Fachhochschule Konstanz
Fachbereich Informatik

Dokumentation

im Fach:

Anwendungen der betrieblichen Systemforschung

zum Software-Prototypen:

Beladungs**O**ptimierungs**P**rogramm

BOP

Bearbeiter:	Jürgen Kambeitz	Matr.-Nr. 271557
	Florian Pütz,	Matr.-Nr. 271470
Betreuer:	Professor Dr. Grütz	
Studiengang:	Wirtschaftsinformatik (Softwareentwicklung)	
Semester:	Sommersemester 2003	

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Allgemeines

Der in dieser Dokumentation beschriebene Software-Prototyp entstand im Rahmen der Veranstaltung „Anwendung der betrieblichen Systemforschung“ bei Herrn Prof. Dr. Grütz an der FH Konstanz.
Die Aufgabe bestand darin, die theoretischen Kenntnisse der vorangegangenen Veranstaltung „Betriebliche Systemforschung“ durch praktische Anwendung zu vertiefen. Dabei standen zwei Möglichkeiten zur Auswahl:

- Weiterentwicklung eines bereits vorhandenen Prototyps, oder
- Eigene Entwicklung eines neuen Prototyps.

Die Wahl fiel auf die Entwicklung eines neuen Prototyps zur Beladungsoptimierung beim Gütertransport.

1.1 Motivation

Die Idee dazu entstand bei einem Besuch der Spedition Dachser in Memmingen im Rahmen einer Exkursion der FH Konstanz.
Da es auf dem Gebiet der Beladungsoptimierung noch nicht eine große Zahl professioneller Software gibt, wurde die Entwicklung eines Prototyps in dieser Richtung angestrebt.

1.2 Kurze Funktionsbeschreibung

Nach der Eingabe verschiedener LKW- und Gebindetypen durch den Benutzer, sowie deren zu verladende Anzahl, errechnet der Prototyp die optimale Beladungsvariante für diesen Transportauftrag und stellt diese auf einer Auswertungsseite übersichtlich dar.

2 Bedienung des Programms

In den nachfolgenden Kapiteln wird die Bedienung der erstellten Software beschrieben.
Um einen ersten Eindruck zu gewinnen, ist nachfolgend ein Screenshot der Oberfläche, wie sie beim Start der Software erscheint, abgebildet:

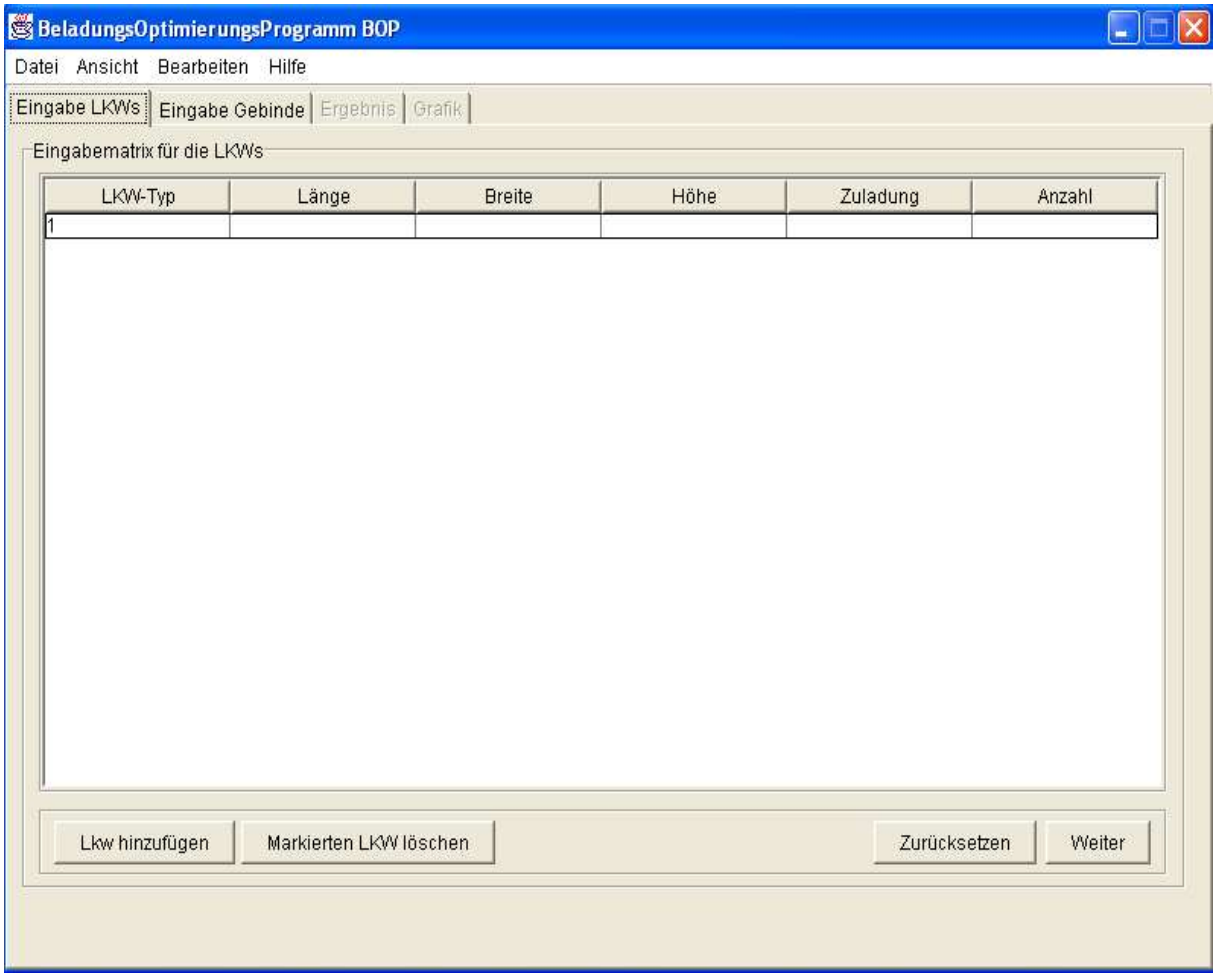

Abbildung 1: Screenshot des Startbildschirms

2.1 Allgemeine Punkte

Über eine Menüleiste hat der Benutzer Zugang zu den verschiedenen allgemeinen Funktionen des Prototyps.

2.1.1 Datei | Neu

Ruft der Benutzer diese Funktion auf, wird das gesamte Programm wieder auf die Einstellungen und Ansichten wie sie nach einem Neustart vorhanden sind, zurückgesetzt. Dabei werden auch evtl. vorhandene Einträge in den Registern „Eingabe LKWs“ sowie „Eingabe Gebinde“ gelöscht.

2.1.2 Datei | Öffnen

Durch den Aufruf dieses Menüeintrags öffnet sich ein Auswahldialog, in dem der Anwender die gewünschte, zuvor abgespeicherte Datei auswählen kann.

Gespeicherte Modelle werden im „BOP“-Format abgespeichert. Um hier nicht unnötig Verwirrung zu stiften, werden alle anderen Dateitypen im Auswahldialog ausgeblendet. Nach der Auswahl einer Datei und einer anschließenden Bestätigung erscheint das zugehörige Modell im Hauptfenster und kann nun berechnet oder modifiziert werden.

2.1.3 Datei | Speichern

Dieser Menüpunkt ist nur dann aktiviert, wenn das aktuell geöffnete Modell bereits zuvor mit „Speichern unter…“ gespeichert wurde. Ist dies der Fall, führt die Auswahl von „Speichern“ dazu, dass das Modell ohne weitere Rückfragen gespeichert wird.

2.1.4 Datei | Speichern unter

Wird ein Modell neu angelegt, kann es mit dieser Auswahl abschließend gespeichert werden. Dabei öffnet sich wieder ein Auswahldialog, in dem der gewünschte Dateiname angegeben werden muss. Das Anhängen der Dateierweiterung „.BOP“ erfolgt automatisch.

2.1.5 Datei | Beenden

Diese Auswahl führt zum Beenden des Programms.

2.1.6 Hilfe | Über

Klickt der Benutzer auf diesen Menüeintrag, erscheint ihm ein kurzer „Über-Dialog“, welcher den Namen des Programms sowie die Namen der Programmierer enthält.

2.2 Programmspezifische Punkte

Die programmspezifischen Punkte sind über zwei verschiedene Wege anwählbar.

- Über die Menüleiste, analog zu den allgemeinen Punkten; oder
- Über Buttons am unteren Ende des Applikationsfensters.

Wichtige Punkte sind über beide Wege zu erreichen.

2.2.1 Menüpunkt „Ansicht“

Über diesen Punkt kann durch die vier Register der Anwendung navigiert werden. Das Selbe lässt sich über die Buttons in der Navigationsleiste am unteren Rand erreichen.
Hier sind nicht alle Einträge immer aktiv, da z. B. der Punkt „Ergebnis“ erst aktiviert wird, nach dem eine Berechnung erfolgt ist.

2.2.2 Menüpunkt „Bearbeiten“

Unter diesem Punkt sind einige wichtige Programmfunktionen zusammen gefasst. So lässt sich über die Auswahl von „Berechnen“ der Vorgang zur Ermittlung der optimalen Beladungsvariante auslösen.

Im Punkt „Solver“ stehen verschiedene Solver zur Auswahl. Allerdings wurde bisher nur LP-Solve im Prototyp berücksichtigt. Die Programmstruktur sieht aber durch entsprechende Maßnahmen vor, dass andere Solver problemlos integriert werden können.

Wählt der Benutzer den Punkt „Solverpfade…“ aus, öffnet sich folgendes Fenster:

Abbildung 2: Setzen des Solverpfades

Hier kann der jeweils zugehörige Solverpfad ausgewählt und abgespeichert werden.

3 Funktionalität

Über die Register „Eingabe LKWs“ und „Eingabe Gebinde“ können die, für die spätere Berechnung relevanten Daten eingegeben werden.

Im Register „Eingabe LKWs“ sind dies:

- LKW-Typ: Durchlaufende Nummer, die vom System automatisch gepflegt wird,
- Länge; Die Länge des LKWs (in mm),
- Breite; Die Breite des LKWs (in mm),
- Höhe; Die Höhe des LKWs (in mm),
- Zuladung; Die Zuladung des LKWs (in kg) und
- Anzahl, die zur Verfügung stehende Anzahl dieses LKW-Typs

Durch einen Klick auf den Button „Weiter“ gelangt man in das Register Eingabe Gebinde“. Hier sind folgende Daten einzutragen:

- Gebinde-Typ: Durchlaufende Nummer, die vom System automatisch gepflegt wird,
- Länge; Die Länge des Gebindes (in mm),
- Breite; Die Breite des Gebindes (in mm),
- Höhe; Die Höhe des Gebindes (in mm),
- Zuladung; Das Gewicht des Gebindes (in kg) und
- Anzahl, die zu transportierende Menge dieses Gebinde-Typs

In beiden Eingaberegistern stehen Buttons bereit, um weitere Typen einzufügen oder bereits bestehende zu löschen. Dabei wird die Nummerierung der ersten Spalte immer automatisch mitgepflegt.

Sind alle Daten eingegeben, kann über den Button „Berechnen“ im Register „Eingabe Gebinde“ oder über den Menüpunkt „Berechnen“ die Berechnung der optimalen Lösung gestartet werden.

Die Ausgabe der Lösung erfolgt im Register Ergebnis, in welches nach erfolgreicher Berechnung automatisch umgeschaltet wird. Hier ist die Anzahl der Spalten dynamisch generiert, da sie von der Anzahl der zu transportierenden Gebinde abhängt.
In der ersten Spalte steht der verwendete LKW-Typ, gefolgt von der benutzten Anzahl in der nächsten Spalte. Nachfolgend sind dann die zu transportierenden Gebinde in der entsprechenden Anzahl aufgeführt.

4 Angaben zur Implementierung

4.1 Verwendete Programmierumgebung

Zur Implementierung des Prototyps wurde JBUILDER 7 von Borland mit dem JDK 1.3.1 eingesetzt.

4.2 Anbindung an den Solver

Um die Anbindung an verschiedene Solver so flexibel wie möglich zu halten, wurde ein Interface (siehe Abbildung 3: Klassendiagramm) eingeführt. Damit ist gewährleistet, dass auch andere Solver wie XA oder MOPS zukünftig ohne Probleme integriert werden können.

Für den Datenaustausch zwischen dem Prototypen und dem Solver sind zwei verschiedene Möglichkeiten vorgesehen.

Die erste Möglichkeit besteht darin, dass die Daten zwischen den beiden Instanzen über einen Daten-Stream ausgetauscht werden. Der Vorteil besteht darin, dass keine Dateien angelegt werden müssen, der Austausch erfolgt sozusagen „on-the-fly“. Ein Nachteil ergibt sich jedoch daraus, dass bei einer großen Anzahl von Variablen die Performanz dieser Lösung stark nachlässt. Dies äußert sich dann in einer sehr langen Antwortzeit des Solvers. Da bei BOP häufig viele Variable benutzt werden, musste noch eine alternative Lösung her.

Diese bestand darin, dass in ein Standardausgabeverzeichnis (hier: C:\temp) eine Datei geschrieben wird, aus der der Solver dann die Daten einlesen kann. Ebenso erfolgt nach der Berechnung die Ausgabe des Solvers in eine Datei, aus der dann das Ergebnis eingelesen und anschließend dargestellt wird.

4.3 LP-Ansatz

Da es sich bei dieser Problemstellung um ein mehrdimensionales Verschnittproblem handelt, musste vor dem eigentlichen LP-Ansatz noch das Subproblem der Variantenfindung gelöst werden. Aus diesem Grunde wurde ein Algorithmus konstruiert, der dieses Problem löst.

4.3.1 Verwendeter Algorithmus

Bei dem zu findenden Algorithmus kommt nicht nur das mehrdimensionale Verschnittproblem zum tragen, sondern auch ein Mischproblem. Dies soll ein kleines Beispiel verdeutlichen. Auf einen LKW mit spezieller Größe kann man 10 Platten vom Typ 1, 20 vom Typ 2 und 40 vom Typ 3 laden, dies ist das typische Mischproblem. Der entwickelte Algorithmus berechnet zuerst die maximale Anzahl für jedes Gebinde, die auf den LKW-Typ geladen werden können. Anhand dieser Werte können die Verhältnisse zwischen den einzelnen Gebindetypen berechnet werden.
Berechnet werden soll nun eine Matrix, die die verschiedenen Varianten findet, dazu wird der Datentyp ListArray und das normale Array verwendet. Gründe für die Wahl sind die folgenden Tatsachen. Die Breite ist beim Aufruf der Methode bekannt,

weshalb auf ein einfaches Array zurückgegriffen werden kann. Die Länge der Matrix bleibt bis zum Schluss variable, dafür eignet sich der Datentyp ListArray, da man beliebig viele Daten anhängen kann.
Im ersten Schritt wird eine Zeile generiert, die die maximale Anzahl vom ersten Gebindetyp gespeichert hat. In den folgenden Schritten wird die vorherige Zeile kopiert und der Wert, der als letztes geändert worden ist dekrementiert. Der Restwert wird dann entsprechend dem berechneten Verhältnis in die nächste Spalte geschrieben und der Algorithmus startet von vorne. So lassen sich alle Varianten finden.
Da in diesem Algorithmus zahlreiche Berechnungen vorkommen und JAVA ebenfalls nicht sehr performant ist, kann einige Zeit dauern bis alle Varianten berechnet sind.

4.3.2 Übergebenes Modell an den Solver

Nachdem der Algorithmus seine Berechnungen erledigt hat, wird zunächst die Methode writeToInFile() aufgerufen. Dieses Funktion bekommt als Parameter zum einem die errechnete Matrix und zum anderem die Eingabetabelle der Gebinde. Die Funktion wirteToInFile() wurde bewusst in der eigentlichen Solverklasse implementiert, da jeder Solver ein unterschiedliches Eingabeformat wünscht.
Die Methode erstellt ihrerseits die Zielfunktion, die Restriktionen und die Bedingung, dass jede Variable ganzzahlig sein muss, welche dann in privaten Variablen der Klasse gespeichert.
Im Anschluss erfolgt der Aufruf der Funktion executeSolver() mit dem Pfad der Solverdatei als Parameter. Wie bereits erwähnt, gibt es zwei Möglichkeiten der Übergabe der variablen an den Solver. Nachdem Aufruf wird zunächst überprüft, ob die Übergabepfad mit .bat endet, ist dies der Fall, so werden zeilfunktion, Restriktionen und Bedingung in eine Datei auf C:\Temp geschrieben, anderenfalls erfolgt die Eingabe über einen Datenstream direkt an das entsprechenden EXE-File. Das Ergebnis des Solvers wird in ein ListArray gespeichert und an die aufrufende Methode zurückgegeben.

4.4 Klassendiagramm

Für die Erstellung des Klassendiagramms wurde die Software Rational Rose verwendet.

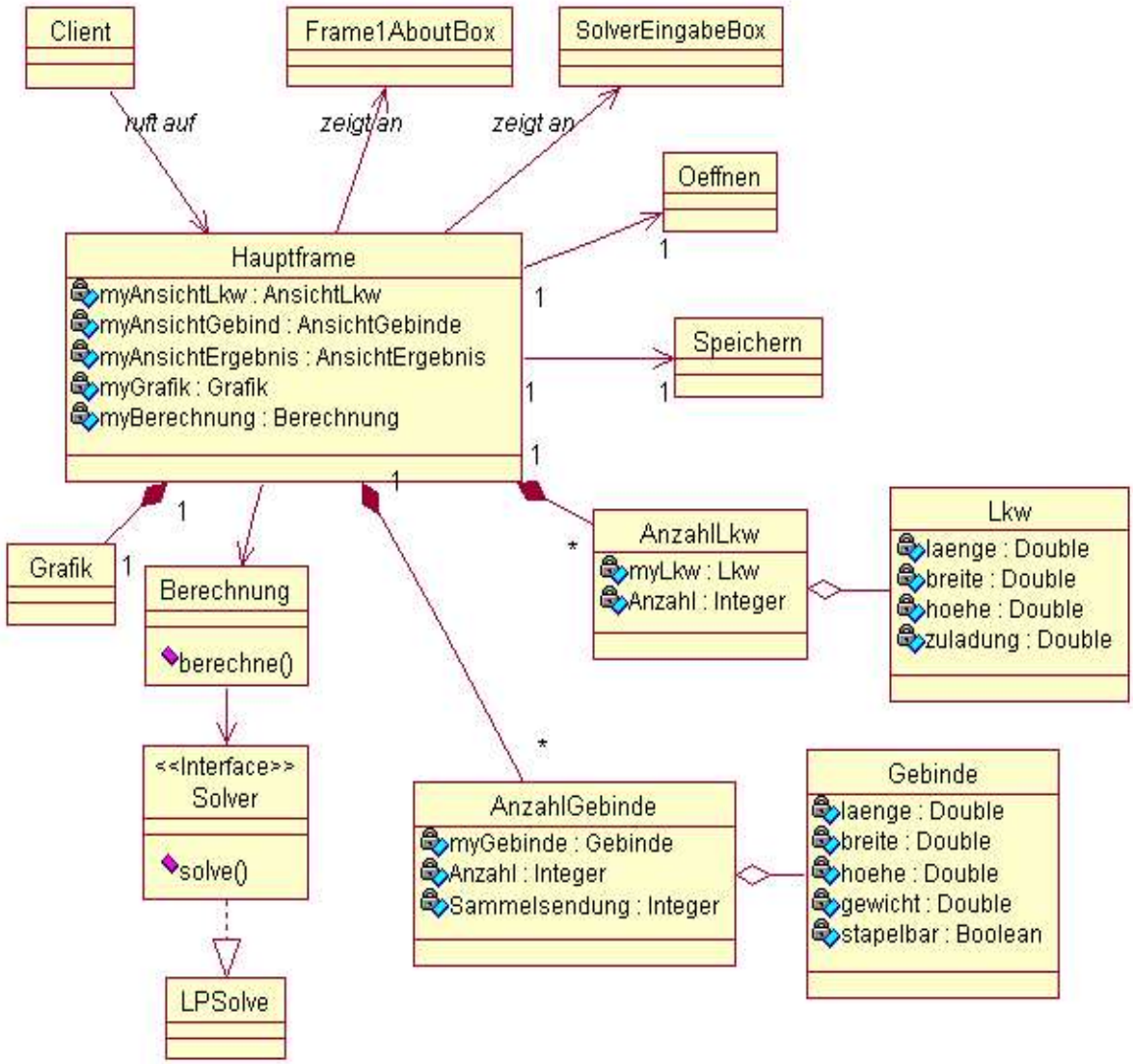

Abbildung 3: Klassendiagramm

4.5 Beschreibung der Klassen und Methoden

Die Dokumentation der Klassen und Methoden wurde durch die Verwendung einer entsprechenden Kommentar-Syntax mit Hilfe des Tools JAVADOC automatisch erzeugt. Sie befindet sich in HTML-Form auf der mitgelieferten CD im Verzeichnis \doc. Durch den Aufruf der Datei index.html in einem Browser wird sie gestartet.

Hier ein Screenshot der Startseite:

All Classes

AnzahlGebinde
AnzahlLkw
Berechnung
BopFilter
Client
Gebinde
HauptFrame
HauptFrame_AboutBox
Lkw
LPSolve
Oeffnen
Solver
SolverEingabeBox
Speichern

Package Class Tree Deprecated Index Help

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES

Package absf1

Interface Summary

Solver	Title: Solver Interface Description:Interface Schnittstelle für die Solverklassen Copyright: Copyright (c) 2003 Company:

Class Summary

AnzahlGebinde	Title: AnzahlGebinde Description: Bildet eine Zuordnung eines einzelnen Gebindes zu einer Anzahl und einer evtl.
AnzahlLkw	Title: AnzahlLkw Description: Bildet den Mengenaspekt bezueglich der verfuegbaren LKWs ab Copyright: Copyright (c) 2003 Company: FH Konstanz
Berechnung	Title: Solver Interface Description:Interface Schnittstelle für die Solverklassen Copyright: Copyright (c) 2003 Company:
BopFilter	Title: BopFilter Description: Klasse zur Filterung der Dateien beim Speichern- und Öffnendialog.
Client	Title: Client Description: Ausgangspunkt der Applikation.
Gebinde	Title: Gebinde Description: Bildet ein Gebinde (Palette, Gitterbox o. ae. ab Copyright: Copyright (c) 2003 Company: FH Konstanz

Abbildung 4: Startseite der HTML-Dokumentation

5 Verbesserungsvorschläge

Im Rahmen dieser Arbeit wurden im Laufe der Zeit bereits einige Verbesserungsvorschläge und Erweiterungen erkannt, die aus zeitlichen Gründen nicht mehr implementiert werden konnten. Trotzdem sollen sie in diesem Kapitel als Anregung für zukünftige Gruppen Erwähnung finden:

- Berücksichtigung der maximalen Zuladung,
- Abfrage und Berücksichtigung, ob die Gebinde stapelbar sind,
- Graphische Auswertung vom Ergebnis,
- Überprüfung der eingegebenen Daten auf Plausibilität,
- Berücksichtigung von Sammeltransporten,
- Anbindung weiterer Solver und
- Quer-Beladung von Gebinden.

Dies ist nur als eine Auswahl an Möglichkeiten zu sehen. Mit Sicherheit gibt es aber noch weitere, sinnvolle Verbesserungen.