Dokumentation zur betrieblichen Projektarbeit

Thema:

Labormodul

Abschlussprüfung
Winter 2004

Fachinformatiker für Anwendungsentwicklung

Praktikumsbetrieb:	AM-SoFT GmbH IT-Systeme Hameln Niederlassung Rötha
Projektbetreuer:	Herr Harrendorf
Durchführungszeitraum:	30.08. bis 08.10.2004
IHK-Nummer:	5019
Autor:	Thomas Weise

Rötha am 29.09.2004

Dokumentation

des Projektes

Labormodul

Inhaltsverzeichnis

1. Beschreibung des Projektauftrages

1.1 Einleitung

Bestandteil der Abschlussprüfung zum Fachinformatiker für Anwendungsentwicklung ist die nachfolgend beschriebene Projektdurchführung.
Der ausbildende Betrieb ist die AM-SoFT GmbH IT-Systeme Hameln mit Niederlassung in Rötha, die unter anderem mit "DocumentBeam", dem elektronischen Briefkasten für Ämter und Gerichte in Deutschland bekannt geworden ist.

1.2 Ausgangslage

Aufgrund der ab 01. 01. 2005 in Kraft tretenden EU-Verordnung 178, nach der eine lückenlose Rückverfolgbarkeit von Nahrungsmittelprodukten seitens der Hersteller gewährleistet werden muss, sieht sich die GK-Rötha dazu veranlasst, ihre Datenverwaltung neu zu strukturieren.
Bisher erfolgt die Datenerfassung im Einkauf, Lager, Labor und in der Produktion auf verschiedenen Listen und Ordnern weitgehend schriftlich und wird dann später in der Buchhaltung in das Warenwirtschaftssystem (nachfolgend WWS genannt) eingegeben. Somit kann weder Echtzeit-Zugriff auf vorhandene Datenbestände, noch eine Rückverfolgbarkeit gewährleistet werden.

1.3 Aufgabenstellung

Aufgabe dieses Projektes ist es, ein Modul zu programmieren, welches die Erfassung und Änderung von verschiedenen Produktspezifikationen und deren Speicherung in einer Datenbank gewährleistet. Dabei ist darauf zu achten, dass sich dieses Modul in ein Hauptprogramm einbinden lässt und unter Windows XP lauffähig ist. Die Datenbankstruktur (z.B. Kundenstammdaten, Artikelstammdaten) des Unternehmens ist dabei zu übernehmen.

1.4 Projektziel

Ziel dieses Projektes ist es, einen Teil der Gesamtumstellung der Datenverwaltung in der GK Rötha zu übernehmen. Praktisch heißt das, die Wareneingangskontrolle durch das Labor soll durch das direkte Zugreifen auf die bereits im System gespeicherten Rohstoffchargen und Qualitätsanforderungen effizienter und sicherer werden. Desweiteren kann dadurch erst die Rückverfolgbarkeit sämtlicher Produkte bzw. Daten gewährleistet werden.

1.5 Projektumfeld

Der Auftraggeber dieses Projektes ist die AM-SoFT GmbH IT-Systeme Hameln mit Niederlassung in Rötha. Sie übernimmt die Abwicklung des Gesamtauftrages von der GK Rötha und ist somit für Anfragen an den Auftraggeber zuständig.
Das bedeutet, dass Fragen zum Projekt an den Ansprechpartner bei AM-SoFT zu richten sind. Das ist Herr Harrendorf, der zuständige Leiter des Gesamtauftrages und gleichzeitig mein Betreuer während des Praktikums.

1.6 Projektschnittstellen

Technische Schnittstellen des Modules sind:

- das Betriebssystem Windows XP
- das Hauptprogramm MAX-X (Perl-Programm)
- die BBX-Datenbank (eine in BusinessBasic geschriebene Datenbank)
- vorhandene Module der Firma AM-SoFT (Perl-Module)

Ansprechpartner zu diesem Projekt ist:

- Herr Harrendorf (AM-SoFT GmbH IT-Systeme)

1.7 Änderungen gegenüber Projektantrag

Es sind keine Änderungen gegenüber dem Projektantrag zu verzeichnen.

2. Projektplanung

2.1 Ist-Analyse

Die Ist-Analyse beschreibt die Ausgangssituation für das Projekt. Sie befindet sich im Anhang an die Projektdokumentation (Siehe: **ANLAGE:A - Ist-Analyse**).

Ablauf: 30.08. bis 02.09.2004 (geplant:10h - benötigt: 8h)

Da in der AM-SoFT GmbH viel Wert auf Praxisorientierung gelegt wird, erfolgt gleich zum Anfang ein längeres Gespräch mit dem Leiter des Gesamtauftrages Herrn Harrendorf, in dem das Prozessmanagement des Auftraggebers erörtert wird. Besonderes Augenmerk wird dabei auf die Datenverwaltung bzw. -erfassung gelegt.
Auf Basis dieses Gespräches wird, auch zum Zweck des besseren Hineindenkens in diesen Prozessablauf und im Hinblick auf die am Ende zu erstellende Dokumentation, ein Prozess- und Datenflussdiagramm erstellt.
(Siehe: **ANLAGE:C - Prozess- und Datenflussdiagramm**)

Am zweiten Tag des Projektes erfolgt eine Einweisung durch Herrn Harrendorf in den Aufbau der Datenbank und des Hauptprogrammes MAX-X. Dabei werden alle Namen für Variablen, Module, Tabellen und Felder zusammengestellt, die global (im gesamten Programm) gültig sind.
Es wird weiterhin festgelegt, welche Anforderungen zu beachten sind, dass sich das Modul optisch in das Gesamtkonzept einfügt (z.B. Layout, Farben, Schrift usw.).

Zum Schluss wird die Ist-Analyse (Siehe: **ANLAGE:A - Ist-Analyse**) fertiggestellt.

Damit ist die Analysephase abgeschlossen. Die geplante Zeit wurde um 2 Stunden unterschritten, da sich die Anforderungen als weniger komplex herausstellten, als vorher angenommen.

2.2 Soll-Analyse

Die Soll-Analyse (das Pflichtenheft) beschreibt die Anforderungen an das Projekt aus programmiertechnischer sowie anwendungsspezifischer Sichtweise. Sie befindet sich im Anhang an die Projektdokumentation (Siehe: **ANLAGE:B - Pflichtenheft**).

Ablauf: 03.09. bis 06.09.2004 (geplant:7h - benötigt: 7h)

Zu Beginn findet ein weiteres Gespräch mit Herrn Harrendorf statt, in dem nun klar festgelegt wird, welche Anforderungen an das Modul gestellt werden. Aus den daraufhin gewonnenen Erkenntnissen kann das Pflichtenheft angelegt werden, anhand dessen ein Konzept zum Aufbau des Modules und dessen Anbindung an die Datenbank und an das Hauptprogramm erstellt wird. Hierbei wird ebenfalls ein Augenmerk darauf gelegt, wie die schon vorhandenen Module eingebunden werden können.
Da es während der Programmierung sehr hilfreich sein kann, wenn man auf eine grafische Darstellung der Umgebungsvariablen blicken kann, habe ich mich entschlossen, ein Modulkonzept zu erstellen (Siehe: **ANLAGE:D - Modulkonzept**).

2.3 Wirtschaftlichkeit

1. Hardwarekosten
   Es fallen keine weiteren Hardwarekosten an, da das Notebook und der Datenbankserver-Rechner bereits vorhanden sind.
2. Softwarekosten
   Als Softwarekosten steht nur der Betrag für das Programm Optiperl mit 51,52 € zu Buche, da sämtliche anderen Produkte entweder kostenlos, oder in der Firma AM-SoFT vorhanden sind.
3. Personalkosten
   Der Tagessatz eines Praktikanten beträgt ca. 43,-€.
   Der Tagessatz eines AM-SoFT-Mitarbeiters beträgt ca. 900,-€.
4. Lizenzkosten
   Es sind keine zusätzlichen Lizenzkosten für dieses Projekt zu verbuchen, da sämtliche Lizenzen schon durch das Hauptprogramm abgedeckt werden.
5. Kosten für Zeitaufwand
   10 Manntage (1 Manntag = 8 Arbeitsstunden)

2.4 Alternativen

Bei der Durchführung eines Projektes, das die Erstellung einer Software beinhaltet, gibt es erfahrungsgemäß immer viele verschiedene Vorgehensweisen; Jede mit ihren spezifischen Vor- bzw. Nachteilen.

Da es nun eine ganz normale menschliche Eigenschaft ist, an bewährten Methoden festzuhalten und im Hinblick auf die zur Verfügung stehende Bearbeitungszeit, neueren Methoden eher skeptisch gegenüberzustehen, ist es eher ein schwieriges Unterfangen, den Praktikumsbetrieb von anderen Methoden zu überzeugen. Es ist auch nicht immer eine neue Methode automatisch besser als eine altbewährte Methode, nur weil sie neu ist.

Angesichts dieser Betrachtungsweise habe ich in dieser Planungsphase nur die zwei, aus meiner Sicht erfolgversprechendsten, Alternativen vorgeschlagen:

1. Da es für die Programmiersprache Perl keine IDE, wie z.B. den Borland C++Builder für C++ gibt, ist es sehr zeitaufwändig und fehlerträchtig, eine grafische Benutzeroberfläche in der Programmiersprache Perl zu erstellen.
   Da mein erster Vorschlag, dieses Modul in der Programmiersprache Java mit dem Entwicklungstool Eclipse3.0 (wegen der besseren Unterstützung für grafische Oberflächen) als eigenständiges Programm zu erstellen und somit Entwicklungszeit einzusparen, abgewiesen wurde, habe ich mir Gedanken über eine effektivere Programmerstellung mit Perl gemacht.
   Im Hinblick auf die weiteren (in Zukunft noch zu erstellenden) Module halte ich es für angebracht, eigene Klassen für die verschiedenen Frames der Programmoberfläche zu erstellen und diese dann objektorientiert in den verschiedenen Modulen einzusetzen. Frames sind Teile von Programmfenstern, die oft in ihrer Art im Programm immer wieder vorkommen.
   Zur Zeit werden die Frames in jedem Modul neu erstellt, was meist über 300 Zeilen Programmcode beansprucht.
   Nach meiner Ansicht würde sich die Programmerstellungszeit bzw. Fehlerwahrscheinlichkeit für zukünftige Projekte bei dieser Vorgehensweise wesentlich verringern.

2. Angesichts der Tatsache, dass der in Punkt 3.2.2 beschriebene Anmeldevorgang die eingegebenen Werte nur mit den in einer Konfigurationsdatei unverschlüsselt enthaltenen Daten vergleicht und dieser Ablauf heutigen Datenschutzanforderungen nicht mehr voll entspricht, wäre es ratsam, andere Verfahren anzuwenden.
   Da sich weiterhin herausgestellt hat, dass die verwendete BBX-Datenbank bei gleichzeitigem Zugriff mehrerer Clients sehr große Geschwindigkeitsdefizite zeigt, ist es aus meiner Sicht ebenfalls angebracht, über Alternativen zur verwendeten Datenbank nachzudenken.
   Ich schlage deshalb die Verwendung einer Mysql-Datenbank vor.
   Diese Datenbank ermöglicht eine interne Benutzerverwaltung anhand von 6 miteinander verknüpften Tabelen, die eine extrem sichere Zugangskontrolle gewährleistet. Daten können MD-5-verschlüsselt abgelegt werden.
   Desweiteren ist diese Datenbank sehr schnell und geeignet für große Datenmengen bzw. mehrfachen Clientzugriff, was in offiziellen Benchmarktestergebnissen nachgewiesen wird.

2.5 Entscheidung

Nach der Darlegung meiner Vorschläge zu den in Punkt 2.4 aufgeführten altenativen Vorgehensweisen zur Projektdurchführung wurde entschieden, dass diese Vorschläge nicht angewendet werden.

Als Begründung der Abwahl seitens des Praktikumsbetriebes wurde angeführt, dass diese Umstellungsvorschläge zwar für die Zukunft erfolgversprechend, die Umsetzung im Rahmen dieses Projektes aber zu zeitintensiv sind.

Daraufhin schlug ich vor, diese Umstellungen in meinem zweiten Praktikum durchzuführen, was durchaus positiv aufgenommen wurde.

2.6 Ressourcenplanung

Die Erstellung des Projektes wird an einem Notebook mit dem Betriebssystem Windows XP durchgeführt. Dies ist günstig, da das fertige Programm am Einsatzort ebenfalls auf dem Betriebssystem Windows XP lauffähig sein muss.

Um das Problem zu umgehen, dass die im Hauptprogramm verwendete BBX-Datenbank nicht während der gesamten Projektdurchführungsphase zur Verfügung steht, wird die auf dem Notebook schon installierte Mysql-Datenbank4.0.13 während der Durchführung des Projektes verwendet. Die originale BBX-Datenbank auf dem Notebook zu benutzen ist aus lizenztechnischen Gründen leider nicht gestattet.

Dies soll aber für die Projektdurchführung kein unüberwindbares Hindernis darstellen, da die Einbindung der jeweiligen Datenbank im Programmcode sowieso über ein sogenanntes Datenbank-Handle erfolgt und somit bei der Anbindung an das Hauptprogram nur eine Zeile Programmcode abzuändern ist.

Die Erstellung der Perl-Skripte erfolgt mit dem Perl-Editor Optiperl Professional 4.5.55 bzw. mit dem VIM6.2-Editor. Getestet wird über die DOS-Konsole.
Für die Erstellung der Dokumentation wird OpenOffice1.1 verwendet und die Skizzen bzw. Diagramme werden mit Visio erstellt.

2.6 Zeit- und Ablaufplanung

Für die Zeit- und Ablaufplanung wird eine tägliche Projektarbeitszeit von zwei bis fünf Stunden angenommen, da neben der Arbeit an dem Abschlussprojekt noch Arbeiten im Praktikumsbetrieb zu erledigen sind. Daraus resultiert eine Gesamtlaufzeit des Projektes, die zwischen 18 und 22 Arbeitstagen liegt.

Die Durchführung findet gemäß Projektantrag im Zeitraum vom 30. 08. 2004 bis 08.10.2004 statt.

In der unten stehenden Tabelle sind die einzelnen Aktivitäten mit den geplanten Zeitspannen aufgeführt, die sich aus den Projektphasen ergeben.

Projektphase	***Beschreibung der Aufgabe***	***Soll-Stunden***
Analyse	Analyse des Produktionsablaufes der GK Rötha	1
	Erstellung eines Prozess- und Datenflussdiagrammes	3
	Analyse des Projektumfeldes	2
	Analyse der zentralen Datenbank	2
	Ist-Aufnahmeprotokoll und grafische Darstellung	2
Planung	Erstellung Konzept	3
	Erstellung Pflichtenheft	2
	Erstellung grafischer Darstellungen	2
Durchführung	Bereitstellung der Entwicklungsumgebung	2
	Erstellung Layout	2
	Programmierung des Modules	19
	Anbindung der Zusatzmodule	3
	Testlauf des Labormodules	12
	Anbindung an Hauptprogramm und Datenbank	6
	Erstellung der Dokumentation	7
	Übergabe des Projektes	1
Auswertung	Auswertung des Projektes	1
	Gesamtstundenanzahl:	**70**

3. Projektdurchführung

3.1 Bereitstellung der Entwicklungsumgebung

Ablauf: 07.09.2004 (geplant:2h - benötigt: 2h)

Die Programmierung des Modules erfolgt auf einem Notebook mit dem Betriebssystem Windows XP Professional.
Wie bereits in Punkt 2.6 angesprochen, erfolgt die Erstellung der in **ANLAGE: B** aufgeführten Tabellen in der auf dem Notebook installierten Mysql-Datenbank.
Dies erfolgt unter Verwendung der in **ANLAGE: E** dargestellten SQL-Skripte.
Da abzusehen ist, dass dieses umfangreiche Projekt mit dem vorhandenen Vi-Editor in der vorgegebenen Zeit zu programmieren, eine sehr große Herausforderung darstellt, wird der leichtere Weg gewählt und das Programm Optiperl der griechischen Firma Xarka zur Erstellung der Perlskripte installiert. Diese Software ermöglicht eine bessere Übersicht des Programm-Codes z.B. durch Code-Folding (Zusammenklappen des Codes gesamter Subroutinen), Bracket-Highlighting (Anzeigen von Klammerbereichen), verbesserten Suchmöglichkeiten etc., was bei Modulen mit 3000 bis 4000 Zeilen Programm-Code wohl eine Erleichterung darstellt, jedoch in seiner Komfortabilität nicht z.B. mit dem Borland C++Builder konkurrieren kann.

Perl wird in der Version 5.8.4 von ActiveState installiert, die eine Grafikbibliothek Namens "Tk" enthält, welche es ermöglicht, eine grafische Programmoberfläche zu erstellen.
Da diese Grafikbibliothek nur Standard-Widgets enthält, deren Verwendung nicht die geforderten Designansprüche des Projektes abdeckt, müssen aus dem Internet noch spezielle Widgets, wie z.B. "Tk:Balloon" oder "Tk:ToolBar" von den sogenannten Perl-Repositories heruntergeladen werden. Dies erfolgt bei einer bestehenden Internet-verbindung durch Eingabe z.B. folgenden Befehles in der DOS-Eingabeaufforderung:

```
ppm install Tk:ToolBar
```

Daraufhin installiert sich dieses Modul selbstständig auf dem Rechner und kann danach mit verwendet werden.

Ein kurzer Test der kompletten Entwicklungsumgebung verläuft erfolgreich.

3.2 Programmierung

3.2.1 Erstellung Layout

Ablauf: 08.09.2004 (geplant:2h - benötigt: 4h)

Heute, da alle Vorbereitungen wie Analyse, Planung und Bereitstellung abgeschlossen sind, kann endlich mit der Programmierung des Labormodules begonnen werden.
Als erstes wird das Layout realisiert, das heißt das Aussehen des Programmfensters wird in Frames unterteilt, die Farben und Schriftformate eingebunden und die benötigten Widgets (z.B. Buttons oder Texteingabefelder) in den Frames plaziert.
Bei der Namensvergabe wird korrekt nach den Vorgaben aus dem Pflichtenheft (Siehe: **ANLAGE:B - Pflichtenheft**) vorgegangen. Das Labormodul selbst bekommt den Name labmod.pm und wird in dieser Dokumentation ab jetzt auch so genannt.

Diesen Teilabschnitt habe ich bei der Zeitplanung unterschätzt und konnte ihn nicht in der geplanten Zeitspanne realisieren.

3.2.2 Programmierung des Modules

Im Zusammenhang mit der Zeit- und Ablaufplanung sei an dieser Stelle darauf verwiesen, dass sich während der Programmierung folgende Erkenntnis ergab:
Es ist nicht ratsam, die drei Planungspunkte Programmierung, Anbindung der Zusatzmodule und Testlauf in der zeitlichen Abfolge zu trennen. Aufgrund dieser Erkenntnis erfolgt somit die Abarbeitung dieser drei Planungspunkte im ständigem Wechsel und wird in der Dokumentation auch so geschildert.

Ablauf: 09.09. bis 23.09.2004 (geplant:19h - benötigt: 37h)

Nun folgt der Hauptteil der Projektdurchführung. Da labmod.pl (das Labormodul) durch den Anwender vom Hauptprogramm MAX-X über ein Anmeldefenster nur mit Angabe eines gültigen Firmennamens, Benutzernamens und eines Passwortes zu erreichen sein darf, wird dieses Anmeldefenster als erstes erstellt. Es erhält den Name anmeld.pm und ist in der nebenstehenden <***Abbildung 1: anmeld.pm***> zu sehen.

Abbildung 1 anmeld.pm

Es vergleicht programmiertechnisch gesehen nur die eingegebenen Werte mit den in einer Konfigurationsdatei enthaltenen Werten. Sind die Werte in der Konfigurationsdatei enthalten, öffnet anmeld.pm das Labormodul labmod.pm, welches auf der Folgeseite unter <**Abbildung 2: labmod.pm**> abgebildet ist.

Im Hauptprogramm MAX-X wird nun ein Link (ein Button <Labormodul>) auf das Anmeldefenster anmeld.pm eingebaut und sogleich ein Test durchgeführt, welcher erfolgreich verläuft. Das heißt, das Anmeldefenster öffnet sich, fordert zur Eingabe eines gültigen Benutzeraccountes auf und verwehrt bei falschen Angaben den Zugang, öffnet aber wiederum bei Angabe eines gültigen Accounts das Laborprogramm.
Somit ist die Anbindung der ersten Schnittstelle (das Hauptprogramm) realisiert.

Um während der Programmierphase, die das ständige Testen beinhaltet, nicht ständig erst das Hauptprogramm starten und den Benutzeraccount eingeben zu müssen, erstelle ich ein Testskript test.pl. Dies nimmt zwar eine Stunde in Anspruch, soll sich aber im weiterem Verlauf durch Zeiteinsparung wieder auszahlen.

Dieses Testskript test.pl enthält Firmenname, Benutzername, Passwort, Verknüpfung zu der Konfigurationsdatei und das aktuell generierte Datum und öffnet sofort labmod.pm, das auf der folgenden Seite als <***Abbildung 2: labmod.pm***> abgebildet ist.

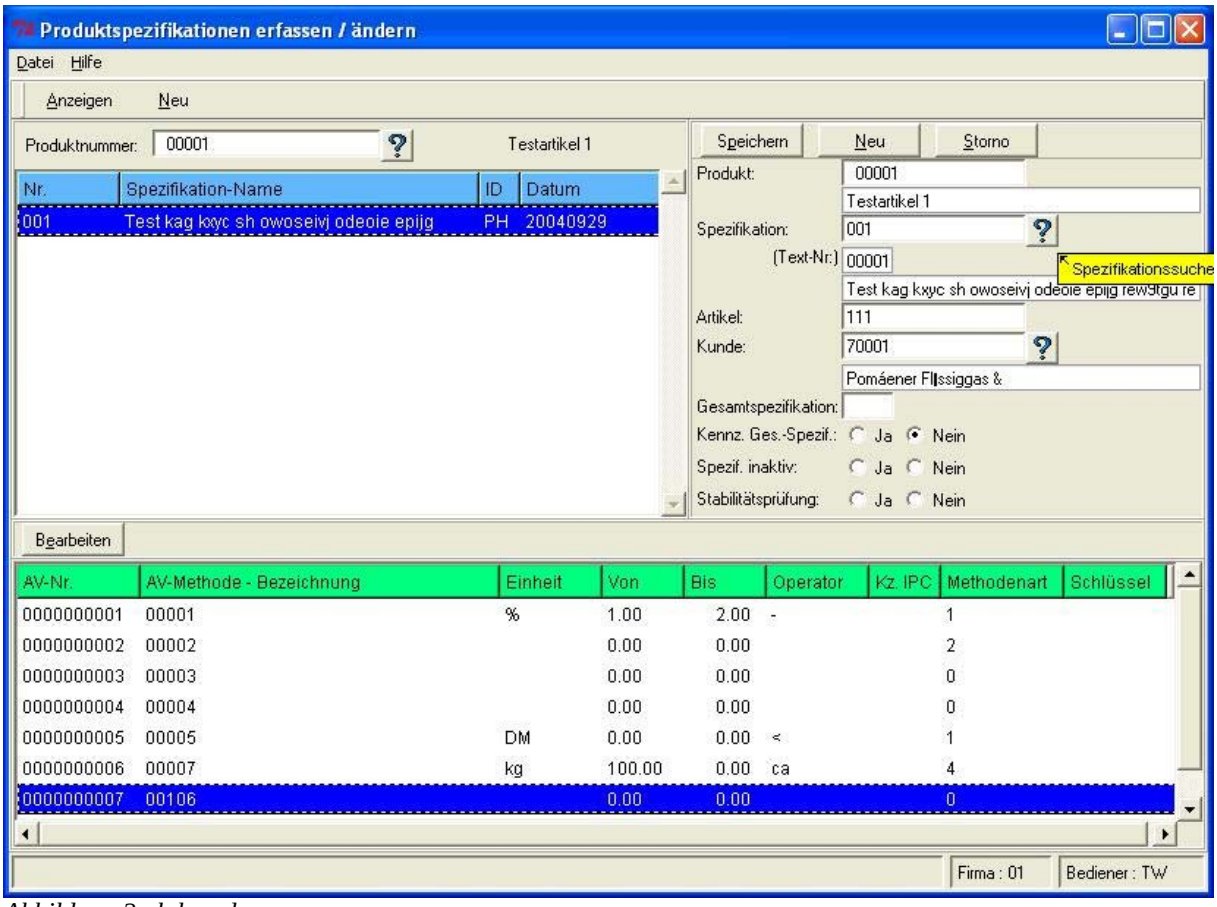


Abbildung 2 labmod.pm

Nun werden in labmod.pm zuerst alle Texteingabefelder mit Variablen verknüpft, um deren Inhalt jederzeit im Programm lesen, ändern oder speichern zu können.
Bei der Vergabe der Namen für die Variablen benutze ich dieselben Bezeichnungen, wie sie in der Datenbank enthalten sind, nur in Kleinbuchstaben und ein "*e_*" davor (z.B. "*e_cparnr*" für Artikelnummer).
Als Nächstes werden die Buttons mit Funktionen hinterlegt. Das geschieht in der Programmiersprache Perl in Form von sogenannten Subroutinen, die in etwa mit Funktionen vergleichbar sind, wie sie in der C und C++ Programmierung verwendet werden. Ich werde dies nun anhand eines Beispieles erleutern, natürlich nur in einer, den Rahmen dieser Projektdokumentation nicht überschreitenden, kurzen Darstellungsweise:

Nehmen wir z.B. den Button für die Suche nach schon erstellten Spezifikationen (dieses Beispiel veranschaulicht nebenbei gleich noch die Einbindung eines Zusatzmodules).
Dieser Button (in der <***Abbildung 2: labmod.pm***> gelb mit "*Spezifikationssuche*" bezeichnet) also ist in Perl ein Objekt, welches verschiedene Methoden besitzt, so auch eine Methode "command", die auf eine Subroutine verweist.

Nun ist es in Perl möglich, auch Subroutinen aus anderen Modulen zu verwenden, wenn diese anderen Module in der entsprechenden Form am Anfang des eigenen Modules eingebunden werden. Dies geschieht z.B. in der Form:

use amsoft_tk_chpssu;

Zwei Dinge sind dabei mindestens noch zu beachten:

1. Das einzufügende Modul muss in einem Pfad liegen, der vom Perlinterpreter gefunden wird.
2. Im eigenen Modul muss ein Objekt vom dem neuem Modul angelegt werden, z.B. in der Form:

   $chpssu_obj = amsoft_tk_chpssu->new(-[Option] => [Wert]);

Nun kann der Programmierer in der eingangs beschriebenen Button-Methode "command" z.B. folgendes angeben:

-command => sub { chpssu_obj->suche(-[Optionen] => [Werte]); }

Dadurch wird beim Betätigen des Buttons das Suchfenster aus dem eingebundenem Modul gestartet. Dieses eben erwähnte Suchfenster ist in <**Abbildung 3: spez_suche.pm**> abgebildet.

Abbildung 3 spez_suche.pm

Dies sollte aber nur einmal einen kleinen Einblick in die Perl-Programmierung geben und ich möchte nun weiter fortfahren mit der eigentlichen Projektdokumentation.

In der eben beschriebenen Art und Weise werden also nun alle Subroutinen (Funktionen) angelegt, was eine große Zeitspanne in Anspruch nimmt. Dies resultiert daraus, dass während der Erstellung der Subroutinen ständig deren Funktionsweise getestet wird. Da dies meist auch mit einem Zugriff auf die Datenbank zusammenhängt, liegt in dieser Phase der Hauptaufwand der gesamten Programmerstellung.
Um dem Benutzer die Bearbeitung der sogenannten AV-Methoden zu ermöglichen, entscheide ich mich, zu diesem Zweck zwei neue Module zu programmieren.
Ersteres trägt den Name av_bearb.pm und enthält ein neues Widget (Programmfenster).
Es ist im Bild <***Abbildung 4: av_bearb.pm***> dargestellt.

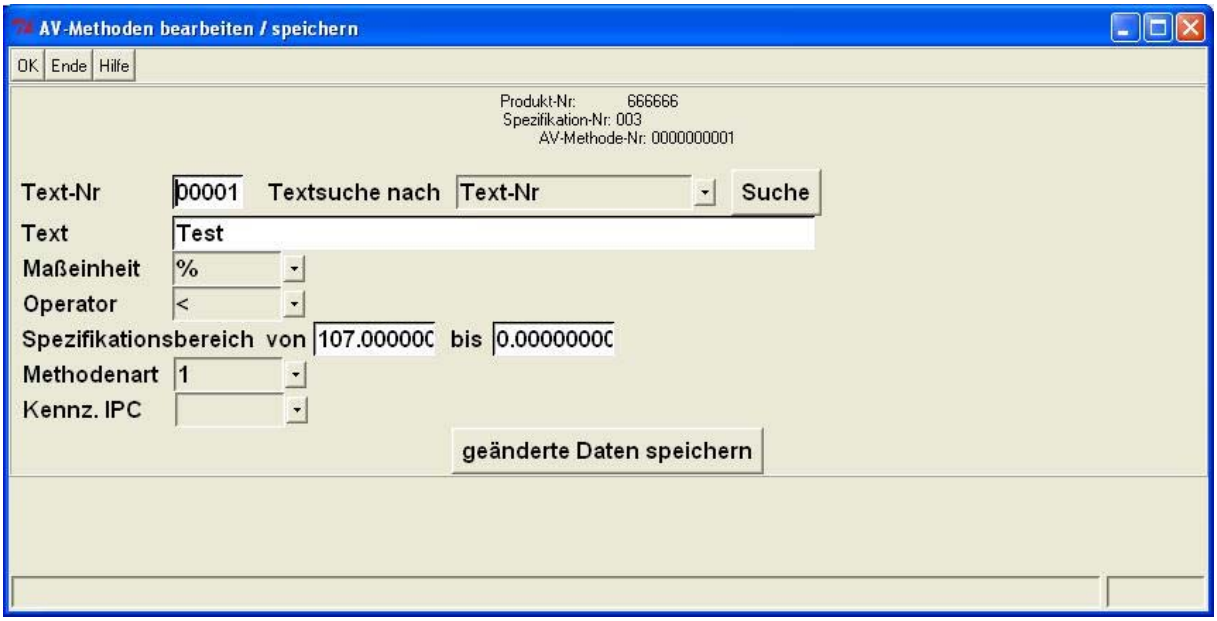


Abbildung 4 av_bearb.pm

Da dieses Programmfenster für den Anwender wiederum eine Suchmöglichkeit für schon vorhandene Spezifikations-Bezeichnungen (Text-Nr) bietet, wird noch ein zweites Programmfenster benötigt, was durch die Erstellung eines weiteren Modules bewerkstelligt wird. Letzteres erhält den Name txlfd_suche.pm und ist in der folgenden Abbildung <**Abbildung 5: txlfd_suche**> zu betrachten.

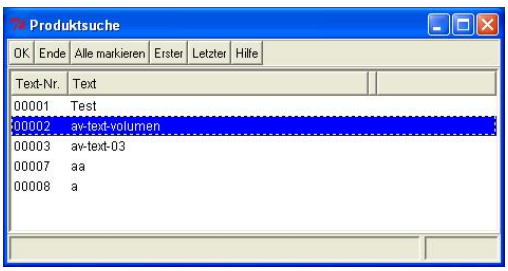


Abbildung 5 txlfd_suche.pm

Der nächste Schritt desteht darin, das Programm dahingehend anzupassen, dass der Benutzer mit möglichst wenigen Tastenkombinationen bzw. Mausklicks dieses Programm bedienen kann. Zu diesem Zweck, werden die Texteingabefelder und die verschiedenen Buttons in einer angemessenen Reihenfolge mit dem Tastaturfocus belegt. Im günstigstem Fall muss der Benutzer die Maus gar nicht mit verwenden. In dieser Phase stellt es sich als günstig heraus, dass der Prozessablauf in der Analysephase gründlich analysiert wurde.

Als letzter großer Programmierabschnitt steht noch die Belegung der geforderten Tastenkombinationen an (z.B. Strg+D für Drucken). Das ist deshalb etwas aufwendiger, als es wohl klingen mag, weil je nachdem wo sich der Tastaturfocus gerade befindet, verschiedene Routinen gestartet werden. Spätestens hier jedoch kommt mir der von Anfang an durchgeführte, objektorientierte Programmierstil zugute, wodurch ich die einmal erstellten Subroutinen nun dementsprechend immer wieder verwenden kann.

Zum Schluss möchte ich noch auf die Datensicherheit eingehen. Dies werde ich an einem Beispiel erläutern, z.B. die Prüfung der Benutzereingaben auf Fehleingaben.
Hierzu wird der Inhalt der Variable, die einem Eingabefeld zugeordnet ist z.B. in folgender Form geprüft:

```
   ###(prüfen auf 1 bis 5 Zahlen vor dem Punkt und 1 bis 2 Zahlen nach dem Punkt)
if ($variable =~/^-?[0-9]{1,5}?\.[0-9]{1,2}?$/)
{
                return 1;
}
```

Das heißt, wenn der Inhalt der Variable der Form dieses "regulären Ausdrucks" (eine Perl-spezifische Formatprüfung, im Beispielcode fett dargestellt) entspricht, wird 1 zurückgeliefert (sonst 0). So kann nun für die jeweils erforderlichen Eingabeformate abgeprüft werden, ob die Eingabe des Benutzers dem gewünschten Format entspricht und evtl. bei falscher Eingabe zur erneuten Eingabe aufgefordert werden.

Die geplante Zeitspanne für die Programmierung wurde nicht eingehalten, sondern auch mit Anrechnung der Punkte "*Anbindung der Zusatzmodule*"(3h) bzw. "*Testlauf des Labormodules*"(14h), um 3 Stunden überschritten.

3.2.3 Anbindung der Zusatzmodule

Ablauf: 09.09. bis 23.09.2004 (geplant:3h - benötigt: 0h)

Dieser Abschnitt wurde schon im Punkt "*3.2.2 Programmierung des Labormodules*" mit abgehandelt.

3.2.4 Testlauf des Labormodules

Ablauf: 09.09. bis 23.09.2004 (geplant:12h - benötigt: 0h)

Dieser Abschnitt wurde ebenfalls schon im Punkt "*3.2.2 Programmierung des Labormodules*" erleutert. Es sei an dieser Stelle nur noch erwähnt, dass die Zeitüberschreitung im Wesentlichen durch diese Testläufe hervorgerufen wurde.

3.3 Anbindung an Hauptprogramm und Datenbank

Ablauf: 24.09.2004 (geplant:6h - benötigt: 2h)

Die Anbindung an das Hauptprogramm erfolgt nun durch das Einbinden der von mir erstellten Perl-Module in das Gesamtkonzept.
Ich muss an dieser Stelle einfügen, dass dies der Abschnitt ist, vor dem ich im Vorfeld den meisten Respekt zollen musste. Angesichts der Tatsache, dass bis hierhin die geplante Zeit schon reichlich überschritten wurde, ist es umso höher einzuordnen, dass dieser Abschnitt so reibungslos vonstatten ging. Mit anderen Worten; Die Module werden eingebunden, die Datenbankzugangsdaten in der Konfigurationsdatei geändert, das Hauptprogramm gestartet und nach dem Öffnen des Labormodules können beim Testlauf keine Mängel festgestellt werden.
Somit wird die volle Funktionsfähigkeit des Modules bestätigt.

In diesem Zusammenhang sei mir noch eine kurze Bemerkung zur Datensicherheit erlaubt. Solange nur die fertigen Perlskripte an den Kunde geliefert werden, kann eine ausreichende Datensicherheit nicht garantiert werden, weil diese Skripte jederzeit manipuliert werden könnten. Um dies zu verhindern, werden in der Firma AM_SoFT sämtliche Skripte mit dem Programm "perl2exe" in lauffähige, startbare Dateien umgewandelt, bevor sie ausgeliefert werden.

Das Programm "Perl2exe" ist ein Kommandozeilen-Programm, welches Perl-Skripte in ausführbare Dateien (*.exe) umwandelt. Damit kann man eigenständige PC-Programme in Perl erstellen, die dann auf der Anwenderplattform (Windows oder Unix) ohne Perl-Interpreter lauffähig sind.
In diesem Zusammenhang sei darauf hingewiesen, dass im Perl-Skript je nach Verwendung verschiedene Codezeilen eingefügt werden müssen, wie z.B.:

```
#perl2exe_exclude Win32Util
#perl2exe_include "Tk/arrowdownwin.xbm"
```

Beim Kunde bereits vorhandene Datenbestände werden mit speziellen BusinessBasic-Tools in die BBX-Datenbank übernommen (ist aber nicht Bestandteil dieses Projektes).

3.4 Erstellung der Dokumentation

Ablauf: 27.09. bis 29.09.2004 (geplant:7h - benötigt: 8h)

Die Erstellung der Dokumentation erfolgt im Anschluss an die Fertigstellung des Projektes. Im gesamten Verlauf der Projektdurchführung wurden ständig nebenbei Stichpunkte bzw. ganze Beschreibungen der durchgeführten Tätigkeiten angelegt. Dies erleichtert, im Nachhinein gesehen, die Erstellung der Gesamtdokumentation in dieser Phase.
Die Dokumentation wird dem Praktikumsbetrieb zur Einsichtnahme und Kontrolle auf inhaltliche bzw. datenschutztechnische Korrektheit vorgelegt.

3.5 Übergabe des Projektes

Ablauf: 01.10.2004 (geplant:1h - benötigt: 1h)

Die Übergabe des Projektes erfolgt an Herrn Harrendorf, den zuständigen Leiter des Gesamtauftrages von der GK Rötha.
Dabei verläuft alles ordnungsgemäß. Es werden keine Mängel festgestellt und die erfolgreiche Durchführung dieses Projektauftrages wird bestätigt.

4 Auswertung

Nach der Durchführung dieses Projektes haben sich folgende Erkenntnisse ergeben:

1. Es ist wichtig, dass man sich im Verlauf der Programmierungsphase voll auf die Programmierung konzentrieren kann. Das heißt:
    - alle Varialblennamen sind bekannt
    - alle datenbankspezifischen Namen sind bekannt
    - eine klare Layout-Vorgabe liegt vor
    - der Prozessablauf ist bekannt und nachvollziehbar
2. Es ist günstig, wenn man unabhängig von äußeren Einflüssen ist. Das heißt:
    - durch objektorientierte Programmierung besteht keine Abhängigkeit vom Zugriff auf das Hauptprogramm während der Programmier- und Testphase
    - jederzeit ungestörter Zugriff auf eine lokale Datenbank (ohne die Gefahr, in der Programmier- und Testphase wichtige Daten zu löschen) möglich
3. Es ist vorteilhaft, bei der Dokumentation auf Stichpunkte zurüchgreifen zu können. d.h:
    - alle während der Planung und Durchführung erstellte Notizen sammeln
4. Es ist abzuraten, eine zeitliche Trennung von Programmier- und Testphase zu planen.

Nach der Durchführung und Auswertung des Projektes kann man folgendes Fazit ziehen:

1. persönliches Fazit:
    - Ich konnte während dieses Projektes wichtige Erfahrungen in der Programmierung sammeln und erhielt einen guten Einblick in die Vorgehensweise bei einem Programmierauftrag.
    - Durch die Analyse des Prozessablaufes in der GK Rötha ist es mir gelungen, die internen Abläufe in der Produktion sowie im Datenfluss zu begreifen.
2. Fazit von Unternehmerseite (AM-SoFT GmbH):
    - Für die Firma AM-SoFT ergibt sich eine Einsparung der Erstellungskosten von 8570,-€ .
      Dies resultiert aus dem geringeren Stundensatz eines Praktikanten im Vergleich zu einem festangestellten Programmierer(Siehe: Punkt 2.3).
3. Gesamtfazit:
    - Dieses Projekt kann als gelungen betrachtet werden, weil dem Kunde durch dieses Modul eine effektive und sichere Verwaltung der Spezifikationsdaten ermöglicht wird und eine Rückverfolgbarkeit gewährleistet ist.

5 Anlagenverzeichnis

Ist-Analyse

ANLAGE: A

zur Projektdokumentation

Labormodul

Inhaltsverzeichnis

1. Prozessorientierte Sicht

1.1 Einkauf / Wareneingang

Die Bestellung der Roh- und Hilfsstoffe erfolgt per Fax nach einem manuellen Vergleich der Lagerbestände im Lagerbuch mit den geplanten Produktionsmengen.
Der Wareneingang wird anhand der mitgelieferten Lieferscheine geprüft und in einem Eingangsbuch dokumentiert. Außerdem wird im Labor ein Wareneingangsprotokoll geführt, in dem Qualitätsparameter vermerkt werden. Chargennummern und Mindesthaltbarkeitsdaten werden im Labor auf gesonderten Listen eingetragen.
Weiterhin erfolgt durch die Abteilung Einkauf eine Eintragung in ein Anlieferbuch, um später den ordnungsgemäßen Rechnungseingang überprüfen zu können.

1.2 Produktion

Anhand von Soll-Rezepturen, die im Labor für jeden Ansatz handschriftlich erstellt wurden, erfolgt der Ansatz der Produkte in der Ausmischanlage ohne Erfassung von Rohstoffchargennummern. Dabei werden für jeden Ansatz die Rohstoffmengen in einem Ansatzprotokoll einer Ist-Rezeptur handschriftlich zugeordnet. Dieses Ansatzprotokoll ist die Grundlage zum Abbuchen der verbrauchten Rohstoffe. Danach werden die Produkte ohne weitere Kennzeichnung abgefüllt, verpackt und auf Paletten gestapelt.

1.3 Fertigwarenlager

Hier werden die Paletten mit einer Folienumwicklung gesichert, in die einzelnen Lagerbereiche verteilt und nach einer manuellen Zählung und Artikelzuordnung in ein Lagerbuch eingetragen.

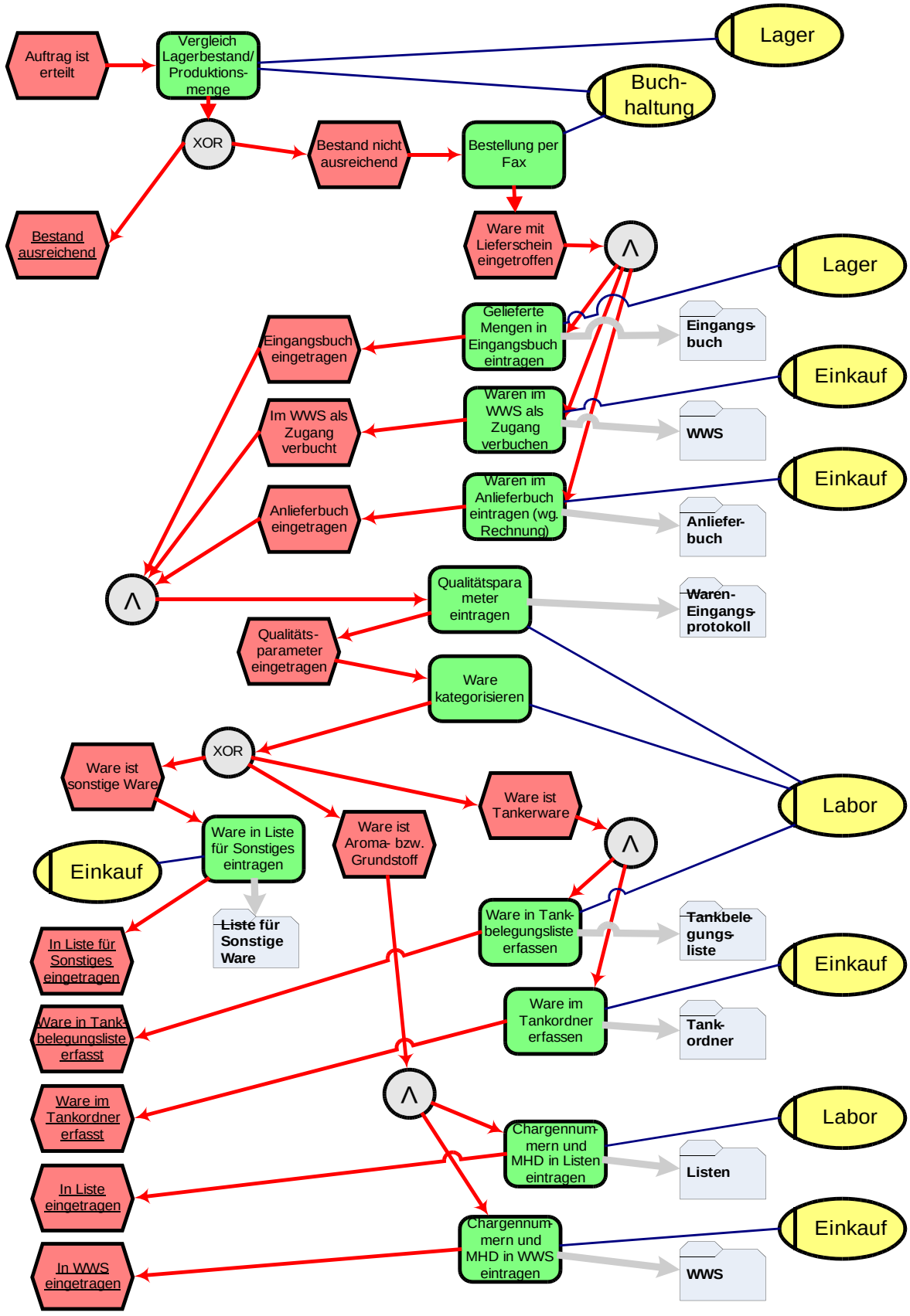
1.4 Versand

Kundenaufträge gehen per Fax ein und werden manuell in das WWS eingegeben. Die Lieferscheine werden aus dem WWS ausgedruckt und mit der jeweiligen Ware versandt.

1.5 Buchhaltung

Die Lieferantenrechnungen gehen per Post ein und werden manuell in das WWS eingegeben. Rechnungen werden aus dem WWS ausgedruckt und per Post verschickt.

1.6 grafische Darstellung



2. Datenverwaltungs-Sicht

2.1 Eingangsbuch

Der Wareneingang wird anhand der mitgelieferten Lieferscheine geprüft und in einem Eingangsbuch dokumentiert.

2.2 Anlieferbuch

Die Abteilung Einkauf erledigt die Eintragung der angelieferten Roh- und Hilfsstoffe in ein Anlieferbuch, um später den ordnungsgemäßen Rechnungseingang überprüfen zu können.

2.3 Wareneingangsprotokoll

Im Labor wird ein Wareneingangsprotokoll geführt, in dem Qualitätsparameter der angelieferten Roh- und Hilfsstoffe vermerkt werden.

2.4 Liste für Chargennummer und Mindesthaltbarkeitsdatum

Chargennummern und Mindesthaltbarkeitsdaten werden im Labor auf gesonderten Listen eingetragen.

2.5 Ansatzprotokoll

In der Produktion werden für jeden Ansatz eines Produktes die Rohstoffmengen in einem Ansatzprotokoll einer Ist-Rezeptur handschriftlich zugeordnet.

2.6 Lagerbuch

Im Fertigwarenlager werden die Artikel nach einer manuelle Zählung in ein Lagerbuch eingetragen.

2.7 WWS

Die per Fax erhaltenen Kundenaufträge, sowie die per Post eingehenden Lieferantenrechnungen werden manuell in das WWS eingegeben.
Lieferscheine und Lieferantenrechnungen für ausgehende Waren werden aus dem WWS ausgedruckt und mit der Post verschickt bzw. der ausgelieferten Ware beigelegt.

3. Software- und Datenbanktechnische Sicht

3.1 vorhandene Softwarestruktur

Zur Realisierung des Gesamtauftrages verwendet die AM-SoFT GmbH IT-Systeme ihre schon vorhandene Software MAX-X. Nach unternehmensbezogenen Anpassungen ist es mit diesem Programm möglich, Stammdaten wie z.B Kunden-, Artikel- oder Rohstoffdaten zu verwalten und eine effiziente Lagerwirtschaft in Verbindung mit einem WWS zu gewährleisten.

Diese Software ist in der Programmiersprache Perl geschrieben und ermöglicht durch seinen objektorientierten, modularen Aufbau die Einbindung zusätzlicher Module.

Verschiedene diverse Module, wie z.B. eine Erweiterte Suche, stehen ebenfalls bereit.
Der Benutzer arbeitet unter einer UNIX-Oberfläche, das heißt es ist keine grafische Benutzeroberfläche integriert.

3.2 vorhandene Datenbankstruktur

Das Programm MAX-X ist an eine BBX-Datenbank zur Speicherung sämtlicher Daten angebunden. Darin sind Tabellen zum Verwalten aller betriebsrelevanten Daten schon vorhanden.

BBX ist ein Produkt der Firma BASIS International Ltd., die rund um die Welt über 7000 Entwicklern innovative, hochwertige Software-Werkzeuge zur Verfügung stellt, um führende, insbesondere kommerzielle Anwendungen zu erstellen.

Weiterführende Angaben sind auf der Internetseite http://www.basis.com zu erfragen.

Pflichtenheft

ANLAGE: B

zur Projektdokumentation

Labormodul

Inhaltsverzeichnis

1. Anwendungsspezifische Sicht

1.1 Bedienung des Modules

Nach dem Start des Hauptprogrammes MAX-X kann z.B. der Laborarbeiter das Labormodul durch Anklicken des Buttons <Labormodul> öffnen.
Daraufhin zeigt sich ein Fenster, in dem die Firma, der Benutzername und das dazugehörige Passwort eingegeben werden muss. Daraufhin prüft das Programm die Eingaben und öffnet das Labormodul nur, wenn die eingegebene Kennung einen gültigen Benutzeraccount darstellt.
Nun, im eigentlichem Programm angekommen, kann der Benutzer eine Produktnummer eingeben oder eine Suche nach schon vorhandenen Produkten starten.
Nachdem das Produkt ausgewählt wurde hat man die Möglichkeit, entweder eine Spezifikationsnummer einzugeben, eine Suche nach schon vorhandenen Spezifikationen zu starten, oder eine neue Spezifikation anzulegen.
Wurde dies getan, zeigt das Programm die Eigenschaften dieser ausgewählten Produktspezifikation an, die da wären:

- Spezifikationstext	- Bezeichnung dieser Spezifikation
- Artikel	- Artikel, der dieser Spezifikation zugeordnet ist
- Kunde	- Kunde, der dieser Spezifikation zugeordnet ist
- Gesamtspezifikation	- Nummer, falls diese eine Gesamtspezifikation ist
- Kennz. Gesamtspez.	- Ja/Nein (ist dies eine Gesamtspezifikation)
- Spezifikation inaktiv	- Ja/nein (ist sie inaktiv, wird sie bei Suche versteckt)
- Stabilitätsprüfung	- Ja/Nein (wurde die Prüfung bestanden?)

Nun kann der Benutzer diese Eigenschaften entweder bearbeiten und speichern, oder sich die sogenannten AV-Methoden (Prüfmethoden) für diese Produktspezifikation anzeigen lassen.
Hat er die Anzeige der AV-Methoden gewählt, kann er widerum entweder eine AV-Methode bearbeiten oder neu anlegen.
Steht bei einer Spezifikation in allen zu ihr gehörenden AV-Methoden das Feld Stabilitätsprüfung auf "Ja", dann hat diese Spezifikation den Qualitätstest bestanden und wird auf inaktiv gesetzt.

1.2 Tastenbelegungen des Modules

Je nachdem, wo sich der Tastaturfocus (Cursor) gerade befindet, hat der Anwender während der Benutzung des Programmes jederzeit die Möglichkeit, verschiedene Aktionen mit den folgenden Tasten auszulösen:

F1	Aufruf eines speziellen Hilfefensters zu dem jeweiligem Feld
F4	Abbruch des gerade ausgeführten Vorganges (zurück)
F5	Starten eines Fensters zur erweiterten Suche (falls es ein Stammdaten-Feld ist)
F7	Neuanlage von Stammdaten (z.B. Artikel oder Spezifikation)

Enter	Daten in Datenbank übernehmen
Z + F11	Aufruf eines Menüs für Zusätze

Strg + S	Stornieren der gesamten Spezifikation
Strg + K	Kopieren einer schon eingegebenen Spezifikation
Strg + D	Ausdruck aller Spezifikationen zu diesem Produkt
Strg + H	Schreiben der Änderungshistorie
Strg + P	Spezifikationsspezifische Produktnamen erfassen
Strg + U	Kundenspezifische Zusatzdaten erfassen
Strg + G	Neuschreiben der Gesamtspezifikation (nur bei Kennz. "Gesamtspezifikation")
Strg + X	Export der Produktspezifikation

Pfeil oben	scrollen in den Spezifikationen aufwärts
Pfeil unten	scrollen in den Spezifikationen abwärts

2. Programmiertechnische Sicht

2.1 Einbindung des Modules

Das zu erstellende Labormodul ist vom Hauptprogramm MAX-X aus erreichbar. Bei dessen Aufruf erscheint ein Fenster zur Eingabe der Firma, des Benutzers und dessen Passwort. Nur wenn alle 3 Eingaben okay sind, öffnet sich das neue Fenster.
Das Labormodul selbst nutzt wiederum folgende vorhandene Module:

-amsoft_pfadset.pm	- Programmpfad setzen
-amsoft_ammand.pm	- Werte aus Konfigurationsdatei holen
-amsoft_dbh.pm	- Datenbank-Handle erstellen
-amsoft_tk_amdate.pm	- Datumsfunktionen
-amsoft_tk_fontholen.pm	- Daten aus Konfigurationsdateien holen
-amsoft_dialog.pm	- Dialogfenster und Messageboxen
-amsoft_espsta.pm	- Suche nach Kundenstammdaten
-amsoft_ampst1.pm	- Suche nach Artikelstammdaten
-spez_suche.pm.pm	- erweiterte Suche Produktspezifikationen

2.2 zu benutzende Variablen

Das Labormodul bekommt bei seiner Erzeugung aus dem Hauptprogramm folgende Variablen übergeben:

-pwidget	=> $mw,	- das Widget des Hauptprogrammes
-config	=> \$config,	- der Pfad zu der Konfigurationsdatei
-dbh	=> $dbh,	- das Datenbank-Handle (Verbindung zur Datenbank)
-firma	=> $firma,	- der Name der Firma
-bediener	=> $bediener,	- der Name des Bedieners

Ferner bekommt das Eingabeprüffenster noch die Variable des Passwortes übergeben.

2.3 zu benutzende Tabellennamen

Um eine reibungslose Datenbankanbindung zu gewährleisten, sind folgende Tabellen- bzw. Feldnamen zu verwenden:

CHPS00	Tabelle für Spezifikationen
CHPS01	Tabelle für AV-Methoden
CST101	Tabelle für Artikel-Stammdaten
CST3K01	Tabelle für Produkt-Stammdaten
CHTX01	Tabelle für Texte

2.4 zu benutzende Tabellenfeldnamen

Tabelle CHPS00:

CRNR	Feldname für Produktnummer
CPSPE	Feldname für Spezifikationsnummer
CPID	Feldname für Bediener
CPDAT	Feldname für (Änderungs-) Datum
TXLFD	Feldname für laufende Nummer des Textes
CPARNR	Feldname für Artikelnummer
K0	Feldname für Kundennummer
SPEH	Feldname für Gesamtspezifikation
SPHKZ	Feldname für Kennzeichen Hauptspezifikation
SPINA	Feldname für Spezifikation inaktiv
SPSTA	Feldname für Kennzeichen Stabilitätsprüfung

Tabelle CHPS01:

CPSLFD	Feldnamr für Produktspezifikation laufende Nummer
CPOP	Feldname für Operator für Bereichsprüfung
CPSBV	Feldname für Spezifikationsbereich von
CPSBB	Feldname für Spezifikationsbereich bis
CPSBT	Feldname für Nummer des Textes
CPDMAE	Feldname für Maßeinheit
CPFART	Feldname für Art der Analysemethode
IPCKNZ	Feldname für Kennzeichen IPC

Ppozess- und Datenflussdiagramm

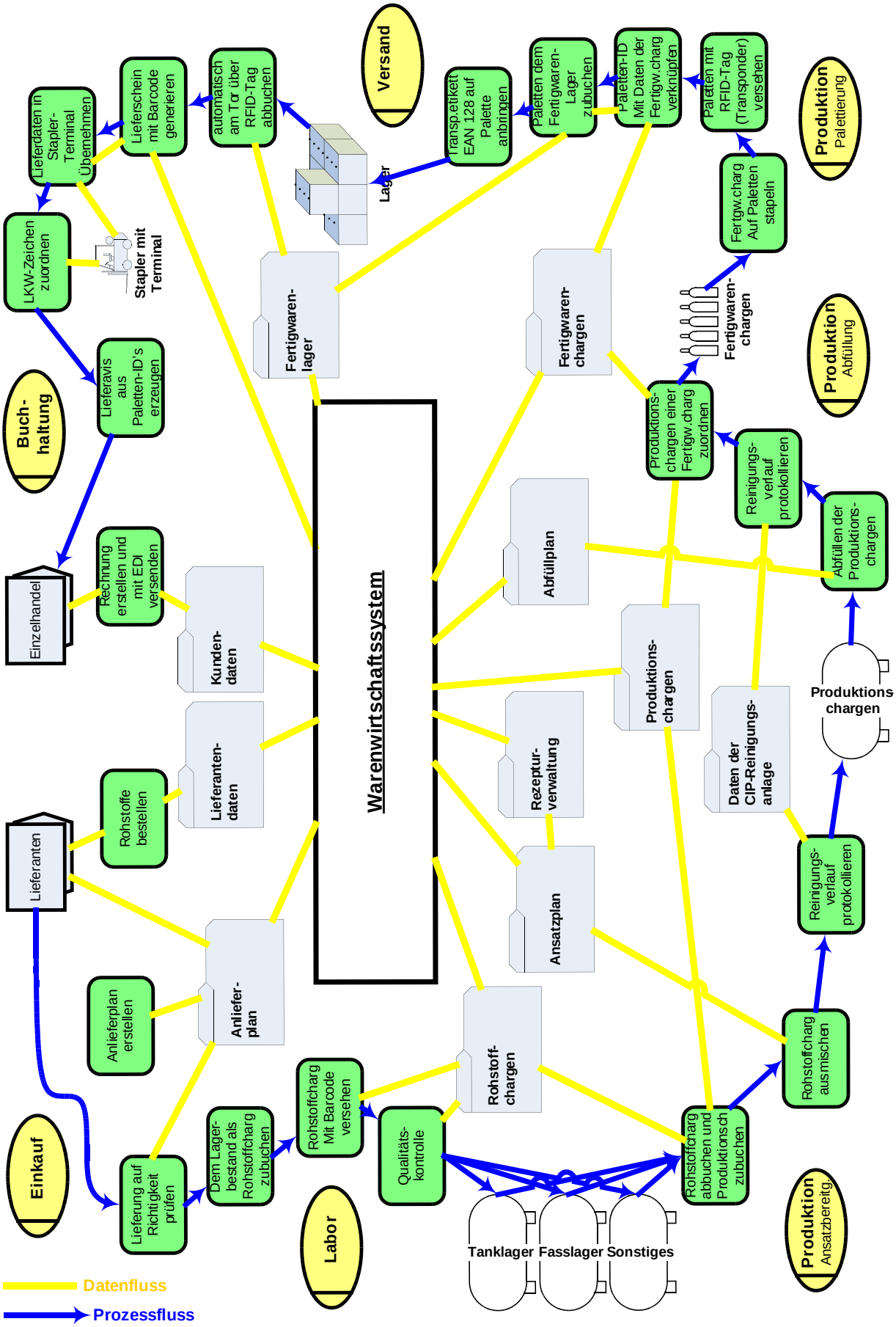
ANLAGE: C

zur Projektdokumentation

Labormodul

ANLAGE: C

Labormodul
Prozess- und Datenflussdiagramm



Modulkonzept

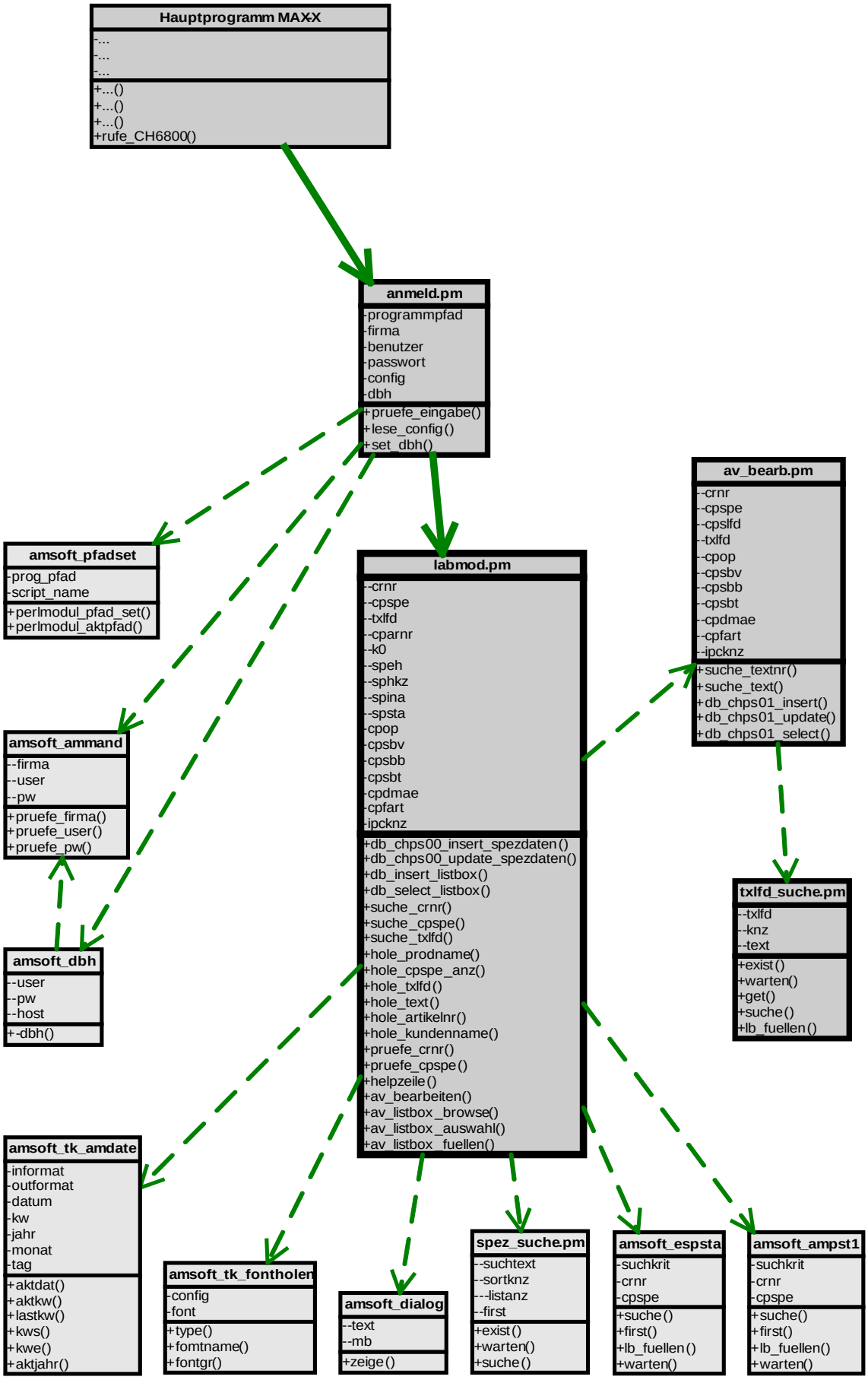
ANLAGE: D

zur Projektdokumentation

Labormodul

ANLAGE: D

Labormodul
Modulkonzept



SQL-Skripte

ANLAGE: E

zur Projektdokumentation

Labormodul

Tabelle CHPS00:

```
create database if not exists maxx;
use maxx;
create table if not exists CHPS00 (
        CRNR   char(20),
        CPSPE  char(20),
        CPID   char(4),
        CPDAT  char(8),
        TXLFD  char(5),
        CPARNR char(20),
        K0     char(5),
        SPEH   char(20),
        SPHKZ  char(1),
        SPINA  char(1),
        SPSTA  char(1),
        PRIMARY KEY(CRNR, CPSPE)
        );
```

Tabelle CHPS01:

```
create table if not exists CHPS01 (
        CRNR   char(20),
        CPSPE  char(20),
        CPSLFD char(10),
        TXLFD  char(5),
        CPOP   char(3),
        CPSBV  double,
        CPSBB  double,
        CPSBT  char(5),
        CPDMAE char(10),
        CPFART char(1),
        IPCKNZ char(1),
        PRIMARY KEY(CRNR, CPSPE, TXLFD)
        );
```

Tabelle CST101:

```
create table if not exists CST101 (
        CRNR   char(20) PRIMARY KEY,
        CRSU   char(15),
        CRHN1  char(40),
        CRHN2  char(40),
        CRCAS  char(15),
        CREIN  char(15),
        CRID   char(4),
        CRDAT  char(8),
        ASTA1  char(10),
        ASTA2  char(10)
        );
```

Tabelle CHTX01:

```
create table if not exists CHTX01 (
                TXKNZ  char(5),
                TXNUM  char(10),
                TXLFD  char(5),
                TXSUCH char(15),
                CHTX1D char(75),
                CHTX2D char(75),
                CHTX3D char(75),
                CHID   char(4),
                CHDAT  char(8),
                CHTXME char(10),
                CHTXKD char(20),
                CHTXKL char(1),
                PRIMARY KEY(TXKNZ, TXLFD)
                );
```

Daten einfügen:

```
use maxx;

load data local infile 'daten_CST101.txt'
into table CST101
fields terminated by ',';

load data local infile 'daten_CHPS00.txt'
into table CHPS00
fields terminated by ',';

load data local infile 'daten_CHTX01.txt'
into table CHTX01
fields terminated by ',';

load data local infile 'daten_CHPS01.txt'
into table CHPS01
fields terminated by ',';
```

Beispieldatei CHTX01.txt:

```
00001,0000000000,00001,TEST,Test,,,PH,20040715,%,,N,
00001,0000000000,00002,AV-02,AV-Methode 02,,,TW,20040906,KG,,,
00001,0000000000,00003,AV-03,AV-Methode 03,,,TW,20040906,CBM,,,
00001,0000000000,00004,AV-04,AV-Gewicht,,,TW,20040906,KG,,,
00006,0000000000,00001,TEST,Test,,,PH,20040715,,,,
00006,0000000000,00002,SPEZ-02,fuer Aldi,,,TW,20040906,,,,
00006,0000000000,00003,SPEZ-03,fuer Penny,,,TW,20040906,,,,
00006,0000000000,00004,SPEZ-04,fuer Netto,,,TW,20040906,,,,
```

Literatur- und Quellenverzeichnis

ANLAGE: F

zur Projektdokumentation

Labormodul

1. verwendete Literatur:

– **Mysql4**	Paul Dubois	Markt+Technik-Verlag
– **Mysql&Perl Developer's Guide**	Paul Dubois	Markt+Technik-Verlag
– **Perl Module**	Eric Foster-Johnson	mitp-Verlag
– **GoTo Perl**	Michael Schilli	Addison-Wesley

2. Webseiten:

zur BBX-Datenbank:

http://www.bsg-basis.de/body_weiterleitung.html (deutsch)
http://www.basis-gmbh.com/company/index.html (international)

Foren:

http://www.entwickler-forum.de/
http://www.talkaboutprogramming.com/group/comp.lang.perl.tk/

Perl:

http://www.perltk.org/
http://renormalist.net/cgi-bin/twiki/view/PM/PerlLinks
http://ppm.activestate.com/PPMPackages/zips/8xx-builds-only/Windows/ (Packages)
http://www.xarka.com/optiperl/index.html