Dokumentation der betrieblichen Projektarbeit

PDF - Parser & Splitter

Entwicklung einer Anwendung zur Unterstützung
einer automatisierten Rechnungsstellung

Fachinformatiker für Anwendungsentwicklung
Abschlussprüfung Sommer 2004

Thomas Roschinsky
Elsterweg 3
15859 Kehrigk

Tel.: 030 / 420 89 290
Handy: 0170 / 70 20 489
E-Mail: t.roschinsky@cbxnet.de

Azubi-Ident-Nummer: 107/2854868
Prüfungsausschuss: FIAN12

Projektbetreuer:
Herr Robert Senger
E-Mail: r.senger@cbxnet.de

CBXNET com.box internet service gmbh
Potsdamer Straße 96
10785 Berlin

www.cbxnet.de

Stand: 19.05.2004

Thomas Roschinsky
19.05.2004

Inhaltsverzeichnis

Thomas Roschinsky
19.05.2004

1 Projektbeschreibung

1.1 Problemstellung

Die Firma CBXNET com.box internet service gmbh (nachfolgend CBXNET genannt) ist ein Internet Service Provider. Die Rechnungsstellung erfolgt durch das Warenwirtschaftssystem, welches am Monatsende die Rechnungen aller Kunden auf einem Drucker ausgibt.

Aus Gründen der Kostenersparnis durch Automatisierung der regulären Abläufe möchte die Firma CBXNET die Rechnungen an die Kunden in elektronischer Form per E-Mail versenden. Der Versand der Rechnungen soll im Format PDF realisiert werden, indem das Warenwirtschaftssystem die Rechnungen nicht mehr auf dem Standarddrucker, sondern auf dem Druckmodul des Adobe® Acrobat PDF Writers ausgibt. So wird eine PDF-Datei mit allen Rechnungen erzeugt.
Die Möglichkeit, einzelne Rechnungen als PDF-Dokument mit zugeordneten Kundennummern auszugeben bietet das Warenwirtschaftssystem nicht.

Es soll die jeweils zu dem Kunden gehörende Rechnung im PDF Format von einem bereits existierendem Mail-Script als E-Mail Anhang versendet werden.

1.2 Betriebliches Umfeld

Das normale Geschäftsfeld der Firma CBXNET ist der Vertrieb von Internetanbindungen sowie Webhosting und Webprogrammierung vorwiegend für Businesskunden.
Die Erschließung der Zielgruppe der privaten Endkunden ist in Form von WLAN-Produkten in Planung, wobei jedoch die Rechnungsstellung auf digitalem Wege erforderlich wird.

Die Entwicklung der internen Projekte wird in der Firma CBXNET weitestgehend von eigenen Mitarbeitern ausgeführt.
Das Projekt ist ein firmeninterner Auftrag und bedarf aus diesem Grund keiner kundenorientierten Auftragsannahme und Abrechnung.
Die Auftragsannahme erfolgt durch eine Besprechung mit dem Projektbetreuer und dem Entwickler des Automatisierungssystems.

1.3 Technische Voraussetzungen

Die Zielplattform für die Anwendung ist vorerst ein Microsoft ™ Windows 2000 Advanced Server, da die Extraktion der PDF-Dokumente auf dem Rechner mit dem Warenwirtschaftssystem der Firma CBXNET stattfinden soll.
Im zweiten Schritt, der zeitlich noch nicht feststeht, soll das Warenwirtschaftssystem zusammen mit der Anwendung "PDF - Parser & Splitter" auf einem Linux Redhat Server laufen.

Aufgrund der verschiedenen Zielplattformen muss die Anwendung betriebssystemunabhängig entwickelt werden, so dass die Programmiersprache JAVA verwendet wird.
Die Entwicklung erfolgt auf einem Microsoft ™ Windows 2000 Professional System mit Hilfe der freien Entwicklungsumgebung Eclipse.

Um Zeit bei der Entwicklung zu sparen, soll eine bereits existierende Möglichkeit zur Be- und Verarbeitung der PDF-Dokumente (z. B. als Open-Source- oder Freeware-Bibliothek) gewählt werden.

1.4 Grundlegende Überlegungen und Aufgabenstellung

Wie aus der Problembeschreibung hervorgeht, ist ein Zwischenschritt notwendig, der aus dem PDF-Dokument mit allen Rechnungen die Einzelrechnungen extrahiert und sie zuordnungsfähig durch Verwendung der Kundennummer im Dateinamen macht.

Zur Realisierung dieser Funktionalität muss die Anwendung "PDF - Parser & Splitter" entwickelt werden, die im betrieblichen Umfeld auf einem Microsoft System und später auf einem Linux System lauffähig sein soll.
Die Anwendung wird das Rechnungsdokument des Warenwirtschaftssystems mit allen Rechnungen eines Monats nach vorgegebenen Kriterien (z. B. Nach Kundennummer) durchsuchen, die Einzelseiten der Rechnung eines Kunden extrahieren und dem Kunden zugeordnet, versandfertig als PDF-Dokument abspeichern.

2 Dokumentation

2.1 Grobkonzept

Im Grobkonzept sind die Unterpunkte der Ist-Analyse sowie des Soll-Konzepts enthalten. Mit der Ist-Analyse sollen vor der Ausführung der Arbeit am Projekt Schwachstellen aufgedeckt werden sowie Umwege und Fehler in der Entwicklung vermieden werden.
Im Soll-Konzept werden für die Problemstellung und die erkannten Schwachstellen Lösungsansätze angeführt.

2.1.1 Ist-Analyse

Nach entsprechender Konfiguration des Warenwirtschaftssystems der Firma CBXNET, wird bei dem monatlichen Rechnungslauf ein Adobe® PDF-Dokument erstellt, in welchem sämtliche Rechnungen enthalten sind. Es handelt sich um ein Rechnungsaufkommen von ca. 500 Exemplaren.

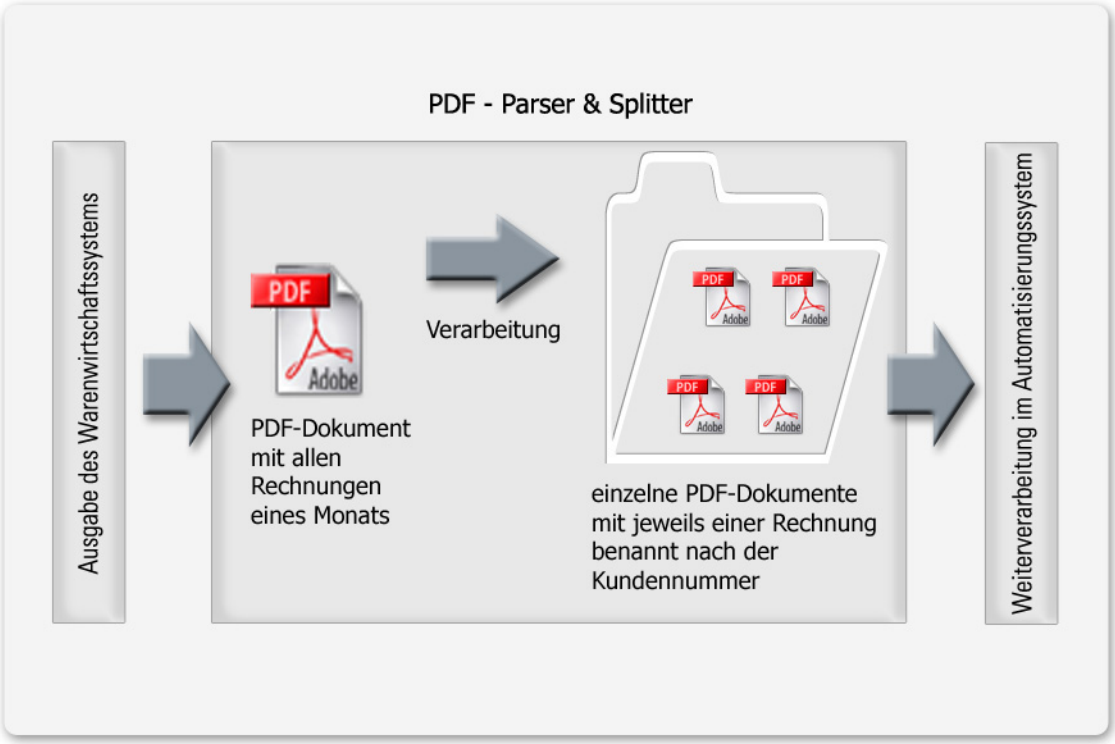
Dieses PDF-Dokument liegt als Eingangsformat für das Projekt vor. Im Dokument sind die kundenspezifischen Angaben wie Adresse, Name, Kundennummer, die Rechnungsnummer und der Rechnungstext enthalten.

Das bereits existente Automatisierungssystem (das unter dem Punkt Problemstellung als "Mail-Script" angeführt wurde) kann Verzeichnisse durchsuchen und anhand von Kundennummern im Dateinamen die einzelnen Rechnungsdokumente den Endkunden zuordnen.

2.1.2 Soll-Konzept

Die Anforderung besteht darin, aus dem PDF-Dokument mit allen Rechnungen eines Monats Einzelrechnungen im PDF-Format zu erstellen, die von dem Automatisierungssystem an den Kunden per E-Mail versand werden sollen.

Voraussetzung für den automatisierten Versand ist, dass die generierten Dateien zwecks Zuordnung von Kunde und Rechnung im Dateinamen die Kundennummer des Kunden enthalten, für den die Rechnung bestimmt ist.
Die Kundennummer ist im Eingangsformat enthalten und muss aus dem Dokument gefiltert werden, um beim Abspeichern für den Dateinamen zur Verfügung zu stehen.



Eine Benutzeroberfläche ist nicht erforderlich, da die Anwendung in den Automatisierungs-prozess eingebunden werden soll und zeitgesteuert vom System bzw. nach erfolgreichem Rechnungslauf auszuführen ist.

Für die kaufmännische Überprüfung ist eine optionale Ausgabe der verarbeiteten Rechnungen zur Laufzeit oder in eine Logdatei zu implementieren. Nach Rücksprache mit dem Projektbetreuer ist vorerst eine umfassende Ausgabe der verarbeiteten Daten an der Kommandozeile gewünscht.

Die Anwendung soll mit der Sprache JAVA und einer externen Open-Source-Bibliothek zur Verarbeitung des Adobe® PDF Formates realisiert werden.
Als Open-Source-Bibliothek wurde das Projekt PDF-Box von Ben Litchfield ausgewählt, das unter der BSD-Lizenz veröffentlicht ist.

2.2 Feinkonzept

Im Feinkonzept stellen Systementwurf und Programmentwurf einen Leitfaden für die Entwicklung der Anwendung dar.

2.2.1 Systementwurf

Folgende Operationen und Funktionen muss die Anwendung erfüllen:
- Datei- und Verzeichnisoperationen:
    - Öffnen der Eingangsdatei
        - Übergabe der Datei erfolgt per Kommandozeile
    - Schreiben der Ausgabedateien in ein festgelegtes Ausgabeverzeichnis
        - automatisches Generieren nach Datum und laufender Nummer
        - alternative Angabe des Pfades erfolgt per Kommandozeile
    - Anlegen eines Ausgabeverzeichnisses bei nicht Vorhandensein
- Funktionen:
    - Interpretieren von PDF-Dokumenten
    - Durchsuchen von PDF-Dokumenten
    - Schreiben der PDF-Dokumentstruktur

Folgende Benutzer-Anforderungen muss die Anwendung erfüllen:
- Übergabe von Parametern zur Steuerung:
    - Quell PDF-Datei
    - Ausgabeverzeichnis
    - Vorlage PDF-Datei (hier wird ein leeres PDF-Dokument verwendet)
      Ermöglicht später die Verwendung von PDF-Vorlagen.
    - Passwort (für verschlüsselte PDF-Dokumente)
    - Ausgabe der Protokollierung (Informationen der Verarbeitung) zur Laufzeit

2.2.2 Programmentwurf

Der Ablauf der Anwendung gliedert sich in die Punkte:
- Überprüfung der Parameter
- Einlesen der Quelldatei
- Generieren des Ausgabepfades mit aktuellem Zeitstempel
- Verarbeiten der Quelldatei (jeweils seitenweise bis Ende erreicht)
    - Finden des Suchstrings (Kundennummer)
    - Generieren des Dateinamens mit dem Suchstring
    - Generieren der Einzelseite (Zusammenfügen mit einer leeren PDF-Vorlage)
    - Speichern der Einzelseite unter dem generierten Dateinamen
    - Ggf. Ausgabe der zusätzlichen Informationen
- Ausgabe einer abschließenden Meldung

2.3 Pflichtenheft

Das Pflichtenheft liegt der Dokumentation als Anlage bei (Anlage 1)

3 Realisierung des Projekts

3.1 Entwicklungs- und Systemanforderungen

- Notwendige Software zur Umsetzung des Projekts:
  - Sun JDK in der Version 1.4.1
  - Entwicklungsumgebung Eclipse in der Version 2.1.2
  - Open-Source-Bibliothek PDFBox von Ben Litchfield in der Version 0.6.4
  - Muster-Exemplar der Rechnungsdatei im Eingangsformat (PDF)
  - Adobe® Acrobat™
- Anforderungen zum Betrieb der Anwendung:
  - ein Betriebssystem für welches eine JAVA VM erhältlich ist
  - JAVA VM 1.4

3.2 Entwicklung der Extraktionsfunktion

Es gilt die Funktionen des Programmentwurfs und des Systementwurfs effektiv umzusetzen sowie die Anforderungen der Soll-Analyse zu berücksichtigen.

Durch die technisch betrachtet nicht zeitkritische Anwendung kann die gesamte Funktionalität innerhalb einer Klasse realisiert werden. Die Klasse wird verkürzt nach dem Titel der Anwendung benannt: *PDFParSpl*.

Damit die Klasse *PDFParSpl* auf die Funktionen der Open-Source-Bibliothek PDFBox zugreifen kann, werden die benötigten Klassen im Programm per import-Anweisung eingebunden. Im Quelltext sind die wichtigsten Import-Anweisungen mit deren Funktion kommentiert. Zusätzlich werden ebenfalls per Import-Anweisung benötigte Java-Grundfunktionen angegeben.

Bei der Betrachtung der im Programmentwurf festgehaltenen Schritte, werden nur die Wichtigsten erläutert.

3.2.1 Parameterübergabe / Aufruf

Das Programm kann alle im Systementwurf angegebenen Parameter verarbeiten. Der Syntax ist wie folgt anzugeben:

PDFParSpl [OPTIONS] <Quell PDF-Datei> [<Ausgabe Ordner>]

-template	<PDF-Vorlage>	Angabe der PDF-Vorlage mit Pfad
-passwort	<Passwort>	Angabe des Passworts, falls das Dokument verschlüsselt ist
-lzausgabe		Ausgabe der Protokollierung zur Laufzeit

Die Verwendung der Optionen und des Ausgabeordners sind optional. Sollte die Option *-template* nicht verwendet werden, wird im Ausführungsordner nach der Datei *template.pdf* gesucht. Diese Datei wird zur Ausführung benötigt!

3.2.2 Einlesen der Quelldatei

Mittels Dateistrom (java.io.FileInputStream) wird die Quell-Datei eingelesen. Der Strom wird mit dem PDFParser (org.pdfbox.pdfparser.PDFParser) in ein COS-Dokument umgewandelt, welches daraufhin der Anwendung als Objekt zur Verfügung steht und damit die Grundlage für die Bearbeitung des PDF-Dokument Standards bildet.

3.2.3 Suchstring finden / Dateinamen generieren

Das COS-Dokument wird Seite für Seite in reinen Text umgewandelt. So kann die Such-funktion die jeweilige Seite nach dem Suchstring (der Kundennummer) durchsuchen und die gefundene Kundennummer zu Weiterverarbeitung im Dateinamen zwischenspeichern.
Der Dateiname wird mit einem Präfix (CBXNETRechnung_) versehen und ergänzend mit der Kundennummer aufgefüllt. Sollte die Rechnung bzw. der Dateiname schon existieren, wird zusätzlich eine laufende Nummer mit dem Präfix (_p) vergeben.

3.2.4 Generieren der Einzelseite

Um die Einzelseite zu generieren, werden insgesamt drei Methoden verwendet, welche selbst in der Klasse *PDFParSpl* implementiert sind:
- ***processPDF***: Zusammenfügen von zwei PFD-Dokumenten und schreiben der Ausgabe-Datei (ruft die Methode *buildDocument* auf)
- ***buildDocument***: Verknüpfung auf Ebene der COS-Dokumentstruktur (ruft die Methode *buildPage* auf)
- ***buildPage***: Zusammenfügen der Seitenobjekte

Dieser Schritt ist für das Verarbeiten der PDF-Dokumente auf physischer Ebene (Schreiben der Ausgabedatei) und auf logischer Ebene (Bearbeiten der Dokumente auf Ebene der COS-Dokumentstruktur) zuständig.

3.2.5 Ausgaben im Programmverlauf

Die Ausgaben während des Programms sind zur Kontrolle der Anwendung implementiert. Mit der Option (Schalter an der Kommandozeile) "*-lzausgabe*" kann eine detaillierte Darstellung der Verarbeitung erreicht werden, die unter anderem den gesamten Text des Dokuments enthält.

3.3 Test der Anwendung

Entwicklertest:
Während der Implementierung wurde die Anwendung jeweils nach dem Kompilieren mit einer Testdatei geprüft, die sechs Rechnungen enthielt.
Dabei wurde speziell geprüft:
- Das Verhalten bei Tests mit den verschiedenen Kombinationen der Übergabeparameter
- Das Verhalten auf falsche Übergabeparameter

Die hier gefundenen Fehler wurden direkt berücksichtigt und korrigiert.

Systemtest:
Nach Abschluss der Implementierung wurde die Anwendung mit sechs verschiedenen Testdateien getestet, welche 6, 70 & 600 Rechnungen enthielten und mit verschiedenen PDF-Generatoren erzeugt wurden. Der Test wurde im Beisein des Projektbetreuers durchgeführt. Dabei wurden mit Hilfe einer Stapelverarbeitungsdatei getestet:

- Verhalten bei Tests mit den verschiedenen Kombinationen der Übergabeparameter
- Verhalten auf falsche Übergabeparameter
- Verhalten auf zwei verschiedenen Plattformen
- Verhalten unter zwei verschiedenen JAVA VMs

Die hier gefundenen Fehler wurden berücksichtigt und zur Änderung und Optimierung der Anwendung verwendet.

3.4 Zeitlicher Verlauf des Projekts

Mit der Bearbeitung des Projekts wurde auf Grund einer Verzögerung der Einführung des internen Automatisierungssystems nicht wie im IHK-Antrag angegeben am 05.04.2004 begonnen. Das Projekt wurde im Zeitraum vom 13.04. bis 19.05.2004 bearbeitet.

Datum	Tätigkeit	Zeitaufwand
	Erfassen des Auftrags und Planung	
13.04.2004	Gespräch mit dem Projektbetreuer	2h
13.04.2004	Planung des Projekts	3h
14.04.2004	Erstellung des Pflichtenheft	5h
	Einarbeitungsphase	
15.04.2004	Einarbeitung in den Aufbau einer PDF-Datei [1]	6h
16.04.2004	Einarbeitung in die Funktionen der Bibliothek PDFBox [3]	4h
26.04.2004	Einarbeitung in die Funktionen der Bibliothek PDFBox [3]	6h
	Realisierung der Anwendung	
05.05.2004	Erstellen der Dokumentation	1h
06.05.2004	Erstellen der Dokumentation	1h
10.05.2004	Implementierung / Programmierung	8h
11.05.2004	Implementierung / Programmierung	6h
12.05.2004	Erstellen der Dokumentation	6h
13.05.2004	Erstellen der Dokumentation	4h
	Test der Anwendung	
14.05.2004	Durchführung des Systemtests mit Testdateien	7h
17.05.2004	Beheben von aufgetretenen Problemen	6h
19.05.2004	Aktualisierung der Dokumentation	0,5h
	Abnahme des Projekts	
19.05.2004	Einweisung des Programmierers der Automatisierungssoftware	1h
19.05.2004	Übergabe der Anwendung und der Dokumentation an den Projektbetreuer, Durchführen der Funktionskontrolle sowie Unterzeichnung des Abnahmeprotokolls	1,5h

Von der Planung im Projektantrag wurde außer in den Punkten "Implementierung" und "Erstellen der Dokumentation" nur geringfügig abgewichen.
Im Punkt der "Implementierung" trat die Abweichung auf, weil der Aufwand für die Implementierung nicht adäquat abgeschätzt werden konnte. Die Funktionsweise der PDF-Bearbeitung mit PDFBox zur Erstellung des Projektantrags war zu dem Zeitpunkt nicht hinreichend bekannt.

Es wurde ein Gesamtzeitaufwand von 68 Stunden benötigt.

4 Qualitätssicherung

Um die notwendige Qualitätssicherung zu gewährleisten, wurden die unter Punkt 3.3 "Test der Anwendung" Tests mit größter Sorgfalt und Gewissenhaftigkeit durchgeführt.
In Regelmäßigen Abständen wurde dem Projektbetreuer eine Version des Quelltextes übergeben, welcher unabhängig getestet wurde.

Bei der Entwicklung wurde streng auf die Vereinbarungen mit dem Entwickler der Automatisierungssoftware geachtet.

5 Ergebnisbetrachtung

Die Anforderungen wurden erfüllt und das Projekt wurde erfolgreich abgenommen.
Für eine Weiterführung durch Dritte wurde der Quelltext ausreichend kommentiert und der Entwickler des Automatisierungssystems eingewiesen.

Anhand der Arbeitszeit von 68 Stunden ergeben sich auf Grundlage des Gehaltes für einen Fachinformatiker, Fachrichtung Anwendungsentwicklung, in der Ausbildung die Fertigungskosten für die Entwicklung der Anwendung von ca. 290,- Euro.
Zuzüglich des Materialkostenzuschlagssatzes für die Benutzung des Entwicklungsrechners, für Ausdrucke von Dokumentationen sowie die Internetnutzung liegen die reinen Entwicklungskosten bei ca. 320,- EUR.

Da die Entwicklung firmenintern verwendet wird, werden keine weiteren Zuschläge wie z. B. Gewinnspanne erhoben. Es sind keine zusätzlichen kundenspezifischen Kosten entstanden.

Die Nutzung dieser Entwicklung stellt für die Firma CBXNET eine finanzielle und personelle Einsparung der laufenden Kosten dar.

Der Nutzen der Anwendung wird deutlich, wenn die Kosten des bisherigen Rechnungsversands (Papier + Druckernutzung + Umschlag + Porto + Arbeitszeit des Kuvertierens + Arbeitsaufwand des Versands) gegen die Kosten des Versands per E-Mail und die Entwicklungskosten gegenübergestellt werden.
Diese Kosten können nicht in Euro dargestellt werden, da die Kostenstruktur im Unternehmen nicht erfasst vorliegt.

6 Glossar

BSD-Lizenz:
Unter der BSD-Lizenz versteht man eine Open-Source-Lizenz, die die freie Verwendung der Lizenz unterliegenden Quellen gestattet, solange das Urheberrecht des Autors gewahrt bleibt bzw. der Hinweis auf die Urheberrechte nicht entfernt werden.

COS-Dokumentstruktur:
Die COS-Dokumentstruktur ist die unterste Schicht eines PDF-Dokuments. Auf diese "Low-Level"-Ebene setzen die PDF-internen Elemente wie das Dictionary-Objekt oder das Pages-Objekt auf, die zur Verarbeitung und Erstellung von PDF-Dokumenten benötigt werden.

JDK:
Das JAVA Developmentkit ist eine Distribution der JAVA VM, welche speziell für Entwickler zugeschnitten ist. Es enthält zahlreiche Hilfsmittel zur Entwicklung von JAVA-Anwendungen.

JVM:
Die JAVA Virtual Machine wird zur Ausführung einer JAVA-Anwendung benötigt. Sie enthält unter anderem den Interpreter, der den Bytecode einer JAVA-Anwendung in den spezifischen Maschinencode der Zielplattform übersetzt und ausführt.

Open-Source:
Open-Source (zu Deutsch: offene Quelle) bezeichnet Software, welche mit frei verfügbarem Quelltext angeboten wird. Meist unterliegt Open-Source eine der dafür bestimmten Lizenzierungsarten, die die Urheberrechte schützen.

PDF-Dokument:
PDF (Portable Document Format) ist ein von Adobe® entwickeltes Dateiformat, welches auf unterschiedlichsten Systemen zu gleichen Ergebnissen in der Betrachtung führt. Es wird dafür eine spezielle Software benötigt, die für viele verschiedene Zielplattformen von Adobe frei erhältlich ist (Adobe® Acrobat Reader).

String:
Ein String ist ein Datentyp, der Zeichenketten speichert und in vielen Programmiersprachen verstanden wird.

VM:
Siehe JVM.

7 Quellenangabe

[1] Meehan, J.; Taft, E.; Chernicoff, S.; u. a.: PDF Reference (fourth edition) Adobe® Portable Document Format - Version 1.5., August 2003
http://partners.adobe.com/asn/acrobat/sdk/public/docs/PDFReference15_v5.pdf

[2] Steyer, R.: Java 2 – Das Programmierhandbuch, Heyne Verlag Juli 2001, ISBN 3-453-19559-0

[3] Litchfield, B.: Documentation PDFBox (JavaDoc)
http://www.csh.rit.edu/~ben/projects/pdfbox/javadoc/index.html

[4] Java 2 Platform, Standard Edition, v 1.4.0 - API Specification

8 Ressourcenangabe

Folgende Dateien stehen zusätzlich zum Download bereit:

Die Projektdokumentation (dieses Dokument):
http://www.tommy879.net/Main/menu2/download/tro_projekt_pdfparspl_doku.pdf
Größe: 276 KB

Der vollständige Quelltext:
http://www.tommy879.net/Main/menu2/download/tro_projekt_pdfparspl_src.java
Größe: 15 KB

Die verwendete Version von PDFBox von Ben Litchfield:
http://www.tommy879.net/Main/menu2/download/PDFBox064.zip
Größe: 3.798 KB

Anlage 1

Pflichtenheft

PDF - Parser & Splitter

Stand: 19.05.2004

1 Zielstellung

1.1 Muss-Anforderungen

Die primäre Anforderung ist die hundertprozentige Funktion, ohne Bedienungsarbeiten und Wartungsarbeiten erforderlich zu machen.
Die Anwendung soll den Zwischenschritt der PDF-Extraktion aus dem Hauptrechnungs-dokument in kundenzugeordnete Einzelrechnungsdokumente im PDF-Format gestalten.
Die Kundenzuordnung soll über den Dateinamen erfolgen, in dem die Kundennummer integriert ist.

1.2 Wunsch-Anforderungen

Zusätzlich sind die optionale Ausgabe einer detaillierten Aufstellung der verarbeiteten Dokumente mit dem jeweiligen Rechnungstext und die Möglichkeit der Verarbeitung von verschlüsselten PDF-Dokumenten erwünscht.
Außerdem soll für die in Zukunft geplante grafische Gestaltung von Rechnungsdokumenten die Verwendung einer Rechnungsvorlage bei der Entwicklung bedacht werden.

2 Produkteinsatz

2.1 Anwendungsbereiche

Die Anwendung soll den Ablauf der automatisierten Rechnungserstellung im laufenden Betrieb unterstützen.

2.2 Zielgruppe

Die Zielgruppe begrenzt sich ausschließlich auf die Mitarbeiter der CBXNET, welche an der Rechnungsstellung direkt beteiligt sind. Sobald die Anwendung vollständig in den Automatisierungsprozess integriert ist, ist kein Benutzer mehr erforderlich.
Vorhaben, das Produkt Endkunden anzubieten, sind nicht geplant.

3 Produktumgebung

3.1 Software

- ein Betriebssystem für welches eine JAVA VM erhältlich ist
- JAVA VM 1.4

Die Zielplattform für die Anwendung ist vorerst ein Microsoft ™ Windows 2000 Advanced Server. Im zweiten Schritt, der zeitlich noch nicht feststeht, soll die Anwendung "PDF - Parser & Splitter" auf einem Linux Redhat Server laufen.

3.2 Hardware

Hardwareanforderungen sind nur in sofern gegeben, dass die unter Punkt 3.1 angeführte Software lauffähig sein muss.
Ein freier Festplattenspeicherplatz zur Ausführung von 500 MB wird auf Grund der großen Quelldateien empfohlen.

3.4 Produkt-Schnittstellen

Eingangsschnittstelle:
Die Übergabe der Quelldatei und der alternativen Parameter erfolgt per Kommandozeilenoption.

Ausgangsschnittstelle:
Das Vorhandensein der Einzelrechnungen im PDF-Format mit Kundennummer im Dateinamen ist erforderlich. Diese müssen in einem Ordner gespeichert werden, welcher mit einem Zeitstempel versehen ist.
Der Dateiname muss zusätzlich den Präfix *CBXNETRechnung_* enthalten.

4 Produktfunktionen

4.1 Operationen und Funktionen

- Datei- und Verzeichnisoperationen:
  - Öffnen der Eingangsdatei
    - Übergabe der Datei erfolgt per Kommandozeile
  - Schreiben der Ausgabedateien in ein festgelegtes Ausgabeverzeichnis
    - automatisches Generieren nach Datum und laufender Nummer
    - alternative Angabe des Pfades erfolgt per Kommandozeile
  - Anlegen eines Ausgabeverzeichnisses bei nicht Vorhandensein

- Funktionen:
  - Interpretieren von PDF-Dokumenten
  - Durchsuchen von PDF-Dokumenten
  - Schreiben der PDF-Dokumentstruktur

4.2 Benutzer-Anforderungen

- Übergabe von Parametern zur Steuerung:
  - Quell PDF-Datei
  - Ausgabeverzeichnis
  - Vorlage PDF-Datei (hier wird ein leeres PDF-Dokument verwendet)
    Ermöglicht später die Verwendung von PDF-Vorlagen.
  - Passwort (für verschlüsselte PDF-Dokumente)
  - Ausgabe der Protokollierung (Informationen der Verarbeitung) zur Laufzeit

5 Qualitätsbestimmungen

Um die notwendige Qualitätssicherung zu gewährleisten, sind umfangreiche Tests der Anwendung, die unter Punkt 6 näher erläutert werden, durchzuführen.

Die Einhaltung dieser Qualitätsbestimmungen ist von größter Wichtigkeit, um die Hauptanforderung der hundertprozentigen Funktion zu garantieren.

Bei der Entwicklung muss sich streng an die Vereinbarung mit dem Entwickler der Automatisierungssoftware gehalten werden, die unter Punkt 3.4 der Ausgangsschnittstellendefinition festgehalten ist.

6 Testszenarien

Während des Verlaufs des Projektes sind ständig Entwicklertests der Anwendung durchzuführen.

Nach Abschluss der Implementierung muss die Anwendung einen abschließenden Systemtest mit sechs verschiedenen Testdateien fehlerfrei durchlaufen, welche 6, 70 & 600 Rechnungen enthalten und mit verschiedenen PDF-Generatoren erzeugt werden.
Die Tests werden im Beisein des Projektbetreuers durchgeführt, der die bestandenen Testkriterien auf dem Abnahmeprotokoll (Anhang 2) gegenzeichnet.

7 Entwicklungsumgebung

7.1 Software

- Microsoft™ Windows 2000 Professional System
- Sun JDK in der Version 1.4.1
- Entwicklungsumgebung Eclipse in der Version 2.1.2
- Open-Source-Bibliothek PDFBox von Ben Litchfield in der Version 0.6.4
- Adobe® Acrobat™

7.2 Hardware

Hardwareanforderungen sind nur in sofern gegeben, dass die unter Punkt 7.1 angeführte Software lauffähig sein muss.

Anlage 2

Abnahmeprotokoll

PDF - Parser & Splitter

Stand: 19.05.2004

Abnahmeprotokoll

für die Anwendung / für das Projekt

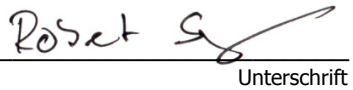
PDF – Parser & Splitter

Die Anwendung wurde für die interne Verwendung der Firma CBXNET com.box internet service gmbh geplant und in einem Projekt realisiert.

Die folgenden abschließenden Prüfungen zur Abnahme wurden erfolgreich durchlaufen:

- ☑ Verwenden von sechs unterschiedlichen Testdateien, welche von unterschiedlichen PDF-Generatoren erzeugt wurden und jeweils 6, 70 & 600 Einzelrechnungen zum extrahieren enthielten.
- ☑ Fehlerfreies Verhalten bei Tests mit den verschiedenen Kombinationen der Übergabeparameter.
- ☑ Fehlerfreies Verhalten auf falsche Übergabeparameter.
- ☑ Fehlerfreies Verhalten auf zwei verschiedenen Plattformen.
- ☑ Fehlerfreies Verhalten unter zwei verschiedenen JAVA VMs.

Mit dem gemeinschaftlichen Test der Anwendung durch den Projektbetreuer und den Entwickler gilt das Projekt hiermit als erfolgreich abgeschlossen und abgenommen von:

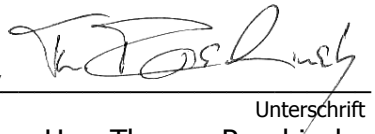
19.05.2004, 

Datum, Unterschrift

Projektbetreuer - Herr Robert Senger

Das Projekt dient gleichzeitig dem Entwickler der Anwendung und Auszubildenden Fachinformatiker für Anwendungsentwicklung der Firma CBXNET com.box internet service gmbh Thomas Roschinsky als Abschlussprojekt zur Abgabe bei der IHK.

Herr Roschinsky bestätigt mit seiner Unterschrift die selbständige Erstellung des gesamten Projekts und der Dokumentation inklusive Anlagen, Grafik und Layout.

19.05.2004, 

Datum, Unterschrift

Entwickler - Herr Thomas Roschinsky

Anlage 3

Relevante Passagen des Quelltextes

PDF - Parser & Splitter

Stand: 19.05.2004

Thomas Roschinsky 19.05.2004

1. Die verwendeten Importanweisungen

```
import java.io.IOException;
import java.io.InputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.OutputStream;
import java.io.Writer;
import java.util.Iterator;
import java.util.*;
import java.text.*;

import org.pdfbox.cos.COSArray;
import org.pdfbox.cos.COSDocument;
import org.pdfbox.cos.COSDictionary;
import org.pdfbox.cos.COSName;
import org.pdfbox.cos.COSObject;

import org.pdfbox.encryption.DecryptDocument;

import org.pdfbox.exceptions.InvalidPasswordException;
import org.pdfbox.exceptions.COSVisitorException;

import org.pdfbox.pdfparser.PDFParser;

import org.pdfbox.pdfwriter.COSWriter;

import org.pdfbox.pdmodel.PDDocument;
import org.pdfbox.pdmodel.PDDocumentInformation;

import org.pdfbox.util.PDFTextStripper;
```

2. Die Methode main() der Klasse *PDFParSpl*

```
public static void main( String[] args ) throws Exception {
      boolean ConsoleOut = false;
      int currentArgumentIndex = 0;
      String templateFile = "template.pdf";
      String password = "";
      String encoding = DEFAULT_ENCODING;
      String version = "0.5";
      PDFTextStripper stripper = new PDFTextStripper();
      String pdfFile = null;
      String outPath = null;
      // Einlesen der Kommandozeilenparameter
      for( int i=0; i<args.length; i++ )
      {
            if( args[i].equals( PASSWORD ) )
            {
                  i++;
                  if( i >= args.length )
                  {
                        usage();
                  }
                  password = args[i];
            }
            else if( args[i].equals( TEMPLATE ) )
            {
            i++;
                  if( i >= args.length )
                  {
```

```
                        usage();
                    }
                    templateFile = args[i];
                }
                else if( args[i].equals( LZAUSGABE ) )
                {
                    ConsoleOut = true;
                }
                else
                {
                    if( pdfFile == null )
                    {
                        pdfFile = args[i];
                    }
                    else
                    {
                        outPath = args[i];
                    }
                }
            }

            // Wenn keine Quelldatei angegeben wurde, die Kurzreferenz ausgeben
            if( pdfFile == null ) {
                usage();
            }

            InputStream input = null;
            InputStream input2 = null;
            Writer output = null;
            COSDocument document = null;
            PDDocument pddocument = null;
            try
            {
                input = new FileInputStream( pdfFile );
                input2 = new FileInputStream( pdfFile );
                long start = System.currentTimeMillis();

                PDFParser parser = new PDFParser( input );
                parser.parse();
                document = parser.getDocument();

                PDFParser parser2 = new PDFParser( input2 );
                parser2.parse();
                pddocument = parser2.getPDDocument();

                long stop = System.currentTimeMillis();
                int pagefirst = stripper.getStartPage();
                int pagelast = pddocument.getPageCount();

                int pagecount = pagefirst;
                String txtFILEout = null;
                String pdfFILEout = null;
                String pdfcontent = new String();
                String FolderPrefix = "RechnungsL ";
                String FilePrefix = "CBXNETRechnung_";
                String toSearch1 = "Kundennr.: ";
                int toSearch1L = 6;
                String lastresult1 = "000000";
                int lastresult1counter = 0;

                // Gernerieren des Ausgabepfades
                Date ts = new Date();
                SimpleDateFormat df = new SimpleDateFormat( "yyMMdd_HHmmss" );
                df.setTimeZone( TimeZone.getDefault() );
                String timestamp = df.format( ts );
                PDFParSpl dircheck = new PDFParSpl();
```

```
            if(outPath == null){
                  dircheck.makeDir(".\\"+ FolderPrefix + timestamp + "\\");
                  outPath = ".\\" + FolderPrefix + timestamp + "\\";
            }
            else{
                  dircheck.makeDir(outPath +  timestamp + "\\");
                  outPath = outPath +  timestamp + "\\";
            }

            // Auf Verschlüsselung prüfen
            if( document.isEncrypted() ){
                  try   {
                        DecryptDocument decryptor = new
                        DecryptDocument( document );
                        decryptor.decryptDocument( password );
                  }
                  catch( InvalidPasswordException e ) {
                  // Flasches Passwort
                        if( args.length == 4 ) {
                              System.err.println("Fehler: Das angegebene"+
                              "Passwort ist falsch." );
                              System.exit( 2 );
                        }
                  // Kein Passwort angegeben und Standard "" nicht korrekt
                        else {
                              System.err.println("Fehler: Das Dokument ist"+
                              "verschlüsselt." );
                              usage();
                        }
                  }
            }

            // Ausgabe - Start der Anwendung
            System.out.println("PDF PARSER & Splitter - Version " + version
            + " - Copyright 2004 CBXNET com.box internet service gmbh");

            System.out.println("-----------------------------------------"
            + "-----------------------------------------------\n");
            PDFParSpl meta = new PDFParSpl();
            meta.printMetadata( pddocument );


            // Anfang Zeitmessung
            start = System.currentTimeMillis();

            // Extraktionsprozess
            while(pagecount<(pagelast+1)){
                  stripper.setStartPage(pagecount);
                  stripper.setEndPage(pagecount);

                  pdfcontent = stripper.getText(document);

                  // Suche nach Element: "Kundennummer"
                  int index1 = pdfcontent.indexOf(toSearch1);
                  String result1 =
                        pdfcontent.substring(index1+toSearch1.length(),
                        index1+toSearch1.length()+toSearch1L);

                  // Gernerieren des Dateinamens
                  if(lastresult1.equals(result1)) {
                        lastresult1counter++;
                        pdfFILEout = outPath + FilePrefix + result1 +
                        "_p"+lastresult1counter+".pdf";
                  }
                  else{
                        lastresult1counter = 0;
```

```
                        pdfFILEout = outPath + FilePrefix + result1 +
                        ".pdf";
                }

                // Aufruf der Extraktionsfunktion
                PDFParSpl ParSpl = new PDFParSpl();
                try {
                        ParSpl.processPDF(templateFile ,pdfFile,
                        pdfFILEout, pagecount, pagecount);
                }
                catch (Exception e)     {
                        e.printStackTrace();
                }
                // Wenn gewünscht dann Ausgabe der Protokollierung
                if( ConsoleOut ) {
                        System.out.println("\n** Extraktion Seite (" +
                        pagecount + "/" + pagelast +
                        ") **\nInhalt des Dokuments:");
                        System.out.println(pdfcontent);
                        System.out.println("** Details (Lauf: "
                        +pagecount+") **");
                        System.out.println(" -Ausgabedatei: "+pdfFILEout);
                        System.out.println(" -Kundennummer: "+result1);
                        System.out.print("\n");
                }
                lastresult1 = result1;
                pagecount++;
        }

        // Ende Zeitmessung
        stop = System.currentTimeMillis();


        // Ausgabe - Ende der Anwendung
        System.out.println( "\nAlle Rechnungen wurden verarbeitet und"+
                        " im Ordner \""+ outPath +"\" abgelegt");
        System.out.println( "Die benötigte Verarbeitungszeit beträgt: "
                        +((stop-start)/1000)+" Sekunden.");

    }
    finally     {
        if( input != null ) {
                input.close();
        }
        if( input2 != null ) {
                input2.close();
        }
        if( output != null ) {
                output.close();
        }
        if( document != null ) {
                document.close();
        }
        if( pddocument != null ) {
                pddocument.close();
        }
    }
}
```