden muss. Dort wird die Rechnung geprüft, und die Rechnungsdaten werden in einem teuren, zeitintensiven und fehleranfälligen Vorgang, entweder durch manuelles Abtippen der Daten oder durch Scannen der Rechnung, in das System B übernommen.

Nichtsdestotrotz ist der gerade geschilderte Prozess in österreichischen KMU heutzutage Realität. Eventuell können zwar durch den Einsatz eines Faxgerätes einige Teilprozesse automatisiert werden, den kostenintensiven Medienbruch kann man dadurch aber kaum

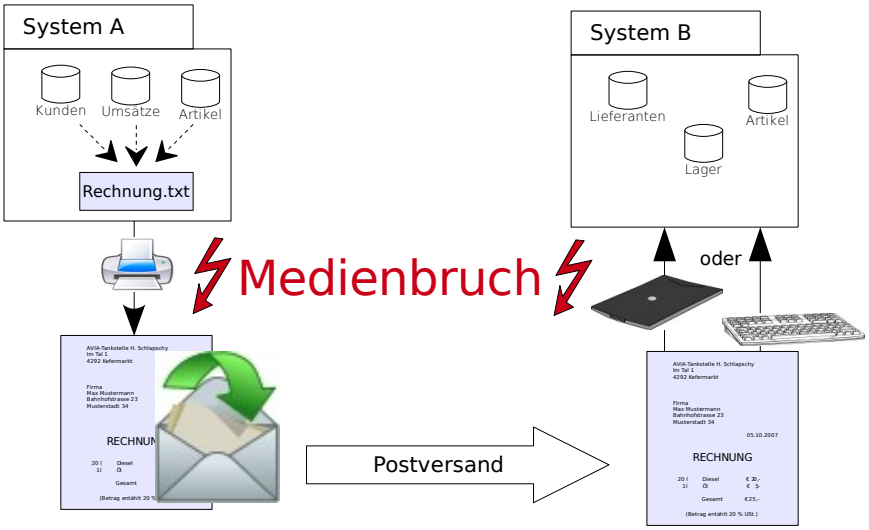

Abbildung 1.1.: Rechnungsaustausch mit Medienbruch

verhindern. Außerdem ist die Faxrechnung ein Auslaufmodell, welches vom Gesetzgeber zwar derzeit noch geduldet wird, aber schon seit längerer Zeit mit einem Ablaufdatum versehen ist.

Die Lösung ist, Rechnungen medienbruchfrei auf elektronischem Weg zu verschicken. Im Idealfall passiert dies wie in Abbildung 1.2 gezeigt: Die Daten sind im System A gespeichert, welches daraus die Rechnung generiert. Diese wird elektronisch versandt und beim Empfänger automatisiert geprüft. Die Rechnungsdaten werden automatisiert extrahiert und in das System B eingepflegt.

Der Austausch von elektronischen Rechnungen ermöglicht sowohl Rechnungssteller als auch Rechungsempfänger eine Automatisierung des Rechnungsmanagements und trägt so maßgeblich zur Kosteneffizienz bei.

Trotz dieser Vorteile hat sich die e-Rechnung in Österreich bisher kaum durchgesetzt. Die notwendigen rechtlichen Grundlagen (siehe „2. Rechtliche Sicht“ auf S. 17), um Rechnungen auf elektronischem Weg auszutauschen, wurden vom Gesetzgeber zwar schon im Jahr 2003 geschaffen, die e-Rechnung wurde aber von der Wirtschaft in dieser Form nicht weitläufig angenommen. „Abschnitt 2.2 – Probleme der derzeitigen Rechtslage“ zeigt Gründe für dieses Scheitern auf, während „Abschnitt 2.3 – Vorgeschlagene Änderungen“ einige Änderungsvorschläge des BMF darlegt, durch welche die elektronische Rechnung für die Wirtschaft an Attraktivität gewinnen soll.

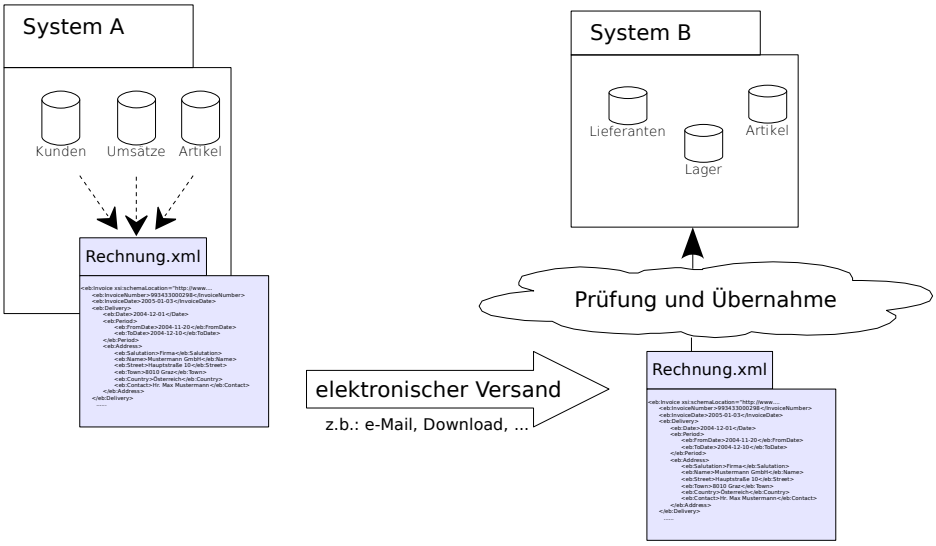

Abbildung 1.2.: Elektronischer Rechnungsaustausch ohne Medienbruch

Auch die WKÖ[1] ist maßgeblich daran beteiligt, den Einsatz der elektronischen Rechnung voranzutreiben. Grund für die Bestrebungen des BMF und der WKÖ sind die enormen Einsparungen, welche durch den Einsatz der e-Rechnung möglich sein sollen. Basierend auf Zahlen einer viel zitierten „Ovum"-Studie[2] aus dem Jahr 2003, könnten z. B. laut [45] jährlich 600 Millionen Euro Produktivitätsgewinn für die österreichische Volkswirtschaft realisiert werden, wenn 70 % der B2B-Rechnungen[3] in Österreich elektronisch ausgetauscht werden.

Aufgrund der möglichen Einsparungen und des wahrscheinlichen Auslaufens der Vorsteuerabzugsfähigkeit von Fax-Rechnungen[4] mit Ende 2008 ist damit zu rechnen, dass sich die elektronische Rechnung in nicht allzu ferner Zukunft verstärkt durchsetzen wird. Die Umstellung könnte für die einzelnen Betriebe aufgrund des sich aufbauenden Druckes bald zur betriebswirtschaftlichen Notwendigkeit werden. Es ist davon auszugehen, dass größere Unternehmen zuerst umstellen werden (oder schon umgestellt haben), und dass diese auch von ihren Geschäftspartnern die Ausstellung von elektronischen Rechnungen fordern werden. Dieser Druck ist laut [63] bereits heute offensichtlich:

[1] `http://www.wko.at`

[2] gemeint ist das Marktforschungsinstitut Ovum

[3] Es wird im Zusammenhang mit der e-Rechnung meist von B2B-Rechnungen ausgegangen, da Rechnungen an Endkunden für das Finanzministerium von geringerem Interesse sind, und deshalb nicht den selben strengen Auflagen unterworfen sind (siehe „1.3. Die Bedeutung der Rechnung im B2B-Verkehr" auf S. 12).

[4] Diese sollte ursprünglich schon Ende 2005 auslaufen. Die Gültigkeit wurde aber zuerst bis Ende 2006, dann bis Ende 2007 und zuletzt bis Ende 2008 verlängert [22].

„Große Händelshäuser verrechnen bis zu € 10,– (!) pro Rechnung, wenn der Lieferant der Ware die Rechnung nicht „formatskonform“ (EDIFACT) anliefert.“

1.2. Methodik und Gliederung

1.2.1. Allgemeines zur schriftlichen Ausarbeitung

Fremdwörter, Fachvokabular, Abkürzungen

Die Sprache der Informatik ist stark geprägt von englischen Begriffen. Aus diesem Grund wird darauf verzichtet, Begriffe künstlich ins Deutsche zu übersetzen. Sollte für ein englisches Wort eine gebräuchliche deutsche Übersetzung existieren, so wird diese auch verwendet. Ist das englische Wort aber auch im deutschen Sprachgebrauch verbreitet, so wird es gelegentlich synonym zum deutschen Wort verwendet, und es findet sich dafür ein Eintrag im Glossar.

Fachvokabular wird im Fließtext erklärt, falls dies der eigentliche Sinn des Abschnittes ist, indem das Wort vorkommt. Ansonsten findet sich ein Eintrag für das jeweilige Fachvokabel im Glossar.

Die volle Bedeutung der im Text verwendeten Abkürzungen kann im Abkürzungsverzeichnis nachgeschlagen werden.

Zitate

Bei sinngemäßen Zitaten, also dem Wiedergeben von Informationen Dritter in eigenen Worten, ist ein Verweis auf den Ursprung der Information angegeben.

Beispiel: Da die elektronische Rechnung von der Wirtschaft bisher nur sehr zögerlich angenommen wurde, hat das BMF nach neuen Lösungsmodellen gesucht und schlägt vor, die derzeitige Regelung durch die Modelle Bestätigung, FinanzOnline und Signatur zu ersetzten [33].

Wortwörtliche Zitate sind im Fließtext grundsätzlich kursiv, sowie unter Anführungszeichen geschrieben und eingerückt, sollten Sie eine gewisse Länge überschreiten.

Beispiel: [33] beurteilt die möglichen Einsparungen folgendermaßen:

> *„Eine wesentlichere Ersparnis als beim Rechnungsaussteller ist beim Empfänger zu erzielen, wenn die E-Rechnung in einer elektronisch weiter verarbeitbaren Daten-Struktur (z. B. ebInterface) übermittelt wird.“*

Lässt das Schriftbild z. B. bei zitierten Aufzählungen eine Einrückung, oder das Setzen von Anführungszeichen nicht zu, so wird das Zitat nur kursiv geschrieben.

Beispiel: Laut [33] zeigen die folgenden Punkte, dass sich die e-Rechnung bisher nicht durchgesetzt hat:

- *27,8 Prozent der Unternehmen wissen, dass man die elektronisch übermittelten Rechnungen für den Vorsteuerabzug digital signieren muss.*
- *Ein weiteres Drittel glaubt fälschlich, dass der Ausdruck der Rechnungen reicht.*
- *Das restliche Drittel ist sich diesbezüglich überhaupt unsicher.*

Namen von Gesetzestexten

Verweise auf Gesetzestexte sind in einer nichtproportionalen Schrift hervorgehoben.

Beispiel: `UStG § 11 Abs 2`

Codeausschnitte, Konfigurationsdateien, Beispieldokumente

Es wird versucht, Codeausschnitte sowie Auszüge aus Konfigurationsdateien und Beispieldokumenten im Text nur sehr sparsam zu verwenden. Namen von Klassen, Variablen oder Dateien sind im Fließtext in einer nichtproportionalen Schrift hervorgehoben.

Beispiel: `javax.persistence.EntityManager`

Methodensignaturen werden etwas kleiner geschrieben als der sie umgebende Fließtext.

Beispiel: Die Fabriksmethode getInstance(String algorithm, String provider) erzeugt Objekte.

Kurze Code-Beispiele werden vom Fließtext abgesetzt und grau hinterlegt dargestellt.

Beispiel:

```
1 Desktop desktop = Desktop.getDesktop();
2 desktop.browse(new URI(tmpInvoicePath));
```

Listing 1.1: Beispiel für ein Listing

Im Anhang finden sich einige Beispieldokumente, wie Rechnungen im ebInterface- oder PDF-Format oder auch Quelltext- und XML-Dateien. Größere Codeteile können im kommentierten Quellcode der Anwendung nachgeschlagen werden.

Die im Anhang dargestellten Listings (wie z. B. Listing A.2) verwenden keine Umlaute. So wird z. B. „Ä“ als „Ae“ ausgedrückt. Die Nichtdarstellbarkeit von Unicodezeichen ist eine Eigenheit des LaTeX-Packetes *listings*, mit welchem die Codeausschnitte in dieser Arbeit dargestellt werden. Umlaute in den Codeabschnitten mussten aus diesem Grund ersetzt werden.

1.2.2. Quellen und Aktivitäten

Die in dieser Diplomarbeit aufbereiteten Informationen und Erkenntnisse wurden aus verschiedenen Quellen und Aktivitäten, wie e-Billing-Veranstaltungen, Experteninterviews und natürlich der Implementierung der eigenen Anwendung gewonnen. Im Folgenden werden diese Quellen und Aktivitäten kurz vorgestellt:

Gesetzestexte Dies war der erste Schritt, um sich mit dem Thema sowohl auf nationaler, als auch auf europäischer Ebene vertraut zu machen.

Fallstudien Erfahrungen von Autoren diverser Fallstudien flossen in die eigene Arbeit mit ein.

Analyse potentieller Technologien Technologien zur XML-Verarbeitung oder zur digitalen Signatur wurden von verschiedenen Seiten beleuchtet. Dieser Schritt war enorm wichtig, nicht nur für die Implementierung des e-Billing-Systems, sondern auch, um die Herausforderungen und Probleme im Bereich der elektronischen Rechnung zu verstehen.

Besuch der e-Billing-Arbeitskreise Der e-Billing-AK ist in „Abschnitt 1.4 – Austria-Pro“ kurz beschrieben. Durch den regelmäßigen Besuch dieser Veranstaltung konnten Informationen aus erster Hand gewonnen werden, und die Standpunkte der verschiedenen Stakeholder im Bereich e-Billing verfolgt werden.

Besuch weiterer Veranstaltungen Zusätzlich zum Arbeitskreis wurden noch weitere Veranstaltungen wie z. B. eine ebTransfer-Schulung (siehe „1.4. AustriaPro“ auf S. 15) besucht, um weitere Kontakte zu knüpfen und Standpunkte kennen zu lernen.

Experteninterviews Um spezielle, tiefergehende Fragen, z. B. zur Zukunft der elektronischen Rechnung, zu klären, wurden mehrere Experteninterviews geführt. Tabelle 1.1 zeigt eine Übersicht über diese Experteninterviews, und die besprochenen Themenschwerpunkte. Obwohl zu den Interviews Tonbandaufnahmen existieren, wurde aus Platzgründen auf eine vollständige Wiedergabe der Gespräche verzichtet. Die aus den Interviews gewonnenen Informationen und Erkenntnisse finden sich aber natürlich in dieser Arbeit wieder. Zusätzlich sind an geeigneten Stellen Zitate und Auszüge aus diesen Gesprächen eingefügt.

Interviewpartner	**Position / Unternehmen**	**Themen**	**Dauer**[a]
Mag. Robert Kromer	Projektpartner AustriaPro	• Zukunft der elektronischen Rechnung • Format ebInterface	90 min
Mag. Erich Waldecker	BMF	• Mögliche Gesetzesänderungen	110 min
Mag. Lothar Winkelbauer	AK-Leiter e-Billing	• Zukunft der elektronischen Rechnung • Format ebInterface • e-Billing international	70 min 90 min[b]
Alexandra Sladek	Projektleiterin AustriaPro	• Zukunft der elektronischen Rechnung • Format ebInterface	80 min 30 min[b]
Franz Gepp	A-Trust	• Zukunft der elektronischen Rechnung • Digitale Signaturen	50 min 35 min[b]

[a] ungefähre Werte
[b] zwei Termine

Tabelle 1.1.: Experteninterviews

Fallstudie Hierbei wurden die traditionellen Abläufe bei der Rechnungsstellung, die Einsparungspotentiale sowie mögliche weitere Vorteile durch die elektronische Rechnung in einem kleinen Unternehmen analysiert.

Implementierung der Applikation Im Rahmen der Fallstudie wurde für den betrachteten Betrieb eine e-Billing-Applikation entwickelt, welche Rechnungen aus einem Altsystem importieren kann, und diese den Kunden zur Verfügung stellt.

1.2.3. Verschiedene Sichten

Nach eingehender Beschäftigung mit der elektronischen Rechnung haben sich für den Autor drei Sichten auf das Thema gebildet, die auch in der Gliederung dieser schriftlichen Arbeit wieder aufgegriffen werden. Es sei darauf hingewiesen, dass die folgenden Schilderungen den persönlichen Zugang des Autors zum Thema wiedergeben, und von der Auffassung anderer Personen abweichen können.

Abbildung 1.3 zeigt diese Sichten, deren Beziehungen untereinander im Folgenden beschrieben sind:

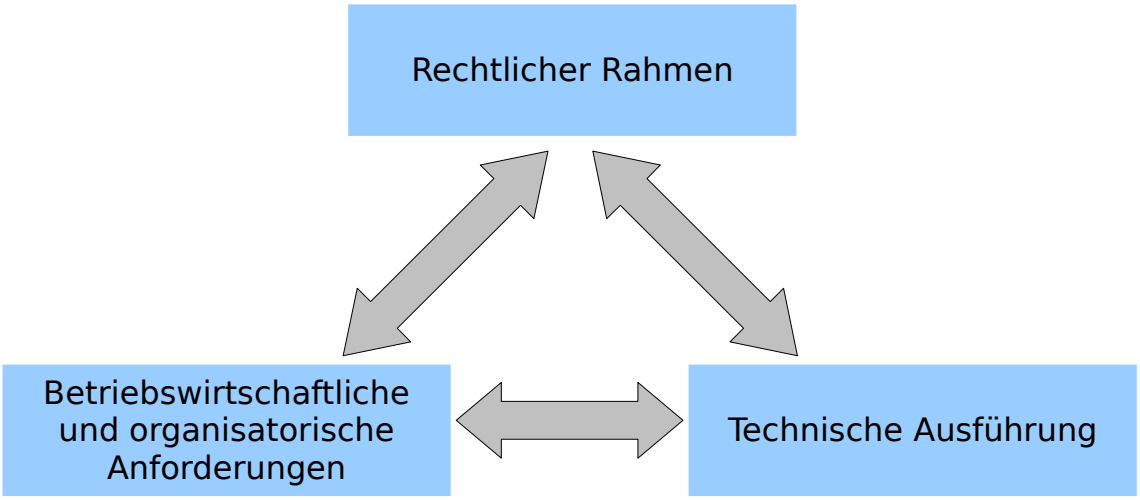

Abbildung 1.3.: e-Billing-Sichten

Der rechtliche Rahmen ist durch den Gesetzgeber vordefiniert und muss durch die anderen Sichten erfüllt werden. Im Verständnis des Autors sollte der Gesetzgeber aber darauf Rücksicht nehmen, dass die definierten Gesetze technisch praktikabel umsetzbar sind und die betriebswirtschaftlichen und organisatorischen Anforderungen erfüllen. Als negatives Beispiel kann hier ein Fall konstruiert werden, in dem der rechtliche Rahmen eine zu komplexe technische oder organisatorische Ausführung bedingt, sodass die betriebswirtschaftlichen Vorteile der elektronischen Rechnung nicht mehr zum Tragen kommen können. Als weiteres praktisches Beispiel könnte man hier die Situation in Österreich betrachten (siehe „2.2. Probleme der derzeitigen Rechtslage“ auf S. 27), wo der gesetzliche Rahmen zwar eine relativ einfache technische Ausführung zulässt, aber nicht zu 100 % konsistent zu anderen Gesetzen ist, und so den Betrieben nicht ausreichend Rechtssicherheit bietet. Als Folge eines solchen Ungleichgewichtes kann es passieren, dass die elektronische Rechnung von der Wirtschaft abgelehnt wird.

Die technische Ausführung muss also versuchen sowohl den rechtlichen Rahmen als auch die betriebswirtschaftlichen und organisatorischen Anforderungen zu erfüllen, wobei die Abhängigkeit vom rechtlichen Rahmen hier sicher als dominant einzustufen ist. Vereinfacht gesagt muss die technische Ausführung ein gesetzeskonformes e-Billing ermöglichen. Darüber hinaus sollte sie aber auch wirtschaftlich sein und sich bestmöglich in die Arbeitsabläufe der Unternehmen einfügen. Würde das Gesetz z. B. verlangen, dass jeder Mitarbeiter, der elektronische Rechnungen versendet, diese auch mittels eigener personalisierter Hardware bearbeiten muss (z. B. mit Hilfe einer SmartCard), so wäre dies zwar technisch machbar, aber wirtschaftlich und organisatorisch sehr aufwendig.

Die betriebswirtschaftlichen und organisatorischen Anforderungen werden einerseits durch den rechtlichen Rahmen, der selbst natürlich organisatorische Anforderungen definiert (z. B. dass eine Rechnung archiviert werden muss, ...), andererseits aber auch durch die technische Machbarkeit eingeschränkt.

1.2.4. Gliederung

Die in „Abschnitt 1.2.3 – Verschiedene Sichten“ dargelegten Sichten finden sich auch grob in der Gliederung dieser schriftlichen Arbeit wieder.

Kapitel 1 - Einleitung

Kapitel 1 gibt zuerst einen kurzen Überblick über Potential und Probleme der elektronischen Rechnung (siehe „1.1. Motivation“ auf S. 1) und erläutert dann Methodik und Gliederung dieser Arbeit. „Abschnitt 1.3 – Die Bedeutung der Rechnung im B2B-Verkehr“ erklärt einige Begriffe des Steuerrechts und beschreibt die Rolle der Rechnung in der Wirtschaft. Dieser Abschnitt vermittelt dem Laien steuerrechtliches Basiswissen zum leichteren Verständnis der darauf folgenden Kapitel.

„Abschnitt 1.4 – AustriaPro“ widmet sich schließlich kurz dem Verein AustriaPro und seinen e-Billing-Aktivitäten.

Kapitel 2 - rechtliche Sicht

Kapitel 2 beginnt mit der Schilderung der derzeit gültigen Rechtslage (siehe „2.1. Derzeitige Rechtslage“ auf S. 17) sowohl aus nationaler als auch aus europäischer Sicht. In „Abschnitt 2.2 – Probleme der derzeitigen Rechtslage“ wird versucht, Gründe zu erarbeiten, warum die elektronische Rechnung ihren Siegeszug in Österreich noch nicht antreten konnte. Der Rest dieses Kapitels widmet sich den möglichen gesetzlichen An-

passungen, welche von Seiten der Administrative vorgenommen werden könnten, um die elektronische Rechnungslegung für österreichische Unternehmen attraktiver erscheinen zu lassen.

Kapitel 3 - Betriebswirtschaftliche Sicht – Fallstudie

Dieses Kapitel stellt zuerst den für die Fallstudie betrachteten Betrieb vor. Danach wird die betriebswirtschaftliche und organisatorische Sicht sowohl von einem allgemeinen Standpunkt als auch speziell für das gewählte Unternehmen analysiert. Diese Analyse beinhaltet Schätzungen der möglichen Einsparungen und beleuchtet weitere Vorteile, die durch eine Umstellung auf die elektronische Rechnung gewonnen werden können. Ebenso wird aufgezeigt, auf welche Arten e-Billing betrieben werden kann.

Kapitel 4 - Technische Sicht

Kapitel 4 behandelt die technische Sicht, wobei zu Beginn das entwickelte e-Billing-System überblicksartig vorgestellt wird. Die danach diskutierten technischen Konzepte werden zu diesem System in Bezug gestellt, in einigen Punkten geht die Diskussion aber auch deutlich über dieses hinaus. In diesem Kapitel wird auch eine sehr kurze Einführung in die Prinzipien asymmetrischer Kryptographie und digitaler Signatur gegeben (siehe „4.2.1. Einführung in die digitale Signatur“ auf S. 79). Dieser Teil ist für technische Laien gedacht, die sich schnell einen Überblick über die prinzipielle Funktionsweise der besprochenen Technologien verschaffen möchten. Darüber hinaus werden spezielle Themengebiete der digitalen Signatur, sowie verschiedene Rechnungsformate diskutiert (siehe „4.3. Rechnungsformate“ auf S. 103). Hierbei liegt der Schwerpunkt eindeutig auf dem Format ebInterface, welches in „Abschnitt 4.3.1 – ebInterface“ vorgestellt wird.

Der Rest dieses Kapitels widmet sich im Detail der entwickelten Applikation, und beleuchtet verschiedene technische Aspekte wie Konfiguration, Architektur und Erweiterungsmöglichkeiten.

Kapitel 5 - Fazit

Das letzte Kapitel zieht Bilanz über die im betrachteten Betrieb durchgeführten e-Billing-Aktivitäten, und beschreibt die dabei gemachten Erfahrungen. Abschließend werden die persönlichen Erfahrungen des Autors während der Arbeit dargelegt.

1.3. Die Bedeutung der Rechnung im B2B-Verkehr

Dieser Abschnitt gibt einen kurzen Überblick über die steuerrechtliche Bedeutung der Rechnung für Unternehmen. Die hier dargelegten Informationen sollen dem Laien nötiges steuerrechtliches Basiswissen vermittlen, um die später folgenden Details besser verstehen zu können. Die folgenden Ausführungen folgen neben dem UStG [21] vor allem auch Sekundärquellen wie [23] oder [75].

Wie der Name schon sagt, wird mit einer Rechnung eine gewisse erbrachte Leistung abgerechnet. Bei der Leistung kann es sich um eine Warenlieferung oder um eine Dienstleistung handeln. Der Leistende dokumentiert mit der Rechnung eine Forderung an den Empfänger. Sie ist allerdings nicht Voraussetzung für die Bezahlung, sondern konkretisiert die erbrachte Leistung, dient zur Information sowie als Zahlungsaufforderung.

Während die Rechnung im B2C-Verkehr vom Empfänger[5] meist nur dazu verwendet wird einen eventuellen Garantieanspruch geltend zu machen, besitzt die Rechnung im B2B-Verkehr eine viel wichtigere Bedeutung:

Eine dem UStG entsprechende Rechnung berechtigt den die Leistung in Anspruch nehmenden Unternehmer[6], der die Leistung im Rahmen seiner unternehmerischen Tätigkeit in Anspruch nimmt, dazu, sich die in Rechnung gestellte und gesondert ausgewiesene Umsatzsteuer als Vorsteuer abzuziehen.

1.3.1. Vorsteuerabzug

Eine korrekte Rechnung enthält fast[7] immer eine ausgewiesene Umsatzsteuer. In Österreich gibt es verschiedene Umsatzsteuersätze. Am gebräuchlichsten sind der 20 %-ige Steuersatz, sowie der ermäßigte Steuersatz von 10 %, welcher auf Lebensmittel oder kulturelle Veranstaltungen angewandt wird.

Laut § `1 Abs 1 UStG` unterliegen folgende Vorgänge der USt:

- *Lieferungen und sonstige Leistungen, die ein Unternehmer im Inland gegen Entgelt im Rahmen seines Unternehmes ausführt*
- *der Eigenverbrauch (Leistungen des Unternehmens, welche für den eigenen Bedarf konsumiert werden)*

[5] Damit ist im B2C-Verkehr meist eine natürliche Person gemeint, welche die Leistung nicht im Zuge einer unternehmerischen Tätigkeit in Anspruch genommen hat.
[6] Eine natürliche oder juristische Person.
[7] siehe dazu Tabelle 2.1

- *die Einfuhr von Gegenständen aus Drittlandsgebieten (Einfuhrumsatzsteuer)*
- *die Einfuhr von Waren aus der EU (Erwerbssteuer)*

[75] beschreibt die Beziehung zwischen Vorsteuer und Umsatzsteuer folgendermaßen:

> *„Die Umsatzsteuer wird auf jeder Wirtschaftsstufe einbehalten (z.B.: beim Produzenten, beim Hersteller, beim Groß- und Einzelhändler), wegen des Vorsteuerabzugs stellt sie jedoch innerhalb der Unternehmenskette keinen Kostenfaktor dar, sondern wird wie ein „durchlaufender Posten" behandelt."*

Dies beschreibt den Charakter der Umsatzsteuer als eine „Verbrauchersteuer", welche vom Endverbraucher bezahlt wird. Ebenso handelt es sich bei der USt um eine „indirekte Steuer", da sie nicht vom wirtschaftlich Betroffenen, sondern vom Unternehmer als Steuerschuldner abgeliefert wird, sowie um eine „Verkehrssteuer", da sie durch die Teilnahme am Leistungsaustauschverkehr ausgelöst wird. [72]

Der Prozess des Vorsteuerabzugs wird in [75] folgendermaßen beschrieben:

> *„Zuerst ermittelt der Unternehmer seine gesamte Umsatzsteuer aufgrund seiner Lieferungen und Leistungen, die er innerhalb eines bestimmten Zeitraumes (in der Regel ein Kalendermonat) an seine Kunden erbracht hat. Von dieser Summe werden dann die innerhalb dieses Zeitraums angefallenen Vorsteuern in Abzug gebracht. Das Ergebnis ist entweder eine Umsatzsteuerzahllast oder - bei einem Vorsteuerüberhang - ein Guthaben."*

Dieser Rechengang, also das Aufschlüsseln der Umsätze für die verschiedenen Steuersätze und das anschließende Abziehen der Vorsteuer, wird in der Umsatzsteuervoranmeldung festgehalten, welche dann an das Finanzamt übermittelt wird.

Laut [75] bestehen folgende Voraussetzungen für den Vorsteuerabzug:

- *Die der Rechnung zugrunde liegenden Lieferungen oder Leistungen müssen für das Unternehmen ausgeführt worden sein. Dies gilt als gegeben, wenn es zu einer mindestens 10 %igen betrieblichen Nutzung kommt. Bei gleichzeitiger privater Nutzung wird anteilsmäßig ein steuerpflichtiger Eigenverbrauch errechnet.*
- *Die Lieferung oder Leistung muss bereits ausgeführt worden sein.*
- *Die Rechnung muss bereits gelegt worden sein.*
- *Die Rechnung muss den Anforderungen des UStG entsprechen* (Anm. siehe auch Tabelle 2.1).

- *Die in Anspruch genommenen Leistungen müssen zur Ausführung steuerpflichtiger oder echt steuerbefreiter Umsätze*[8] *verwendet werden.*

1.3.2. Steuerprüfung

Im Zuge einer Steuerprüfung kontrolliert der Finanzbeamte meist die Berechtigung zum Vorsteuerabzug. Dabei wird das Vorhandensein und die Korrektheit von Rechnungen geprüft. Eventuell werden auch die beim Empfänger aufbewahrten Originalrechnungen mit den bei den leistenden Unternehmen gespeicherten Daten[9] verglichen.

1.3.3. Reverse-Charge

Eine Alternative zur derzeit praktizierten mehrphasigen Umsatzbesteuerung (siehe „1.3.1. Vorsteuerabzug“ auf S. 12) ist das sogenannte „Reverse Charge“-Modell. Die Grundidee dieses viel diskutierten Modells ist es, die Durchlaufcharakteristik der Umsatzsteuer zu beseitigen. Bei Reverse-Charge werden Rechnungen zwischen Unternehmen nur mehr „netto“[10] ausgestellt. Erst der Endverkäufer oder Dienstleister hebt vom Verbraucher die Umsatzsteuer ein und führt diese dann ab. Da es somit von Seiten der Unternehmen zu keinen Forderungen gegenüber dem Finanzamt kommen kann, argumentieren die Verfechter dieses Modells damit, dass dadurch dem Steuerbetrug wirksam vorgebeugt wird. Außerdem reduziere es die Kosten für den Verwaltungsapparat. [33]

In Österreich wird derzeit in der Bauwirtschaft nach dem Reverse-Charge-Modell abgerechnet. Im Zuge eines EU-Pilotprojekts möchte Österreich weitere Branchen auf diese Form der Umsatzsteuererhebung umstellen [25].

Wie jedoch Experteninterviews gezeigt haben, gibt es in Hinsicht auf dieses Modell anscheinend auch Befürchtungen, dass die Anzahl an „Scheinunternehmen“ bei einer Einführung von Reverse-Charge plötzlich sprunghaft ansteigen würde, und dass die Überprüfung der Leistungen[11], und das dann möglicherweise nötige Eintreiben einer Steuerschuld, mehr Kosten verursachen würden, als eingespart werden könnten.

Die Bedeutung der Rechnung könnte sich bei Einführung eines Reverse-Charge-Modelles grundlegend ändern – sie wäre dann nicht mehr Grundlage für einen Vorsteuerabzug. Dies könnte somit viele Probleme, welche heute für die e-Rechnung bestehen

[8] z. B.: Exporte an Drittländer

[9] Das leistende Unternehmen ist nicht dazu verpflichtet, Kopien der auf Papier ausgelieferten Rechnungen vorzuhalten, muss aber jederzeit dazu in der Lage sein, eine äquivalente Rechnungskopie zu erstellen.

[10] netto = nur das Entgelt ohne die Umsatzsteuer; brutto = Entgelt + Umsatzsteuer

[11] Also die Prüfung, ob die Leistungen oder Lieferungen wirklich im Rahmen einer unternehmerischen Tätigkeit konsumiert wurden.

(siehe „2.2. Probleme der derzeitigen Rechtslage“ auf S. 27) lösen, und das e-Billing vereinfachen.

1.4. AustriaPro

Bei AustriaPro handelt es sich um einen gemeinnützigen Verein im Umfeld der WKÖ. Als Teil seines „Mission Statements“ informiert AustriaPro über die Bedeutung und den Nutzen standardkonformer B2B-Datenaustauschprozesse für die Wirtschaft, und unterstützt deren Einführung und Verbreitung [15]. Der Verein versteht sich selbst als *„die B2B-Standardisierungsplattform innerhalb der Wirtschaftskammer Österreichs“* [15].

Einer der aktuellen Schwerpunkte von AustriaPro ist das Thema e-Billing. In diesem Bereich sind vor allem die folgenden Initiativen zu nennen:

Arbeitskreis e-Billing [28] fasst die Aktivitäten des Arbeitskreises folgendermaßen zusammen:

> *„Der AK e-Billing befasst sich mit technischen, rechtlichen und organisatorischen Belangen der elektronischen Rechnungsstellung zwischen Unternehmen (B2B) und zwischen Unternehmen und Staat (B2G).“*

Im Arbeitskreis treffen sich Softwarehersteller, Unternehmensberater, Vertreter der öffentlichen Verwaltung und verschiedene andere Stakeholder, um über die elektronische Rechnung zu diskutieren und zu informieren. Zum Zeitpunkt des Schreibens dieser Arbeit beschäftigte sich der AK e-Billing hauptsächlich mit den vom Bundesministerium für Finanzen geplanten Änderungen an der e-Rechnung.

ebInterface Ziel dieses von September 2004 bis September 2005 laufenden Projektes war die Entwicklung einer XML-basierten e-Billing-Standardschnittstelle zu – und zwischen – e-Billing-Systemen. Ebenfalls Teil des Projektes war die Implementierung dieser Schnittstelle durch die am Projekt teilnehmenden Softwareentwicklungsunternehmen in ihren ERP- und FIBU-Systemen. Das Ergebnis dieser Arbeit ist die ebInterface-Spezifikation (siehe „4.3.1. ebInterface“ auf S. 103).

ebInvoice Im Rahmen dieses Projektes setzten zwölf österreichische KMU, in Zusammenarbeit mit den am Projekt ebInterface beteiligten Softwareentwicklungsunternehmen, den Standard praktisch um, und konnten laut [10] *„so ihre Fakturierungs- und ERP-Prozesse optimieren.“*

ebCrossBorder Ziel dieses ursprünglich von April 2006 bis Februar 2007 laufenden und dann bis Ende 2007 verlängerten Projektes war es, die im Projekt ebInterface

geschaffene Spezifikation international zu etablieren, und so grenzüberschreitend nutzbar zu machen. Im Jänner 2008 wurde von AustriaPro der Endbericht zum Projekt ebCrossBorder publiziert [13]. „Abschnitt 4.3.1.1 – Internationalisierung des Standards“ fasst kurz die Ziele, Aktivitäten und Ergebnisse dieses Projektes zusammen, wie sie im ebCrossborder-Abschlussbericht beschrieben sind.

ebTransfer Dieses Projekt[12] zielte darauf ab, die Verbreitung der elektronischen Rechnung unter österreichischen KMU zu fördern. Dazu wurden in mehreren österreichischen Städten ganztägige Veranstaltungen abgehalten, bei denen ebTransfer-Berater ausgebildet wurden. Diese Schulungen konnten von den auszubildenden Beratern kostenlos besucht werden.

Bei der Telefit-Roadshow[13] 2007 wurden dann e-Billing-Beratungsschecks an die anwesenden Unternehmer ausgegeben. Solch ein Scheck berechtigte den Inhaber zu einer kostenlosen e-Billing-Beratung durch einen im Rahmen von ebTransfer geschulten Berater.

[12] `http://portal.wko.at/wk/format_detail.wk?StID=307002&AngID=1`.

[13] Eine Veranstaltung der WKÖ, bei der KMU über Neuerungen in der Informationstechnologie informiert werden.

2 Rechtliche Sicht

2.1. Derzeitige Rechtslage

Viele Unternehmen scheinen die Vorteile von elektronisch versandten Rechnungen bereits erkannt zu haben. So stellen bereits einige österreichische Unternehmen ihren Kunden Rechnungen in elektronischer Form zur Verfügung. Weniger bekannt ist allerdings, dass elektronische Rechnungen speziellen gesetzlichen Anforderungen genügen müssen, falls der Rechnungsempfänger einen Vorsteuerabzug geltend machen möchte. Dies führt dazu, dass ein großer Teil der heute in Österreich auf elektronischem Weg ausgetauschten Rechnungen wahrscheinlich nicht gesetzeskonform ist.

Im B2C-Bereich stellt dies auch kein wesentliches Problem dar, und dort sind solche Rechnungen auch an der Tagesordnung. So werden Rechnungen für Diskont-Handys z. B. nur auf elektronischem Weg zur Verfügung gestellt, ohne dass auf die besonderen gesetzlichen Anforderungen Rücksicht genommen wird. Diese Rechnungen sind trotzdem gültig und müssen auch bezahlt werden. Sobald es sich beim Rechnungsempfänger allerdings um ein Unternehmen handelt, gilt es zu beachten, dass dieser durch eine nicht gesetzeskonforme Rechnung sein Recht auf einen Vorsteuerabzug verlieren kann. Obwohl auch im e-Billing-Arbeitskreis Stimmen vernehmbar sind die meinen, dass das BMF hier derzeit noch nachsichtig sei, sollten weder Rechnungssteller noch Rechnungsempfänger dieses Risiko eingehen.

Die gesetzlichen Anforderungen an eine elektronische Rechnung sind in der `RL 2001/115/EG` über die mehrwertsteuerlichen Anforderungen an Rechnungen geregelt [9]. Diese europäische Richtlinie ändert die Richtlinie `77/388/EWG`, in welcher nur sehr wenige Anforderungen an die Rechnungsstellung festgelegt wurden. Somit konnten die Mitgliedstaaten die wichtigsten Pflichten bisher selbst bestimmen [9]. Ziel der Änderungen war es deshalb, innerhalb der EU einen Rechtsrahmen, sowie gemeinsame Modalitäten für die elektronische Rechnung zu finden.

Die RL `2001/115/EG` hält grundsätzlich fest, dass Rechnungen auf Papier oder – mit Zustimmung des Empfängers – auch elektronisch übermittelt werden können. Weiters

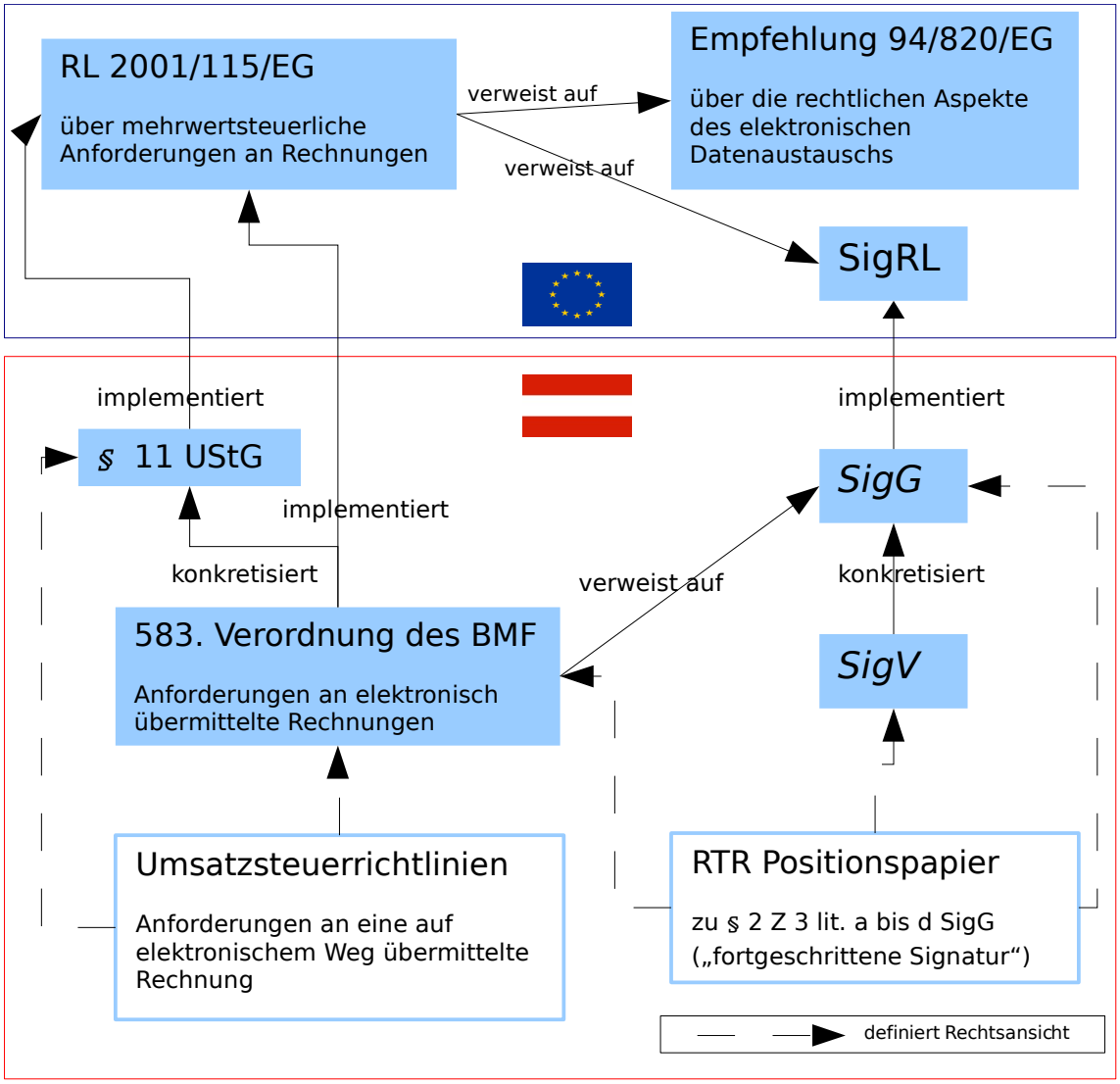

Abbildung 2.1.: Rechtsvorschriften

wird definiert, dass die Mitgliedstaaten eine elektronische Rechnung dann als Grundlage für einen Vorsteuerabzug akzeptieren, wenn

- Echtheit der Herkunft und
- Unversehrtheit des Inhalts

gewährleistet werden. Genau diese beiden Punkte werden auch vom österreichischen `UStG` in § `11 Abs 2` gefordert. Dieses definiert außerdem in § `11 Abs 1` die Bestandteile, welche eine gesetzeskonforme Rechnung in Österreich beinhalten muss.[1] Die notwendigen Bestandteile sind in Tabelle 2.1 aufgelistet.

[1] Diese gelten auch für Papierrechnungen.

Bestandteil	**< €150**	**-**	**> €10.000**
Name und Anschrift des liefernden/leistenden Unternehmens	+	+	+
Handelsübliche Bezeichnung	+	+	+
Menge	+	+	+
Tag/Zeitraum der Lieferung/Leistung	+	+	+
Entgelt für Leistung und Steuerbetrag	+	+	+
Entgelt für Leistung und Steuerbetrag getrennt	o	+	+
Anzuwendender Steuersatz	+	+	+
Name und Anschrift des Rechnungsempfängers	o	+	+
Ausstellungsdatum	o	+	+
Fortlaufende Nummer	o	+	+
UID-Nummer des liefernden/leistenden Unternehmens	o	+	+
UID-Nummer des Rechnungsempfängers	o	o	+
	+ benötigt o optional		

Tabelle 2.1.: Bestandteile einer Rechnung

Die `RL 2001/115/EG` hält außerdem fest, dass Rechnungen grundsätzlich nicht unterschrieben werden müssen. Abbildung 2.1 zeigt die in Verbindung mit der elektronischen Rechnung wichtigen Rechtsvorschriften und ihre Verbindung untereinander.

2.1.1. Rechnungssignatur

Echtheit der Herkunft und Unversehrtheit des Inhalts können laut `RL 2001/115/EG` und der daraus abgeleiteten Verordnung des Bundes[2] gewährleistet werden, wenn eine der folgenden Eigenschaften vorliegt:

- fortgeschrittene elektronische Signatur der Rechnung im Sinne von `Art 2, Z 2 der RL 1999/93/EG` (Signaturrichtlinie [8]).
  - Mitgliedstaaten können allerdings verlangen, dass die Signatur auf einem qualifizierten Zertifikat beruht, und von einer sicheren Signaturerstellungseinheit erstellt wird.[3] Diese Kombination kann zu einer sogenannten qualifizierten Signatur führen, verlangt wird diese allerdings nicht.[4]
- Austausch durch elektronischen Datenaustausch (EDI), wenn in der Vereinbarung über diesen Datenaustausch der Einsatz von Verfahren vorgesehen ist, welche die Echtheit der Herkunft und die Unversehrtheit der Daten gewährleisten.

[2] `583. Verordnung des BMF: Anforderungen an elektronisch übermittelte Rechnungen`
[3] Siehe dazu auch `SigRL Art 2, Z 6 und Z 10`, sowie „Abschnitt 4.2 – Digitale Signatur“
[4] Siehe dazu auch die Ausführungen zur Massensignatur in diesem Abschnitt.

– die Mitgliedstaaten können außerdem, unter von ihnen festzulegenden Bedingungen, den Austausch eines zusätzlichen, zusammenfassenden Dokumentes (Sammelrechnung) vorschreiben.

Das heißt, Rechnungen welche nicht über EDI übermittelt werden, oder EDI-Sammelrechnungen, sofern diese elektronisch übermittelt werden, müssen mindestens mit einer fortgeschrittenen Signatur versehen sein. Die `SigRL` [8] definiert eine elektronische Signatur als *„Daten in elektronischer Form, die anderen Daten beigefügt oder logisch mit ihnen verknüpft sind und die zur Authentifizierung dienen“*, und eine fortgeschrittene Signatur als eine elektronische Signatur, welche außerdem die folgenden Anforderungen erfüllt:

- *Sie ist ausschließlich dem Unterzeichner zugeordnet;*
- *sie ermöglicht die Identifizierung des Unterzeichners;*
- *sie wird mit Mitteln erstellt, die der Unterzeichner unter seiner alleinigen Kontrolle halten kann;*
- *sie ist so mit den Daten, auf die sie sich bezieht, verknüpft, daß eine nachträgliche Veränderung der Daten erkannt werden kann.*

Die `SigRL` wird in Österreich durch das `SigG` [1] umgesetzt. Dieses wurde mit 1. Jänner 2008 in einigen wichtigen Punkten geändert, und gleichzeitig wurde Anfang 2008 eine neue `SigV` verabschiedet. Eine Übersicht über die Zusammenhänge bietet auch Abbildung 2.1.

Als Ergebnis dieser Änderungen, gibt es nun auch im österreichischen `SigG` den Term „fortgeschrittene elektronische Signatur“, der dort in § `2 Abs 3 lit. a bis d` definiert ist, und die eben genannten Eigenschaften aus der `SigRL` besitzt. Vor der Änderung existierten diese Punkte ebenfalls, wurden aber nur indirekt über eine andere Quelle[5] als fortgeschrittene Signatur definiert.

Das österreichische `SigG` kannte vor 2008 nur eine sogenannte „sichere Signatur“, welche nun im geänderten `SigG` unter dem Namen „qualifizierte elektronische Signatur“ firmiert, und dort in § `2 Z 3a` definiert ist als:

- *eine fortgeschrittene elektronische Signatur, die auf einem qualifizierten Zertifikat beruht und von einer sicheren Signaturerstellungseinheit erstellt wird*

Anhang A.3 zeigt, was sich seit Anfang 2008 geändert hat.

[5] Im Erlass zur „Änderung der Umsatzsteuerrichtlinien – Anforderungen an eine auf elektronischem Weg übermittelte Rechnung“ in `RZ 1561`

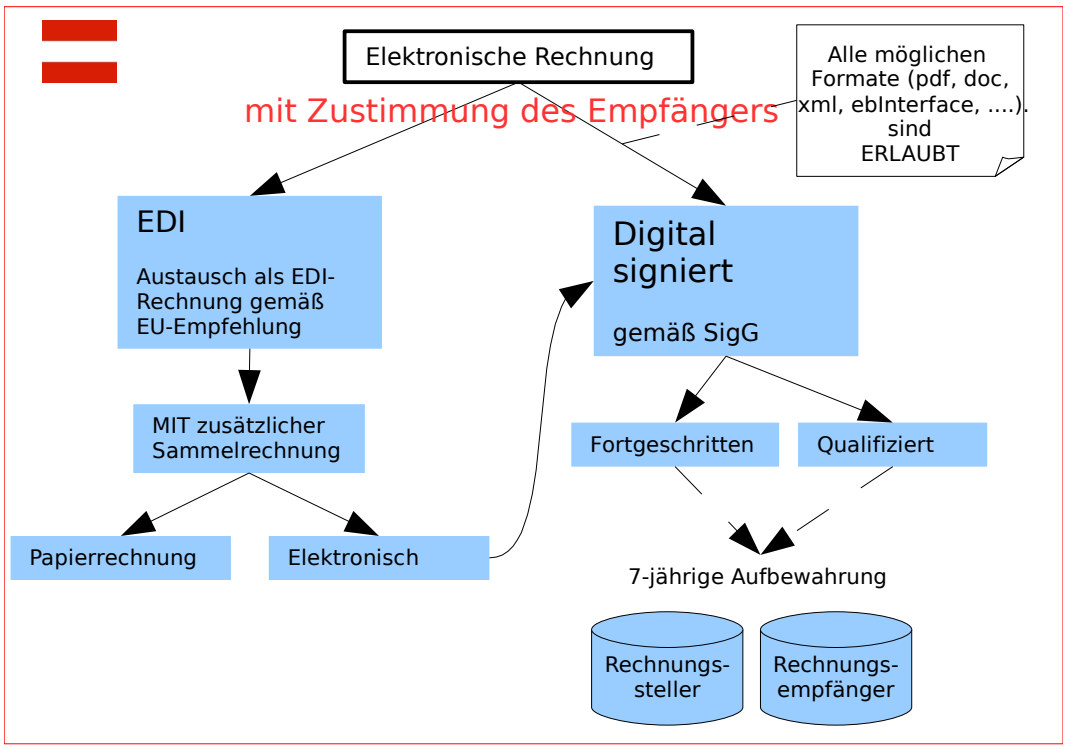

Abbildung 2.2.: Möglichkeiten zum elektronischen Rechnungsversand nach derzeitiger österreichischer Rechtslage

Die Begriffe im `SigG` und in der `SigRL` sind relativ technologieneutral gehalten. In der Praxis wird unter Signatur meist eine Form der asymmetrischen Kryptographie verstanden. Die Verbindung zwischen Signatur und asymmetrischer Kryptographie, sowie die Bedeutung von Zertifikaten und die Besonderheit von qualifizierten Zertifikaten wird in „Abschnitt 4.2 – Digitale Signatur“ behandelt.

In Österreich wird derzeit für elektronische Rechnungen nur eine fortgeschrittene Signatur verlangt. Der Verzicht auf ein qualifiziertes Zertifikat bringt in organisatorischer und technischer Hinsicht Erleichterungen mit sich. So gelten z. B. bei der Ausstellung eines Zertifikates für die fortgeschrittene Signatur nicht die selben hohen Anforderungen zur Identifikation des Zertifikatswerbers.

Viel bedeutender ist aber der Umstand, dass laut dem Positionspapier der RTR [53] eine fortgeschrittene Signatur

> *„nicht notwendigerweise mit einer sicheren Signaturerstellungseinheit (wie z. B. einer entsprechend geprüften und bescheinigten Chipkarte) erstellt werden muss.“*

§ 2 Z 3 lit. c SigG fordert zwar, dass die Signatur mit Mitteln erstellt wird, die der Unterzeichner unter seiner alleinigen Kontrolle halten kann. Der RTR [53] folgend bedeutet dies aber eben nicht, dass

> *„... spezielle Hardware (...) verwendet werden muss, aber es bedeutet, dass – insbesondere dann, wenn der private Schlüssel ausschließlich auf auslesbaren Datenträgern gespeichert wird – Sicherheitsmaßnahmen eingesetzt werden müssen, damit der Signator die Kontrolle über den Schlüssel halten kann (z. B.: Verschlüsselung der Datei, in welcher der private Schlüssel gespeichert ist, sowie Zugangs- und Zugriffsbeschränkungen zum Computer und zu dieser Datei).“*

Daraus folgt, dass der Signator entweder nicht auslesbare Datenträger benutzen muss, oder der ZDA ihn vertraglich zum Schutz eines auf auslesbaren Datenträgern gespeicherten Schlüssels verpflichten muss.

Abbildung 2.2 zeigt die Möglichkeiten, wie nach derzeitiger österreichischer Rechtslage elektronische Rechnungen ausgetauscht werden können.

2.1.2. Server- und Massensignatur

Der Umstand, dass elektronische Rechnungen nur fortgeschritten signiert werden müssen, erlaubt aus technischer und organisatorischer Sicht einen sehr praktikablen Umgang beim Signieren von Rechnungen. Größere Unternehmen, aber auch z. B. der im Rahmen der Fallstudie betrachtete Betrieb (siehe „3. Betriebswirtschaftliche Sicht – Fallstudie“ auf S. 39), versenden mehrere hundert oder auch tausend Rechnungen pro Monat, was effiziente Methoden zur Rechnungssignatur erfordert.

Dazu beschreibt [53] wie ein Server so konfiguriert werden kann, dass er eine große Anzahl von Signaturen für eine juristische oder natürliche Person erstellen kann. Hier ist zuerst zu beachten, dass [53] sich auf die alte Version von SigG und SigV bezieht. Darin gab es noch einen wichtigen Unterschied zur europäischen SigRL. [59] beschreibt dies folgendermaßen:

> *„Die SigRL definiert in Art 2 Z 3 einen „Unterzeichner“ allgemein als eine Person (strittig!), während das SigG in § 2 Z 2 einen Signator ausschließlich als natürliche Person, mit Ausnahme von Zertifizierungsdiensteanbietern, festlegt.“*

Dies hatte zur Folge, dass man ein Zertifikat und das damit verbundene Schlüsselpaar nicht auf einen Server oder auf eine juristische Person ausstellen konnte. Durch die Änderungen Anfang 2008 definiert das SigG einen Signator nun als:

> *„eine Person oder eine sonstige rechtsfähige Einrichtung, der Signaturerstellungsdaten und Signaturprüfdaten zugeordnet sind und die im eigenen oder fremden Namen eine elektronische Signatur erstellt“*

Dadurch sollte es nun wenigstens für fortgeschrittene Zertifikate[6] möglich sein, diese auf eine juristische Person auszustellen.

Aus praktischer Sicht war es aber auch schon vor 2008 möglich, einen Servernamen oder eine Firmenbezeichnung in ein Zertifikat aufzunehmen. Das `SigG` erlaubt ausdrücklich die Verwendung von Pseudonymen auch in qualifizierten Zertifikaten. Dadurch ist es laut RTR-Positionspapier [53] möglich, den Domainnamen eines Servers als Pseudonym des Signators in das CN-Feld eintragen zu lassen. Der Name des Signators muss also nicht mit dem im Zertifikat aufscheinenden Namen übereinstimmen. Ein Verweis auf eine juristische Person, und damit eine Vertretungsbefugnis, kann laut RTR-Positionspapier auf verschiedene Art und Weise kenntlich gemacht werden. Eine einfache Möglichkeit wäre es, den Hinweis in die zu signierenden Rechnungen selbst aufzunehmen. Häufiger wird jedoch der in „Abschnitt 3.6.3 – Auswahl eines Zertifikats“ beschriebe Fall zutreffen, dass der Name einer Firma im „organisationName“-Attribut („O“) und die Bezeichnung der Abteilung im „organisationUnitName-Attribut“ („OU“) des Zertifikates vermerkt sind. Eine dritte Möglichkeit wäre die Verwendung von Attributzertifikaten [53]. Wurde ein Zertifikat auf eine natürliche Person ausgestellt und enthält es eine Vertretungsmacht, so muss es widerrufen werden, falls der Signator das Unternehmen verlässt.

Da meist mehrere Rechnungen auf einmal signiert werden sollen, spricht man auch von einer Massensignatur. Laut RTR Positionspapier [53] gibt es dafür in der Praxis zwei Möglichkeiten:

Signatur in Echtzeit

Bei der serverseitigen Massensignatur in Echtzeit kann der Signaturserver wirklich automatisch eine große Anzahl von Signaturen in Echtzeit und ohne Willensakt des Signators auslösen. Dies kann – und wird auch meist – in Abwesenheit des Signators erfolgen. Dieser muss die Signaturerstellungsdaten nur aktivieren. Er muss allerdings zu jeder Zeit der Einzige sein, der wirklichen Zugriff auf den privaten Schlüssel hat, und er muss jederzeit dazu in der Lage sein, das Signieren von Rechnungen mit seinem Zertifikat zu unterbinden. Dazu sind beim Server geeignete Sicherheitsmaßnahmen zu treffen.

[6] Denn qualifizierte Zertifikate können weiterhin nur auf natürliche Personen ausgestellt werden (siehe „A.3. Änderungen im SigG und die neue SigV“ auf S. 158).

Solch eine Maschine kann Dokumente wie z. B. Ausgangsrechnungen vollkommen automatisch ohne Zutun des Signators signieren. Einige der am Markt erhältlichen Signaturserver arbeiten zum Beispiel als Mail-Proxy, oder werden über Web-Services angesprochen (siehe „3.6.2. e-Billing-Umsetzungsmöglichkeiten“ auf S. 59).

Fortgeschrittene Signaturen können auf die beschriebene Art und Weise von einem Signaturserver in Echtzeit erstellt werden. Für qualifizierte Signaturen ist dies so nicht möglich, da dafür sichere Signaturerstellungseinheiten (SSCD) verwendet werden müssen, und diese nach § `4 Abs 1 SigV` eine qualifizierte Signatur nur nach erfolgtem Willensakt auslösen dürfen. Dieser erfolgt normalerweise durch Eingabe eines PIN-Codes, oder durch Abgabe eines Fingerabdruckes. In Deutschland, wo zur Rechnungssignatur qualifizierte Zertifikate benötigt werden, hat man hier eine Vereinfachung geschaffen. Dort können sichere Signaturerstellungseinheiten durch eine PIN-Eingabe für eine gewisse Zeit freigeschalten werden, in welcher dann ankommende Dokumente auch ohne erfolgtem Willensakt signiert werden können. Dies könnte man als „fortgeschrittene Signatur mit qualifizierten Zertifikaten“ bezeichnen.

Sammeln und signieren

Sollen mehrere Rechnungen durch eine einzige PIN-Eingabe mit einer qualifizierten Signatur versehen werden, müssen diese zuerst gesammelt werden. In einem zweiten Schritt wird dann ein ganzer Stapel an Rechnungen vom Signator willentlich, z. B. durch Eingabe seines PIN-Codes, signiert. Wichtig ist, dass der Signator nicht jede zu signierende Rechnung einzeln lesen muss. Laut § `18 Abs 2 SigG` und § `4 Abs 1 SigV` muss es ihm aber wenigstens ermöglicht werden, dies zu tun. Ebenso muss ihm laut § `18 Abs 2 SigG` zum Zeitpunkt des Signierens die exakte Anzahl der zu signierenden Dokumente bekannt sein.

Auf diese Art könnte auch ein Signaturserver mit der aktiven Unterstützung des Signators sichere Signaturen erzeugen.

2.1.3. „Third Option“

Zusätzlich zu den bereits genannten Varianten EDI und Signatur (siehe „2.1.1. Rechnungssignatur“ auf S. 19) sieht die `RL 2001/115/EG` in `Art 2 Abs 2 lit. c` folgende Regelung vor:

> *„Die Rechnungen können vorbehaltlich der Zustimmung des betreffenden Mitgliedstaats oder der betroffenen Mitgliedstaaten auch auf andere Weise elektronisch übermittelt werden.“*

Dadurch ist es den Mitgliedstaaten erlaubt, alternative Übermittlungswege vorzusehen, was in einigen Ländern wie z. B. England auch gemacht wurde. In Österreich ist nach derzeitiger Rechtslage kein solcher alternativer Übermittlungsweg vorgesehen.

Ebenfalls in `Art 2 Abs 2 lit. c` dieser Richtlinie heißt es weiters:

> *„Die Mitgliedstaaten können den Steuerpflichtigen, die in ihrem Hoheitsgebiet Lieferungen von Gegenständen oder Dienstleistungen bewirken, keine weiteren Pflichten oder Formalitäten in Bezug auf die elektronische Übermittlung der Rechnungen auferlegen.“*

Dies lässt die Interpretation zu, dass eine dem `Art 2 Abs 2 lit. c SigRL` entsprechende EDI-Rechnung, oder eine auf Basis eines qualifizierten Zertifikates signierte Rechnung von den Mitgliedstaaten akzeptiert werden muss.

Abbildung 2.3 fasst die möglichen Arten des Rechnungsversandes laut Richtlinie `RL 2001/115/EG` in einer Übersicht zusammen.

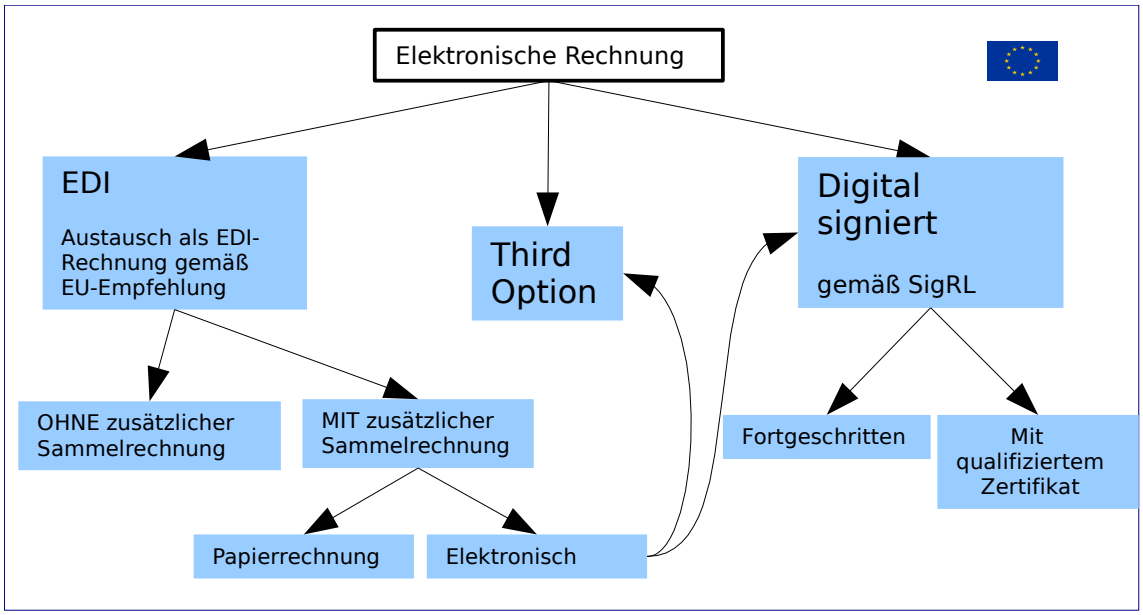

Abbildung 2.3.: Möglichkeiten zum Rechnungsversand nach RL 2001/115/EG

2.1.4. Aufbewahrung der Rechnungen

Elektronische Rechnungen müssen genauso lange aufbewahrt werden, wie ihre auf Papier gedruckten Pendants. Im Regelfall also für sieben Jahre. Im Gegensatz zur Pa-

pierrechnung, wo das Original der Rechnung nur vom Empfänger aufbewahrt werden muss, müssen elektronische Rechnungen auch vom Rechnungssteller archiviert werden. Bisher musste dieser beim Versand von Papierrechnungen nur in der Lage sein, diese später zu reproduzieren.

Wichtig ist hierbei, dass die archivierten Rechnungen die gesamte Aufbewahrungszeit über lesbar bleiben müssen. Dies betrifft nicht nur die Speichermedien, sondern auch die verwendeten Formate, was laut Meinung einiger Experten durchaus zu Problemen führen könnte. Archivierte Rechnungen sind der Finanzbehörde auf Verlangen auszuhändigen. `Rz 1561` der Umsatzsteuerrichtlinien [24] hält jedoch fest, dass es nicht zu beanstanden ist, *„wenn der Unternehmer als vorläufigen Nachweis einen Ausdruck der elektronisch übermittelten Rechnung vorlegt“.*

Die Gültigkeit der Rechnungssignatur muss vom Rechnungsempfänger geprüft werden. Bei einer ungültigen Signatur darf er die Rechnung nicht annehmen, sondern muss dies beim Rechnungssteller bemängeln. Eine Streitfrage ist, ob man auch den Nachweis für eine erfolgte Prüfung archivieren muss. Mitarbeiter des BMF haben in Vorträgen anklingen lassen, dass man dies dort gerne sehen würde. Eine rechtliche Grundlage scheint es dafür allerdings nicht zu geben. Außerdem ist an keiner Stelle ausgeführt, wie eine solche Dokumentation erfolgen sollte [47]. Die Art und Weise, wie dieser Nachweis z. B. von der „ebRe“-Applikation archiviert wird (siehe „4.3.1.5. Verifikation von ebInterface-Rechnungen“ auf S. 116), garantiert auf jeden Fall weder die Echtheit der Herkunft, und vor allem nicht die Unversehrtheit des Inhalts einer solchen Dokumentation.

Weiters ist zu beachten, dass der Begriff des Originals bei der elektronischen Rechnung etwas verschwommen ist, da natürlich jede Kopie der digitalen Rechnung vom Original ununterscheidbar ist. Andererseits darf es aber in einer anderen Hinsicht nur ein Original geben. Ein und dieselbe Rechnung einmal im PDF- und einmal im XML-Format zu signieren, würde zu zwei Originalrechnungen mit unterschiedlichen „Fingerabdrücken“ führen, und ist deshalb nicht erlaubt.

2.1.5. Zustimmung des Empfängers

Laut `RL 2001/115/EG` und § `11 UStG` muss der Empfänger dem Erhalt von elektronischen Rechnungen zustimmen. Der Umsetzungserlass zu den Anforderungen an elektronische Rechnungen relativiert dies jedoch und hält fest, dass diese Zustimmung keiner besonderen Form bedarf [24]. Stillschweigen und tatsächliches Praktizieren werden explizit als ausreichende Zustimmung aufgeführt.

Im Extremfall würde es also auch möglich sein, dem Kunden ohne Vorwarnung anstatt einer Papierrechnung eine elektronische Rechnung zukommen zu lassen. Wer-

den die Rechnungen weiterhin gezahlt und es ergeht auch kein Widerruf, so würde die e-Rechnung vorläufig als akzeptiert gelten. Verfügt man allerdings nicht über die Marktmacht eines internationalen Großkonzernes, empfiehlt es sich aber sicherlich einen partnerschaftlicheren Weg einzuschlagen.

2.2. Probleme der derzeitigen Rechtslage

Obwohl in Österreich schon im Jahr 2003 die rechtlichen Grundlagen geschaffen wurden, um Rechnungen gesetzeskonform und vorsteuerabzugsfähig auf elektronischem Weg auszutauschen, hat sich die elektronische Rechnung bisher nicht auf breiter Basis durchsetzen können. Laut einer von Marktagent.com im Auftrag der WKÖ bei österreichischen KMU durchgeführten Studie[7], *„wissen lediglich 27,8 Prozent der befragten Unternehmen, dass man die elektronisch übermittelten Rechnungen für den Vorsteuerabzug digital signieren muss“*. Laut der selben Studie, *„glaubt ein weiteres Drittel fälschlicherweise, dass der Ausdruck der Rechnung reicht, und das restliche Drittel ist diesbezüglich überhaupt unsicher“*.

Selbst wenn das notwendige rechtliche Basiswissen vorhanden ist, wird die notwendige digitale Signatur von vielen als zu kompliziert und aufwendig angesehen. Als Folge dessen wird entweder auf die gesetzlich vorgeschriebene Signatur verzichtet, oder es werden weiter Papier- beziehungsweise Faxrechnungen ausgetauscht. Aufgrund der geringen Verbreitung der elektronischen Rechnung hat das BMF auch die Gültigkeit der Faxrechnung bereits einige Male, das letzte Mal bis Ende 2008, verlängert.

Anlass für Kritik an den derzeitigen Anforderungen bieten vor allem die folgenden Umstände:

Unsicherheit für Rechnungsempfänger

Obwohl die in Österreich geforderte fortgeschrittene Signatur im Vergleich zur qualifizierten Signatur relativ einfach und unkompliziert zu handhaben wäre, birgt sie einen gewissen Unsicherheitsfaktor. Laut § `5 Abs 1 Z 1 SigG` muss ein qualifiziertes Zertifikat den Hinweis enthalten, dass es sich um ein solches handelt. Diese Eigenschaft kann beim Prüfen einer auf einem solchen Zertifikat basierenden Signatur erkannt werden. Ein Zertifikat für fortgeschrittene Signaturen, hier einfach ein fortgeschrittenes Zertifikat genannt, enthält keinen ähnlichen Vermerk, der es als fortgeschrittenes Zertifikat identifizieren würde. Aus der Sicht des Prüfenden unterscheidet es sich auf den ersten Blick eigentlich nicht von einem einfachen Zertifikat, wie sie ebenfalls von einigen

[7] 4. Quartal 2006 [12]

ZDA angeboten werden.[8] Nun ist der Rechnungsempfänger, sofern er sich die Vorsteuer abziehen möchte, aber dazu verpflichtet, eingehende Rechnungen zu prüfen. Ist eine Rechnung nicht gesetzeskonform, dürfte er diese nicht akzeptieren. Er muss also prüfen, ob eine Rechnung mit einer fortgeschrittenen Signatur versehen ist. Dies ist ihm allerdings nicht so ohne weiteres möglich, da sich diese Information aus dem Zertifikat nicht automatisiert erkennen lässt.[34]

Natürlich würde es hier Mittel und Wege geben, dieses Problem zu lösen. So kann ein fortgeschrittenes Zertifikat z. B. anhand des Zertifizierungspfades und der beim ZDA hinterlegten Informationen als solches identifiziert werden. Die dazu nötigen Informationen scheinen allerdings meist nur als Fließtext z. B. in der Certificate Policy des ZDA oder in Produktbeschreibungen auf und sind nicht automatisiert auswertbar. Ebenso wäre es denkbar, auch in fortgeschrittenen Zertifikaten ein Feld einzufügen, welches deren Qualität klarstellt.

Möchte man dieses Problem duch den Einsatz qualifizierter Signaturen lösen, so sollte bedacht werden, dass auch beim Vorliegen einer vermeintlich qualifizierten Signatur nicht mit absoluter Sicherheit festgestellt werden kann, dass es sich um eine solche handelt. Es ist dann zwar offensichtlich, dass für die Signatur ein qualifiziertes Zertifikat verwendet wurde, und technisch sollte es auch möglich sein, den Signator zur Verwendung einer SSCD zu zwingen, ob der Signator aber Software zur Signatur verwendet hat, die ihm z. B. die Anzeige der Dokumente ermöglicht hat, kann er nicht wirklich kontrollieren. Eine ähnliche Argumentation findet sich auch in der Stellungnahme der WKÖ zur Änderung des `SigG` [76], welche vorschlägt, Rechtswirkungen an die Qualität der Zertifikate und nicht an die Signatur zu binden, da dies besser prüfbar ist. Diese Unsicherheit bei der Prüfung qualifizierter Signaturen ist vor allem durch deren spezielle Rechtswirkungen problematisch, wäre bei der elektronischen Rechnung aber wohl vernachlässigbar.

Ein weiterer Unsicherheitsfaktor für den Empfänger ergibt sich durch dessen nicht näher spezifizierte Prüfungspflicht. Dieser Umstand ist in „Abschnitt 2.1.4 – Aufbewahrung der Rechnungen“ kurz beschrieben.

Signatur führt nur zum Signator

Der ebenfalls in „Abschnitt 3.6.2.1 – e-Billing auslagern“ beschriebene Umstand, dass die Signatur nur zum Signator und nicht zum Leistungsbringer führt, erklärt sich damit, dass Rechnungen auch durch Dritte signiert werden können. Der in der Rechnung angegebene Name des Leistungsbringers muss also nicht mit dem im Zertifikat auf-

[8] Zum Beispiel a.sign light von A-Trust.

scheinenden Namen übereinstimmen. Die Signatur hat dadurch laut einigen Kritikern keinen Mehrwert für den Rechnungsempfänger.

Auch bei der Papierrechnung ist eine Rechnungsstellung durch Dritte zulässig. Papierrechnungen werden aber in der Regel nicht unterschrieben und so bleibt dies dem Empfänger verborgen, während elektronische Rechnungen eben eine Signatur tragen, welche Rückschlüsse auf den Signator zulässt.

Signatur wird als zu kompliziert angesehen

Der Betrieb einer eigenen Signaturlösung an sich ist nicht gerade trivial. Vor allem aber der Umstand, dass Rechnungsempfänger die digitale Rechnungssignatur prüfen müssen, könnte eine gewisse Hemmschwelle darstellen.

Dieser Umstand zeigt sich dadurch, dass digitale Signaturen und Zertifikate gut zur Absicherung der Kommunikation zwischen Maschinen eingesetzt werden können. Als Beispiele seien hier eine mit Zertifikaten gesicherte VPN-Einwahl oder die Absicherung einer Webapplikation durch SSL/TLS genannt. Traditionell ergeben sich aber auch bei diesen Anwendungsfällen Probleme, wenn z. B. der unerfahrene Benutzer mit der Prüfung eines Zertifikates konfrontiert wird. Auch deswegen können „Phishing-“ und „Pharming-Angriffe“ dermaßen große Schäden verursachen.

So elegant und nützlich Technologien wie asymmetrische Kryptographie, digitale Signatur und Zertifikate für den Techniker auch scheinen, so verwirrend und undurchsichtig können sie für technische Laien sein. Für diese kann daher schon das Prüfen der digitalen Signatur einer einfachen Rechnung eine Herausforderung darstellen.

Abwertung der fortgeschrittenen Signatur

Die Problematik der fortgeschrittenen Signatur könnte durch die Änderungen im `SigG` verschärft werden (siehe „Anhang A.3 auf S. 158“). Seit 1.1.2008 gilt dieses bis auf wenige Ausnahmen nur mehr für ZDA, welche qualifizierte Zertifikate anbieten. Dadurch wird natürlich die fortgeschrittene Signatur abgewertet, da für diese nun weder verpflichtende Verzeichnis- und Widerrufsdienste noch eine Kontrolle durch die Aufsichtsstelle[9] vorgesehen sind. Ein Szenario in dem Zertifikate, die auf ehemalige Mitarbeiter ausgestellt sind und den Firmenwortlaut tragen, nicht mehr widerrufen werden können, ist für Unternehmen sicher nicht wünschenswert. Natürlich können ZDA von sich aus weiter solche Dienste anbieten, gesetzlich geregelt und von der Aufsichtsstelle kontrolliert wird dies aber nun nicht mehr. Diese Argumentationen sind auch in den

[9] Der RTR GmbH

Stellungnahmen der Firmen „it20one“ [43] und „Data Systems Austria“ [31] zum neuen `SigG` zu finden.

2.3. Vorgeschlagene Änderungen

Aufgrund der oben genannten Probleme und der daraus resultierenden relativ geringen Verbreitung der elektronischen Rechnung ist das BMF bestrebt, die gesetzlichen Anforderungen an elektronische Rechnungen zu ändern.

Begonnen wurde diese Diskussion bereits Ende 2006, als die WKÖ auf die Rechtsunsicherheit für den Rechnungsempfänger hinwies. Infolge dessen wurden im Verlauf des Jahres 2007 Arbeitskreise abgehalten, um mögliche Lösungen zu erarbeiten. Im Folgenden werden diese Lösungen diskutiert. Die hier präsentierten Informationen wurden bei den e-Billing Arbeitskreisen sowie bei diversen anderen e-Billing-Veranstaltungen und durch Experteninterviews gesammelt.

2.3.1. Vorschlag 1: Drei gleichrangige Modelle

Dieser erste Vorschlag zur Änderung der Anforderungen an elektronische Rechnungen wurde Mitte April 2007 erstmals veröffentlicht und ist in [34] beschrieben. Er schlägt vor, das derzeitige Modell der fortgeschrittenen Rechnungssignatur durch drei gleichrangige Modelle zu ersetzen. Für EDI-Rechnungen würde sich bei diesem Vorschlag nichts ändern, sie könnten wie bisher ausgetauscht werden. Lediglich die Sammelrechnung, welche vom BMF gefordert wird, müsste, sofern sie elektronisch verschickt wird, ebenfalls den neuen Regelungen entsprechen.

Zwei der vorgeschlagenen Modelle zum Ersatz der fortgeschrittenen Signatur, nämlich das Modell „Bestätigung“ und das Modell „Finanzonline“, wären komplett neu und würden dem in „Abschnitt 2.1.3. „Third Option“ “ beschriebenen Prinzip folgen.

Ziel der Änderungen sollte es laut BMF sein, einerseits die Unsicherheiten der fortgeschrittenen Signatur zu beheben, und andererseits die derzeitigen Lösungen um „einfachere“ und leichter zugängliche Varianten zu erweitern. Ursprünglich wollte man diese Änderungen per Verordnung bis Anfang 2008 umsetzen.

Abbildung 2.4 zeigt die vom BMF vorgeschlagenen drei Varianten des Rechnungsversands. Tabelle 2.2 stellt die derzeitige Rechtslage den drei vorgeschlagenen Modellen gegenüber.[10]

[10] Die Möglichkeit des Rechnungsaustausches per EDI wurde hier nicht betrachtet, da diese von der in dieser Arbeit betrachteten Problemstellung größtenteils sehr weit entfernt ist.

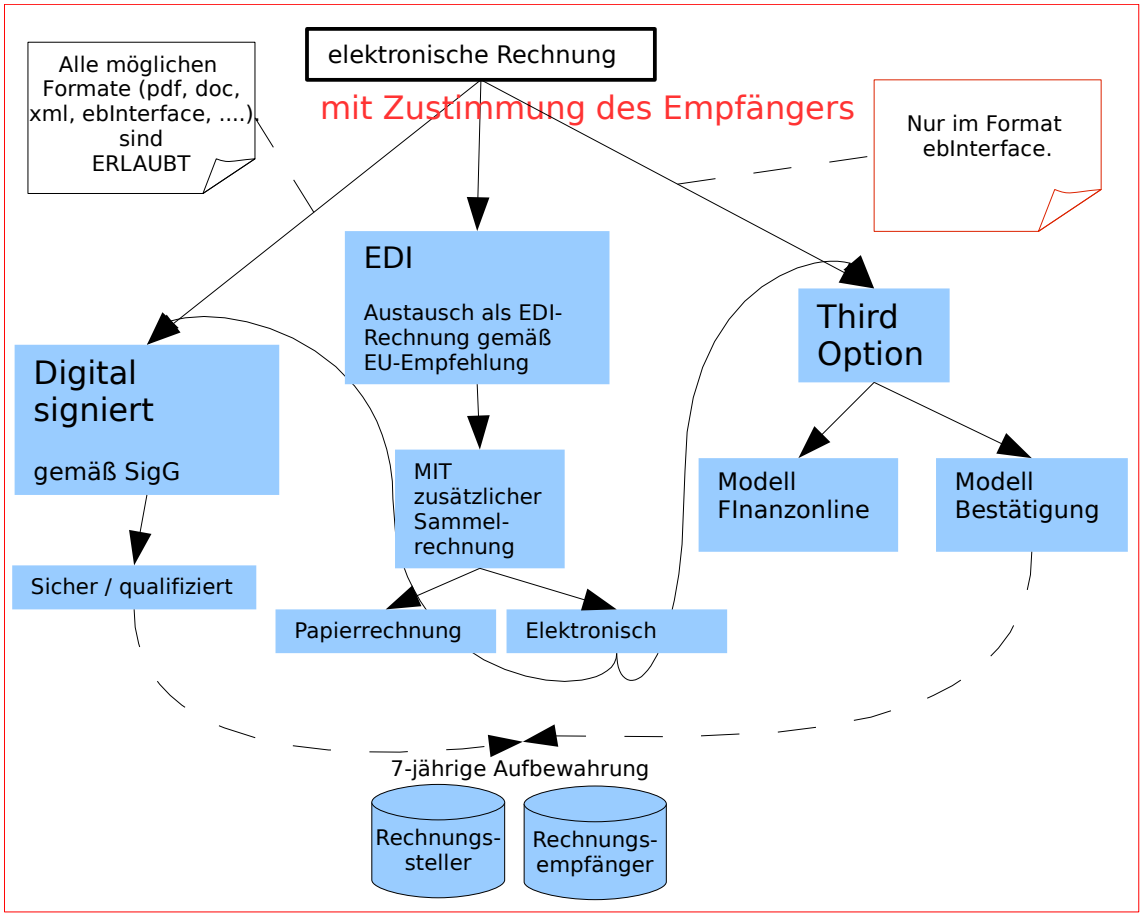

Abbildung 2.4.: Möglichkeiten zum elektronischen Rechnungsversand mit den drei vorgeschlagenen Modellen

Modell „Investitionsschutz“

Dieses Modell entspricht weitgehend der aktuellen Rechtslage, allerdings mit dem wichtigen Unterschied, dass laut [35] sichere/qualifizierte[11] Signaturen verwendet werden müssten. Dadurch sollten die Probleme der fortgeschrittenen Signatur beseitigt werden (siehe „2.2. Probleme der derzeitigen Rechtslage“ auf S. 27).

Dies würde allerdings die Praxis der Rechnungsstellung verkomplizieren, da Serversignaturen in Echtzeit nicht mehr möglich wären (siehe „2.1.2. Server- und Massensignatur“ auf S. 22), und der Einsatz von SmartCards (inklusive Lesegeräten mit integriertem PIN-Pad) obligat werden würde.[12] Es scheint nicht ganz klar zu sein, ob man hier wirklich sichere/qualifizierte Signaturen fordern würde, oder ob man sich auch mit fortgeschrittenen Signaturen auf Basis von qualifizierten Zertifikaten zufrieden geben würde.

[11] Zum Zeitpunkt der Veröffentlichung dieser Änderungsvorschläge, wurde die qualifizierte Signatur noch als „sichere Signatur“ bezeichnet. Aus diesem Grund werden in diesem Abschnitt immer beide Bezeichnungen genannt.

[12] Eine, wenn auch sehr teure, Alternative soll hier durch den Einsatz zertifizierter HSM gegeben sein.

Letzteres scheint hier eher wahrscheinlich zu sein, denn die Rechtswirkungen einer sicheren/qualifizierten Signatur werden bei elektronischen Rechnungen nicht benötigt und die praktische Durchführung der Rechnungsstellung würde trotz der Verwendung qualifizierter Zertifikate noch handhabbar bleiben. Dadurch würde eine ähnliche Abwicklung der Rechnungssignatur wie in Deutschland ermöglicht werden (siehe „2.1.2. Server- und Massensignatur“ auf S. 22). Die Firma A-Trust hat für diesen Fall bereits ein Produkt angekündigt, welches in „Abschnitt 3.6.3 – Auswahl eines Zertifikats“ kurz beschrieben ist.

Eine erzwungene Verwendung von qualifizierten Zertifikaten würde in jedem Fall sowohl für Anbieter von e-Billing-Lösungen als auch für Anwender einen nicht zu unterschätzenden Mehraufwand bedeuten.

Modell „Bestätigung“

Dieses Modell ähnelt den Anforderungen in anderen europäischen Staaten. Es müssten dabei vom Rechnungsempfänger keine digitalen Signaturen mehr geprüft werden, aber er müsste die Rechnungen sachlich richtig stellen und dies dann dem Finanzamt gegenüber bestätigen. Eine Rechnung sachlich richtig zu stellen bedeutet nicht mehr, als die Rechnung auf ihre inhaltliche und formale Korrektheit und Gültigkeit zu überprüfen. Dies ist natürlich auch bei einer Papierrechnung nötig und endet meist mit der Bezahlung der selbigen. Durch die Bezahlung bestätigt der Rechnungsempfänger in der Praxis ihre Richtigkeit, denn eine nicht korrekte Rechnung würde er wahrscheinlich auch nicht bezahlen. Genau hier setzt das Modell „Bestätigung“ an. Im Rahmen eines Experteninterviews beschreibt Frau Alexandra Sladek von AustriaPro dieses Modell folgendermaßen:

> *„Das Bestätigungsmodell besagt, dass man elektronische Rechnungen verschickt, und der Empfänger dem Finanzamt gegenüber klarstellt, dass er die Echtheit der Herkunft und die Unversehrtheit der Daten bestätigt – deshalb auch Bestätigung. Das macht man einfach wenn man gezahlt hat. Das ist wie bei einer Papierrechnung – man bestätigt durch die Zahlung, dass die Rechnung rechtens ist.“*

Obwohl eigentlich die Bezahlung die Bestätigung darstellt, würde dies alleine nicht ausreichen. Gewisse Formalismen wären hier zusätzlich notwendig, um die Prüfung der Rechnung zu bestätigen. Als geeignete Nachweise wären z. B. eine automatisierte Bestätigung durch ein ERP-System oder eine pauschale Sammelbestätigung in der Umsatzsteuervoranmeldung denkbar.

Das Modell würde vorsehen, dass die Rechnungen weiterhin digital signiert werden, es wurde aber festgehalten, dass dafür keine besonderen Anforderungen gelten würden.

Es ist also auf alle Fälle davon auszugehen, dass hier eine fortgeschrittene Signatur genügen würde. Diese müsste vom Empfänger auch nicht mehr verpflichtend geprüft werden.

Dieses Modell könnte laut Frau Sladek z. B. für Unternehmen interessant sein, die Rechnungen eigentlich per EDI austauschen, da die nötigen Sammelrechnungen dann sehr elegant per Bestätigungsmodell ausgetauscht werden könnten.

Rechnungen müssten bei diesem Modell in einem vom BMF festgelegten, standardisierten und strukturierten Format ausgetauscht werden. Hier könnte der ebInterface-Standard Verwendung finden.

Modell „Finanzonline“

Das Modell „Finanzonline“ ist wohl das innovativste, aber auch gleichzeitig das meist kritisierte Modell unter den drei Varianten des ersten Vorschlags.

Dabei würde das bekannte und von vielen österreichischen Firmen und Bürgern bereits erfolgreich verwendete „Finanzonline“-Portal als Clearing-Plattform für B2B-Rechnungen verwendet werden. Rechnungssteller könnten dabei ihre elektronischen Rechnungen an Finanzonline liefern, welche dort den Empfängern zugeordnet würden. Diese könnten dann Rechnungen von all ihren Lieferanten und Dienstleistern über diese Plattform beziehen.

Um dieses Modell nutzen zu können müssten die Rechnungen, wie schon beim Bestätigungsmodell, in einem vom BMF festgelegten standardisierten und strukturierten Format ausgetauscht werden. Auch hier ist es wahrscheinlich, dass der ebInterface-Standard Verwendung finden könnte (siehe „4.3.1. ebInterface“ auf S. 103). Hintergrund ist hier natürlich nicht nur, dass man den strukturierten Rechnungsaustausch fördern möchte, sondern dass damit auch Möglichkeiten zur automatisierten Rechnungsprüfung geschaffen würden.

Besonders interessant an diesem Modell scheint, dass Rechnungen, die über Finanzonline ausgetauscht werden, nicht mehr signiert werden müssten. Außerdem würde für solche Rechnungen auch die Aufbewahrungspflicht[13] entfallen, da diese sowieso automatisiert bei der Finanzverwaltung abgeliefert werden würden.

Dieser Service würde sowohl für Rechnungssteller, als auch für Rechnungsempfänger kostenlos angeboten werden, was das Modell vor allem für KMU interessant machen sollte.

[13] Sowohl für Rechnungssteller als auch -empfänger.

Die Grundidee des Modells ist durchaus interessant und es könnte zu erheblichen Kosteneinsparungen bei teilnehmenden Unternehmen führen. Trotzdem bleibt die Frage, ob die Wirtschaft es in dieser Art akzeptieren würde, da das Finanzamt dadurch doch einen sehr umfassenden Einblick in das Tagesgeschäft der teilnehmenden Unternehmen erhalten würde. Ebenfalls nicht vergessen sollte man die Gefahren potentiellen Datenmissbrauchs. Auch in Rechnungen können durchaus schützenswerte Informationen wie Rabattwerte oder Preisangaben enthalten sein, deren Veröffentlichung Schaden anrichten könnte. Weiters ist anzuführen, dass die Finanzverwaltung eigentlich beliebig auf die Rechnungsdaten zugreifen könnte. Natürlich handelt es sich bei den gespeicherten Daten um Informationen, welche auch im Rahmen einer traditionellen Steuerprüfung jederzeit geprüft werden könnten, aber bei einer derartig umfassenden Ansammlung von automatisiert auswertbaren Daten klingeln bei Datenschützern natürlich die Alarmglocken. So ist in [32] sogar vom „Parallelbuchhalter Staat“ die Rede. Es ist davon auszugehen, dass sich das BMF wohl zu umfassenden Datenschutzmaßnahmen verpflichten müsste, um die kritischen Stimmen zu beruhigen.

Man ist sich im BMF anscheinend auch darüber bewusst, dass es sich hier um ein heikles Thema handelt. In einem Experteninterview mit Vertretern des BMF wurde auch die Idee geäußert, die Rechnungen nicht auf einem System beim BMF zu speichern, sondern nur dort durchzuschleifen. Dabei könnte ein Hash-Wert gebildet werden, welcher dann bei einer Prüfung mit dem vom Geprüften ausgehändigten Original verglichen werden könnte. Die Unternehmen würden sich dann die Signatur und deren Prüfung sparen, nicht aber die Archivierung.

Laut Frau Alexandra Sladek von AustriaPro haben jedenfalls einige ebTransfer-Berater (siehe „1.4. AustriaPro“ auf S. 15) Bedenken geäußert, ob ihre Kunden solch ein Modell verwenden würden. Trotzdem ist es nicht auszuschließen, dass die Abnahme der gesetzlichen Archivierung und der einfache Prozess der Rechnungsstellung ohne eigener Signaturlösung verbunden mit den möglichen Kosteneinsparungen durch die elektronische Rechnung eine gewisse Anziehungskraft ausüben könnten.

	Derzeitige Rechtslage	**Investitionsschutz**	**Bestätigung**	**Finanzonline**
Art der Signatur	fortgeschritten	sicher	fortgeschritten[a]	keine Signatur
Signaturprüfung nötig	ja	ja	nein	keine Signatur
Massensignatur[b] möglich -"- auch in Echtzeit	ja ja	ja nein	ja ja	keine Signatur keine Signatur
Aufbewahrungspflicht	ja	ja	ja	nein
Formate: PDF, Word, XML, andere ebInterface	 ja ja	 ja ja	 nein ja	 nein ja
Komplexität[c]: für Rechnungssteller für Rechnungsempfänger	 komplex sehr komplex	 sehr komplex komplex	 einfach einfach	 sehr einfach sehr einfach
benötigte Komponenten[d]: Rechnungssteller	• Zertifikat (Software) • Signatursoftware • optional: Software zur Erzeugung von ebInterface-Rechnungen • Archivierung	• Zertifikat (auf SmartCard) • Kartenlesegerät • Signatursoftware • optional: Software zur Erzeugung von ebInterface-Rechnungen • Archivierung	• Zertifikat (Software) • Signatursoftware • Software zur Erzeugung von ebInterface-Rechnungen • Archivierung	• Software zur Erzeugung von ebInterface-Rechnungen • optional: Software zum Einspeisen in Finanzonline
Rechnungsempfänger	• Prüfsoftware • Archivierung	• Prüfsoftware • Archivierung	• Archivierung • optional: Software zur Bestätigung	• optional: Software zum Abholen bei Finanzonline

[a] Nach derzeitigem Wissen.
[b] Siehe „2.1.2. Server- und Massensignatur“ auf S. 22
[c] (1) sehr komplex, (2) komplex, (3) einfach, (4) sehr einfach
[d] Dies sind die Mindestvoraussetzungen um mit der e-Rechnung arbeiten zu können. Natürlich wird man zum effizienteren Arbeiten wohl auch noch in passende FIBU- oder ERP-Software investieren müssen.

Tabelle 2.2.: Vergleich der Modelle

2.3.2. Vorschlag 2: Verzicht auf die digitale Signatur

Beim e-Billing-Arbeitskreis am 14. November 2007 erwarteten die Teilnehmer eine Konkretisierung der Mitte 2007 vorgeschlagenen drei Modelle. Anstatt dessen verkündete Herr Dr. Laga von der WKÖ, dass das BMF in einem anderen Arbeitskreis überraschend durchblicken ließ, dass man offenbar dazu bereit wäre, alle Formalerfordernisse an die elektronische Rechnung fallen zu lassen. Einzige Ausnahme soll die Zustimmung des Empfängers sein, welche weiterhin nötig sein würde. Damit würden die zuerst vorgeschlagenen drei Modelle natürlich obsolet werden.

Damit würde auch die Anforderung wegfallen, elektronische Rechnungen digital signieren zu müssen. Es wurde angemerkt, dass der Empfänger weiterhin auf eine digitale Signatur bestehen könnte, wobei die Wahrscheinlichkeit, dass er dies tun würde, aus Sicht des Autors eher als gering einzuschätzen ist.

Es wurde bei diesem Arbeitskreis auch über weitere mögliche Folgen spekuliert. So würde man bei der Archivierung laut Meinung einiger Teilnehmer wieder zum Modell der Papierrechnung zurückkommen, wo der Rechnungssteller Rechnungen nur reproduzieren können, aber nicht archivieren muss. Auch würden viele der Teilnehmer keine Notwendigkeit mehr für die Abschaffung der Faxrechnung und die Pflicht zur EDI-Sammelrechnung sehen, falls dieser Vorschlag umgesetzt werden würde.

Aus verwaltungstechnischer Sicht wäre dieser Vorschlag laut Herrn Dr. Laga interessant, weil dadurch Prozesse vereinfacht werden könnten. Außerdem entspräche es einer Legalisierung der Praxis, da derzeit bereits sehr viele unsignierte elektronische Rechnungen ausgetauscht werden. Andererseits sahen einige Teilnehmer mögliche Probleme mit der `RL 2001/115/EG`, laut welcher „Echtheit der Herkunft“ und „Unversehrtheit des Inhalts“ einzuhalten sind (siehe „2.1. Derzeitige Rechtslage“ auf S. 17). Ihrer Meinung nach würden diese Anforderungen auch für die sogenannte „Third Option“ gelten, welche diesen Vorschlag abdecken würde. Von einer Gewährleistung dieser Anforderungen kann beim vorliegenden Vorschlag natürlich nicht gesprochen werden, wodurch hier ein Verstoß vorliegen könnte. Als Gegenargument kann hier die Umsetzung in einigen anderen europäischen Ländern wie z. B. England dienen, wo diese Punkte auch nicht klar erfüllt zu sein scheinen.

2.4. Fazit

Die beiden Vorschläge, also die drei Modelle und der Verzicht auf die digitale Signatur, wurden von den Stakeholdern unterschiedlich aufgenommen. Während z. B das Modell Finanzonline von vielen aufgrund seiner Innovativität gefeiert wurde, sahen andere

darin einen Schritt hin zum „gläsernen Unternehmen“. Auch die Konzentration auf ein einziges Format wurde beim Vorschlag der drei Modelle heftigst kritisiert. Bei der Verkündung der Absicht, auf die digitale Signatur verzichten zu wollen, schienen die Überraschung der Teilnehmer und die Skepsis über die Durchführbarkeit im Vordergrund zu stehen.

Um eine raschere Verbreitung der elektronischen Rechnung zu begünstigen, müssten schnell klare Regelungen geschaffen werden. Leider ist man davon derzeit anscheinend sehr weit entfernt. Bei der Vorstellung der drei Modelle wurde als Umsetzungszeitpunkt Anfang 2008 genannt. Mit dem im November vollzogenen Meinungswechsel hat sich dieser Termin auf unbestimmte Zeit verschoben. Natürlich ist es schwierig, bei einem derart komplexen Thema eine für die Mehrheit befriedigende und vernünftige Lösung zu finden. Es darf auch nicht vergessen werden, dass obwohl oft die geringe Verbreitung der elektronischen Rechnung kritisiert wird, trotzdem schon viele Firmen Rechnungen auf Basis der geltenden Rechtslage austauschen und somit schon entsprechende Investitionen getätigt haben.

Es ist also durchaus verständlich, dass verschiedene Gedankenmodelle überlegt und zur Diskussion gestellt werden. Absolut unbefriedigend ist aber die Tatsache, dass diese Diskussion seit Vorstellung des letzten Vorschlages im November 2007 anscheinend nicht mehr geführt wird. Offensichtlich sind nicht einmal Experten im Bereich e-Billing über den derzeitigen Status informiert, und es hat den Anschein, als würde hier im Augenblick Stillstand herrschen.

Abbildung 2.5 zeigt die Meilensteine rund um die elektronische Rechnung auf einer Zeitachse aufgetragen.

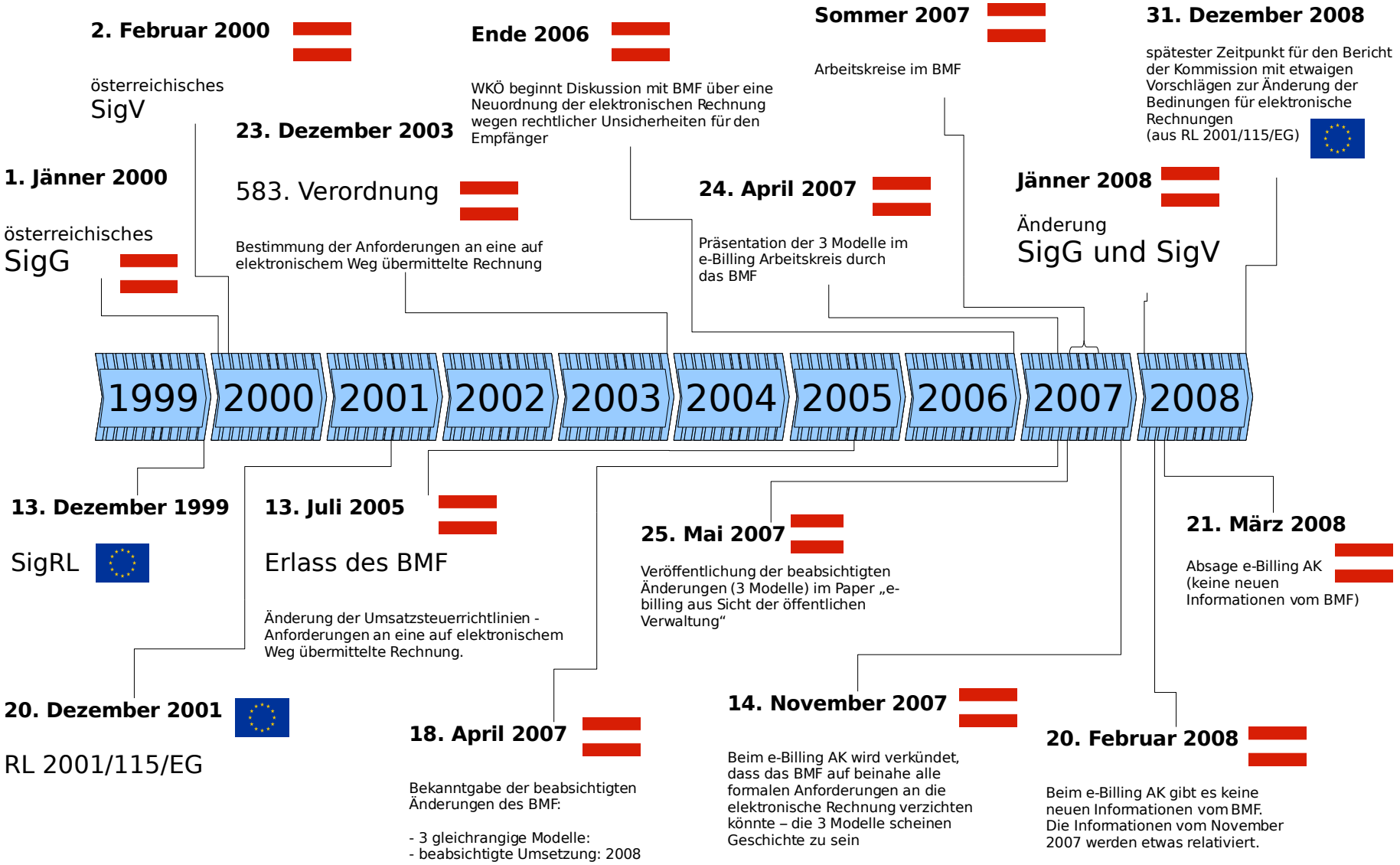

Abbildung 2.5.: Chronologie der elektronischen Rechnung

3 Betriebswirtschaftliche Sicht – Fallstudie

3.1. Vorstellung des Unternehmens

Die „AVIA-Tankstelle H. Schlapschy“ ist eine unabhängige Tankstelle mit dazugehörigem Gastgewerbe im Bezirk Freistadt (Oberösterreich). Zudem sind auch eine Autowaschanlage und ein Reifenhandel Teil des Angebotes. Der Betrieb wird vom Besitzer mit Unterstützung von 2-3 Angestellten geführt.

Es gibt cirka 350 registrierte und aktive Stammkunden, welche mit einer speziellen Kundenkarte[1] auch außerhalb der Öffnungszeiten an einem Automaten tanken können. Diese Stammkunden erhalten eine monatliche Rechnung über ihre Gesamtumsätze, und der Rechnungsbetrag wird entweder über einen Abbuchungsauftrag vom Bankkonto des Kunden abgebucht, oder vom Kunden per Überweisung beglichen. Die Stammkundschaft setzt sich sowohl aus Privatkunden (ca. 92 %), als auch aus Firmenkunden (ca. 8 %) zusammen. Eine zum Vorsteuerabzug berechtigende Rechnung wird nur von Letzteren benötigt. Wahrscheinlich kann aber davon ausgegangen werden, dass nicht alle Firmenkunden automatisch auch vorsteuerabzugsberechtigt für die von ihnen konsumierten Treibstoffe sind, da ein Vorsteuerabzug im Zusammenhang mit dem Betrieb von Kraftfahrzeugen anderer Art als Lastkraftwagen nur in Ausnahmefällen möglich ist [20].

Laufkundschaft, welche an der Kasse zahlt, erhält eine den Anforderungen für Kleinstbetragsrechnungen (siehe auch Tabelle 2.1) entsprechende Quittung.

Die Tankstelle verfügt über ein bereits etwas älteres Soft- und Hardwaresystem der Firma „Task Technology GmbH“[2], mit dem nicht nur die Zapfsäulen und die Kasse gesteuert, sondern auch die Kundendaten und Umsatzzahlen verwaltet werden. Das Softwaresystem besteht aus zwei Teilen, die Daten über das Dateisystem austauschen:

[1] Dabei handelt es sich um eine Magnetstreifenkarte im Format ID-1.
[2] `www.tasksystems.de`

Frontend "KT" steuert die Kasse und die Zapfsäulen

Backend "Possum" ist das Verwaltungsprogramm im Hintergrund (Kundenmanagement, Monatsabrechnung, ...)

3.2. Ist-Zustand

Momentan werden Stammkundenrechnungen[3] ausgedruckt und auf Papier übergeben.

3.2.1. Rechnungsdruck

Am Monatsende werden, im Zuge der Monatsabrechnung, die Rechnungen ausgedruckt. Tabelle 3.1 zeigt diesen Vorgang, der aufgrund des etwas älteren und unflexiblen Systems relativ kompliziert anmutet.

Die farbig hervorgehobenen Tätigkeiten könnten durch den Einsatz der e-Rechnung vereinfacht werden.

[3] Es werden in der Fallstudie nur die Stammkundenrechnungen betrachtet, da es derzeit sicher keinen Sinn macht, elektronische Rechnungen an Laufkundschaft zu verschicken. Wenn im Zusammenhang mit der Fallstudie das Wort Rechnung gebraucht wird, so bezieht sich dies auf eine Stammkundenrechnung.

System	Schritte	Dauer
Produktivsystem	• Abrechnung im Backend-Programm starten • Rechnungsdatum einstellen • Sicherung auf Diskette (vor Monatsabschluss) • Bankdiskette erstellen (Debitorensalden auf Diskette speichern)	17 min
Produktivsystem	• System versucht automatischen Rechnungsdruck[a] • Druck abbrechen • Daten aktualisieren • Datensicherung (nach Monatsabschluss)	8 min
Parallelsystem	• Sicherung in das Parallelsystem einspielen • Angabe einer Kopf- und Fußzeile für die Rechnungen • Rechnungsdruck im Parallelsystem starten	33 min
	Gesamt:	58 min

[a]Das System versucht beim Monatsabschluss immer, die Rechnungen zu drucken. Dies wird aber abgebrochen, da ansonsten das Produktivsystem über eine halbe Stunde lang durch den Rechnungsdruck blockiert wäre.

Tabelle 3.1.: Rechnungsdruck

3.2.2. Rechnungsversand

Die ausgedruckten Rechnungen werden derzeit nicht per Post versandt, sondern dem Kunden, abhängig von seiner Bankverbindung, auf verschiedene Art und Weise zur Verfügung gestellt:

Zuordnung zum Kontoauszug Rechnungen an Kunden eines bestimmten Kreditinstitutes werden von dessen örtlicher Filiale entgegengenommen. Dort werden die Rechnungen mit den Unterlagen des Kunden abgelegt oder an die Filiale weitergeleitet, bei welcher der Kunde seine Bankgeschäfte tätigt. Beim nächsten Bankbesuch kann sich der Kunde die Rechnung z. B. mit seinen Kontoauszügen abholen.

Dieser Service wird vom Kreditinstitut derzeit kostenlos zur Verfügung gestellt. Die Rechnungen müssen aber zuvor gefaltet und nach Bankfilialen sortiert werden.[4]

Abholung Kunden, welche nicht über ein Konto dieses Kreditinstitutes verfügen, holen sich ihre Rechnungen direkt im Geschäft ab.

[4]Die Sortierung nach Bankfilialen erfolgt anhand der auf der Rechnung abgedruckten Bankleitzahl.

Ihre Rechnungen werden ungefaltet in einen Ordner geheftet. Einige Rechnungen müssen aufgrund von diversen Kundenwünschen gesondert bearbeitet werden. Diese Rechnungen werden im selben Ordner abgelegt.

Eine schematische Darstellung des gesamten Rechnungsausgangsprozesses bietet das Aktivitätsdiagramm in Abbildung 3.1.

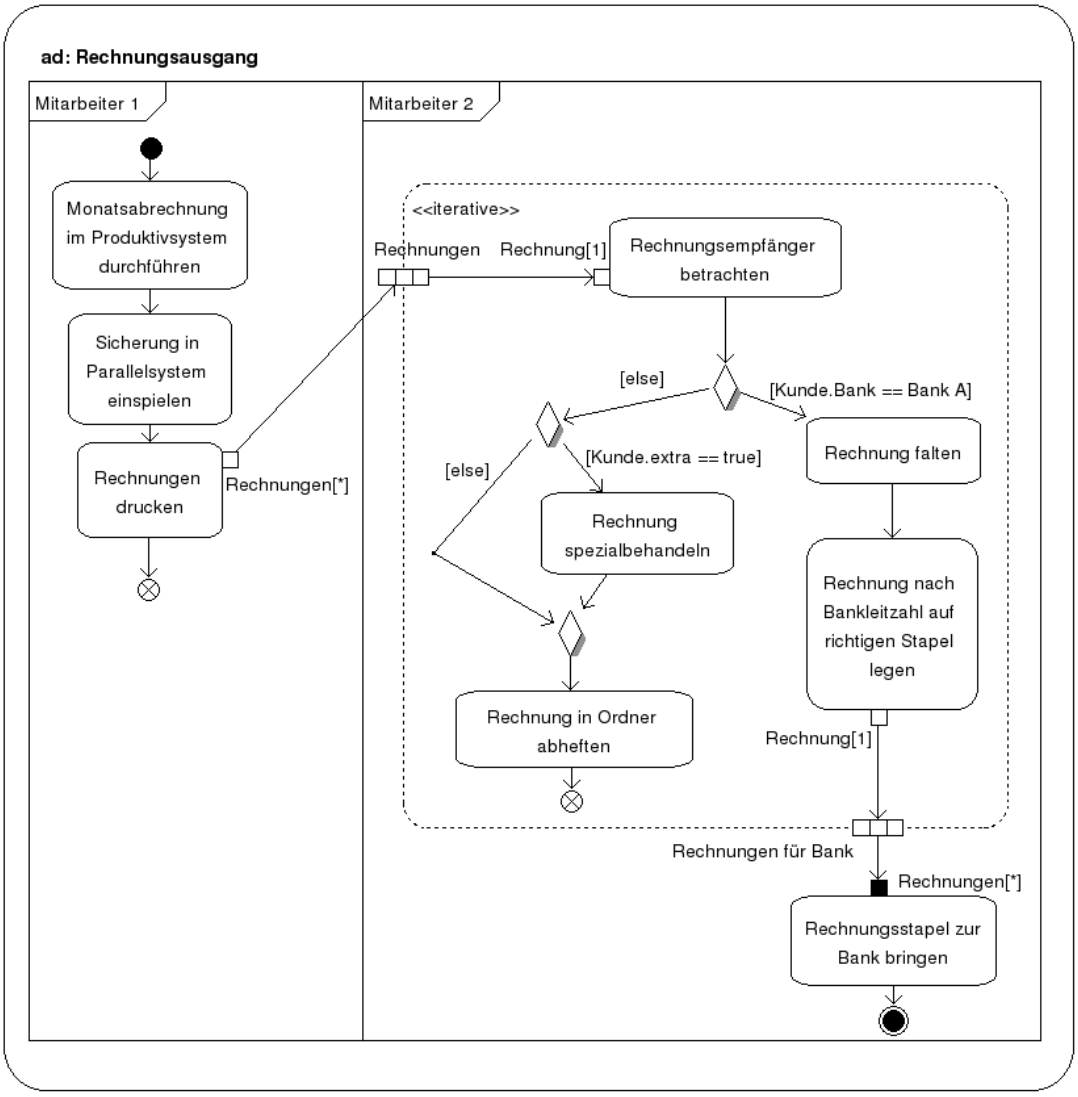

Abbildung 3.1.: Aktivitätsdiagramm Rechnungsausgang

Aus den möglichen Zustellungsarten, ergeben sich die in Tabelle 3.2 angeführten Arbeitsschritte und die dazu gemessenen Bearbeitungszeiten.

Wieder wurden die durch die e-Rechnung zu vereinfachenden Tätigkeiten farbig hervorgehoben.

Schritte	Dauer
Sortieren der Rechnungen	35 min
Falten / kuvertieren	20 min
Einheften in Ordner	5 min
Restbehandlung div. Sonderfälle	25 min
Gesamt:	85 min

Tabelle 3.2.: Vorbereitung der Rechnungszustellung

Zwischen Privat- und Firmenkunden wird bei der Rechnungsstellung nicht unterschieden, allerdings wurde im Laufe der Jahre festgestellt, dass viele Privatkunden ihre Rechnungen niemals abholen. Leider ist das Altsystem aber nicht flexibel genug, um dauerhaft festhalten zu können, für welche Kunden ein Rechnungsdruck nötig ist.

3.2.3. Jährlicher Rabatt

Stammkunden wird ein jährlicher Rabatt auf einen Großteil der Waren gewährt, sofern diese mit der Stammkundenkarte bezahlt werden. Lediglich auf Waren mit zu niedriger Marge wird kein Rabatt gewährt. Am Ende des Jahres werden die nicht rabattfähigen Rechnungsposten manuell aussortiert und der Rabattbetrag mit einer Tabellenkalkulationssoftware berechnet.

3.2.4. Rechnungseingang

Der Optimierungsfokus im betrachteten Betrieb liegt ganz klar auf Seiten des Rechnungsausgangs. Durch die elektronische Rechnung kann aber auch eine Optimierung der Rechnungseingangsprozesse beim Empfänger erreicht werden. Um das vorhandene Rationalisierungspotential zu zeigen, werden im Folgenden beispielhaft die traditionellen Rechnungseingangsprozesse des betrachteten Betriebes erläutert.

Das Unternehmen erhält Eingangsrechnungen entweder per Post, oder sie werden vom Lieferanten mitgebracht bzw. bei Selbstabholung der Ware selbst mitgenommen. Rechnungen für erhaltene Lieferungen werden anhand des Lieferscheines auf sachliche Richtigkeit geprüft. Rechnungen, welche direkt mit der Ware übergeben werden, müssen natürlich sofort gegen die Ware geprüft werden. Rechnungen für Dienstleistungen werden gegen einen Kostenvoranschlag geprüft. Alle offenen Rechnungen werden dann in einem Ordner abgeheftet. Periodisch werden alle offenen Rechnungen elektronisch bezahlt und danach mit Hilfe einer FIBU-Software verbucht.

Die Anzahl der erhaltenen Eingangsrechnungen beläuft sich nach Schätzungen der Mitarbeiter auf cirka 50 Rechnungen pro Monat.

Das Aktivitätsdiagramm in Abbildung 3.2 zeigt die beim Rechnungseingang beschriebenen Vorgänge.

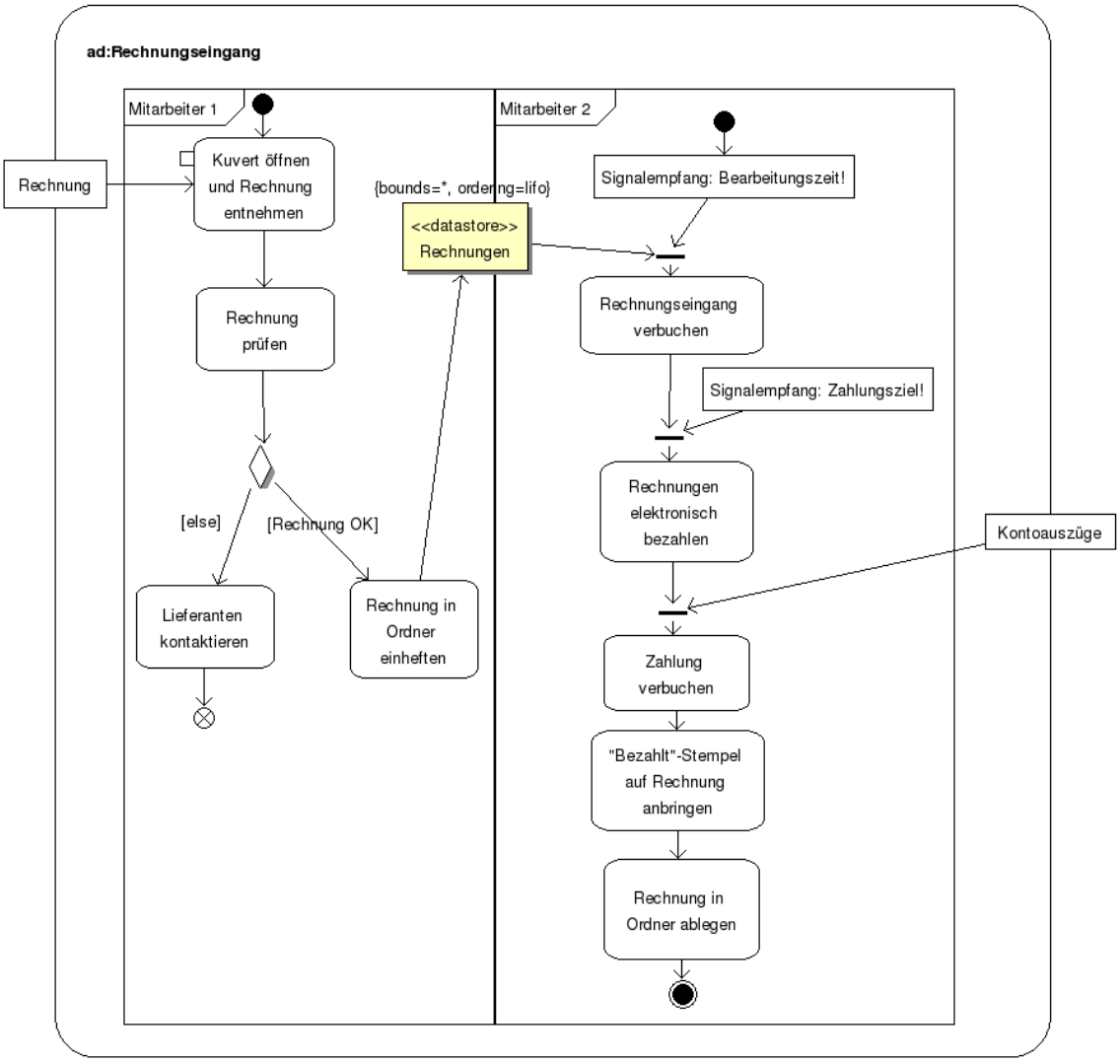

Abbildung 3.2.: Aktivitätsdiagramm Rechnungseingang

Da es, wie oben beschrieben, viele verschiedene Rechnungseingangsszenarien gibt, ist es schwierig, den genauen Zeitaufwand für eine eingehende Rechnung anzugeben. Tabelle 3.3 zeigt deshalb auch nur schemenhaft die notwendigen Schritte für die Bearbeitung einer per Post eingegangenen Lieferantenrechnung. Auf die Angabe des Zeitaufwandes für die einzelnen Schritte wurde verzichtet. Wieder wurden die Schritte, welche durch die elektronische Rechnung vereinfacht werden könnten, farbig dargestellt. Wichtig ist

Schritte
Kuvert öffnen und Rechnung entnehmen
Lieferschein suchen
Rechnungsposten vergleichen
Rechnung lochen und in Ordner abheften
Rechnungseingang verbuchen
Rechnung elektronisch bezahlen
Zahlung verbuchen
Bezahlt-Stempel anbringen

Tabelle 3.3.: Rechnungseingang

hier, dass Einsparungen nur erzielt werden können, wenn einige Voraussetzungen erfüllt sind. Diese werden in „Abschnitt 3.4.2 – Einsparungen für den Rechnungsempfänger“ betrachtet.

3.3. Verbesserungspotential durch die e-Rechnung

Obwohl der direkte Kundenkontakt beim Abholen der Rechnungen durchaus positive Nebeneffekte haben könnte, gibt es mit der derzeitigen Vorgehensweise doch einige Probleme:

- Viele Rechnungen werden vom System ausgedruckt und bearbeitet, obwohl der Kunde sie nicht abholt. Dies trifft sowohl auf die im Geschäft hinterlegten als auch auf die bei den Banken deponierten Rechnungen zu. Vor allem für Privatkunden scheint sich der Aufwand, sich Rechnungen abholen zu müssen, oft nicht zu lohnen. Auch für die Mitarbeiter des Kreditinstitutes stellt der ständig wachsende Stapel an nicht abgeholten Rechnungen ein Problem dar.

- Manche Kunden holen ihre Rechnungen gesammelt einmal im Jahr ab. Ein Mitarbeiter muss dann die Rechnungen aus den zwölf Monatsordnern zusammensuchen. Dies ist vor allem dann unangenehm, wenn Unternehmenskunden ihre Buchhaltung ausgelagert haben, und das zuständige Steuerbüro bereits dringend auf die Rechnungen wartet.

- Durch die aufwändige Bearbeitung stehen Rechnungen oft nicht schon am ersten Tag des Folgemonates für die Kunden bereit.

- Die Selbstabholung der Rechnungen durch die Kunden kann nicht als besonders kundenfreundlich bezeichnet werden. Jedoch erlauben die Kosten einer postalisch versendeten Papierrechnung keine komfortablere Zustellung.

Dem gegenüber verspricht der Einsatz der e-Rechnung hier sowohl für die Rechnungsempfänger als auch für den Rechnungssteller große Vorteile:

1. Die digitale Erstellung und Verteilung der Rechnungen kann viel Zeit und Geld einsparen (siehe „3.4.1. Einsparungen für den Rechnungssteller“ auf S. 48).

2. Die Kunden können die Rechnungsdaten ohne Medienbruch (Abbildung 1.1) weiterverarbeiten, und somit ebenfalls Zeit und Kosten sparen.

3. Die gespeicherten und gesammelten Rechnungsdaten würden auch auf Seiten des Rechnungsstellers eine komfortable Weiterverarbeitung, z. B. zur Kalkulation des jährlichen Rabattes, ermöglichen.

4. Die Kunden erhalten ihre Rechnungen immer rechtzeitig.

5. Die Archivierung der Rechnungen beim Kunden kann einfacher und effizienter durchgeführt werden. Archivierte elektronische Rechnungen müssen nicht ausgedruckt werden, können elektronisch archiviert werden, und sind somit einfach und übersichtlich zu verwalten. So kann z. B. eine gesuchte elektronische Rechnung mit Hilfe einer Desktopsuchmaschine innerhalb von Sekunden gefunden werden.

6. Firmenkunden können die Rechnungen zeitgerecht, einfach und kostengünstig an ein Steuerbüro weitergeben.

7. Auch Privatkunden können ohne großen Aufwand ihre Rechnungen beziehen.

8. Der Berg an nicht abgeholten Papierrechnungen wächst nicht weiter.

Alleine schon die ersten beiden Punkte würden eine Umstellung auf die e-Rechnung rechtfertigen. Diese möglichen Kosteneinsparungen werden in den nächsten Abschnitten sowohl aus allgemeiner Sicht, als auch speziell für den betrachteten Betrieb besprochen.

3.4. Einsparungen

Sowohl der Rechnungssteller, als auch der Rechnungsempfänger können durch den Einsatz der e-Rechnung Einsparungen erzielen. Dieser Abschnitt betrachtet beide Seiten zuerst aus allgemeiner Sicht. Für die Fallstudie wird außerdem versucht, die Einsparungen, welche der betrachtete Betrieb vor allem beim Rechnungsversand erzielen könnte, abzuschätzen.

Viele Publikationen zum Thema e-Billing verweisen auf das gesamtwirtschaftliche Einsparungspotential. Die beiden prominentesten Studien zu diesem Thema stammen von den Marktforschungsinstituten Ovum[5] und Evolaris[6], welche eigentlich allen in diesem Zusammenhang genannten Zahlen zu Grunde liegen.

In einem kurzen Telefoninterview[7] mit Herrn DI Christian Kittl von der „evolaris Privatstiftung“, wurden folgende oft genannten Zahlen bestätigt:

- in Österreich werden pro Jahr cirka 700 Millionen Rechnungen verschickt
- dabei handelt es sich bei cirka 200 Millionen Rechnungen um B2B-Rechnungen
- die Kosten pro B2B-Rechnung betragen bei allen Beteiligten[8] zusammen durchschnittlich cirka 5 Euro
- die durchschnittlichen Einsparungen pro B2B-Rechnung betragen bei allen Beteiligten[8] zusammen bis zu 3 Euro
- die durchschnittlichen Einsparungen pro Rechnung betragen bei allen Beteiligten[8] zusammen cirka 2,1 Euro. Hier sind nicht nur B2B-Rechnungen, sondern auch B2C-Rechnungen enthalten. Der niedrigere Betrag erklärt sich damit, dass die Rechnungseingangsbearbeitung beim Endverbraucher kaum Kosten verursacht. Deshalb ist auch das Einsparungspotential, und somit die durchschnittlichen Einsparungen, geringer.

Auf diesen Zahlen aufbauend lassen sich laut Herrn Kittl z. B. die oftmals genannten Einsparungen von 600 Millionen Euro im B2B-Sektor als das Produkt von 200 Millionen B2B-Rechnungen mit Einsparungen von 3 Euro pro Rechnung erklären. In einigen Publikationen (z. B. in [34]) wird auch von Einsparungen in der Höhe von 1,5 Milliarden Euro gesprochen. Diese Zahl bezieht sich laut Herrn Kittl nicht nur auf den B2B-Sektor, sondern beinhaltet auch B2C-Rechnungen, und ist also das gerundete Produkt aus 700 Millionen Rechnungen mit Einsparungen von 2,1 Euro pro Rechnung.

Aufgrund unterschiedlicher Ausgangspunkte können die genannten Werte je nach Publikation etwas abweichen. So spricht z. B. [12] von Einsparungen bis zu 70 %, oder zwischen 1 bis 3 Euro pro Rechnung, und [45] berichtet von *„jährlich 600 Millionen Euro Produktivitätsgewinn für die österreichische Volkswirtschaft, wenn 70 % der B2B-Rechnungen in Österreich elektronisch ausgetauscht werden.“*

[5] `http://www.ovum.com`
[6] `http://www.evolaris.net`
[7] vom 17. März 2008
[8] Also beim Rechnungssteller sowie beim Rechnungsempfänger.

Welchen Formulierungen, und welchen Zahlen man nun auch immer Glauben schenken mag, aus der Sicht des Autors haben diese Zahlen meist nur einen sehr plakativen Charakter. Eine individuelle Prozesskostenanalyse, wie sie im folgenden Abschnitt im Zuge der Fallstudie durchgeführt wurde, ist in jedem Fall empfehlenswert.

3.4.1. Einsparungen für den Rechnungssteller

Beim Rechnungssteller muss man von gewissen Kosten pro Papierrechnung ausgehen. Darin enthalten sind meist:

Portokosten Die Kosten für den Postversand betragen in Österreich für einfache Rechnungen[9] derzeit 55 Cent pro Rechnung.

Sachkosten Darin enthalten sind die Kosten für das Papier[10] und den Druck[11]. In vielen Studien werden diese Sachkosten allerdings auch einfach als Druckkosten bezeichnet.

Prozesskosten Darin enthalten sind Kosten für das Sortieren, Frankieren, Kuvertieren und Archivieren der Rechnungen[12]. Ebenfalls oft eingerechnet ist der Postweg, welcher zum Aufgeben der Rechnungen zurückgelegt werden muss, oder die Kosten für das Archivieren der Rechnungen.

Für die einzelnen Kostenfaktoren sind je nach Studie und Quelle oft sehr unterschiedliche Werte zu finden. Manche Studien berücksichtigen z. B. explizit den Postweg, andere verrechnen dafür höhere Prozesskosten. Tabelle 3.4 vergleicht die von verschiedenen Organisationen angegebenen Kosten pro Rechnung. In der Tabelle wurde versucht, die oft unterschiedlichen Aufschlüsselungen auf Kostenfaktoren möglichst gut zu vereinheitlichen.

Es sollte dabei aber beachtet werden, dass solche Kalkulationen oft von Unternehmen erstellt werden, welche selbst e-Billing-Produkte anbieten, und somit auch als Werbebotschaft zu verstehen sind. Außerdem ist es ebenso bemerkenswert, dass z. B. Prozesskosten, speziell in KMU, nicht immer als solche wahrgenommen werden. Als Beispiel sei hier der Postweg genannt. Dieser verursacht zweifelsfrei gewisse Kosten, welche aber oft nicht erkannt werden, wenn z. B. die Rechnungen am Ende des Tages auf dem Nachhauseweg vom Inhaber zum Postamt gebracht werden.

[9] Zustellung im Inland; Gewicht bis 20 g [50]
[10] Hier sind meist auch die Kosten für das Kuvert eingerechnet.
[11] Also für Tinte oder Toner sowie Abschreibung des Druckers.
[12] Errechnet sich z. B. als anteilsmäßiger Stundenlohn inklusive aller Abgaben, oder aus den Kosten für entsprechende Maschinen.

Kostenfaktor	AustriaPro[12]	ebRechner[a]	DataSystems Austria[b]
Porto	0,55	0,55	0,55
Druck	0,20	0,18	0,20
Postweg	0,03	-	-
Archivierung	0,10	-	0,25
Frankierung und Kuvertierung	0,05	-	0,12
Sonst. Fulfillment	-	1,38[c]	-
Gesamt	0,93 Euro	2,11 Euro	1,12 Euro

[a]Eine Kalkulationshilfe der Firma „Mesonic“. Die in diesem Programm angegebenen und zur Kalkulation verwendeten Beträge wurden in die Tabelle eingegliedert.
[b]Die entsprechenden Unterlagen wurden im Zuge des ebTransfer-Workshops [12] zur Verfügung gestellt und sind auf der ebTransfer-CD verfügbar. Die in der Tabelle angegebenen Beträge sind gerundet, und die Teilbeträge wurden auf die in der Tabelle angegebenen Kostenfaktoren aufgeschlüsselt.
[c]Dieser Wert beinhaltet das gesamte Fulfillment inklusive Postweg, Frankierung, ...

Tabelle 3.4.: Kosten für eine Papierrechnung

Im Zuge der Fallstudie wurde versucht, die Kosten pro Ausgangsrechnung zu schätzen, um einen ungefähren Eindruck über das mögliche Einsparungspotential zu erhalten. Dabei wurde der Zeit- und Kostenaufwand für die in Tabelle 3.1 und Tabelle 3.2 angegebenen Schritte bei der Rechnungsstellung ermittelt. Die Druckkosten wurden anhand von vorhandenen Rechnungen für Papier, Toner und Drucker ermittelt. Die Deckung von 3,5 % wurde durch den Vergleich einer durchschnittlichen Rechnung mit dem sogenannten „Dr.-Gilbert-Testbrief[13]“ geschätzt. Aufgrund der bereits beschriebenen Rechnungszustellung (siehe „3.2.2. Rechnungsversand“ auf S. 41) fallen keine Portokosten an. Tabelle 3.5 zeigt die ermittelten Kosten für die Rechnungsstellung beim betrachteten Betrieb.

[13]Dieser weist eine Deckung von genau 5 % auf (`http://www.druckerchannel.de/artikel.php?ID=34&t=druckerchannel_testdokumente`).

Kostenfaktor	Kostenfaktor detail		Kosten pro Monat	Kosten pro Jahr	Kosten pro Rechnung
Porto	Porto		-	-	-
Prozesskosten [1)]	Abrechnung	17 min	€ 6,16	€ 73,92	€ 0,0176
	Sicherung	8 min	€ 2,90	€ 34,79	€ 0,0083
	Drucken	33 min	€ 11,96	€ 143,50	€ 0,03
	Sortieren	35 min	€ 12,68	€ 152,19	€ 0,04
	Falten / kuvertieren	20 min	€ 7,25	€ 86,97	€ 0,02
	Einheften	5 min	€ 1,81	€ 21,74	€ 0,01
	Restbehandlung div. Kunden	25 min	€ 9,06	€ 108,71	€ 0,03
Druckkosten	Papier [2)]		€ 1,75	€ 21,00	€ 0,0050
	Drucker [3)]		€ 5,83	€ 70,00	€ 0,0167
	Toner [4)]		€ 1,06	€ 12,75	€ 0,0030
		Gesamt:	€ 60,46	€ 725,57	€ 0,17

[1)] Prozesskosten:

Gehalt pro Jahr (btto, inkl. Urlaubs- u. Weihnachtsgeld)		€ 32.702,00	1 Jahr = 12 Monate
+Arbeitnehmeranteil	33 %	€ 10.791,66	1 Monat = 4,33 Wochen
		€ 43.493,66	1 Woche = 38,5 Stunden
Kosten pro Stunde Arbeit:		€ 21,74	

[2)] Papier (1 Packung, 500 Blatt): € 2,50

[3)] Drucker (Brother HL-1250)

Neupreis:	€ 350,00
auf 5 Jahre gerechnet:	€ 70,00

[4)] Toner (für Brother HL-1250) € 85,00 — 28000 Blatt bei ca. 3,5 %iger Deckung

Anzahl Rechnungen / Monat: 350

Tabelle 3.5.: Ermittelte Kosten für Rechnungsstellung

Die rötlich markierten Gesamtkosten für die Papierrechnung bewegen sich in den von AustriaPro in Tabelle 3.4 angegebenen Dimensionen, wenn man bedenkt, dass im betrachteten Betrieb keine Porto- und Frankierungskosten anfallen.

Da die im Rahmen der Fallstudie errechneten Gesamtkosten pro Rechnung deutlich unter den werbewirksam kommunizierten Werten der e-Billing-Softwarehersteller liegen, ist natürlich auch das Einsparungspotential durch die e-Rechnung dementsprechend geringer. Die in Tabelle 3.5 bläulich hinterlegten Vorgänge entsprechen dem bereits vorher identifizierten Rationalisierungspotential (siehe „3.2. Ist-Zustand“ auf S. 40) und könnten durch eine Umstellung auf die e-Rechnung wesentlich vereinfacht werden.

Lässt man kurzzeitig außer Acht, dass auch eine e-Billing-Lösung natürlich gewisse Investitionskosten, sowie geringe Prozesskosten verursacht, kann man erkennen, dass sich beim betrachteten Betrieb durch die Ablöse der Papierrechnung bis zu 0,15 Euro pro Rechnung einsparen ließen. Im Vergleich mit Tabelle 3.4 erscheint dieses Ergebnis natürlich entäuschend. Der Grund für diesen niedrigen Wert ist allerdings, wie in diesem Abschnitt mehrmals erklärt wurde, darin zu sehen, dass der zurzeit praktizierte Prozess hauptsächlich darauf abzielt, die Kosten für die Ausgangsrechnungen zu minimieren. Kosteneinsparungen beim Rechnungssteller sind aus der Sicht des Autors allerdings nur einer von vielen Gründen für eine Umstellung auf die elektronische Rechnung. Warum die Einführung der elektronischen Rechnung im betrachteten Betrieb trotzdem Sinn macht, wird im übernächsten Abschnitt besprochen (siehe „3.4.3. Fazit“ auf S. 55).

3.4.2. Einsparungen für den Rechnungsempfänger

Meist wird beim e-Billing zuerst an das Einsparungspotential beim Rechnungssteller gedacht (siehe „3.4.1. Einsparungen für den Rechnungssteller“ auf S. 48). Dies ist auch insofern verständlich, als dieser zuerst direkt durch den Wegfall der Versandkosten profitiert, aber auch die ersten Investitionskosten für eine e-Billing-Infrastruktur[14] zu tragen hat.

Vergessen wird dabei aber oft, dass auf Seiten des Rechnungsempfängers ebenfalls ein großes, wenn nicht sogar noch größeres Einsparungspotential liegt. Dieses Potential wird erst begreifbar, wenn man sich die Rechnungseingangsprozesse vor Augen führt. In einer Präsentation beim „e-Day 2007“[15] beschreibt ein Referent den Ablauf cirka folgendermaßen (nach [48], leicht modifiziert):

[14] Es wird hier nicht unterschieden, ob er die Infrastruktur selbst betreibt, oder Dienstleistungen zukauft. Die verschiedenen Möglichkeiten e-Billing zu betreiben werden in Abschnitt „Abschnitt 3.6.2 – e-Billing-Umsetzungsmöglichkeiten“ betrachtet.

[15] Der e-Day (`www.eday.at`) ist eine Veranstaltung der WKÖ, bei der KMU die Möglichkeit haben, sich über aktuellen Entwicklungen und Einsatzmöglichkeiten der Informations- und Kommunikationstechnologie zu informieren. Schwerpunkt der Veranstaltung 2007 war die elektronische Rechnung.

1. *Post bringt Rechnungen vom Lieferanten*
2. *Sekretariat öffnet Brief, streicht Rechnung glatt, führt eventuell noch händisch ein Rechnungseingangsbuch*
3. *Sekretariat macht 1 bis 5 Kopien, legt natürlich eine Kopie im Sekretariat ab*
4. *Sekretariat gibt Original in die Buchhaltung zum Einbuchen, und Kopien an die Fachabteilungen*
5. *Buchhaltung und Fachabteilungen machen sich „zur Sicherheit“ wieder je eine Kopie*
6. *Fachabteilungen prüfen Rechnungsinhalt mit tatsächlicher Warenlieferung/-Dienstleistung*
7. *Fachabteilungen schicken Kopien mit Korrektheitsvermerk an die Buchhaltung*
8. *Buchhaltung macht eine Kopie der Kopie, da nun zusätzlich ein Vermerk angebracht wurde*
9. *Buchhaltung bereitet Rechnung zur Zahlung vor, und schickt eine Kopie zur Zahlungsgenehmigung an den Vorgesetzten*
10. *Genehmigung kommt in Buchhaltung zurück*
11. *Buchhaltung zahlt*
12. *Buchhaltung bringt Stempel auf Kopie und Original der Rechnung an*
13. *Rechnung landet in einer Ablage*

Obwohl die Komplexität dieser Vorgänge in diesem Beispiel vielleicht etwas überzeichnet dargestellt ist, gestaltet sich der Rechnungseingang, vor allem in größeren Unternehmen, alles andere als trivial, und erfordert einen nicht zu vernachlässigenden Personaleinsatz. Der Rechnungseingangsprozess im betrachteten Betrieb ist in „Abschnitt 3.2.4 – Rechnungseingang“ skizziert.

Rationalisierungspotential liegt hier nicht nur im vollständigen Verzicht auf Papier und der damit verbundenen Einsparung unzähliger Kopien, sondern auch in der schnelleren Bearbeitung der Rechnungen. Es muss kein physisches Objekt mehr von Abteilung zu Abteilung wandern, denn alles wird digital versandt und direkt am Bildschirm bearbeitet. Auch die Ablage der Rechnungen zur Archivierung ist in der digitalen Version platzsparend und unkompliziert. Mussten früher, um Papierrechnungen zu finden, noch

mühsam unübersichtliche Archive durchsucht werden, so hat man eine elektronische Rechnung innerhalb weniger Sekunden vor Augen.

Das durchdringendste Argument für die elektronische Rechnung auf der Empfängerseite ist allerdings die Möglichkeit einer automatisierten Weiterverarbeitung der Rechnungsdaten. Kommt eine Papierrechnung in der Buchhaltung oder in einer Fachabteilung an, so werden die Rechnungsdaten meist händisch erfasst[16] und die so gewonnenen Daten in ein Lagerverwaltungsprogramm eingegeben (oder ein Buchhalter bildet daraus einen Buchungssatz zum Verbuchen der Rechnung).

Ein geeignetes Format (siehe „4.3. Rechnungsformate“ auf S. 103) und die nötigen Werkzeuge vorausgesetzt, kann die elektronische Rechnung hier viele Schritte automatisieren, wie z. B. :

- Aufnahme der Rechnungsdaten in ein Warenwirtschaftssystem
- Erzeugung eines Buchungssatzes aus den Rechnungsdaten, welcher als Vorschlag für die Verbuchung dienen kann
- Übernahme der Rechnungsdaten in eine e-Banking-Applikation
- Überprüfung der Rechnungsdaten gegen gespeicherte Lieferschein- oder Bestelldaten

Im Gegensatz zum Rechnungssteller, welcher direkt durch Investitionen in seine Infrastruktur von der elektronischen Rechnung profitieren kann, sind die möglichen Einsparungen beim Empfänger von einigen Rahmenbedingungen abhängig. Damit z. B die in Tabelle 3.3 gelisteten Schritte bestmöglich vereinfacht werden können, sollten folgende Bedingungen erfüllt sein (nach ihrer Wichtigkeit sortiert):

1. Der Großteil der Eingangsrechnungen ist elektronisch verfügbar.
2. Die Rechnungssteller verwenden ein standardisiertes, und strukturiertes Format, welches eine automatisierte Weiterverarbeitung ermöglicht.
3. Die beim Empfänger vorhandenen FIBU-, ERP-, Workflow- oder e-Banking-Systeme unterstützen die von den Rechnungsstellern verwendeten Formate, sodass eine hohe Automatisierung möglich ist.
4. Die Lieferschein- oder Bestelldaten sind ebenfalls elektronisch gespeichert, sodass Rechnungen, dort wo es möglich ist, automatisiert geprüft werden können.

[16] In besser organisierten Unternehmen wird die Papierrechnung bereits im Sekretariat durch händisches Abtippen oder Einscannen digitalisiert. Den anderen Abteilungen wird dann die digitalisierte Version zur Verfügung gestellt.

Eine der wichtigsten Forderungen ist hier der Einsatz geeigneter Formate. Als geeignet kann ein Format dann bezeichnet werden, wenn es sich durch eine großflächige Verbreitung auszeichnet. Dabei ist nicht nur die geographische, sondern auch die branchenspezifische Verbreitung ausschlaggebend. Außerdem sollte es die Rechnungsdaten in standardisierter und strukturierter Form abbilden. PDF-Dateien oder die Datei-Formate der verschiedenen Office-Programme sind dafür nur eingeschränkt geeignet. Diese Formate sind zwar natürlich meist in irgend einer Form standardisiert, weisen aber nicht die Strukturierung auf die nötig wäre, damit verschiedene Systeme einfach und automatisiert Rechnungsdaten austauschen können. AustriaPro stellt mit dem Standard ebInterface (siehe „4.3.1. ebInterface“ auf S. 103) ein geeignetes Format für den standardisierten und strukturierten Rechnungsaustausch zur Verfügung.

Eine allgemeine Schätzung des Rationalisierungspotentiales auf der Rechnungseingangsseite gestaltet sich schwierig, hängen die Werte doch von der Größe des betrachteten Betriebes, und des bereits genutzten Automatisierungsgrades ab. Eine, laut eigenen Aussagen vorsichtige, Schätzung von AustriaPro stellt die Kosten für Papiereingangsrechnungen den Eingangsrechnungskosten mit ebInterface gegenüber (siehe Tabelle 3.6).

Kostenfaktor	Papierrechnung	ebInterface
Prozesskosten:		
Rechnungseingangserfassung	€ 0,36	€ 0,12
Rechnungsüberprüfung	€ 3,33	€ 3,33
Übernahme in ERP-System	€ 2,80	€ 1,26
Gesamt:	€ 6,70	€ 4,70
Jährliche Sachkosten:		
ebInterface	€ 0,00	€ 400,-

Tabelle 3.6.: Gegenüberstellung der Kosten/Rechnung beim Rechnungseingang [12]

Die jährlichen Sachkosten ergeben sich hier dadurch, dass ein ebInterface-fähiges System[17] angeschafft werden muss. Der Kalkulation von Tabelle 3.6 folgend, argumentiert AustriaPro, dass sich diese Investitionen schon ab cirka 250 empfangenen elektronischen Rechnungen pro Jahr amortisieren [12]. In der Vergleichstabelle (Tabelle 3.6) fällt auf, dass bei der Rechnungsüberprüfung keinerlei Kostenersparnisse veranschlagt wurden. Eventuell wären hier zusätzliche Einsparungen vorstellbar. Diese könnten etwa durch eine Unterstützung der Rechnungsprüfung in Form einer automatisierten Kontrolle der Rechnungsdaten gegen die gespeicherten Lieferschein- oder Bestelldaten erzielt werden.

[17] z. B. ein ERP- oder FIBU-System mit Wartungsvertrag

3.4.3. Fazit

Im Vergleich zur Papierrechnung sind durch die elektronische Rechnung sowohl beim Versender als auch beim Empfänger Einsparungen möglich.

Das mögliche Rationalisierungspotential beim betrachteten Betrieb liegt allerdings einige Größenordnungen unter den von verschiedenen Organisationen und Herstellern errechneten möglichen Einsparungen. Dies ist hauptsächlich dem Umstand zuzuschreiben, dass die Rechnungen hier nicht per Post versandt werden (siehe „3.2.2. Rechnungsversand“ auf S. 41), und somit ohnehin ein großer Teil der Kosten eingespart wird. Wie jedoch „Abschnitt 3.3 – Verbesserungspotential durch die e-Rechnung“ zeigt, würde die elektronische Rechnung dem betrachteten Betrieb noch viele weitere Vorteile bieten. Vor allem die Erleichterungen beim Rechnungsstellungsprozess, sowie die erwartete Erhöhung der Kundenzufriedenheit wiegen mindestens genau so schwer wie die Kostenersparnisse, und rechtfertigen eine Umstellung, trotz des niedrigeren Einsparungspotentiales.

Aus Sicht der Fallstudie, würden sich die angesprochenen Einsparungen beim Rechnungsempfang zum Teil sicherlich auch für den betrachteten Betrieb ergeben, da sowohl eine moderne FIBU-Software, als auch eine e-Banking-Applikation eingesetzt werden. Das Hauptaugenmerk dieser Arbeit liegt allerdings auf dem Rechnungsversand. Wie in diesem Abschnitt bereits angeführt, kann der Rechnungssteller hier aktiv Investitionen tätigen, von denen er dann auch direkt profitieren kann, während die Einsparungen beim Rechnungsempfänger von mehreren Faktoren abhängig sind. Auch dies ist ein Grund, warum im Zuge dieser Diplomarbeit im betrachteten Betrieb noch keine Maßnahmen auf der Rechnungseingangsseite gesetzt wurden.

Vielmehr sollten die möglichen Vorteile für den Rechnungsempfänger werbewirksam kommuniziert werden, um möglichst vielen Stammkunden den Mehrwert einer elektronischen Rechnung näher zu bringen. Hier könnte allerdings die Kundenstruktur des Unternehmens eine gewisse Hürde darstellen, denn grundsätzlich liegt der größte Nutzen für den Rechnungsempfänger in der automatisierten Weiterverarbeitbarkeit. Bei den Stammkunden des betrachteten Betriebes handelt es sich aber zum Großteil um Privatpersonen sowie Klein- oder Kleinstunternehmen (siehe „A.1. Einteilung von Unternehmen“ auf S. 153). Bei dieser Empfängerstruktur ist anzunehmen, dass der Vorteil der automatisierten Weiterverarbeitbarkeit nicht besonders stark zum Tragen kommen wird. Möglicherweise würde ein den Kunden besser bekanntes Format, wie z. B. das PDF-Format, als Rechnungsformat sogar besser akzeptiert werden, selbst wenn es im Bereich der automatisierten Weiterverarbeitbarkeit Defizite aufweist. Selbst bei einem Wegfall dieses größten Vorteiles für den Empfänger, sollten aber die anderen in „Abschnitt 3.3 – Verbesserungspotential durch die e-Rechnung“ genannten Argumente, wie

einfache Archivierung und Weitergabe, sowie der Komfort bei der Zustellung, Anreiz genug für eine Akzeptanz der e-Rechnung sein.

3.5. Soll-Zustand

In den letzten Abschnitten wurden die Probleme mit dem derzeitigen Rechnungsstellungsprozess geschildert und aufgezeigt, wie alle Beteiligten von der elektronischen Rechnung profitieren können. Daraus ergibt sich für den betrachteten Betrieb die Forderung nach einem System, welches die folgenden Eigenschaften erfüllt:

- Die in Tabelle 3.2 und Tabelle 3.1 farbig markierten Schritte sollen durch das e-Billing-System obsolet oder zumindest stark vereinfacht werden, was zu den in „Abschnitt 3.3 – Verbesserungspotential durch die e-Rechnung“ erwähnten Vorteilen und zu den in „Abschnitt 3.4.1 – Einsparungen für den Rechnungssteller“ errechneten Einsparungen führen soll.

- Es sollen Formate verwendet werden, welche dem Kunden die automatisierte Weiterverarbeitung der Rechnungen ermöglichen. Alternativ dazu soll aber auch ein bekanntes und weit verbreitetes Format, wie z. B. das PDF-Format, unterstützt werden, selbst wenn dieses Kriterium dadurch nicht erfüllt wird.

- Alle gesetzlichen Anforderungen (siehe „2. Rechtliche Sicht“ auf S. 17) für vorsteuerabzugsfähige Rechnungen müssen eingehalten werden.

- Das System muss flexibel genug sein, um bei Änderungen der gesetzlichen, organisatorischen oder betriebswirtschaftlichen Anforderungen leicht angepasst werden zu können.

- Das System soll sich in die vorhandene Infrastruktur einfügen und mit den vorhandenen Systemen zusammenarbeiten.

- Es soll einfach zu bedienen sein.

- Für die Zustellung der elektronischen Rechnungen sollen mehrere Alternativen geboten werden. Vorstellbar sind z. B. eine Zustellung im Anhang einer Email-Nachricht oder auch die Ablage im Kundenbereich[18] einer Webseite.

- Das System soll, soweit dies möglich ist, zusätzlich das Errechnen der Jahresrabatte übernehmen können.

[18] Zu welchem der Kunde durch Eingabe eines Benutzernamens und eines Passwortes Zugang erlangen kann.

„Abschnitt 3.6.2 – e-Billing-Umsetzungsmöglichkeiten“ zeigt welche Möglichkeiten für den betrachteten Betrieb bestehen, um diesen Soll-Zustand zu erreichen.

3.6. Umsetzungsmöglichkeiten

Dieser Abschnitt beleuchtet, welche Möglichkeiten denkbar sind, um den eben beschriebenen Soll-Zustand zu erreichen. Zuerst wird der Rechnungsbearbeitungsprozess beim Versender sowie beim Empfänger in verschiedene Phasen unterteilt, um die anschließende Diskussion der Umsetzungsmöglichkeiten zu vereinfachen.

3.6.1. Phasen der Rechnungsbearbeitung

Abbildung 3.3 zeigt die Phasen der Rechnungsbearbeitung sowohl für traditionelle als auch für elektronische Rechnungen beim Rechnungssteller wie auch beim Empfänger. Diese einzelnen Phasen können sowohl bei der traditionellen Papierrechnung und bei der elektronischen Rechnung durch geeignete Applikationen oder Dienste unterstützt werden (siehe „3.6.2. e-Billing-Umsetzungsmöglichkeiten“ auf S. 59). Es können dabei einzelne oder auch mehrere Phasen abgedeckt werden. Der dunkelfarbig markierte Teil der Grafik zeigt jene Phasen, welche von einer Applikation oder einem Dienst unterstützt werden sollen, um im Zuge der Fallstudie den beschriebenen Soll-Zustand zu erreichen. Ein(e) in Frage kommende(r) Service, bzw. Applikation sollte diese Phasen möglichst gut integrieren, und zusätzlich eine einfache Bedienung ermöglichen.

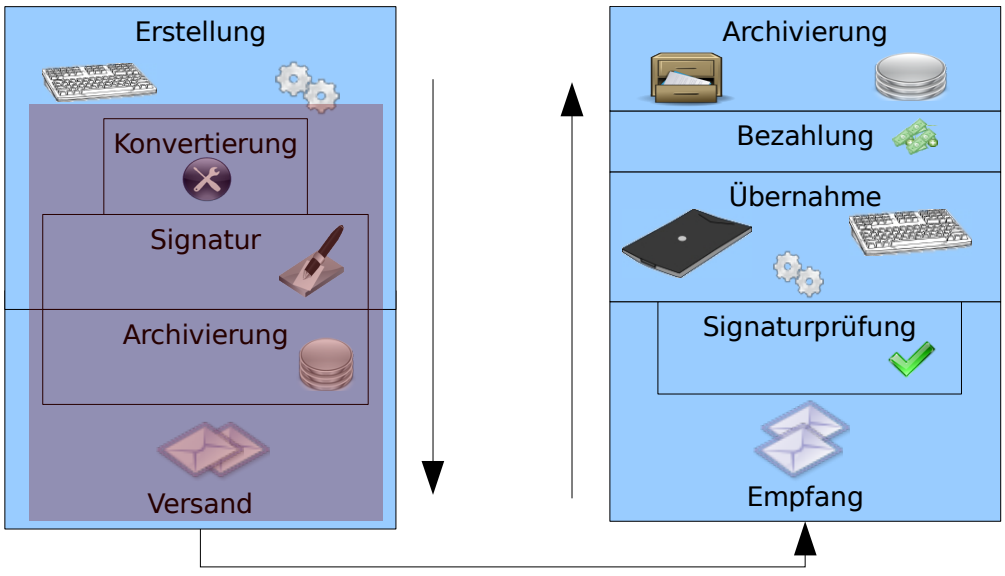

Abbildung 3.3.: Phasen der Rechnungsbearbeitung

Im Folgenden werden die einzelnen Phasen kurz diskutiert. Applikationen und Dienste zu deren Unterstützung werden in „Abschnitt 3.6.2 – e-Billing-Umsetzungsmöglichkeiten“ genannt.

Erstellung Rechnungen können einerseits einzeln händisch erstellt werden, oder andererseits von einem ERP- oder Fakturierungs-System automatisch aus dort gespeicherten Daten generiert werden. Einzelne elektronische PDF-Rechnungen können z. B. mit gängigen Textverarbeitungsprogrammen wie OpenOffice erzeugt werden. Zur Erzeugung von ebInterface-Rechnungen, stehen ebenfalls bereits einige Werkzeuge zur Verfügung. So bietet z. B. AustriaPro dazu eine Web-Applikation namens Webforms[19] an, und die Firma NTx ein Plugin zur Erstellung von ebInterface-Rechnungen mit Microsoft Word 2007.[20]

Sobald eine Papierrechnung erstellt wurde, ist der darauf folgende Schritt meist der Versand. Bei elektronischen Rechnungen kann man noch einige Zwischenphasen identifizieren, bevor es zum Versand kommt.

Konvertierung Konvertierungen sind nötig, wenn die erstellten Rechnungen nicht in dem Format vorliegen, in welchem sie versandt werden sollen. Dies kann z. B. der Fall sein, wenn Rechnungen aus einem älteren System exportiert werden, welches ursprünglich nicht dazu vorgesehen war, elektronische Rechnungen zu verschicken. Dieser Umstand trifft in der betrachteten Fallstudie zu.

Wie diese Phase von Applikationen und Diensten unterstützt werden kann, hängt vor allem von den verwendeten Formaten ab.

Signatur Dabei werden elektronische Rechnungen entweder einzeln, oder mehrere Rechnungen in einem Schritt (siehe „2.1.2. Server- und Massensignatur“ auf S. 22) digital signiert.

Archivierung Wird die Rechnung elektronisch versandt, so muss sie nicht nur vom Empfänger sondern auch vom Rechnungssteller für 7 Jahre aufbewahrt werden.

Versand Papierrechnungen werden in der Regel per Post versandt, was neben den Portogebühren auch teure Verarbeitungsschritte wie Drucken, Falten, Kuvertieren und Frankieren mit sich bringt.

Elektronische Rechnungen können z. B. als Email-Anhang versandt werden. Auch die Einspeisung in eine Web-Plattform ist eine populäre Verteilungsmöglichkeit. Dem Empfänger könnte auch die Möglichkeit geboten werden, sich Rechnungen per Web-Service abzuholen.

[19] `http://www.ebinterface.at/webforms/`
[20] NTx eBilling@Word (`http://www.ntx.at/Services/Seiten/NTxSoftware.aspx`)

Empfang Wie der Empfänger die Rechnungen erhält, hängt natürlich von der Versandart ab. Für den Empfänger gut automatisieren lassen sich z. B. die Abholung aus einem Email-Postfach oder per Webservice.

Signaturprüfung Signaturen auf elektronischen Rechnungen können z. B. mit jener Applikation überprüft werden, welche auch zum Ansehen der Rechnungen verwendet wird. Beispielhaft sei hier der „Adobe Acrobat Reader“ zur Signaturverifikation von PDF-Rechnungen genannt. Die Überprüfung kann, vor allem bei der Verwendung automatisiert weiterverarbeitbarer Formate, aber auch in einem ERP-System integriert sein.

Übernahme Bei Papierrechnungen oder bei Verwendung nicht automatisiert weiterverarbeitbarer Formate müssen die Rechnungsdaten händisch abgetippt oder eingescannt werden. Rechnungen in dazu geeigneten Formaten können dagegen von ERP- oder FIBU-Applikationen automatisiert verarbeitet werden.

Bezahlung Elektronische Rechnungen in einem automatisiert weiterverarbeitbaren Format können von einigen e-Banking-Applikationen übernommen und per Knopfdruck bezahlt werden.

Archivierung Rechnungsempfänger müssen elektronische Rechnungen, ebenso wie Papierrechnungen, 7 Jahre lang aufbewahren.

3.6.2. e-Billing-Umsetzungsmöglichkeiten

Dieser Abschnitt soll dem Leser eine grobe Idee vermitteln, welche Möglichkeiten es gibt, um Rechnungsprozesse auf die elektronische Rechnung umzustellen, und welche davon für ihn in Frage kommen könnte.

Bei der Betrachtung dieser Möglichkeiten werden auch einige Beispiele für e-Billing-Produkte oder -services genannt, deren Auswahl willkürlich getroffen wurde. Der Versuch eine Aussage über die Qualität dieser Produkte und Dienste zu machen, würde den Rahmen dieses Abschnittes sprengen. Es wurde deshalb auch bewusst darauf verzichtet, Produkte oder Services zu vergleichen. Tabelle A.3 bietet lediglich eine Übersicht über e-Billing-Anbieter und ihre Applikationen bzw. Dienste. Dies soll über die Marktlage informieren und als Startpunkt für weitere Nachforschungen dienen. In diesem Abschnitt werden auch Preise für Services und Produkte genannt, die den Webseiten, Produktbeschreibungen oder Aussagen der Anbieter entnommen wurden und zum Zeitpunkt der Erstellung dieser Arbeit gültig sind bzw. waren.

3.6.2.1. e-Billing auslagern

Wie viele andere IT-Prozesse auch lassen sich e-Billing-Prozesse outsourcen. Die Rechnungsstellung durch Dritte ist sowohl durch `RL 2001/115/EG Art 2 Abs 2`, als auch durch die österreichischen Umsatzsteuerrichtlinien[21] explizit erlaubt. In den Umsatzsteuerrichtlinien ist zur Rechnungsstellung durch Dritte festgehalten:

> *„... werden eine oder mehrere natürliche Personen beim Dritten bevollmächtigt, für den leistenden Unternehmer oder im Fall der Gutschrift[22] für den Leistungsempfänger Rechnungen mit einer elektronischen Signatur zu versehen.“*

Dies bedeutet meist, dass die Rechnungen des Leistungsbringers durch einen Dritten mit dem Zertifikat des Dritten signiert werden.[23] Dies befreit den Leistungsbringer davon sich mit Themen wie digitaler Signatur überhaupt auseinandersetzen zu müssen. Gleichzeitig führt es aber zu dem z. B. in [34] kritisierten Zustand, dass eine Signatur nur zum Signator und nicht zum Leistungsbringer führt. Deshalb trennt dieser Abschnitt auch die Begriffe Leistungsbringer als denjenigen, der mit der Rechnung eine Forderung dokumentiert, und Rechnungssteller als denjenigen, der die Rechnung im Namen des Leistungsbringers stellt.

Für die Rechnungsstellung durch Dritte werden verschiedene Dienste angeboten. Meist ist in den Angeboten auch schon die Archivierung der Rechnungen inkludiert.

Besonders interessant sind jene Dienste, bei denen der Leistungsbringer die Rechnungen in dem von seinem System erzeugten Format abliefern kann, und diese dann vom Serviceanbieter in das gewünschte Format umgewandelt werden. Der in „Abschnitt 3.1 – Vorstellung des Unternehmens“ betrachtete Betrieb verfügt beispielsweise über ein Altsystem, welches alle Rechnungen eines Monats in einer Textdatei exportiert. Statt des Rechnungsdruckes würde nun diese Datei an den Serviceanbieter gesendet[24], welcher alle weiteren Schritte erledigen würde. Wird dieser Service nicht geboten, muss sich der Leistungsbringer selbst darum kümmern, die Rechnungen in dem von ihm gewünschten, und vom Diensteanbieter unterstützten Format bereitzustellen.

Oft ist es nicht möglich, sämtliche Kunden für die elektronische Rechnung zu gewinnen. In solch einem Fall müssen einzelne Rechnungen weiter auf Papier zugestellt wer-

[21] `RZ 1564b`

[22] Anm.: bezieht sich hier auf die in den Umsatzsteuerrichtlinien ebenfalls explizit erlaubte Rechnungsstellung im Gutschriftverfahren durch den Leistungsempfänger

[23] Technisch gesehen ist diese Aussage nicht vollständig korrekt. Siehe dazu auch „Abschnitt 4.2 – Digitale Signatur“.

[24] Glücklicherweise ist in den Umsatzsteuerrichtlinien festgehalten, dass die in „Abschnitt 2 – Rechtliche Sicht“ beschriebenen Anforderungen, für diesen Rechnungsversand vom Leistungsbringer zum rechnungsstellenden Dritten nicht anzuwenden sind.

den. Hier kann es eine große Erleichterung sein, wenn der Diensteanbieter auch diese Rechnungen elektronisch übernimmt, daraus Papierrechnungen erstellt, und diese den e-Billing-verweigernden Empfängern z. B. postalisch zustellt.

Abgerechnet wird bei den meisten Angeboten pro Rechnung. Der Leistungsbringer hat also keine Fixkosten zu tragen, zahlt aber für jede versandte Rechnung einen gewissen Obolus.

Im Folgenden werden einige Angebote kurz vorgestellt.

EBPP[25]

Die Firma „EBPP Electronic Bill Presentment and Payment GmbH“ ist ein Tochterunternehmen dreier österreichischer Großbanken. EBPP bietet hauptsächlich zwei verschiedene Dienste an, nämlich „e-Rechnung Mail“ und „e-Rechnung Consolidator“. Nach Aussagen von Herrn Mag. Gschwandtner[26] kann die Firma EBPP aber darüberhinaus auch weitere Phasen der Rechnungsbearbeitung unterstützen, wie z. B. das Digitalisieren von Papierrechnungen für Rechnungsempfänger.

e-Rechnung Mail

Dabei handelt es sich laut Aussagen des Anbieters [39] um ein *„Dokumentenübermittlungssystem auf Basis von Mails“*. Technisch betrachtet schickt der Leistungsbringer Mails an seine Rechnungsempfänger über einen speziellen e-Rechnung-Mailserver. Die Rechnung, oder ein beliebiges anderes Dokument in einem beliebigen Format [39], befindet sich im Anhang der Mail und wird am Mailserver mit einer fortgeschrittenen digitalen Signatur versehen, bevor es den Rechnungsempfänger erreicht [38]. Laut Produktbeschreibung erfolgt die Signatur entweder mit dem Zertifikat der EBPP GmbH oder mit einem eigenen Zertifikat des Leistungsbringers. Nach [38] werden die Rechnungen dabei nicht gespeichert. Laut Herrn Mag. Gschwandtner[26] rät EBPP aber all seinen Kunden, sich selbst als Empfänger einer Rechnungskopie einzutragen, und die Rechnung dann mit den sonstigen erhaltenen Emails zu archivieren. [38] beschreibt auch das Anliefern der Rechnungen in ein bestimmtes Verzeichnis[27] als alternative Schnittstelle.

Dieser Service kostet derzeit laut Anbieterinformation pro Rechnung € 0,45 exkl. USt.

[25] `http://www.e-rechnung.at`

[26] Emailverkehr vom 17.03.2007 mit Herrn Mag. Wolfgang Gschwandtner, Geschäftsführer der EBPP GmbH

[27] Wahrscheinlich ist hier ein FTP-Verzeichnis auf einem Server von EBPP gemeint.

e-Rechnung Consolidator

Hierbei handelt es sich laut Aussagen des Anbieters [39] um ein *„Clearing System für Dokumente“*. Dieses System verwendet intern für Rechnungen ein XML-Format namens PBInvoice[28], welches dem Format ebInterface in gewisser Weise ähnlich ist. Die Verbindung zwischen PBInvoice und ebInterface wird in „Abschnitt 4.3.1 – ebInterface“ erläutert. Laut Herrn Mag. Gschwandtner[26] hat das EBPP-System ebInterface aber ebenfalls als natives Format eingebettet.

Bei diesem System sollte der Leistungsbringer seine Rechnungen in einem von EBPP unterstützten Format einspeisen. Dies sind Formate, für die ein Mapping nach PBInvoice definiert ist.[28] Laut Herrn Mag. Gschwandtner[26] werden vereinzelt z. B. aber auch Textdateien angeliefert, die dann vor der weiteren Bearbeitung konvertiert werden. Die Rechnungen werden vom System mit einer fortgeschrittenen XML-Signatur (siehe „4.3.1.4. Signatur von XML-Dokumenten“ auf S. 113) versehen und den gesetzlichen Anforderungen entsprechend, sowohl für den Rechnungssteller, als auch für den Rechnungsempfänger[26] aufbewahrt. Die Signatur wird dabei mit dem Zertifikat der EBPP GmbH erzeugt. Die Anlieferung kann per Webservice[29], FTP oder Connect:Direct[30] erfolgen.

Laut [37] werden in diesem System die Rechnungssteller, die Banken und die Rechnungsempfänger zusammengeführt. Dies wird dadurch erreicht, dass die signierten Rechnungen direkt ins Internet-Banking des Rechnungsempfängers geliefert werden. Dieser kann dort neben seinen Konten und Überweisungen auch seine Eingangsrechnungen einsehen und diese direkt zur Bezahlung freigeben. Eine Auflistung der Banken, deren Systeme mit e-Rechnung Consolidator zusammenarbeiten, findet sich unter [39]. Für Rechnungsempfänger mit abweichenden Bankverbindungen stellt der Anbieter eine eigene neutrale Web-Oberfläche zum Einsehen der Rechnungen zur Verfügung.

Über e-Rechnung Consolidator werden nicht nur Rechnungen sondern z. B. auch Gehaltszettel zugestellt.

Die Preise für den „e-Rechnung Consolidator“-Service sind laut Herrn Mag. Gschwandtner[26] nach dem Rechnungsaufkommen gestaffelt und betragen höchstens 56 Cent exkl. USt, wenn eine reine Rechnungspräsentation gewünscht ist, und 78 Cent exkl. USt, wenn die direkte Bezahlung der Rechnungen im Internet-Banking des Empfängers möglich sein soll.

[28] Beschreibung unter `https://www.e-rechnung.at/docs/Rechnungsformate_2.0.pdf`
[29] Genauer: per HTTPS-POST eines XML Request
[30] Siehe dazu `http://www.sterlingcommerce.de/Products/MFT/ConnectDirect/`

Flexdoc[31]

Bei Flexdoc handelt es sich um eine Plattform zum Dokumenteversand, welche von der Firma „GRZ IT Center Linz“ betrieben wird. Neben Dokumenten wie Bestellungen oder Mahnungen können über diese Plattform auch Rechnungen ausgetauscht werden.

Das Interessante an diesem System ist der ganzheitliche Ansatz, denn es handelt sich dabei nicht nur um ein System für Leistungsbringer, sondern auch für Leistungsempfänger. Rechnungen können vom Leistungsbringer unter anderem mittels eines speziellen Druckertreibers an Flexdoc geliefert werden. Laut Auskunft von Herrn Strobl[32] vom GRZ IT Center Linz verwendet Flexdoc den Druckdatenstrom. Dadurch ist es möglich, dass alles was gedruckt werden kann, auch an Flexdoc gesendet werden kann. Der Leistungsbringer verfährt dabei mit einer Rechnung genauso, als würde er sie ausdrucken. Anstatt die Rechnung aber an den Drucker zu schicken, wird diese an Flexdoc übermittelt. Nach Auskunft des Betreibers soll es noch weitere Möglichkeiten geben, die Rechnungen bei Flexdoc abzuliefern.

Ist eine Rechnung im System eingelangt wird entschieden, auf welche Art sie dem Empfänger zugestellt werden soll. Ist der Empfänger ebenfalls bei der Flexdoc-Plattform registriert, wird die Rechnung über die Plattform ausgeliefert. In diesem Fall wird die Rechnung innerhalb von Flexdoc sowohl für den Versender, als auch für den Empfänger gesetzeskonform archiviert. Sollte der Empfänger nicht beim System registriert sein, aber seine Emailadresse ist bekannt, so wird die Rechnung auf diesem Weg zugestellt. In beiden Fällen wird die elektronische Rechnung dann mit einer fortgeschrittenen digitalen Signatur versehen. Flexdoc setzt laut Aussagen des Betreibers [63] dazu die Signaturtechnik „xyzmo Seal“[33] ein. Laut Auskunft von Herrn Strobl[32] wird die Rechnung dabei immer als signiertes PDF versandt. Ist der Rechnungsempfänger aber ebenfalls bei Flexdoc registriert, kann er auch verschiedene andere strukturierte Empfangsformate wie z. B. ebInterface auswählen. Kann der Empfänger nicht auf elektronischem Weg erreicht werden, wird die Rechnung vom Diensteanbieter gedruckt und postalisch zugestellt.

Für Rechnungsempfänger die bei Flexdoc registriert sind, wird noch ein weiterer Service angeboten. Diese können auch alle erhaltenen Papierrechnungen und elektronische Rechnungen in unstrukturierten Formaten an Flexdoc schicken. Diese werden dann digitalisiert und in einem automatisiert weiterverarbeitbaren Format an den Rechnungsempfänger zurückgesendet.

Kosten für den Rechnungssteller fallen nur pro versandter Rechnung an und sind je nach Rechnungsaufkommen gestaffelt, betragen aber laut Herrn Strobl[32] höchstens € 0,38

[31] `https://www.flexdoc.at/`
[32] Laut Mailverkehr vom 26. März 2008
[33] `http://www.xyzmo.com/`

exkl. USt pro Rechnung. Allerdings sind, sollte eine Rechnung von Flexdoc postalisch verschickt werden, natürlich noch die Portogebühren hinzu zu addieren. Möchte der Empfänger die Rechnungen in einem strukturierten elektronischen Format zugestellt bekommen, werden die Kosten dafür auch ihm zugerechnet und sind abhängig von der Komplexität der jeweiligen Umwandlung.

Telekom Austria[34]

Auch die Telekom Austria bietet e-Billing-Services an. Dabei unterscheidet die Telekom laut Webseite zwischen Diensten für KMU[35] und Diensten für größere Unternehmen[36]. Leider können im Folgenden zu diesen beiden Diensten nur die auf der Webseite des Anbieters vorhandenen Informationen zusammengefasst werden, da es trotz versuchter Kontaktaufnahme von Seiten des Autors keine Antworten des Diensteanbieters auf Detailfragen zu den angebotenen Services gab.

Die kleine Variante ist speziell für KMU gedacht. Das Angebot ist ähnlich zu e-Rechnung Mail der Firma EBPP. Rechnungen werden dabei mit dem Zertifikat der Telekom Austria fortgeschritten signiert und anschließend per Email versandt. Außerdem werden die Rechungen für den Leistungsbringer gesetzeskonform archiviert. Unterstützt werden laut [66] die Formate ebInterface und PDF, wobei die Rechnungen vom Leistungsbringer vorbereitet werden müssen. Eine Konvertierung zwischen Rechnungsformaten ist nicht vorgesehen. Schnittstelle zum Service ist eine Web-Oberfläche, in welcher der Leistungsbringer die Kontaktdaten der Rechnungsempfänger verwalten und die Rechnungen hochladen kann.

Das Angebot richtet sich anscheinend wirklich im Besonderen an KMU, da laut der in [66] gezeigten Videopräsentation offensichtlich jede Rechnung einzeln importiert und versandt werden muss. Auch die Preisstaffelung lässt diesen Schluss zu. Mit dem kleinsten Paket können für € 10,00 exkl. USt pro Monat 20 Rechnungen versandt werden, während das größte angebotene Paket einen Versand von 100 Rechnungen pro Monat für € 30,00 exkl. USt erlaubt. Werden diese Grenzen überschritten, so werden laut [66] die zuviel versandten Rechnungen gesondert abgerechnet.

Die große Variante richtet sich anscheinend eher an Leistungsbringer, welche ihre Rechnungsdaten aus einem vorhandenen System in einem gewissen Format exportieren können. Aus den Rechnungen, in diesem laut [67] *„zu vereinbarenden Format“*, werden dann vom System der Telekom PDF- oder XML-Rechnungen erstellt. Die Einlieferung erfolgt laut [67] durch Upload in einem Web-Formular. Es wäre aber denkbar, dass hier noch weitere, besser automatisierbarere Möglichkeiten angeboten werden.

[34] `http://www.telekom.at`
[35] Hier „kleine Variante“ genannt, siehe dazu [66]
[36] Hier „große Variante“ genannt, siehe dazu [67]

Angelieferte Rechnungen werden vom System mit dem Zertifikat des Kunden, welches laut [67] *„am Rechner der Telekom Austria hinterlegt ist“*, signiert.

Welche Phasen der Rechnungsverarbeitung weiters unterstützt werden hängt davon ab, welche der in [67] beschriebenen Varianten gewählt wurde. So können Rechnungen nach der Signatur wieder vom Leistungsbringer übernommen werden, der sich dann selbst um Versand und Archivierung kümmert. Die Telekom übernimmt für den Leistungsbringer aber auch zusätzlich jeweils den Versand per Email, die Archivierung oder auch beides.

3.6.2.2. Standardsoftware nutzen

Es gibt ein großes Angebot an Standardsoftware um die in Abbildung 3.3 gezeigten Phasen der Rechnungsbearbeitung zu unterstützen. Einige Applikationen sind auf einzelne Phasen spezialisiert, während andere wiederum Unterstützung für mehrere Phasen bieten.

Vor allem ERP-Systeme sind bestens dafür geeignet, einen Großteil der in „Abschnitt 3.6.1 – Phasen der Rechnungsbearbeitung“ genannten Phasen abzudecken. Sie können Rechnungen aus den gespeicherten Daten erzeugen oder Rechnungsdaten exportieren. Moderne ERP-Systeme bieten oft auch die Erzeugung und den Versand von automatisiert weiterverarbeitbaren elektronischen Rechnungen an. Werden solche Formate verwendet, bieten sich solche ERP- und FIBU-Systeme auch für die Weiterverarbeitung der Rechnungen auf Seiten des Empfängers an. Der Verein AustriaPro (siehe „1.4. AustriaPro“ auf S. 15) bietet eine Übersicht von Unternehmen, die an den Projekten ebInterface und ebInvoice beteiligt sind, und entsprechende Produkte anbieten.[37]

Sollte ein ERP-System keine Möglichkeit zur Rechnungssignatur bieten, so erlaut es wahrscheinlich aber zumindest den Emailversand von Rechnungen im PDF- oder XML-Format. Hier setzen viele Signaturprodukte an, welche in Form eines Mailproxys oder Mailservers eingehende Emails oder deren Anhänge signieren und an den Empfänger weiterversenden. Oft werden auch Datei-Schnittstellen unterstützt, die Dokumente signieren, wenn diese in einem gewissen Verzeichnis abgelegt werden. Andere Signaturapplikationen werden als Druckertreiber in die Rechnungsprozesse integriert. Dabei wird ein Teil der Signaturapplikation als Drucker installiert, und kann in einem Programm als solcher angewählt werden. Anstatt die Rechnung aber auf Papier auszudrucken wird das Dokument z. B. in das PDF-Format umgewandelt und signiert. Eine weitere Integrationsmöglichkeit bietet auch die Anbindung über Webservices.

[37] `http://www.ebinterface.at/partner.html`

Zur Erstellung einzelner elektronischer Rechnungen im Format ebInterface wurden bereits in „Abschnitt 3.6.1 – Phasen der Rechnungsbearbeitung“ die frei erhältlichen Applikationen ebInterface WebForms und NTx eBilling@Word genannt. Beide bieten auch die Möglichkeit, erstellte Rechnungen zu signieren. Bei NTx eBilling@Word ist dies allerdings nur in einer kostenpflichtigen Variante möglich.

Bei der Signaturprüfung kann ebenfalls aus einer großen Menge von angebotenen Applikationen gewählt werden. Sollen nur wenige elektronische Eingangsrechnungen geprüft werden, so kann dies manuell mit Programmen erfolgen, die meist sogar kostenfrei erhältlich sind. So können PDF-Signaturen z. B. mit dem Adobe Acrobat Reader geprüft werden. Für signierte ebInterface-Rechnungen wird ebenfalls eine kostenfreie Applikation namens ebRe[38] zur Verfügung gestellt. Sollen sehr viele Rechnungssignaturen geprüft werden, bieten die Signaturserver einiger Hersteller die Möglichkeit einer automatisierten Signaturprüfung. Dazu kann z. B. ein Emailpostfach auf eingehende e-Rechnungen überwacht werden. Wird ein Eingang erkannt, erfolgt eine automatische Signaturprüfung.

„Anhang A.2 auf S. 154“ zeigt einen Überblick über einige am Markt erhältliche e-Billing-Applikationen.

3.6.2.3. Eigenentwicklung einer e-Billing-Software

Eine e-Billing-Software für den eigenen Gebrauch selbst zu entwickeln, kommt wohl nur für die wenigsten KMU wirklich in Frage, da dazu umfassendes Wissen in den folgenden Bereichen notwendig ist:

- Rechtliche Anforderungen an elektronische Rechnungen
- Funktionsweise der digitalen Signatur
- Technische Details von Rechnungsformaten
- Programmiersprachen
- Frameworks im Umfeld der jeweiligen Programmiersprachen
- Kryptographie-APIs der jeweiligen Programmiersprachen

[38] Erhältlich unter `http://portal.wko.at/wk/format_detail.wk?AngID=1&StID=316012&DstID=0&BrID=0`

Außerdem müssen beim Einsatz von selbstentwickelter Software auch Änderungen an gesetzlichen Anforderungen immer im Auge behalten werden. Verwendet man zugekaufte Software oder Services, wird einem diese Aufgabe größtenteils abgenommen.

Die Vorteile einer selbstentwickelten Applikation liegen natürlich darin, dass diese speziell für den Anwendungsfall entworfen werden kann und sich daher natürlich hervorragend in die vorhandenen Arbeitsabläufe eines Unternehmens eingliedern lässt.

Wie in der modernen Softwareentwicklung üblich, kann man sich auch beim Bau einer e-Billing-Anwendung auf bereits vorhandene Bausteine stützen. Einige dieser Bausteine und Bibliotheken werden in „Abschnitt 4 – Technische Sicht“ beschrieben.

3.6.3. Auswahl eines Zertifikats

In den letzten drei Abschnitten wurden die Möglichkeiten behandelt, welche sich Unternehmen bei einer Umstellung auf die elektronische Rechnung bieten. Ist die Entscheidung auf das Outsourcen des e-Billing-Prozesses gefallen, wird sich der Leistungsbringer kaum mit der Auswahl eines Zertifikats auseinandersetzen müssen. Sollte dies doch der Fall sein, so wird er sicher auf die Unterstützung des Serviceanbieters bauen können. Hat sich der Rechnungssteller aber dafür entschieden, e-Billing-Standardsoftware einzusetzen, oder sollen eigene Applikationen entwickelt werden, muss nach derzeitiger Rechtslage auch ein Zertifikat angeschafft werden.

Um fortgeschrittene digitale Signaturen zu erstellen, bedarf es geeigneter Zertifikate eines Zertifizierungsdiensteanbieters. Eine Liste österreichischer ZDA wird von der RTR GmbH[39] geführt. Seit Inkrafttreten des neuen SigG (siehe „A.3. Änderungen im SigG und die neue SigV“ auf S. 158) werden dort nur mehr ZDA für qualifizierte Zertifikate gelistet. Allerdings ist auch eine historische Liste vom 31.12.2007 abrufbar, welche auch andere ZDA enthält.

Um für die Signatur von Rechnungen geeignet zu sein, müssen Zertifikate zur Zeit in den eben genannten Listen mit den Prädikaten „qualifiziert“ (QF) oder „Fortgeschrittene elektronische Signatur“ (F) gekennzeichnet sein.[40] Theoretisch könnten auch gleichwertige ausländische Zertifikate verwendet werden, worauf hier aber nicht eingegangen wird.

Wie in „Abschnitt 4.2 – Digitale Signatur“ genauer erläutert wird, weist ein Zertifikat einen Signator aus. Ein Rechnungsempfänger kann somit feststellen, ob die Rechnung wirklich von dem im Zertifikat ausgewiesenen Signator stammt. Es ist dabei nicht zwingend nötig, dass der in der Rechnung ausgewiesene Name des Leistungsbringers auch

[39] `http://www.signatur.rtr.at/de/providers/providers.html`
[40] Die Liste der RTR GmbH lässt sich auch nach diesen Kriterien sortieren.

im Zertifikat aufscheint. Genau diese Situation kommt eben dann vor, wenn Rechnungen durch einen Dienstleister signiert werden. Das bedeutet, dass z. B. Rechnungen des betrachteten Betriebes auch mit dem Zertifikat eines „Max Mustermann“ signiert werden könnten. Es ist dann das Problem des Rechnungsempfängers, herauszufinden, ob die nötigen Befugnisse gegeben sind.

Deshalb sollte Wert darauf gelegt werden, die Rolle des Signators für den Rechnungsempfänger möglichst transparent zu gestalten. Bei Einzelunternehmen kann zwar der Name des Signators, der im Firmenwortlaut vorkommen muss, ausreichen um diese Transparenz und das nötige Vertrauen zu schaffen. Trotzdem sollte danach getrachtet werden, den Firmenwortlaut in das Zertifikat aufzunehmen (siehe „2.1.2. Server- und Massensignatur“ auf S. 22). Dieser kann als Pseudonym, oder auch in das „OrganizationName“-Attribut (O) im DN-Feld eingetragen werden. Des weiteren ist es oft auch möglich, im „OrganizationalUnitName“-Attribut (OU) den Namen einer Abteilung anzugeben, oder festzulegen, für welche Aufgaben der Signator vertretungsbefugt ist. Angaben über eine Vertretungsmacht müssen dem ZDA nachgewiesen werden [59]. Dazu kann auch die Vorlage einer Vollmacht notwendig sein. Es sollte allerdings erwähnt werden, dass nicht alle Zertifikatsprodukte alle Möglichkeiten zur Aufnahme des Firmenwortlautes zulassen.

Es werden nun zwei österreichische ZDA und einige ihrer Produkte kurz vorgestellt.

A-Trust[41]

Die „A-Trust GmbH“ ist Österreichs einziger akkreditierter ZDA, und zur Zeit der einzige österreichische ZDA, der qualifizierte Zertifikate ausstellt. Zu den Gesellschaftern der A-Trust gehören laut Unternehmens-Webseite unter anderem auch einige österreichische Banken sowie die WKÖ.

a.Sign premium

Bei diesem Angebot handelt es sich eigentlich um zwei Zertifikate, die auf einer Smart-Card aufgebracht werden. Dazu kann z. B. auch eine bereits vorhandene Bankomatkarte verwendet werden, wenn diese z. B. durch den Aufdruck „a sign premium“ entsprechend gekennzeichnet ist. Bei den beiden Zertifikaten handelt es sich um ein qualifiziertes Signaturzertifikat, und ein fortgeschrittenes[42] Verschlüsselungszertifikat. Beide sind zum Signieren von Rechnungen geeignet.

[41] `http://www.a-trust.at`

[42] Zertifikate zur Erstellung von fortgeschrittenen Signaturen werden zur Vereinfachung des Satzbildes in dieser Arbeit einfach „fortgeschrittene Zertifikate“ genannt.

Die Angabe eines Firmenwortlautes im OrganizationName-Attribut des Zertifikates ist bei diesem Produkt nicht möglich. Allerdings sollte es möglich sein, den Firmenwortlaut als Pseudonym in das Zertifikat aufzunehmen. Ob dies akzeptiert wird, hängt vom ZDA ab.[43] Herr Gepp von der Firma A-Trust hat in einem Experteninterview allerdings bestätigt, dass einem solchen Antrag meist statt gegeben wird.

Da die Zertifikate bei a.Sign premium auf einer SmartCard gespeichert werden, muss auch ein SmartCard-Lesegerät vorhanden sein. Allerdings sind nicht alle Lesegeräte dazu geeignet, qualifizierte Stapelsignaturen zu erzeugen. Ein Gerät, welches dazu in der Lage ist, wird in „Abschnitt 4.2 – Digitale Signatur“ erwähnt, und ist auch in Tabelle 3.7 aufgelistet.

Tabelle 3.7 zeigt eine Auflistung der Kosten, um mit a.sign premium effizient qualifizierte Rechnungssignaturen zu erstellen.

Einmalige Kosten:[a]	
Registrierung und Zertifikatserstellung	€ 12,00
Kartenlesegerät „ReinerSCT cyberJack pinpad stapelsignatur“	€ 186,00
Gesamt:[b]	€ 198,00
Jährliche Kosten:[a]	
Zertifikat	€ 15,60
Gesamt:	€ 15,60

[a] Alle Beträge inkl. USt
[b] Sollen die Zertifikate nicht auf einer vorhandenen Karte aufgebracht werden, so müssen noch € 30,- inkl. USt pro Karte addiert werden.

Tabelle 3.7.: Kosten a.Sign premium

e-card

Als Ersatz der mit Ende 2007 ausgelaufenen Verwaltungssignatur, ist es nun auch möglich, qualifizierte a.sign premium Zertifikate mit Bürgerkartenfunktion auf der e-card aufbringen zu lassen. Auf der e-card werden zwei Zertifikate aufgebracht, ein Einfaches, und ein Qualifiziertes, welches zum Signieren von elektronischen Rechnungen verwendet werden kann. Für den Bürger entstehen dabei keine Kosten.

[43] Das `SigG` sagt in § `8 Abs 4` nur aus, dass ein Pseudonym nicht anstößig sein darf, und auch nicht offensichtlich zu Verwechslungen führen darf.

a.Sign business

Das Produkt a.sign business entspricht weitgehend dem Produkt a.sign premium. Bei a.sign business ist es aber auf jeden Fall möglich, den Firmenwortlaut in das Zertifikat aufzunehmen. Auch eine Aufnahme in das OrganizationName-Attribut ist hier möglich. Um die Vertretungsmacht des Signators gegenüber dem ZDA zu beweisen, kann das Vorweisen einer Vollmacht nötig sein. Das Zertifikat kann dann nicht nur vom Signator, sondern auch vom Machtgeber, also dem Unternehmen, widerrufen werden.

Genauso wie bei a.Sign premium muss auch hier ein spezieller Kartenleser verwendet werden, um mehrere Rechnungen mit einer PIN-Eingabe zu signieren. Tabelle 3.8 zeigt eine Auflistung der Kosten um mit a.sign business effizient qualifizierte Rechnungssignaturen zu erstellen. Der im Vergleich zum „a.sign premium“-Produkt höhere Preis für Registrierung und Zertifikatserstellung lässt sich durch das aufwendigere Prüfungsprozedere und die jeweils anvisierte Zielgruppe erklären.

Einmalige Kosten:[a]	
Registrierung und Zertifikatserstellung	€ 48,00
Kartenlesegerät „ReinerSCT cyberJack pinpad stapelsignatur“	€ 186,00
Gesamt:[b]	€ 234,00
Jährliche Kosten:[a]	
Zertifikat	€ 60,00
Gesamt:	€ 60,00

[a] Alle Beträge inkl. USt
[b] Sollen die Zertifikate nicht auf einer vorhandenen Karte aufgebracht werden, so müssen noch € 30,- inkl. USt pro Karte addiert werden.

Tabelle 3.8.: Kosten a.Sign business

a.Sign corporate

Dabei handelt es sich um ein fortgeschrittenes Serverzertifikat in verschiedenen Versionen. Die light-Version ist ein reines Softwarezertifikat und kann z. B. in einem Windows-Keystore installiert werden. Die medium- sowie die strong-Variante müssen auf einer SmartCard oder in einem HSM installiert werden. Alle drei Varianten können allerdings automatisiert angesprochen werden, und sind somit zur Serversignatur in Echtzeit geeignet.

Ein Firmenwortlaut kann in die „O“- und „OU“-Attribute des Zertifikats aufgenommen werden. Im CN-Attribut findet sich auch der Firmenwortlaut, oder der Domainname wieder.

Tabelle 3.9 zeigt eine Auflistung der Kosten für die A-Trust corporate Zertifikate.

Jährliche Kosten:[a]			
	light	medium	strong
Zertifikat	€ 240,00	€ 600,00	€ 2400,00
Gesamt	€ 240,00	€ 600,00	€ 2400,00

[a] Alle Beträge inkl. USt

Tabelle 3.9.: Kosten a.Sign corporate

in Planung: a.Sign business advanced

Dieses Zertifikatsprodukt ist derzeit noch nicht erhältlich, könnte aber laut Aussagen von Herrn Gepp von A-Trust angeboten werden, falls die Nachfrage dazu zukünftig bestehen würde. Dies ist wahrscheinlich auch von den gesetzlichen Entwicklungen im Bereich der elektronischen Rechnungslegung abhängig (siehe „2.1. Derzeitige Rechtslage“ auf S. 17).

Bei diesem Produkt handelt es sich um ein qualifiziertes Zertifikat, welches die Aufnahme des Firmenwortlautes als Pseudonym und im „O“- sowie im „OU“-Attribut zulassen würde. Die zu dem Zertifikat gehörenden Signaturerstellungsdaten würden sich dabei auf einer SmartCard befinden.

Das Besondere an diesem Angebot wäre aber, dass damit fortgeschrittene Signaturen auf Basis von qualifizierten Zertifikaten ermöglicht würden. Zur Erstellung von qualifizierten Signaturen verlangt der Gesetzgeber ja nicht nur, dass ein qualifiziertes Zertifikat und eine SSCD[44] verwendet werden. Es darf auch nur nach einem Willensakt des Signators, also meist nach einer PIN-Eingabe, signiert werden. Weiters muss der Signator unmittelbar vor der Signatur dazu im Stande sein, die zu signierenden Dokumente zu prüfen, und er muss darüber Bescheid wissen, wie viele Dokumente mit einer PIN-Eingabe signiert werden (siehe „2.1.2. Server- und Massensignatur“ auf S. 22). Soll aber nur fortgeschritten signiert werden, könnte man auf diese Zusatzforderungen verzichten, und trotzdem ein qualifiziertes Zertifikat verwenden. Zur Signatur mit einem solchen Zertifikat würde also keine sichere Signaturumgebung, d. h. keine spezielle Software, und

[44] Darunter wird im Allgemeinen nur die SmartCard verstanden, Kartenleser und Software gehören zur Signaturumgebung

kein spezieller Kartenleser vorgeschrieben werden. Es ist allerdings anzunehmen, dass der Verwendungszweck eines solchen Zertifikats insoweit eingeschränkt würde, als dass damit keine qualifizierten Signaturen erstellt werden könnten. Dafür könnten damit Dokumente automatisiert und in Echtzeit von einem Signaturserver mit fortgeschrittenen Signaturen und qualifizierten Zertifikaten versehen werden. Dies würde ungefähr dem in Deutschland praktizierten Verfahren entsprechen, wo man eine Signaturerstellungseinheit mit einer einzigen PIN-Eingabe für eine gewisse Zeit freischalten kann, in der dann automatisiert signiert wird. Eine weitere Möglichkeit wäre es, mit einer PIN-Eingabe die Signatur einer gewissen Anzahl von Dokumenten in Echtzeit zuzulassen.

A-Cert[45]

Laut eigenen Angaben auf der Webseite der A-CERT, werden die A-Cert Zertifizierungsdienste vom Verein „ARGE DATEN - Österreichische Gesellschaft für Datenschutz“ betrieben. Die A-Cert bietet unter anderem auch das fortgeschrittene Zertifikat „A-Cert advanced“ speziell für die Rechnungssignatur an.

Der zum Zertifikat passende private Schlüssel ist dabei nicht auf einer SmartCard gespeichert, sondern wird zusammen mit dem Zertifikat als Datei geliefert, und kann dann z. B. direkt auf einem Server installiert werden (siehe „4.2. Digitale Signatur“ auf S. 79). Dies ermöglicht eine sehr einfache Durchführung von serverbasierten Massensignaturen, wie dies in „Abschnitt 2.1.2 – Server- und Massensignatur“ beschrieben wird. Wenn es gewünscht wird, ist es aber auch möglich, Schlüssel und Zertifikat auf spezieller Hardware abzulegen.

Bei diesem Produkt kann der Unternehmenswortlaut in das Zertifikat aufgenommen werden. Tabelle 3.10 zeigt eine Auflistung der Kosten für ein A-Cert advanced Zertifikat.

Jährliche Kosten:[a]	
Zertifikat	€ 48,00
Gesamt[b]:	€ 48,00

[a] Alle Beträge inkl. USt
[b] Bei Vorauszahlung für 2 Jahre. Wird für 4 Jahre im Voraus bezahlt, verringert sich der Betrag auf € 42,00 pro Jahr.

Tabelle 3.10.: Kosten A-Cert advanced

Qualifizierte Zertifikate werden von A-Cert nicht angeboten.

[45] `http://www.a-cert.at`

3.6.4. Fazit

Obwohl für ein Kleinstunternehmen außerhalb der IT-Branche eine Eigenentwicklung im Normalfall wohl keine Option darstellt, ist dies der Weg welcher für den betrachteten Betrieb gewählt wurde. Die Gründe dafür sind wie folgt:

- Die Rechnungen oder Rechnungsdaten müssen nach dem Export aus dem Altsystem umfangreich konvertiert werden, um den in „Abschnitt 3.5 – Soll-Zustand“ skizzierten Anforderungen zu entsprechen. Da hier aller Voraussicht nach die Entwicklung einer spezifischen Konvertierungsapplikation nötig ist, macht es Sinn, in diese Applikation den gesamten Prozess zu integrieren.

- Die Angebote zum Outsourcen von e-Billing-Prozessen sind zwar durchaus interessant, und würden auch einige charmante Zusatzdienste bieten. Als Beispiel sei hier der Service der EBPP mit der Möglichkeit, die elektronischen Rechnungen in das Internet-Banking des Empfängers einzuspeisen genannt. Allerdings scheint sich die Preispolitik der Serviceanbieter an den Portokosten zu orientieren. Man verrechnet also dem Leistungsbringer Gebühren, die meist knapp unter den Portokosten liegen. Angesichts der weiteren Einsparungsmöglichkeiten und Vorteile der elektronischen Rechnung erscheint diese Preispolitik durchaus gerechtfertigt. Im betrachteten Betrieb fallen aber beim Rechnungsversand keine Portokosten an (siehe „3.4. Einsparungen“ auf S. 46). Trotz der möglichen anderen Vorteile, würde in diesem Fall ein Auslagern des Rechnungsstellungsprozesses die Kosten pro Rechnung mitunter verdoppeln, und kann daher nicht als Option angesehen werden.

  Um in Einzelfällen die Vorteile besonders innovativer Servicemodelle zu nutzen, könnten jedoch in eine selbst entwickelte Anwendung Schnittstellen zu solchen Diensten eingebaut werden.

- Da es sich bei dieser Arbeit um eine Diplomarbeit aus dem Fachgebiet der Informatik handelt, besteht das notwendige Wissen und natürlich auch das Interesse, sich mit dem Thema auch auf einer technischen Ebene auseinander zu setzen.

Bei der Zertifikatsauswahl fiel die Entscheidung auf ein „A-Cert advanced“-Zertifikat. Gründe dafür sind die relativ geringen Kosten, die Möglichkeit zur Aufnahme des Firmenwortlautes in das Zertifikat, und die einfache und unkomplizierte Handhabung, insbesondere bei der Erzeugung von Massensignaturen. Es soll aber nicht verschwiegen werden, dass die in „Abschnitt 2.1 – Derzeitige Rechtslage“ beschriebenen möglichen Änderungen der gesetzlichen Grundlagen es nötig machen könnten, diese Entscheidung zu überdenken.

4 Technische Sicht

Dieses Kapitel beginnt mit einer überblicksartigen Vorstellung der für den betrachteten Betrieb entwickelten Applikation. Danach werden ausgewählte technische Themenbereiche detailliert behandelt, bevor Konfiguration und Architektur der entwickelten Applikation umfassend erläutert werden.

4.1. Die InvoiceManager-Applikation im Überblick

Bei der InvoiceManager-Applikation handelt es sich um eine eigens für den betrachteten Betrieb entwickelte Desktopapplikation. Für die Implementierung fiel die Wahl auf die Programmiersprache Java, da dies einerseits die Sprache ist, mit welcher der Autor am Besten vertraut ist, und da Java andererseits eine sehr hohe Flexibiltät bietet. Damit ist nicht nur gemeint, dass Java eine plattformunabhängige Entwicklung erlaubt, sondern vor allem auch, dass mit Java sowohl Webandwendungen, Desktopanwendungen, als auch Anwendungen nach dem SOA-Prinzip entwickelt werden können. Wird beim Entwurf auf diese verschiedenen Architekturen Rücksicht genommen, so könnten große Teile der Anwendung bei einer Migration auf eine andere Architektur wiederverwendet werden.

Es wurde versucht die Anwendung so flexibel zu gestalten, damit diese an definierten Punkten einfach angepasst bzw. erweitert werden kann. Dies ist nötig, um den sich ändernden organisatorischen und juristischen Anforderungen gerecht zu werden, sowie um die Anwendung eventuell auch in anderen Betrieben einsetzen zu können.

Die folgende Aufzählung fasst kurz die Eigenschaften der InvoiceManager-Applikation zusammen, welche später in diesem Kapitel noch genauer beschrieben sind:

- 2-tier Java Desktop-Applikation mit Swing
- Datenhaltung in einer relationalen Datenbank (entkoppelt durch OR-Mapping)

- Anpassbar durch eine XML-Konfigurationsdatei
- Unterstützung der Rechnungsformate ebInterface und PDF
- Versand der Rechnungen per Email (auf andere Versandarten erweiterbar)
- Import der Rechnungen aus einer Textdatei des Altsystemes (erweiterbar auf andere Datenformate und Importverfahren)
- Fortgeschrittene Signatur der Rechnung (experimentielle Unterstützung für Signaturen mit qualifizierten Zertifikaten)
- Aufzeichnung des Verarbeitungsstatus für jede Rechnung

4.1.1. Workflow

Welche Phasen der Rechnungsverarbeitung von einer Applikation unterstützt werden müssen, um den in „Abschnitt 3.5 – Soll-Zustand“ beschriebenen Zustand zu erreichen, wurde bereits in „Abschnitt 3.6.1 – Phasen der Rechnungsbearbeitung“ gezeigt. Abbildung 4.1 zeigt den Workflow, welcher von der InvoiceManager-Applikation gefordert und unterstützt wird. Diese Workflow-Phasen sowie die beiden begleitenden Funktionalitäten Archivierung und Logging sind im Folgenden kurz beschrieben:

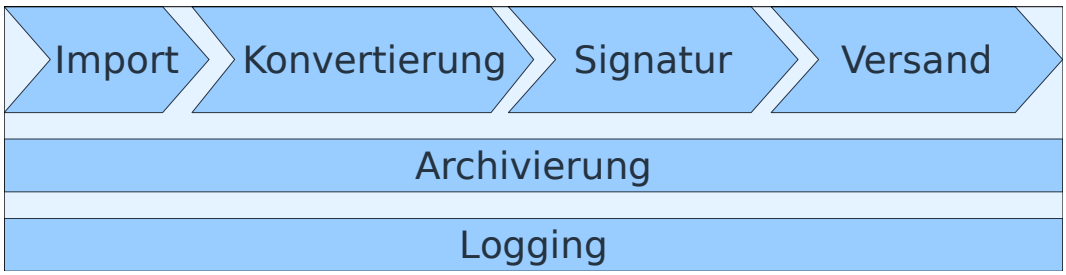

Abbildung 4.1.: Workflow der InvoiceManager-Applikation

Import Dies ist der erste Schritt in der Bearbeitung von Rechnungen mit dem InvoiceManager. Rechnungen aus dem Altsystem werden importiert, um daraus ebInterface-Rechnungen zu erzeugen.

Konvertierung Neben der ersten Konvertierung aus dem Format des Altsystems nach ebInterface werden die Rechnungen bei Bedarf auch in das PDF- oder HTML-Format konvertiert.

Signatur Rechnungen werden im gewünschten Format signiert. Wichtig dabei ist, dass wie in „Abschnitt 2.1.4 – Aufbewahrung der Rechnungen“ beschrieben, jede Rechnung nur in einem Format signiert werden kann. „Abschnitt 4.2 – Digitale Signatur“ beschreibt die digitale Signatur und deren verschiedene Qualitäten, während „Abschnitt 4.3 – Rechnungsformate“ erläutert, wie Dokumente in den beiden unterstützten Rechnungsformaten signiert werden können.

Versand Rechnungen können von der InvoiceManager-Applikation derzeit per Email an die Empfänger gesendet werden. Die Implementierung dieser und anderer Versandarten ist in „Abschnitt 4.4.2 – Zustellung der Rechnungen“ beschrieben.

Archivierung Rechnungen werden durch die InvoiceManager-Applikation mit anderen Daten in einer relationalen Datenbank abgelegt. Die Verbindung zwischen Datenbank und der Programmiersprache Java erfolgt durch OR-mapping.

Logging Das zentrale Artefakt der Anwendung ist ein Rechnungsobjekt. Jede Aktion auf ein solches Objekt wird aufgezeichnet und in der Datenbank gespeichert.

4.1.2. Entwicklungsgeschichte

Am Beginn dieser Diplomarbeit stand der Wunsch, den in „Abschnitt 3.2.2 – Rechnungsversand“ geschilderten Prozess der Rechnungsstellung zu vereinfachen. Dieser Prozess sollte weitgehend automatisiert werden und sowohl für den Rechnungssteller, vor allem aber auch für den Rechnungsempfänger, komfortabler gestaltet werden.

Zu Projektbeginn wurde in gemeinsamen Gesprächen mit dem Rechnungssteller festgestellt, dass diese Ziele am Besten durch eine Webplattform erreicht werden könnten. Diese sollte in eine Unternehmenswebseite integrierbar sein, und über die folgenden Eigenschaften verfügen:

- der Rechnungssteller kann Rechnungen aus dem Altsystem per Upload in die Webplattform einspeisen
- Rechnungen werden von der Webapplikation in die gewünschten Formate konvertiert
- Rechnungen werden durch die Webapplikation digital signiert
- Rechnungen werden den bei der Webplattform registrierten Empfängern zugeordnet

- Rechnungsempfänger können sich über eine gesicherte SSL/TLS-Verbindung mit einem Benutzernamen und einem Passwort bei der Plattform authentifizieren, und über die Plattform ihre Rechnungen beziehen.

In den Entwicklungsphasen Analyse und Entwurf wurden diese Use Cases und die geplante Anwendungsarchitektur aus folgenden Gründen noch einmal grundsätzlich in Frage gestellt:

- Als mit der Entwicklung dieser Applikation begonnen wurde, war der in „Abschnitt 2.3.1 – Vorschlag 1: Drei gleichrangige Modelle“ beschriebene Vorschlag zur Änderung der Anforderungen an die elektronische Rechnung in aller Munde. Wie aus diversen Experteninterviews hervorging, wurde von beinahe allen Seiten mit einer Umsetzung dieser Modelle gerechnet. Besonders zwei Eigenschaften dieses Vorschlages machten ein Überdenken des angedachten Vorgehens notwendig:

  - Von Seiten einiger Experten wurde vor allem dem Modell Finanzonline, welches einen Verzicht auf die digitale Signatur ermöglichen würde, ein großes Potential zugeschrieben. Bei diesem Modell, wie es in „Abschnitt 2.3.1 – Vorschlag 1: Drei gleichrangige Modelle“ beschrieben ist, müsste eine Applikation die Anbindung an das Finanzonline-System ermöglichen. Eine Bereitstellung über eine eigene Webplattform erscheint dann allerdings nicht mehr besonders sinnvoll.

  - Vom BMF wurde als Alternative zu Finanzonline auch das Modell Investitionsschutz (siehe „2.3.1. Vorschlag 1: Drei gleichrangige Modelle“ auf S. 30) vorgeschlagen. Dieses würde eine qualifizierte/sichere Rechnungssignatur nötig machen, welche in einer Webapplikation nur sehr umständlich (z. B. durch ein Java-Applet oder durch eine Implementierung der „Schnittstelle Security Layer“ [26]) erzeugt werden kann. Technischer Hintergrund ist hier dass solche Signaturen, wie in „Abschnitt 2.1.2 – Server- und Massensignatur“ beschrieben, nicht vollständig automatisiert in Echtzeit erstellt werden können. Außerdem ist zu bedenken, dass bei deren Erstellung Zugriff auf lokale Ressourcen wie ein Kartenlesegerät und eine SmartCard bestehen muss. Dies ist in einer Desktopapplikation deutlich einfacher zu implementieren als in einer Webapplikation.[1]

- Aus der Sicht des Rechnungsempfängers ist der angestrebte Rechnungsdownload eine Art „Pull“-Modell, bei welchem er Rechnungen aktiv abholen muss. Ein Rechnungsversand z. B. per Email, würde einem „Push“-Modell entsprechen. Obwohl die zu Beginn angedachte Variante die Kunden zu regelmäßigen Besuchen einer

[1] Anm.: bei der Schnittstelle Security Layer handelt es sich ja im Endeffekt auch um eine Desktopapplikation, die einen lokalen Server zur Verfügung stellt, welcher von der Webseite im Browser angesprochen wird

Unternehmenswebseite motivieren würde, ist der Ansatz, Rechnungen per Email zu versenden, für die Empfänger sicher komfortabler und besser automatisierbar.

Obwohl eine Implementierung als Webapplikation natürlich grundsätzlich möglich ist, hat vor allem der erste Punkt dazu geführt, dass das ursprüngliche Ziel zugunsten einer Desktopapplikation mit Namen „InvoiceManager“ aufgegeben wurde. Dies ist ein sehr gutes Beispiel dafür wie äußere Einflüsse, z. B. rechtliche Rahmenbedingungen, ein Softwareprojekt beeinflussen können. Das Abkommen vom Vorschlag der drei Modelle (siehe „2.3. Vorgeschlagene Änderungen“ auf S. 30) würde es hingegen wieder attraktiv erscheinen lassen, zumindest einen Teil der Applikation als Webplattform zu implementieren. Da beim Entwurf und bei der Implementierung darauf geachtet wurde, die Anwendung möglichst unabhängig von der Architektur zu gestalten, wäre eine Migration grundsätzlich auch möglich, ist derzeit aber nicht geplant.

4.2. Digitale Signatur

Dieser Abschnitt gibt zuerst eine kurze Einführung in die grundsätzliche Funktionsweise digitaler Signaturen. Danach werden einige fortgeschrittene Themen der digitalen Signatur und deren Nutzung in der Programmiersprache Java betrachtet. Die Ausführungen dieses Abschnittes folgen weitgehend den Quellen [41], [56], und [55].

4.2.1. Einführung in die digitale Signatur

Zur digitalen Signatur von elektronischen Daten – wie z. B. Rechnungen – benötigt man vor allem zwei technische Konzepte, nämlich „kryptographische Hashfunktionen“ sowie „asymmetrische Kryptographie“.

Kryptographische Hashfunktionen

Im Anhang der `SigV` wird eine kryptographische Hashfunktion definiert als [6]:

> *„Der Algorithmus „Hash-Funktion“ ist eine nicht umkehrbare Funktion, die eine umfangreiche Datenmenge (in der Regel einen Text) auf eine im Allgemeinen wesentlich kleinere Zielmenge fester Länge (Hash-Wert) abbildet.“*

Es handelt sich dabei also um eine mathematische Funktion $H(M)$, welche aus einer beliebig langen Nachricht M eine Art Fingerabdruck h mit fixer Länge n erzeugt [55]. Dieser Fingerabdruck wird dann Hashwert genannt. Die beliebig lange Nachricht würde im Kontext dieser Diplomarbeit einer Rechnung entsprechen.

Um für die Aufgabe als kryptographische Hashfunktion geeignet zu sein, muss eine solche mathematische Funktion zwei Eigenschaften aufweisen [55]:

Einwegeigenschaft (One-way) Der Fingerabdruck h einer elektronischen Rechnung muss sich leicht berechnen lassen. Andererseits darf es aber nicht hinreichend einfach sein, zu einem gegebenen Fingerabdruck, eine passende Rechnung zu konstruieren.

Kollisionsresistenz Es darf nicht einfach[2] möglich sein, zwei Nachrichten M und M' zu finden, für die gilt $H(M) = H(M')$, es sei denn $M = M'$. Das heißt, es darf nicht hinreichend einfach sein, zwei unterschiedliche Rechnungen zu finden, die den selben Hashwert besitzen. Eine wichtige Eigenschaft dieser Funktionen ist, dass sich möglichst der komplette Fingerabdruck der digitalen Rechnung ändert, falls man nur eine einzige Zahl, einen einzigen Buchstaben, oder aus technischer Sicht nur ein einziges Bit, der digitalen Rechnung verändert.

Die Sicherheit einer kryptographischen Hashfunktion hängt vom verwendeten Algorithmus und von der Länge n des erzeugten Fingerabdruckes ab. Sehr weit verbreitet ist zum Beispiel der *SHA-1*-Algorithmus, welcher 160 Bit lange Fingerabdrücke generiert. In den letzten Jahren wurde die Sicherheit dieses Algorithmus wiederholt in Frage gestellt. Sein Sicherheitswert wird nun nur mehr mit 69 Bit, anstatt der aufgrund der Länge der erzeugten Hashwerte geforderten 80 Bit, angegeben [41]. Oft wird empfohlen, anstatt *SHA-1* die neueren Varianten *SHA-224*, *SHA-256*, *SHA-384* oder *SHA-512* zu verwenden, welche 224 Bit, 256 Bit, 384 Bit bzw. 512 Bit lange Fingerabdrücke generieren. Dies ist aufgrund der hohen Verbreitung von *SHA-1* nicht immer ohne weiteres möglich. So sieht z. B. der in „Abschnitt 4.3.1.4 – Signatur von XML-Dokumenten“ vorgestellte Standard zur Signatur von XML-Dokumenten nur den *SHA-1*-Algorithmus als „required“ vor [69].

Zur Frage, welche Algorithmen zur digitalen Signatur verwendet werden können, ist laut § `3 Abs 2 SigV` der Anhang der `SigV` maßgeblich. In der derzeit gültigen Fassung[3] sieht dieser zur Erstellung von qualifizierten Signaturen die folgenden Hashfunktionen geeignet: *SHA-1*, *RIPEMD160*, *SHA-224*, *SHA-256*, *SHA-384*, *SHA-512*, und *WHIRLPOOL*.

Ein Dokument der Aufsichtsstelle über „Empfohlene Algorithmen und Parameter für elektronische Signaturen“ [54] empfiehlt allerdings, *SHA-1* ab 2008 nicht mehr zu verwenden. *SHA-224* bis *SHA-512* erscheinen laut dieser Argumentation bis Ende 2013 zur Erstellung von qualifizierten Signaturen geeignet zu sein.

[2] Die Komplexität dafür muss mindestens bei $2^{n/2}$ liegen, wobei n die Länge des Hashwertes ist. [55]
[3] Die derzeit gültige Fassung der `SigV` trat Anfang 2008 in Kraft.

Asymmetrische Kryptographie

Signaturalgorithmen sind meist sehr eng mit asymmetrischen Verschlüsselungsalgorithmen verwandt. Abbildung 4.2 zeigt allgemein die Vorgänge beim Ver- und Entschlüsseln. Die Klartextnachricht wird mit einem Verschlüsselungsschlüssel K_E auf eine spezielle Art verknüpft. Als Ergebnis erhält man das verschlüsselte Chiffrat, welches nur durch eine weitere Verknüpfung mit einem Schlüssel K_D wieder in die ursprüngliche Klartextnachricht zurückverwandelt werden kann.

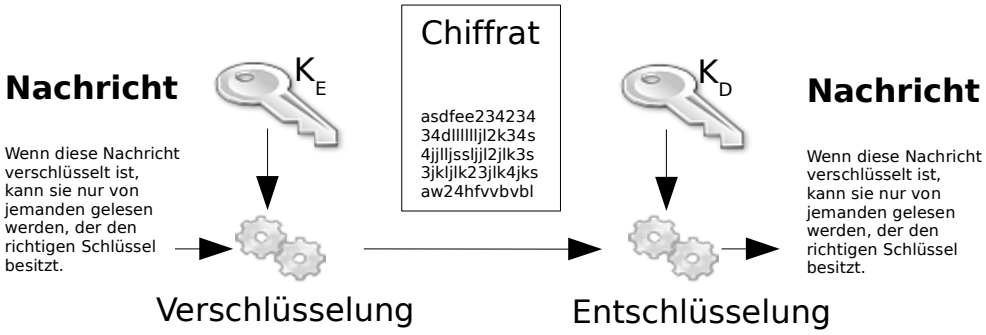

Abbildung 4.2.: Schematische Darstellung des Ver- und Entschlüsselns

Symmetrische Kryptographie gibt es bereits seit mehreren tausend Jahren. Diese Form der Verschlüsselung ist auch als „private key cryptography“ bekannt, da hierbei nur ein Schlüssel existiert, welcher zum Ver- und Entschlüsseln verwendet wird, und der dadurch sowohl dem Nachrichtensender, als auch dem Nachrichtenempfänger bekannt sein muss. Es gilt also: $K_E = K_D$. Symmetrische Verschlüsselung zeichnet sich durch eine hohe Geschwindigkeit beim Ver- und Entschlüsseln aus, leidet aber unter dem Problem der Schlüsselverteilung [56]. Möchten n Parteien untereinander jeweils sicher kommunizieren, sodass Nachrichten jeweils zwischen einem Versender und einem Empfänger vertraulich bleiben, so müssen $\frac{n\cdot(n-1)}{2}$ Schlüssel getauscht werden.

Die asymmetrische Kryptographie, welche dieses Schlüsselverteilungsproblem lösen kann, entstand erst in den 1970er Jahren. Diese Art der Kryptographie wird auch „public key cryptography“ genannt. Hier sind nun zwei Schlüssel im Spiel, von denen zwar einer, der private Schlüssel (private key), geheimgehalten werden muss, der andere aber problemlos öffentlich gemacht und an eine beliebige Anzahl von Kommunikationspartnern verteilt werden kann. Aus diesem Grunde wird dieser Schlüssel auch öffentlicher Schlüssel (public key) genannt. Bei der Verwendung dieser Art der Kryptographie, gilt also: $K_E \neq K_D$.

Wird die asymmetrische Kryptographie zum Verschlüsseln und nicht zur Signatur verwendet, so verteilt der Besitzer eines Schlüsselpaares, hier Bob genant[4], seinen öffentlichen Schlüssel an seine Kommunikationspartner. Möchte nun Alice eine Nachricht an Bob verschlüsseln, verwendet sie dazu den von Bob erhaltenen öffentlichen Schlüssel. Nun kann auch Alice selbst die Verschlüsselung nicht mehr rückgängig machen. Nur Bob kann die Nachricht mit dem zu seinem öffentlichen Schlüssel passenden privaten Schlüssel entschlüsseln.

Signatur mittels asymmetrischer Kryptographie

Bei der digitalen Signatur werden der private und der öffentliche Schlüssel genau umgekehrt wie bei der Verschlüsselung eingesetzt. Daten werden nun von Bob mit seinem privaten Schlüssel verschlüsselt. Diese Verschlüsselung kann dann von Alice oder jedem Anderen, der Bobs öffentlichen Schlüssel kennt, rückgängig gemacht werden. Was auf den ersten Blick nicht besonders nützlich erscheint, ist aber bei genauerer Betrachtung ein enorm mächtiges Werkzeug. Nur Bob kennt seinen privaten Schlüssel, und nur dieser private Schlüssel passt zu Bobs öffentlichem Schlüssel. Deshalb kann auch nur er die Daten so verschlüsseln, dass diese Verschlüsselung mit dem öffentlichen Schlüssel wieder rückgängig gemacht werden kann. Alice erhält die Klartextdaten, sowie das Chiffrat. Sie entschlüsselt das Chiffrat durch Einsatz des öffentlichen Schlüssels, und vergleicht die originalen Klartextdaten mit den entschlüsselten Daten. Sind beide identisch, so ist die Signatur gültig. Ist dies nicht der Fall, so wurden die Daten entweder nach der Signatur verändert, oder der zum Entschlüsseln verwendete öffentliche Schlüssel passt nicht.

Da asymmetrische Verfahren im Vergleich zu symmetrischen Verfahren sehr langsam sind, wird bei der digitalen Signatur nicht die gesamte ursprüngliche Nachricht signiert. Im Kontext dieser Arbeit verstehen wir unter einer zu signierenden Nachricht eine elektronische Rechnung. Der Signator signiert also nicht die Rechnung direkt, sondern einen Hashwert, welcher zuvor aus der Rechnung generiert wurde. Bei der Signaturprüfung wird dann ebenfalls ein Hashwert der Rechnung gebildet und mit dem entschlüsselten Wert verglichen. Abbildung 4.3 zeigt schematisch den Vorgang einer Signatur und einer darauffolgenden erfolgreichen Signaturprüfung. Der Teil der im Allgemeinen als Signatur verstanden wird, ist die Kette an Buchstaben, welche aus der Verschlüsselungsphase hervorgeht. In der Grafik nicht dargestellt ist, dass Bob und Alice auch einige administrative Informationen austauschen müssen, wie z. B. welche Hash- oder Signaturalgorithmen verwendet wurden. Wie „Abschnitt 4.3 – Rechnungsformate“ zeigt, enthalten die meisten Formate Möglichkeiten, um diese Informationen direkt im Rechnungsdoku-

[4] Die Teilnehmer an der asymmetrischen Kryptographie werden in der Fachliteratur traditionell mit den Namen Bob, Alice, und Eve bezeichnet.

ment zu hinterlegen. Auch die Signatur selbst kann meist eingebettet (embedded) im Rechnungsdokument transportiert werden.

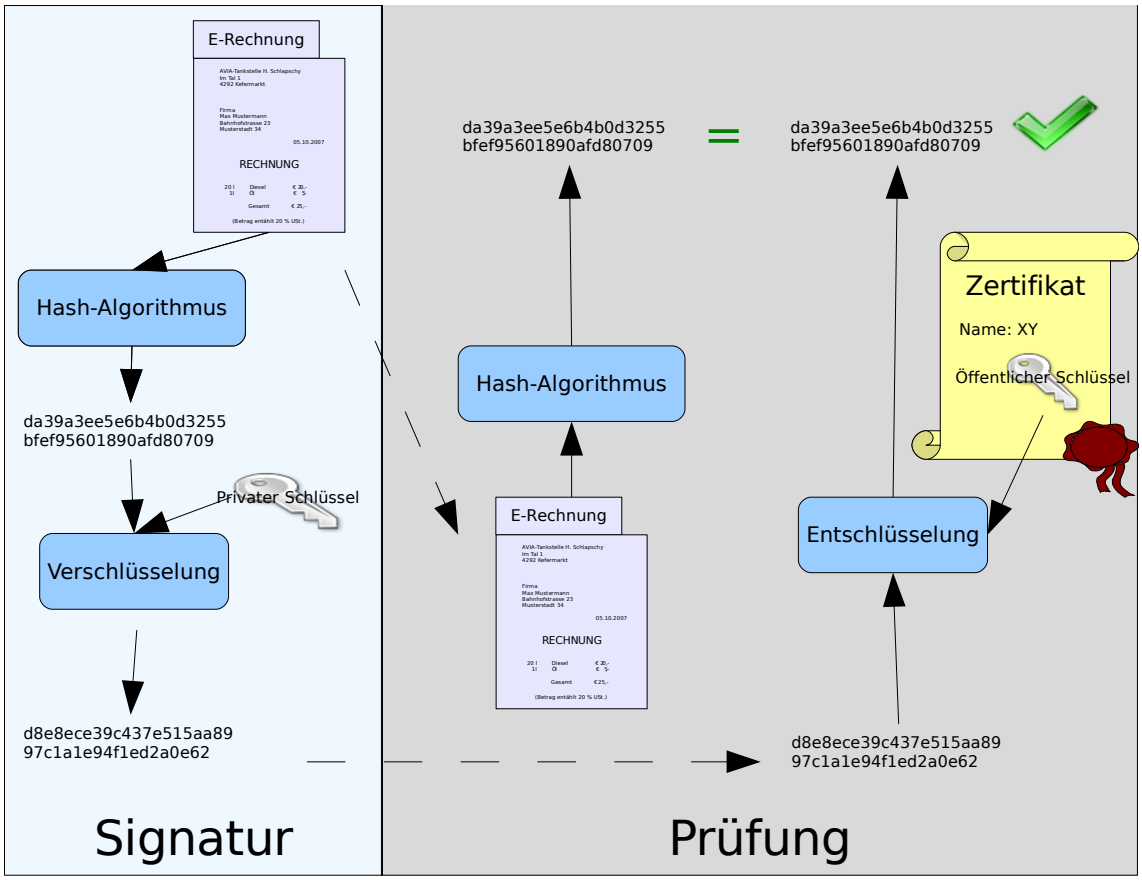

Abbildung 4.3.: Schematische Darstellung von Signaturerstellung und Signaturprüfung

4.2.2. Algorithmen

Zur Erstellung und Verifikation einer Signatur braucht man also einen privaten Schlüssel, einen öffentlichen Schlüssel, sowie eine Hashfunktion und einen Signaturalgorithmus. Zur Erzeugung der Schlüssel werden vom ZDA spezielle, auf der Erzeugung von Zufallszahlen basierende Schlüsselerzeugungsalgorithmen verwendet. Die dafür möglichen Verfahren sind im Anhang der `SigV` aufgelistet.

Asymmetrische Verfahren basieren meist auf Einwegfunktionen [56]. Eine Einwegfunktion ist eine Funktion, welche nicht umkehrbar ist. [56] nennt als Gegenbeispiel die Verdopplung, welche augenscheinlich umkehrbar ist. Nützlich für die asymmetrische Kryptographie sind allerdings Funktionen, bei denen es trotz Kenntnis der Funktion

und des Funktionswertes nicht, oder nur sehr schwer, möglich ist, den ursprünglichen Wert zu errechnen, es sei denn man kennt eine gewisse Information, welche dies erleichtert.

Die laut `SigV` verwendbaren Hashfunktionen wurden in diesem Abschnitt bereits genannt. Zur Signatur sind dort z. B. die folgenden, weit verbreiteten, Algorithmen aufgeführt:

RSA Dieser 1977[5] von den Forschern Ronald Rivest, Adi Shamir und Leonard Adleman (*RSA*) entdeckte Algorithmus beruht auf der Schwierigkeit, große ganze Zahlen zu faktorisieren. Je nachdem welcher Schlüssel zum Verschlüsseln und welcher Schlüssel zum Entschlüsseln verwendet wird, kann mit diesem Algorithmus entweder signiert oder verschlüsselt werden.

Laut Anhang der `SigV` eignet sich der *RSA*-Algorithmus mit einer Bitlänge von mindestens 1024 Bit (minimale Länge des Modulus) auch für qualifizierte Signaturen. Das bereits erwähnte Dokument der Aufsichtsstelle[6] [54] empfiehlt allerdings, *RSA* mit 1024 Bit langen Schlüsseln ab 2008 nicht mehr zu verwenden. Schlüssel der Länge von 2048 Bit eignen sich laut Aufsichtstelle bis Ende 2013 zur Erstellung qualifizierter Signaturen.

Viele in Österreich angebotene Zertifikatsprodukte (siehe „3.6.3. Auswahl eines Zertifikats“ auf S. 67), wie z. B. A-Cert advanced oder das Verschlüsselungszertifikat auf „a.sign premium“-Karten, verwenden RSA-Schlüssel mit verschiedenen[7] Schlüssellängen.

DSA Der „Digital Signature Algorithm“ (*DSA*) wurde 1991 vom *NIST* vorgeschlagen, und war im Rahmen des „Digital Signature Standard“ das erste von einer Regierung anerkannte Signaturverfahren [41]. Im Gegensatz zum *RSA*-Algorithmus kann *DSA* wirklich nur zur Signatur, und nicht auch zur Verschlüsselung verwendet werden.

Mathematisch beruht die Sicherheit des DSA-Algorithmus auf der Schwierigkeit, den diskreten Logarithmus in der multiplikativen Gruppe eines Primkörpers zu berechnen (siehe Anhang von [6]).

Im Anhang der `SigV` wird der *DSA*-Algorithmus für qualifizierte Signaturen als geeignet eingestuft, falls der öffentliche Schlüssel eine Mindestlänge von 1024 Bit

[5] [56] berichtet davon, dass nahezu derselbe Algorithmus bereits einige Jahre früher von Forschern am britischen „Government Communications Headquarter“ (GCHQ) entdeckt wurde, jedoch aus Geheimhaltungsgründen nicht veröffentlicht wurde.

[6] Die Funktion einer Aufsichtstelle wird in Österreich von der RTR GmbH ausgeführt.

[7] Das Verschlüsselungszertifikat auf der „a.sign premium“-Karte des Autors hat eine Schlüssellänge von 1536 Bit. Bei der Bestellung eines „A-Cert advanced“-Zertifikats kann zwischen den Schlüssellängen 1024 Bit, 1536 Bit, 2048 Bit, 3072 Bit sowie 4096 Bit gewählt werden.

aufweist und der private Schlüssel mindestens 160 Bit lang ist.[8] Die Aufsichtsstelle dagegen empfiehlt, *DSA*-Signaturen mit diesen Schlüssellängen ab 2008 nicht mehr zu verwenden. *DSA* mit Schlüsselängen von 2048 Bit für den öffentlichen Schlüssel und 224 Bit für den privaten Schlüssel wird bis 2013 als sicher angesehen.

ECDSA Anstatt mit dem traditionellen *DSA*-Algorithmus zu signieren kann dazu auch eine *DSA*-Version eingesetzt werden, welche auf *ECC* (Elliptic Curve Cryptography) basiert. Die Sicherheit dieser Variante liegt in der Schwierigkeit, den diskreten Logarithmus über elliptischen Kurven zu berechnen.

Mit der Schlüssellänge steigt bei der digitalen Signatur nicht nur die Sicherheit, sondern auch die Anforderungen an Speicher- und Rechenkapazität. Auf ECC basierende Signaturverfahren haben gegenüber den traditionellen asymmetrischen Verfahren den Vorteil, schon bei vergleichsweise geringen Schlüssellängen akzeptable Sicherheit zu bieten. Sie sind daher z. B. besonders gut für den Einsatz auf SmartCards geeignet. So sind laut Anhang der `SigG` für qualifizierte Signaturen schon Schlüssellängen von 160 Bit[9] ausreichend. [54] dagegen empfiehlt, bis Ende 2011 Schlüssellängen von mindestens 192 Bit und danach bis Ende 2013 mindestens 224 Bit zu verwenden.

Die qualifizierten Zertifikate auf „a.sign premium“-SmartCards verwenden 192 Bit *ECDSA*-Schlüssel.

Kryptographische Hashfunktion und Signaturalgorithmus bilden zusammen eine Einheit, welche nach dem Schema *<kryptographische Hashfunktion>With<Signaturalgorithmus>* identifiziert wird. Mögliche Kombinationen sind z. B. *SHA1withECDSA*, *SHA256withRSA*, *SHA1withDSA*, *RIPEMD160withRSA*, ...

4.2.3. OIDs

Objekte im Zusammenhang mit Algorithmen und Zertifiakten werden über sogenannte „Object Identifier“ (OID) identifiziert. Dies sind Nummern, welche einen globalen Namensraum bilden, in dem jedes Objekt einzigartig ist. Die Struktur eines OIDs kann man sich in Form eines Pfades oder eines Baumes vorstellen [41]. Alle Algorithmen oder Zertifikatserweiterungen werden über solche OIDs identifiziert. So hat z. B. die Kombination aus *SHA-1* und *ECDSA*, also *SHA1withECDSA* den OID *1.2.840.10045.4.1* und die in Abbildung 4.4 dargestellte Baumdarstellung. Eine andere Notationsform für OIDs ist die „ASN.1“-Notation, welche für *ECDSAWithSHA1* $iso(1)member-body(2)us(840)ansi-x962(10045)signatures(4)ecdsa-with-SHA1(1)$

[8] Diese Werte werden in den genannten Dokumenten immer für die Parameter p und q genannt, deren Länge aber mit dem öffentlichen bzw. dem privaten Schlüssel übereinstimmt.
[9] Entspricht $qMinLen = 160Bit$; außerdem gefordert: $r0Min = 10^4$, $MinClass = 200$

lautet. [2] erlaubt die Suche nach OIDs und gibt Informationen über einzelne OIDs aus.

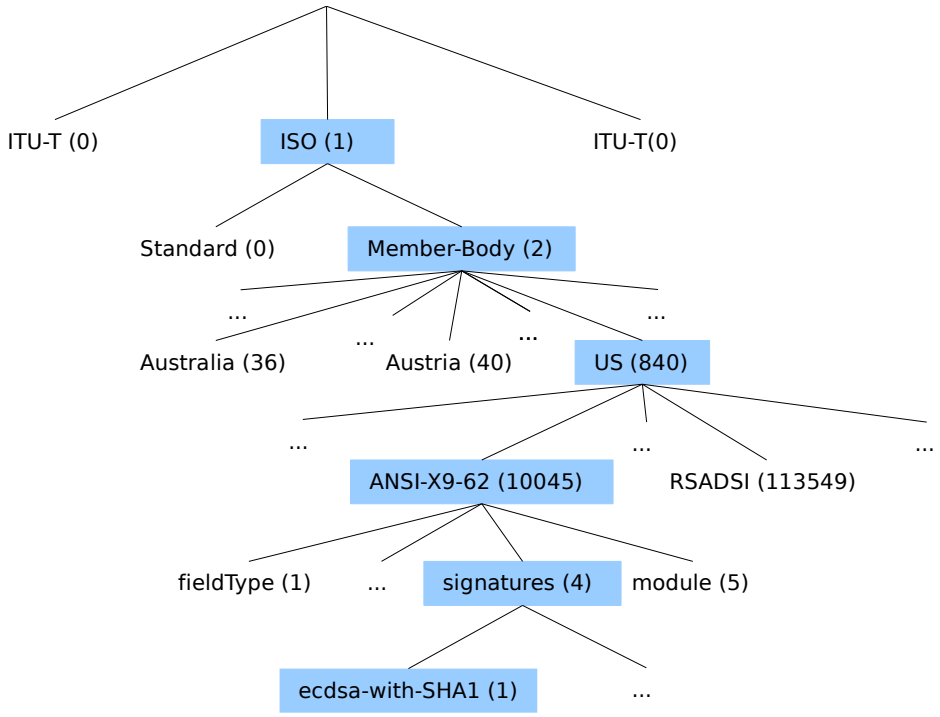

Abbildung 4.4.: OID-Baum für *ECDSAWithSHA1*

4.2.4. JCA/JCE

Kryptographische Operationen in Java werden am Besten mit Hilfe der „Java Cryptography Architecture / Java Cryptography Extension“ (JCA/JCE) ausgeführt. Die Trennung in zwei Teile ist historisch bedingt und spielt bei der Verwendung nur eine untergeordnete Rolle. Das JCA/JCE-API bietet Schnittstellen für kryptographische Operationen aller Art, implementiert die kryptographischen Funktionalitäten aber nicht selbst. Dies wird von Providern erledigt, die bei JCA/JCE registriert werden müssen. Diese Registrierung kann statisch in der Konfigurationsdatei `$JRE_HOME/lib/security/java.security` erfolgen, oder dynamisch im Quellcode einer Anwendung. Listing 4.1 zeigt die statische Konfiguration anhand eines Ausschnittes aus der Datei `java.security`, während Listing 4.2 einen Codeabschnitt zur dynamischen Registrierung eines Providers zeigt.

```
1 security.provider.1=sun.security.provider.Sun
```

```
2 security.provider.2=sun.security.rsa.SunRsaSign
3 security.provider.3=com.sun.net.ssl.internal.ssl.Provider
4 security.provider.4=com.sun.crypto.provider.SunJCE
5 security.provider.5=sun.security.jgss.SunProvider
6 security.provider.6=com.sun.security.sasl.Provider
7 security.provider.7=org.jcp.xml.dsig.internal.dom.XMLDSigRI
8 security.provider.8=sun.security.smartcardio.SunPCSC
9 security.provider.9=sun.security.mscapi.SunMSCAPI
10 security.provider.10=sun.security.pkcs11.SunPKCS11 c:\\pkcs11\\pkcs11.cfg
```

Listing 4.1: Statische Providerregistrierung in der Datei java.security

Bei der statischen Registrierung ist die Reihenfolge der registrierten Provider wichtig. Diese wird durch die Zahl vor dem „=“-Zeichen ausgedrückt. Es kann sein, dass eine kryptographische Funktionalität von mehr als einem Provider angeboten wird. Wird beim Aufruf einer Funktion dann nicht explizit ein Provider angegeben, so wird der erste passende Provider verwendet, wobei niedrigere Nummern Vorrang vor höheren Nummern haben [41]. Durch die explizite Angabe eines Providernamens bei oder vor der Ausführung einer kryptographischen Operation zwingt man die JCA/JCE zur Verwendung eines bestimmten Providers. Dies wird in [41] auch ausdrücklich empfohlen, da sich laut selbiger Quelle nicht alle Provider bei den selben Operationen gleich verhalten. Andererseits können dann natürlich Probleme auftreten, wenn der geforderte Provider auf dem ausführenden System nicht vorhanden ist. Diese Präzedenzregeln gelten auch für dynamisch registrierte Provider.

```
1  java.Security.Provider p = Security.getProvider("BC");
2  if(p == null){
3    p = new org.bouncycastle.jce.provider.BouncyCastleProvider();
4    Security.addProvider(p);
5  }
```

Listing 4.2: Dynamische Registrierung des Bouncy Castle Providers

Listing 4.2 zeigt die dynamische Installation des „Bouncy Castle“-Providers. Bouncy Castle[10] lässt sich einerseits als JCA/JCE-Provider verwenden, bietet aber andererseits auch eigene, darüber hinaus gehende APIs, z. B. zur Verarbeitung von Zertifikatserweiterungen oder für OCSP, an. Von diesem Provider werden auch ECC-Operationen unterstützt. Zeile 1 aus Listing 4.2 zeigt, dass Provider über ihren Namen angesprochen werden, der bei Bouncy Castle „BC“ lautet.

Objekte, auf denen kryptographische Operationen ausgeführt werden können, werden in JCA/JCE nicht selbst erzeugt, sondern nach dem „Factory-Method“-Pattern instantiiert. Jede spezielle Art von kryptographischer Operation hat eine Grundklasse. Spezielle Ausprägungen dieser kryptographischen Operation können über eine getInstance(

[10] http://www.bouncycastle.org/

String algorithm, String provider)[11]-Methode der Klasse erzeugt werden. Dadurch hält sich die Menge an benötigten Klassen in Grenzen, da nicht für jede Kombination aus Algorithmus, Art der Operation, und Parameter eine eigene Klasse gebraucht wird. So ist z. B. die Klasse `javax.crypto.Cipher` zuständig für alle Arten der symmetrischen und asymmetrischen Verschlüsselung, `java.security.MessageDigest` kümmert sich um das Erstellen von Hashwerten, und mit `java.security.Signature` können digitale Signaturen erstellt und verifiziert werden. Listing 4.3 zeigt die Erzeugung eines MessageDigest-Objektes durch eine Fabriksmethode. Der erste Parameter spezifiziert den benötigten Algorithmus. Es könnten im ersten Parameter auch noch weitere Informationen übergeben werden, was zum Beispiel bei symmetrischen Verschlüsselungsverfahren benötigt wird, um „Padding“ und „Mode“ zu übergeben. Der zweite Parameter spezifiziert, dass die Hashoperation vom Bouncy Castle-Provider ausgeführt werden soll. Wird der zweite Parameter nicht angegeben, so treten die bereits erwähnten Präzedenzregeln in Kraft. [41]

```
1 MessageDigest hash = MessageDigest.getInstance("SHA-1", "BC");
```

Listing 4.3: Instantiierung eines MessageDigest-Objektes

Weiters sei darauf hingewiesen, dass die JCA/JCE einer Java-Standardinstallation betreffend der verwendbaren Schlüssellängen beschnitten ist. Als Grund dafür nennt [41], dass starke Verschlüsselung nicht in allen Ländern gesetzlich erlaubt ist. Durch einen Austausch der betroffenen Policy-Dateien mit den „Unlimited Strength Jurisdiction Policy Files“ der Firma SUN kann hier Abhilfe geschaffen werden.

4.2.5. Zertifikate

Das österreichische `SigG` definiert ein Zertifikat als

> *„eine elektronische Bescheinigung, mit der Signaturprüfdaten einer bestimmten Person zugeordnet werden und deren Identität bestätigt wird.“*

Durch ein Zertifikat wird also bestätigt, dass ein öffentlicher Schlüssel sich im Besitz einer bestimmten Person befindet, welche auch über den dazu passenden privaten Schlüssel verfügt. Es handelt sich dabei vereinfacht gesagt um ein elektronisches Dokument, welches mindestens den Namen des Besitzers, sowie dessen öffentlichen Schlüssel beinhalten muss. Dieses Verhältnis zwischen Schlüssel und Besitzer wird vom ZDA dadurch zertifiziert, dass er dieses elektronische Dokument mit seinem privaten Schlüssel signiert. Auch der ZDA hat wiederum ein Zertifikat, welches bestätigt, dass er der Besitzer des zu diesem privaten Schlüssel passenden öffentlichen Schlüssels ist. Auf

[11] Diese überladene Methode gibt es noch in zwei weiteren, ähnlichen Ausführungen.

diese Weise kann ein Zertifikatspfad (Certificate Path) aufgebaut werden. Dieser wird oft auch als Zertifikatskette (Certificate Chain) bezeichnet. Die Wurzel dieses Pfades ist immer ein sogenanntes „Wurzelzertifikat“ („Root Certificate“) eines ZDA, welches „selbst signiert“ („self signed“) ist. Das heißt, der Schlüssel, welcher dem ZDA durch das Zertifikat zugeordnet wird, ist der Selbige, welcher zum Prüfen der Zertifikatssignatur verwendet werden muss. Zertifikate welche keine Wurzelzertifikate sind, aber zum Signieren anderer Zertifikate verwendet werden, werden als „Intermediate Zertifikate“ bezeichnet. Zertifikate, welche nicht zur Signatur anderer Zertifikate verwendet werden, nennt man „Endbenutzerzertifikate“ („End User Certificate“ oder „End Entity Certificate“). Abbildung 4.5 zeigt die schematische Darstellung eines solchen Zertifizierungspfades.

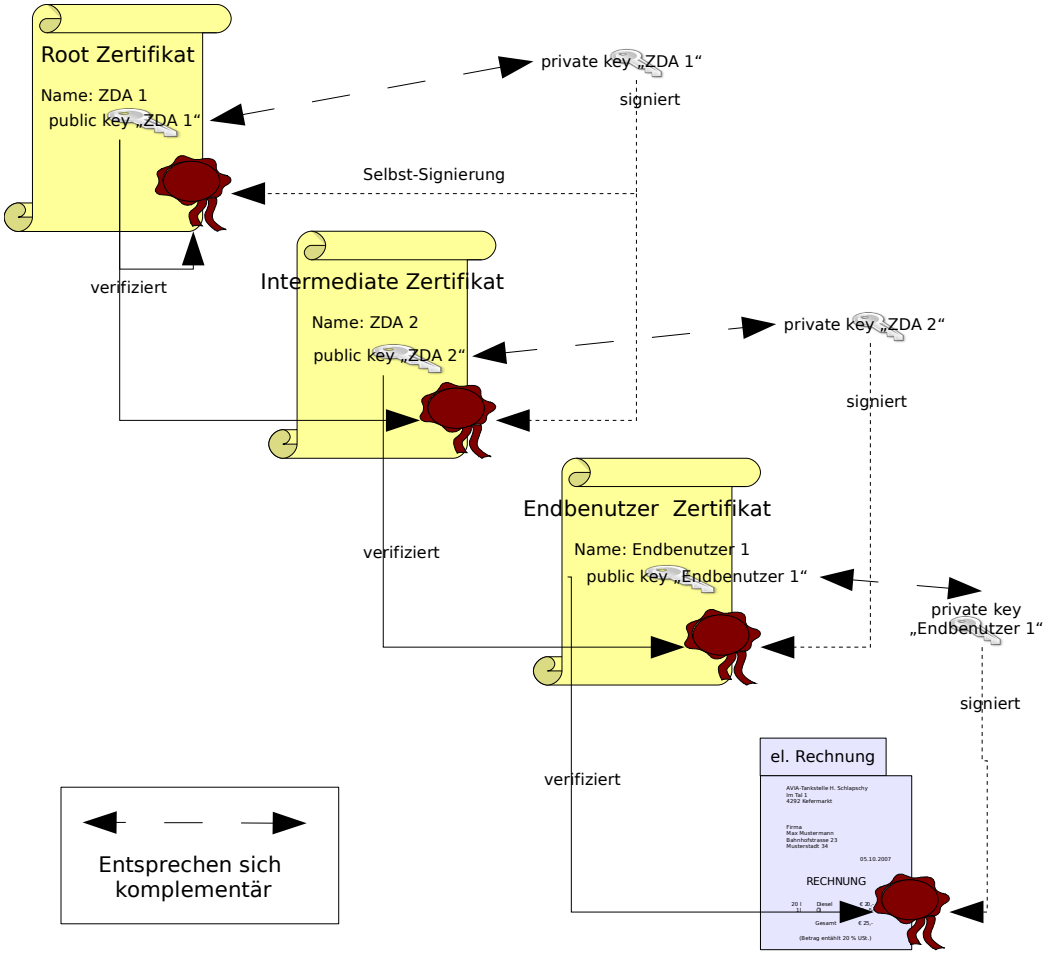

Abbildung 4.5.: Schematische Darstellung eines Zertifizierungspfades

An einigen Stellen in dieser Arbeit wurde vereinfacht erklärt, dass eine Rechnung mit dem Zertifikat des Signators signiert wird. Dies ist technisch natürlich nicht ganz richtig, da das Zertifikat streng genommen nur zum Transport des öffentlichen Schlüssels

und zur Bestätigung der Bindung dieses Schlüssels an eine bestimmte Person dient. Signiert wird dagegen mit dem zum im Zertifikat enthaltenen öffentlichen Schlüssel komplementären privaten Schlüssel.

Der wohl am weitesten verbreitete Standard für Zertifikate ist „X.509“. Die aktuellste Version dieses Standards ist „X.509v3“, und Zertifikate, welche diesem Standard folgen, nennt man X.509v3-Zertifikate. Der Aufbau und die Bestandteile eines X.509v3-Zertifikats[12] werden in der Beschreibungssprache „ASN.1“ definiert. Zertifikate selbst liegen meist in „DER“-kodierter Form vor, was einfach der Byterepräsentation des Zertifikats entspricht. Sollen Zertifikate z. B. per Email verschickt werden, oder soll das Zertifikat in das Textfeld eines Webformulares eingefügt werden, ist eine andere Repräsentation nötig. Hier wird oft das „PEM“-Format verwendet, was einer „Base64“-kodierten Version der DER-Kodierung entspricht, welche durch ein `---BEGIN CERTIFICATE---` und ein `---END CERTIFICATE---` eingeschlossen wird. [41]

X.509-Zertifikate enhalten neben dem öffentlichen Schlüssel und dem Namen seines Besitzers noch viele weitere Informationen, wie den Zeitraum der Gültigkeit, den Namen des Zertifikatsausstellers, oder eine Seriennummer. Seit Version 3 des X.509-Standards sind auch Zertifikatserweiterungen (Extensions) vorgesehen, wodurch die Aufnahme von beinahe beliebigen zusätzlichen Informationen in ein Zertifikat ermöglicht wird. Zertifikatserweiterungen sind optional und das Vorhandensein einer bestimmten Erweiterung wird immer von der Art und dem Verwendungszweck des Zertifikats abhängen. Einige Informationen, welche über Erweiterungen in Zertifikate aufgenommen werden, sind z. B. der Verwendungszweck, alternative Namen[13] für Aussteller und Besitzer, oder ein Hinweis auf die Qualität[14] des Zertifikats. Erweiterungen können als „critical“ markiert sein, was bedeutet, dass sie von einer das Zertifikat prüfenden Anwendung inspiziert werden müssen. Kommt in einem Zertifikat eine auf diese Art markierte Erweiterung vor, welche von der Anwendung nicht verstanden wird, so muss diese die Bearbeitung abbrechen. Zertifikatserweiterungen werden über OIDs identifiziert (siehe „4.2.3. OIDs“ auf S. 85). Eine ausführliche und verständliche Erläuterung der X.5.09-Standarderweiterungen bietet [29].

X.509v3-Zertifikate werden in Java durch Objekte der Klasse `java.security.cert.X509Certificate` abgebildet. Solche Zertifikatsobjekte können z. B. wie in Listing 4.4 gezeigt durch die Klasse `java.security.cert.CertificateFactory` aus einem im Dateisystem abgelegten Zertifikat erzeugt werden. Eine Möglichkeit um Zertifikate aus sogenannten „Key Stores“ zu laden wird in „Abschnitt 4.2.6 – „Key Stores““ gezeigt.

[12] [30] bietet eine gute Einführung zu X.509-Zertifikaten

[13] So kann beispielsweise eine Emailadresse des Besitzers in das Zertifikat aufgenommen werden. Dies ist zwar grundsätzlich auch im DN-Feld möglich, sollte aber in der „Subject Alternative Name“-Erweiterung erfolgen.

[14] Wie dies ja für qualifizierte Signaturen auch gefordert wird

```
1 InputStream in = new FileInputStream("C:\\testCert.cer");
2 CertificateFactory certFac = CertificateFactory.getInstance("X.509");
3 X509Certificate cert = (X509Certificate) certFac.generateCertificate(in);
4 in.close();
```

Listing 4.4: Instantiierung eines X.509Certificate-Objektes

Auf die Standardfelder von X.509-Zertifikaten, wie z. B. die Gültigkeitsdauer oder den Ausstellernamen, kann bequem mit den Methoden der `X509Certificate`-Klasse zugegriffen werden. Die Klasse bietet ebenfalls Methoden zum Prüfen von Signatur und Gültigkeitsdauer. Weiters bietet die Klasse Methoden zum Zugriff auf weit verbreitete und eventuell vorhandene Zertifikatserweiterungen, wie z. B. getBasicConstraints(), getKeyUsage() oder getExtendedKeyUsage(), deren Rückgabewerte einfach dargestellt und interpretiert werden können. Weniger verbreitete oder komplexere Zertifikatserweiterungen müssen mit der Methode getExtensionValue(String oid) gelesen werden, und liefern den DER-kodierten Erweiterungswert als byte-Array. Wie aus der Methodensignatur zu erkennen ist, nimmt die Methode den OID, welcher die jeweilige Erweiterung identifiziert, als Parameter entgegen. Der zurückgegebene DER-kodierte „Octet String“ muss dann weiter bearbeitet werden, um eine lesbare ASN.1-Notation zu erhalten. Die JCA/JCE-Standardklassen bieten hier kaum Hilfe, aber die Bouncy Castle-Bibliotheken (siehe „4.2.4. JCA/JCE“ auf S. 86) bieten im Paket `org.bouncycastle.asn1` nicht nur Hilfsklassen zum Lesen von ASN.1-Streams und ASN.1-Objekten, sondern definieren im Paket `org.bouncycastle.asn1.x509` auch Klassen zur Repräsentation der gängigsten Zertifikatserweiterungen. Listing 4.5 zeigt den Code zur Bearbeitung einer „Authority Information Access“-Erweiterung mit dem Ziel, die Informationen für einen Benutzer darzustellen. Diese Art der Erweiterung enthält weiterführende Informationen zum ausstellenden ZDA, wie zum Beispiel einen Link zu dessen Zertifizierungsrichtlinie oder die Adresse eines „OCSP-Responders“. Die Zeilen 2 bis 6 dieses Codeausschnittes sind bei fast allen auf diese Art zu lesenden Erweiterungen ähnlich zu verwenden.

```
1 private String getAuthorityInfoAccess(X509Certificate cert) throws
    IOException{
2 byte[] byteValue = cert.getExtensionValue(X509Extensions.
    AuthorityInfoAccess.getId());
3 ASN1InputStream a1n = new ASN1InputStream(byteValue);
4 DEROctetString oct = (DEROctetString) a1n.readObject();
5 a1n = new ASN1InputStream(oct.getOctets());
6 ASN1Sequence seq = (ASN1Sequence) a1n.readObject();
7 AuthorityInformationAccess authInfo = new AuthorityInformationAccess(seq);
8 AccessDescription[] accessDescriptions = authInfo.getAccessDescriptions();
9 ...
```

Listing 4.5: Verarbeitung einer „Authority Information Access“-Zertifikatserweiterung

Die InvoiceManager-Anwendung erlaubt die Zertifikatsanzeige bei der Signaturkonfiguration. Dabei werden die Daten aus dem Zertifikat gelesen und für den Benutzer in einem eigenen Dialogfenster aufbereitet. Dieses orientiert sich an der Anzeige von Zertifikaten in Windows und ist in Abbildung 4.6 zu sehen. Viele der durch diese Zertifikatsanzeige visualisierten Erweiterungen werden ähnlich zu dem in Listing 4.5 gezeigten Code verarbeitet.

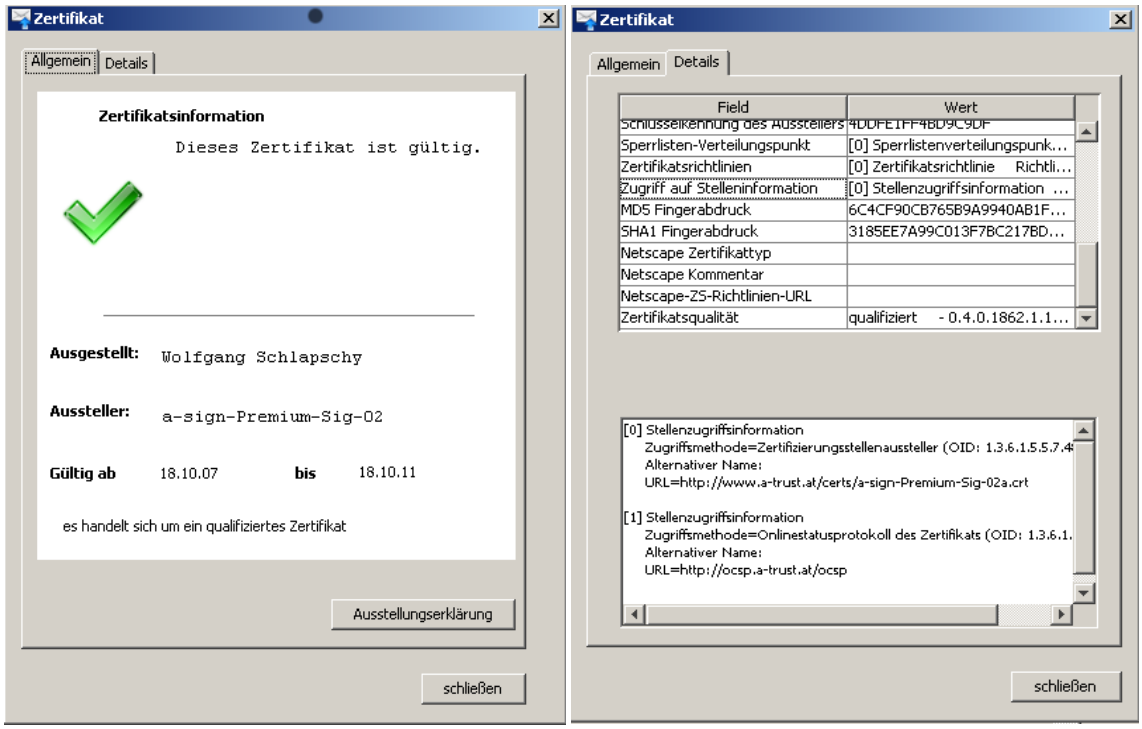

Abbildung 4.6.: Zertifikatsanzeige in der InvoiceManager-Applikation

Eine besondere Eigenschaft von Zertifikaten ist, dass diese vor Ende der Gültigkeitsdauer widerrufen werden können. Dies kann z. B. nötig sein, wenn der Signator verstirbt oder die Signaturerstellungsdaten (der private Schlüssel) korrumpiert wurden. Der Widerruf eines qualifizierten Zertifikats muss vom ZDA vermerkt werden, und die Widerrufsinformation muss einem Zertifikatsprüfer zur Verfügung gestellt werden. Dazu gibt es in der Regel zwei Ansätze:

CRL „X.509 Certificate Revocation Lists“ sind durch ASN.1 definierte Listen, in denen widerrufene Zertifikate aufscheinen. Diese Listen sind vom Aussteller der CRL digital signiert. In einer CRL findet sich für jedes widerrufene Zertifikat eine Seriennummer sowie das Widerrufsdatum. Findet sich die Seriennummer eines zu prüfenden Zertifikates in einer Widerrufsliste, so darf dieses nicht akzeptiert werden. Bei [41] wird deshalb auch der Vergleich mit einer „Blacklist“ gezogen, wie sie

früher an Händler zum Aufspüren von ungültigen Kreditkarten ausgegeben wurden. Der Bezugspunkt für Widerrufslisten für ein bestimmtes Zertifikat kann aus dem Zertifikat selbst abgelesen werden. Dafür gibt es die Zertifikatserweiterung `id-ce-cRLDistributionPoints (OID: 2.5.29.31)`.

CRLs besitzen eine gewisse Gültigkeitsdauer. Dies ist auch einer der größten Nachteile, da widerrufene Zertifikate eventuell erst verspätet als solche identifiziert werden können. Ein weiterer Nachteil ist, dass diese Methode bei einer großen Anzahl an Zertifikaten und vielen Widerrufen schwierig zu verwalten wird.

JCA/JCE unterstützt Widerrufslisten durch die Klasse `java.security.cert.X509CRL` und über das CertPath-API, worauf später noch kurz eingegangen wird.

OCSP Das „Online Certificate Status Protocol“ (OCSP) arbeitet etwas interaktiver. Der ZDA betreibt dazu einen sogenannten OCSP-Responder, welcher Anfragen betreffend der Gültigkeit von Zertifikaten beantwortet. Dies ist sicher leichter zu managen, als CRLs zu erstellen, bringt aber bei der Verifikation nur einen Vorteil, wenn der Dienst auch auf aktuelle Daten zugreift. Genauso wie CRLs vom ZDA signiert werden, sind auch OCSP-Responses signiert. OCSP-Requests können grunsätzlich auch signiert werden, was aber selten praktiziert wird [41]. OCSP definiert selbst kein Transportprotokoll sondern verwendet dafür gängige Protokolle wie z. B. http. Die Information, wohin eine Anwendung einen OCSP-Request stellen muss, erhält sie typischerweise aus dem Zertifikat. Dort kann in der „Authority Information Access“-Erweiterung ein Eintrag `id-pkix-ocsp (OID: 1.3.6.1.5.5.7.48.1)` definiert werden.

`JCA/JCE` selbst bietet keine Unterstützung für OCSP, aber die „Bouncy Castle“-Bibliotheken (siehe „4.2.4. JCA/JCE“ auf S. 86) erlauben die Verwendung von OCSP durch die Klassen im Paket `org.bouncycastle.ocsp`.

Zur Prüfung von Zertifikatsketten bietet JCA/JCE das CertPath-API. Die wichtigsten Klassen sind hier `java.security.cert.CertPathValidator`, und `java.security.cert.PKIXParameters`. Neben dem X.509-Standard spielen hier vor allem auch die sogenannten „PKIX“-Standards eine Rolle, welche ein „Internetprofil“ definieren, durch welches z. B. Erweiterungen, aber auch Prüfungsvorgänge definiert sind. OCSP wird vom CertPath-API nicht out-of-the-box unterstützt, kann aber durch Ableitung von der Klasse `java.security.cert.PKIXCertPathChecker`, und Implementierung der nötigen Logik erreicht werden (siehe [41] auf S. 273 für ein Beispiel).

Bei der Zertifikatsprüfung muss also die gesamte Zertifikatskette bis zum „Root-Zertifikat“ geprüft werden. Der Umstand, dass Zertifikate auslaufen oder widerrufen

werden können, macht bei der Prüfung die Einigung auf ein Gültigkeitsmodell notwendig. [77] nennt dazu die folgenden Modelle:

Schalenmodell Dieses Modell sagt aus, dass ein Zertifizierungspfad dann zu einem gewissen Zeitpunkt gültig ist, wenn zu diesem Zeitpunkt alle darin enthaltenen Zertifikate gültig sind. [77] benennt diesen Zeitpunkt im ursprünglichen *„Schalenmodell nach PEM*[15]*“* als den Prüfungszeitpunkt, stellt aber auch ein *„modifiziertes Schalenmodell“* vor, indem damit der Signaturzeitpunkt gemeint ist. In dieser Arbeit wird davon ausgegangen, dass im Schalenmodell die Gültigkeit eines Zertifikats zu jedem beliebigen Zeitpunkt erfolgen kann, sofern dieser nach der Signatur liegt. In Abbildung 4.7 führt eine Prüfung für den Signaturzeitpunkt zu einer gültigen Zertifikatskette, obwohl das Intermediate-Zertifikat zum Prüfungszeitpunkt bereits widerrufen ist. Würde eine Prüfung für den Prüfungszeitpunkt durchgeführt werden, wäre die Zertifikatskette nicht gültig.

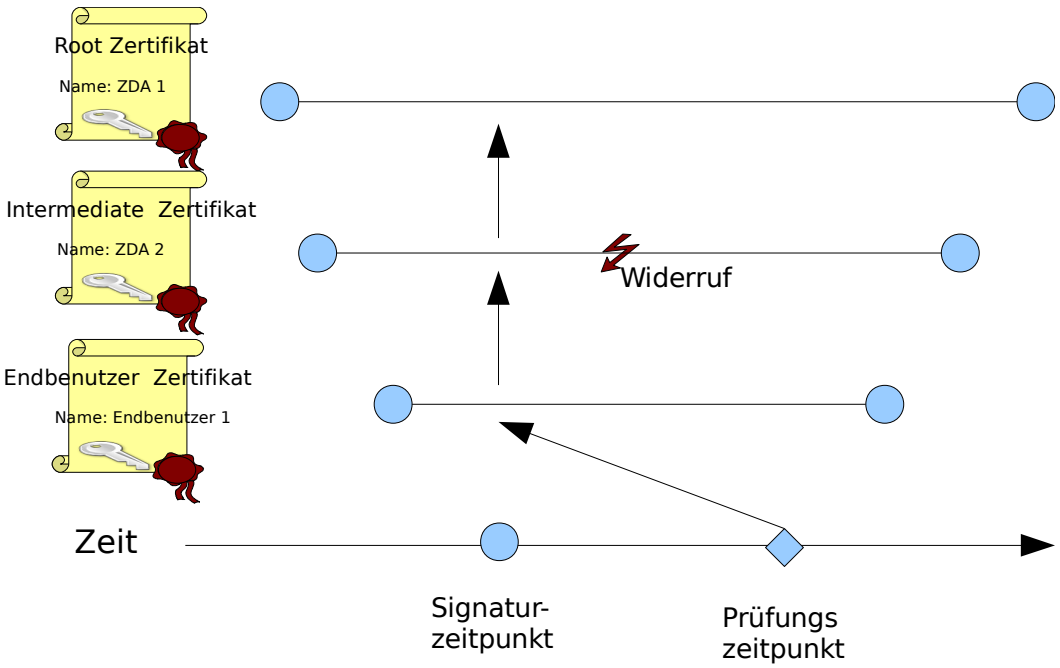

Abbildung 4.7.: Schalenmodell (nach [77])

Kettenmodell Das Kettenmodell definiert laut [77] schwächere Kriterien zur Zertifikatsprüfung: *„Es wird nur gefordert, dass jedes Zertifikat im Zeitpunkt seiner Anwendung gültig war“* [77]. Abbildung 4.8 zeigt, dass Zertifikatswiderrufe und abgelaufene Zertifikate in diesem Modell keine Rolle spielen.

[15] Damit ist der PEM-Standard gemeint.

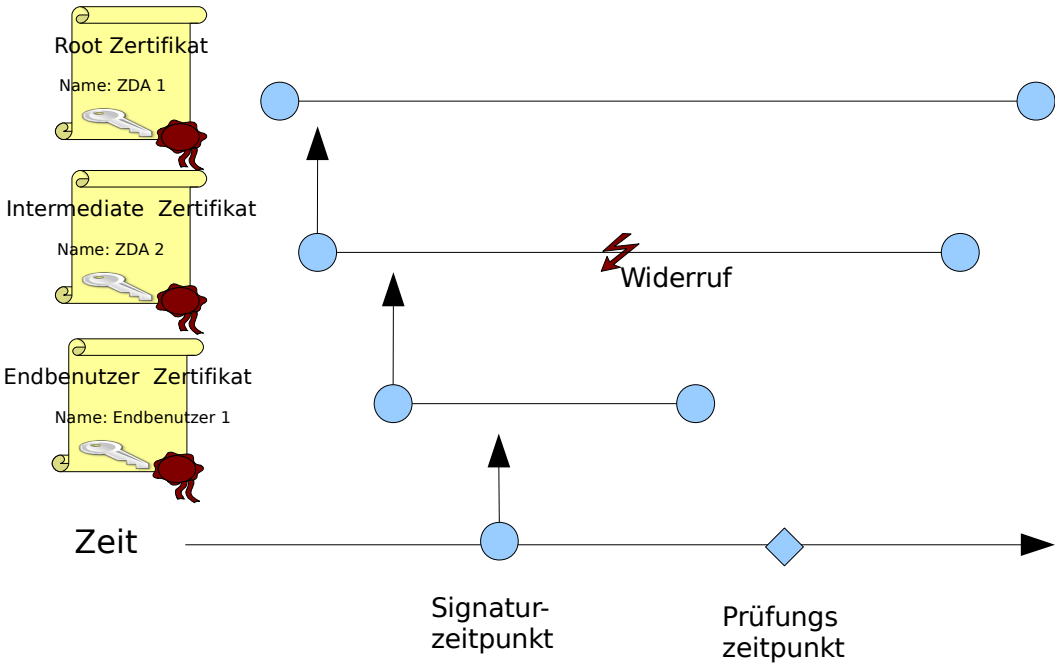

Abbildung 4.8.: Kettenmodell (nach [77])

Das PKIX-Profil verwendet das Schalenmodell, wodurch man dieses auch beim CertPath-API der JCA/JCE wiederfindet. Es ist möglich, einem Objekt der Klasse `java.security.cert.PKIXParameters` mit der Methode setDate(java. util .Date date) den Zeitpunkt mitzuteilen, für den die Gültigkeit des Zertifikatspfades geprüft werden soll. Für elektronische Rechnungen empfiehlt es sich hier, das Datum der Signatur zu verwenden. Ist dieses nicht bekannt, wäre es auch denkbar, das Rechnungsdatum zu verwenden.

Die Prüfung einer Rechnungssignatur beinhaltet also mehr als den in Abbildung 4.3 dargestellten Vorgang. Die folgenden Punkte sollten bei der Verifikation einer Rechnungssignatur geprüft werden:

- Prüfung der Signatur auf der Rechnung (siehe Abbildung 4.3)
- Prüfung der Zertifikatskette auf Gültigkeit per Schalenmodell mit dem Signatur- oder Rechnungsdatum als Zeitpunkt
  - Prüfung aller Zertifikatssignaturen
  - Prüfung ob alle Zertifikate zum fraglichen Zeitpunkt gültig waren (Waren alle Zertifikate zu dem Zeitpunkt schon gültig? War kein Zertifikat zum fraglichen Zeitpunkt bereits abgelaufen?)

- Prüfung ob Zertifikate aus der Kette widerrufen wurden (per OCSP oder CRL)
- Prüfung ob dem Root-Zertifikat vertraut werden soll
- Verarbeitung aller Zertifikatserweiterungen, die als „critical“ markiert sind (z. B.: Sind die Zertifikate überhaupt für die verwendeten Einsatzgebiete verwendbar?)

Zum Aufbau eines Zertifizierungspfades bietet JCA/JCE ein CertPathBuilder-API rund um die Klasse `java.security.cert.CertPathBuilder`. Woher die Zertifikate im Pfad kommen, hängt von den verwendeten Technologien und Formaten ab. Auf „XMLDsig“ (siehe „4.3.1.4. Signatur von XML-Dokumenten“ auf S. 113) basierende XML-Signaturen können Base64-kodierte Zertifikate im XML-Dokument mitliefern. In MS Windows-Betriebssystemen werden Root-Zertifikate in einem sogenannten „Windows-ROOT-Keystore“ vorgehalten (siehe „4.2.6. „Key Stores“ “ auf S. 96). Java-Anwendungen können ihre „Trust Anchors“, also jene Root-Zertifikate denen man vertrauen möchte, in einem globalen JKS namens `cacerts` ablegen. Die dazugehörige JKS-Datei liegt normalerweise im Verzeichnis `$JRE_HOME/lib/security`.

4.2.6. „Key Stores“

Im letzten Abschnitt über Zertifikate ist schon kurz die Frage aufgekommen, wo Zertifikate und Schlüssel gespeichert werden können. Natürlich ist es grundsätzlich möglich, einzelne Zertifikate oder auch ganze Zertifikatspfade einfach in einer Datei zu speichern. Allerdings gibt es speziell zu diesem Zweck sogenannte „Keystores“. Diese erlauben einerseits die Speicherung vertrauenswürdiger Zertifikate, andererseits die Speicherung privater Schlüssel und der dazugehörigen Zertifikate. Elemente werden in Keystores unter einem sogenannten „Alias“ abgelegt, unter welchem sie angesprochen werden können. Im Folgenden werden drei Möglichkeiten gezeigt, private Schlüssel sicher aufzubewahren und mittels JCA/JCE darauf zuzugreifen. Die verschiedenen Arten der Schlüsselaufbewahrung unterscheiden sich in Hinblick auf Sicherheitsanforderungen und Zugriffskomplexität.

In einer Java-Anwendung befinden sich Schlüssel und Zertifikate immer in einem Objekt der Klasse `java.security.KeyStore`. Dem allgemeinen Ansatz der JCA/JCE folgend, können sich dahinter aber ganz verschiedene Schlüsselspeicher befinden. Ein Keystore muss zuerst instantiiert und dann vor der Verwendung geladen werden (siehe Listing 4.6). Einige Keystores verlangen schon beim Laden ein Passwort. Andere Keystores sichern die privaten Schlüssel, welche dann nur gegen Angabe eines Passwortes aus

dem Keystore geladen werden können, oder nur gegen Angabe eines Passwortes zum Anfertigen einer Signatur verwendet werden können.

4.2.6.1. Windows-Keystores

Microsoft Windows Betiebssysteme speichern Zertifikate und private Schlüssel in MS Windows Keystores. Native Windowsprogramme verwenden das „Windows CryptoAPI“ (CAPI) zum Zugriff auf diese Keystores [65]. Seit Java SE 6 ist in der JCA/JCE standardmäßig der SunMSCAPI-Provider registriert. Dieser setzt auf CAPI auf und bietet von Java aus Zugriff auf native kryptographische Operationen des Windows-Betriebssystems. Auch der Zugriff auf die im folgenden erklärten Schlüsselspeicher ist damit möglich.

Windows-ROOT Dieser Keystore speichert alle Root-Zertifikate, die als Trust Anchor dienen können. Dieser Schlüsselspeicher ist bei der Signaturverifikation wichtig.

Windows-MY Dieser Keystore speichert die privaten Schlüssel der Benutzer und die dazugehörigen Zertifikate.

Listing 4.6 zeigt den Zugriff auf einen in diesem Keystore gespeicherten privaten Schlüssel und die dazu gespeicherten Zertifikate. Zeile 3 zeigt wie mittels einer Fabriksmethode ein Keystore vom passenden Typ erzeugt wird. Dieser wird in Zeile 4 durch den Aufruf der Methode KeyStore.load(InputStream stream, **char**[] password) geladen. Der erste Parameter muss `null` sein, da der Keystore nicht von einer Datei im eigentlichen Sinne kommt. Der zweite Parameter muss `null` sein, da MS Windows Keystores keine Passwörter verwenden [65]. Nach dem Laden des Keystores wird in den Zeilen 5 bis 7 auf den privaten Schlüssel und die dazugehörigen Zertifikate zugegriffen. Das Objekt `privKey` ist dabei nur ein Wrapper für einen nativen „key handle“ [65]. Soll mit diesem privaten Schlüssel später eine Rechnung im PDF- oder XML-Format (siehe „4.3. Rechnungsformate“ auf S. 103) signiert werden, so muss dabei der SunMSCAPI-Provider explizit angegeben werden.

```
1 // PRE-Condition: SunMSCAPI-Provider ist registriert
2 String alias = "Name des Certificate-Alias"
3 java.security.KeyStore ks = java.security.KeyStore.getInstance("
    Windows-MY");
4 ks.load(null, null);
5 java.security.Key privKey =  ks.getKey(alias, null);
6 java.security.cert.Certificate[]  certChain = ks.getCertificateChain(
    alias);
```

```
7 java.security.cert.Certificate signingCert = ks.getCertificate(alias)
    ;
```

Listing 4.6: Zugriff auf den Windows-MY Keystore

Die mit dem SunMSCAPI-Provider erzeugbaren Signaturen sind derzeit auf *SHA1withRSA*, *MD5withRSA*, und *MD2withRSA* eingeschränkt [65].

Zertifikate und private Schlüssel, welche für fortgeschrittene Signaturen (siehe „2.1.1. Rechnungssignatur“ auf S. 19) verwendet werden, können problemlos im Windows-MY-Keystore gespeichert werden.

4.2.6.2. JKS und weitere dateibasierte Keystores

Javabasierte Anwendungen verwalten ihre Zertifikate und privaten Schlüssel meist in Dateien mit der Endung `jks`. Dies sind sogenannte „Javakeystores“ (JKS). Listing 4.7 zeigt den Zugriff auf solch einen Keystore. Wieder wird dieser zuerst instantiiert, bevor er in Zeile 4 geladen wird. Dazu wird im Parameter `keyStoreStream` ein Klassenobjekt übergeben, welches die Klasse `java.io.InputStream` implementiert, um von der Keystoredatei zu lesen. JKS können schon beim Zugriff auf einen Keystore die Angabe eines Passwortes verlangen. Dieses wird in Form eines character-Arrays als zweiter Parameter übergeben. Alternativ zu der gezeigten Variante könnte der Keystore auch über die innere Klasse `java.security.KeyStore.Builder` erzeugt werden. An deren `newInstance`-Methode kann als dritter Parameter ein `java.security.KeyStore.CallbackHandlerProtection`-Objekt übergeben werden, welches bei Bedarf ein Passwort abfragen kann.

```
1 String alias = "Name des Certificate-Alias"
2
3 java.security.KeyStore ks = java.security.KeyStore.getInstance("JKS");
4 ks.load(keyStoreStream, keyStorePassCharArray);
5
6 java.security.Key privKey =  ks.getKey(alias, keyPasswordCharArray);
7 java.security.cert.Certificate[]  certChain = ks.getCertificateChain(alias
    );
8 java.security.cert.Certificate signingCert = ks.getCertificate(alias);
```

Listing 4.7: Zugriff auf einen JKS

Neben Keystores vom Typ JKS gibt es noch verschiedene andere Keystoretypen, welche mit JCA/JCE auf nahezu die selbe Art und Weise verwaltet werden können. JCEKS z. B. zeichnet sich dadurch aus, dass auch symmetrische Schlüssel aufgenommen werden können. PKCS12 ist ein Format zum sicheren Speichern eines privaten Schlüssels und dem zugehörigen Zertifikat in einer Datei. Oft liefern ZDA ihre Softwarezertifikate

und die dazu gehörigen privaten Schlüssel auf diese Weise aus. PKCS12-Dateien können direkt als Keystore verwendet werden, man kann aber auch die darin enthaltenen Schlüssel und Zertifikate in einen JKS importieren.

Keystores wie JKS können mit dem Kommandozeilentool *Keytool* verwaltet werden. Eine etwas komfortablere, grafische Möglichkeit bietet das Programm *KeyTool IUI*[16].

Auch die hier beschriebene Art von Keystore ist dazu geeignet, Signaturerstellungsdaten zur Erstellung von fortgeschrittenen Signaturen aufzunehmen.

4.2.6.3. Smartcards

Qualifizierte Signaturen müssen laut `SigG` (siehe „2.1.1. Rechnungssignatur“ auf S. 19) mittels eines SSCD erstellt werden, was im Normalfall einer SmartCard entspricht. Aber auch bei der Erstellung fortgeschrittener Signaturen kann es notwendig und sinnvoll sein, eine SmartCard zu verwenden, etwa weil der private Schlüssel nur in dieser Form zur Verfügung steht oder weil ein höheres Sicherheitsniveau erreicht werden soll.

Das höhere Sicherheitspotential wird dadurch erreicht, dass der private Schlüssel bei der Signatur die SmartCard nie verlässt bzw. diese gar nicht verlassen kann.

Um mittels SmartCards zu signieren wird ein Kartenlesegerät benötigt. Solche Lesegeräte existieren in verschiedenen Sicherheitsklassen:

Klasse 1 Solche Lesegeräte stellen lediglich die Kommunikation mit der Karte her

Klasse 2 Lesegeräte dieser Klasse verfügen über ein eigenes Pinpad zur Eingabe eines Passwortes zur Auslösung der Signatur. Dieses wird in keinem Fall an den angeschlossenen PC übertragen.

Klasse 3 Diese Lesegeräte verfügen nicht nur über ein Pinpad, sondern auch über ein Display zur Anzeige von Daten.

Sollen mit einer SmartCard qualifizierte Signaturen nach dem `SigG` erzeugt werden, wird von der Firma A-Trust die Verwendung eines Kartenlesers empfohlen, der mindestens der Sicherheitsklasse 2 entspricht. Sind mehrere Rechnungen mit einer qualifizierten Stapelsignatur zu versehen, müssen einige der bereits in „Abschnitt 2.1.2 – Server- und Massensignatur“ beschriebenen Punkte beachtet werden:

1. Eine Signatur darf nur nach Eingabe einer PIN erfolgen (nach § `4 Abs 2 SigV`).

[16] `http://yellowcat1.free.fr/keytool_iui.html`

2. Bei Stapelsignaturen muss der Signator die Anzahl der zu signierenden Rechnungen kennen (nach § `18 Abs 2 SigG`).

3. Der PIN-Code darf über den Signaturvorgang hinaus nicht im Speicher verbleiben (nach § `4 Abs 2 SigV`).

4. Dem Signator muss die Möglichkeit gegeben werden, die zu signierenden Rechnungen anzuzeigen (nach § `4 Abs 1 SigV`).

5. Aus den zu signierenden Dokumenten müssen dynamisch codierbare Elemente ausgefiltert werden (nach § `4 Abs 1 SigV`).

Ein Ziel der Diplomarbeit war es, für die InvoiceManager-Anwendung die grundsätzliche Machbarkeit der Erstellung qualifizierter Signaturen zu zeigen und bei entsprechenden gesetzlichen Änderungen diese Art der Signatur auch in der Anwendung anzubieten. Dazu wird das „a.sign premium“-Zertifikatsprodukt[17] der Firma A-Trust und ein Kartenleser vom Typ „Reiner SCT cyperJack pinpad stapelsignatur“ verwendet. Dieser Kartenleser der Sicherheitsklasse 2 speichert den eingegebenen PIN für eine gewisse vorher anzugebende Anzahl von Signaturen. Abbildung 4.9 zeigt wie über das Pinpad des Kartenlesers zuerst die Anzahl der anzufertigenden Signaturen und dann der Pincode eingegeben werden muss.[18] [7] zeigt die Verwendung des Kartenlesers anhand des Quellcodes einer Beispielanwendung in der Programmiersprache C++.

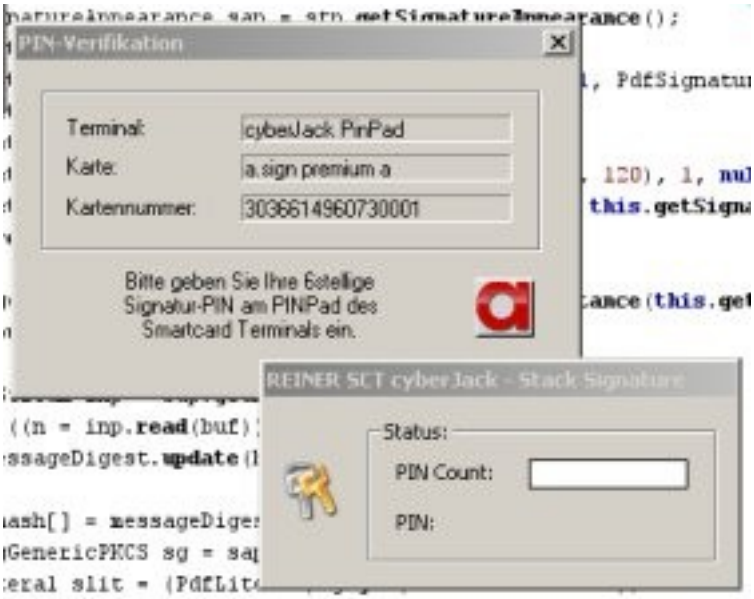

Abbildung 4.9.: Massensignatur mit dem „cyberJack pinpad Stapelsignatur“

[17] Siehe „3.6.3. Auswahl eines Zertifikats“ auf S. 67

[18] Diese Funktionalität wird in Zusammenarbeit mit dem später noch beschriebenen „a.sign Client“ und dessen PKCS11-Modul ermöglicht.

Durch diesen speziellen Kartenleser werden die Punkte 1 bis 3 der Anforderungsliste an qualifizierte Signaturen abgedeckt. Der derzeitige Prototyp der InvoiceManager-Anwendung erlaubt dadurch die Erstellung von Signaturen mit dem auf der „a.sign premium"-SmartCard ebenfalls enthaltenen fortgeschrittenen Verschlüsselungszertifikat. Ebenso könnten damit Signaturen auf Basis eines qualifizierten Signaturzertifikats erstellt werden. Dies sind allerdings noch keine qualifizierten Zertifikate, da dafür auch die Punkte 2 bis 4 der Liste erfüllt sein müssten. Punkt 4 ließe sich bei der Signatur von PDF-Rechnungen (siehe „4.3. Rechnungsformate" auf S. 103) z. B. dadurch erreichen, indem man die Rechnung einer `PDF - TIFF - PDF`-Konvertierung unterzieht [51]. Dabei würde die PDF-Rechnung in ein Bild umgewandelt werden, wodurch alle dynamisch codierbaren Inhalte sicher entfernt werden würden. Danach würde das Bild wieder in ein PDF-Dokument gewandelt und signiert. Punkt 3 ließe sich für PDF-Rechnungen dadurch gewährleisten, dass man eine Möglichkeit bietet, die so konvertierte PDF-Rechnung im „Adobe Acrobat Reader" anzuzeigen. Dies ist die Art und Weise, wie diese beiden Punkte z. B. vom Produkt „a.sign multisign" der Firma A-Trust umgesetzt werden.

Sollten die gesetzlichen Anforderungen an die Rechnungssignatur verschärft werden, wäre es sicher möglich, die derzeitige experimentielle Unterstützung für SmartCards und qualifizierten Signaturen im InvoiceManager auszubauen. In einem Experteninterview mit Herrn Franz Gepp von der Firma A-Trust wurde allerdings festgestellt, dass zur Erstellung von qualifizierten Signaturen im Sinne der A-Trust eine Begutachtung der Anwendung durch die Firma A-Trust nötig wäre.

Im Folgenden werden diese rechtlichen Anforderungen beiseite gelegt und aus rein technischer Sicht geschildert, wie die Signatur mit einem auf einer SmartCard aufbewahrten Schlüssel mittels einer Java-Applikation funktioniert.

Zur Kommunikation mit SmartCards hat sich der „PKCS11-Standard"[19] etabliert, welcher Schnittstellen zur Kommunikation definiert. Das von PKCS11 zur Verfügung gestellte API trägt den Namen „cryptographic token interface" oder kurz „Cryptoki". Implementierungen dieses APIs liegen meist als native C/C++-Bibliotheken, also als *DLL*-Dateien unter Windows bzw. als *SO*-Dateien unter Linux, vor. Diese Bibliotheken werden in der Regel vom Herausgeber der SmartCard zur Verfügung gestellt. Zur Signatur mit „A.sign premium"-Zertifikaten und Schlüssel, welche sich auf einer SmartCard befinden[20], wird durch die Software „A.sign Client" ein solches PKCS11-Modul installiert. Bei einer Windows-Standardinstallation ist dieses im Verzeichnis `C:\windows\system32\asignp11.dll` zu finden.

[19] PKCS steht für „Public Key Cryptography Standard"
[20] Z. B. einer herkömmlichen Bankomatkarte des Types „ACOS EMV"

Zur Kommunikation mit SmartCards in Java-Anwendungen gibt es im Rahmen des *OpenSC-Projektes*[21] Bestrebungen, ein solches API auch nativ für Java zu entwickeln. Für einen praktischen Einsatz wird derzeit allerdings meist die vom Herausgeber der SmartCard zur Verfügung gestellte native Windows- oder Linux-Bibliothek über JNI angesprochen. Dem Autor sind drei Produkte bekannt, welche eine solche Kommunikation ermöglichen:

SunPKCS11 Provider Dieser Provider wird in der Datei `java.security` registriert und kann dann wie in „Abschnitt 4.2.4 – JCA/JCE“ beschrieben, von der JCA/JCE verwendet werden. Bei der Providerregistrierung muss der Pfad zu einer Datei angegeben werden, in welcher der Provider selbst konfiguriert wird. Diese Konfigurationsdatei muss mindestens den Namen des Providers sowie den Pfad zum PKCS11-Modul enthalten. Die genaue Konfiguration dieses Providers ist in [64] beschrieben.

Bei der Evaluierung dieses Providers wurde festgestellt, dass er sich sehr gut zur Erzeugung von *RSA*-Signaturen eignet. Der Versuch mit diesem Provider im Zusammenspiel mit der ECC-Unterstützung des Bouncy Castle-Providers, Signaturen auf Basis des qualifizierten „a.sign premium“-Zertifikats durchzuführen, scheiterte trotz intensiver Konfigurationsversuche.

IAIK-PKCS11-Provider Dieser JCA/JCE-Provider wird vom „Institute for Applied Information Processing and Communication“ (IAIK) der TU Graz entwickelt. Es handelt sich dabei um kommerzielle Software, welche aber für reine Forschungsprojekte sowie zur Lehre und für Open Source-Projekte gebührenfrei lizenziert werden kann.[22] [61]. Es handelt sich dabei ebenfalls um einen Provider, welcher in der JCA/JCE registriert werden kann. Im Rahmen der Diplomarbeit wurde dieser Provider im Zusammenspiel mit der ebenfalls unter den selben Lizenzen erhältlichen ECC-Bibliothek des IAIK erfolgreich in die InvoiceManager-Anwendung integriert. Das grundsätzliche Erstellen von Rechnungssignaturen auf Basis des qualifizierten „a.sign premium“-Zertifikats war dadurch möglich.

Da es sich bei der InvoiceManager-Applikation im Moment allerdings nicht um ein Open Source-Projekt handelt und dieses auch zu kommerziellen Zwecken eingesetzt werden wird, ist diese Unterstützung in der im betrachteten Betrieb eingesetzten Version nicht mehr vorhanden. Die InvoiceManager-Applikation ist jedoch so modular gestaltet, dass sie sehr einfach wieder um diese Unterstützung erweitert werden könnte.

[21] `http://www.opensc-project.org/opensc-java/`

[22] Es werden eine „educational licence“, eine „open source licence“, und eine „commercial licence“ angeboten.

IAIK-PKCS11-Wrapper Diese ebenfalls vom IAIK entwickelte Software wird laut eigenen Aussagen des IAIK unter einer „Apache-style licence“ zur Verfügung gestellt, kann also grundsätzlich auch in kommerziellen Projekten verwendet werden [62]. Der Wrapper bietet ein API, um aus Java über ein PKCS11-Modul aus auf eine SmartCard zugreifen zu können. Es handelt sich dabei aber nicht um einen JCA/JCE-Provider, was die Integration in eine darauf aufbauende Anwendung schwierig machen würde. Der IAIK-PKCS11-Provider basiert auf diesem Wrapper.

4.3. Rechnungsformate

Die InvoiceManager-Applikation unterstützt aus den in „Abschnitt 3.5 – Soll-Zustand“ erläuterten Gründen die Formate PDF und ebInterface. Dieses Kapitel stellt die beiden Formate vor und zeigt, wie Rechnungen in der InvoiceManager-Anwendung in diesen Formaten erzeugt und signiert werden. Außerdem wird darauf eingegangen, wie Rechnungsempfänger Signaturen auf PDF- und ebInterface-Rechnungen prüfen können. Mögliche weitere Formate für elektronische Rechnungen werden auch in [42] diskutiert.

4.3.1. ebInterface

Dieses auf XML basierende Format wurde vom Verein AustriaPro (siehe „1.4. AustriaPro“ auf S. 15) entwickelt. Es stellt ein standardisiertes und strukturiertes Format dar, welches eine automatisierte Weiterverarbeitung ermöglicht. Die Forderung nach einem solchen Format wurde in dieser Arbeit schon einige Male erhoben, und die Vorteile wurden in „Abschnitt 1.1 – Motivation“ dargelegt.

EbInterface beruht, wie Herr Mag. Lothar Winkelbauer in einem Experteninterview erklärte, ursprünglich auf dem XML-Rechnungsstandard der Firma EBPP (siehe „3.6.2.1. e-Billing auslagern“ auf S. 60). Im Zuge des Projektes ebInterface wurde diese ursprüngliche Spezifikation von den Projektteilnehmern an deren Bedürfnisse angepasst.

XML ist eine Metasprache, auf deren Basis hierachisch strukturierte Formate definiert werden können. Ein solches Format ist eben ebInterface. Durch die weite Verbreitung von XML können darauf aufbauende Formate von den vielen vorhandenen Werkzeugen und Anwendungen profitieren. So kann z. B. XSLT verwendet werden, um die eigentlich für Maschinenlesbarkeit optimierten XML-Dokumente in HTML darzustellen und somit für einen menschlichen Betrachter aufzubereiten.

Das Format ebInterface kennt verpflichtende sowie optionale Elemente[23], wobei darauf zu achten ist, dass die verpflichtenden Elemente nicht mit den gesetzlich vorgeschriebenen Bestandteilen einer Rechnung übereinstimmen.

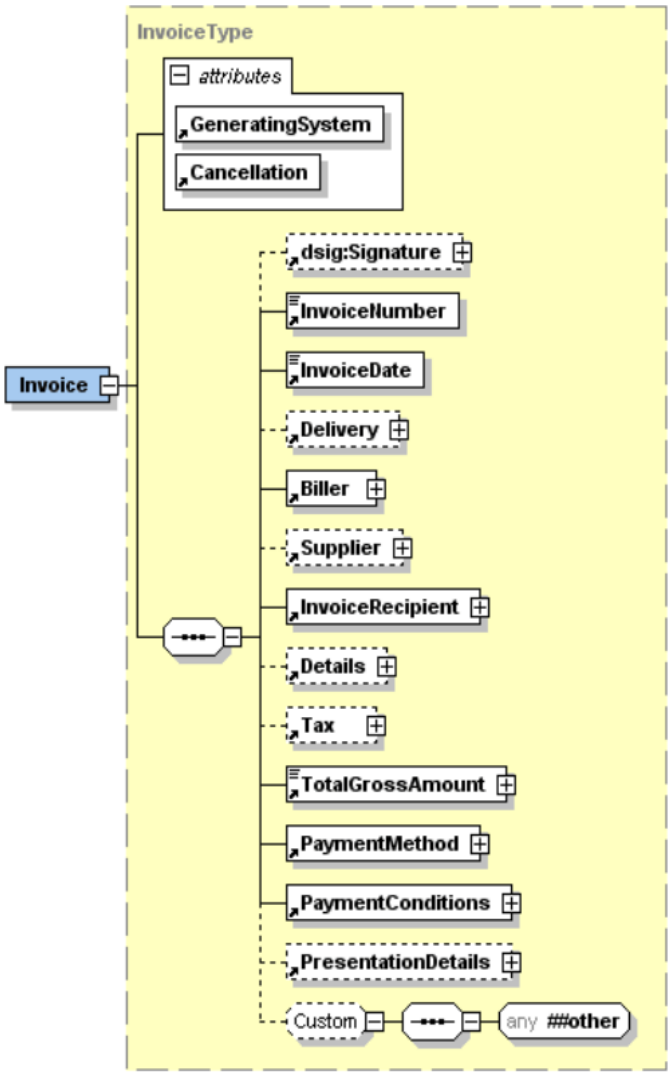

Abbildung 4.10.: Überblick ebInterface XSD [11]

Abbildung 4.10 zeigt den schematischen Aufbau einer ebInterface-Rechnung, welcher im Folgenden kurz beschrieben ist: Eine Rechnung beginnt immer mit dem Tag *Invoice*, welcher alle anderen Tags beinhaltet. Mit dem Attribut *Cancellation* ist es möglich auszudrücken, ob es sich bei dem Dokument um eine Rechnung oder um ein Rechnungsstorno handelt. Von der Möglichkeit Rechnungsstornos abzubilden macht die InvoiceManager-Applikation keinen Gebrauch. Das Element *dsig:Signature* enthält die in „Abschnitt 4.3.1.4 – Signatur von XML-Dokumenten“ beschriebene Rechnungs-

[23] Diese sind in Abbildung 4.10 mit einer nicht durchgängigen Linie gekennzeichnet.

signatur. *InvoiceNumber* enthält die fortlaufende und eindeutige Rechnungsnummer, welche bei der InvoiceManager-Applikation beim Import der Rechnungen übernommen wird. Ebenso wird das Rechnungsdatum übernommen und im Element *InvoiceDate* abgelegt. Das *Delivery*-Element könnte zum Abbilden von Lieferdaten verwendet werden. Die vorliegende Anwendung macht hier nur von der Möglichkeit Gebrauch, eine Lieferperiode mit Start- und Enddatum anzugeben. Im *Biller*-Element werden die Daten des Rechnungsstellers angegeben, während das *InvoiceRecipient*-Element die Empfängerdaten abbildet. Das *Supplier*-Element könnte die Daten eines Lieferanten aufnehmen und wird in der InvoiceManager-Anwendung nicht verwendet. Die *Details*-Komponente nimmt die einzelnen Rechnungszeilen auf, während im *Tax*-Element die Rechnungsumsatzsteuer ausgewiesen ist. Der Gesamtbetrag der Rechnung ist aus dem *TotalGrossAmount* ersichtlich. Als Zahlungsmethode können im Element *PaymentMethod* die Elemente *UniversalBankTransaction*, *DirectDebit* oder *NoPayment* vorkommen, welche auch von der entwickelten Applikation verwendet werden. Das Element *PaymentCondition* erlaubt die Angabe eines Zahlungszieles sowie die Gewährung von Skonti und Rabatten. Im betrachteten Betrieb besteht im Augenblick nur die Notwendigkeit, Rabatte abbilden zu können. Im *PresentationDetails*-Element können einige für die Präsentation der Rechnung wichtige Informationen wie der Pfad zu einem Logo oder die Sprache der Rechnung angegeben werden. Die InvoiceManager-Applikation erlaubt die Aufnahme eines Logos, die Angabe einer Homepageadresse, eines Sprachcodes, sowie zusätzlicher Kommentare. Eine sehr detaillierte Beschreibung des Standards in der Version 2.1 ist unter [11] zu finden.

EbInterface zeichnet sich durch eine sehr hohe Flexibilität aus, welche vor allem durch die Elemente *Details* und *Custom* erreicht wird. Das *Custom*-Element erlaubt die Einbindung eines fremden XML-Namespaces in eine ebInterface-Rechnung. Dadurch können beliebige zusätzliche Informationen in die Rechnung aufgenommen werden. Eventuell wäre es denkbar, dass es in einigen Branchen z. B. zusätzliche gesetzlich vorgeschriebene Rechnungsbestandteile gibt, die sich durch ebInterface-Bordmittel nicht abbilden lassen. Durch Definition eines eigenen XML-Namespaces und das Einfügen von Elementen daraus im *Custom*-Element ist es möglich, solche Informationen in die Rechnung aufzunehmen. Mit Technologien wie *XPointer*-Ausdrücken können diese zusätzlichen Informationen mit den Inhalten der eigentlichen Rechnung verknüpft werden. Ein offensichtlicher Nachteil dieser Erweiterungsmöglichkeit ist, dass Standardunternehmenssoftware die Angaben im *Custom*-Element wahrscheinlich ignorieren wird, da sie nichts von einem fremden Namespace wissen wird.

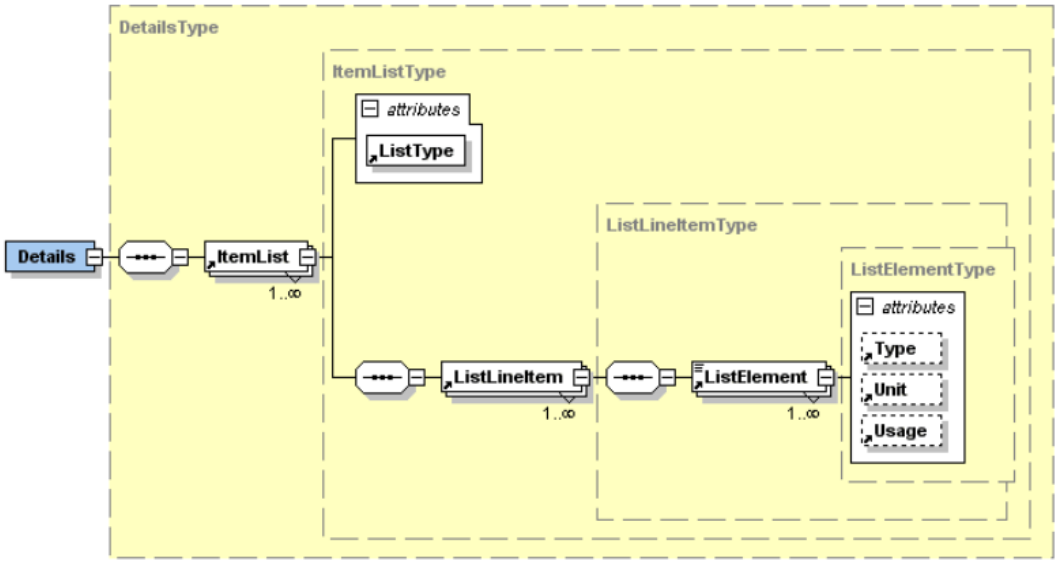

Abbildung 4.11.: Das Details-Element von ebInterface [11]

Ebenfalls sehr flexibel gestaltet sich der *Details*-Bereich von ebInterface. Dieses Element bildet die Rechnungspositionen ab. Eine Liste von Rechnungspositionen wird als *ItemList* dargestellt. Eine Rechnungsposition entspricht einem *ListLineItem*, welches wiederum aus mehreren *ListElement*-Feldern besteht. Ein *ListElement* ist innerhalb einer Rechnungsposition genau ein Wert wie z. B. Preis, Beschreibung oder Quantität. Den Inhalt und die Verwendung eines *ListElement*-Feldes kann man über die Attribute *Type*, *Unit*, und *Usage* nahezu beliebig definieren. [11] beinhaltet eine Übersicht über die möglichen Werte für diese Attribute.

Listing 4.8 zeigt den *Details*-Bereich einer mit der InvoiceManager-Applikation importierten Rechnung. Diese Rechnung besteht nur aus einer einzigen Rechnungsposition. In Zeile 4 definiert das erste *ListElement* das Datum der Konsumation. Zeile 5 definiert eine interne Transaktionsnummer des Systems, welche auch auf der Rechnung aufscheinen soll. Zeile 6 speichert die Nummer der Kundenkarte, mit welcher die Konsumation getätigt wurde. Zeile 7 beschreibt die konsumierte Ware. Die von dieser Ware konsumierte Menge wird in Zeile 8 festgehalten. Hier wird neben *Usage* und *Type* auch noch angegeben, in welcher Einheit die Ware verrechnet wird. Der Preis pro Einheit ist in Zeile 9 beschrieben. Als *Unit* wird hier die Währung angegeben. Schlussendlich gibt die Zeile 10 den Gesamtpreis der Rechnungsposition an.

Dieser Ansatz funktioniert sehr gut, wenn nur die im Standard definierten *Usage*-Werte verwendet werden. Allerdings müssen schon im relativ simplen in Listing 4.8 gezeigten Beispiel mit dem Datum der Konsumation, der Transaktionsnummer und der Kartennummer drei Extraelemente in die Rechnungsposition aufgenommen werden, für die es keine genaue semantische Definition durch ein *Usage*-Attribut gibt. Die

InvoiceManager-Applikation kann diesen Elementen die richtige Bezeichnung zuordnen, da sie z. B. weiß, dass an dritter Stelle einer Rechnungsposition immer die Kartennummer steht. Eine Drittanwendung, wie z. B. die in „Abschnitt 4.3.1.5 – Verifikation von ebInterface-Rechnungen“ beschriebene ebRe-Anwendung, wird zwar z. B. das *ListElement* aus Zeile 10 anhand des *Usage*-Attributes als Gesamtsumme identifizieren können, sie kann aber nicht wissen, dass mit dem *ListElement* aus Zeile 6 die Kartennummer gemeint ist. Man kann hier praktisch für eine Anwendung des Standards einen Unterstandard definieren. Diese Flexibilität ist einerseits vorteilhaft, wenn zusätzliche Informationen abgebildet werden sollen, kann aber andererseits die Interoperabilität beeinträchtigen.

```
1 <eb:Details>
2   <eb:ItemList eb:ListType="structured">
3     <eb:ListLineItem>
4       <eb:ListElement eb:Usage="Description" eb:Type="DateType">02.10</
            eb:ListElement>
5       <eb:ListElement eb:Usage="Number" eb:Type="IdentifierType">7501</
            eb:ListElement>
6       <eb:ListElement eb:Usage="Number" eb:Type="CodeType">11230</
            eb:ListElement>
7       <eb:ListElement eb:Usage="Description" eb:Type="StringType">DIESEL <
            /eb:ListElement>
8       <eb:ListElement eb:Usage="Quantity" eb:Unit="l" eb:Type="StringType"
            >43.1845</eb:ListElement>
9       <eb:ListElement eb:Usage="UnitPrice" eb:Unit="EUR" eb:Type="
            AmountType">1654.6</eb:ListElement>
10      <eb:ListElement eb:Usage="Amount" eb:Unit="EUR" eb:Type="AmountType"
            >1.8</eb:ListElement>
11    </eb:ListLineItem>
12  </eb:ItemList>
13 </eb:Details>
```

Listing 4.8: Beispiel eines Details-Bereiches

Die zusätzlichen Informationen Datum, Transaktionsnummer und Kartennummer hätten auch mit etwas mehr Implementierungsaufwand im *Custom*-Element abgebildet werden können, anstatt diese in das *Details*-Element aufzunehmen. Für die Anzeige der Informationen in Drittsystemen läuft dies, wie eben beschrieben, auf ein Abwägen zwischen keiner Anzeige und Anzeige unter falscher Überschrift hinaus.

4.3.1.1. Internationalisierung des Standards

Für die Verwendung von ebInterface als Format der elektronischen Ausgangsrechnungen des betrachteten Betriebes, ist eine breite internationale Akzeptanz der Spezifikation nicht entscheidend, da alle Kunden des Unternehmens im Inland beheimatet sind. Für

Unternehmen deren Kundschaft teilweise aus ausländischen Unternehmen besteht, ist die Wichtigkeit dieses Punktes allerdings nicht zu unterschätzen und somit auch entscheidend für die Verbreitung und die mittel- und langfristige Überlebensfähigkeit von ebInterface. Im Projekt ebCrossborder verfolgte AustriaPro das Ziel, ebInterface international zu etablieren. Im Jänner 2008 wurde der Abschlussbericht zu diesem Projekt veröffentlicht [13][24]. Darin werden die Ziele, Inhalte und Ergebnisse des Projektes dokumentiert. Die Wichtigkeit einer internationalen Verwendbarkeit von ebInterface wird in dem sehr lesenswerten Bericht folgendermaßen beschrieben[25]:

> *„Ein Standard, welcher nur in Österreich Verwendung findet, wird von österreichischen Unternehmen, welche in einem hohen Maße von internationalen Geschäften abhängig sind, nicht akzeptiert werden. Ein isolierter Länderstandard würde 'gegen' internationale Standards nicht überleben, egal wie lange diese brauchen würden um sich zu etablieren.“*

Der ebCossborder-Abschlussbericht beschreibt die Aktivitäten des Projektes wie folgt [13]:

Festigen der Beziehungen zu internationalen Standardisierungsbehörden Dadurch sollte die ebInterface-Spezifikation bei den europäischen Standardisierungsbehörden und e-Billing-Initiativen bekannt gemacht und auf einem europäischen Level positioniert werden.

Cross-Border Case Studies Um herauszufinden, inwiefern ebInterface für den grenzüberschreitenden Austausch von Rechnungen verwendbar ist, wurde versucht die rechtlichen, wirtschaftlichen und technischen Gegebenheiten in verschiedenen Nachbarländern Österreichs zu evaluieren. Dazu wurden umfangreiche Fragebögen[26] von e-Billing-Experten in Deutschland, Italien, Slowenien, Serbien und der Schweiz ausgefüllt, welche dann vom ebCrossborder-Team ausgewertet wurden. Der Fragebogen enthält vor allem Fragen zu den rechtlichen Anforderungen und zu den im jeweiligen Land verwendeten e-Billing-Standards. Die daraus gewonnenen Erkenntnisse können in [13] nachgelesen werden.

Cross-Border eInvoicing Experiment In Ermangelung eines einheitlichen europäischen Standards ist nicht nur in Österreich eine eigene Spezifikation entstanden, sondern auch andere europäische Länder haben nationale Formate zum elektronischen Rechnungsaustausch entwickelt. In dieser Phase des ebCrossborder-Projektes

[24] Eine Zusammenfassung des Berichtes ist unter [14] erhältlich.
[25] Anm.: Der ebCrossborder-Abschlussbericht liegt in englischer Sprache vor. Die hier zitierten Ausschnitte wurden vom Autor ins Deutsche übersetzt.
[26] Der Fragebogen ist im Anhang des ebCrossborder-Endberichtes abgedruckt [13].

wurde, in Zusammenarbeit mit Partnern aus Italien und Slowenien, versucht elektronische Rechnungen zwischen den nationalen Formaten zu konvertieren[27].

Dabei konnte laut [13] bei allen durchgeführten Mappings zwischen den Formaten eine 100 %ige Abbildung von zwingend erforderlichen Feldern erreicht werden. Bei einer Transformation von Rechnungen im Format ebInterface 2.1 in den italienischen Standard CBI 2.0 konnten 79 % aller Felder[28] und Konzepte abgebildet werden [13].

Laut [13] zeigen die durchgeführten Transformationen, dass ebInterface in der Version 2.1 eine gute Basis für den grenzüberschreitenden elektronischen Rechnungsaustausch darstellt. Als größte Schwierigkeit wurden die unterschiedlichen rechtlichen und wirtschaftlichen Anforderungen in den verschiedenen Ländern identifiziert und es wird ein dringender Bedarf für einen einheitlichen und harmonisierten Zugang zur elektronischen Rechnung gesehen. Mit Hinblick auf den schleppenden Verlauf der Diskussion zu den zukünftigen rechtlichen Anforderungen an elektronische Rechnungen in Österreich (siehe „2.3. Vorgeschlagene Änderungen“ auf S. 30) ist eine solche Harmonisierung, aus Sicht des Autors der vorliegenden Diplomarbeit, wenigsten in nächster Zeit nicht zu erwarten.

Analyse der internationalen e-Billing-Standardisierungstrends Dieser Teil des ebCrossborder-Abschlussberichtes beschreibt die unterschiedlichen Standards im Bereich der elektronischen Rechnung und deren wahrscheinliche zukünftige Entwicklung. Es wird hervorgehoben, dass die Mehrheit der Standards, wie z. B. *UN/EDIFACT*, einen Top-Down-Ansatz verfolgen, bei welchem versucht wird, in einem langen Spezifikationsprozess, alle nur erdenklichen Anwendungsgebiete und Spezialfälle im Standard abzudecken. Durch die überwältigende Anzahl an möglichen Datenelementen und die dadurch oft mögliche ambivalente Abbildung von semantisch gleichwertigen Informationen wird auch die Interoperabilität eingeschränkt. Partnerspezifische Vereinbarungen (partner-specific agreements) zwischen den einzelnen Unternehmen müssen dann sicherstellen, dass elektronische Dokumente auch wirklich von beiden Seiten richtig verarbeitet werden können. EbInterface dagegen verfolgt laut [13] einen Bottom-Up-Ansatz, da es eine sehr schlanke Spezifikation darstellt, welche nur die grundsätzlichen branchenübergreifenden Anforderungen an eine Rechnung abdeckt, und so relativ einfach von Standardsoftware unterstützt werden kann. Der Endbericht verschweigt aber auch nicht, dass ebInterface zum Einsatz in verschiedenen Bereichen um branchenspezifische Plugins erweitert werden muss.

[27] Die beteiligten Formate und die durchgeführten Konvertierungen sind im ebCrossborder-Abschlussbericht beschrieben.
[28] Also sowohl zwingend erforderliche als auch optionale Felder

Obwohl der ebCrossborder-Abschlussbericht den Top-Down-Ansatz der etablierten Standards kritisiert, möchte man sich doch deren positive Aspekte zunutze machen. Der ebCrossborder-Endbericht hält dazu fest:

> *„Das größte Problem von ebInterface ist, dass es auf einem propriäteren XML-Schema basiert. Aus unserer Erfahrung in verschiedenen internationalen Aktivitäten, führt dies üblicherweise zu einer sofortigen Ablehnung durch andere Marktteilnehmer.“*

Die Zukunft von ebInterface liegt daher aus Sicht des ebCrossborder-Projektes nicht nur in der Anpassung an die zukünftigen rechtlichen Anforderungen in Österreich, sondern auch in einer Umstellung der XML-Syntax, um diese auf einem allgemein anerkannten XML-Format[29] basieren zu lassen. Gleichzeitig soll allerdings der Bottom-Up-Ansatz gewahrt bleiben. [13]

4.3.1.2. Erzeugung der ebInterface-Rechnungen

Die InvoiceManager-Anwendung importiert Rechnungen aus einem Altsystem und wandelt diese beim Import in ebInterface-Rechnungen um. Die Rechnungen werden dann im ebInterface-Format gespeichert. Das vorhandene System bietet zwei Möglichkeiten an, Rechnungen zu exportieren:

1. Export der fertigen Rechnungen zum Druck in einer Textdatei

2. Export der Rechnungsdaten zur Fremdfakturierung in einer Textdatei

Beide vom Altsystem erzeugten Dateien sind nach einem gleichbleibenden Schema aufgebaut und ermöglichen ein Parsen der erzeugten Textdatei, um die nötigen Daten für ebInterface-Rechnungen zu sammeln. Die Datei zur Fremdfakturierung ist sicher einfacher zu parsen, da eine Formatbeschreibung vorliegen würde. Dafür müsste bei einem Import aus dieser Datei auch die Rechnungsstellung vom InvoiceManager erledigt werden. Das würde bedeuten, dass z. B. auch Rechnungsnummern von der Anwendung vergeben werden müssten. Auch vom Vertreiber des Altsystemes wurde angeraten, diesen Weg nur im Notfall einzuschlagen, da sich möglicherweise unerwünschte Seiteneffekte ergeben könnten, wenn die Rechnungslegung nicht mehr vom Altsystem durchgeführt wird.

Im Moment implementiert der InvoiceManager nur die erste Variante. Die Anwendung ist allerdings betreffend des Importformates modular aufgebaut und so wäre es relativ einfach, diese um weitere Importformate zu erweitern. Ein Importer muss lediglich das

[29] Im Bericht genannt werden *UBL*, *CEFACT CC + NDR* oder eine zukünftige Verschmelzung von beiden.

Interface `at.ws.business.invoiceImport.InvoiceImporter` (Abbildung 4.12) implementieren. Mit der Methode **public void** setInputFile(File inputFile) wird die Rechnungsdatei bekannt gegeben, welche zum Rechnungsimport verwendet wird. Die Methode **public void** setDaoFact(DAOFactory daoFact) wird zum Speichern der Rechnungen in der Datenbank verwendet. Mit setImportMessageListener(ImportMessageListener importMessageListener) kann ein Objekt registriert werden, welches vom Importer über den Status informiert werden kann. Anstatt diese Parameter über die set-Methoden zu übergeben, können sie natürlich auch über einen Konstruktor der implementierenden Klasse gesetzt werden. Wichtig ist die Methode **public void** convert(), welche den Import schließlich anstößt.

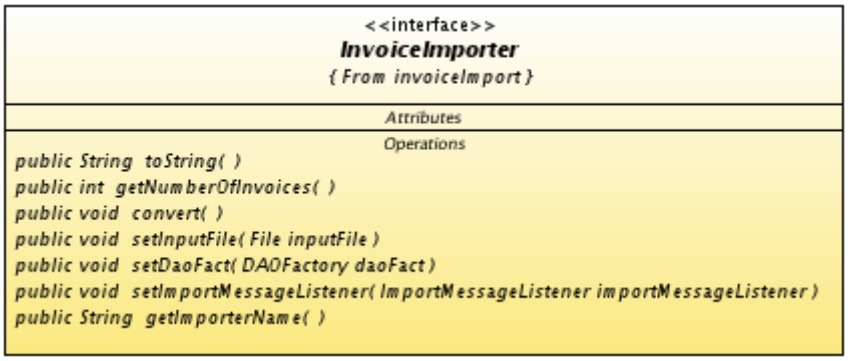

Abbildung 4.12.: Klassendiagramm des Interface `InvoiceImporter`

Die einzelnen ebInterface-Rechnungen werden in der `convert`-Methode über JAXB (Java API for XML Binding) aufgebaut. Bei der Entwicklung der Software wurden dabei die folgenden Schritte durchgeführt:

1. Erzeugung von JAVA-Klassen aus dem ebInterface XML-Schema mittels ANT und JAXB 2.0 (siehe Listing 4.9)

2. Durchführung einiger manueller Anpassungen an den erzeugten Klassen

Beim Parsen der Rechnungen werden dann für jede Rechnung die folgenden Schritte durchgeführt:

1. Aufbau der Rechnung durch die erzeugten JAVA-Objekte

2. Marshalling des Objektbaumes nach XML (siehe Listing 4.10)

```
1 <taskdef name="xjc" classname="com.sun.tools.xjc.XJCTask">
2   <classpath>
3     <fileset dir="/home/test/netbeans-5.5/ide7/modules/ext/jaxws20"
        includes="*.jar" />
```

```
4   </classpath>
5 </taskdef>
6
7 <target name="all">
8   <xjc schema="invoice.xsd" target="./src/" package="at.ws.ebinterface"/>
9 </target>
```

Listing 4.9: Erzeugung von Java-Klassen mit ANT und JAXB

```
1 JAXBContext context = JAXBContext.newInstance(InvoiceType.class);
2 Marshaller m = context.createMarshaller();
3 m.setProperty("com.sun.xml.bind.namespacePrefixMapper", new
      NamespacePrefixMapperImpl());
4 m.setProperty(Marshaller.JAXB_SCHEMA_LOCATION, "http://www.ebinterface.at/
      schema/2p1/http://www.ebinterface.at/schema/2p1/Invoice.xsd");
5
6 StringWriter swriter = new StringWriter();
7 m.marshal(invoice, swriter);
```

Listing 4.10: Erzeugung von XML aus einem Objektbaum

4.3.1.3. Anzeige der ebInterface-Rechnungen

XML ist ein maschinenlesbares Format, welches nicht für die Betrachtung durch den Menschen ausgelegt ist. Dafür existiert XSLT (Extensible Stylesheet Language Transformation), welches eine Transformation von XML in verschiedene andere Formate ermöglicht. Im InvoiceManager ist zur Anzeige der Rechnungen im Browser eine „XML ⇒ HTML“-Transformation implementiert. Um eine Transformation durchzuführen benötigt man eine XSLT-Datei, welche auf die ebInterface-Datei angewandt wird, um diese von einem XSLT-Prozessor wie „Xalan“ oder „Saxon“ transformieren zu lassen. Zur XSLT-Verarbeitung wird in Java „JAXP“ (Java API for XML Processing) verwendet.

Die XSLT-Datei beruht auf der von AustriaPro zur Verfügung gestellten XSLT-Datei [10]. Der zur Verfügung gestellte XSLT-Code hat aus Sicht des Autors einige Schönheitsfehler:

- Die zusätzlich abzubildenden Informationen im *Details*-Abschnitt werden nicht richtig bezeichnet (siehe Listing 4.8)
- Alle Datumswerte werden im selben Format angezeigt, wie sie in der XML-Datei gespeichert sind, nämlich im Format „YYYY-MM-DD“. Im deutschsprachigen Raum verbreitet ist jedoch die Datumsangabe im Format „DD.MM.YYYY“.

- Bei allen Beträgen wird das „.“-Zeichen als Kommatrennzeichen verwendet und es werden keine Dezimaltrennzeichen ausgegeben. Beträge sollten das „.“-Zeichen als Dezimalpunkt und das „ ,“-Zeichen als Kommatrennzeichen verwenden.

Zur Behandlung der beiden letzten Punkte wurden XSLT-Funktionen definiert, welche die Konvertierung durchführen. Dabei wurden XPath 2.0-Funktionen verwendet, weshalb auch ein Parser eingesetzt werden muss, welcher diese unterstützt. Zu diesem Zweck wurde der XSLT-Prozessor „Saxon“ verwendet.

Nach der Erzeugung der HTML-Repräsentation wird in einem Verzeichnis eine temporäre HTML-Datei für die darzustellende Rechnung angelegt, welche dann mit dem Standardbrowser des Systems geöffnet wird. Dazu wird die in JSE 6 neu hinzugekommene Klasse `java.awt.Desktop` wie in Listing 4.11 gezeigt verwendet.

```
1 Desktop desktop = Desktop.getDesktop();
2 desktop.browse(new URI(tmpInvoicePath));
```

Listing 4.11: Anzeige der Rechnung im Browser über die Desktop-Klasse

Um dem Datenschutz Genüge zu tun, wird durch die Verwendung der Methode **public void** deleteOnExit() der Klasse `java.io.File` sichergestellt, dass die durch die Methode **public static** File createTempFile(String prefix, String suffix, File directory)[30] erzeugte temporäre Datei nach Beendigung der Anwendung gelöscht wird. Das Verzeichnis in welchem die temporären Dateien erzeugt werden ist konfigurierbar (siehe „4.4.3. Konfiguration“ auf S. 136).

4.3.1.4. Signatur von XML-Dokumenten

Durch den Aufbau auf dem XML-Standard kann zur Signatur ein weiterer XML-Standard verwendet werden, nämlich die „XML-Signature“-Spezifikation [69]. Dieser, unter seiner Kurzbezeichnung „XMLDsig“ bekannte Standard, definiert eine XML-Notation für das Verarbeiten und Darstellen von digitalen Signaturen. Grundsätzlich kann mit dieser Technologie jede Art von Daten signiert werden. Im vorliegenden Fall handelt es sich bei den zu signierenden Rechnungen ebenfalls um XML-Dokumente. Eine Signatur nach diesem Standard zeichnet sich dadurch aus, dass alle Informationen, die zum Verifizieren gebraucht werden, bereits im XML-Signature-Dokument enthalten sind. So kann das *dsig:Signature*-Element einer ebInterface-Rechnung z. B. die Base64-codierte Signatur, Hinweise auf die verwendeten Algorithmen, Referenzen auf die verwendeten Schlüssel oder aber auch eine ganze Zertifikatskette in Base64-codierter Form enthalten. XMLDsig kennt drei verschiedene Arten von Signaturen:

[30] Diese Methode ist ebenfalls Bestandteil der File-Klasse.

Detached dabei liegen die zu signierenden Daten außerhalb des XML-Signature-Dokumentes

Enveloping dabei sind die zu signierenden Daten Teil des XML-Signature-Dokumentes

Enveloped dabei ist die XML-Signature ein Teil des Elementes, welches signiert werden soll

Im ebInterface-Standard wird die letztgenannte Variante verwendet. Abbildung 4.10 enthält dazu das optionale Element *dsig:Signature*, welches die Signatur aufnehmen kann.

```
<Signature>
  <SignedInfo>
    <CanonicalizationMethod .... />
    <SignatureMethod .... />
    <Reference URI= ... >
      <Transforms ... />
      <DigestMethod ... />
      <DigestValue> .... </DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue> .... </SignatureValue>
  <KeyInfo>
    <X509Data>
      <X509Certificate> .... </X509Certificate>
    </X509Data>
  </KeyInfo>
</Signature>
```

Listing 4.12: Aufbau eines XML-Signature-Elementes

Im Folgenden ist die Funktionsweise einer „enveloped XML-Signature“ kurz beschrieben:

Eine XML-Signature beginnt immer mit dem root-Element *Signature*. Listing 4.12 zeigt den grundsätzlichen Aufbau dieses Elementes. Darin enthalten ist zuerst einmal das Element *SignedInfo*, welches die Information darstellt die signiert werden soll. Das heißt, signiert wird der Inhalt des SignedInfo-Elementes. Aus diesem Grunde muss man auch im darin enthaltenen Element *CanonicalizationMethod* einen Hinweis auf einen Algorithmus angeben, welcher zur Normalisierung der in SignedInfo enthaltenen XML-Daten verwendet werden kann. Dies ist notwendig, da es in XML möglich ist, dass zwei semantisch völlig gleiche Inhalte verschiedene Darstellungen haben und diese daher zuerst in eine einheitliche Form gebracht werden müssen, bevor signiert werden kann. Sonst würde eine spätere Verifikation immer scheitern. Mögliche Canonicalization-Algorithmen sind in [69] angegeben. Danach muss innerhalb von *SignatureMethod* angegeben wer-

den, welcher Algorithmus zur Erzeugung des Hashwertes und der Signatur der Daten in *SignedInfo* verwendet werden soll. Das nun angegebene *Reference*-Element stellt nun einen Verweis auf die Daten dar, welche signiert werden sollen. Da es sich bei den ebInterface-Rechnungen um XML-Dokumente handelt, in welchen die Signatur schlussendlich enthalten sein soll, muss als Ziel der Reference, welches als URI dargestellt ist, immer der leere String angegeben werden. Das Starttag des *Reference*-Tags muss also immer <ds:Reference URI=""> lauten. Weiters muss bei der Signatur von ebInterface-Rechnungen immer das *enveloped-signature-Transforms*-Objekt angegeben werden. Dieses weist den Algorithmus an, bei der Kalkulation des Hashwertes der Rechnung das *Signature*-Element auszusparen. Weiters muss auch hier wieder ein Canonicalization-Algorithmus angegeben werden. Ebenfalls im *Reference*-Element wird dann eine *DigestMethod* angegeben die festlegt, mit welchem Hashalgorithmus das *Reference*-Objekt, also die Rechnung, bearbeitet werden soll. Wie weiter oben bereits erwähnt wurde, ist auch im SignatureMethod-Element ein Digest-Algorithmus enthalten. Dieser bezieht sich aber bereits auf das Verarbeiten der Inhalte im *SignedInfo*-Tag. Der Algorithmus in *DigestMethod* dagegen legt nur fest, wie das Objekt „gehasht“ werden soll, welches über Reference angesprochen wird. Der resultierende Hashwert wird danach in das Element *DigestValue* geschrieben. Ein SignedInfo-Element könnte auch mehrere *Reference*-Elemente enthalten, und so könnten z. B. auch Logos, auf die in der Rechnung verwiesen wird, mitsigniert werden. Das Element *KeyInfo* schlussendlich enthält einen Hinweis darauf, mit welchem Schlüssel die Signatur verifiziert werden kann. Dieser Hinweis kann die Form einer Seriennummer eines Zertifikats haben, es kann aber auch sein, dass der Base64-codierte Schlüssel, oder sogar die ganze Zertifikatskette, selbst enthalten ist.

Sun bietet mit JSR-105 (Java Specification Request 105) zwar eine Referenzimplementierung für XML-Dsig an, diese unterstützt aber nur wenige Signaturalgorithmen. Die Implementierung der „Apache Foundation“ im Projekt „Apache XML Security“[31] bietet eine umfangreichere Auswahl an Algorithmen und unterstützt z. B. auch den *SHA1withECDSA*-Algorithmus. Bei der Verwendung dieses APIs ist zu beachten, dass es vor der Verwendung über einen Aufruf der Methode org.apache.xml.security.init.init() zu initialisieren ist. Listing 4.13 zeigt wie bei Apache XML Security zur Erzeugung der Signatur ein spezieller Provider angegeben werden kann (siehe analog dazu „4.2.6. „Key Stores“ “ auf S. 96 oder „4.2.4. JCA/JCE“ auf S. 86).

```
1 JCEMapper.setProviderId(cryptoProvider.getName());
```

Listing 4.13: Spezifizierung eines Providers für kryptographische Operationen bei Apache XML Security

[31] http://santuario.apache.org/

4.3.1.5. Verifikation von ebInterface-Rechnungen

Im Auftrag des BMWA hat die Firma Mesonic die Anwendung *ebRe* (e-Billing Rechnungseingangsbuch) entwickelt [27], welche zur Förderung der e-Rechnung kostenlos verwendet werden kann.[32] Bei ebRe handelt es sich um eine auf den Microsoft Betriebssystemen lauffähige Applikation zur Archivierung und Signaturprüfung von ebInterface-Rechnungen. Zur Signaturprüfung wird natürlich eine aktive Internetverbindung benötigt. Zu prüfende Rechnungen müssen in einem In-Verzeichnis abgelegt werden, damit sie einer Signaturprüfung unterzogen werden. Eine ausführliche Beschreibung dieses Programmes ist unter [49] erhältlich.

Rechnungsempfänger könnten darauf hingewiesen werden, dass sie dieses Programm kostenlos zur Signaturverifikation ihrer Rechnungen verwenden können. Eine Eigenentwicklung zur Verifikation wäre aber in Zukunft eventuell wünschenswert, da die ebRe-Anwendung aus Sicht des Autors nicht 100 %ig überzeugen kann und z. B. auch nur für Windows-Plattformen erhältlich ist. Im Zuge des Projektes wurde zur Prüfung von ebInterface-Rechnungen das Gedankenexperiment entwickelt, eine Prüfumgebung als Java-Applet zu implementieren. Dieses Applet könnte durch die XSLT-Datei, welche die Rechnungen nach HTML transformiert, zum HTML-Code hinzugefügt und initialisiert werden. Da einige moderne Browser bereits XSLT unterstützen, müsste den ebInterface-Rechnungen dann nur noch der Verweis auf die spezielle XSLT-Datei hinzugefügt werden, damit die Tranformation und Signaturprüfung beim Öffnen der Rechnung im Webbrowser angestoßen werden könnte. In solch einem Szenario wären Anzeige und Signaturprüfung von ebInterface-Rechnungen genauso einfach wie bei PDF-Rechnungen (siehe „4.3.2.3. Verifikation von PDF-Signaturen“ auf S. 120). Nachteilig könnte sich auswirken, dass ein solches Applet ebenfalls digital signiert werden müsste, damit es am System des Benutzers die nötigen Rechte[33] zur Verifikation der Zertifikate erhält. Außerdem könnte solch ein Applet durch die nötigen Kryptographiebibliotheken in der Größe relativ umfangreich werden.

4.3.2. PDF

Das „Portable Document Format“ ist ein sehr weit verbreitetes Dateiformat für Dokumente, das von der Firma Adobe entwickelt wurde. In der InvoiceManager-Anwendung wird es als alternatives Rechnungsformat verwendet. Im Gegensatz zum ebInterface-Standard ist es zur automatisierten Weiterverarbeitung aber nur wenig geeignet. Zwar ist auch dieses Format standardisiert und auch PDF-Dokumente sind in einer gewissen Art und Weise strukturiert, sie verfolgen aber einen ganz anderen Zweck als z. B. XML-

[32] Erhältlich unter `http://portal.wko.at/wk/dok_detail_html.wk?AngID=1&DocID=660371&DstId=7245&StID=316012&SSTId=0`

[33] Zugriff auf KeyStores, Öffnen von Netzwerkverbindungen zur Widerrufsprüfung, ...

Dokumente. PDF-Dokumente sind zur Darstellung für Menschen optimiert. Vorrangiges Ziel ist es, dass ein Dokument in jeder Situation und auf jedem Betriebssystem, auf dem es betrachtet wird, gleich aussieht. Werden Rechnungen im PDF-Format geschrieben, so weiß ein Computer durch die Standardisierung, wie er diese darstellen muss, er kann daraus aber nicht ohne weiteres automatisiert Rechnungsdaten extrahieren und weiterverarbeiten. Durch Konventionen der Art „wir schreiben den Rechnungsbetrag immer an der Stelle x, in Schriftgröße y“ könnte man zwar auch aus PDF-Dokumenten solche Daten gewinnen, dies würde aber wieder einem Standard entsprechen, der erst vereinbart werden müsste. Außerdem wären Daten und Darstellung im selben Format vermischt, was wenig wünschenswert ist.[34] Auf XML basierende Formate wie ebInterface sind zu diesem Zweck deutlich besser geeignet.

Wird aber eine automatisierte Weiterverarbeitung nicht benötigt, so kann das PDF-Format einige Vorzüge ausspielen. Da Programme zur Anzeige von PDF-Dokumenten meist kostenlos verwendet werden können, ist das Format sehr weit verbreitet und fast jeder Computer hat eine solche Anwendung installiert. Dementsprechend sind die Anwender mit dem Format vertraut und es ist anzunehmen, dass sie auch mit Rechnungen in einem solchen Format zurecht kommen würden. Des weiteren ist die Signaturverifikation, wie in „Abschnitt 4.3.2.3 – Verifikation von PDF-Signaturen“ beschrieben, relativ einfach zu bewerkstelligen.

4.3.2.1. Erzeugung der PDF-Rechnungen

PDF-Rechnungen werden vom InvoiceManager in zwei Situationen erzeugt:

- der Kunde wünscht die Zustellung als PDF-Dokument
- der Anwender möchte eine gespeicherte Rechnung als PDF exportieren und diese wurde (noch) nicht als PDF versandt

Die iText-Bibliothek[35] würde es erlauben PDF-Dokumente aus Java-Objekten aufzubauen. Da die Daten aber im vorliegenden Fall in einem XML-Format vorliegen, ermöglicht die Umwandlung per XSLT eine elegantere Lösung. Die Vorgangsweise bei der Transformation ist ähnlich zu der in „Abschnitt 4.3.1.3 – Anzeige der ebInterface-Rechnungen“ gezeigten Vorgangsweise, nur dass die Transformation hier „XML ⇒ XSL-FO“ lautet. „Extensible Stylesheet Language - Formatting Objects“ (XSL-FO) ist eine Art Zwischenformat, von dem dann wiederum in verschiedene Outputformate wie PDF oder PS (PostScript) konvertiert wird. Die vollständige Transformation lautet also: „XML ⇒ XSL-FO ⇒ PDF“. Für die letzte Transformation von XSL-FO in PDF

[34] Vor allem verglichen mit dem eleganten Ansatz, XML mit XSLT in beliebigen Formaten darzustellen („Abschnitt 4.3.1.3 – Anzeige der ebInterface-Rechnungen“).
[35] `http://www.lowagie.com/iText/`

verwendet die InvoiceManager-Anwendung die „Apache FOP-Bibliothek“ (Formatting Objects Processor)[36]. Listing 4.14 zeigt beispielhaft den Code zur Erzeugung von PDF-Dokumenten. Der Parameter `xsltFile` in Zeile 11 spezifiziert jene XSLT-Datei, welche die „XML ⇒ XSL-FO“-Transformation steuert.

```
1  ByteArrayOutputStream out = new ByteArrayOutputStream();
2  org.apache.fop.apps.FopFactory fopFactory = FopFactory.newInstance();
3  org.apache.fop.apps.FOUserAgent foUserAgent = fopFactory.newFOUserAgent();
4  foUserAgent.setAuthor("Wolfgang Schlapschy");
5  foUserAgent.setTitle("Rechnung");
6
7  org.apache.fop.apps.Fop fop = fopFactory.newFop(MimeConstants.MIME_PDF,
       foUserAgent, out);
8
9  // Setup XSLT
10 TransformerFactory factory = TransformerFactory.newInstance();
11 Transformer transformer = factory.newTransformer(new StreamSource(xsltFile
       ));
12
13 // Setup input for XSLT transformation
14 Source src = new StreamSource(new BufferedInputStream(new
       ByteArrayInputStream(xmlInput)));
15
16 // let the generated FO run through Apache FOP
17 Result res = new SAXResult(fop.getDefaultHandler());
18
19 //start XSLT transformation and FOP processing
20 transformer.transform(src, res);
21 byte[] buffer = out.toByteArray();
```

Listing 4.14: Erzeugung von PDF-Rechnungen mit Apache FOP und XSLT

4.3.2.2. Signatur von PDF-Dokumenten

Die bereits erwähnte iText-Bibliothek[35] wurde zwar nicht zur Erzeugung der PDF-Rechnungen verwendet, aber es wurde von der Signaturfunktionalität der Komponente Gebrauch gemacht. Laut [42] handelt es sich bei einer PDF-Signatur um eine in das PDF-Dokument eingebettete CMS-Signatur (Cryptographic Message Syntax). CMS-Signaturen sind von der Grundidee ähnlich zu XML-Signaturen, indem sie nicht nur die eigentliche Signatur, sondern auch sämtliche zur Verifikation benötigte Daten beinhalten. [42] gibt eine sehr gute Einführung in das Thema. Zur PDF-Signatur ist dort unter anderem zu lesen:

[36] `http://xmlgraphics.apache.org/fop/`

> *„Eine Signatur wird in Form eines Verzeichnisses*[37] *(Signature Dictionary) in das PDF-Dokument eingebettet.“*

In diesem Verzeichnis gibt es verschiedene Felder. Das *Contents*-Feld enthält die eigentliche Signatur im CMS-Format. Im *ByteRange*-Feld ist angegeben, auf welchen Teil des PDF-Dokumentes sich die Signatur bezieht. Da die Signatur wie bei einer „enveloped XML-Signature“ im Dokument enthalten ist, muss dieser Bereich ausgespart werden. Im *Type*-Feld wird durch den Wert „Sig“ angezeigt, dass es sich um ein Signature Directory handelt. Im *Filter*- und *SubFilter*-Feld wird festgehalten, mit welchem Signatur-Handler die Signatur erstellt wurde, bzw. um welche Signatur es sich handelt. Diese Informationen werden für eine spätere Signaturverifikation benötigt. [42]

Dem Signatur-Handler kommt dabei besondere Bedeutung zu da er festhält, wie der „Adobe Acrobat Reader“ später die Signatur verifiziert (siehe „4.3.2.3. Verifikation von PDF-Signaturen“ auf S. 120). IText kennt laut [46] drei verschiedene Signatur-Handler:

- *Self signed (Adobe.PPKLite)*
- *VeriSign plug-in (VeriSign.PPKVS)*
- *Windows Certificate Security (Adobe.PPKMS)*

Die InvoiceManager-Anwendung verwendet den *Adobe.PPKMS*-Signatur-Handler, da solche Signaturen z. B. vom „Adobe Acrobat Reader“ einfach verifiziert werden können. Der *VeriSign.PPKVS*-Signatur-Handler funktioniert laut [46] nur mit VeriSign-Zertifikaten und *Adobe.PPKLite* ist laut selbiger Quelle nur für selbstsignierte Zertifikate geeignet.

[57] zeigt anhand einiger Beispiele wie mit iText signiert werden kann. Für die InvoiceManager-Anwendung ist vor allem das Beispiel „An example with an external hash and signature in Windows Certificate Mode“ von Interesse.

IText unterstützt die Erzeugung von sichtbaren und unsichtbaren Signaturen. Unsichtbare Signaturen sind beim Betrachten der Rechnung mit dem Adobe Acrobat Reader nur im „Signatur-Navigationstab“ des Acrobat Readers sichtbar, während sichtbare Signaturen zusätzlich direkt im Dokument angezeigt werden.

Fortgeschrittene *SHA1withRSA*-Signaturen lassen sich mit iText sehr gut erzeugen. Die Verwendung von stärkeren Hashfunktionen wie z. B. *SHA256* ist mit iText zwar grundsätzlich möglich, jedoch ist die Erstellung etwas komplizierter als in den Demonstrationsbeispielen. Außerdem liefert iText die nötigen OIDs nicht mit, und so muss

[37] Anm.: dies entspricht einem besonderen Feld im PDF-Dokument

der Quelltext an einigen Stellen angepasst werden, um solche Signaturen erzeugen zu können [58].

Auch die Erstellung von Signaturen basierend auf ECC, wie z. B. *SHA1withECDSA*, wird von iText nicht unterstützt. Da iText aber OpenSource-Software ist, kann auch diese Funktionalität hinzugefügt werden. Im Projekt wurde im Zuge der experimentiellen Unterstützung für qualifizierte A-Trust-Zertifikate der Quelltext von iText insoweit angepasst, als das für eine Signatur mit dem ECC-Zertifikat erforderlich ist. Diese Änderungen beinhalten hauptsächlich das Hinzufügen der OIDs für die betroffenen Algorithmen. Dabei ging es nur darum, grundsätzlich zu zeigen, dass eine solche Signatur mit iText möglich ist. Da jedoch kein eigener Signatur-Handler definiert wurde, würde dies zu Problemen bei der Verifikation führen (siehe „4.3.2.3. Verifikation von PDF-Signaturen“ auf S. 120).

4.3.2.3. Verifikation von PDF-Signaturen

Der kostenlos erhältliche und weit verbreitete Adobe Acrobat Reader ermöglicht nicht nur die Anzeige von PDF-Dokumenten, sondern auch eine Prüfung von signierten PDF-Dokumenten. Der Acrobat Reader kennt dazu verschiedene Signatur-Handler, mit denen Signaturen und Zertifikate geprüft werden können. Welcher Handler zur Prüfung verwendet werden soll, wird bei der Signatur im *Filter*-Feld angegeben. So können z. B. vom InvoiceManager erzeugte fortgeschrittene *SHA1withRSA*-Signaturen auf Basis eines „A-Cert advanced“-Zertifikates mit dem im Acrobat Reader standardmäßig vorhandenen Adobe.PPKMS-Filter verifiziert werden. Dazu sollte der Acrobat Reader so konfiguriert sein, dass immer zuerst versucht wird, Signaturen mit dem im Dokument angegebenen Handler zu prüfen. „Abschnitt A.7 – Konfiguration des Acrobat Readers zur Verifikation“ behandelt die richtige Konfiguration des Acrobat Readers zur Signaturverifikation.

Root-Zertifikate, welche von den Signatur-Handlern zur Signaturprüfung herangezogen werden, können direkt vom Acrobat Reader verwaltet werden. Auf Windows-Systemen können dazu auch Zertifikate aus dem Windows Zertifikatsspeicher verwendet werden. „Abschnitt A.7 – Konfiguration des Acrobat Readers zur Verifikation“ behandelt die richtige Konfiguration des Readers zur Verifikation.

Erfolgreich geprüfte Signaturen werden im Acrobat Reader durch ein grünes Häkchen visualisiert (siehe Abbildung 4.13), während ungültige Signaturen mit einem roten Kreuz gekennzeichnet werden.

SHA1withECDSA-Signaturen können vom Adobe Acrobat Reader nicht ohne weitere Software verifiziert werden. Zu diesem Zweck können weitere Signatur-Handler als

Abbildung 4.13.: Anzeige einer erfolgreich verifizierten sichtbaren Signatur

Plugins registriert werden. So erzeugt z. B. die „a.sign MultiSign“-Applikation der Firma A-Trust PDF-Signaturen, welche im *Filter*-Feld den Wert `asign.ECDSA` führen. Ein entsprechendes Plugin wird vom „a.sign Client“ im Acrobat Reader installiert und übernimmt die Prüfung von ECC-basierten Signaturen.

Die InvoiceManager-Applikation kann grunsätzlich auch ECC-basierte PDF-Signaturen erzeugen, führt aber im *Filter*-Feld weiter den Wert `Adobe.PPKMS`. Diese Signaturen könnten zwar z. B. mit dem gratis erhältlichen „hotPDFVerify“-Plugin[38] verifiziert werden, dazu müsste aber der Adobe Reader dazu konfiguriert werden, immer diesen Signatur-Handler zu verwenden, egal welcher Handler in der PDF-Rechnung angegeben ist. Dies ist natürlich nicht wünschenswert, da dadurch Signaturen die andere Handler voraussetzen, nicht mehr erfolgreich verifiziert werden könnten. Sollte die experimentelle Unterstützung von qualifizierten A-Trust-Zertifikaten und *ECDSA*-Signaturen in der Praxis eingesetzt werden, müssten hier Lösungen gefunden werden.

4.4. Das e-Billing-System im Detail

Im Folgenden werden kurz einige Details der InvoiceManager-Anwendung vorgestellt, welche bisher in diesem Kapitel noch nicht behandelt wurden, aber wichtig zum Verständnis der Arbeitsweise des Programmes sind.

[38] `http://www.bdc.at/produkte/hotpdfsign.html`

4.4.1. Architektur

4.4.1.1. Systemvoraussetzungen

Die InvoiceManager-Anwendung ist grundsätzlich plattformunabhängig in Java entwickelt worden, wurde aber nur auf dem Betriebssystem Windows XP getestet. Vorausgesetzt wird eine „Java Runtime Environment“ (JRE) in der Version 6 (oder höher). Auch eine Datenbank und deren Konfiguration in der Datei `persistence.xml` wird vorausgesetzt. Um die Rechnungen per Email zu versenden, muss ein SMTP-Server konfiguriert sein (siehe „4.4.2. Zustellung der Rechnungen“ auf S. 134). Um Rechnungen zu signieren, müssen außerdem ein privater Schlüssel und ein dazu gehöriges Zertifikat vorhanden sein, welche in einem Windows-Zertifikatsspeicher oder in einem JKS vorgehalten werden können.[39]

4.4.1.2. Schichten

Bei der InvoiceManager-Anwendung handelt es sich um eine klassische 2-tier-Anwendung, also eine grafische Oberfläche, welche auf eine Datenbank zugreift. Diese Datenbank wird nur von dieser einen Anwendung verwendet. Die Anwendungsschicht ist außerdem in weitere Layer unterteilt, welche sich auch in der Package-Struktur der Anwendung wiederfinden. Abbildung 4.14 zeigt die Aufteilung der Anwendung in Schichten.

persistence Diese Schicht enthält die Zugriffslogik auf die Datenbank, und ist in „Abschnitt 4.4.1.5 – Persistenzschicht und Datenhaltung“ beschrieben.

business Diese Schicht enthält die Domain-Klassen, die Logik zur Erzeugung, zur Konvertierung sowie zum Versand der Rechnungen, und die Logik zur Rechnungssignatur.

gui Diese Schicht implementiert die Benutzeroberfläche nach Vorbild des „Model-View-Presenter“-Musters (MVP-Muster, [68]) und besteht aus folgenden Unterschichten:

model Das Modell verwendet die Schichten *persistence* und *business* um Daten aus der Datenbank zu holen bzw. um diese zu bearbeiten. Die aus der Datenbank geholten Daten werden im Modell in Datenstrukturen gespeichert. Ändert sich das Modell, werden über einen „Observer-Mechanismus“ alle Views zu einem Update aufgefordert.

[39] Die Verwendung des Windows-Zertifikatsspeichers ist nur auf Windows-Betriebssystemen möglich, JKS können auch auf anderen Systemen verwendet werden.

ui Im MVP-Muster sind View und Controller sehr eng verbunden. Die ui-Schicht kümmert sich um die Darstellung der im Modell gespeicherten Daten und verwaltet die Darstellungslogik.

configuration Diese Schicht erlaubt eine Konfiguration der anderen Schichten.

Ganzheitlich betrachtet entspricht die Implementierung der Anwendungsschicht eigentlich einem „Model-Model-View-Presenter-Muster“ (MMVP, [18]), da das Modell nicht sofort auf die Datenbank zugreift, sondern noch weitere Schichten dazwischen geschaltet sind. Durch diese zusätzlichen Schichten wäre es relativ einfach möglich, den Invoice-Manager zu einer verteilten 3-tier-Anwendung zu machen. Dazu könnte z. B. vor die persistence- und business-Schicht eine WebService-Fassade gesetzt werden. Die Modell-Schicht könnte dann so verändert werden, dass sie mit dieser Fassade über WebServices kommuniziert.

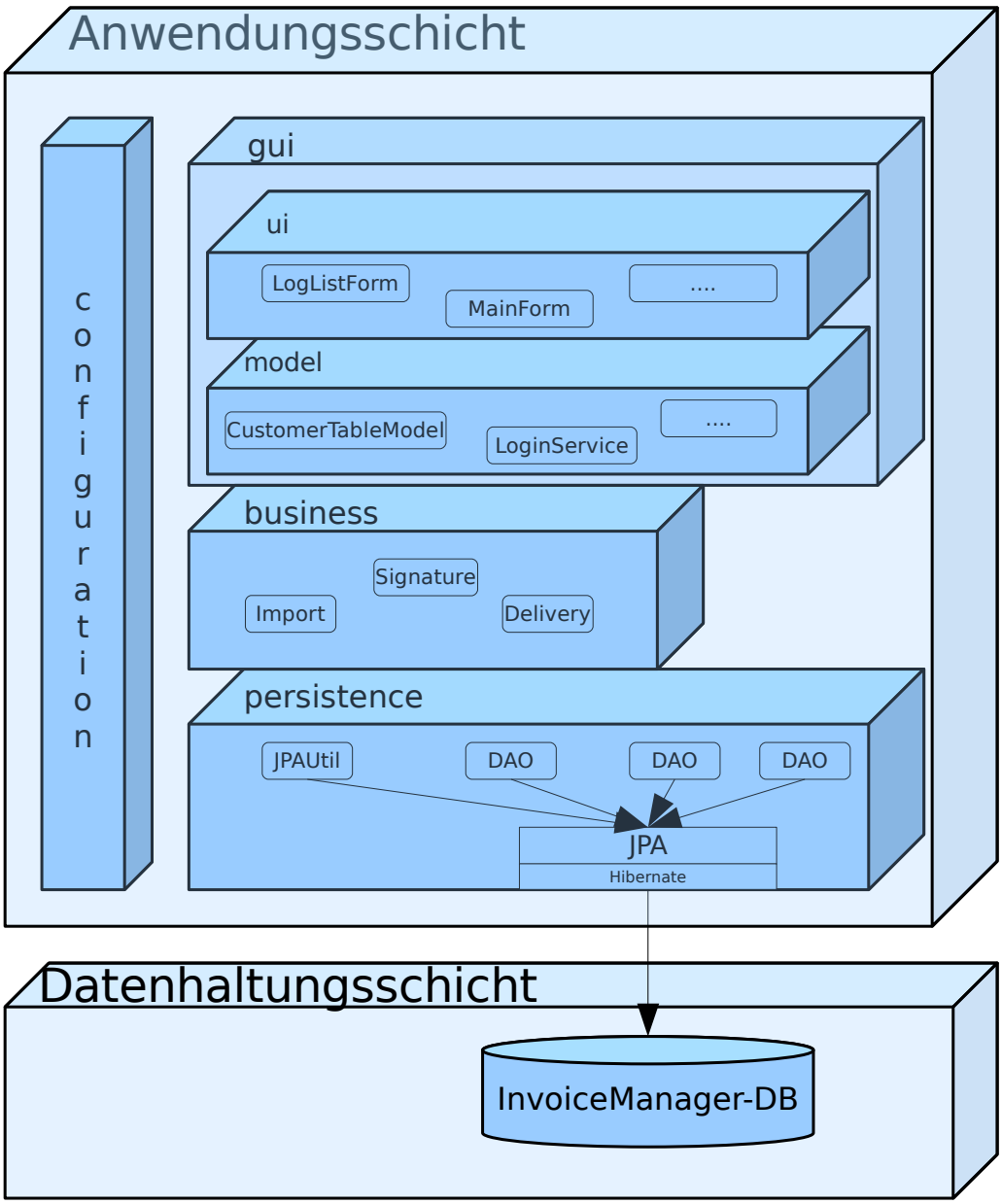

Abbildung 4.14.: Schichtenmodell der InvoiceManager-Anwendung

4.4.1.3. Domain-Modell

Abbildung 4.15 zeigt das Domainmodell der Anwendung[40]. Diese Klassen werden, neben einigen hier nicht erwähnten Hilfsklassen, auch in der Datenbank persistiert. Die zentralen Klassen sind `at.ws.business.entities.Customer` und `at.ws.business.Invoice`. Jedes `Customer`-Objekt hält eine Liste von `ChannelSpecificDeliveryBehaviours`. Diese Liste wird beim Versenden einer Rechnung des Kunden durchgegangen, und es wird für jedes Objekt in der Liste seine `deliver`-Methode aufgerufen. Jedes Kundenobjekt hat außerdem auch ein `FormatSpecificDeliveryBehaviour` definiert, welches das ausgewählte Format darstellt. Das Logging der Rechnungsbearbeitung stellt in dieser Anwendung kein sogenanntes „Crosscutting Concern“ dar, sondern ist als zentrale Anforderung auch Teil des Domain Modelles. Es ist damit auch getrennt vom übrigen Logging der Anwendung, welches z. B. zur Fehleraufzeichnung durchgeführt wird.

[40] Die Darstellung der Methoden wurde der besseren Übersichtlichkeit geopfert

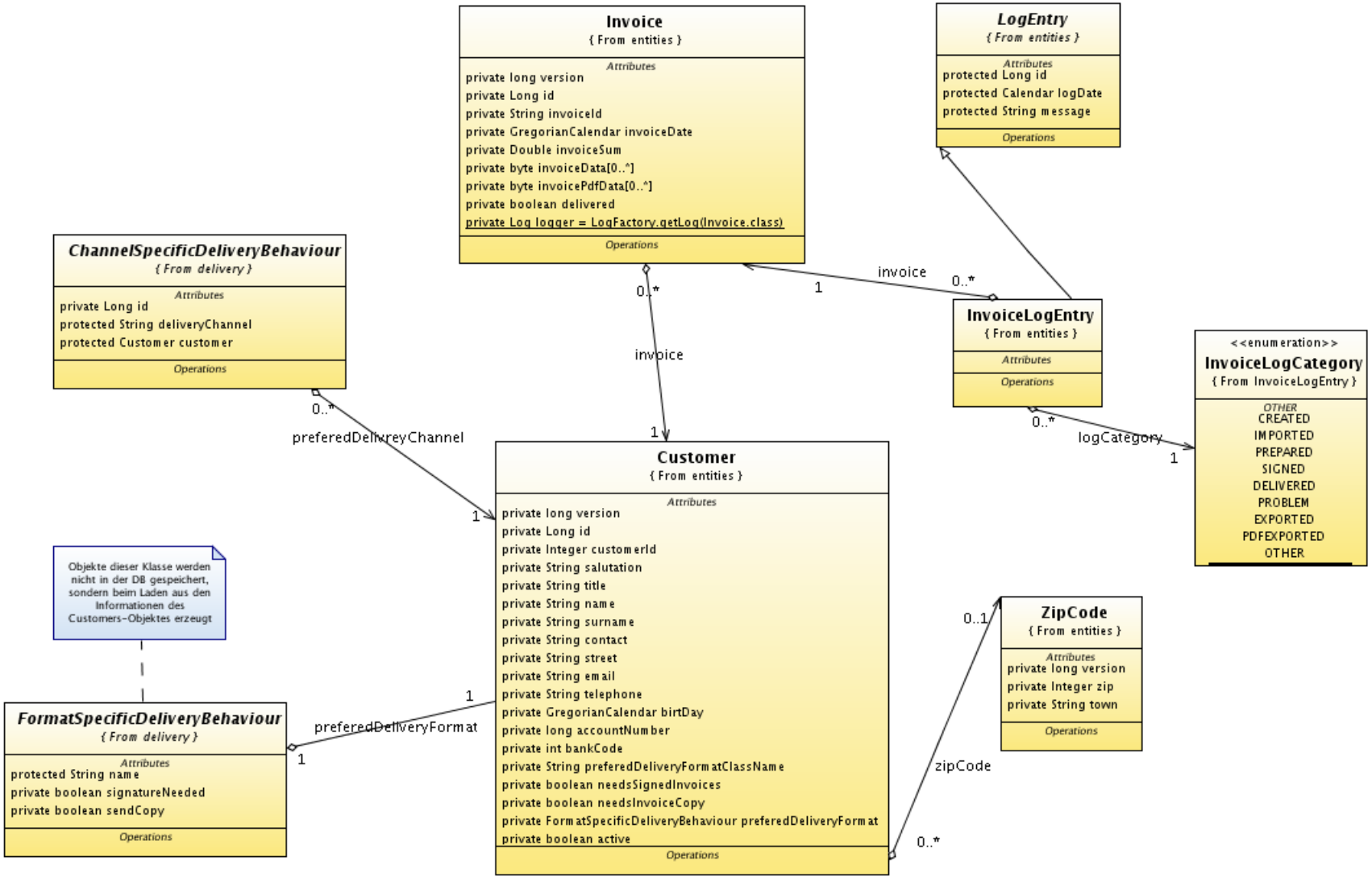

Abbildung 4.15.: Domainmodell der InvoiceManager-Anwendung

4.4.1.4. GUI

Die Benutzeroberfläche wurde mit der Swing-Bibliothek erstellt und umfasst die folgenden Ansichten:

Login Wird das Programm gestartet, bekommt der Benutzer zuerst einen Login-Bildschirm präsentiert, wo er Benutzername und Passwort eingeben muss. Diese entsprechen den Authentifizierungsdaten für die Verbindung zur Datenbank. Sind die Daten korrekt, wird die Verbindung zur Datenbank aufgebaut und das Hauptfenster geöffnet. Kann mit den eingegebenen Daten keine Verbindung zur Datenbank aufgebaut werden, erhält der Benutzer eine entsprechende Meldung.

Kunde anlegen Dieses Fenster bietet eine Maske zur Eingabe der Kundendaten. Hier kann außerdem für jeden Kunden ausgewählt werden, ob er seine Rechnungen im ebInterface-Format oder als PDF-Rechnung erhalten möchte. Für das jeweilige Format kann hier weiters festgelegt werden, ob die Rechnungen signiert werden sollen. Ebenfalls ist es möglich anzugeben, ob der Kunde eine Rechnungskopie im jeweils anderen Format erhalten soll.[41] Außerdem muss hier angegeben werden, auf welche Art der Kunde seine Rechnung erhalten möchte. Wird keine Versandart ausgewählt, bleiben die Rechnungen des Kunden immer im Status „unerledigt“. Sollen Rechnungen nicht versandt werden, aber trotzdem als „erledigt“ markiert werden, sollte die Versandart „No Delivery“ ausgewählt werden.

Kunde verwalten In diesem Fenster können die im Fenster „Kunde anlegen“ eingegebenen Daten verwaltet werden. Es ist vom Aufbau dem Fenster zur Kundenanlage ähnlich, bietet aber auch die Funktion einen Kunden aus der Datenbank zu löschen. Abbildung 4.16 zeigt das Fenster „Kunden verwalten“.

Rechnungen importieren Hier kann der zur Importdatei passende Importfilter ausgewählt, und der Pfad zur Importdatei angegeben werden. Ein Klick auf den Button „importieren“ startet den Importvorgang. Der Button „Details“ zeigt detaillierte Informationen über den Status des Importvorganges. Abbildung 4.17 zeigt das Fenster „Rechnungen importieren“.

Kunden auflisten Dieses Fenster zeigt die gespeicherten Kunden in einer tabellarischen Ansicht und bietet ein Feld, welches die (sortierbare) Tabelle nach Kundennamen filtert. Wird ein Kunde in der Tabelle ausgewählt, können über das Kontextmenü die Fenster „Kunde verwalten“ und „Rechnungen des Kunden anzeigen“ erreicht werden.

[41] Die Wahl des zu signierenden Formates wurde auf diese Weise gelöst, um zu verhindern, dass versehentlich beide Formate signiert werden (siehe „2.1.4. Aufbewahrung der Rechnungen“ auf S. 25).

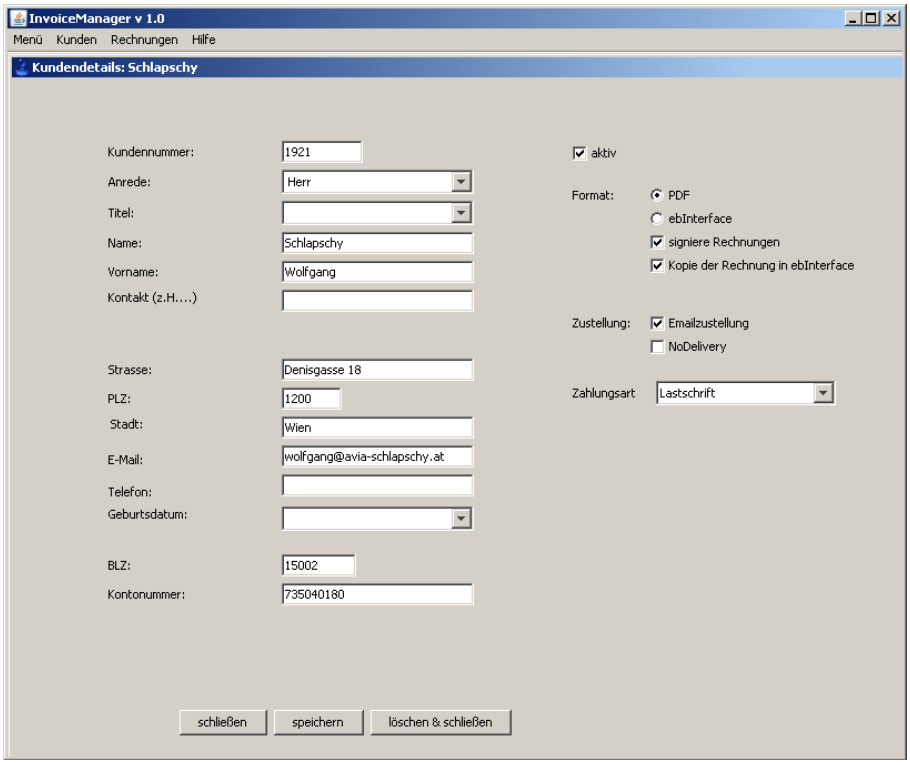

Abbildung 4.16.: Fenster „Kunden verwalten“

Rechnungen des Kunden anzeigen Dieses Fenster zeigt alle gespeicherten Rechnungen eines Kunden in einer (sortierbaren) Tabelle. Wird eine Rechnung ausgewählt, kann über das Kontextmenü das Fenster „Rechnungslog einsehen“ erreicht werden. Außerdem können über das Kontextmenü die Aktionen „Rechnung exportieren“[42], „Rechnung als PDF exportieren“ und „Rechnung anzeigen“[43] angestoßen werden. Bei einem Doppelklick auf eine Rechnung wird diese ebenfalls im Browser angezeigt. Abbildung 4.18 zeigt eine Ausprägung dieses Fensters, wobei für diesen Kunden nur eine einzige Rechnung gespeichert ist.

Rechnungslog einsehen Für jede Rechnung wird ab dem Zeitpunkt des Imports aufgezeichnet, in welchen Zuständen sie sich befindet.

Unerledigte Rechnungen auflisten Dieses Fenster zeigt eine Liste von allen importierten Rechnungen, die noch nicht versandt wurden. Rechnungen können per Mausklick selektiert werden. Wird dabei die „Strg-Taste“ gedrückt gehalten, können mehrere Rechnungen selektiert werden. Über das Kontextmenü kann eine Selektion wieder aufgehoben werden. Außerdem ist per Kontextmenü auch die Anzeige der Rechnung im Webbrowser möglich. Ein Klick auf „Rechnungen versenden“ öffnet den Dialog „Rechnungen zustellen“, wo die Rechnungen versandt werden können. Wurden in der Tabelle keine Rechnungen selektiert, so wird ver-

[42] Exportiert die Rechnung im XML-Format
[43] Zeigt die Rechnung im Webbrowser an

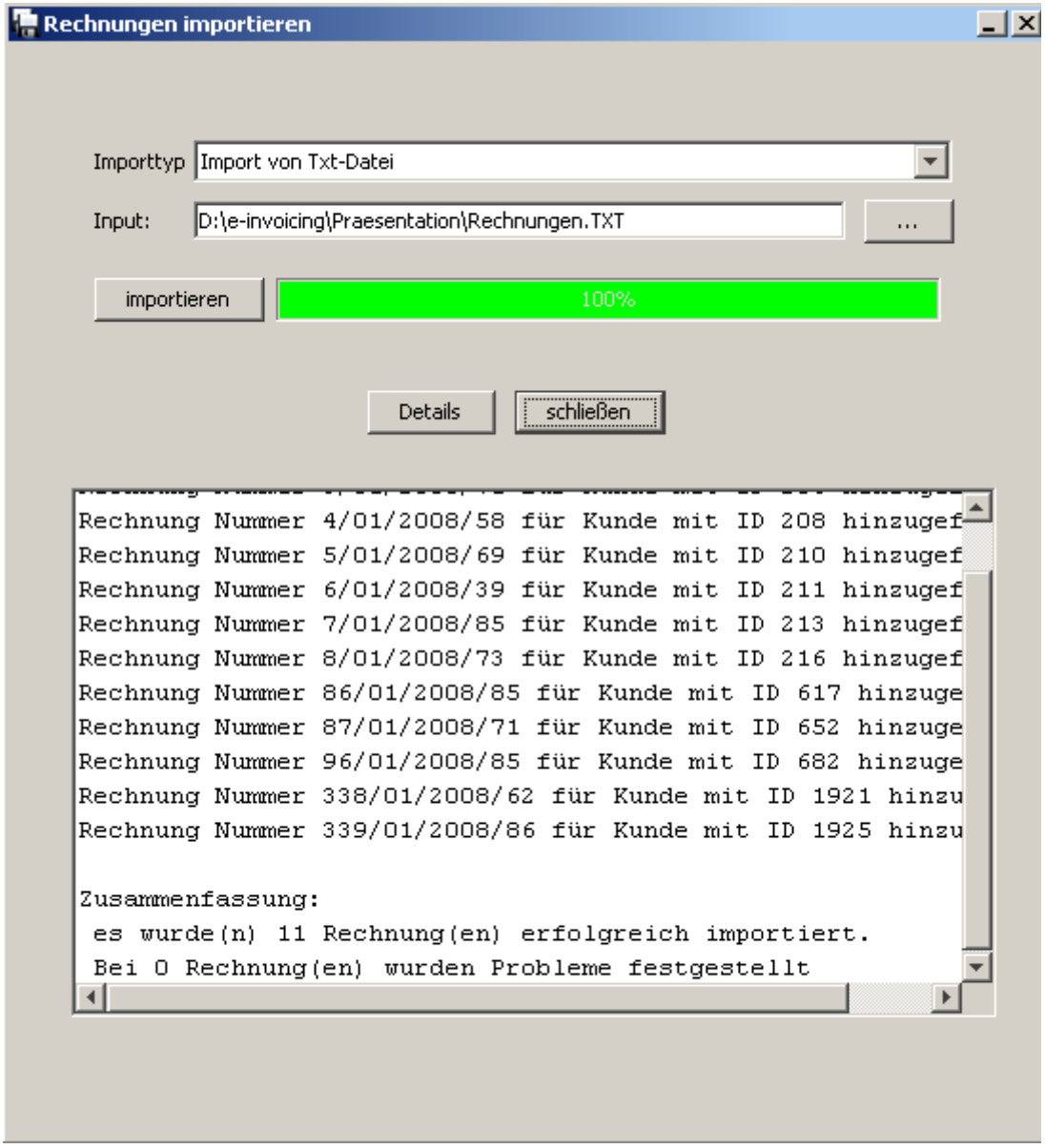

Abbildung 4.17.: Fenster „Rechnungsimport“

sucht, alle Rechnungen die sich im Zustand „unerledigt“ befinden und für deren Empfänger eine Versandart ausgewählt wurde, zu versenden. Wurden Rechnungen selektiert, so wird nur versucht, diese Rechnungen zuzustellen.

Rechnungen zustellen Diese, in Abbildung 4.19 gezeigte, Ansicht erlaubt den Versand der Rechnungen. Der Button „Einstellungen“ macht einen Bereich sichtbar, in dem verschiedene Einstellungen für den Versand getätigt werden können:

Email-Einstellungen In diesem Bereich kann der Emailversand der Rechnungen konfiguriert werden. Dieser ist in „Abschnitt 4.4.2 – Zustellung der Rechnungen“ beschrieben.

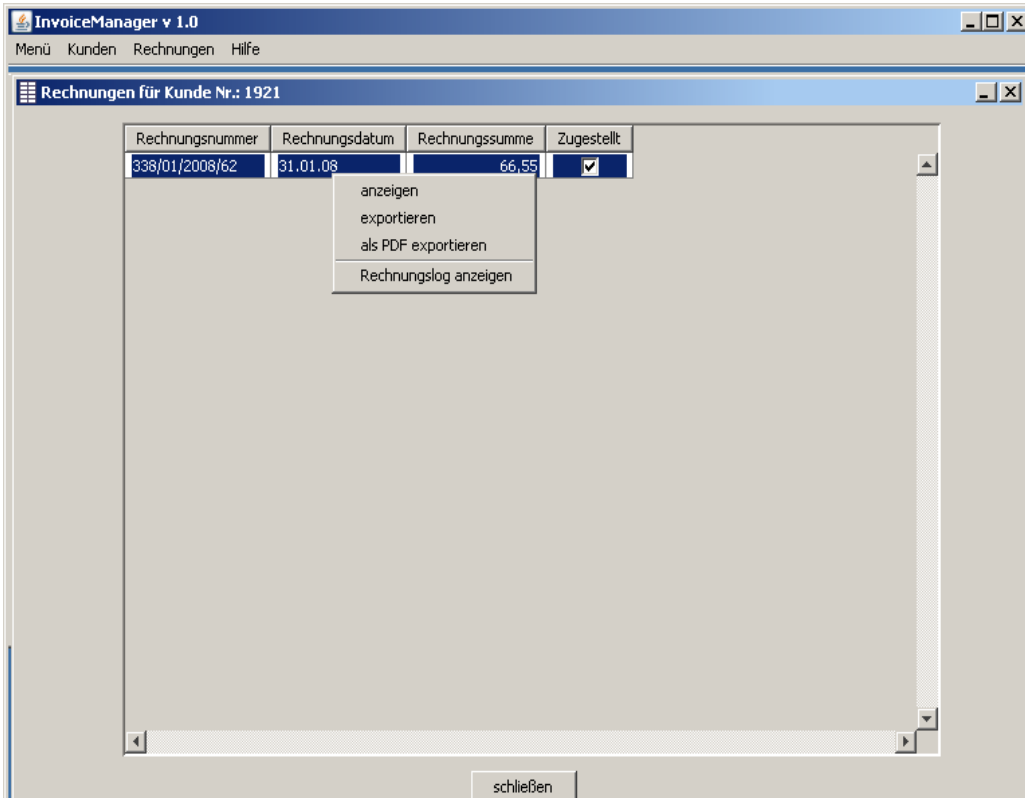

Abbildung 4.18.: Fenster „Rechnungen des Kunden anzeigen“

Signatur-Einstellungen Hier kann der Rechnungsversender den zur Signatur zu verwendenden Schlüssel auswählen (siehe „4.2.6. „Key Stores“ “ auf S. 96). Außerdem kann das zum Schlüssel gehörende Zertifikat angezeigt werden (siehe „4.2.5. Zertifikate“ auf S. 88) und die zu verwendende Hashfunktion ausgewählte werden.

Ein Klick auf den Button „versenden“ startet den Rechnungsversand.

Die Fenster „Kunden anlegen“, „Rechnungen importieren“, „Kunden auflisten“ und „Unerledigte Rechnungen auflisten“ können über das Hauptmenü der Anwendung aufgerufen werden. Abbildung 4.19 zeigt das Fenster mit geöffnetem Konfigurationsbereich.

Bei der Anwendung handelt es sich um eine Art „MDI-Anwendung“ (Multiple Document Interface). Das heißt es gibt ein übergeordnetes Fenster, in welchem mehrere untergeordnete Fenster gleichzeitig geöffnet werden können. Es können also z. B. die Kundenliste und das Importfenster gleichzeitig geöffnet sein. Eine Implementierung des „Observer-Musters“ aktualisiert die Fenster bei Änderungen.

4.4.1.5. Persistenzschicht und Datenhaltung

Die „Domain-Objekte“ (siehe „4.4.1.3. Domain-Modell“ auf S. 125) der Anwendung werden in einer relationalen Datenbank gespeichert. Zur Verbindung zwischen der Programmiersprache Java und dem DBMS wird „JPA“ (Java Persistence API) verwendet. Bei JPA handelt es sich um einen Standard für „objekt-relationales Mapping“. JPA ist ein Teilbereich der „EJB3-Spezifikation“, welche wiederum einen Bestandteil der „Java Enterprise Edition“ darstellt [18]. Das bekannteste OR-Mapping-Tool ist sicherlich „Hibernate“[44], was laut [18] auch Einfluss auf die Spezifizierung von JPA hatte. Aufgrund der Verwendung von JPA ergeben sich folgende Vorteile:

Verminderung der Diskrepanz zwischen OOP und RDBMS Dazu muss der Entwickler spezifizieren, wie die Domain-Objekte aus Java auf die relationale Struktur der Datenbank abgebildet werden sollen. [16] beschreibt dies ausführlich. In Hibernate wurden dazu spezielle Mapping-XML-Dateien verwendet. JPA erlaubt die Angabe des Mappings als Java-Annotation direkt in der Java-Klasse des jeweiligen Domain-Objektes. Dies ist aus der Sicht des Autors weniger fehleranfällig und reduziert die Anzahl der zu verwaltenden Artefakte.

Keine Abhängigkeit von der Datenbank JPA wie auch Hibernate erlauben ein von Datenbankprodukten weitgehend unabhängiges Speichern von Objekten. Dazu müssen nur in einer Konfigurationsdatei[45] der „Dialekt“ der Datenbank und die Verbindungsdaten zu dieser eingetragen werden. Programmiert wird immer gegen die selben APIs und JPA bzw. Hibernate kümmern sich um die Umsetzung für die jeweils konfigurierte Datenbank. Für Datenbankabfragen wird standardmäßig nicht SQL, sondern die dazu ähnliche „JPA Query Language“ verwendet. Darüber hinaus können einzelne Objekte durch spezielle generische Methoden, wie z. B. `public <T> T find(Class<T> entityClass, Object primaryKey)` der Klasse `javax.persistence.EntityManager` auch implizit, d. h. ohne Angabe einer Abfrage, geladen werden.

Unabhängigkeit von einem speziellen „Persistence Provider“ JPA spezifiziert zwar das O/R-Mapping für JEE, implementiert dies aber nicht selbst. Dies übernehmen spezielle „Persistence-Provider“. Hier schließt sich der Kreis, und so kann Hibernate mit den Erweiterungen „Hibernate Annotations“ und „Hibernate EntityManager“ als „Persistence-Provider“ eingesetzt werden [18]. Diese Rolle könnte aber ebenso von anderen Produkten wie z. B. „Toplink“[46] übernommen werden. JPA bietet also nicht nur Unabhängigkeit von der Datenbank, sondern auch Unabhängigkeit vom jeweiligen Persistence Provider. In der Praxis ist diese Eigen-

[44] `http://www.hibernate.org/`
[45] Bei JPA ist dies die Datei `persistence.xml`
[46] `http://www.oracle.com/technology/products/ias/toplink/index.html`

schaft allerdings nicht so wichtig wie die beiden vorher genannten, da JPA noch nicht alle notwendigen Feinheiten definiert, die für eine praktische Nutzung nötig sind. Hier müssen dann Erweiterungen der Persistence Provider verwendet werden, die nicht portabel sind. Weiters wird es wohl auch nur selten nötig sein, den Persistence Provider auszutauschen.

Die Persistenzschicht ist in der InvoiceManager-Anwendung durch das in [16] beschriebene GenericDAO-Muster implementiert. Dies hat den Vorteil, dass der JPA-Zugriffscode an einer Stelle gekapselt wird. Sollte es z. B nötig werden, zur Abfrage der Datenbank auch natives SQL anstatt der JPA Query Language zu verwenden, so bleibt der Zugriff für das Model[47] trotzdem gleich.

Da JPA auf Hibernate basiert, stellen sich bei der Entwicklung auch die selben Herausforderungen betreffend des Öffnens und Schließens der „Hibernate-Session“. Der Zugriff auf die Datenbank passiert in Hibernate immer innerhalb einer solchen Session[48]. Sollen bei der InvoiceManager-Applikation Kundenobjekte in einer Tabelle angezeigt werden, wird von Hibernate eine Session geöffnet. In dieser werden die Daten aus der Datenbank geladen und daraus die Kundenobjekte erstellt. Ein Kundenobjekt besitzt eine Liste mit den Rechnungen des Kunden, welche ebenfalls in der Datenbank gespeichert sind. Nun stellt sich natürlich die Frage, ob beim Laden der Kunden auch die Rechnungen mitgeladen werden. Dieses Verhalten ist konfigurierbar. Aus Performanzgründen wird man die Rechnungen meist aber erst dann nachladen wollen, wenn im Programm auf diese zugegriffen werden soll. Dieses „Lazy Loading“ genannte Nachladen ist aber nur möglich, so lange die Session noch nicht geschlossen ist, ansonsten erhält man eine „LazyInitializationException“. [18] und [16] beschreiben dieses Problem im Detail. Der optimale Zeitpunkt um Sessions zu öffnen und zu schließen ist eine zentrale Frage bei der Verwendung von JPA und Hibernate. Ein Eintrag in der „Community Area“ der Hibernate-Webseite beschreibt die Verwendung von Hibernate in Desktopanwendungen und zeigt die rege Diskussion zu diesem Thema [40]. Das für Webanwendungen meist vorgeschlagene „Open Session in View“-Muster ist für Desktopapplikationen nicht anwendbar, da diese nicht dem Request/Response-Zyklus von Webanwendungen folgen. Anstatt dessen holen sich Desktopanwendungen meist zuerst einige wenige Domain-Objekte, von welchen ausgehend dann auf dem Domain-Modell „gesurft“ wird [44]. Oft wird für Desktopanwendungen auch ein „Session per Application“-Muster vorgeschlagen. Dabei wird eine einzige Session für die Laufzeit einer Applikation geöffnet. Dies könnte aus der Sicht des Autors Probleme im Fehlerfall mit sich bringen[49] und müsste außerdem bei einer Architekturumstellung geändert werden. Für die InvoiceManager-Anwendung wurde deshalb beschlossen, dass eine Operation (oder mehrere zeitlich zusammengehörende Operationen) in einer Session ausgeführt wird und dass diese Session dann

[47] Anm.: aus MVP (Model-View-Presenter)
[48] In JPA wird die Session eigentlich „Entity Manager“ genannt. Der Begriff einer Session ist nach Meinung des Autors aber besser zur Schilderung des Problemes geeignet, und wird hier auch verwendet.
[49] Da empfohlen wird, bei Auftreten eines Hibernatefehlers die Session nicht weiter zu verwenden.

geschlossen wird. Öffnen und Schließen einer Session wird von der Model-Komponente erledigt, welche das jeweilige Objekt bereits fertig initialisieren muss, um das Auftreten von `LazyInitializationExceptions` zu verhindern.[50]

Zentrale Klasse im Domain-Modell ist die Klasse `at.ws.business.entities.Invoice`. Diese Klasse enthält auch die ebInterface-Version und die PDF-Version der Rechnung. PDF-Rechnungen werden im Objekt in einem byte-Array abgelegt und landen in der Datenbank in einem *BLOB* (Binary Large Object). Zum Speichern von XML-Daten in einer Datenbank gibt es laut [19] folgende Möglichkeiten:

Clobbing Dabei wird das XML-Dokument wie normaler Text z. B. als *CLOB* gespeichert. Um den Zugriff auf die im Dokument gespeicherten Informationen zu vereinfachen und zu beschleunigen, können bestimmte Daten aus dem Dokument zusätzlich in eigenen Feldern einer Datenbanktabelle abgelegt werden. Diese Redundanz ist durchaus gewollt, könnte aber ausarten, wenn z. B. Dokumente in SQL-Abfragen häufig nach verschiedenen Gesichtspunkten durchsucht werden müssen. Werden Daten aus dem XML-Dokument benötigt, für die es keine redundante Speicherung gibt, muss dieses zuerst durch einen XML-Interpreter aus der Textdarstellung in der Datenbank rekonstruiert werden.

Shredding Hier werden nicht nur gewisse Daten aus dem XML-Dokument als eigene Felder gespeichert, sondern das ganze Dokument wird in seine Teile zerlegt und jeder Teil wird in einem Feld der relationalen Datenbank gespeichert. Hibernate bietet Möglichkeiten, ein relationales Datenbankschema auf XML anstatt auf Java-Objekte zu mappen. Wird Shredding verwendet, kann sehr einfach durch SQL-Aufrufe mit den Daten gearbeitet werden. Allerdings ist damit zu rechnen, dass die Datenbankschemen bei komplexen XML-Dokumenten sehr schnell unübersichtlich werden können. Noch größere Probleme könnte eine Modifikation des XML-Schemas mit sich bringen.

Hierachische Speicherung Native XML-Datenbanken oder hybride Datenbanken erlauben eine hierarchische Speicherung der XML-Daten in der Datebank. In solchen Datenbanken gibt es ein spezielles Feld für XML-Dokumente, und XML-Daten können innerhalb von SQL-Abfragen mit XQuery abgefragt werden.

In der InvoiceManager-Anwendung wird zur Speicherung der ebInterface-Rechnungen nach dem Clobbing-Prinzip vorgegangen. Dazu wird das byte-Array `invoiceData` aus der `Invoice`-Klasse in der Datenbank in einem *BLOB* gespeichert. Wünschenswert wäre es aber, z. B. mit XQuery Teile der XML-Dokumente abfragen zu können. Als mögli-

[50] [44] beschreibt zum Lösen solcher Aufgaben ein generisches „Preload"-Muster. Auch in der InvoiceManager-Anwendung werden Objekte vorinitialisiert. Dies geschieht aber im Gegensatz zu diesem Muster nicht generisch in den DAOs, sondern einfach durch Aufruf der nötigen get-Methoden im Model.

cher Anwendungsfall kann hier die Berechnung des Jahresrabattes aus den gespeicherten ebInterface-Rechnungen dienen. Das DBMS-Produkt „PostgreSQL“[51] bietet eine Unterstützung von XQuery, wobei die Dokumente trotzdem in einem herkömmlichen *TEXT*-Feld abgelegt werden [36]. Es sollte aber nicht vergessen werden, dass die Verwendung DBMS-spezifischer Eigenschaften durch Hibernate und JPA zwar durchaus möglich ist, die Anwendung aber dann auch an dieses DBMS-Produkt gebunden ist.

4.4.2. Zustellung der Rechnungen

Pro Kunde kann ausgewählt werden, wie er seine Rechnungen erhalten möchte. Derzeit ist neben der „No Delivery“-Zustellung nur eine Zustellung per Email implementiert. Dabei erhält jeder Rechnungsempfänger, für den eine Emailadresse vermerkt wurde und für den diese Versandart ausgewählt wurde, eine Email, welche im Anhang die Rechnung im gewählten Format[52] enthält. Das Fenster „Rechnungen zustellen“ erlaubt die Konfiguration des Emailversandes im Bereich `Einstellungen - Email-Einstellungen`. Dort können in zwei Textfeldern der Betreff sowie der Text für die Emails angegeben werden. Beide Textfelder erlauben über das Kontextmenü ein Hinzufügen von Platzhaltern. Diese Platzhalter werden beim Versand mit den Werten der jeweiligen Rechnung bzw. mit den Werten des Empfängers ersetzt.[53] Platzhalter existieren für die Anrede, die Rechnungsnummer, das Rechnungsdatum, sowie den Rechnungsbetrag. Anhang A.8 zeigt eine Übersicht über diese Platzhalter und ihre Verwendung. Im unteren Teil dieses Konfigurationsbereiches kann der Rechnungssteller beliebige Dateien hinzufügen, welche dann ebenfalls jeder Email als Anhang beigefügt werden. Auf diese Weise könnten in den Rechnungs-Emails, neben den Rechnungen selbst, z. B. auch Werbematerial versandt werden. Diese Funktion wurde beim Versand der Rechnungen des Monats April 2008 auch dazu verwendet, um den Kunden eine Anleitung zur Signaturprüfung zukommen zu lassen (siehe „5. Fazit“ auf S. 139). Abbildung 4.19 zeigt das Fenster „Rechnungen zustellen“ und die Konfiguration des Emailversands. Abbildung 4.20 zeigt eine mit den Einstellungen aus Abbildung 4.19 versandte und vom Kunden empfangene Email im Programm „Outlook Express“.

Zur Implementierung des Rechnungsversands wurde das JavaMail-API verwendet. Um Rechnungen per Email versenden zu können, muss in der Konfigurationsdatei der Anwendung ein SMTP-Server konfiguriert sein (siehe „4.4.3. Konfiguration“ auf S. 136), welcher die Rechnungen übernimmt und an die Empfänger zustellt.

Durch die modulare Entwicklung kann die InvoiceManager-Applikation einfach um neue Versandarten erweitert werden. Eine Versandart muss die abstrakte Klasse

[51] `http://www.postgresql.org/`
[52] Und eventuell die Rechnungskopie im jeweils anderen Format
[53] Z. B. wird der Platzhalter „%invoiceDate“ beim Versand mit der Rechnungsnummer der jeweiligen Rechnung ersetzt

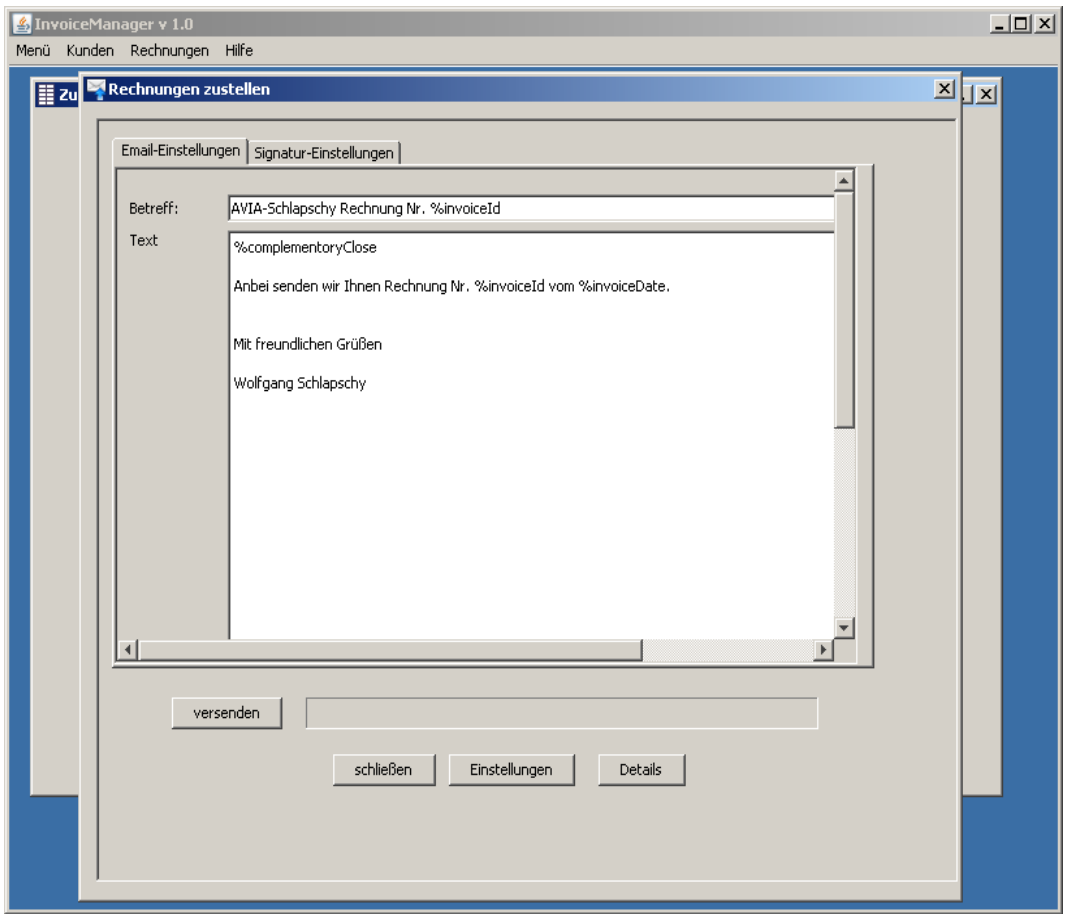

Abbildung 4.19.: Fenster „Rechnungen zustellen“

`at.ws.business.entities.delivery.ChannelSpecificDeliveryBehaviour` erweitern und deshalb auch ein JPA-Entity darstellen (siehe „4.4.1.5. Persistenzschicht und Datenhaltung“ auf S. 131). Weiters muss die Versandart in der Konfigurationsdatei der Anwendung angeführt werden. Einzig falls eine GUI-unterstützte Konfiguration im Dialog „Rechnungen versenden“ ermöglicht werden soll, müssen größere Teile der Anwendung geändert werden.

Grundsätzlich sind neben dem Emailversand noch einige weitere Versandarten denkbar, wie z. B.:

- Ablage im Kundenbereich einer Webseite
- Übergabe an einen e-Billing-Diensteanbieter
- Übergabe an Finanzonline
- Ausdruck der Rechnungen

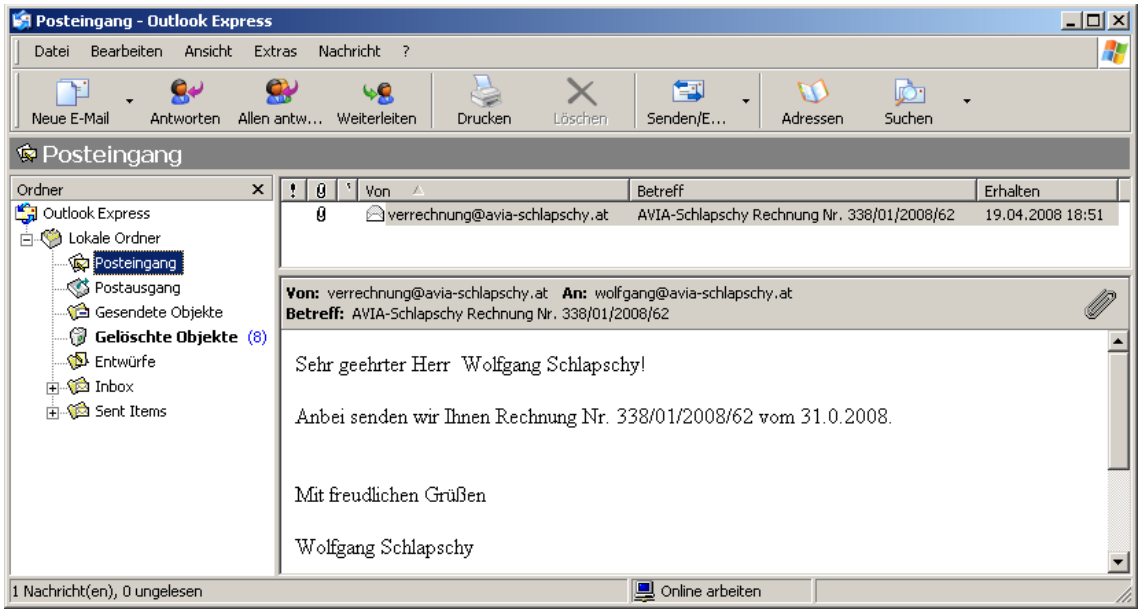

Abbildung 4.20.: Empfang einer Rechnung per Email

4.4.3. Konfiguration

Die InvoiceManager-Anwendung lässt die Konfiguration einiger Parameter über eine XML-Konfigurationsdatei zu. Das Einlesen dieser Datei erfolgt ähnlich wie die Erzeugung der ebInterface-Rechnungen mit JAXB (siehe „4.3.1.2. Erzeugung der ebInterface-Rechnungen“ auf S. 110). Im Gegensatz zur Rechnungserzeugung werden hier aber keine XML-Dateien erzeugt, sondern vorhandene XML-Daten in Java-Objekte umgewandelt.

Konfigurierbar sind unter anderem folgende Parameter:

- Versandarten
- Titel der Applikation
- Daten des Rechnungsstellers (Name, Kontaktdaten, Adresse, UID-Nummer, ...)
- Serverdaten für den Emailversand (SMTP-Host, SMTP-Benutzer, SMTP-Passwort, Adresse des Absenders)
- Standardtext und -betreff sowie Anreden für den Emailversand
- Texte für verschiedene Bestandteile der ebInterface-Rechnung, wie z. B. `directDebitPaymentComment`, `headerComment`, ... Auch hier können analog zum Emailversand Platzhalter vergeben werden, die bei der Rechnungserzeugung mit

den jeweiligen Werten ersetzt werden (z. B. „%bankCode“, „%month“, ...). Anhang A.8 zeigt eine Übersicht über diese Platzhalter und ihre Verwendung.

- Zahlungsziel für die importierten Rechnungen (bei den derzeit importierten Rechnungen ist das Zahlungsziel immer auf 0 Tage gesetzt).

- ...

Als eine Schwachstelle der derzeitigen Implementierung könnte es gesehen werden, dass in der Konfigurationsdatei teilweise sensible Daten wie Passwörter gespeichert werden. Um dieses Problem etwas zu entschärfen, könnte die Konfigurationsdatei verschlüsselt abgelegt werden. Die Datei müsste dann vor dem Einlesen der Konfiguration durch den InvoiceManager entschlüsselt werden. Zur Durchführung von Konfigurationsänderungen müsste es außerdem eine Möglichkeit geben, die Datei außerhalb der Anwendung zu ver- und entschlüsseln. Zur praktischen Durchführung einer solchen Verschlüsselung würde sich eine PBE-Verschlüsselung (Password-Based Encryption), wie in [41] beschrieben, anbieten. Als Passwort zur Schlüsselgenerierung könnte hier eventuell das Loginpasswort verwendet werden, sodass beim Start weiterhin nur eine einzige Passworteingabe erforderlich wäre.

„Abschnitt A.9 – Beispiel: Konfigurationsdateien der InvoiceManager-Anwendung“ zeigt ein Beispiel für die Konfiguration durch die `config.xml`-Datei.

Ganzheitlich betrachtet kann die InvoiceManager-Anwendung durch folgende Dateien konfiguriert werden:

config.xml die in diesem Abschnitt beschriebene Konfigurationsdatei

persistence.xml zur Konfiguration der Datenbankverbindung (siehe „4.4.1.5. Persistenzschicht und Datenhaltung“ auf S. 131)

invoice2html.xslt steuert die Transformation von ebInterface-Rechnungen in das HTML-Format zur Anzeige im Browser (siehe „4.3.1.3. Anzeige der ebInterface-Rechnungen“ auf S. 112)

invoice2fo.xslt steuert die Transformation von ebInterface-Rechnungen in das XSL-FO-Format zur Umwandlung in PDF-Rechnungen (siehe „4.3.2.1. Erzeugung der PDF-Rechnungen“ auf S. 117)

4.4.4. Erweiterungsmöglichkeiten

Abhängig von den zukünftigen rechtlichen Entwicklungen und organisatorischen Anforderungen sind folgende Erweiterungsmöglichkeiten für die InvoiceManager-Anwendung denkbar:

- Ausbau der Unterstützung qualifizierter Zertifikate (siehe „Abschnitt 4.3.2.2 – Signatur von PDF-Dokumenten“ und „Abschnitt 4.2.6.3 – Smartcards“)
- Entwicklung einer Anwendung zum einfachen Prüfen von ebInterface-Signaturen (nach dem in „Abschnitt 4.3.1.5 – Verifikation von ebInterface-Rechnungen“ beschriebenen Ansatz)
- Hinzufügen weiterer Versandarten (konkret geplant ist die Erweiterung der Anwendung um einen Rechnungsdruck)
- Umstellung der Speicherung der XML-Dokumente in der Datenbank (siehe „4.4.1.5. Persistenzschicht und Datenhaltung“ auf S. 131)
- Integration einer GUI zum manuellen Schreiben von Einzelrechnungen

5 Fazit

Im Moment befindet sich die InvoiceManager-Anwendung im betrachteten Unternehmen in einer Testphase, welche im Jänner 2008 begonnen wurde. Tabelle 5.1 zeigt die in der Testphase versandten Rechnungen. In der Testphase werden PDF-Rechnungen versendet, um die Rechnungsempfänger mit einem bekannten Format an den Erhalt elektronischer Rechnungen zu gewöhnen. Ist die PDF-Rechnung erst einmal gut eingeführt, können die Kunden auf die Möglichkeit hingewiesen werden, Rechnungen in einem weiterverarbeitbaren Format zu erhalten. Hier sollten dann vor allem die in dieser Arbeit oft erwähnten Vorteile eines solchen Formates kommuniziert werden. EbInterface-Rechnungen können den Kunden durch die InvoiceManager-Anwendung entweder als signierte Originalrechnung zugestellt werden, oder aber als Zusatzinformation zur signierten PDF-Rechnung. Die InvoiceManager-Anwendung wurde noch nicht an den be-

Monat	**Anzahl Rechnungen**	**Format**	**Signatur**
Jänner 2008	80 Rechnungen	PDF	nein
Februar 2008	108 Rechnungen	PDF	nein
März 2008	131 Rechnungen	PDF	nein
April 2008	143 Rechnungen	PDF	ja
Mai 2008[a]	150 Rechnungen	PDF	ja

[a] voraussichtlich

Tabelle 5.1.: In der Testphase versandte Rechnungen

trachteten Betrieb übergeben und wird in der Testphase vom Autor betrieben. Dabei werden noch kleinere Fehler beseitigt und die Anwendung wird noch an einigen Stellen angepasst und erweitert. Als Beispiel kann hier die Implementierung eines Rechnungsdruckes als zusätzliche Zustellart (siehe „4.4.2. Zustellung der Rechnungen“ auf S. 134) genannt werden, wodurch der Rechnungsdruck im Altsystem vollständig abgelöst werden soll. Die Anwendung wird zukünftig weiterentwickelt werden, falls juristische oder betriebswirtschaftlich/organisatorische Anforderungen dies verlangen.

Aufgrund der anhaltenden Unsicherheit über künftige rechtliche Rahmenbedingungen wurde in der Testphase zuerst auf eine Rechnungssignatur verzichtet. Die an der Testphase teilnehmenden Firmenkunden wurden auf das Fehlen der Rechnungssignatur hingewiesen und erhielten ihre Rechnung zusätzlich auch auf Papier.

Es wurde damit gerechnet, dass diese Unsicherheit innerhalb des ersten Quartals 2008 von Seiten des BMF beseitigt werden würde. Da es derzeit leider nicht danach aussieht, als ob dies in den nächsten Wochen und Monaten geschehen würde, wurde im April damit begonnen, die Monatsrechnungen mit einer fortgeschrittenen Signatur zu versehen. Dazu wurde ein Zertifikat des, in „Abschnitt 3.6.3 – Auswahl eines Zertifikats“ beschriebenen, Typs „A-Cert advanced“ erworben. Mit den Rechnungen wurde im April 2008 auch ein Dokument mit einer Anleitung zur Signaturprüfung, sowie ein Dokument mit allgemeinen Informationen zur elektronischen Rechnung versandt (siehe Anhang A.10).

Rückmeldungen, Akzeptanz und Erfahrungen

Bisher wurde mit dem elektronischen Versand der Monatsrechnungen sehr gute Erfahrungen gemacht. Die Privatkunden scheinen den zusätzlichen Service sehr gut anzunehmen. Einerseits erkennbar ist dies an einigen explizit positiven Kommentaren von Kunden, andererseits aber auch durch erhaltene Rückmeldungen betreffend Adressänderungen und sonstigen Anfragen. Auch die Firmenkunden scheinen größtenteils die Vorteile der elektronischen Zusendung ihrer Rechnungen zu erkennen. Es ist damit zu rechnen, dass sie auch bei einer Totalumstellung auf die derzeit noch ausgegebenen Papierrechnungen verzichten können. Von einigen Unternehmen wurde auch schon erklärt, dass für sie auch unsignierte elektronische Rechnungen kein Problem darstellen, da sie für ihre Tankrechnung nicht vorsteuerabzugsberechtigt sind. Es soll aber hier auch nicht verschwiegen werden, dass bereits einzelne Firmenkunden angekündigt haben, der Zusendung von elektronischen Rechnungen zu widersprechen. Trotz einiger Ausnahmen wurde das in „Abschnitt 3.4.3 – Fazit“ definierte Ziel, den Kunden eine bequeme und moderne Alternative zur früher praktizierten Rechnungsverteilung zu bieten, aber sicherlich erreicht.

Derzeit existieren jedoch noch einige Zweigleisigkeiten. So müssen Rechnungen für Firmenkunden und für Kunden, die noch keine Emailadresse bekannt gegeben haben, weiterhin ausgedruckt werden. Trotzdem konnte bereits in der Testphase darauf verzichtet werden, die Rechnungen wie in „Abschnitt 3.2.2 – Rechnungsversand“ beschrieben, an das Kreditinstitut zur Verteilung zu übergeben. Damit wurde bereits jetzt ein ganz wichtiges Ziel der Umstellung erreicht. Eine vollständige Umstellung auf die elektronische Rechnung ist im betrachteten Betrieb in den nächsten Jahren eher als utopisch anzusehen, da es wahrscheinlich immer Firmenkunden geben wird, die nicht auf eine

Papierrechnung verzichten möchten. Es wird aber davon ausgegangen, dass die Anzahl der zu verteilenden Papierrechnungen relativ gering sein wird und auch kontinuierlich abnehmen wird.

Persönliche Erfahrungen

Während der Erstellung dieser Diplomarbeit konnte Kontakt zu sehr vielen interessanten Personen und Organisationen geknüpft werden. Vor allem die Teilnahme am e-Billing-Arbeitskreis war für den Autor eine tolle Erfahrung. Die regelmäßigen Meetings erlaubten es, nicht nur Informationen aus erster Hand zu gewinnen, sondern halfen auch, die Standpunkte der Stakeholder besser zu verstehen.

Vor allem auch die Diskussionen bei Experteninterviews betreffend wirtschaftlicher und rechtlicher Kriterien waren für den Autor sehr interessant und hilfreich, denn als Techniker neigt man von Zeit zu Zeit dazu, sich zu sehr vom technisch Machbaren begeistern zu lassen und zu vergessen, dass auch wirtschaftliche Anforderungen und rechtliche Rahmenbedingungen sowie vorhandene Benutzerkenntnisse eine sehr wichtige Rolle spielen. Die digitale Signatur ist hier praktisch ein Paradebeispiel. Sie ist aus technischer Sicht enorm elegant und auch juristisch definiert, kann im Alltag den durchschnittlichen Benutzer aber schon einmal überfordern.

Die Erstellung dieser Diplomarbeit war aufgrund der verschiedenen Sichten auf das Thema (siehe „1.2.3. Verschiedene Sichten“ auf S. 9) und der andauernden, aber spannenden Umbruchsphase eine große Herausforderung.

6 Literaturverzeichnis

[1] *Bundesgesetz über elektronische Signaturen (Signaturgesetz - SigG).* http://www.signatur.rtr.at/repository/legal-sigg-19990819-de.pdf, 1999.

[2] *OID Info.* http://www.oid-info.com/, 2007.

[3] *Textgegenüberstellung - Änderung des Signaturgesetzes.* http://www.parlinkom.gv.at/PG/DE/XXIII/ME/ME_00090/imfname_084434.pdf, 2007.

[4] *Vorblatt - Änderung des Signaturgesetzes.* http://www.parlinkom.gv.at/PG/DE/XXIII/ME/ME_00090/imfname_084433.pdf, 2007.

[5] *Bundesgesetz über elektronische Signaturen (Signaturgesetz - SigG).* http://www.signatur.rtr.at/de/legal/sigg.html, 2008.

[6] *Verordnung des Bundeskanzlers über elektronische Signaturen (Signaturverordnung 2008 - SigV 2008).* http://www.signatur.rtr.at/de/legal/sigv.html, 2008.

[7] A-Trust GmbH: *CPP Multi File Signer - eBilling Kartenleser.* http://labs.a-trust.at/source/cpp_multisigner.asp.

[8] Amtsblatt der Europäischen Gemeinschaften: *Richtlinie 1999/93/EG über gemeinschaftliche Rahmenbedingungen für elektronische Signaturen.* ABl. L13 vom 19.1.2000, Seiten 12–20, 2000.

[9] Amtsblatt der Europäischen Gemeinschaften: *Richtlinie 2001/115/EG.* ABl. L15 vom 17.1.2002, Seiten 24–28, 2002.

[10] AustriaPro: *ebInterface.* http://www.ebinterface.at.

[11] AustriaPro: *ebInterface 2.1 - Der österreichische Rechnungsstandard.* http://www.ebinterface.at/download/documentation/ebInvoice_2p1.pdf, 2007.

[12] AustriaPro: *Foliensatz: Workshop ebTransfer*. http://portal.wko.at/wk/dok_detail_file.wk?AngID=1&DocID=706025&StID=336023-, 2007.

[13] AustriaPro: *ebCrossBorder - Final Report*. http://portal.wko.at/wk/dok_detail_file.wk?AngID=1&DocID=804134&ConID=302285, 2008.

[14] AustriaPro: *ebCrossBorder - Zusammenfassung Final Report*. http://portal.wko.at/?382865, 2008.

[15] AustriaPro: *Wir über uns*. http://portal.wko.at/wk/startseite_dst.wk?AngID=1&DstID=1637, 2008.

[16] Bauer, Christian und Gavin King: *Java Persistence with Hibernate*. Manning Publications, 2006. ISBN 9781932394887.

[17] BDC EDV-Consulting GmbH. http://www.bdc.at, 2007.

[18] Beeger, Robert F., Arno Haase, Stefan Roock und Sebastian Sanitz: *Hibernate - Persistenz in Java-Systemen mit Hibernate und der Java Persistence API*. dpunkt.verlag, 2007. ISBN 9783898644471.

[19] Bialek, Boris: *Strukturwandel - Tabellen und XML-Objekte in derselben SQL-Datenbank*. CT - Magazin für computer technik, (20/07):14–19, September 2007.

[20] BMF: *Aufwendungen, im Zusammenhang mit Personenkraftwagen, Kombinationskraftwagen oder Krafträdern*. http://www.bmf.gv.at/Steuern/Fachinformation/Umsatzsteuer/AuslndischeUnternehmer/Vorsteuern/_start.htm.

[21] BMF: *Umsatzsteuergesetz (UStG)*. http://www.steuerberater.at/gesetze/ustg/.

[22] BMF: *Umsatzsteuerrichtlinien*. http://www.steuerverein.at/umsatzsteuer/, 2000.

[23] BMF: *Vorsteuer und Vorsteuerabzug*. http://www.help.gv.at/Content.Node/80/Seite.800231.html, 2005.

[24] BMF: *Änderung der Umsatzsteuerrichtlinien - Anforderungen an eine auf elektronischem Weg übermittelte Rechnung*. http://www.a-sit.at/pdfs/erlass_el_re_legung_UStR-elektronische_Rechnung0705.pdf, 2005.

[25] BMF Presseinformation: *Pilotversuch bei Reverse Charge Modell für Österreich gut vorstellbar.* http://bmf.gv.at/Pressecenter/Archiv/2007/6774.htm, 2007.

[26] Bundeskanzleramt Österreich: *Die österreichische Bürgerkarte - Applikationsschnittstelle Security Layer.* http://www.buergerkarte.at/konzept/securitylayer/spezifikation/aktuell/core/Core.html, 2005.

[27] Bundesministerium für Wirtschaft und Arbeit: *ebInvoice Rechnungseingangsbuch - kostenloses Prüftool für elektronische Rechnungen.* http://www.bmwa.gv.at/BMWA/Schwerpunkte/Wirtschaftspolitik/InnovaTechnol/Initiativen/ebinvoice.htm, 2007.

[28] Burger, Martin: *Analyse der Veränderungen von Geschäftsprozessen durch den Einsatz des eb-Interface.* http://www.ebinterface.at/download/2006_06_ebInterface_WUStudie.pdf, 2006.

[29] Cryptas it-Security GmbH Austria: *X.509 Zertifikatserweiterungen.* http://www.cryptoshop.com/de/knowledgebase/pki/certificates/msextensions.php, 2004.

[30] Cryptas it-Security GmbH Austria: *Zertifikate.* http://www.cryptoshop.com/de/knowledgebase/pki/certificates, 2004.

[31] Data Systems Austria: *Stellungnahme der Data Systems Austria zur Änderung des SigG.* http://www.parlinkom.gv.at/PG/DE/XXIII/ME/ME_00090_22/imfname_086723.pdf, 2007.

[32] Denk, Petra: *Parallelbuchhalter Staat.* Industrie Magazin, (5):84–86, Mai 2007.

[33] Der Standard: *Reverse-Charge-Modell.* http://derstandard.at/?url=/?id=2472507, 2006.

[34] Dittrich, Alfred: *e-Billing aus Sicht der öffentlichen Verwaltungen.* http://e-government.adv.at/2007/pdf//Dittrich_e-Billing_Paper_20070525.pdf, 2007.

[35] Dittrich, Alfred: *e-Billing in Österreich - Der Rechtliche Rahmen und Aktuelles aus Sicht der Finanzverwaltung.* http://www.wifiwien.at/Images/Download/e-Billing_Austria/A.Dittrich%20-%20BM%20f.%20Finanzen.pdf, 2007.

[36] Dyson, Tom: *PostgreSQL and XML.* http://www.throwingbeans.org/postgresql_and_xml.html, 2007.

[37] EBPP. `https://www.e-rechnung.at`.

[38] EBPP: *e-Rechnung Mail Produktbeschreibung*. `https://www.e-rechnung.at/docs/Produktbeschreibung_e-Rechnung_Mail_V1.4.pdf`, 2006.

[39] Gschwandtner, Wolfgang: *e-Rechnung*. `http://www.austriapro.at/projekte/ebinterface/2005_07_13_gschwandtner.pdf`, 2005.

[40] Hibernate: *Best Practices for Thick-Client Applications*. `http://www.hibernate.org/333.html`, 2005.

[41] Hook, David: *Beginning Cryptography with Java*. Wiley Publishing, Inc., 2005. ISBN 0764596330.

[42] Hühnlein, Detlef und Ulrike Korte: *Signaturformate für elektronische Rechnungen*. `http://www.competence-site.de/itsecurity.nsf/749B6BFA69B8CDA5C12572F10053D595/$File/signaturformate_f%C3%BCr_elektronische_rechnungen.pdf`.

[43] it20one: *Stellungnahme der it20one zur Änderung des SigG*. `http://www.parlinkom.gv.at/PG/DE/XXIII/ME/ME_00090_19/imfname_086392.pdf`, 2007.

[44] Kohl, Jürgen: *Preload Pattern - Faules Vorausladen*. Java Magazin, (4.08):14–19, März 2008.

[45] Laga, Gerhard: *Rechtliche Rahmenbedingungen*. `http://www.daa.at/getbinobj.asp?id=649`, 2007.

[46] Lowagie, Bruno: *iText in Action*. Manning Publications, 2007. ISBN 1932394796.

[47] Mayer, Peter F.: *Das Desaster e-Rechnung*. `http://www.telekom-presse.at/services/meinung_28329.html`, 2007.

[48] Mesonic Datenverarbeitungs GmbH: *Das Synergiepotential von e-Rechnungen*. `http://wko.at/ebusiness/eday_2007/vortraege/aller_eday2007.pdf`, 2007.

[49] Mesonic Datenverarbeitungs GmbH: *EbRe - e-Billing-Rechnungseingangsbuch*. `http://wko.at/ebusiness/e-rechnung/EbRe.pdf`, 2007.

[50] Post AG: *Der Brief*. `http://www.post.at/files/briefpost_brief_folder_200707.pdf`, 2007.

[51] Ramin Sabet, Gregor Jöbstl: *a.sign MultiSign*, 2008.

[52] RTR GmbH. `http://www.signatur.rtr.at`.

[53] RTR GmbH: *Positionspapier zu § 2 Z 3 lit. a bis d SigG ("fortgeschrittene elektronische Signatur").* `http://www.signatur.rtr.at/repository/rtr-advancedsignature-10-20040413-de.pdf`, 2004.

[54] RTR-GmbH und A-SIT: *Empfohlene Algorithmen und Parameter für elektronische Signaturen.* `http://www.signatur.rtr.at/repository/rtr-algorithms-20070601-de.pdf`, 2007.

[55] Scharinger, Josef: *Foliensatz LVA „Kryptographie"*, 2005.

[56] Singh, Simon: *Geheime Botschaften - Die Kunst der Verschlüsselung von der Antike bis in die Zeit des Internet.* Deutscher Taschenbuch Verlag, 2005. ISBN 3423330716.

[57] Soares, Paulo: *How to sign a PDF using iText and iTextSharp.* `http://itextpdf.sourceforge.net/howtosign.html`.

[58] Soares, Paulo: *iText Mailarchiv - Signatures with SHA256.* `http://www.mail-archive.com/itext-questions@lists.sourceforge.net/msg29685.html`, 2007.

[59] Sonntag, Michael: *E-Business Recht - Eine Einführung für Informatiker.* Trauner Verlag, 2006. ISBN 9783854991205.

[60] Statistik Austria: *Hauptergebnisse der Leistungs- und Strukturstatistik 2005 nach Beschäftigungsgrößenklassen.* `http://www.statistik.at/web_de/static/leistungs-_und_strukturstatistik_2005_-_hauptergebnisse_der_leistungs-_und_024258.pdf`, 2005.

[61] Stiftung Secure Information and Communication Technologies SIC: *PKCS11 Provider.* `http://jce.iaik.tugraz.at/sic/products/core_crypto_toolkits/pkcs_11_provider`.

[62] Stiftung Secure Information and Communication Technologies SIC: *PKCS11 Wrapper - Lizenz.* `http://jce.iaik.tugraz.at/sic/products/core_crypto_toolkits/pkcs_11_wrapper/license`, 2002.

[63] Strobl, Gerhard: *Flexdoc - e-Billing Portal.* `http://portal.wko.at/wk/dok_detail_file.wk?AngID=1&DocID=761015&StID=361145`, 2007.

[64] Sun: *Java PKCS11 Reference Guide*. http://java.sun.com/j2se/1.5.0/docs/guide/security/p11guide.html, 2004.

[65] Sun Developer Network: *Leveraging Security in the Native Platform Using Java SE 6 Technology*. http://java.sun.com/developer/technicalArticles/J2SE/security/#1, 2006.

[66] Telekom Austria: *eBilling Online*. http://kmu.telekom.at/Produkte/OnlineServices/eBilling/index.php.

[67] Telekom Austria: *eBilling Service*. http://business.telekom.at/produkte/ebusiness/ebilling/index.php.

[68] Ullenboom, Christian: *Java ist auch eine Insel - Programmieren mit der Java Standard Edition Version 6*. http://www.galileocomputing.de/openbook/javainsel7/, 2007.

[69] W3C: *XML-Signature Syntax and Processing*. http://www.w3.org/TR/xmldsig-core/, 2002.

[70] Wikipedia: *Abstract Syntax Notation One*. http://en.wikipedia.org/wiki/Abstract_Syntax_Notation_One, 2008.

[71] Wikipedia: *Distinguished Encoding Rules*. http://en.wikipedia.org/wiki/Distinguished_Encoding_Rules, 2008.

[72] Wikipedia: *Umsatzsteuer*. http://de.wikipedia.org/wiki/Umsatzsteuer, 2008.

[73] WKÖ: *e-Rechnung*. http://www.wko.at/e-rechnung.

[74] WKÖ: *Klein- und Mittelbetriebe in Österreich*. http://wko.at/statistik/kmu/kmu.htm.

[75] WKÖ: *Umsatzsteuer und Vorsteuer - Eine Einführung*. http://www.wkw.at/docextern/abtfinpol/extranet/wkoat/Umsatzsteuer/UmsatzsteuerVorsteuerEinfuehrungwko.pdf, 2006.

[76] WKÖ: *Stellungnahme der WKÖ zur Änderung des SigG*. http://www.parlinkom.gv.at/PG/DE/XXIII/ME/ME_00090_17/imfname_086375.pdf, 2007.

[77] Wolf, Ruben: *Verifikation digitaler Signaturen*. http://www.informatik.tu-darmstadt.de/BS/Lehre/Sem98_99/T11/index.html#tth_sEc4.2, 1999.

Alle in dieser Arbeit angegebenen Weblinks wurden am 25.04.2008 zum letzten Mal geprüft.

Glossar

Begriff	Beschreibung
ASN.1	Eine Beschreibungssprache zur Definition von Datenstrukturen und deren Aufbau. Der Inhalt und die Struktur von X.509 Zertifikaten werden durch ASN.1 beschrieben. ASN.1 ist ein Standard der ITU-T und der ISO und ist durch die ITU-T Empfehlungen X.209 und X.690 definiert. [70]
Debitorensalden	die Verbindlichkeiten der Kunden
DER	Eindeutige Kodierungsvorschriften, die sicherstellen, dass in ASN.1 definierte Objekte auf allen Plattformen gleich kodiert werden. Dadurch wird z. B. sichergestellt, dass ein X.509-Zertifikat auf allen Maschinen die selbe Repräsentation besitzt. Dies ist vor allem auch deshalb wichtig, da Zertifikate signiert sind und das Ergebnis der Signaturprüfung nicht von der Kodierung beeinflusst werden darf. DER ist in der ITU-T Empfehlung X.690 spezifiziert. [71]
Distinguished Name	unterscheidbarer Name; stellt sicher, dass ein Zertifikatseigentümer eindeutig identifizierbar ist
e-Billing	Synonym für die elektronische Rechnungsstellung; umfasst Erstellung sowie Entgegennahme
e-Card	Die österreichische e-card ist eine SmartCard im Scheckkartenformat. Sie ersetzt den Krankenschein, ist für die österreichischen Bürger Schlüssel zum Gesundheitssystem und ermöglicht den Zugang zu e-Government-Diensten.

Begriff	**Beschreibung**
ERP-System	ein IT-System zur Planung des Einsatzes von Unternehmensressourcen
Framework	ein Softwarehalbfabrikat; Frameworks sind Softwarebauteile, welche wiederverwendbare Teile einer Aufgabe implementieren, und dem Entwickler so ein Grundgerüst oder eine Architektur vorgeben, wo er seine projektspezifische Implementierung „einstecken“ kann
Fulfillment	Ausführung, Erfüllung
Marge	Preisspanne
Outsourcing	Auslagerung
SmartCard	Ein Mikroprozessor, der meist auf einer Plastikkarte mit den Ausmaßen einer Kreditkarte aufgebracht ist (Format ID-1).
Stakeholder	Personen oder Personengruppen, die an einem gewissen Themengebiet interessiert sind
Web	umgangssprachliche Kurzversion für das Internet
Workflow	ein Arbeitsablauf; eine definierte Abfolge von Aktivitäten, um eine Aufgabe zu lösen

A Anhang

A.1. Einteilung von Unternehmen

Tabelle A.1 gibt einen Überblick über die Kategorisierung von Unternehmen nach Anzahl der Beschäftigten und die prozentuelle Aufteilung der österreichischen Unternehmen in diese Kategorien. Unternehmen könnten auch nach anderen Gesichtspunkten wie dem Umsatz eingeteilt werden, worauf hier nicht weiter eingegangen wird (siehe [74] für weiterführende Informationen). Die angegebenen Daten stammen von [60], und können umfassend aufbereitet ebenfalls von [74] bezogen werden. [74] bietet außerdem weitere Statistiken und allgemeine Erläuterungen zum Thema KMU.

Unternehmen	**Anzahl Mitarbeiter**	**Anteil in %**[a]
Kleinstunternehmen	bis 9	87,3 %
Kleinunternehmen	10 bis 49	10,6 %
Mittlere Unternehmen	50 bis 249	1,7 %
Großunternehmen	ab 250	0,4 %

[a]Prozentueller Anteil der Unternehmen dieser Kategorie an der Anzahl der Gesamtunternehmen

Tabelle A.1.: Kategorisierung von Unternehmen und Aufteilung

A.2. Übersicht: e-Billing-Produkte und -Services

Tabelle A.3 gibt eine Übersicht über einige am Markt erhältliche e-Billing-Lösungen. Die Auswahl der Produkte ist völlig willkürlich getroffen und es sollen auch keine Aussagen über die Qualität einzelner Produkte oder Dienste gemacht werden. Die Tabelle dient rein zur Information und als Startpunkt für weitere Nachforschungen. Die Informationen zu den einzelnen Produkten wurden aus Produktbeschreibungen und den Web-Seiten der Hersteller entnommen. Für deren Richtigkeit und Vollständigkeit kann daher nicht garantiert werden. Bei Interesse an einem der Produkte oder Services ist der beste Weg, den Anbieter direkt zu kontaktieren. Dazu listet Tabelle A.2 die Hersteller und ihre Web-Adressen auf.

Die in Tabelle A.3 verwendeten Abkürzungen:

Signatur F = fortgeschritten, Q = qualifiziert

Phasen ER=Erstellung, KO=Konvertierung, S=Signatur, AV=Archivierung Versender, V=Versand, EM=Empfang, SP=Signaturprüfung, Ü=Übernahme, B=Bezahlung, AE=Archivierung Empfänger, eingeschränkt SP = es werden zwar Funktionen zur Verifikation geboten, automatisiert ist dies allerdings nicht möglich

Anbieter	**Art des e-Billing-Produktes**	**Homepage**
A-Cert	ZDA	`http://www.a-cert.at`
A-Trust	akkreditierter ZDA, Signaturlösungen	`http://www.a-trust.at`
BDC	Signaturlösungen	`http://www.bdc.at`
EBPP GmbH	e-Billing-Services	`https://e-rechnung.at`
GRZ IT Center Linz	e-Billing-Services	`https://www.flexdoc.at`
Infoworx	Fakturierungssystem	`http://www.infoworx.at`
itSolution	Signaturlösungen	`http://www.itsolution.at`
Mesonic	ERP- und FIBU-Systeme	`http://www.mesonic.com`
Telekom Austria	e-Billing-Services	`http://www.telekom.at`
xyzmo	Signaturlösungen	`http://www.xyzmo.com`

Tabelle A.2.: Anbieter

Hersteller	Produkt	Beschreibung	unterstützte Phasen	Signatur	Integration
Services:					
EBPP	e-Rechnung Mail	siehe „3.6.2.1. “ auf S. 61	S, V	F	SMTP, Datei-Schnittstelle
EBPP	e-Rechnung Consolidator	siehe „3.6.2.1. “ auf S. 62	S, KO, V, AV, EM, SP, Ü, AE	F	Webservices, FTP, Connect Direct
GRZ	Flexdoc	siehe „3.6.2.1. “ auf S. 62	S, KO, V, AV, EM, SP, AE	F	Webservices, Web-Plattform, Druckertreiber
Telekom Austria	kleine Variante	siehe „3.6.2.1. “ auf S. 64	S, V, AV	F	Web-Oberfläche
Telekom Austria	große Variante	siehe „3.6.2.1. “ auf S. 64	S, KO, V, AV	F	Web-Oberfläche
Signaturapplikationen:					
A-Trust	MultiSign	Desktop-Applikation zur Massensignatur von PDF-Daten	S, eingeschränkt SP	F, Q	über GUI
BDC	hotBill	Server-Applikation für KMU, ermöglicht die Signatur von PDF, XML und ganzen Mails (S/MIME)	S, V, eingeschränkt SP	F, Q	SMTP, Datei-Schnittstelle
BDC	hotInvoice	Server-Applikation für größere Unternehmen. Ermöglicht außerdem die Verschlüsselung von Rechnungen und den Einsatz eines HSM.	S, V, SP	F, Q	SMTP, Datei-Schnittstelle
BDC	hotSign	Desktop-Applikation zur Signatur von PDF- und XML-Daten	S, eingeschränkt SP	F, Q	über GUI
itSolution	trustDesk	Desktop-Applikation zur Massensignatur von PDF- und XML-Daten	S, eingeschränkt SP	F, Q	über GUI

itSolution	PKI business server	Server-Applikation für größere Unternehmen; Mandantenfähig; unterstützt HSMs	S, V, SP	F, Q	SMTP, über GUI von trustDesk
xyzmo	Seal Client	Desktop-Applikation zur Einzel- und Massensignatur von Dokumenten und Umwandlung in PDF. Erzeugt spezielle Xyzmo Seal Signaturen. Abrechnung pro Signatur (Prepaid-System).	S, KO, eingeschränkt SP	F	Druckertreiber
xyzmo	Seal Server	Server-Applikation zum Signieren von PDF, XML und anderen Dokumenten. Kann auch spezielle Xyzmo Seal Signaturen erzeugen.	S, KO, V, SP	F, Q	Druckertreiber, Webservices, andere
ERP-Systeme und Fakturierungssoftware:					
Mesonic	winline e-Billing	Server-Applikation zum Signieren, Versenden, und Empfangen von elektronischen Rechnungen im Format ebInterface.	ER, S, V, SP, EM	k. A.	k. A., wahrscheinlich integriert in Winline
Infoworx	Suite 2008	Desktop-Fakturierungsapplikation welche ebInterface-Rechnungen erstellt	ER, S, V	F	in Applikation integriert
Rechnungserstellung:					
AustriaPro	WebForms	frei benutzbare Web-Applikation zum Erstellen und Signieren von ebInterface-Rechnungen	ER, S	F, Q	Web-Oberfläche
NTx	eBilling@-Word 2007	Microsoft Word 2007 Plugin zum Erstellen und Signieren von ebInterface-Rechnungen	ER, S	F, Q	Microsoft Word 2007
sonstige:					
Mesonic	ebRe	frei erhältliche Desktop-Applikation zur Signaturprüfung und Archivierung von ebInterface-Rechnungen	SP, AE	-	Verzeichnis

NTx	eBilling@-Sharepoint	erlaubt die Signaturprüfung und Weiterverarbeitung von ebInterface-Rechnungen mittels Microsoft Sharepoint Portalserver und Portal Service	SP, AE, Ü	-	Sharepoint
Adobe	Acrobat Reader	Desktop-Applikation zum Betrachten von PDF-Dokumenten. Erlaubt auch eine Signaturprüfung.	eingeschränkt SP	-	über GUI

Tabelle A.3.: e-Billing Produkte und Services

A.3. Änderungen im SigG und die neue SigV

Anfang 2008 wurde das österreichische `SigG` [5] in einigen Punkten geändert, und es wurde eine neue `SigV` [6] geschaffen. Im Folgenden werden einige für die elektronische Rechnung wichtige Änderungen erläutert. Eine vollständige Textgegenüberstellung zwischen alter und neuer Version bietet [3]. [4] kommentiert die einzelnen Änderungen und zeigt die jeweiligen Gründe auf.

Gegenstand und Anwendungsbereich

- Nach § `1 Abs 3` ist das `SigG` nun nur mehr auf ZDA anzuwenden, die qualifizierte Zertifikate ausstellen. Davon ausgenommen sind nur drei Abschnitte, nämlich:
  - § `6 Abs 1`: *„Die Aufnahme und die Ausübung der Tätigkeit eines ZDA bedürfen keiner gesonderten Genehmigung.“*
  - § `22`: regelt den Datenschutz in Verbindung mit den Diensten der ZDA
  - § `24`: regelt die Anerkennung ausländischer Zertifikate

  Das `SigG` gilt zwar nach wie vor auch für die Anfertigung von fortgeschrittenen und einfachen elektronischen Signaturen, und es regelt dafür z. B. auch weiterhin die Pflichten des Signators. Durch die eben erwähnte Einschränkung, ergeben sich für fortgeschrittene und einfache Signaturen aber folgende Änderungen:
  - das Führen von Widerrufsdiensten ist nicht mehr gesetzlich vorgeschrieben
  - das Führen von Verzeichnisdiensten ist nicht mehr gesetzlich vorgeschrieben
  - ZDA welche nur einfache oder fortgeschrittene Zertifikate anbieten, unterliegen nicht mehr der Prüfung durch die RTR GmbH

  Dadurch wird die fortgeschrittene Signatur, wie sie für elektronische Rechnungen mindestens verwendet werden muss, ganz klar in ihrer Qualität beschnitten. Als offizieller Grund für diese Einschränkung wird angegeben, dass man das Ausüben der Tätigkeit eines ZDA attraktiver gestalten möchte. Kosteneinsparungen auf Seiten der RTR GmbH könnten hier aber ebenfalls eine Rolle spielen.

Begriffsbestimmung

- § 2 Z 2 hält nun fest, dass ein Signator nicht mehr nur eine natürliche Person, sondern eine *„Person oder sonstige rechtsfähige Einrichtung“* sein kann. Dies könnte die elektronische Rechnungssignatur etwas vereinfachen, da nicht mehr ein Mitarbeiter als Signator auftreten muss. Vor der Änderung musste ein Zertifikat auf einen Mitarbeiter ausgestellt werden, welches bei einem Ausscheiden des Mitarbeiters widerrufen werden musste. Qualifizierte Zertifikate stellen aber weiterhin eine Ausnahme dar, und können auch weiterhin nur auf natürliche Personen ausgestellt werden.

- § 2 Z 3 definiert nun ausdrücklich eine „fortgeschrittene elektronische Signatur“. Früher wurde dieser Begriff nur implizit über andere Texte wie z. B. die Umsatzsteuerrichtlinie an § 2 Z 3 lit. a bis d zugewiesen.

- Die „sichere Signatur“ heißt nun „qualifizierte Signatur“.

Ausstellung qualifizierter Zertifikate

- Laut § 8 Abs 1 muss nun bei der Ausstellung eines qualifizierten Zertifikates nicht mehr unbedingt ein amtlicher Lichtbildausweis kontrolliert werden. Hier können nun auch andere gleichwertige Methoden, wie die Zustellung zur eigenen Hand per RSa-Brief, verwendet werden.

Technische Komponenten und Verfahren

- Die Bestimmungen zur qualifizierten Massensignatur, nach welcher dem Signator zur Zeit der PIN-Eingabe die Anzahl der zu signierenden Dokumente bekannt sein muss, ist nun auch im SigG in § 18 Abs 2 geregelt.

A.4. Signatur und Verifikation mit XMLDsig

Bei der Erzeugung einer „XML-Signature“ werden die folgenden Schritte abgearbeitet (nach [69]):

1. für jedes *Reference*-Element:
   a) hole das Objekt welches durch *Reference* referenziert werden soll
   b) wende die Liste der *Transforms* konsekutiv auf das Objekt an
   c) bilde mittels des gegebenen Hash-Algorithmus einen Hashwert über die transformierten Daten
   d) erzeuge ein *Reference*-Objekt:
      i. gib den URI an, der festlegt wie dieses Objekt geholt werden kann (dieser Punkt ist optional)
      ii. schreibe die Liste der angewendeten Transformationen in das *Transforms*-Element
      iii. schreibe den verwendeten Hash-Algorithmus in das *DigestMethod*-Element
      iv. schreibe den berechneten Hashwert in das *DigestValue*-Element
2. erzeuge ein *SignedInfo*-Element mit *SignatureMethod*, *CanonicalizationMethod* und den *References* (welche in 1.4. erzeugt wurden)
3. normalisiere die nun in *SignedInfo* enthaltenen XML-Inhalte mittels dem gegebenen Algorithmus, und schreibe den verwendeten Algorithmus in das *CanonicalizationMethod*-Element
4. bilde einen Hashwert über den Inhalt in *SignedInfo* und signiere diesen; Schreibe die verwendeten Algorithmen (Kombination aus Hashverfahren und Signaturalgorithmus) in das Element *SignatureMethod*; Schreibe den Base64-codierten Wert der Signatur in das *SignatureValue*-Element
5. erzeuge das *KeyInfo*-Element und schreibe einen Hinweis auf den Schlüssel hinein, welcher für das Verifizieren verwendet werden kann

6. erzeuge das *SignatureElement*, welches *SignedInfo*, *SignatureValue* und *KeyInfo* enthält

Bei der Verifikation einer solchen Signatur wird dann folgendermaßen vorgegangen (nach [69]):

1. hole den Schlüssel zum Verifizieren (z. B. von *KeyInfo*)

2. entschlüssle mit ihm die Signatur in *SignatureValue* (es bleibt der Hashwert)

3. für jedes *Reference*-Element:

   a) hole das Objekt welches referenziert wird

   b) wende die Liste der in *Transforms* angegebenen Transformationen konsekutiv an

   c) bilde mittels des in *DigestMethod* gegebenen Algorithmus einen Hashwert über die transformierten Daten, und vergleiche diesen mit dem in *DigestValue* gegebenen Wert; stimmen die Werte nicht überein schlägt die Prüfung fehl

4. normalisiere den Inhalt von *SignedInfo* mittels dem in *CanonicalizationMethod* gegebenen Algorithmus

5. bilde einen Hashwert über den Inhalt von *SignedInfo* mit dem in *SignatureMethod* angegebenen Hash-Algorithmus (nur Hashwert bilden, keine Signatur), und vergleiche diesen Wert mit dem in Punkt 2) erhaltenen Wert; stimmen die Werte nicht überein, so schlägt die Prüfung fehl

A.5. Beispiel: Eine mit dem InvoiceManager erstellte und signierte ebInterface-Rechnung

Listing A.1 zeigt eine mit dem InvoiceManager erzeugte und signierte ebInterface-Rechnung. Die Base64-codierte Darstellung des Zertifikats in der Rechnung ist aus Platzgründen nur angedeutet. Abbildung A.1 zeigt die durch XSLT erzeugte HTML-Darstellung dieser Rechnung im Browser. Aus Platzgründen wurde auf die Angabe der äußerst umfangreichen XSLT-Dateien `Invoice2fo.xslt` und `Invoice2html.xslt` verzichtet.

```
1  <eb:Invoice xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" xmlns:eb="
       http://www.ebinterface.at/schema/2p1/" xmlns:xsi="http://www.w3.org
       /2001/XMLSchema-instance" eb:Cancellation="false" eb:GeneratingSystem=
       "InvoiceManager v 1.0" xsi:schemaLocation="http://www.ebinterface.at/
       schema/2p1/ http://www.ebinterface.at/schema/2p1/Invoice.xsd">
2    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3      <ds:SignedInfo>
4        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-
             xml-c14n-20010315"></ds:CanonicalizationMethod>
5        <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-
             more#ecdsa-sha1"></ds:SignatureMethod>
6        <ds:Reference URI="">
7          <ds:Transforms>
8            <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#
                 enveloped-signature"></ds:Transform>
9          <ds:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n
               -20010315#WithComments"></ds:Transform>
10   </ds:Transforms>
11   <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></
         ds:DigestMethod>
12   <ds:DigestValue>P6HP7EikHEQZZgEsime/PdcFoc0=</ds:DigestValue>
13       </ds:Reference>
14     </ds:SignedInfo>
15     <ds:SignatureValue>H3Kk0FyL0eSQieNmTHz+2Ub9+Veq4UT+Gx0dPoGUU0Nq/
           AraI0JRVDSDtV9Y94iq</ds:SignatureValue>
16     <ds:KeyInfo>
17       <ds:X509Data>
18         <ds:X509Certificate>
19           MIIEsjCCA5qgAwIBAgIDA2B1MA0GCSqGSIb3DQEBBQUAMIGX ...
20   </ds:X509Certificate>
21       </ds:X509Data>
22       <ds:KeyValue></ds:KeyValue>
23     </ds:KeyInfo>
24   </ds:Signature>
25   <eb:InvoiceNumber>338/01/2008/62</eb:InvoiceNumber>
26   <eb:InvoiceDate>2008-01-31</eb:InvoiceDate>
27   <eb:Delivery>
```

```
28      <eb:Period>
29        <eb:FromDate>2008-01-01</eb:FromDate>
30        <eb:ToDate>2008-01-31</eb:ToDate>
31      </eb:Period>
32    </eb:Delivery>
33    <eb:Biller>
34      <eb:VATIdentificationNumber>ATU23415665</eb:VATIdentificationNumber>
35      <eb:Address>
36        <eb:Salutation>Firma</eb:Salutation>
37        <eb:Name>AVIA Tankstelle H. Schlapschy</eb:Name>
38        <eb:Street>Im Tal 2</eb:Street>
39        <eb:Town>Kefermarkt</eb:Town>
40        <eb:ZIP>4292</eb:ZIP>
41        <eb:Phone>07947/5903</eb:Phone>
42        <eb:Email>aviaschlapschy@aon.at</eb:Email>
43      </eb:Address>
44    </eb:Biller>
45    <eb:InvoiceRecipient>
46      <eb:BillersInvoiceRecipientID>1921</eb:BillersInvoiceRecipientID>
47      <eb:Address>
48        <eb:Salutation>Herr</eb:Salutation>
49        <eb:Name>Wolfgang Schlapschy</eb:Name>
50        <eb:Street>Denisgasse 18</eb:Street>
51        <eb:Town>Wien</eb:Town>
52        <eb:ZIP>1200</eb:ZIP>
53      </eb:Address>
54    </eb:InvoiceRecipient>
55    <eb:Details>
56      <eb:ItemList eb:ListType="structured">
57        <eb:ListLineItem>
58          <eb:ListElement eb:Type="DateType" eb:Usage="Description">19.01</
              eb:ListElement>
59          <eb:ListElement eb:Type="IdentifierType" eb:Usage="Number">9378</
              eb:ListElement>
60          <eb:ListElement eb:Type="CodeType" eb:Usage="Number">0</
              eb:ListElement>
61          <eb:ListElement eb:Type="StringType" eb:Usage="Description">
              ALLGEMEIN </eb:ListElement>
62          <eb:ListElement eb:Type="StringType" eb:Unit="St" eb:Usage="
              Quantity">1.0</eb:ListElement>
63          <eb:ListElement eb:Type="AmountType" eb:Unit="EUR" eb:Usage="
              UnitPrice">6.4</eb:ListElement>
64          <eb:ListElement eb:Type="AmountType" eb:Unit="EUR" eb:Usage="
              Amount">6.4</eb:ListElement>
65        </eb:ListLineItem>
66        <eb:ListLineItem>
67          <eb:ListElement eb:Type="DateType" eb:Usage="Description">19.01</
              eb:ListElement>
68          <eb:ListElement eb:Type="IdentifierType" eb:Usage="Number">9378</
              eb:ListElement>
```

```
69          <eb:ListElement eb:Type="CodeType" eb:Usage="Number">0</
                eb:ListElement>
70          <eb:ListElement eb:Type="StringType" eb:Usage="Description">DIESEL
                 </eb:ListElement>
71          <eb:ListElement eb:Type="StringType" eb:Unit="l" eb:Usage="
                Quantity">51.02</eb:ListElement>
72          <eb:ListElement eb:Type="AmountType" eb:Unit="EUR" eb:Usage="
                UnitPrice">1.179</eb:ListElement>
73          <eb:ListElement eb:Type="AmountType" eb:Unit="EUR" eb:Usage="
                Amount">60.15</eb:ListElement>
74        </eb:ListLineItem>
75      </eb:ItemList>
76    </eb:Details>
77    <eb:Tax>
78      <eb:VAT>
79        <eb:Item>
80          <eb:TaxedAmount eb:Currency="EUR">55.45</eb:TaxedAmount>
81          <eb:Percentage>20.00</eb:Percentage>
82          <eb:Amount eb:Currency="EUR">11.10</eb:Amount>
83        </eb:Item>
84      </eb:VAT>
85    </eb:Tax>
86    <eb:TotalGrossAmount eb:Currency="EUR">66.55</eb:TotalGrossAmount>
87    <eb:PaymentMethod xsi:type="eb:DirectDebitType">
88      <eb:Comment>Betrag wird abgebucht von Konto-Nr. 735040180 (BLZ 15002)<
            /eb:Comment>
89    </eb:PaymentMethod>
90    <eb:PaymentConditions>
91      <eb:DueDate>2008-01-31</eb:DueDate>
92    </eb:PaymentConditions>
93    <eb:PresentationDetails>
94      <eb:URL> </eb:URL>
95      <eb:LogoURL>C:/TSVK.jpg</eb:LogoURL>
96      <eb:LayoutID>0100</eb:LayoutID>
97      <eb:Language>ger</eb:Language>
98      <eb:DocumentTitle>Rechnung</eb:DocumentTitle>
99      <eb:ShortComment>Abrechnung fuer Januar</eb:ShortComment>
100     <eb:HeaderComment>Tanken Sie jetzt mit der Kundenkarte der AVIA
            Tankstelle Schlapschy!</eb:HeaderComment>
101     <eb:FooterComment>Und tanken Sie bargeldlos guenstiger!</
            eb:FooterComment>
102   </eb:PresentationDetails>
103 </eb:Invoice>
```

Listing A.1: 338-01-2008-62.xml

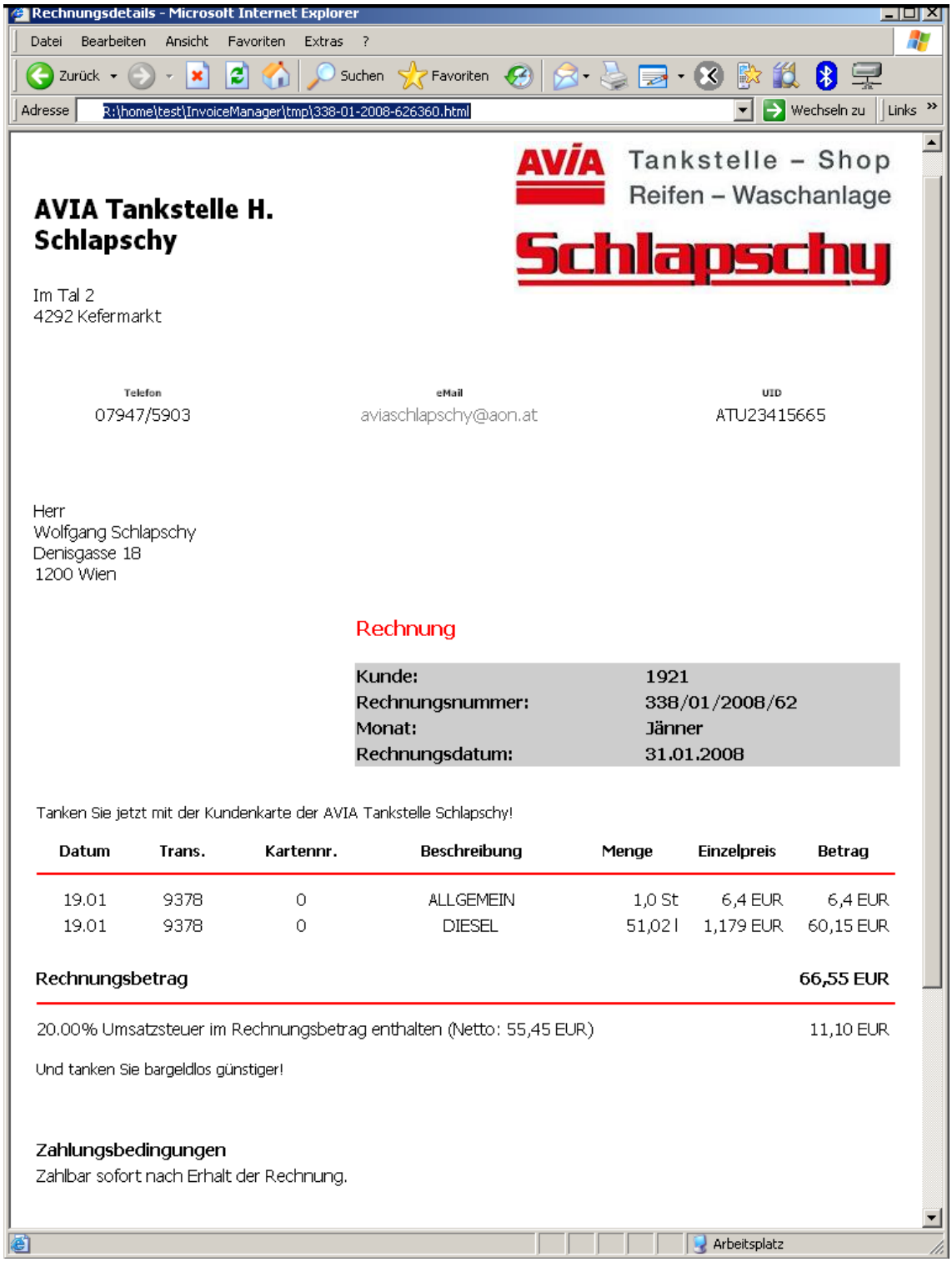

Abbildung A.1.: Darstellung der Rechnung aus Listing A.1 im Browser

A.6. Beispiel: Eine mit dem InvoiceManager erstellte und signierte PDF-Rechnung

Abbildung A.2 zeigt eine mit dem InvoiceManager erstellte und signierte PDF-Rechnung im Adobe Acrobat Reader Version 7.0.8 unter Ubuntu Linux.

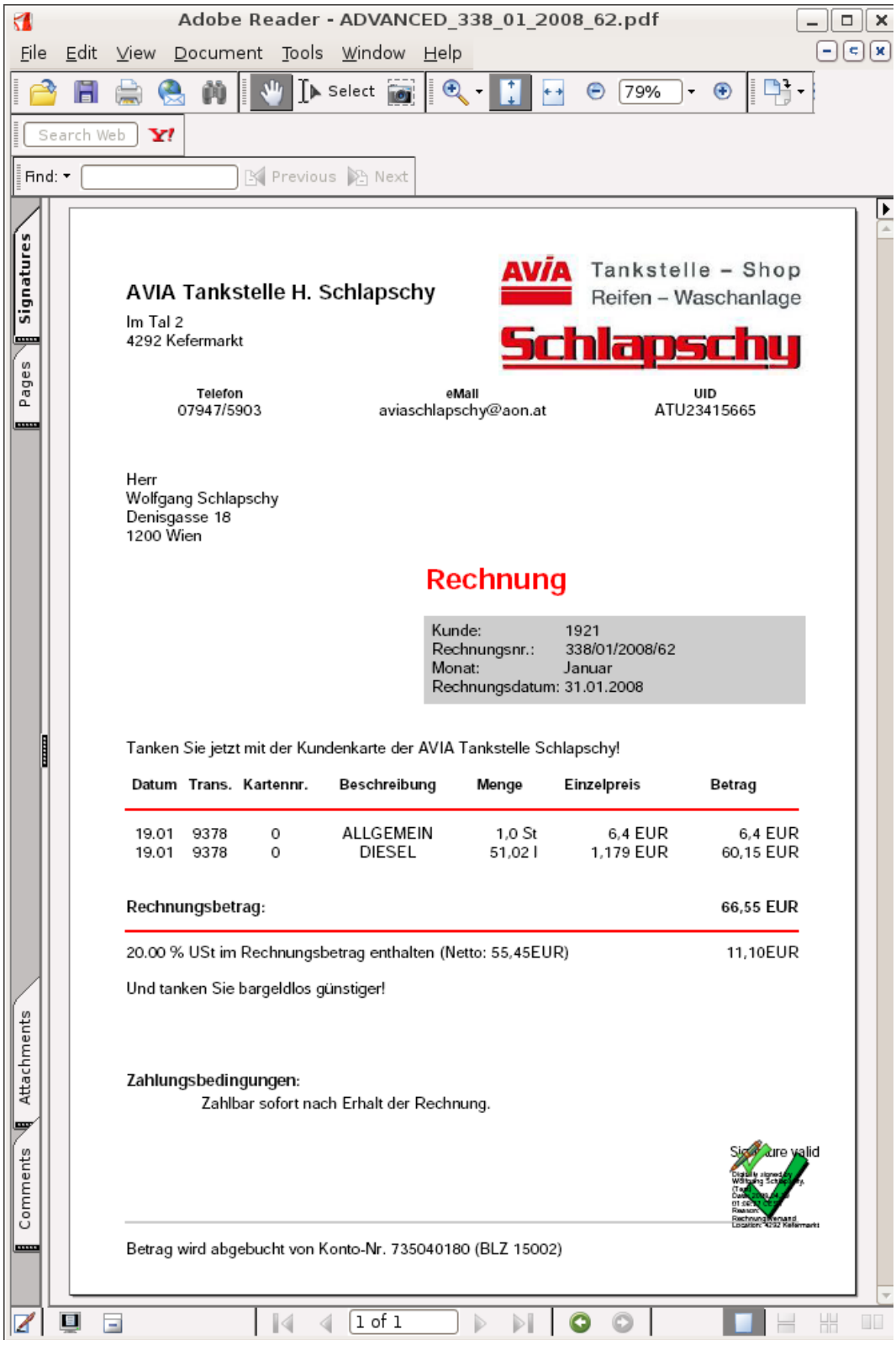

Abbildung A.2.: 338-01-2008-62.pdf

Abbildung A.3.: Vergrößerung der sichtbaren Signatur aus Abbildung A.2

A.7. Konfiguration des Acrobat Readers zur Verifikation

Ein großer Vorteil des Acrobat Readers ist, dass er für viele Betriebssysteme verfügbar ist. Geprüft wurden die vom InvoiceManager erstellten fortgeschrittenen Signaturen mit dem Adobe Acrobat Reader in der Version 7.0.8 unter Ubuntu Linux und in der Version 8.1.2 unter Windows.

Für die Signaturverifikation ist es wichtig, dass dafür die richtigen Signatur-Handler verwendet werden. Dazu ist unter `Bearbeiten - Grundeinstellungen - Sicherheit - Erweiterte Grundeinstellungen - Überprüfung` eine der beiden in Abbildung A.4 markierten Optionen ausgewählt ist. Bei Windows-Betriebssystemen ist es emp-

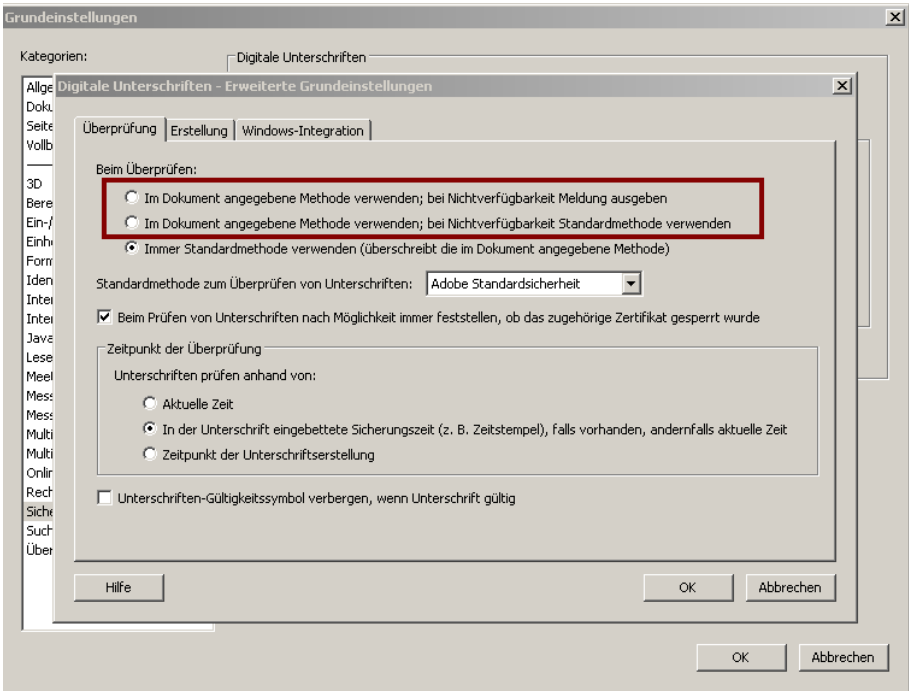

Abbildung A.4.: Konfiguration der Signatur Handler im Acrobat Reader

fehlenswert, Root-Zertifikate, welche als „Trust Anchor“ dienen sollen, im Windows-Zertifikatsspeicher zu verwalten. Dies hat den Vorteil, dass diese von mehreren Anwendungen verwendet werden können. Der Adobe Acrobat Reader kann auf die dort gespeicherten Zertifikate zugreifen, wenn unter `Bearbeiten - Grundeinstellungen - Sicherheit - Erweiterte Grundeinstellungen - Windows-Integration` mindestens die beiden in Abbildung A.5 markierten Optionen ausgewählt sind. Eine detaillierte Konfigurationsanleitung für den Acrobat Reader ist in Anhang A.10 zu finden.

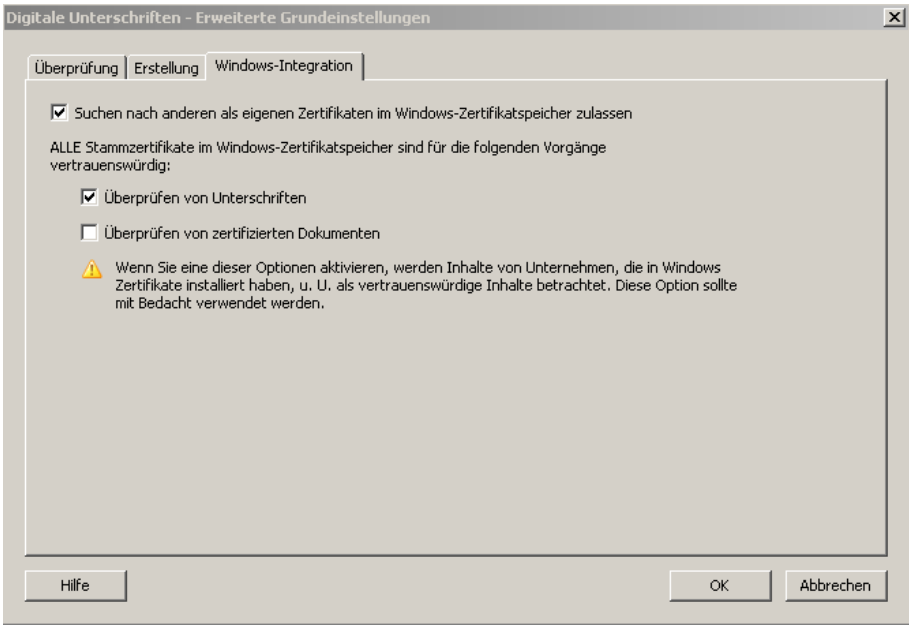

Abbildung A.5.: Windows-Zertifikatsspeicher im Acrobat Reader verwenden

A.8. Verwendung von „Platzhaltern“ im InvoiceManager

An manchen Stellen, wie beim Emailversand oder beim Rechnungsimport, können in der InvoiceManager-Anwendung Platzhalter verwendet werden. Tabelle A.8 gibt eine Übersicht über die in der derzeitigen Version verwendbaren Platzhalter.

Platzhalter	**Bedeutung**	**Beispiel**
Erzeugung der Rechnung (Angabe in `config.xml`; wird vom Importer berücksichtigt)		
%bankCode	Trägt im ebInterface-Element *directDebitPaymentComment* die Bankleitzahl des Kunden ein.	Blz.: %bankCode ⇒ Blz.: 15002
%accountNr	Trägt im ebInterface-Element *directDebitPaymentComment* die Kontonummer des Kunden ein.	Kontonummer: %accountNr ⇒ Kontonummer: 000735080120
%month	Trägt das Rechnungsmonat in das ebInterface-Element *shortComment* ein.	Abrechnung für %month ⇒ Abrechnung für April
%invoiceId	Trägt die Rechnungsnummer in das ebInterface-Element *shortComment* ein.	Rechnungsnr.: %invoiceId ⇒ Rechnungsnr.: 2311/2008/05
Email-Versand (Angabe im Dialog „Rechnungen zustellen“ über Kontextmenü möglich)		
%invoiceId	Erlaubt die Angabe der Rechnungsnummer im Betreff sowie im Text der Email.	Rechnungsnr.: %invoiceId ⇒ Rechnungsnr.: 2311/2008/05
%invoiceDate	Erlaubt die Angabe des Rechnungsdatums im Betreff sowie im Text der Email.	Rechnung vom %invoiceDate ⇒ Rechnung vom 30.3.2008
%invoiceSum	Erlaubt die Angabe der Rechnungssumme im Betreff sowie im Text der Email.	Rechnung über € %invoiceSum ⇒ Rechnung über € 500

%complementoryClose	Konstruiert eine Grußformel aus den Kundendaten und fügt diese in den Emailtext ein. Welche Grußformel konstruiert wird, hängt von der für den jeweiligen Kunden gespeicherten Anrede ab (für jede Anrede ist der Text der Grußformel in der Datei `config.xml` konfigurierbar).	%complementoryClose ⇒ Sehr geehrter Herr Wolfgang SCHLAPSCHY!

Tabelle A.4.: Platzhalter und ihre Verwendung

A.9. Beispiel: Konfigurationsdateien der InvoiceManager-Anwendung

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:configurationHandler xmlns:ns2="http://wolfgang-schlapschy.at/
    InvoiceStorage">
    <applicationTitle>InvoiceManager v 1.0</applicationTitle>
    <deliveryConfig>
        <deliveryTypeClasses>at.ws.business.invoiceDelivery.channels.
            EmailDeliveryType</deliveryTypeClasses>
        <deliveryTypeClasses>at.ws.business.invoiceDelivery.channels.
            LazyDeliveryType</deliveryTypeClasses>
        <emailDeliveryConfig>
            <senderAddress>verrechnung@test.at</senderAddress>
            <smtpHost>test.test.at</smtpHost>
            <smtpUser>verrechnung@test.at</smtpUser>
            <smtpPassword>test</smtpPassword>
            <subject>AVIA-Schlapschy Rechnung Nr. %invoiceId</subject>
            <text>%complementoryClose

Anbei senden wir Ihnen Rechnung Nr. %invoiceId vom %invoiceDate.

Mit freundlichen Gruessen

AVIA-Tankstelle H. Schlapschy
      </text>
            <complementaryCloses>
                <entry>
                    <key>Familie</key>
                    <value>Sehr geehrte Familie %name!</value>
                </entry>
                <entry>
                    <key>Herr</key>
                    <value>Sehr geehrter Herr %title %surname %name!</
                        value>
                </entry>
                <entry>
                    <key>Frau</key>
                    <value>Sehr geehrte Frau %title %surname %name!</value
                        >
                </entry>
                <entry>
                    <key>Firma</key>
                    <value>Sehr geehrte Firma %name!</value>
                </entry>
                <entry>
                    <key>empty</key>
                    <value>Sehr geehrte Damen und Herren!</value>
                </entry>
```

```
42              </complementaryCloses>
43          </emailDeliveryConfig>
44      </deliveryConfig>
45      <visualConfig>
46          <tmpDir>./tmp</tmpDir>
47      </visualConfig>
48      <importConfig>
49          <importClasses>at.ws.business.invoiceImport.impl.
                TxtInvoiceImporter</importClasses>
50          <txtImportConfig>
51              <currency>EUR</currency>
52              <directDebitPaymentComment>Betrag wird abgebucht von Konto-Nr.
                    %accountNr (BLZ %bankCode)</directDebitPaymentComment>
53              <bankTransactionPaymentComment>Wir bitten um termingerechte
                    Ueberweisung auf unser Konto.</
                    bankTransactionPaymentComment>
54              <billerBankName>Oberbank Freistadt</billerBankName>
55              <billerBankCode>15002</billerBankCode>
56              <billerBankAccountNumber>652303082</billerBankAccountNumber>
57              <documentTitle>Rechnung</documentTitle>
58              <language>ger</language>
59              <layoutId>0100</layoutId>
60              <logoUrl>C:/TSVK.jpg</logoUrl>
61              <generatingSystem>InvoiceManager v 0.1</generatingSystem>
62              <invoiceYear>2007</invoiceYear>
63              <daysToDue>0</daysToDue>
64              <shortComment>Abrechnung fuer %month</shortComment>
65              <headerComment>Tanken Sie jetzt mit der Kundenkarte der AVIA
                    Tankstelle Schlapschy!</headerComment>
66              <footerComment>Und tanken Sie bargeldlos guenstiger!</
                    footerComment>
67              <billerUrl> </billerUrl>
68              <billerVatIdentificationNumber>ATU23415665</
                    billerVatIdentificationNumber>
69              <billerSalutation>Firma</billerSalutation>
70              <billerName>AVIA Tankstelle H. Schlapschy</billerName>
71              <billerStreet>Im Tal 2</billerStreet>
72              <billerZIP>4292</billerZIP>
73              <billerTown>Kefermarkt</billerTown>
74              <billerEmail>aviaschlapschy@aon.at</billerEmail>
75              <billerPhone>07947/5903</billerPhone>
76              <UnitIdentifierList>
77                  <UnitIdentifier Unit="1" Name="BENZIN "/>
78                  <UnitIdentifier Unit="1" Name="DIESEL "/>
79                  <UnitIdentifier Unit="1" Name="EUROSUPER "/>
80              </UnitIdentifierList>
81              <UstIdentifierList>
82                  <UstIdentifier Percentage="0.0" Symbol="0"/>
83                  <UstIdentifier Percentage="20.0" Symbol="1"/>
84                  <UstIdentifier Percentage="20.0" Symbol="2"/>
85                  <UstIdentifier Percentage="10.0" Symbol="3"/>
```

```
86                  <UstIdentifier Percentage="20.0" Symbol="4"/>
87                  <UstIdentifier Percentage="20.0" Symbol="5"/>
88              </UstIdentifierList>
89              <ReductionIdentifierList>
90                  <UstIdentifier Percentage="2.0" Symbol="a"/>
91              </ReductionIdentifierList>
92          </txtImportConfig>
93      </importConfig>
94  </ns2:configurationHandler>
```

Listing A.2: config.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java
       .sun.com/xml/ns/persistence/persistence_1_0.xsd">
3
4    <persistence-unit name="InvoiceManager" transaction-type="RESOURCE_LOCAL
       ">
5      <provider>org.hibernate.ejb.HibernatePersistence</provider>
6      <properties>
7        <!-- Configure Database - Username and Password are provided by the
             application at runtime -->
8        <property name="hibernate.connection.driver_class" value="com.mysql.
             jdbc.Driver"/>
9        <property name="hibernate.connection.url" value="jdbc:mysql://
             localhost:3306/invoicemanager"/>
10       <property name="hibernate.cache.provider_class" value="org.hibernate
             .cache.NoCacheProvider"/>
11       <property name="hibernate.connection.isolation" value="java.sql.
             connection.TRANSACTION_READ_COMMITTED"/>
12       <property name="hibernate.dialect" value="org.hibernate.dialect.
             MySQLDialect"/>
13       <!-- SQL stdout logging -->
14       <property name="hibernate.show_sql" value="true"/>
15       <property name="hibernate.format_sql" value="true"/>
16       <property name="use_sql_comments" value="true"/>
17       <!-- connection pooling-settings (3CP0) -->
18       <property name="hibernate.c3p0.min_size" value="5"/>
19       <property name="hibernate.c3p0.max_size" value="20"/>
20       <property name="hibernate.c3p0.timeout" value="300"/>
21       <property name="hibernate.c3p0.max_statements" value="50"/>
22       <property name="hibernate.c3p0.idle_test_period" value="3000"/>
23       <property name="hibernate.connection.autocommit" value="false"/>
24     </properties>
25   </persistence-unit>
26 </persistence>
```

Listing A.3: persistence.xml

A.10. Zusätzliche Dokumente zur Information der Kunden

Im April 2008 wurde damit begonnen, digital signierte Rechnungen zu versenden (siehe „5. Fazit“ auf S. 139). Um die Kunden über die Besonderheiten einer signierten elektronischen Rechnung zu informieren und um ihnen die Prüfung der Rechnungssignatur zu erleichtern, wurden die Dokumente „Wichtige Informationen zu Ihrer elektronischen Rechnung“ und „Signaturprüfung mit dem Adobe Acrobat Reader“ mit den Rechnungen versandt.

Wichtige Informationen zu Ihrer elektronischen Rechnung

- **Warum befindet sich auf meiner elektronischen Rechnung eine digitale Signatur?**

  Alle von uns seit April 2008 versandten elektronischen Rechnungen sind mit einer fortgeschrittenen digitalen Signatur nach dem österreichischen Signaturgesetz versehen.

  Dies gewährleistet laut der „583. Verordnung des Bundesministers für Finanzen“ die Erfüllung der im UStG § 11 Abs 2 genannten Anforderungen an elektronische Rechnungen.

- **Muss ich die digitale Signatur prüfen?**

  Wenn Sie für die von uns auf elektronischem Weg erhaltenen Rechnungen **keinen** Vorsteuerabzug geltend machen möchten, können Sie die digitale Signatur auf Ihrer Rechnung ignorieren (sollten Sie diese als störend empfinden, können wir Ihnen auf Verlangen auch unsignierte elektronische Rechnungen zur Verfügung stellen).

  Sollten Sie als Unternehmen für die von uns erhaltenen Rechnungen vorsteuerabzugsberechtigt sein, müssen Sie die in der PDF-Rechnung enthaltene digitale Signatur prüfen. Dazu empfehlen wir die Verwendung des Acrobat Readers, z. B. in der Version 8.1.2.

- **Was ist eine digitale Signatur?**

  Bei der digitalen Signatur handelt es sich um ein technisches Verfahren, bei welchem ein digitaler Fingerabdruck der Rechnung errechnet wird, welcher dann durch unseren privaten Schlüssel digital unterschrieben wird. Dadurch werden zwei wichtige Punkte sichergestellt:

  1. Sie können sich durch Prüfung der Signatur vergewissern, dass die Rechnung wirklich von uns stammt.
  2. Eine erfolgreiche Signaturprüfung garantiert, dass die Rechnung nach der Signatur nicht mehr verändert wurde.

  Die digitale Signatur auf Ihrer Rechnung wird vom Acrobat Reader geprüft und angezeigt. Die zur Prüfung benötigten Daten werden in Form unseres Zertifikates zur Verfügung gestellt. Dieses ist ebenfalls in der PDF-Rechnung enthalten und wird vom Acrobat Reader zur Prüfung verwendet. Sie müssen lediglich das A-Cert Wurzelzertifikat auf ihrem System installieren und den Acrobat Reader entsprechend konfigurieren (beide Punkte sind im Dokument „Signaturprüfung mit dem Adobe Acrobat Reader.pdf“ beschrieben).

- **Was ist ein Zertifikat?**

  Um die digitale Rechnungssignatur prüfen zu können, werden spezielle Signaturprüfdaten benötigt. Das Zertifikat bestätigt die Bindung dieser Prüfdaten an den Signator.

  Diese Bindung wird von einem Zertifizierungsdiensteanbieter (ZDA) geprüft und dadurch bestätigt, dass das Zertifikat des Signators mit dem Zertifikat des ZDA signiert wird. Dadurch ergibt sich eine „Kette des Vertrauens“ - vertraut man dem Zertifikat des ZDA, so vertraut man den damit signierten Zertifikaten.

  Wir verwenden ein Zertifikat der Firma A-Cert, welches durch deren Wurzelzertifikat signiert wurde. Sie müssen auf ihrem System nur dieses Wurzelzertifikat als vertrauenswürdig einstufen, damit die digitalen Rechnungssignaturen erfolgreich geprüft werden können.

- **Muss ich die elektronische Rechnung archivieren?**

  Wenn Sie für die elektronische Rechnung einen Vorsteuerabzug geltend machen möchten, müssen Sie diese, genauso wie eine Papierrechnung, für mindestens 7 Jahre archivieren.

- **Reicht auch ein Ausdruck der PDF-Rechnung für einen Vorsteuerabzug?**

  Laut den Umsatzsteuerrichtlinien (RZ 1561) können Sie dem Finanzamt als vorläufigen Nachweis einen Ausdruck einer elektronischen Rechnung vorlegen. Zum Vorsteuerabzug berechtigt allerdings nur die elektronische Form der Rechnung, welche auf Verlangen dem Finanzamt auch vorzuweisen ist.

- **Mit welchem Programm kann ich meine elektronische Rechnung ansehen/prüfen?**

  Derzeit versenden wir elektronische Rechnungen im PDF-Format. Zum Betrachten der Rechnung sowie zum Prüfen der Rechnungssignatur empfehlen wir den Adobe Acrobat Reader, welchen Sie kostenlos in der jeweils neuesten Version unter http://www.adobe.com/de/products/acrobat/readstep2.html beziehen können.

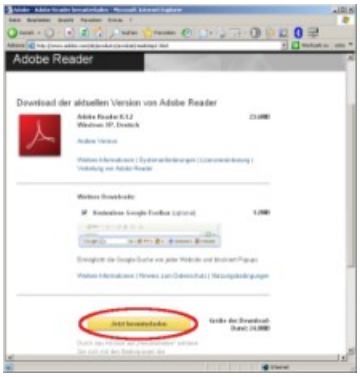

Abbildung 1: Download Adobe Acrobat Reader

Falls dieser noch nicht auf ihrem System vorhanden ist, klicken Sie auf der angegebenen Seite auf „Jetzt herunterladen“ und folgen Sie den dort angegebenen weiteren Anweisungen zur Installation des Acrobat Readers.

Zum Prüfen der Rechnungssignatur muss der Acrobat Reader richtig konfiguriert werden. Dazu haben wir für Sie eine eigene Anleitung erstellt („Signaturprüfung mit dem Adobe Acrobat Reader.pdf“).

Quellen:

- Österreichisches Signaturgesetz; http://www.signatur.rtr.at/de/legal/sigg.html
- 583. Verordnung des Bundesministers für Finanzen, mit der die Anforderungen an eine auf elektronischem Weg übermittelte Rechnung bestimmt werden; http://signatar.at/content/RG/Verordnung.pdf
- Umsatzsteuergesetzt § 11 Abs 2; http://www.steuerberater.at/gesetze/ustg/11/
- Änderung der Umsatzsteuerrichtlinien – Anforderungen an eine auf elektronischem Weg übermittelte Rechnung; http://www.a-sit.at/pdfs/erlass_el_re_legung_UStR-elektronische_Rechnung0705.pdf
- A-Cert; http://www.a-cert.at

AVIA Tankstelle – Shop
Reifen – Waschanlage
Schlapschy

Signaturprüfung mit dem Adobe Acrobat Reader

- **Mit welchem Programm kann ich meine elektronische Rechnung ansehen?**

  Derzeit versenden wir elektronische Rechnungen im PDF-Format. Zum Betrachten der Rechnung sowie zum Prüfen der Rechnungssignatur empfehlen wir den Adobe Acrobat Reader, welchen Sie in der jeweils neuesten Version unter http://www.adobe.com/de/products/acrobat/readstep2.html beziehen können.

  Falls dieser noch nicht auf ihrem System vorhanden ist, folgen Sie dem angegebenen Link und klicken Sie auf der dann angezeigten Webseite (Abbildung 1) auf „Jetzt herunterladen“. Folgen Sie den dort angegebenen weiteren Anweisungen zur Installation des Acrobat Readers.

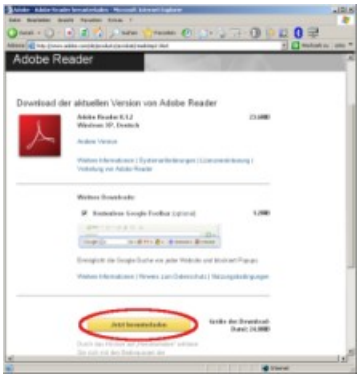

Abbildung 1: Download Acrobat Reader

- **Woran erkenne ich eine gültige Signatur?**

  Wird die Signatur einer unsere Rechnungen im Acrobat Reader geprüft, zeigt dieser das Prüfungsergebnis an zwei Stellen an. Zum einen werden die Signaturdaten im Navigationsfenster „Unterschriften“ angezeigt und zum anderen ist die Signatur direkt auf der letzten Seite Ihrer Rechnung sichtbar (siehe Abbildung 2).

  Eine gültige Signatur wird mit einem grünen Häkchen gekennzeichnet (siehe Abbildung 2), während ein rotes Kreuz eine ungültige Signatur signalisiert (da die Rechnung z. B. nachträglich verändert wurde). Ein gelbes Fragezeichen weist z. B. darauf hin, dass das verwendete Zertifikat abgelaufen ist, oder dass eine ordnungsgemäße Prüfung nicht möglich war. Dies kann z. B. der Fall sein, wenn der Acrobat Reader nicht korrekt konfiguriert wurde. Der genaue Grund ist im Navigationsfenster „Unterschriften“ angegeben oder wird bei einem Klick auf die Signatur im Dokument angezeigt.

  Ein Klick auf die Signatur im Dokument und ein anschließender Klick auf „Unterschriftseigenschaften“ zeigt weitere Informationen zur Signatur. Ein Klick auf „Zertifikat anzeigen“ im Karteireiter „Unterzeichner“ erlaubt Ihnen außerdem eine manuelle Überprüfung unseres Zertifikates.

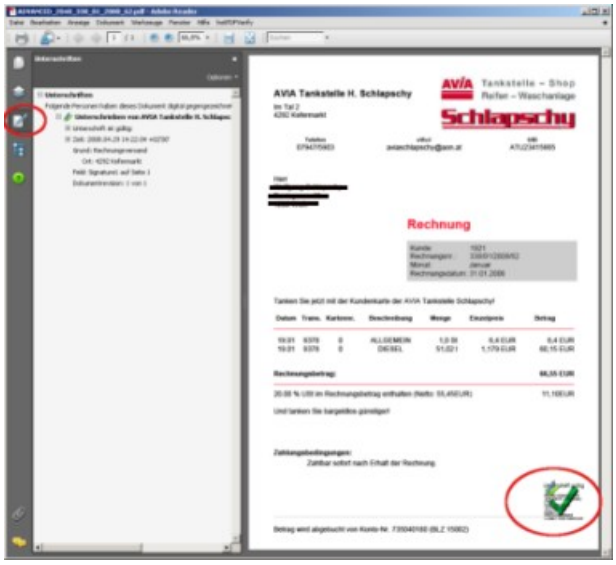

Abbildung 2: eine gültige Signatur im Acrobat Reader

- **Wie kann ich die digitale Signatur auf der Rechnung prüfen?**

  Zur Signaturprüfung der von uns versandten PDF-Rechnungen empfehlen wir den Adobe Acrobat Reader. Sollte dieser noch nicht auf ihrem System vorhanden sein, sollten Sie ihn, wie zu Beginn dieses Dokumentes beschrieben, installieren. Grundsätzlich sollte eine Signaturprüfung mit dem Acrobat Reader ab der Version 6.x möglich sein, die folgenden Ausführungen beziehen sich aber auf die aktuelle Version 8.1.2.

  Zur Signaturprüfung muss der Acrobat Reader **EINMALIG** konfiguriert werden. Diese Konfiguration sollte je nach Betriebssystem unterschiedlich erfolgen. Auf jeden Fall sollten aber zuerst die folgenden beiden Schritte durchgeführt werden:

1. Stellen Sie sicher, dass unter `Bearbeiten – Grundeinstellungen – Sicherheit` der Punkt „Beim Öffnen des Dokuments Unterschriften prüfen“ aktiviert ist. Dann werden Signaturprüfungen beim Öffnen einer Rechnung zukünftig automatisch durchgeführt.

2. Stellen Sie sicher, dass unter `Bearbeiten – Grundeinstellungen – Sicherheit – Erweiterte Grundeinstellungen – Überprüfung` unter „Beim Überprüfen“, wie in Abbildung 3 gezeigt, einer der ersten beiden Punkte ausgewählt ist.

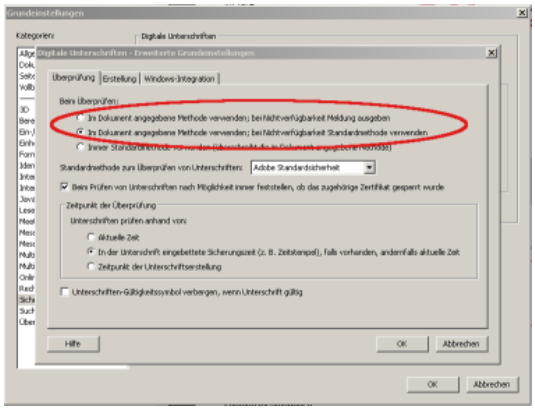

Abbildung 3: Konfiguration der Signaturprüfung

Die nun folgenden Schritte sind je nach Betriebssystem unterschiedlich:

Microsoft Windows (XP oder Vista)

Zur Verwaltung von Zertifikaten verwenden Microsoft-Betriebssysteme einen speziellen Zertifikatsspeicher, welcher auch vom Acrobat Reader verwendet werden kann. Dazu muss zuerst das Wurzelzertifikat der Firma A-Cert in diesen Zertifikatsspeicher installiert werden. Danach muss der Acrobat Reader zur Verwendung dieses Zertifikatsspeichers konfiguriert werden. Beides wird durch die folgenden Schritte durchgeführt:

3. Laden Sie das A-Cert-Wurzelzertifikat von http://www.a-cert.at/static/a-cert-advanced.crt herunter und speichern Sie es an einem beliebigen Ort (bitte verwenden Sie zum Herunterladen des Wurzelzertifikats den Internet Explorer, da z. B. Firefox über eine eigene Zertifikatsverwaltung verfügt und bei Eingabe des Links das Zertifikat in diesem Speicher installieren möchte).

4. Führen Sie einen Doppelklick auf die heruntergeladene Zertifikatsdatei aus. Sollten Sie nun gefragt werden, ob Sie die Datei öffnen möchten, klicken Sie auf den Button „Öffnen“. Danach wird das Zertifikat angezeigt. Zur Installation klicken Sie auf den Button „Zertifikat installieren“ (Abbildung 4).

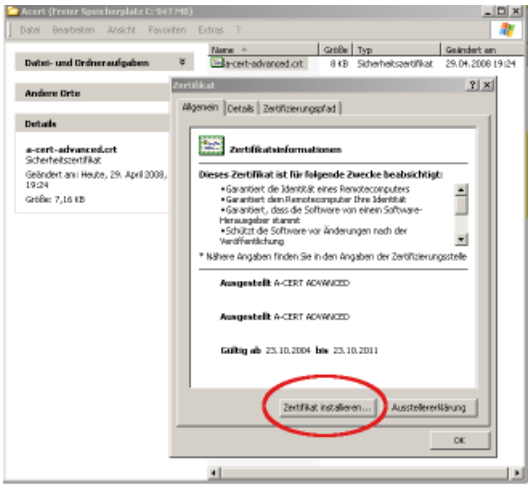

Abbildung 4: Zertifikatsanzeige

5. Klicken Sie im sich nun öffnenden Fenster auf „Weiter“ (Abbildung 5). Lassen Sie im nächsten Fenster den Punkt „Zertifikatsspeicher automatisch auswählen“ selektiert (dies ist auch die Voreinstellung) und klicken Sie auf „Weiter“ (Abbildung 6). Im Folgenden Dialog wählen Sie „Fertig stellen“ (Abbildung 7).

Anmerkung: Nach einer Meldung über den erfolgreichen Import, sollte das A-Cert Wurzelzertifikat nun unter `Systemsteuerung - Internetoptionen - Inhalte - Zertifikate` im Karteireiter „Vertrauenswürdige Stammzertifizierungsstellen“ angezeigt werden.

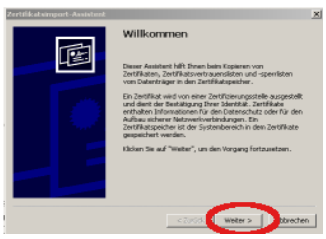

Abbildung 5: Zertifikatsinstallation 1

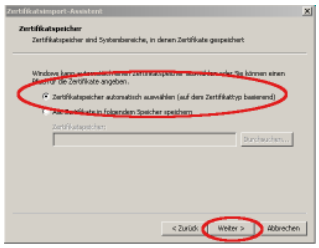

Abbildung 6: Zertifikatsinstallation 2

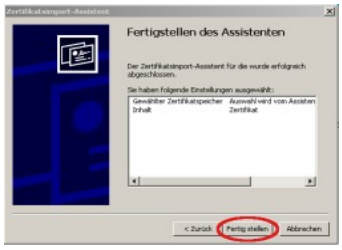

Abbildung 7: Zertifikatsinstallation 3

6. Stellen Sie in `Bearbeiten – Grundeinstellungen – Sicherheit – Erweiterte Grundeinstellungen – Windows-Integration` ein, dass der Acrobat Reader bei der Prüfung auf den Windows-Zertifikatsspeicher zugreifen soll. Dazu müssen, wie auf Abbildung 8 gezeigt, mindestens die beiden obersten Punkte ausgewählt werden. Nach einem Klick auf „OK“ werden die Einstellungen übernommen und die Konfiguration ist abgeschlossen.

   Wenn Sie das nächste Mal eine unserer Rechnungen öffnen, wird die Signatur automatisch geprüft. Eine manuelle Überprüfung können Sie zusätzlich jederzeit über `Dokument – Unterschreiben – Alle Unterschriften prüfen` anstoßen.

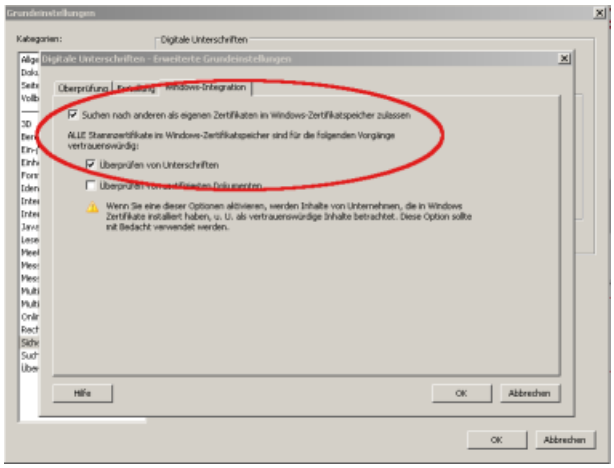

Abbildung 8: Windows-Integration

Andere (Linux, Mac)

Steht der Windows-Zertifikatsspeicher nicht zur Verfügung, oder möchten Sie diesen nicht verwenden, kann der Acrobat Reader die Wurzelzertifikate auch selbst verwalten. Diese Variante kann sowohl mit Microsoft-Betriebssystemen als auch mit anderen Betriebssystemen eingesetzt werden und wird wie folgt konfiguriert.

3. Laden Sie die Zertifikatskette des A-Cert Wurzelzertifikats von http://www.a-cert.at/static/a-cert-advanced.p7b herunter und speichern Sie diese an einem beliebigen Ort.

4. Im Acrobat Reader klicken Sie dann unter `Dokument – Vertrauenswürdige Identitäten verwalten` auf den Button „Kontakt hinzufügen“ (Abbildung 9).

   Im sich dann öffnenden Fenster klicken Sie auf den Button „Durchsuchen“ und wählen die Zertifikatskettendatei aus, welche Sie eben heruntergeladen haben. Mit einem Klick auf „Importieren“ wird das Wurzelzertifikat installiert (Abbildung 10).

   Danach erscheint die Mitteilung, dass ein Zertifikat importiert wurde, welche Sie mit „OK“ bestätigen.

Leitfaden:

Rechnungssignatur und -verifikation mit der Bürgerkarte (e-Card) unter MS Windows

Die Diplomarbeit „e-Billing – Entwicklung und Einführung in der Praxis“, im Zuge welcher dieser Leitfaden geschrieben wurde, beschäftigt sich mit automatisiert weiterverarbeitbaren Rechnungsformaten, Integration in Workflows, Stapelsignaturen und dem Import von Rechnungen aus einem Altsystem. Sollen nur wenige Rechnungen sporadisch versandt werden oder möchte man die Erstellung und Signatur von elektronischen Rechnungen nur einmal ausprobieren, so ist dieser Ansatz wahrscheinlich zu aufwendig. Aus diesem Grund wurde dieser Leitfaden erstellt, welcher im Folgenden zeigt, wie die manuelle Erstellung von einzelnen elektronischen Rechnungen möglichst einfach und kostengünstig durchgeführt werden kann.

Als Rechnungsformat bietet sich hier das PDF-Format an. Rechnungen in diesem Format sind zwar nicht gut automatisiert weiterverarbeitbar, das Format überzeugt jedoch durch eine weite Verbreitung und hohe Akzeptanz.

Ursprünglich war geplant, einen Leitfaden zur kostenlosen Rechnungssignatur unter Verwendung der österreichischen Bürgerkarte zu erstellen. Obwohl die Bürgerkarte wie auch die zur Signatur notwendige Bürgerkartenumgebung für österreichische Bürger kostenlos verwendet werden können, fallen doch einige Kosten, wie z. B. für ein Kartenlesegerät oder eine Signatursoftware, an.

1. Voraussetzungen

1.1. Allgemeines

- **Microsoft Windows**
  Um diese Anleitung möglichst einfach zu halten wird die Signaturerstellung und -verifikation nur für Microsoft Windows Betriebssysteme betrachtet (getestet auf Windows XP Professional).

1.2. Erstellung der Rechnung

- **OpenOffice**
  Liegen die Rechnungen noch nicht als PDF-Dokument vor, sondern müssen diese manuell erstellt werden, bietet sich das freie Textverarbeitungsprogramm „OpenOffice.org Writer“ an, da es Dokumente ins PDF-Format konvertieren kann, ohne dass dazu Zusatzsoftware nötig ist (es kann unter http://de.openoffice.org/ kostenlos bezogen werden). Alternativ dazu kann zur Rechnungserstellung auch ein anderes Programm in Verbindung mit einem PDF-Konverter verwendet werden. Im Folgenden wird aber davon ausgegangen, dass OpenOffice verwendet wird und auf dem System installiert ist.

1.3. Signaturerstellung

- **e-Card**
  Die österreichische e-Card ersetzt nicht nur den Krankenschein sondern kann auch als Bürgerkarte verwendet werden. Dazu kann auf der e-Card seit Anfang 2008 auch ein qualifiziertes A-Trust-Zertifikat gespeichert werden, welches zur Rechnungssignatur verwendet werden kann. Für den Bürger entstehen dadurch keine Kosten.

- **Kartenlesegerät**
  Um mit einem qualifizierten Zertifikat auf einer SmartCard eine qualifizierte Signatur erzeugen zu können, wird ein Kartenlesegerät der Sicherheitsklasse 2 benötigt. Solche Geräte verfügen über ein eigenes PinPad zur Eingabe des PINs. Ein Gerät welches dieses Kriterium erfüllt, ist der „Reiner SCT cyberJack pinpad“, welcher inkl. USt ca. € 50,00 kostet. [1] zeigt eine Auflistung von SmartCard-Lesegeräten, welche von der Firma A-Trust empfohlen werden und bei www.cryptoshop.at erhältlich sind.

  Im Folgenden wird davon ausgegangen, dass ein einsatzbereites Kartenlesegerät mit den notwendigen Treibern installiert ist.

- **Signatursoftware**
  Dies ist der eigentliche Kostenfaktor. Zwar wird zur Signatur mit der Bürgerkarte sowohl von der Firma

BDC [2] als auch von der Firma IT-Solution [3] eine Bürgerkartenumgebung kostenlos zur Verfügung gestellt, diese umfassen aber nur die Signatur über die SecurityLayer-Schnittstelle [4]. Auf http://www.buergerkarte.at/de/pdf-signieren/index.html wird dafür zwar ein kostenloser Dienst zur PDF-Signatur und -verifikation zur Verfügung gestellt, bei den erstellten Signaturen handelt es sich aber um eine sogenannte, nicht zum PDF-Standard kompatible, Amtssignatur (unter [5] ist auch eine Desktop-Applikation zur Erzeugung von Amtssignaturen erhältlich). Solch eine Signatur kann daher nicht vom Acrobat Reader verifiziert werden. Für eine zur PDF-Spezifikation kompatible Signatur mit der e-Card stellen die Firmen BDC und IT-Solution kostenpflichtige Erweiterungen ihrer Bürgerkartenumgebungen bereit:

- BDC: „hotSign eCard und PDF Modul“ für € 144,00 inkl. USt
- IT-Solution: „trustDesk Professional“ für € 108,00 inkl. USt

Beide Anbieter bieten auch eine 30-tägige Testversion ihrer Produkte an. Diese verfügen ebenso über die PDF-Signaturfunktion. In diesem Leitfaden wird die Rechnungssignatur und -verifikation anhand der BDC-Software durchgeführt.

Bedacht werden sollte, dass mit dieser Kombination aus Kartenlesegerät, e-Card und Signatursoftware immer nur eine Rechnungssignatur auf einmal durchgeführt werden kann, was für die Zielgruppe dieses Leitfadens allerdings kein Problem darstellen sollte.

1.4. Signaturverifikation

- **Adobe Acrobat Reader**
  Der Adobe Acrobat Reader kann PDF-Dateien anzeigen und deren Signatur verifizieren. Die neueste Version ist kostenlos unter http://www.adobe.com/de/products/acrobat/readstep2.html erhältlich.

- **BDC: hotPDFVerify-Plugin**
  Der Adobe Acrobat Reader kann ECC-Signaturen (Elliptic Curve Cryptography), wie sie mit dem qualifizierten Zertifikat auf der e-Card erzeugt werden, nicht ohne Hilfe verifizieren. Zu diesem Zweck kann für Signaturen, welche mit dem BDC-Produkt erstellt wurden, ein Plugin für den Acrobat Reader kostenlos von http://www.bdc.at/fileadmin/downloads/hotPDFVerifySetup_v2.03.exe erhalten werden. Dieses Plugin benötigt einen Acrobat Reader der Version 8.x [2].

2. Signatur

2.1. Einmalige Vorbereitung

Bevor Rechnungen signiert werden können, müssen einmalig die folgenden Schritte durchgeführt werden:

1. Schließen Sie das Kartenlesegerät an den Computer an und stellen Sie sicher, dass alle Treiber korrekt installiert wurden. Stecken Sie Ihre e-Card in das Kartenlesegerät.

2. Folgen Sie den Anweisungen auf http://www.buergerkarte.at/de/aktivieren/online.html zur Aktivierung der Bürgerkartenfunktion auf Ihrer e-Card. Falls sie bereits vor 2008 die Bürgerkartenfunktion Ihrer e-Card verwendet haben, müssen Sie zuvor unter https://www.sozialversicherung.at/onlineformulare-e-card-Bestellung/ eine neue e-Card beantragen und mit den folgenden Schritten warten, bis diese zugestellt wird.

   Hinweis: Am einfachsten und schnellsten führen Sie die Vorbereitungen durch, wenn Sie wie unter dem Link beschrieben, den a.sign Bürgerkarteninstaller (https://www.a-trust.at/onlineinstaller/installer.exe) downloaden und bei der Installation den Anweisungen folgen. Dann gehen Sie zum e-Card-Portal der A-Trust (https://www.a-trust.at/e-card), wählen „weiter zur online-Aktivierung“ und folgen den weiteren Anweisungen. Es wird Ihnen dann ein RSa-Brief mit den Aktivierungsdaten zugestellt. Mit einem weiteren Besuch des e-Card-Portals schließen Sie die Aktivierung ab. Alternativ dazu können Sie zur Registrierung Ihrer e-Card auch eine der in https://www.a-trust.at/e-card/rafinder.aspx aufgelisteten Stellen aufsuchen (die e-Card und ein amtlicher Lichtbildausweis sind dazu mitzubringen).

3. Laden Sie die „BDC hotSign“-Signatursoftware (http://www.bdc.at/fileadmin/downloads/hotSignSetup-_V2.0.0_mit_Signaturdrucker.exe) herunter und installieren Sie diese. Stellen Sie sicher, dass während der Installation das Kartenlesegerät angeschlossen und betriebsbereit ist. Installieren Sie dabei auch das „BDC hotPDFVerify“-Plugin (wird automatisch bei der Installation von hotSign zur Installation angeboten).

2.2. Rechnungserstellung und -signatur

1. Schreiben Sie die Rechnung mit dem „OpenOffice.org Writer“.

2. Klicken Sie die Schaltfläche „Direktes Exportieren als PDF“ und erzeugen Sie eine PDF-Rechnung.

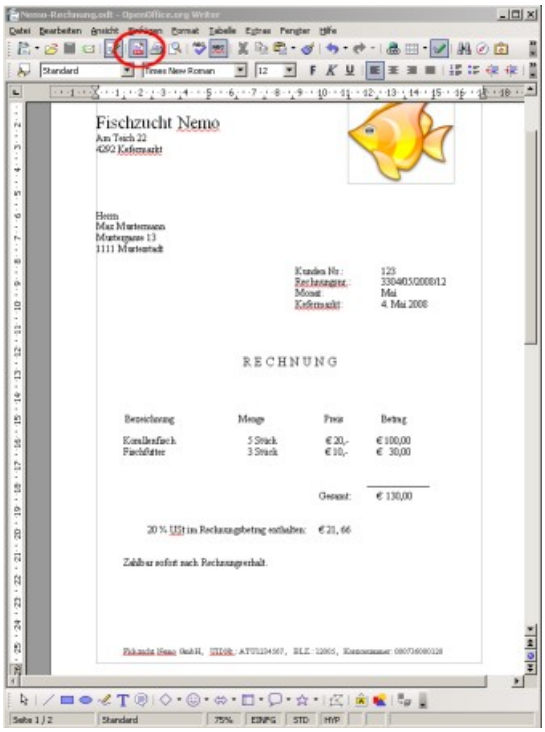

Abbildung 1: Erzeugen der PDF-Rechnung

3. Stellen Sie sicher, dass das Kartenlesegerät angeschlossen ist und, dass die e-Card eingesteckt ist. Starten Sie das Programm „BDC hotSign“. Das Programm ist dann als Icon im SystemTray der Taskleiste sichtbar.

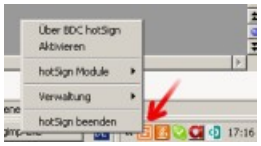

Abbildung 2: BDC hotSign in der Taskleiste

4. Durch einen Rechtsklick öffnet sich das Menü. Stellen Sie einmalig unter `Verwaltung - PDF Einstellungen` ein, dass Sie eine sichtbare Signatur erzeugen möchten. Dazu setzen Sie ein Häkchen in die erste Auswahlbox und geben die Koordinaten für die Signatur an. Wird hier für die X-Koordinate 160 und für die Y-Koordinate 275 angegeben, so sollte die sichtbare Signatur später auf einer A4-Rechnung in der rechten unteren Ecke angezeigt werden. Bestätigen Sie die Änderung mit einem Klick auf „OK“.

5. Wählen Sie nun `hotSign Module - Datei signieren`. Wählen Sie die mit OpenOffice.org erzeugte PDF-Rechnung aus. Stellen Sie außerdem sicher, dass die Option „Sichere digitale Signatur“ ausgewählt ist. Klicken Sie auf „Signieren“.

Abbildung 3: Signieren der Rechnung

Daraufhin wird der „SecureViewer“ geöffnet. Hier können Sie durch einen Klick auf „Dokument anzeigen“ die Rechnung im Adobe Acrobat Reader öffnen. Unter dem Link wird ein Text eingeblendet der darüber informiert, dass PDF-Dokumente, die mit einer qualifizierten Signatur versehen werden sollen, keine dynamisch veränderbaren Inhalte aufweisen dürfen. Die nach dieser Anleitung erzeugte PDF-Rechnung sollte keine solchen Inhalte aufweisen. Sie können dies überprüfen, indem Sie den ebenfalls in diesem Fenster angegebenen Anweisungen folgen. Außerdem können Sie hier angeben, wo die signierte Rechnung gespeichert werden soll.

Abbildung 4: SecureViewer

6. Klicken Sie nun „Daten signieren“. Daraufhin erscheint ein Fenster, welches Sie zur Eingabe Ihres Signatur-PINs auffordert. Nachdem Sie diesen eingegeben und bestätigt haben, sollte eine Meldung die erfolgreiche Signatur der Rechnung anzeigen.

3. Verifikation

3.1. Einmalige Vorbereitung

1. Laden Sie das BDC hotPDFVerify-Plugin von http://www.bdc.at/fileadmin/downloads/hotPDFVerifySetup_v2.03.exe herunter und starten Sie die Installation durch einen Doppelklick.

2. Selektieren Sie im zweiten Fenster die Option „Ja, Stammzertifikate herunterladen“. Klicken Sie „Weiter“ und folgen Sie den Anweisungen. Nach der Installation wird eventuell gefragt, ob Zertifikate installiert werden sollen, was mit „Ja“ zu beantworten ist. Alles weitere wird vom Plugin der Firma BDC erledigt.

3.2. Prüfung einer Rechnung

1. Öffnen Sie die erstellte Rechnung im Adobe Acrobat Reader.

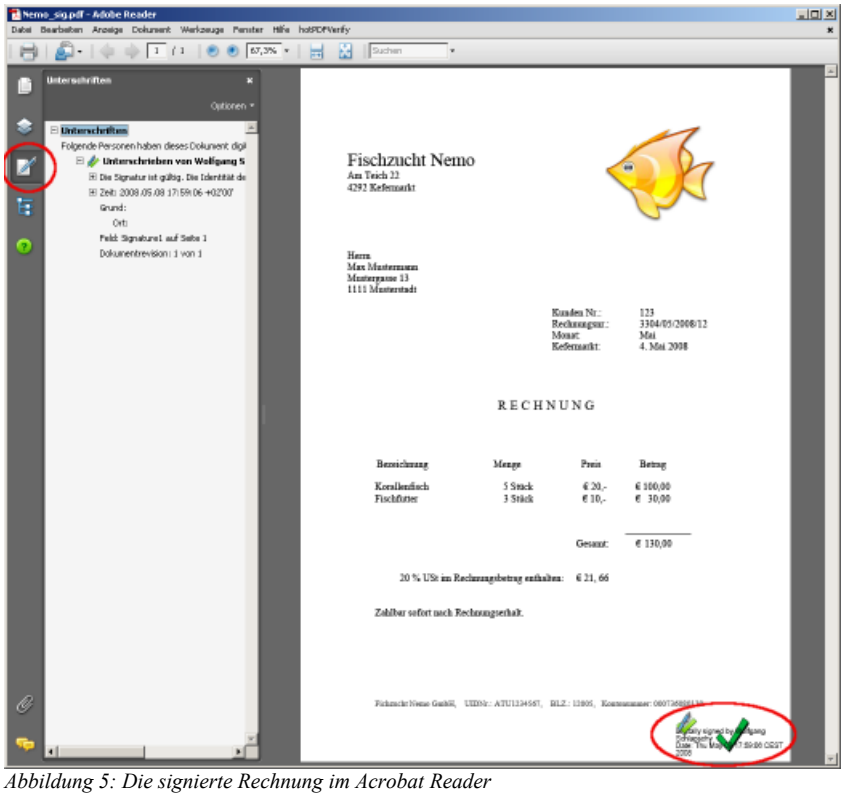

Abbildung 5: Die signierte Rechnung im Acrobat Reader

2. Das Ergebnis der Signaturprüfung wird an zwei Stellen dargestellt. Zum einen werden die Signaturdaten im Navigationsfenster „Unterschriften“ angezeigt und zum anderen ist die Signatur direkt auf der Rechnung sichtbar (siehe Abbildung 5).

   Eine gültige Signatur wird mit einem grünen Häkchen gekennzeichnet (siehe Abbildung 2), während ein rotes Kreuz eine ungültige Signatur signalisiert (da die Rechnung z. B. nachträglich verändert wurde). Ein gelbes Fragezeichen weist z. B. darauf hin, dass das verwendete Zertifikat abgelaufen ist oder, dass eine ordnungsgemäße Prüfung nicht möglich war.

   Ein Klick auf die Signatur im Dokument und ein anschließender Klick auf „Unterschriftseigenschaften“ zeigt weitere Informationen zur Signatur. Ein Klick auf „Zertifikat anzeigen“ im Karteireiter „Unterzeichner“ erlaubt die manuelle Überprüfung des Zertifikates.

Quellen:

[1] Cryptoshop; A-Trust empfohlene SmartCard Reader; http://www.cryptoshop.com/de/products/reader/atrustreader.php;

[2] BDC EDC Consulting GmbH; http://www.bdc.at

[3] IT-Solution GmbH; http://www.itsolution.at

[4] Die österreichische Bürgerkarte – Applikationsschnittstelle Security-Layer;
http://www.buergerkarte.at/konzept/securitylayer/spezifikation/aktuell/core/Core.html

[5] EGIZ; PDF-Signatur; https://demo.egiz.gv.at/plain/projekte/signatur_im_e_government/pdf_signatur

[6] Bundeskanzleramt Österreich; FAQ zur Amtssignatur; http://www.cio.gv.at/faq/Amtssignatur/

A.12. Curriculum Vitae

Persönliche Angaben

Name:	Wolfgang Schlapschy, Bakk.techn.
Geburtsdatum:	15. November 1980
Staatsbürgerschaft:	Österreich

Kontakt:	Wolfgang Schlapschy Denisgasse 18/5 1200 Wien

Tel.:	0043 / 650 / 9020606
E-Mail:	wolfgang.schlapschy@aon.at

Ausbildung

Universität

2001 - 2006	Bakkalaureatsstudium der Informatik an der Johannes Kepler Universität Linz
09/2006 - 01/2007	Auslandssemester an der Dublin City University (DCU) in Irland
seit 2007	Magisterstudium der Informatik an der Johannes Kepler Universität Linz

Schule

1995 - 2000	5 Kl. Handelsakademie in Freistadt (Abschluss mit Matura im Jahr 2000)
1991 - 1995	4 Kl. Privathauptschule „Marianum“ in Freistadt

Kenntnisse & Fähigkeiten

Sprachen

- Deutsch (Muttersprache)
- Englisch (fließend)

- Französisch (fortgeschritten)

Sonstiges

- Führerscheine der Klassen A, B, C, E

Berufserfahrung

Auslandserfahrung

03/2005 - 09/2005	France Telecom R&D in Paris (Frankreich) Entwicklung von Sicherheitsprototypen für Pocket PCs
07/2006 - 08/2006	Vivendi Universal Games in Dublin (Irland) Softwaretests (Computerspiele)

Projektarbeiten und Ferialpraktika

12/2005 - 05/2006	Fabasoft Institute of Technology Bakkalaureatsarbeit: Entwicklung eines Prototypen zur digitalen Signatur in Webapplikationen
06/2004 - 08/2004	Donau-Touristik (Reiseveranstalter) Ferialarbeit im Bereich Reisegruppenbetreuung
07/2003 - 08/2003	LinzNet (Internetserviceprovider) Installation, Administration und Wartung von Servern, Workstations und Funkantennen
07/2002	AVIA Schlapschy (Tankstelle mit Buffet) Service, Tagesabrechnung
10/1999 - 05/2000	KFZ – Lindner (Autohändler) Maturaprojekt: Erstellen eines EDV-Verwaltungsprogrammes

Sonstiges

seit 2003	Ing. L. Putschögl Bauges m.b.H Betreuung der EDV-Anlage
seit 2001	AVIA Schlapschy (Tankstelle mit Buffet) Betreuung der EDV-Anlage

Präsenzdienst

10/2000 - 05/2001	Kraftfahrer in Langenlebarn (NÖ)

A.13. Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplom- bzw. Magisterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Des weiteren versichere ich, dass ich diese Diplom- bzw. Magisterarbeit weder im Innoch im Ausland in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

Wien, 29. Mai 2008

Wolfgang Schlapschy