Informatikprojekt

Verwaltung von Sun-Workstations

Bechter Silvia
Betreuung: Stiefenhofer Jürg

Zusammenfassung

Zur Verwaltung der Server und Workstations, welche der Systemadministration des IAM unterstehen, wurde ein Programm entwickelt, welches die System-Daten soweit als möglich direkt von den Maschinen abfragt und darstellt. Darstellung und Mutationen können sowohl von der graphischen Oberfläche als auch von einer Shell aus ausgeführt werden. Zusätzlich besteht die Möglichkeit, diverse Darstellungen im PostScript-Format auszudrucken. Als Programmiersprache wurde hauptsächlich Perl, für die Oberfläche Tcl/Tk verwendet.

Inhaltsverzeichnis

1 Einleitung

1.1 Projektbeschreibung

1.1.1 Problemstellung

Das System, welches die System-Administration zu warten und unterhalten hat, besteht aus 6 Solaris-Servern und ca. 150 Workstations. Jedes Jahr werden durchschnittlich 15-20 Geräte neu beschafft. Ungfähr ebensoviele Maschinen werden umgestellt oder ausgebaut. Das Inventar wird zur Zeit als EXCEL-Datei geführt. Viele dieser Angaben lassen sich aber direkt auf den SUN-Workstations abfragen, mit 2 Ausnahmen: Standort der Maschine (inkl. Fachgruppe und Benutzer) sowie die Garantienummer/Seriennummer von SUN. Die Angaben werden zur Zeit bereits für die „finger'"Datenbank, sowie für eine Konfigurationsabfrage der Workstations verwendet. Dieses Projekt soll das Verzeichnis der Workstations führen und auf einfache Art und Weise bedienbar sein. Insbesondere sollen alle Informationen, die direkt von den Workstations abgefragt werden können, von dort bezogen werden. Auf Verlangen soll die gesamte Liste auf den letzten Stand gebracht werden (Inventar). Es sollen verschiedene Reports generiert werden können. Das System soll ebenfalls alle Maschinenabgänge in eine separate Liste aufnehmen.

Das Programm soll folgende Ausgaben liefern können:

- Übersicht über alle Maschinen (Inventar).
- Übersicht über alle Maschinen einer gegebenen Fachgruppe (Inventarkontrolle).
- Interface zum Abfragen und Mutieren von Konfigurationsdaten (Name, IP, Ethernet, Memory, Disk, Seriennummer, Standort).

Die Benutzeroberfläche soll eine Liste sein, welche nach praktisch jedem Kriterium sortiert werden kann. Eine Maschine soll selbst eingetragen werden können, wenn sie nicht am Laufen ist. Inkonsistenzen von bestehenden Daten und neuen Informationen von den Maschinen sollen als Warnungen aufgelistet werden.

Die Abfrage der Daten einer speziellen Maschine sollte sowohl via graphisches Interface, als auch in Form eines Kommandozeilen-Programms gestartet werden können.

Die hauptsächlichen Projektziele sind:

- Erstellen einer Benutzeroberfläche mittels beliebiger Programmiersprache.
- Erstellung eines Command-Line-Programms mit sämtlichen Funktionalitäten.
- Erstellung eines Programms zum automatischen Nachführen von Systemdaten.

1.1.2 Spezielle Aspekte

Das Update sollte in der Lage sein festzustellen, ob die Workstations aktiv sind und demzufolge aktuelle Daten liefern können. Ist dies nicht der Fall, soll auf die alten Daten zurückgegriffen werden.

1.1.3 Abänderungen vor Projektbeginn

Ursprünglich war eine Anwendung für das WWW vorgesehen, welche mittels Java-Studio realisiert werden sollte. Jedoch wurde die Anforderung abgeändert, da ein graphisches Interface für Solaris schneller und nützlicher erschien.
Auch die Erstellung einer Datenbank wurde vor Projektbeginn verworfen, da der Aufwand und Overhead auf Grund der doch relativ geringen Datenmenge nicht gerechtfertigt erschien.
Zugangsbeschränkungen mittels Passwort wurden weggelassen, da sowieso nur berechtigte Personen Zugriff auf das Programm haben werden. Dafür wird beim Abändern von an und für sich unveränderbaren Daten eine Warnung angezeigt.

1.1.4 Anwendungsbereich

Das Programm war ursprünglich dazu gedacht, den Systemadministratoren die Möglichkeit zu geben, Daten über Sun-Workstations schnell und übersichtlich zu erhalten und gegebenenfalls zu verändern.
Zusätzlich sollte es Gruppen-Administratoren einen Überblick über ihre Maschinen geben. Diese Anforderung wurde im Verlauf des Projektes leicht abgeändert, so dass momentan das Programm nur für Systemadministratoren zur Verfügung steht.
Die Möglichkeit, nur die Maschinen einer Gruppe darzustellen oder auch (als Post-Script-File) auszudrucken, wurde jedoch implementiert.

1.2 Allgemeine Beschreibung der Werkzeuge/HW/SW

1.2.1 Hardware

Das Programm soll als Sun-Workstationverwaltung dienen. Eben diese Sun-Workstations standen als Hardware zur Verfügung.

1.2.2 Software

Gesamthaft wurde mit drei Programmier/Script-Sprachen gearbeitet. Der Hauptteil des Programmes (Command-Line-Programm) wurde in Perl implementiert. Zur Gewinnung der Systemvariablen der einzelnen Workstations werden noch zwei Script-Programme in Shell-Script aufgerufen. Alle die Oberfläche betreffenden Teile wurden in Tcl/Tk programmiert, von wo aus auch wieder Funktionen des Perl-Programmes aufgerufen werden.
Das Command-Line-Programm kann aus jedem beliebigen Terminal aus aufgerufen werden und hat die volle Funktionalität.

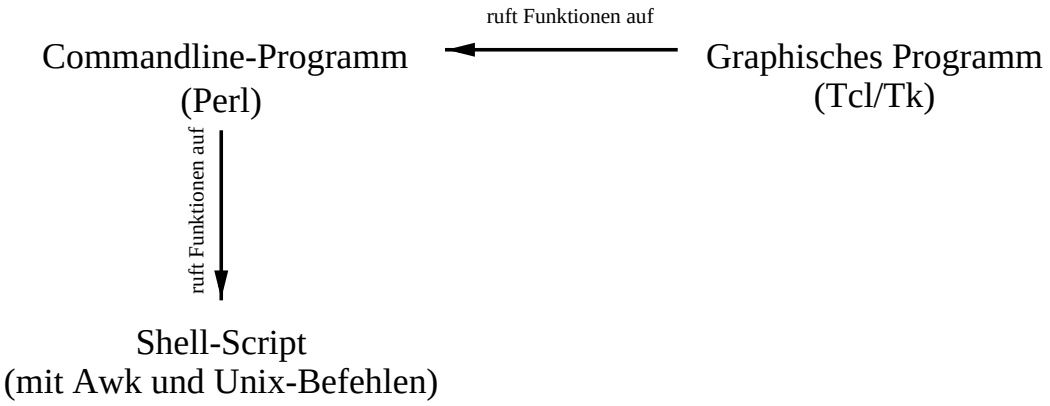


Abbildung 1: Software-Elemente

1.3 Zeitplan

1.3.1 Vorgegebener Zeitplan

Phase	Tätigkeit
1	Definition der Datenbank und des Interfaces
2	Design und Implementation eines Prototypen der Benutzeroberfläche
3	Design, Implementation und Tests der Hilfsprogramme
4	Übernahme und Adaption der bestehenden Inventardatenbank

- Phase 4 sollte nicht mehr als 20% der Projektdauer beanspruchen
- Geschätzter Gesamtaufwand: $>$ 200 Stunden

1.3.2 Selbstersteller Zeitplan

Phase	Tätigkeit	Geschätzter Aufwand
1	Einlesen in Script-Sprachen (Perl, C-Shell, Awk) und die UNIX-Umgebung, erste Testprogramme, Grobdesign	Einlesen vor Projektbeginn, ab Beginn noch 2-3 Tage
2	Erstellen des Prototyps für das Command-Line-Programm	1 Woche
3	Design der graphischen Oberfläche, Einlesen in Tcl/Tk, erste Testprogramme	1 Woche
4	Erstellen der graphischen Oberfläche, Testen des Command-Line-Programms	1 Woche
5	Fertigstellen, Testen, eventuelle Änderungen einbringen	1-3 Wochen

- Geschätzter Gesamtaufwand: 5-6 Wochen volle Arbeitszeit
- Nicht berücksichtigt: Projektbericht

1.3.3 Effektiver Zeitplan

Phase	Tätigkeit	Geschätzter Aufwand
1	Einlesen in Script-Sprachen (Perl, C-Shell, Awk) und die Unixumgebung, erste Testprogramme, Grobdesign	Einlesen vor Projektbeginn, ab Beginn noch 1-2 Tage
2	Erstellen des Prototyps für das Command-Line-Programm	$<$1 Woche
3	Design der graphischen Oberfläche, Einlesen in Tcl/Tk, erste Testprogramme	1 Woche
4	Erstellen der graphischen Oberfläche,	$>$ 2 Wochen
5	Integrieren und Testen des Command-Line-Programms	2-3 Tage
6	Fertigstellen, Testen	1-2 Wochen
7	Änderungen einbringen	wiederkehrend
7	Projektbericht	1-2 Wochen

- Die Programmierung der graphischen Oberfläche zog sich - wie erwartet - etwas in die Länge, dafür war der Prototyp des Command-Line-Programms eher fertig als erwartet.
- Sehr Zeitaufwendig waren die wiederkehrenden Änderungen, die sich aus dem existierenden Programm heraus ergaben: Neue Ideen, Teile besser zu implementieren, Code umschreiben/verbessern, etc.
- Nach Fertigstellung des Hauptprogramms zogen sich die Änderungen und der Projektbericht über einen längern Zeitraum hinaus, als Zeit ist die Gesamtarbeitszeit angegeben.

2 Design

2.1 Generell

Bei beiden Programmteilen wurde zuerst das Gesamtkonzept beschrieben oder entworfen, von welchem aus dann ein Prototyp implementiert wurde. Von da aus wur-

den Verfeinerungen und Verbesserungen ausgeführt. Programmteile, welche unabhängig von anderen Teilen funktionieren, wurden teilweise erst im Detail entworfen, implementiert und erst dann am Hauptprogramm angefügt (z.B. sun_entries und generate_ps in Perl oder Eingabefenster in Tcl/Tk). Daraus resultiert eine Mischung von Top-Down und Bottom-Up Design.

2.2 Command-Line-Programm

Für das Command-Line-Programm wurde kein spezifisches Design erstellt, sondern nur eine detailierte Spezifikation der Anforderungen, welche durch „stepwise refinement“ mehr oder weniger direkt implementiert werden konnte.

Abbildung 2: Design des Command-Line-Programms

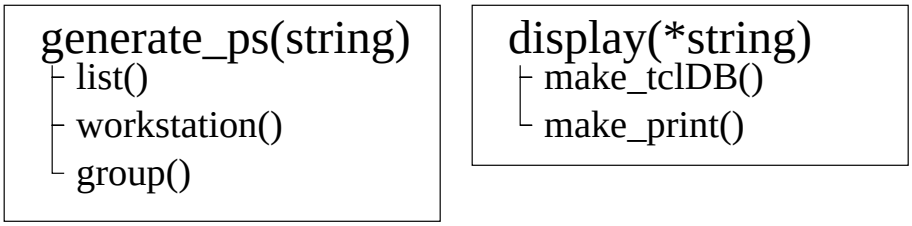


Abbildung 3: Hilfsprogramme zum Command-Line-Programm

2.3 Graphische Oberfläche

Zuerst wurde die generelle Auslegung der graphischen Oberfläche definiert. Diese wurde dann in Elemente unterteilt. Daraus wurde eine Tcl/Tk-Hierarchie erstellt.

Hierarchie:

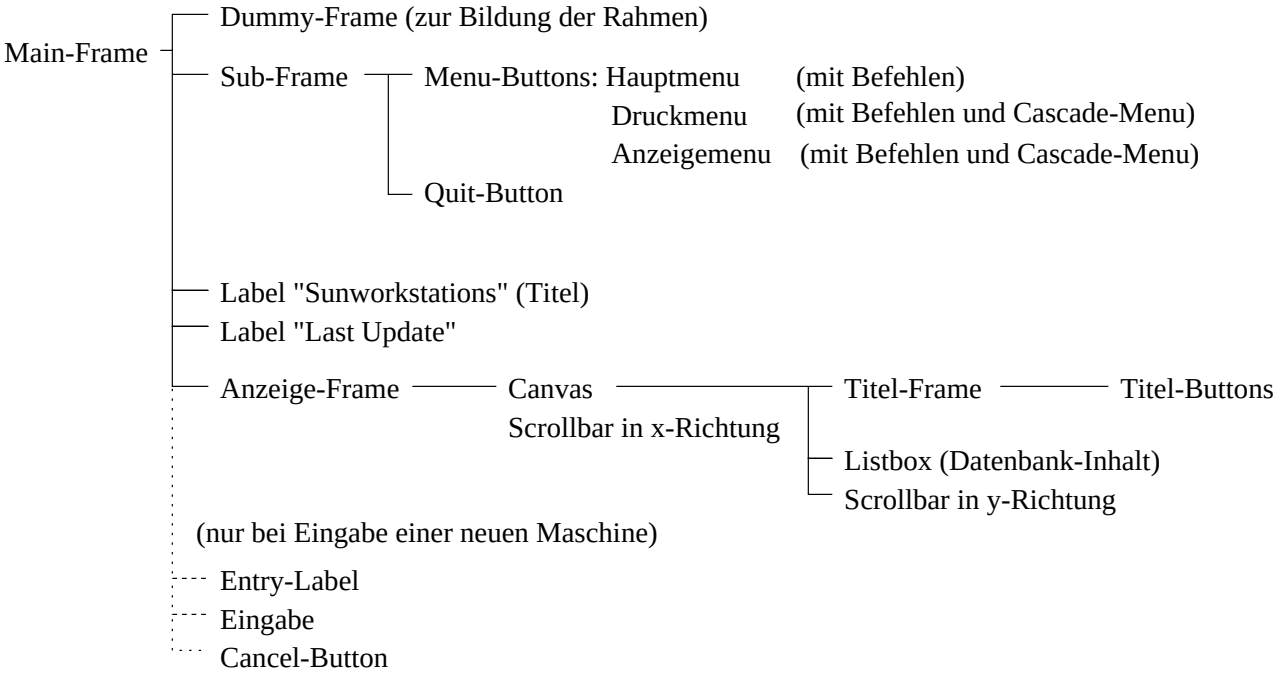


Abbildung 4: Hierarchie der Teilelemente

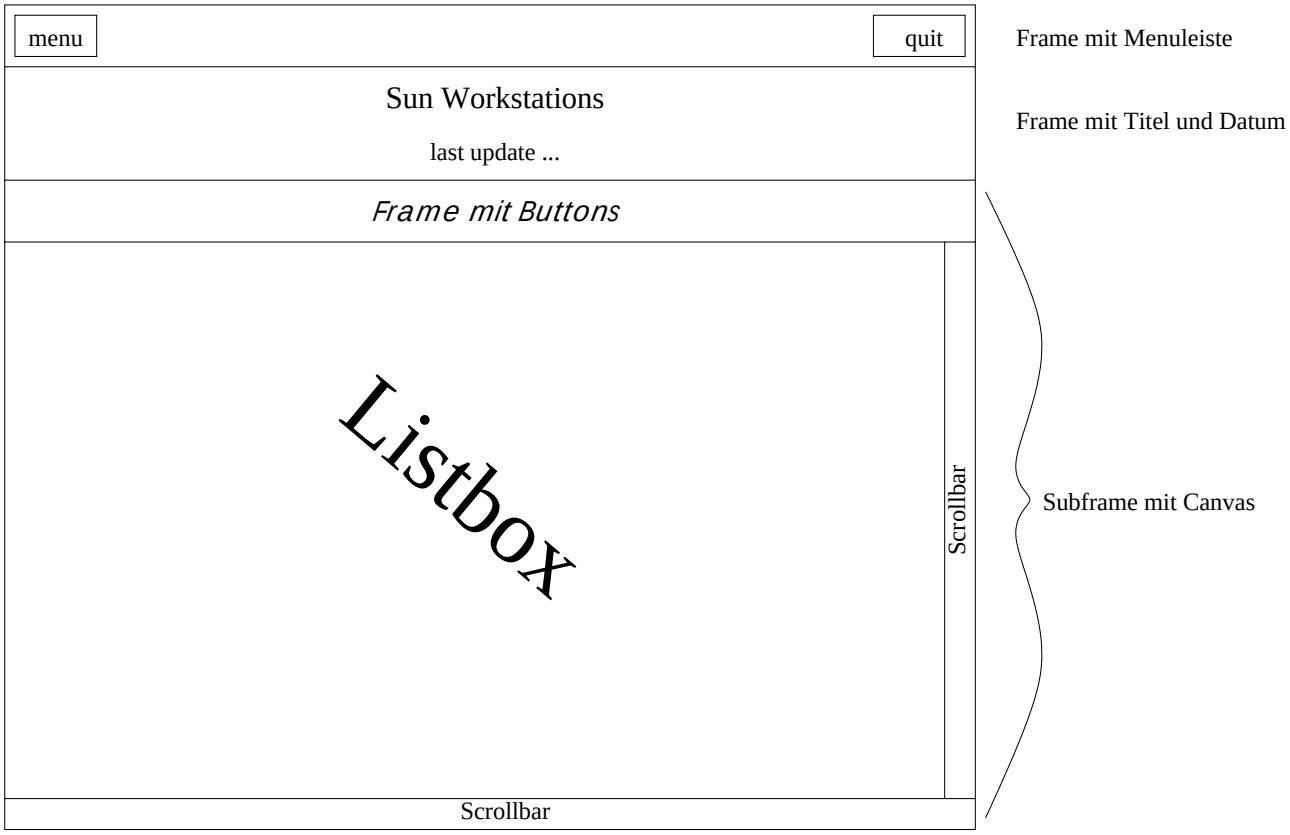


Abbildung 5: Hierarchie-Design der Oberflächenelemente

Für die einzelnen Unterfenster, welche von Hauptprogramm aus aufgerufen werden können, wurden spezielle Detailentwürfe erstellt, da bei Tcl/Tk die genaue Hierarchie der einzelnen Elemente bekannt sein muss.
Waren die Oberflächenelemente entworfen und implementiert, so wurden sie noch mit den entsprechenden Funktionen im Perl-Programm zusammengefügt.

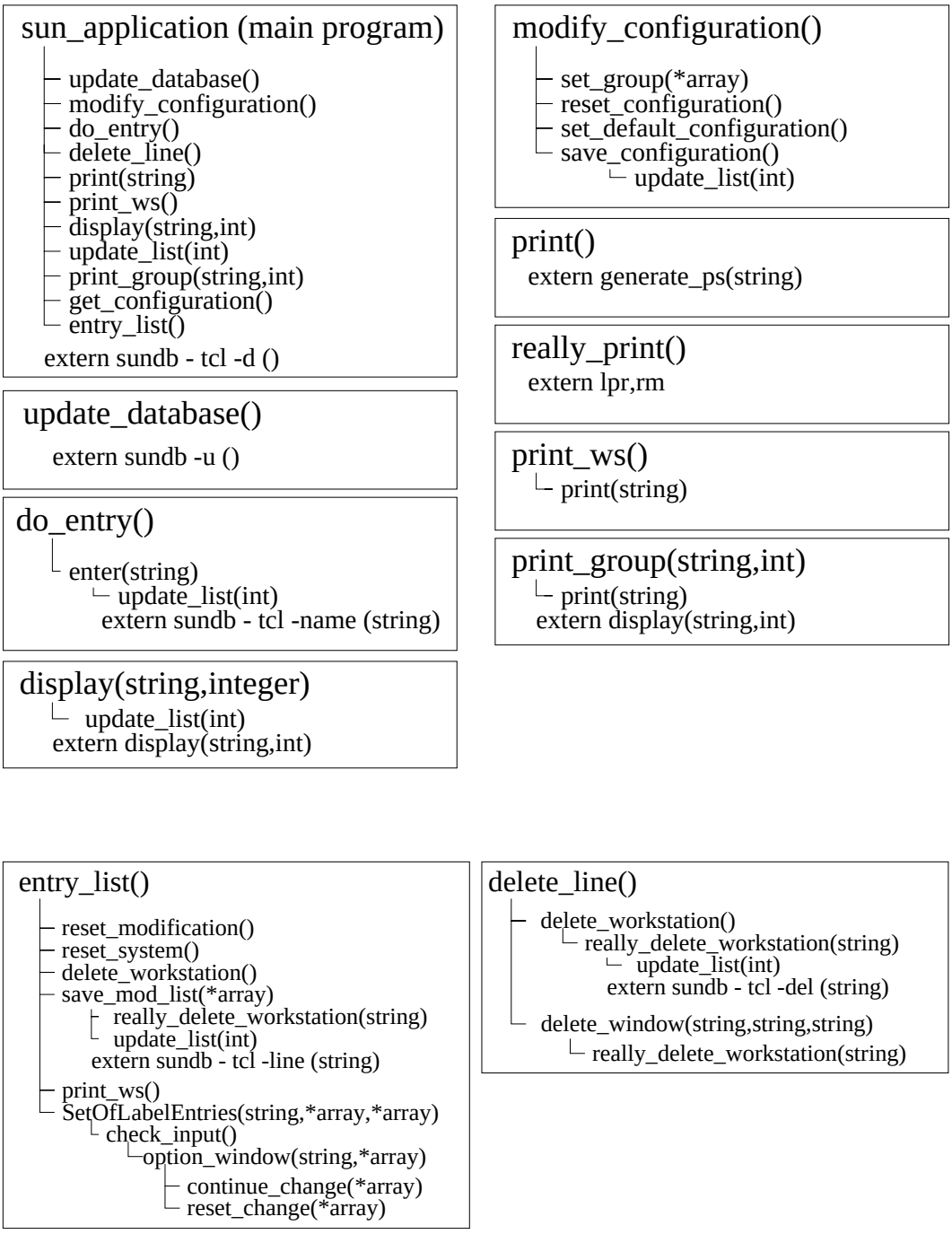


Abbildung 6: Design der Oberflächenfunktionen

3 Implementation

3.1 Übersicht der Funktionen

3.1.1 Command-Line-Programm

3.1.1.1 Allgemeine Beschreibung Teilaufgabe war es, ein oberflächenunabhängiges, von der Shell aus aufrufbares Programm zu schreiben, welches die volle Funktionalität bereitstellt.
Das resultierende Programm enthält folgende Teile:

- Update
- Eingabe von Daten (Workstations)
- Löschen von Daten (Workstations)
- Ergänzen von Daten
- Verändern von Daten
- Workstation in den Exit-Status setzen
- Hilfe
- Konvertierung des alten Datenbankformates in das neue Datenbankformat

Die einzelnen Teile werden über Programmname -Option aufgerufen. Ist der erste Programmaufruf falsch, wird als Hilfe die korrekte Syntax ausgegeben (“Usage: sundb -u [file] — -m “). Spezifische Optionen können während des Programmablaufes angegeben werden.
Temporäre Files werden im Directory /tmp/ abgespeichert. Auf Fehler oder eventuelle Änderungen in den Daten wird in der Command-Line hingewiesen, die expliziten Fehlerbeschreibungen werden in einem Log-File abgespeichert.

3.1.1.2 Voraussetzungen In der Regel ist bereits eine Datenbank im benötigten Format vorhanden. Aus Sicherheit wurden in einem separaten File alle Spaltenüberschriften abgespeichert, die bei einem Update jedesmal neu eingelesen werden. Dies erhöht auch die Erweiterbarkeit des Programmes, da beim Einfügen einer neuen Spalte nur dieses File erweitert werden muss. Beim folgenden Update werden automatisch alle Daten angepasst.

Ist keine Datenbank vorhanden, kann mit einer Liste der Workstationnamen (gespeichert in einem File, Namen getrennt durch Newline) und dem File mit den Spaltennamen einfach mittels Update-Funktion eine neue, vollständige Datenbank erstellt werden.
Ist eine Datenbank im EXCEL-Format vorhanden, kann sie mittels Konvertierungsprogramm umgewandelt werden.

3.1.1.3 Generelles Vorgehen Werden in der Datenbank Änderungen vorgenommen, so wird erst eine Sicherheitskopie erstellt, welche im Directory /tmp/ abgespeichert wird. Mit dieser Kopie wird bis zum erfolgreichen Abschluss der Funktion weitergearbeitet. Folgend wird von der Original-Datenbank eine Backup-Kopie erstellt (sun.db.bck) und die neu erstellte bzw. veränderte Datenbank wird mittels Unixbefehl „cp“ an die alte Stelle kopiert.
Dies geschieht aus Sicherheitsgründen (Konsistenz): Sollte das Programm vorzeitig unterbrochen werden (Control C, Crash, etc.), so bleibt der Zustand vor Funktionsbeginn erhalten.

3.1.1.4 Update Nach Erstellen der Sicherheitskopie wird mit der Unterfunktion „get_bootserver_info“ eine assoziative Liste mit den IP-Adressen der Workstations als Schlüssel und den entsprechenden Bootservern als Wert erstellt. Dazu wird auf alle Bootserver (hier: Haegar, Asterix und Obelix) eingeloggt und die entsprechenden Informationen werden ausgelesen.
Nun wird die neue Datenbank erstellt, indem erst das aktuelle Datum (über Unix) und dann die Titel eingelesen werden. Anschliessend wird nacheinander auf sämtliche in der Datenbank vorhandenen Maschinen eingeloggt, sofern das möglich ist. Ist ein Einloggen nicht möglich (keine Berechtigung, Maschine gibt keine Antwort), wird eine Fehlermeldung ausgegeben, der Grund im Log-File vermerkt und die Maschine übersprungen.
Das Einloggen erfolgt mit Hilfe eines externen Shell-Scripts. Nach dem Einloggen wird die gesuchte Information ausgelesen (Sysinfo, dmesg und df -k) und (auch im Shell-Skript) mittels NAWK selektiert und formatiert. Rückgabewert an das Perl-Programm ist ein String mit den Informationen, die aus den Workstations direkt ausgelesen werden können. Dort werden die Daten in die entsprechenden Positionen der Datenbank eingetragen.
Sind in Feldern, deren Inhalt nicht direkt von der Maschine ablesbar ist, noch Daten vorhanden, werden diese in die neue Datenbank übernommen. Andernfalls wird eine leerere Eintragung gemacht.

Sind alle Maschinen erfasst und ist die Datenbank kopiert, so wird die alte mit der neuen Datenbank-Version verglichen (Funktion „compare_databases“). Sämtliche Änderungen werden im Log-File vermerkt („In [Maschinenname]: [Alter Wert] has changed to [Neuer Wert]“).

3.1.1.5 Eingabe von Daten (Workstations) Zum Erfassen einer neuen Workstation gibt es zwei Varianten: Man kann einerseits das Programm Informationen selbst erfassen lassen (so weit dies möglich ist, Funktionsweise analog Update mittels rlogin) oder andererseits sämtliche Daten manuell eingeben. Dies ist eventuell für Maschinen nötig, die nicht am Netz hängen oder für separate Subnetze.
Der Benutzer bestimmt die Variante während des Programmdurchlaufs. Es können auch mehrere Neueingänge gleichzeitig eingetragen werden.
Die eingegeben/generierten Daten werden erst hinten an die temporäre Datenbank angehängt. Wenn keine weiteren Eingaben folgen, wird die Datenbank mit Hilfe der Unterfunktion “alphabetical_order„ sortiert. In dieser Unterfunktion wird eine Kopie via Unix-Befehl “sort„ alphabetisch sortiert. Die Kopie ist nötig, weil Datum und Titel nicht in den Sortieralgorithmus miteinbezogen werden dürfen.

3.1.1.6 Löschen von Daten (Workstations) Beim Löschen einer Workstation aus der Datenbank kann der Benutzer direkt den Maschinennamen eingeben. Lässt der Benutzer den Maschinennamen weg, so wird für jeden Eintrag abgefragt, ob die Daten zur zugehörigen Maschine gelöscht werden sollen. Die Datenbank wird während der(den) Abfrage(n) umgeschrieben; zu löschende Einträge werden einfach weggelassen.

3.1.1.7 Ergänzen von Daten Analog zum Löschen der Daten kann entweder ein einzelner Eintrag oder alle Einträge gleichzeitig vervollständigt werden. Auch hier wird jeder Eintrag in der temporären Datenbank ergänzt. Abgefragt werden nur leere Felder.

3.1.1.8 Verändern von Daten Analog zum Ergänzen kann die Funktion für eine oder alle Maschinen aufgerufen werden. Hier wird für jedes Feld der bestehende Eintrag ausgegeben und man hat die Möglichkeit, diesen zu ändern oder - falls noch kein Eintrag vorhanden ist - neu einzutragen. Der Benutzer wird also immer

abgefragt : “tom, IP-Adress is 130.92.66.32 Change to: „
Wird das Programm vorzeitig abgebrochen, verfallen die Änderungen.

3.1.1.9 Workstation in den Exit-Status setzen Diese Funktion markiert einen Eintrag auf zwei Arten: Sie stellt dem Namen der eingegebenen Workstation ein “zz„ voran und setzt das Exit flag (ein Eintrag in der Datenbank). Danach sortiert sie die Datenbank mit Hilfe der zuvor beschriebenen Funktion “alphabetical_order„.

3.1.1.10 Hilfe Man kann das Programm auch mit der Option -h aufrufen. Dabei wird ein kleines Helpfile ausgegeben, in welchem kurz die Programmoptionen beschrieben werden.

3.1.1.11 Datenkonvertierung des alten Datenbankformates in das neue Die ursprüngliche Datenbank war in einem anderen Format vorhanden. Es wurden sowohl neue Tabellenüberschriften eingeführt als auch die Reihenfolge der alten Tabellenüberschriften teilweise verändert. Damit nicht alles neu eingegeben werden muss, wurde eine Funktion zur Konvertierung geschrieben, welche das alte Format ins neue Datenbankformat konvertiert.

3.1.2 Graphische Oberfläche

3.1.2.1 Allgemeine Beschreibung Der zweite Teil des Projektes war, eine graphische Oberfläche zum Verwalten der Suns zu gestalten und zu implementieren. Da die Funtionalitäten bereits durch das Command-Line-Programm vorhanden waren, ging es hauptsächlich um die graphische Darstellung. Wieder wurde zuerst eine Hauptdarstellung entworfen und implementiert, andere Fenster und Details wurden nach und nach entworfen, implementiert und eingefügt.
Das Oberflächenprogramm ruft zur Bearbeitung der Datenbank die bereits existierenden Perl-Funktionen auf. Zusätzlich mussten jedoch noch Funktionalitäten, welche im Command-Line-Programm nicht benötigt wurden, implementiert werden. Dabei geht es nur um die Darstellung der Datenbank auf dem Bildschirm in diversen Variationen (sortiert oder selektiert nach verschiedenen Aspekten). Für die jeweiligen Darstellungsarten existiert auch noch eine Druckoption, welche die gewünschten Datenbankteile direkt in einem Post-Script-File zwischenspeichert und an den gewünschten Drucker weiterleitet.

3.1.2.2 Voraussetzungen Wie bereits beschrieben, greift die graphische Oberfläche auf die Funktionnalitäten des Perl Programms zurück. Da aber Tcl/Tk andere Einlesefunktionen mit anderen Standards zur Verfügung stellt als Perl, mussten die übernommenen Daten abgeändert und angepasst werden. Somit wird die Datenbank erst in das entsprechende Format umgespeichert und Tcl/Tk liest dieses neue File ein. Dieses Umschreiben wurde in Perl implementiert, da dieses bessere Ein- und Ausgabefunktionen zur Verfügung stellt.

3.1.2.3 Generelles Vorgehen Die spezielle Tcl Datenbank wird beim Neudarstellen auf dem Bildschirm nach jeder Änderung - ausser dem Sortieren - neu generiert. Sie wird abgespeichert als tcl.db.

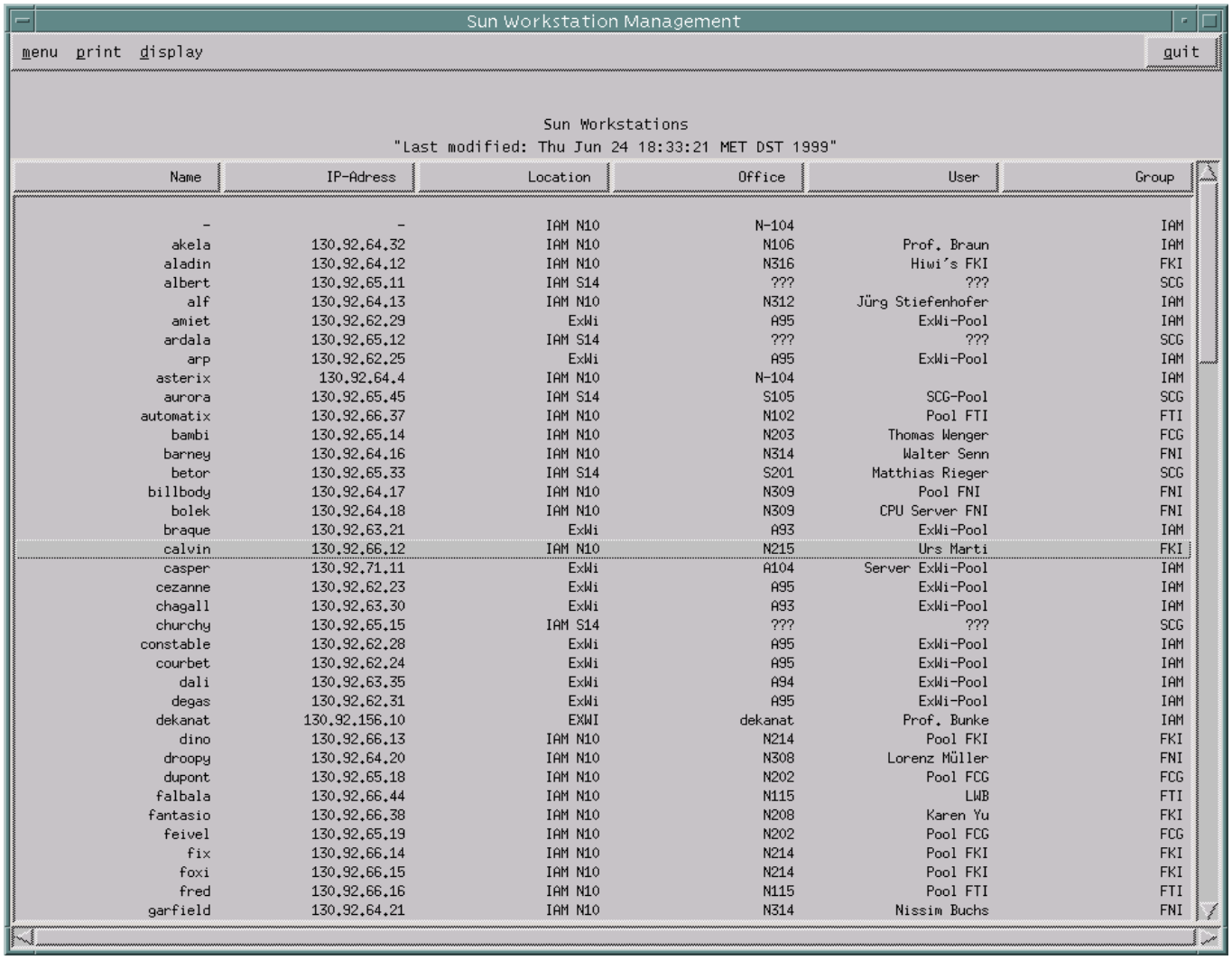
3.1.2.4 Main Window Das Hauptfenster besteht aus mehreren Unterteilen. Zuoberst befindet sich eine Menu-Leiste, von welcher aus alle Funktionen über die verschiedenen Menus bzw. Untermenus aufgerufen werden können.

Folgende Menus stehen zur Auswahl:

- Functions (Update, Modify Configuration, Enter Workstation, Delete Workstation)
- Print (Verschiedene Druck-Möglichkeiten)
- Display (Verschiedene Darstellungsarten)
- Quit

Unter der Menuleiste wird die Datenbank formatiert ausgegeben: Zuerst der Titel und das Datum der letzten Veränderung, darunter die Tabellenüberschriften als Button-Leiste und danach eine Liste mit den Einträgen.

3.1.2.5 Pop-Up Menu Klickt man mit der rechten Maustaste auf die Liste, erscheint ein Pop-Up-Menu mit den selben Einträgen wie das Menu „functions“ in der Menuleiste. Zusätzlich kann man damit noch das Programm verlassen.

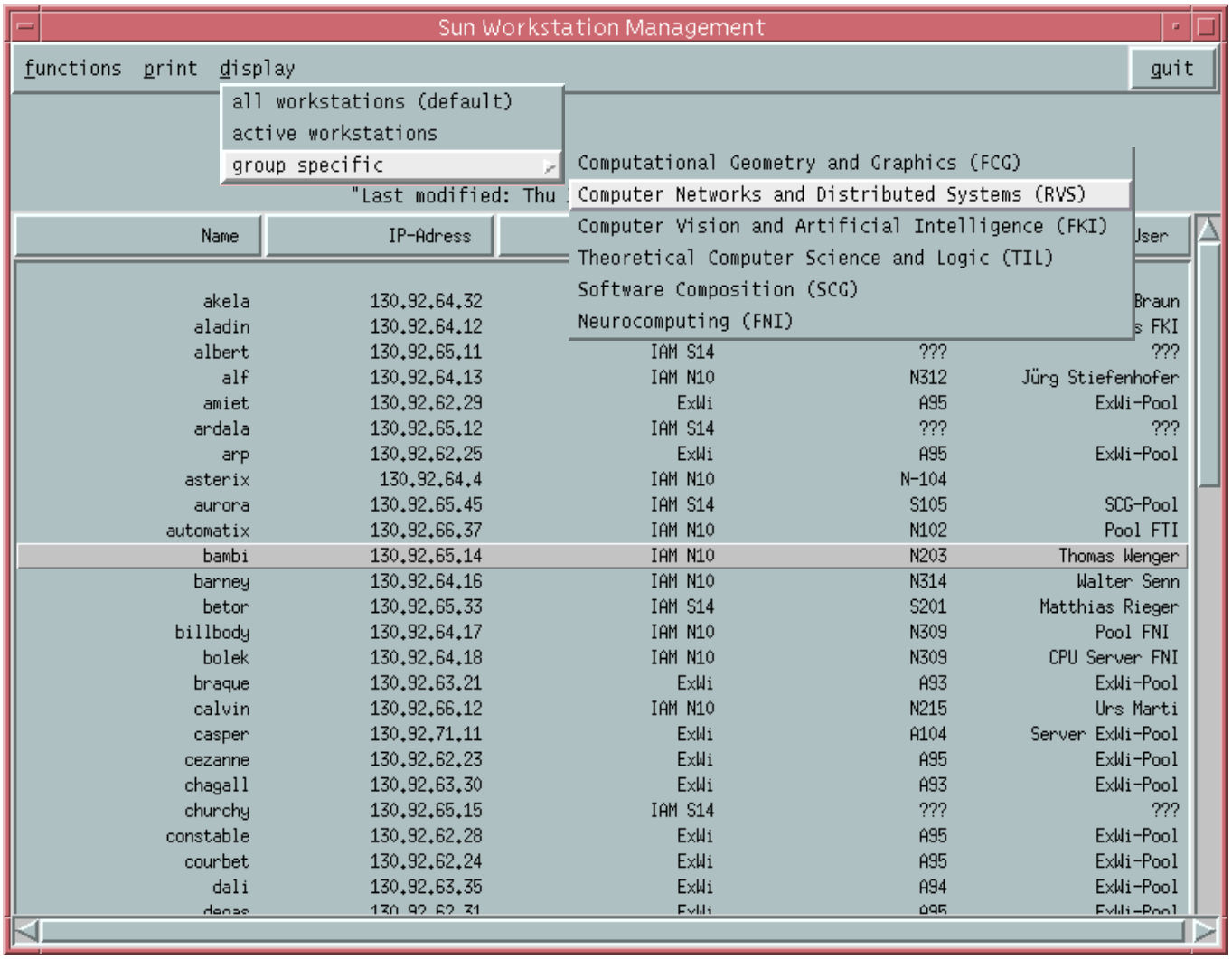


Name	IP-Adress	Location	Office	User	Group
-	-	IAM N10	N-104		IAM
akela	130.92.64.32	IAM N10	N106	Prof. Braun	IAM
aladin	130.92.64.12	IAM N10	N316	Hiwi's FKI	FKI
albert	130.92.65.11	IAM S14	???	???	SCG
alf	130.92.64.13	IAM N10	N312	Jürg Stiefenhofer	IAM
amiet	130.92.62.29	ExWi	A95	ExWi-Pool	IAM
ardala	130.92.65.12	IAM S14	???	???	SCG
arp	130.92.62.25	ExWi	A95	ExWi-Pool	IAM
asterix	130.92.64.4	IAM N10	N-104		IAM
aurora	130.92.65.45	IAM S14	S105	SCG-Pool	SCG
automatix	130.92.66.37	IAM N10	N102	Pool FTI	FTI
bambi	130.92.65.14	IAM N10	N203	Thomas Wenger	FCG
barney	130.92.64.16	IAM N10	N314	Walter Senn	FNI
betor	130.92.65.33	IAM S14	S201	Matthias Rieger	SCG
billbody	130.92.64.17	IAM N10	N309	Pool FNI	FNI
bolek	130.92.64.18	IAM N10	N309	CPU Server FNI	FNI
braque	130.92.63.21	ExWi	A93	ExWi-Pool	IAM
calvin	130.92.66.12	IAM N10	N215	Urs Marti	FKI
casper	130.92.71.11	ExWi	A104	Server ExWi-Pool	IAM
cezanne	130.92.62.23	ExWi	A95	ExWi-Pool	IAM
chagall	130.92.63.30	ExWi	A93	ExWi-Pool	IAM
churchy	130.92.65.15	IAM S14	???	???	SCG
constable	130.92.62.28	ExWi	A95	ExWi-Pool	IAM
courbet	130.92.62.24	ExWi	A95	ExWi-Pool	IAM
dali	130.92.63.35	ExWi	A94	ExWi-Pool	IAM
degas	130.92.62.31	ExWi	A95	ExWi-Pool	IAM
dekanat	130.92.156.10	EXWI	dekanat	Prof. Bunke	IAM
dino	130.92.66.13	IAM N10	N214	Pool FKI	FKI
droopy	130.92.64.20	IAM N10	N308	Lorenz Müller	FNI
dupont	130.92.65.18	IAM N10	N202	Pool FCG	FCG
falbala	130.92.66.44	IAM N10	N115	LWB	FTI
fantasio	130.92.66.38	IAM N10	N208	Karen Yu	FKI
feivel	130.92.65.19	IAM N10	N202	Pool FCG	FCG
fix	130.92.66.14	IAM N10	N214	Pool FKI	FKI
foxi	130.92.66.15	IAM N10	N214	Pool FKI	FKI
fred	130.92.66.16	IAM N10	N115	Pool FTI	FTI
garfield	130.92.64.21	IAM N10	N314	Nissim Buchs	FNI

Abbildung 7: Hauptfenster mit sechs ausgewählten Tabellen

3.1.2.6 Darstellung Beim Aufstarten wird defaultmässig wird die gesamte Datenbank angezeigt. Bei jeder Veränderung der Liste wird in der Regel die tcl-Datenbank neu erstellt. Bei gewissen selektiven Anzeigemodi (z.B. nach dem Sortieren) ist ein neues Generieren nicht nötig. Der entsprechenden Funktion „update_list“ wird dementsprechend ein Argument übergeben.

Über das Display-Menu können bestimmte Gruppen von Workstations separat dargestellt werden. Es sind folgende Gruppen fest implementiert:



akela	130.92.64.32			Braun
aladin	130.92.64.12			s FKI
albert	130.92.65.11	IAM S14	???	???
alf	130.92.64.13	IAM N10	N312	Jürg Stiefenhofer
amiet	130.92.62.29	ExWi	A95	ExWi-Pool
ardala	130.92.65.12	IAM S14	???	???
arp	130.92.62.25	ExWi	A95	ExWi-Pool
asterix	130.92.64.4	IAM N10	N-104	
aurora	130.92.65.45	IAM S14	S105	SCG-Pool
automatix	130.92.66.37	IAM N10	N102	Pool FTI
bambi	130.92.65.14	IAM N10	N203	Thomas Wenger
barney	130.92.64.16	IAM N10	N314	Walter Senn
betor	130.92.65.33	IAM S14	S201	Matthias Rieger
billbody	130.92.64.17	IAM N10	N309	Pool FNI
bolek	130.92.64.18	IAM N10	N309	CPU Server FNI
braque	130.92.63.21	ExWi	A93	ExWi-Pool
calvin	130.92.66.12	IAM N10	N215	Urs Marti
casper	130.92.71.11	ExWi	A104	Server ExWi-Pool
cezanne	130.92.62.23	ExWi	A95	ExWi-Pool
chagall	130.92.63.30	ExWi	A93	ExWi-Pool
churchy	130.92.65.15	IAM S14	???	???
constable	130.92.62.28	ExWi	A95	ExWi-Pool
courbet	130.92.62.24	ExWi	A95	ExWi-Pool
dali	130.92.63.35	ExWi	A94	ExWi-Pool
degas	130.92.62.31	ExWi	A95	ExWi-Pool

Abbildung 8: Hauptfenster mit Menu und Untermenu

- Aktive Workstations
- Alle Workstations (default)
- Fachgruppen (FCG, RVS, SCG, TIL, FNI, FKI)

Dies geschieht mittels einem Perl-Programms, welches mit dem Auswahlparameter ein „grep“ auf die Datenbank ausführt und das Ergebnis in die tcl.db schreibt. Danach wird die Liste neu angezeigt.

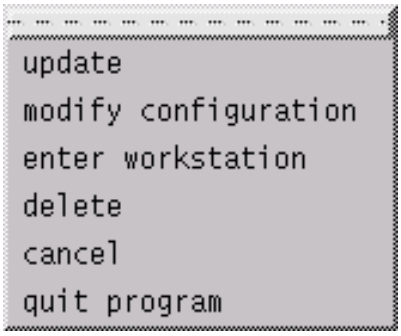


Abbildung 9: Popup-Fenster

3.1.2.7 Darstellungsform verändern Die Datenbank enthält über 20 Tabellen, welche nicht alle gleichzeitig auf dem Bildschirm angezeigt werden können. Via Scroll-Bar können zwar alle Tabellen erreicht werden, dies ist jedoch zu umständlich, weil so für die benötigten Informationen oft hin und her gescrollt werden muss. Meist aber werden nur wenige Spalten benötigt, zum Beispiel um Informationen über Hardware oder Inventar zu erhalten.
Dazu wurde die Möglichkeit implementiert, die Anzeige auf die gewünschten Tabellen zu beschränken. Die Anzeige kann durch die Funktion „Modify Configuration“ ausgewählt werden.
Im daraufhin erscheinenden Fenster können die gewünschten Tabellenüberschriften selektiert werden. Es besteht auch die Möglichkeit, ganze Bereiche miteinander zu selektieren. Defaultmässig sind alle Überschriften selektiert. Dies geschieht auch, wenn man auf den Knopf „set default“ drückt. Reset setzt den ursprünglichen Zustand wieder her und „save/apply“ speichert den veränderten Zustand ab und ändert die Ausgabe auf den Bildschirm entsprechend ab. Der Zustand wird in einem File (rpc.sdb) abgespeichert, so dass bei einem Neustart des Programmes automatisch der zuletzt benutzte Zustand wieder hergestellt wird.
Sollte dieses File nicht vorhanden oder schreib-/lesegeschützt sein, so werden alle Tabellen angezeigt.

3.1.2.8 Sortieren Man kann die Liste nach den Listeneinträgen sortieren lassen, indem man einfach den Knopf der jeweiligen Spaltenüberschrift drückt. Normalerweise geschieht das Sortieren alphabetisch, bei gewissen Spalten wie IP- oder Ethernetadressen jedoch numerisch.
Das Sortieren wurde als Perl-Funktion implementiert. Dies wiederum, weil Perl die besseren Hilfsfunktionen zur Verfügung stellt. Von dieser Perl-Funktion aus wird

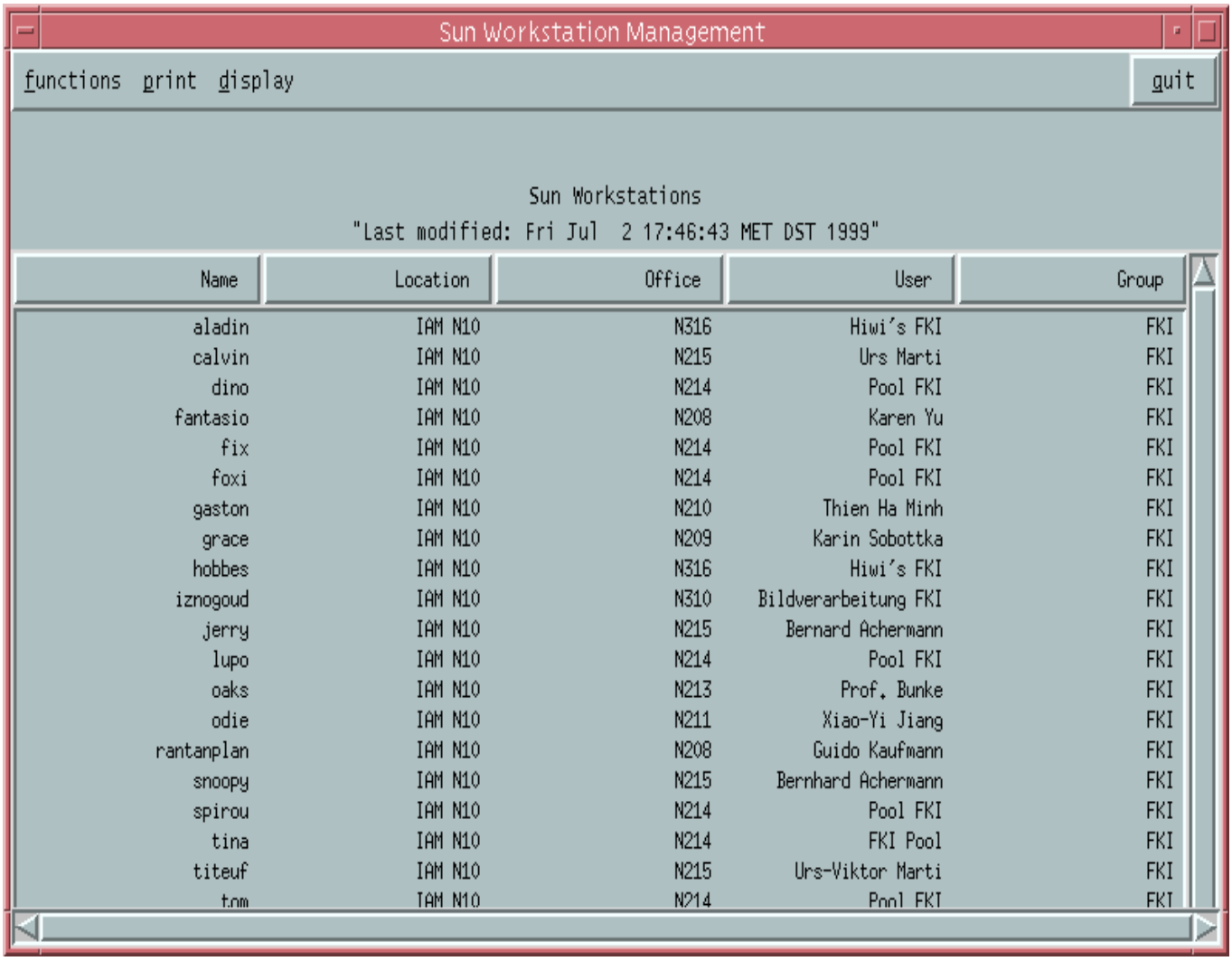


Abbildung 10: Gruppenspezifische Anzeige

- mit den der Spalte entsprechenden Parametern - die Unix-Sortierfunktion „sort“ aufgerufen. Das sortierte File wird wieder als tcl.db abgespeichert und die Liste wird im Hauptfenster neu ausgegeben.

3.1.2.9 Einträge bearbeiten Durch einen Doppelklick auf einen Eintrag erscheint ein Fenster mit den Angaben zur selektierten Maschine. Neben den Angaben befindet sich jeweils ein Eingabefenster mit dem aktuellen Eintrag. Dieser Eintrag kann

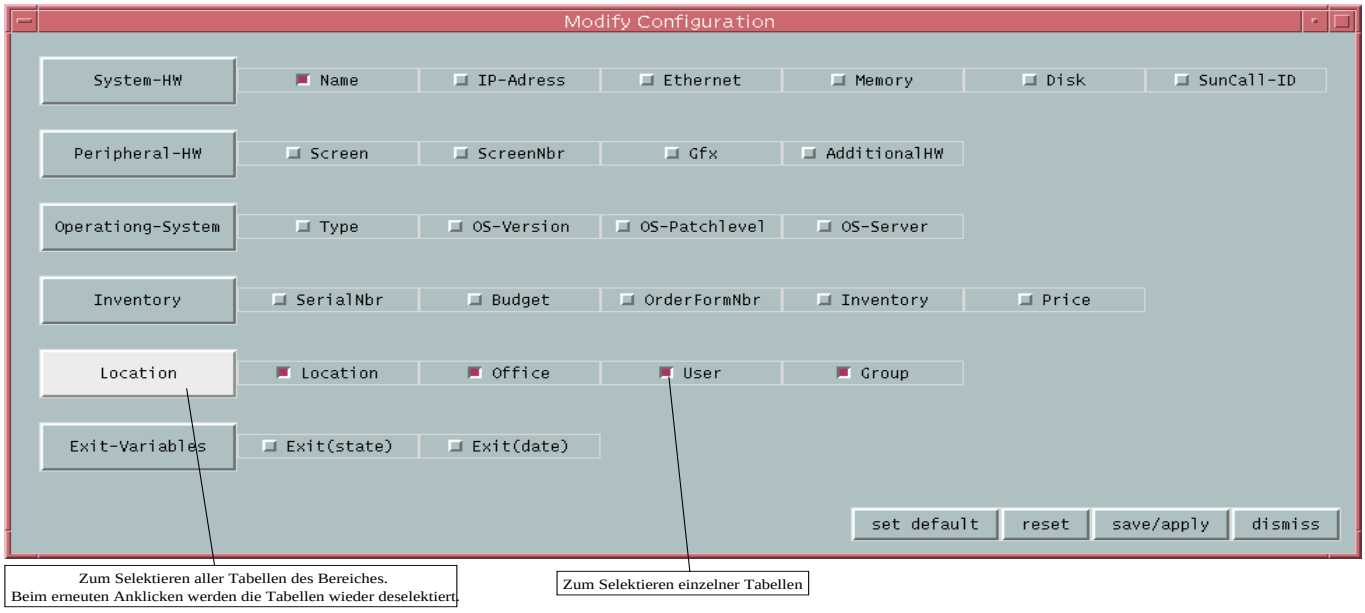


Abbildung 11: Fenster zum Verändern des Anzeigemodus

hier verändert werden. Systemabhängige Einträge, die direkt eingelesen werden, sind rot unterlegt, die anderen grün. Somit erscheint beim Verändern von Systemvariablen auch die Warnung, dass dieser Eintrag nicht verändert werden sollte. Diese Warnung kann jedoch ausgeblendet werden, sollte es nötig sein, mehrere Systemvariablen zu verändern.
Wird ein Eintrag verändert, wird das dem Benutzer angezeigt, indem der Hintergrund des jeweiligen Eingabefensters eine etwas hellere Farbe annimmt.
Mit den Auswahlknöpfen können nun die veränderten Daten weiter verarbeitet werden:

- „save/dismiss“ speichert die Veränderungen mittels der entsprechenden Perl Funktion in der Datenbank ab, das Fenster wird verlassen.
- „reset all“ gibt allen veränderten Einträgen wieder den ursprünglichen Wert.
- „reset system“ setzt die Systemvariablen wieder auf den ursprünglichen Wert.
- „delete“ löscht diese Workstation aus der Datenbank.
- „print“ gibt die Daten zur Workstation als Post Script File an den gewünschten Printer weiter. Allerdings müssen die Änderungen erst abgespeichert werden

Abbildung 12: Fenster zum Ändern von maschinenspezifischen Einträgen

um beim Ausdruck berücksichtigt zu werden.

Verlässt man das Fenster ohne abzuspeichern, gehen alle Änderungen verloren.

3.1.2.10 Workstation eingeben Will man eine neue Workstation eingeben, wählt man im Menu die Option „enter workstation“. Daraufhin erscheint unter der Liste ein Eingabefeld für eine neue Workstation. Gibt man den Namen ein, sucht das Programm nach dieser Workstation auf dem Netz. Ist die Workstation nicht am Netz

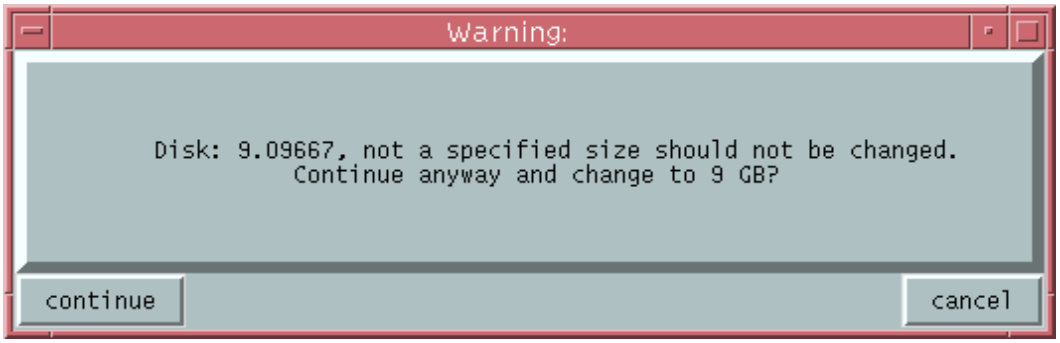


Abbildung 13: Warnfenster, welches beim Ändern der Systemvariablen erscheint. Hier wurde die Harddiskgrösse verändert (vgl. vorherige Abbildung).

angeschlossen, gibt es keinen Eintrag und eine Fehlermeldung erscheint.

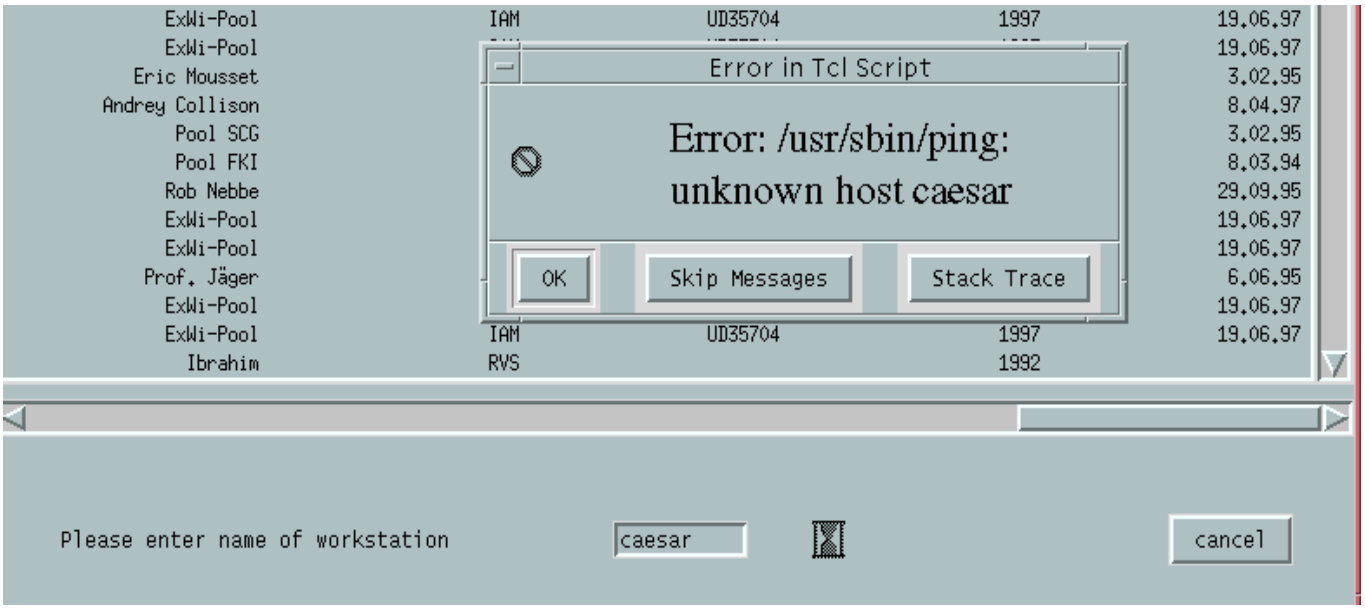


Abbildung 14: Ausschnitt aus dem Hauptfenster zur Eingabe neuer Workstations, hier mit Warnung, dass eingegebene Workstation nicht existiert.

3.1.2.11 Workstation löschen Will man eine Workstation aus der Liste löschen, muss man sie erst markieren. Danach kann man über das Menu oder über das Ände-

rungsfenster die Maschine löschen. Es erscheint ein Warnfenster, bevor der Eintrag definitiv aus der Datenbank gelöscht wird.

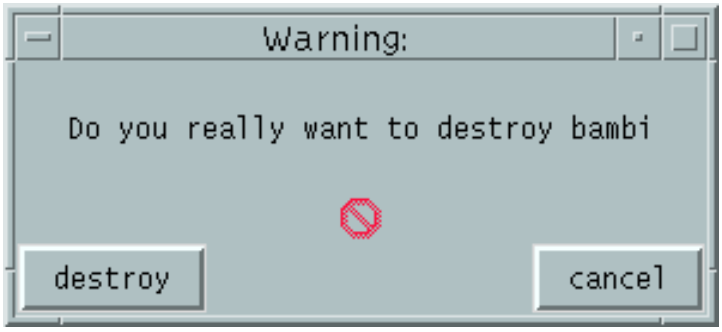


Abbildung 15: Warnfenster zum Verhindern versehentlicher Löschungen

3.1.2.12 Drucken Verschiedene Darstellungen können direkt vom Programm ausgedruckt werden. Gedruckt werden immer die Spalten der momentanen Darstellungsart. Folgende Varianten stehen zur Verfügung:

- alle Workstations
- nur die aktiven Workstations
- alle Worstations im ExWi
- nur die aktiven Workstations im ExWi
- eine markierte Workstation
- alle Workstations einer bestimmten Arbeitsgruppe

Dazu wird zuerst das Perl-Programm „display“ aufgerufen, welches die spezifischen Daten in einem File zwischenspeichert. Danach wird wieder ein Perl-Programm („generate_ps“) aufgerufen, welches eine Liste im Latex-Format generiert, diese kompiliert und ein dazugehöriges Post-Script-File erzeugt. Danach erscheint ein Fenster, in dem der Benutzer den Drucker auswählen kann. Defaultmässig wird ein Drucker angegeben, der im Programm definiert ist.
Ist der Druckvorgang abgeschlossen, werden die zuvor generierten Hilfsdateien wieder gelöscht.

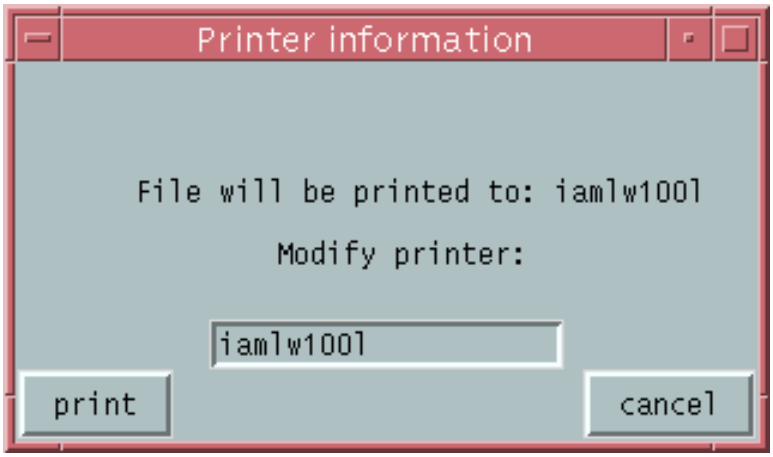


Abbildung 16: Fenster zum Ändern des Default-Printers oder zum Abbruch des Druckvorgangs

4 Schlussbetrachtung

4.1 Wahl der Software

Die Wahl der Software gestaltete sich als relativ schwierig. Gewisse Programmteile, die weiterverwendet werden konnten, standen als Shell-Skripts zur Verfügung. Darin waren auch Programme in Awk enthalten. Um diese Teile weiterverwenden zu können, mussten sie verstanden werden. Als Ergebnis wurden dann ähnliche Probleme in Shell- Skript und Awk implementiert.
Für das Kommandozeilen-Programm standen zwei Sprachen zur Auswahl: C/C++ und Perl.
Ich habe mich für Perl entschlossen, weil ich die Sprache noch nicht gekannt habe. Nach ein paar Anfängerprogrammen habe ich dann versucht, das Gelernte direkt 'brauchbar' umzusetzen, d.h. mit Beispielen zu implementieren, die ich dann auch gleich in meinem Projekt verwenden konnte. Als Resultat erhielt ich dann ein laufendes, jedoch weder optimiertes noch sehr schönes Programm, welches all die geforderten Funktionalitäten erfüllte.

Für die Oberflächengestaltung standen vorerst zwei Programmiersprachen zur Auswahl: Tcl/Tk und Java. Für Java stand das Java-Studio zur Verfügung, welches in Oberfläche und Funktionalität in etwa dem Borland C++-Builder entspricht.
Der Vorteil von Java liegt ganz klar in der Plattformunabhängigkeit, während Tcl/Tk

etwas schneller ist. Nach Diskussionen über Sinn und Zweck (WO wird das Programm WIE verwendet), entschloss ich mich für Tcl/Tk, da das Programm eigentlich immer von einer SunWorkstation aus aufgerufen wird. Auch hat mich das Lernen einer neuen Sprache gereizt; Java kannte ich bereits.

4.1.1 Konklusion

Perl und Shell-Script haben sich bewährt. Wenn auch Perl nicht ganz so schnell ist wie ein compiliertes C/C++ Programm, so ist der Geschwindigkeitsverlust kaum bemerkbar. Der Haupt-Geschwindigkeitsverlust geschieht beim Lesen von Daten direkt ab anderen Workstations (eg. bei updates). Dieser Zeitverlust wäre auch mit C/C++ nicht umgehbar.
Perl hat sich jedoch ideal zum schnellen Ein- und Auslesen, zum Filtern, als auch zum Einfügen von Daten geeignet, da es dazu viele Funktionen und Methoden zur Verfügung stellt.

Tcl/Tk hat sich als weniger geeignet herausgestellt. Es ist relativ langsam und schwierig zum Gestalten von Details. Anwendungen, die nicht zum vorgestellten Standard gehören, mussten mühsam in Tcl implementiert werden. Auch das Variablenhandling gestaltete sich zu Beginn als sehr kompliziert.

Nach einigen Anlaufschwierigkeiten wurden mir noch Python und Perl/Tk als Alternativen vorgeschlagen. Aber da war ich schon zu weit fortgeschritten und fühlte mich zu nah am Projektende, um nochmals eine Sprache zu lernen. Somit blieb ich bei Tcl/Tk und konnte mit der Zeit auch (fast) alle Probleme lösen oder umgehen.

4.2 Allgemeine Aspekte

Wie bereits beschrieben, haben sich die Programmteile nach und nach beim Erlernen der entsprechenden Programmiersprachen entwickelt. Somit war der erste Prototyp auch ein entsprechend „Zusammengewürfeltes“ Programm. Nach und nach wurden Teile umgeschrieben und optimiert.
Mit der Zeit habe ich die Sprachen besser kennengelernt und für viele Implementierungen bessere Methoden und Wege gefunden. Oft jedoch hat sich meiner Meinung nach ein Umschreiben des Codes nicht gelohnt, ähnliche Problemstellungen wur-

den danach aber „besser“ implementiert.
Ganz zu Anfang wurde ein Gesamtdesign gemacht, welches jedoch immer wieder den z.T. wechselnden Anforderungen angepasst werden musste.
Unterteile des Programmes wurden immer vor der Implementation skizziert und dazu ein - je nach dem gröberes oder feineres - Design erstellt, was eine Mischung aus Top-Down und Bottom-Up Design ergeben hat.

4.3 Design / Vorgehen

Die Tatsache, dass ich beim Lernen der Sprachen Testprogramme gleich im Zusammenhang mit der Aufgabenstellung implementierte, hat einerseits den Arbeitsaufwand etwas verringert und das Lernen der Sprachen beschleunigt. Andererseits hatte ich mit zunehmendem Wissen immer bessere Ideen um bestimmte Teile zu implementieren. Kleinere solche Sachen habe ich noch umprogrammiert, zum Teil sogar neu implementiert. Grössere Änderungen jedoch habe ich aus Zeitgründen nicht durchgeführt. Jedoch sehe ich Verbesserungs- und Optimierungsmöglichkeiten in vielen Programmteilen.
Müsste ich das ganze Projekt nochmals von vorne her beginnen oder etwas Ähnliches erarbeiten, würde der Code bestimmt besser. Es war aber eine gute Erfahrung, unter Zeitdruck mit einer neuen Programmiersprache einen Prototypen zu entwerfen.

4.4 Änderungen der Anforderungen

Auch interessant war es, auf Änderungen der Anforderungen einzugehen, zu sehen, dass ursprüngliche Anforderungen zum Teil nicht ideal waren und dass andere Ansätze eventuell besser geeignet wären. In Diskussionen mit dem „Arbeitgeber“ wurden Alternativen besprochen und gegebenenfalls Änderungen beschlossen.

A Mögliche Erweiterungen

- In die graphische Oberfläche kann noch eine Option „View log“ eingebaut werden, in welcher die Fehlermeldungen, welche beim Einlesen der Daten und beim Vergleichen der Datenbanken abgespeichert werden, angesehen werden können.
- Sortieren: Beim einfachen Anklicken der Spaltenüberschrift wird die Datenbank nach dieser Spalte sortiert. Man kann noch einen „Doppelklick“ implementieren, welcher die Datenbank in umgekehrter Reihenfolge sortiert.
- sundb -u : die zz*-Maschinen werden noch nicht ignoriert, was jedesmal einen Eintrag ins Log gibt.
- Gibt eine Maschine Antwort auf ein ping, ist es aber aus irgendeinem Grund nicht möglich, darauf ein rsh zu machen, so bleibt das Programm hängen. Man könnte noch ein Timeout einfügen, so dass nach Ablauf einer bestimmten Zeit diese Maschine übersprungen wird.
- Das Helpfile kann erweitert werden. Auch kann in der graphischen Oberflächendarstellung ein Menu „help“ erstellt werden, welches Hilfe für spezifische Programmteile anbietet.
- WWW: Erstellen eines Programmes (evt. als CGI), welches die einzelnen Darstellungen direkt auf das Netz legt. Dazu können die zum Ausdruck generierten Postscript Files oder auch die Tcl/Tk Tabellen benutzt werden, welche in html-Tabellen eingelesen werden könnten.
- Programmcode optimieren und besser strukturieren

B Bibliographie

- Learning Perl, Randal L. Schwartz, O’Reilly & Associates, Inc.,Sebastopol, USA, 1993
- Programmieren mit Perl, Laddy Wall, Tom Christiansen & Randall L. Schwartz, O’Reilly-Verlag Köln, 1998
- Unix System V.4, Stefan Stapelberg, Addison-Wesley Bonn, 1993
- Unix in a Nutshell, Daniel Gilly & Staff of O’Reilly, O’Reilly USA, 1996

- Sed & Awk, Dale Dougherty & Arnold Robbins, O'Reilly USA, 1997
- Practical Programming in Tcl and Tk, Brent Welch, Draft to be published by Prentice Hall, January 13 1995
- Tcl and the Tk Toolkit, John K. Ousterhout, Addison-Wesley USA, 1994

C Programmcode

Vorläufig nur eine Aufstellung der Funktionen:

```
Sun_application

#! /opt/local/bin/wish -f
# remark: entry .passwd -show *
# result: asteriks are shown instead of the entered chars...
# ...or whatever char follows the -show
# auf alf: /usr/dt/bin/sdtperfmeter
#
# Procedures:
# ===========
#
# proc enter {sun}  : enters a new sunworkstation
  into the database
# proc do_entry     : calles the above
# proc update_database  : reads in the updated version
  of the database in tcl.db
# proc text_window { text_string }  : creates a simple text window
# proc update_list { i_want_to_update } : updates the shown list in the
 canvas, considering the alterations
# proc entry_list { title labels titles}  : creates the win-
dow 'modify
  configuration'
# proc SetOfLabelEntries { canvas labels titles } : con-
tinues the above
# proc check_input { } { : checks, if the alteration is
  acceptable, asks in doubt
# proc option_window {text_string title } : asks whether to change or no
```

```
# proc continue_change { title } :
# proc reset_change { title } :
# proc save_mod_list { titles } { : saves the changes to the workstation
# proc delete_line { } :
# proc delete_workstation { titles } { : deletes selected work-
station,
  leaving window
# proc delete_window { text_string command machine}
# proc really_delete_workstation { machine }
# proc order { word } { : calls sort in perl to order the list
  after the topic given
# proc reset_modifications  : resets the modified fields to the
  original setting
# proc reset_system {  } :
# proc modify_configuration { } {  : window to change the configuration
# proc set_group { number_array } :
# proc get_configuration { } { : gets default or saved configuration
 (if it exists)
# proc save_configuration { } { : saves the changed con-
figuration to
  future startups
# proc reset_configuration { } { : resets the changed (not saved)
  configuration to default/saved
# proc set_default_configuration { }  : sets configurati-
on to "showing
  everything"
# proc print { howmuch }
# proc print_ws { }
# proc really_print { }
# proc print_group { what active}
# proc display { what active}


////////////////////////////////////////////////////////////////////


SUNDB:

# sub update
```

```
# sub compare_databases
# sub warning
# sub enter_new_data
# sub modify_existing_data
# sub menu
# sub shortcut_to_del_ws
# sub del_ws
# sub comp_ws
# sub complete
# sub set_exit
# sub mod_ws
# sub modify
# sub help
# sub alphabetical_order
# sub get_bootserver_info
# sub gethex
# sub insert
# sub make_tcltk_database
# sub enter_modified_line
# sub sort
# sub cdv_to_db
#

////////////////////////////////////////////////////////////////////////

generate ps:

sub list
sub workstation
sub group

////////////////////////////////////////////////////////////////////////

DISPLAY:

Nimmt Configuration und macht eine reduzierte tcl-db, wel-
che dann entweder zum
Ausdrucken oder zum Darstellen verwendet wird - je nach Funktionsaufruf
```

```
sub make_tclDB
sub make_print
////////////////////////////////////////////////////////////////////
sun_entries

kriegt Namen der Workstation und holt so viel Info wie moeg-
lich (ausser Server)
////////////////////////////////////////////////////////////////////
rping
Schaut, ob Workstation innerhalb einer Sekunde antwort gibt

////////////////////////////////////////////////////////////////////

Weitere Dateien:

rpc.sdb ;enthaelt Configuration
sun.db ; Hauptdatenbank, wird nur via Perl veranedert
sunworkstations.csv ; Originaldatenbank
tcl.db ;wird nur von Tcl/Tk verwendet, dient rein zu Darstellungszwecken
database_titles : Enthaelt die Titel, wird beim update zur Si-
cherheit verwendet,
; koennte auch weggelassen werden
```