Universität Stuttgart
Fakultät Informatik

Institut für Informatik
Breitwiesenstraße 20-22
D-70565 Stuttgart

Erfahrungen mit dem System TROSS beim DRK

Friedhelm Buchholz, Frank Wagner

Report Nr. 2000/01

24. Januar 2000

Die Projektgruppe Transportoptimierung hat in einem Jahr (WS 97/98, SS 98) das System “Transportorganisation für soziale Serviceanbieter” (TROSS) zur Disposition für die Fahrdienste des Deutschen Roten Kreuzes (DRK) Stuttgart erstellt. In den Berichten [3] und [4] wird TROSS im Detail vorgestellt. Weil keine Wartung gewährleistet werden kann und das System auch noch einige Fehler hat, wird TROSS vom DRK nicht bei der täglichen Disposition benutzt. Trotzdem ist das Interesse vom DRK an einer computerunterstützten Planung und Optimierung bei der Organisation der Fahrdienste sehr groß. Aus diesem Grund hat man sich dort intensiv mit TROSS beschäftigt. Die dabei gesammelten Erfahrungen und daraufhin durchgeführten Korrekturen an TROSS werden in diesem Bericht zusammengefasst. Am Ende des Berichts werden dann mögliche Erweiterungen des Systems vorgestellt.

1 Geplante Benutzung von TROSS

TROSS ist ein Programm zur Verwaltung und Kontrolle der (planungsrelevanten) Daten der Abteilung Mobile Dienste beim DRK Stuttgart. Unterstützt werden folgende Dienste: Schulfahrdienst, Behindertenfahrdienst, Altenpflege und Essen auf Rädern.

Bevor man richtig mit TROSS arbeiten und seine Touren planen kann, müssen zunächst die zur Verfügung stehenden Ressourcen (Mitarbeiter und Fahrzeuge) eingegeben werden.

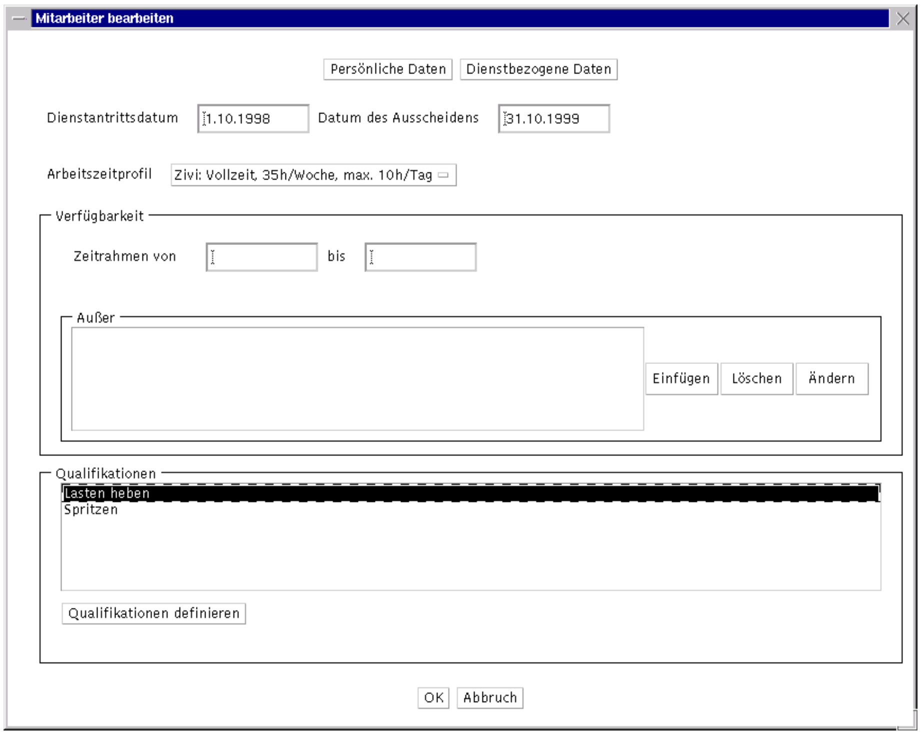

Abbildung 1: Maske mit den planungsrelevanten Daten eines Mitarbeiters

Die Maske für die Eingabe von *Mitarbeiterdaten* besteht aus zwei Teilen. Der erst Teil enthält die eindeutige Mitarbeiternummer und den Namen des Mitarbeiters sowie dessen Adressen und Telefonnummern.

Der zweite Teil (Abb. 1) enthält Randbedingungen für die Planung. Dienstantrittsdatum und das Datum des Ausscheidens sind Grenzen für die Verfügbarkeit. Das Arbeitszeitprofil gibt den Spielraum für die Einsatzzeiten des Mitarbeiters an. Die Profile können in TROSS definiert werden. Dabei gibt es drei Typen:

1. das Vollzeitprofil: hier wird die wöchentliche und die maximale tägliche Arbeitszeit festgelegt. Zivildienstleistende arbeiten zum Beispiel 35 Stunden pro Woche und maximal 10 Stunden am Tag.

2. Teilzeit tageweise: hier wird für jeden Tag festgelegt, wann der Mitarbeiter

einsetzbar ist. Diese Profile sind für Mitarbeiter gedacht, die nebenberuflich beim DRK arbeiten (630 Mark Jobs). Bei diesem Typ hat in der Regel jeder Mitarbeiter sein eigenes Profil.

3. Teilzeit wochenweise: Hier wird eine monatliche Arbeitszeit und eine maximale tägliche Arbeitszeit festgelegt.

Im Rahmen Verfügbarkeit werden die Tage festgelegt, an denen der Mitarbeiter verfügbar ist. Der Zeitrahmen gibt den Beschäftigungszeitraum an. Ausnahmen sind z.B. Ferien und Schulungen.

Die Qualifikationen können frei definiert werden. Gedacht sind sie zum Beispiel für "Lasten heben" oder "Spritze setzen". Die Qualifikationen können von den Kunden gefordert werden, was dann bei der Planung berücksichtigt wird. Der TROSS-Benutzer kann daraus resultierende Konsistenzverletzungen aber auch einfach akzeptieren.

Die Maske für *Fahrzeugdaten* ist auf zwei Seiten verteilt. Die erste Seite (Abb. 2) enthält zunächst die KFZ-Nummer und eine eindeutige Bezeichnung für das Fahrzeug. Die "technischen Termine" prüft TROSS beim Starten und informiert den Benutzer gegebenenfalls über demnächst fällige TÜV- und ASU-Termine. Die Verfügbarkeit wird wie bei den Mitarbeitern festgelegt.

Die zweite Seite beschreibt die Ausstattung des Fahrzeuges. Im oberen Teil können bewegliche Hilfsmittel und deren Anzahl festgelegt werden, die im Fahrzeug mitgenommen werden sollen. Der untere Teil gibt die nicht so leicht veränderbare Ausstattung an. Zunächst können zu jedem Fahrzeugtyp verschiedene Konfigurationen definiert werden, von denen dann eine für das Fahrzeug ausgewählt wird. Das Fahrzeug in der Abbildung hat gegenwärtig drei normale Sitzplätze, keine festen Sitzhilfen und drei Plätze für Rollstuhlfahrer.

Die Eingabe von *Kundendaten* gliedert sich in drei Teile: persönliche Daten, dienstbezogene Daten und Dienstwünsche. Die persönlichen Daten (Abb. 3) enthalten unter anderem die maximale Fahrdauer pro Fahrt und die Heimadresse, die als Vorgabe für den Startort von Dienstwünschen verwendet wird.

Die Abneigungen und Zuneigungen der dienstbezogenen Daten (Abb. 4) beeinflussen die Auswahl der Mitarbeiter; Hilfsmittel und zulässige Fahrzeuge schränken die Wahl des Fahrzeugs ein. Die Bezugspersonen dagegen sind mehr informativ. Bezugspersonen sind z.B. der Hausarzt oder auch Verwandte. Sie können auf dem Tourplan als Information für die Fahrer ausgedruckt werden.

Im dritten Teil werden die Dienstwünsche des Kunden aufgelistet. Hier können Dienstwünsche eingefügt, geändert oder gelöscht werden. Ein Beispiel für einen Dienstwunschdialog ist in Abbildung 9 zu sehen.

Nach der Eingabe der Dienstwünsche können diese zu Touren gruppiert werden. In einem Tour-Dialog (Abb. 5) wird zunächst eine Nummer und eine Bezeichnung für die Tour vergeben. Mit dem nächsten Feld kann die Tour für folgende Optimierungen gesperrt werden. Als nächstes werden dann die Mitarbeiter, die die Tour fahren sollen, und das entsprechende Fahrzeug gewählt. Im mittleren Teil der Maske werden die Dienstwünsche ausgewählt, von denen dann im unteren Teil noch einige zusammengefasste Daten angezeigt werden.

Die Touren sind nur Mengen von Dienstwünschen, die Reihenfolge der Stationen und die Zeiten werden in Untertouren festgelegt. Hin- und Rückfahrten einer Tour sind somit zwei Untertouren. Abbildung 6 zeigt einen Dialog zur Eingabe von Untertouren. Die Anfangs- und Endstationen können z.B. für eine

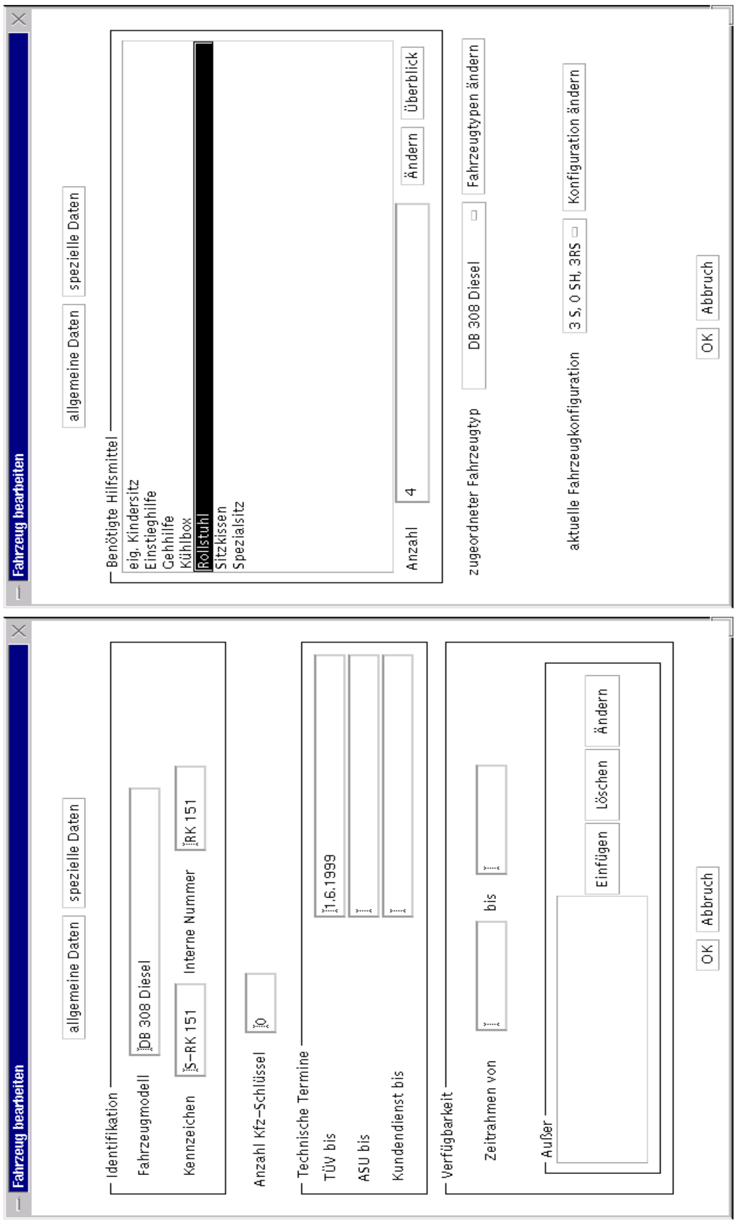

Abbildung 2: Eingabemasken für die Daten zu einem Fahrzeug: Bezeichnungen, Termine, Hilfsmittel, Fahrzeugtyp und Konfiguration

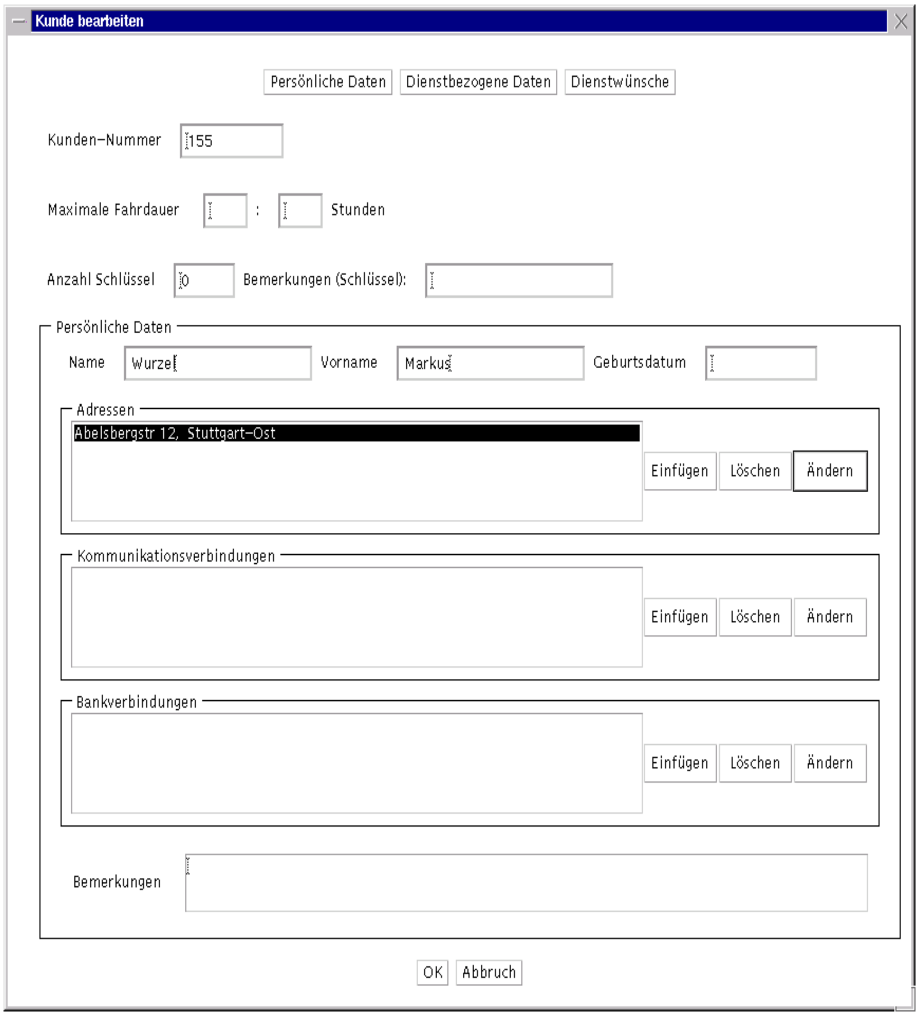

Abbildung 3: Kundendaten: persönliche Daten

tourübergreifende Planung der Fahrzeuge verwendet werden. Der untere Teil des Dialogs gibt die anzufahrenden Stationen mit den jeweiligen Zeiten an. Hier können dann auch Stationen vertauscht oder Zeiten geändert werden.

Damit sind dann alle Eingaben beendet. Der TROSS-Benutzer kann nun Einsatzpläne ausdrucken (Abb. 8) oder Analysen vornehmen (Abb. 10, 11), um seine Planung zu optimieren.

2 Bisherige Verwendung beim DRK

Ein DRK-Mitarbeiter hat damit angefangen, Stammdaten (Fahrzeuge, Mitarbeiter und Kunden) in das System einzugegeben. Es wurden leider nicht die von der Projektgruppe angelegten "Testdaten" verwendet. Diese Daten dienten der Projektgruppe zwar auch zum Testen, sie enthielten jedoch reale Daten vom DRK mit allen Mitarbeitern und Fahrzeugen sowie den Schultouren mit ihren

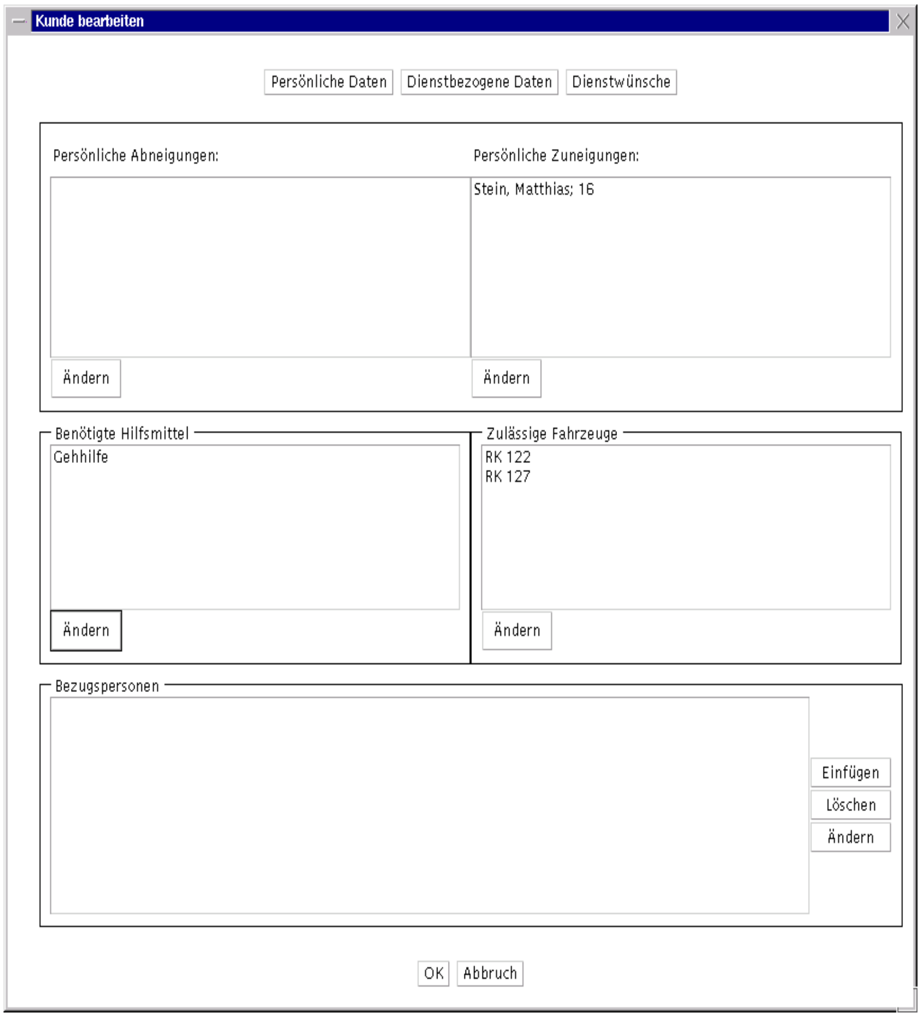

Abbildung 4: Kundendaten: Dienstbezogene Daten

Kindern.

Tabelle 1 listet die getätigten Eingaben auf, wobei nicht immer klar zwischen den eingegebenen Daten und den zu spät doch noch eingefügten Testdaten unterschieden werden konnte. Insbesondere wurden nur bei fünf Kunden Dienstwünsche eingegeben, wobei jeweils der Rhythmus fehlte, der die Zeiten festlegt (in Abb. 9 ganz unten). Von den eingegebenen Touren hatte nur die erste Tour eine Menge von Dienstwünschen, bei den anderen Touren waren nur der Name, die beteiligten Mitarbeiter und ein entsprechendes Fahrzeug vergeben.

Nach schätzungsweise mindestens fünf Stunden wurden die Versuche wegen den im folgenden Abschnitt beschriebenen Fehlern und Systemabstürzen abgebrochen, ohne dass eine Tour vollständig mit ihren Untertouren eingegeben wurde. Die Unterstützung der ersten Schritte bei der Verwendung des Systems ist anscheinend zu gering. Eine Benutzereinführung für TROSS wäre hier sinnvoll gewesen.

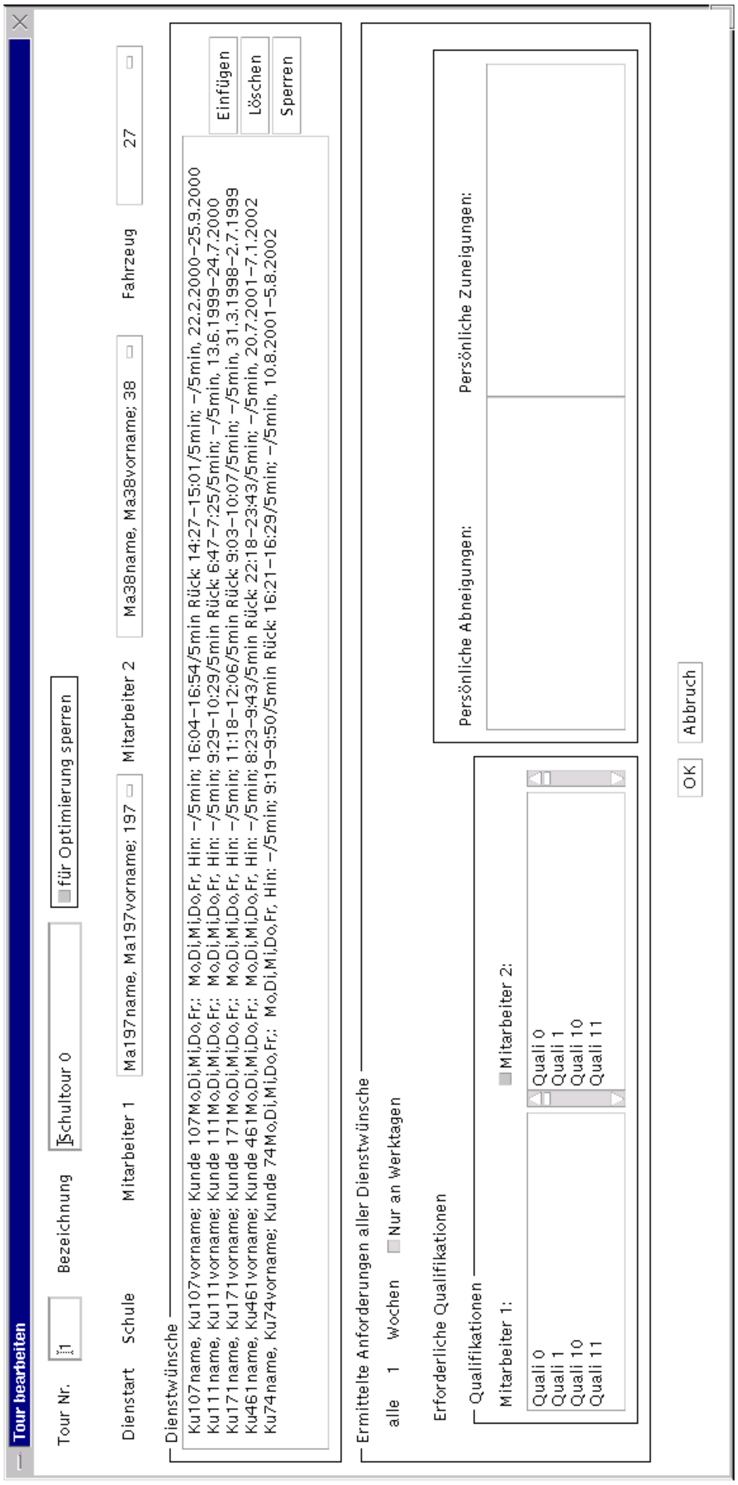

Abbildung 5: Eine Tour

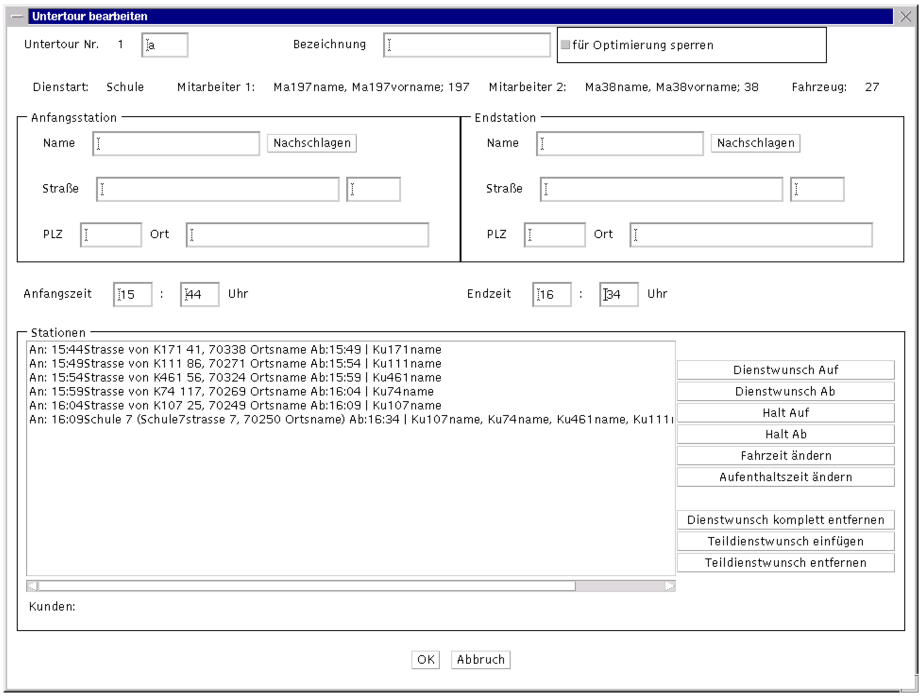

Abbildung 6: Eine Untertour

Das Verkehrstool Map&Guide, das zur Ermittlung der Fahrzeiten verwendet wird, ist nicht installiert. Da noch keine Untertouren eingegeben wurden, wurden Fahrzeiten von TROSS auch noch nicht benötigt.

3 Fehlerbeseitigung

3.1 Doppelte Einträge in den Listen

Wie in Abbildung 7 zu sehen ist, traten Einträge mehrfach auf. Der Grund dafür ist der Menüpunkt *Testdaten aufnehmen*, der in der Endversion von TROSS nicht enthalten sein sollte. Der Fehler wurde durch Beseitigung der doppelten Einträge und die Entfernung des Menüpunktes behoben.

3.2 Fahrzeuge ohne korrekte Nummer

In der Fahrzeugliste waren drei Fahrzeuge enthalten, deren Fahrzeugnummern leer waren (die Lücke oben in der Liste in Abbildung 7). Es stellte sich heraus, dass Aufrufe von Methoden der Klasse *TRO.Konsistenztests.FahrzeugPruefer* konsequent vergessen wurden. An ihrer Stelle standen nur Kommentare, die auf die offenen Arbeiten hinwiesen. Die Aufrufe wurden eingefügt und in der Folge aufgetretene kleinere Fehler an diesem Konsistenzprüfer wurden beseitigt.

Datentyp	Anzahl
Fahrzeugtypen	15
Fahrzeuge	55
Hilfmittel	7
Essensarten	17
Qualifikationen	0
Kunden	50
Dienstwünsche	5
Touren	43
Untertouren	0

Tabelle 1: Anzahl der eingegebenen Daten je Datentyp

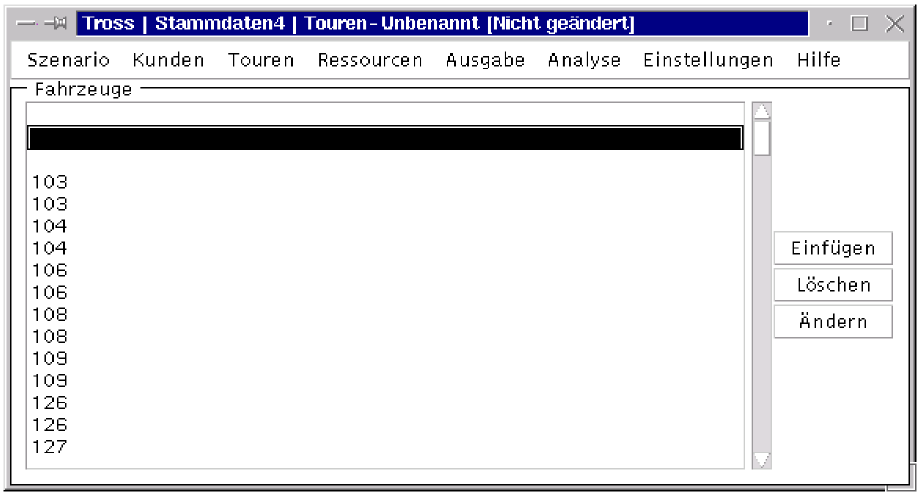

Abbildung 7: Die fehlerhafte Fahrzeugliste: ganz oben drei Fahrzeuge ohne Nummern gefolgt von mehreren doppelten Fahrzeugnummern

3.3 Löschen von Fahrzeugen funktioniert nicht

Der DRK-Mitarbeiter versuchte die Fahrzeuge mit den leeren Nummern zu löschen. Dies klappte zwar zunächst in der angezeigten Liste, jedoch waren die gelöschten Fahrzeuge nach einem erneuten Aufruf der Fahrzeugliste wieder da.

Der Grund für dieses Verhalten ist eine doppelte Speicherung: Java verwendet für die Anzeige eine eigene Liste, die unabhängig von den TROSS-Daten verwaltet wird (es wurden die AWT-GUI-Klassen und nicht Swing verwendet, da es Swing damals noch nicht gab). Die erste Vermutung war daher, dass der Löschaufruf für die Daten vergessen wurde. Die Methode für das Löschen wurde aber durchaus an den betroffenen Stellen im Programm aufgerufen.

Um das Problem zu erklären, zunächst ein kurzer Einblick in die Datenhaltung: In der Klasse *TRO.Szenario* werden alle Daten sortiert nach ihrem Listentext gespeichert (bei Fahrzeugen ist das z.B. die Fahrzeugnummer). Wird nun in einem Dialog der Listentext verändert, ändert sich somit auch der Sortierschlüssel. Jedoch wurde diese Tatsache übersehen und die Daten wurden somit auch nicht neu sortiert. Die Folge davon ist, dass ein Eintrag bei binärer Suche nicht mehr gefunden werden kann. Beim Löschen dieses Eintrags wurde dieser dann nicht gefunden und auch keine Fehlermeldung zurückgegeben.

Die falsche Sortierung der Daten ist nicht aufgefallen, da die Daten vor der Anzeige immer nochmals sortiert werden.

Betroffen von diesem Problem waren beinahe alle Daten: Mitarbeiter, Kunden, Institutionen, Qualifikationen, Fahrzeuge, Fahrzeugtypen, Essensarten, Hilfsmittel, ArbeitszeitProfile und TourSzenarien.

Die einfache Lösung wäre gewesen, die Daten nicht mehr sortiert zu speichern. Der Aufwand für die Suche nach einem Eintrag wäre dann zwar größer (im Mittel $n/2$ statt $log(n)$ bei binärer Suche), jedoch wären das einfache Zeiger-Vergleiche und nicht Vergleiche von Zeichenketten (der Listentext), die jedes Mal neu erstellt werden müssen. Diese Möglichkeit schied jedoch aus, weil die eingegebenen Daten, die mit der Serialisation von Java gespeichert wurden, dann nicht mehr verwendbar wären. Daher werden die Daten jetzt nach Änderung eines Schlüssels mit Shaker-Sort neu sortiert. Dies passiert in maximal $2n$ Schritten, wobei n die Anzahl der vorhandenen Daten des jeweiligen Typs ist. In unserer Anwendung ist $n \leq 1000$.

3.4 Leere Ausdrucke

Zum Teil waren die Ausdrucke ziemlich leer. Zum Beispiel erschien beim Versuch, die angezeigte Kundenliste auszudrucken, nur ein Rahmen mit dem Titel "Kunden". Das liegt am Menüpunkt *Ausgabe-Angezeigten Plan drucken...*. Dieser schickt einfach das aktuelle Hauptfenster, z.B. den angezeigten Plan, an den Drucker. Jedoch wurde hier die Methode *print* anstatt *printAll* verwendet, wodurch immer nur die oberste Komponente in der Fenster-Hierarchie angezeigt wurde. Bei einem Fenster mit Rahmen ist das eben nur der Rahmen. Es sei auch noch erwähnt, dass der Menüpunkt *Ausgabe-Angezeigten Plan drucken...* eigentlich nur zum Ausdrucken von Plänen aus dem *Ausgabe*-Menü gedacht ist und nicht für beliebige Fenster, wie z.B. die Liste der Kunden.

Beim Ausdrucken von Tourplänen erschienen Blätter, die nur eine Zeile enthielten:

```
Tour 1 Schule 1
```

Das ist zwar nicht besonders schön aber richtig so, da die "Tour 1" noch keine Untertouren hat. Hätte die Tour Untertouren, würden diese im Ausdruck folgen.

3.5 Sanduhr-Cursor bleibt stehen

Gelegentlich blieb der Sanduhr-Cursor auch nach dem Beenden einer Aufgabe noch stehen, so dass man den Eindruck hatte, das Programm arbeitet noch. Weshalb es dazu kommt, war nicht genau festzustellen. Das System wurde so abgeändert, dass beim Zurücksetzten der Form des Mauszeigers nicht mehr die vorher gespeicherte Form, sondern direkt der normale Cursor (Pfeil) angezeigt wird.

3.6 Abstürze des Systems

Diese waren nicht reproduzierbar, was vieleicht daran gelegen hat, dass Java unter verschiedenen Betriebssystemen verwendet wurde. Jedoch darf ein Java-Programm auch nicht mit der berühmten Windows-Meldung "Fehler in Anwendung" abstürzen. Dies ist nur durch Fehler in der Java-Maschine möglich.

3.7 Falsche Eingaben

In den eingegebenen Daten war zwar wie geplant eine benannte Station für die Schule vorhanden, jedoch wurden in verschiedenen Dienstwünschen neue Stationen nur mit dem Straßennamen der Schule angegeben. Wenn man aber nicht immer die gleiche (benannte) Station wählt, geht das Programm beim Aufbau von Untertouren von verschiedenen Stationen aus und fügt diese dann auch mehrfach in die Untertour ein. Zur Behebung muß zunächst in den Dienstwünschen die benannte Station ausgewählt werden. Dann kann die Untertour erneut angelegt werden.

Schöner für den Anwender wäre es natürlich, wenn TROSS bei gleichen Strassennamen nachfragt, ob die schon vorhandene Station verwendet werden soll. Solche Mechanismen wurden während der Entwicklung von TROSS als Eingabehilfen oder "nice" bezeichnet, dann aber aufgrund von Zeitnot nicht implementiert.

4 Ein zweiter Anlauf

Die vom DRK-Mitarbeiter eingegebenen Daten wurden bereinigt und zusammen mit einer neuen Version von TROSS beim DRK installiert. Es wurde eine Tour komplett mit ihren vier Untertouren angelegt. Dabei entdeckte die Konsistenzprüfung nicht eingehaltene Forderungen eines Dienstwunsches, die dann aber vom DRK-Mitarbeiter akzeptiert wurden.

Auch die Ausgabe von Dienstplänen (Abb. 8) und Analysedaten wie z.B. die Mitarbeiterauslastung (Abb. 10) funktionierten einwandfrei.

5 Oberfläche und Navigation

Die Navigation im System TROSS funktionierte reibungslos, die Anordnung der Oberflächenelemente ist stellenweise jedoch nicht optimal. Ein Beispiel ist der Dienstwunsch-Dialog (Abb. 9): Die wichtigsten Informationen zu einem Dienstwunsch, die Rhythmen, die die Wiederholungsrate und die Termine festlegen, befinden sich erst auf der zweiten Seite und auch dort erst ganz unten. Das ist vermutlich auch der Grund dafür, dass der DRK-Mitarbeiter in seinen Daten nie Rhythmen eingegeben hat.

6 Funktionalität

Ein Problem kann sich aus dem Zusammenspiel von Untertouren und der Entfernungstabelle ergeben: Wird die Fahrzeit zwischen zwei Stationen einer Untertour verändert, so wird die neue Fahrzeit auch in der Entfernungstabelle gespeichert, die für alle Stationen aller Untertouren gilt. Diese Änderung wirkt sich

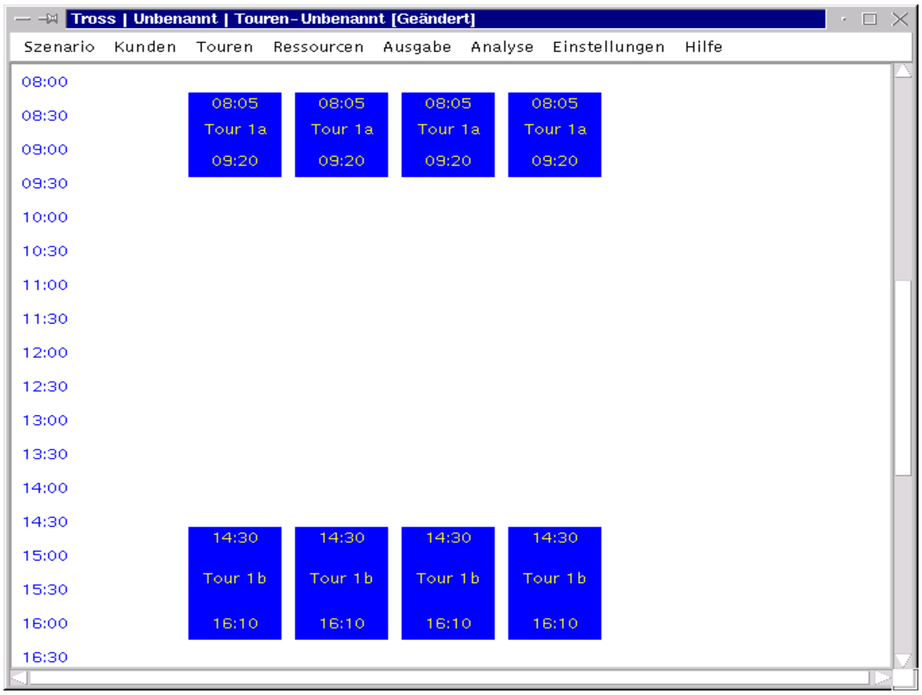

Abbildung 8: Dienstplan für einen Mitarbeiter

dann auch auf andere Untertouren aus, jedoch erst, wenn diese selbst bearbeitet werden. Das gilt nicht nur für die Untertouren einer Tour, sondern für alle Touren. Diese Anpassung zu einem späteren Zeitpunkt ist für den TROSS-Benutzer überraschend und nicht nachvollziehbar. Sinnvoller wäre es, die Veränderungen entweder sofort oder überhaupt nicht an andere Untertouren weiterzugeben. Diese Entscheidung sollte der Benutzer für jede betroffene Untertour selbst treffen können.

Ein weiteres Problem ist die fehlende Unterstützung für geplante geringfügige Veränderungen. Es ist z.B. nicht vernünftig möglich, einen Kunden ab einem bestimmten Datum in eine Tour aufzunehmen. Die Dienstwünsche der anderen Mitfahrer müsste man an diesem Tag aufspalten. Da es keine Funktion "Aufspalten" gibt, muss dazu der zukünftige Teil des Dienstwunsches neu eingegeben werden. Dann könnte man die Untertouren auf ähnliche Weise aufspalten. Hier fehlt ein Konzept zur Behandlung von Gültigkeitszeiträumen von Touren und Untertouren.

7 Kritik

Auf dem Rechner beim DRK Stuttgart, ein Pentium Pro 350 mit 32MB RAM und Windows 95, kann man mit dem Programm vernünftig arbeiten. Beide Mitglieder der Projektgruppe, die das Programm dort erlebt haben, waren von der Geschwindigkeit überrascht. Die Befürchtung bei der Wahl der Implementierungssprache, Java sei zu langsam, hat sich somit nicht bestätigt.

Ein Grund für den zurückhaltenden Einsatz des Programms beim DRK ist

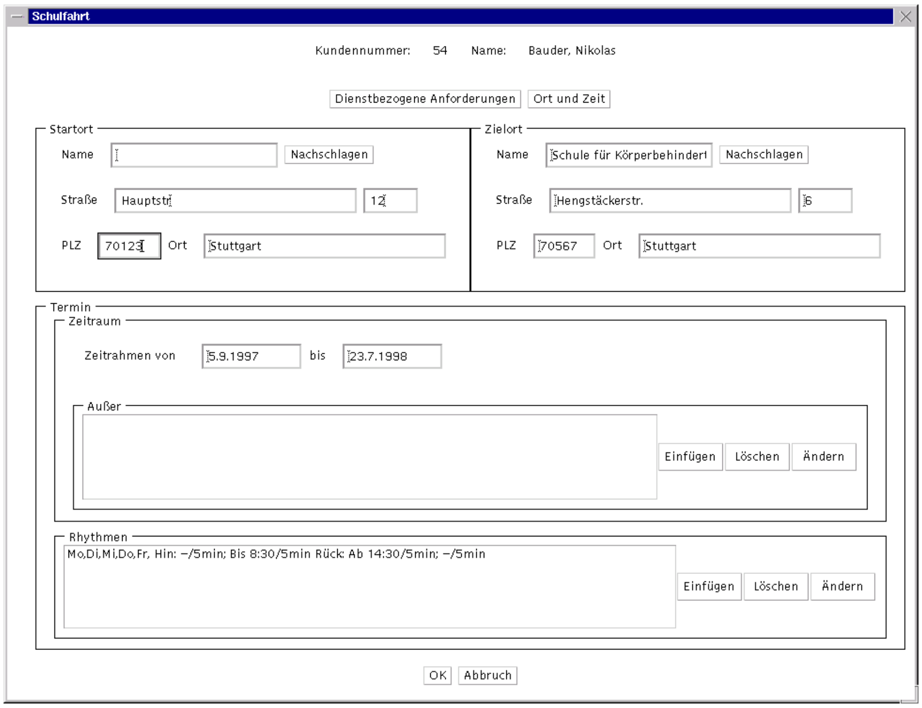

Abbildung 9: Der zweite Teil des Dienstwunschdialogs

die aufwendige Eingabe. Da zudem noch keine Optimierungsalgorithmen implementiert sind, ist der vom Benutzer erwartete Nutzen eher gering. Die einzigen neuen Erkenntnisse für ihn sind die Analysedaten wie Mitarbeiterauslastung (Abb. 10) und Fahrzeugbesetzungsgrad (Abb. 11).

Bei zukünftigen Projekten (mit externen Kunden) sollte auch darauf geachtet werden, dass der Einstieg für den Benutzer stärker unterstützt wird und nicht nur von einem laufenden Betrieb ausgegangen wird. Die Eingabe und Veränderung einzelner Daten ist in TROSS zwar vernünftig machbar, aber das Eingeben der vorhandenen Touren ist sehr aufwendig: zunächst die Kunden mit ihren Dienstwünsche eingeben, dann Touren und schließlich Untertouren erzeugen. Hier wäre eine direkte Eingabe der Tour und die Erzeugung von Dienstwünschen, Touren und Untertouren schön gewesen. Damit hätte sich der DRK-Mitarbeiter auch mit anderen Teilen des Systems befassen können. Unter der stundenlangen Eingabe der Daten und den dabei aufgetretenen Problemen hat die Motivation des DRK-Mitarbeiters stark gelitten.

8 Ausblick

Ein Problem für die Wartung und Erweiterung des Programms ist die Verwendung der Serialisation von Java zur Speicherung der Daten. Das war zwar sehr einfach zu implementieren, Veränderungen an der Struktur der Klassen machen die gespeicherten Daten jedoch unlesbar. Bei einer neuen Version sollte deshalb die Anbindung an eine Datenbank eingeplant werden.

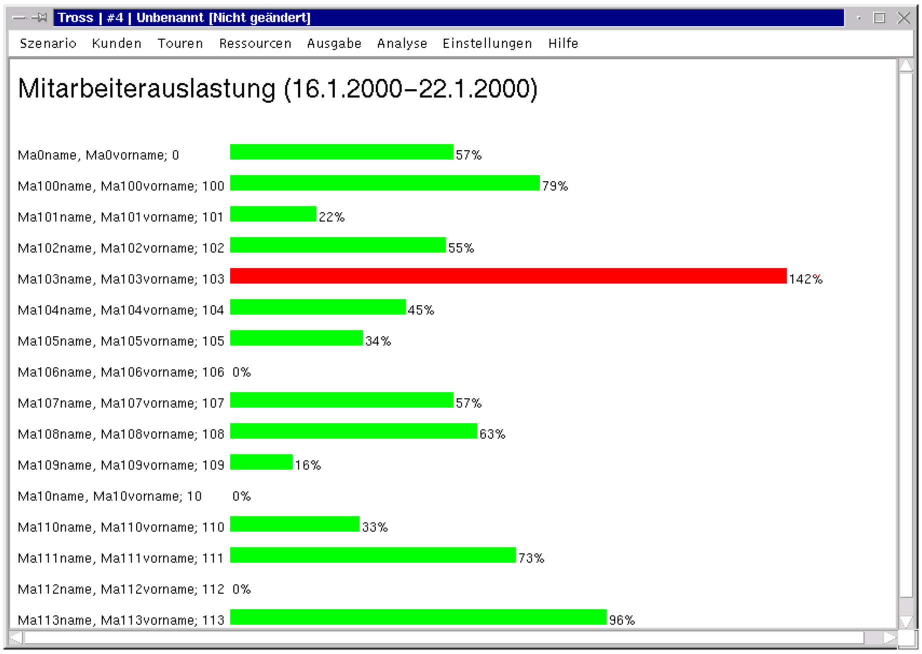

Abbildung 10: Auslastung der Mitarbeiter

Eine andere Erweiterung besteht darin, dem “O” in TROSS wieder zu seiner ursprünglichen Bedeutung zu verhelfen. Eigentlich stand es für “Optimierung”, als der Projektgruppe dann aber die Zeit ausging, wurde daraus “Organisation”. Interessant ist hier vor allem die automatische Zusammenstellung und Optimierung von Touren und Untertouren. Eine Optimierung der Fahrten, zum Beispiel bei Ausfällen von Mitarbeitern, ist zumindest beim DRK eher nicht sinnvoll, da die Kundschaft großen Wert auf Kontinuität legt.

Die Übertragung von TROSS auf andere Fahrdienste mit ähnlichen Aufgabenstellungen wäre sicher auch interessant (Schulbussysteme, Essensservice). Eine Erweiterung und Anpassung des Datenmodells und der Randbedingungen ist dann entsprechend erforderlich.

Literatur

[1] Balzert, Helmut: *Lehrbuch der Softwaretechnik*, Band 1: Software-Entwicklung. Spektrum-Verlag, 1996.

[2] Booch, G.: *Objektorientierte Analyse und Design*. Addison-Wesley, 1994.

[3] Fleischmann, Jörg, Hermes, Lars, Spribille, Tobias und Wagner, Frank: *Zwischenbericht der Projektgruppe Transportoptimierung*. Universität Stuttgart, Fakultät Informatik, Bericht Nr. 1998/06, 1998.

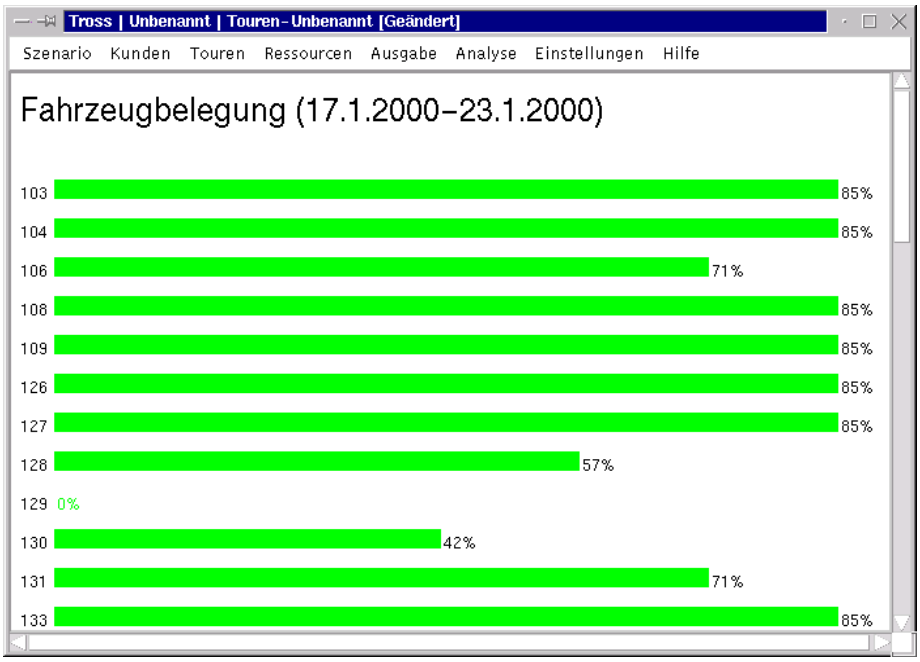

Abbildung 11: Fahrzeugbelegung

[4] Fleischmann, Jörg, Hermes, Lars, Spribille, Tobias und Wagner, Frank: *Endbericht der Projektgruppe Transportoptimierung.* Universität Stuttgart, Fakultät Informatik, Bericht Nr. 1998/10, 1998.

[5] Ottmann, T. und P. Widmayer: *Algorithmen und Datenstrukturen.* BI Wissenschaftsverlag, 1993.