INSTITUT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Entwurf und Implementierung von Managementfunktionen für HP Dolphin zum Management von Windows PC-Netzen

Bearbeiter	: Michael Dzik
Aufgabensteller	: Prof. Dr. Heinz-Gerd Hegering
Betreuer	: Stephen Heilbronner

Inhaltsverzeichnis

1 Einleitung

Managementplattformen gewinnen gegenwärtig im Bereich des Netz- und Systemmanagements zunehmend an Bedeutung. Die Eigenschaft, verschiedene Managementanwendungen in eine Plattform integrieren zu können, ist auch das Thema dieses Fortgeschrittenenpraktikums. Ziel ist es, für den Plattformprototyp Dolphin Funktionen zu entwickeln, um einfache Managementfunktionen für Windows Peer-to-Peer-Netze zu ermöglichen.

Hierfür wird ein „Model“ mit Managementobjekten und entsprechenden Operationen für die Plattform Dolphin modelliert und implementiert.

Parallel dazu soll dieser Entwicklungsvorgang dokumentiert werden, um einen Überblick über die proprietäre Modellierungssprache von Dolphin zu bekommen.

Die Kommunikation zwischen der UNIX-basierten Plattform Dolphin und einem PC stützt sich auf Shellskripten ab, die ebenfalls für diesen Zweck implementiert werden.

Teile der Managementfunktionalität werden Dolphin durch die File- und Printersharing-Software SAMBA zur Verfügung gestellt und mit Hilfe der oben erwähnten Shellskripten eingesetzt.

Das Ergebnis ist das Model „SmbBase“, das der Ausarbeitung im Anhang beigefügt ist.

Die Ausarbeitung gliedert sich in folgende Abschnitte:

Im zweiten Kapitel wird das Informations- und Kommunikationsmodell der Plattform Dolphin erläutert. Im dritten Kapitel wird die Software SAMBA in seiner Funktion als Lieferant managementrelevanter Informationen für die Plattform Dolphin näher untersucht. Der Einführung im Umgang mit der grafischen Benutzerschnittstelle von Dolphin wird ein weiteres Kapitel gewidmet. Das fünfte Kapitel beschreibt die Objekthierarchie des entwickelten Models, sowohl nach dem Top-down-Ansatz, als auch nach dem Bottom-up-Ansatz. Im sechsten Kapitel wird auf die Verwendung der Dolphin-eigenen Modellierungssprache anhand des Models SmbBase eingegangen.

2 Einführung in HP-Dolphin

2.1 Das Informationsmodell von Dolphin

Dolphin ist der Prototyp einer Managementplattform und wird von der Firma Hewlett Packard entwickelt. Es besitzt ein objektorientiertes Informationsmodell. Durch eine eigene Modellierungssprache wird ein Abbild der „realen Welt" für Dolphin erstellt.

Alle Netz- und Systemressourcen werden plattformintern auf Instanzen von Objektklassen abgebildet. Objekteigenschaften werden durch Attribute beschrieben. Mit einer konkreten Wertbelegung dieser Attribute ist Dolphin in der Lage, ein Spiegelbild der Realität darzustellen.

Logisch zusammengehörige Objektklassen werden in sog. **Models** zusammengefasst (siehe Abbildung 2-1).

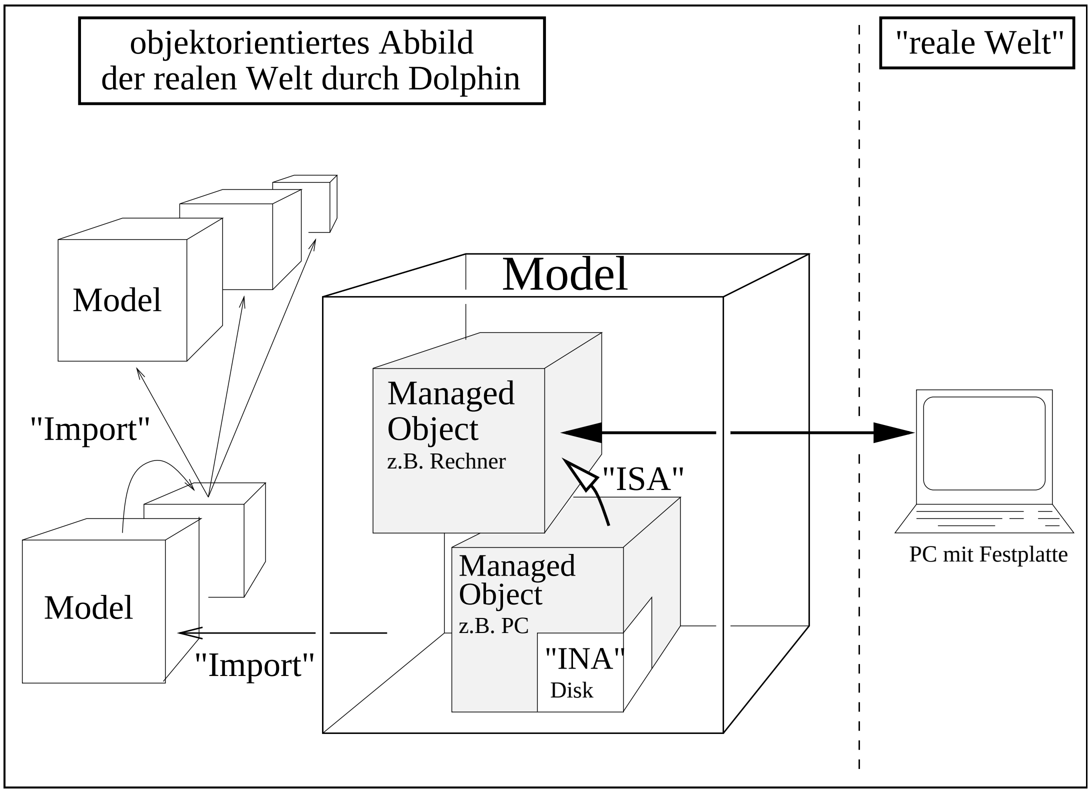

Abbildung 2-1: Das objektorientierte Informationsmodell von Dolphin

Dolphin bietet zudem die Möglichkeit an, die gesamte Objekthierarchie grafisch darzustellen. Für nähere Informationen siehe [Sch95] und Kapitel 4.2.

2.1.1 Models

Models stellen konzeptuelle Behälter für logisch zusammengehörige Objektklassen dar. Sie dienen dem alleinigen Zweck die Objekthierarchie übersichtlich zu gliedern.

Damit Objektklassen über die Grenzen ihres Models hinweg auf bereits bestehende Objektklassen anderer Models zugreifen können - um sich gegebenenfalls durch Vererbung deren Eigenschaften anzueignen - benützt Dolphin das IMPORT-Konstrukt. Dadurch wird den Objekten eines Models Zugang zu allen Objekten eines anderen Models gewährt (siehe Abbildung 2-1 und 2-2). Es können beliebig viele Models importiert werden. Es folgt ein Beispiel für den Definitionskopf eines Models in der Dolphin-eigenen Modellierungssprache:

```
MODEL SmbBase
IMPORT  ComputerDynamics ManageableObject ManageableDevice
 Network ComputerUsersGroups ComputerFileSystem ComputerBase
```

Abbildung 2-2: Der Deklarationskopf des Models SmbBase

Am Beginn des Models steht sein Name, gefolgt von einer fortlaufenden Zeile, in der alle zu importierenden Models aufgeführt sind. Das Beispiel in Abbildung 2-2 stellt den Kopf des Models **SmbBase**[1] dar.

2.1.2 Objekte

Die Objektklassen repräsentieren die grundlegende Struktur in Dolphin. Sie können, wie bereits erwähnt worden ist, nicht nur abgeleitete Unterklassen von Objekten desselben Models sein, sondern auch von Objekten importierter Models abstammen. Es bleibt zu beachten, daß die Modelllierungssprache von Dolphin nur strikte Vererbung unterstützt. In der Dolphin-eigenen Modellierungssprache wird das Vererbungskonstrukt mit ISA bezeichnet. Ausführliche Beispiele zu der Modellierungssprache werden in Zusammenhang mit dem Model SmbBase in Kapitel 6 behandelt.

Neben der **Vererbungshierarchie** existiert in Dolphin noch die **Enthaltenseinhierarchie**. Sie wird mit dem Konstrukt INA modelliert (siehe Abbildung 1). Dolphin hat in der Vergangenheit nur eine Abstufung in der Enhaltenseinhierarchie gestattet. Mit der neuen Version 14.18 ist diese Einschränkung aufgehoben worden. Die mehrfache Abstufung in der Enthaltenseinhierarchie wird ebenfalls im Model SmbBase eingesetzt (siehe hierzu Kapitel 5.2).

[1] Das Model SmbBase ist im Rahmen dieses Fortgeschrittenenpraktikums entworfen worden; ein Ausdruck von SmbBase befindet sich im Anhang.

Dolphin unterscheidet allgemein zwischen zwei Kategorien von Objektklassen (siehe hierzu [Pel95]). Zum einen gibt es

- sog. „Core Object"-Klassen, wie z.B. Integer, Strings oder Boolean. Sie leiten sich von zugrundeliegenden Smalltalkklassen ab.

Zum anderen gibt es

- **Modelobjektklassen**. Sie repräsentieren Ressourcen aus der realen Welt, wie z.B. Rechner oder Drucker. Abstrakte Objekte, z.B. Listen oder User werden ebenfalls durch diese Objektklassen dargestellt.

Kommen Modelobjektklassen als Blätter im „Vererbungsbaum" vor, so kann sie der Benutzer von Dolphin interaktiv, über eine vorgegebe Benutzerschnittstelle, instanziieren (siehe [Sch95], Kapitel 4.2 und Kapitel 4.3). Anschließend können auf den neu erstellten Objekten vom Modellierer definierte Managementoperationen ausgeführt werden.

2.1.3 Objektattribute

Die Eigenschaften von realen Objekten werden in Dolphin durch Attribute dargestellt. Sie setzen sich aus einer beliebigen Anzahl von Argumenten zusammen, wobei ein Argument das zulässige Minimum ist. Das erste Argument ist stets eine Objektklasse, auf die sich das Attribut bezieht. Die Kardinalität[2] der einzelnen Argumente zueinander wird durch einen Klammerausdruck angegeben. Dabei steht „M" für **Many** und „O" für **One** (siehe hierzu [Pel95] und [Mit95]).

```
[abstractSmbNode] name [string] (M:O)
DESCRIPTION 'name of an abstract SMB node'
CATEGORY 'SMB properties'
```

Abbildung 2-3: Das Attribut name des Objekts AbstractSmbNode

Abbildung 2-3 zeigt ein Beispiel eines Attributs aus dem Model SmbBase in der Dolphin-Modellierungssprache.

Die Attribute werden bei der Instanziierung ihrer zugehörigen Objektklasse mit konkreten Werten belegt. Der genaue Ablauf dieses Vorgangs wird in Kapitel 6 erläutert.

[2] mengenmäßige Quantifizierung der Assoziation

Insgesamt unterscheidet Dolphin fünf verschiedene Arten von Attributen:

- Methodenattribute (method attributes),
- **einfache Attribute** (basic attributes),
- abgeleitete Attribute (derived attributes),
- Kommandoattribute (command attribute),
- und virtuelle Attribute (virtual attribute).

In diesem Fortgeschrittenenpraktikum werden nur einfache Attribute behandelt.

2.2 Das Kommunikationsmodell von Dolphin

Um mit den Netzressourcen managementrelevante Informationen austauschen zu können, besitzt Dolphin unterschiedliche Möglichkeiten:

- über SNMPv1 (siehe [Gro90a], [Gro90b] und [Gro91]),
- über den Schnittstellenstandard DMI der DMTF,
- über RPC,
- oder durch die Verwendung von Shellskripten

(siehe hierzu auch [Sch95]).

In diesem Fortgeschrittenenpraktikum wird ausschließlich von Shellskriptaufrufen Gebrauch gemacht. Die Shellskripten stützten sich dabei wiederum auf Befehle einer speziellen Peer-to-Peer-Netzwerksoftware ab. Diese Software wird in Kapitel 3 eingehend auf seine Funktionalität als „Lieferant" managementrelevanter Informationen für Dolphin untersucht.

2.2.1 Queries

Queries und **Actions** spezifizieren die Verbindung zwischen den einfachen Objektattributen in Dolphin und ihren tatsächlichen Wertbelegungen in der realen Welt. Die Wertbelegungen von Attributen werden in Dolphin auch als **Fakten** bezeichnet.

Eine Query hat die Aufgabe, Informationen über die Wertbelegung der Attribute zu erfragen, zu sammeln und zu verarbeiten[3] (siehe hierzu Kapitel 2.1.3). Besitzt Dolphin keine Informationen über die Attribute seiner Objekte, werden die entsprechenden Queries aufgerufen und die

[3]Eine Query kann auch mehrere Attribute und Attribute mehrerer Objekte aus der realen Welt gleichzeitig aquirieren.

ermittelten Fakten in der Dolphin-eigene Datenbank, der sog. **Fact Base** abgelegt. Der Querymechanismus greift bei der Ermittlung benötigter Informationen auf die vier oben erwähnten Kommunikationsarten zurück. Der genaue Abfrageprozeß wird in Kapitel 6.5 dargestellt.

Actions hingegen können Fakten verändern, um auf diese Weise konkreten Einfluß auf die reale Welt zu nehmen. Man spricht dabei auch von „Fixing the world“ (siehe hierzu [Fed95] und Kapitel 6.7).

Die folgende Abbildung zeigt zur Verdeutlichung eine abstrakte Darstellung der verschiedenen Kommunikationsarten, der sich Dolphin bedient:

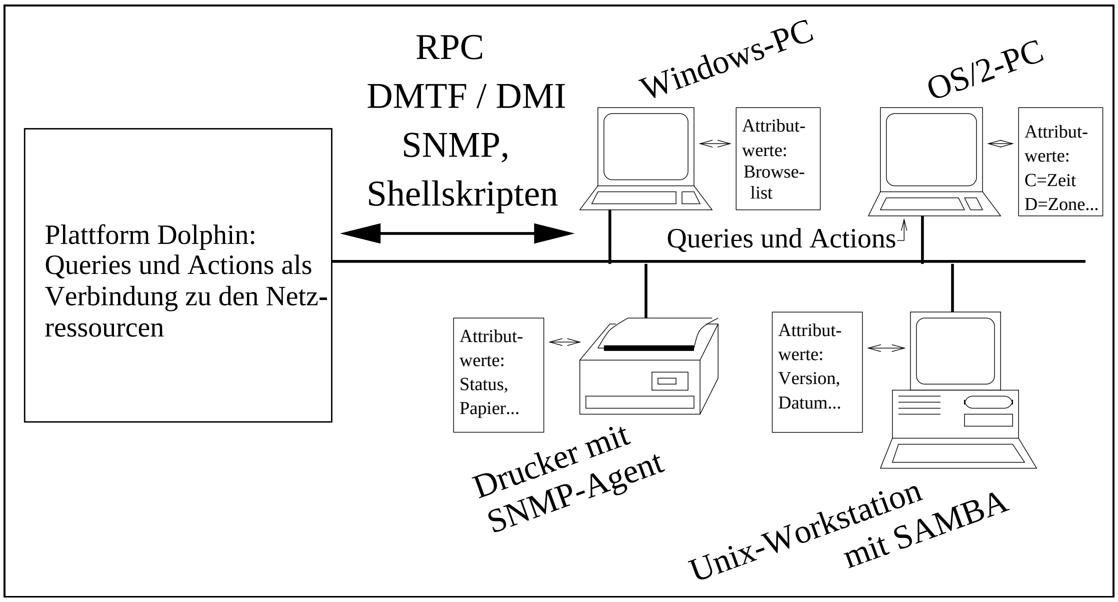

Abbildung 2-4: Das Kommunikationsmodell von Dolphin

3 Das SMB-Protokoll und die SAMBA-Software

3.1 Das SMB-Protokoll

PCs, die zu Peer-to-Peer Netzen zusammengeschlossen sind, verwenden in der Regel zur Kommunikation in der Anwendungsschicht das **Server Message Block**-Protokoll (SMB-Protokoll).

Das SMB-Protokoll ist aus einer Kooperation der Firmen Microsoft, IBM und Intel entstanden. Es wird gemäß dem ISO/OSI-Referenzmodel der siebten Schicht, der Anwendungsschicht, zugerechnet. Das SMB-Protokoll erfreut sich heute einer breiten Unterstützung durch die verschiedenen Softwarehersteller von Peer-to-Peer-Netzen. In der Regel ist es in allen bedeutenden Peer-to-Peer Netzsoftwareprodukten enthalten, wie z.B. Windows for Workgroups, OS/2, LAN Manager, LAN Server oder auch **SAMBA**, das im nächsten Kapitel 3.2 näher untersucht wird.

Das SMB-Protokoll bietet sog. „Services“ zum Zugriff auf bestimmte Ressourcen eines Rechners in einem Peer-to-Peer Netz an. Im weiteren Verlauf dieser Ausarbeitung soll der deutsche Begriff „Dienst“ für den englischen Begriff „Service“ verwendet werden.

Ein Peer-Rechner gewährt entweder einem anderen Peer-Rechner Zugriff auf seine Ressourcen, oder er selbst greift auf Ressourcen anderer Peers zu. Die dabei verwendeten Dienste werden im Zusammenhang mit dem SMB-Protokoll allgemein auch als **Shares** bezeichnet. Bei den angesprochenen Diensten handelt es sich beispielsweise um Datei- und Druckerdienste (siehe hierzu auch Kapitel 3.3, in dem anhand konkreter Beispiele eine Aufstellung von Shares eines SMB-Rechners angegeben wird; außerdem wird der Zugriff und die Benutzung dieser Shares durch andere Peer-Rechner dargestellt).

3.2 SAMBA

Die SAMBA-Software implementiert das SMB-Protokoll für Unix-Rechner. SAMBA ermöglicht es Unix-Rechnern über eine zeilenorientierte Benutzerschnittstelle auf Datei- und Drucker-Shares anderer SMB-Rechner (z.B. Windows-PCs) zuzugreifen und diese mit entsprechenden Befehlen zu bearbeiten (siehe [Tri95c] und Kapitel 3.3).

Diese Funktionalität kann Dolphin zur Verfügung gestellt werden und es so ermöglichen, über die Plattform managemetrelevante Informationen zu ermitteln und zu verarbeiten. In Kapitel 6.4.1 wird dieser Prozeß näher untersucht.

SAMBA selbst ist eine Anwendungssoftware und setzt in der Protokollhierarchie (nach dem OSI Referenzmodell) auf TCP/IP auf. Zwischen

Transportprotokoll und der Anwendungssoftware SAMBA befindet sich das NetBIOS.

NetBIOS definiert eine Schnittstelle zwischen einem PC-Betriebssytem und einem beliebigen Transportsystem. Damit Unix-Rechner mittels SAMBA mit anderen PC-Peers auf der Anwendungsschicht miteinander Informationen austauschen können, muß von SAMBA auch das NetBIOS unterstützt werden. Die Abbildung 3-1 veranschaulicht die Protokollhierarchie:

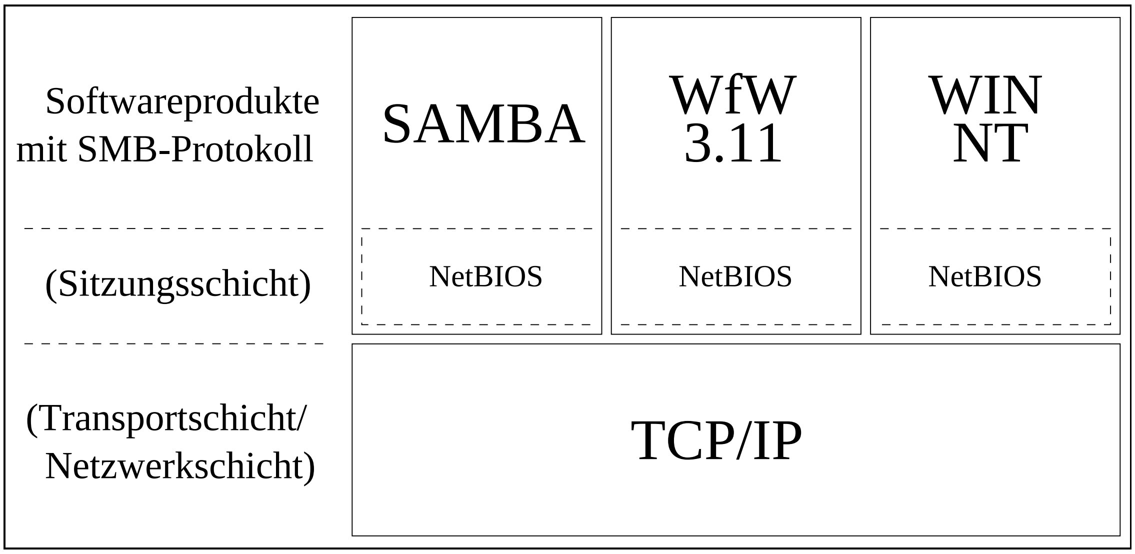

Abbildung 3-1: Auszug aus der Protokollhierarchie von Peer-to-Peer Netzsoftwareprodukten

Bevor in Kapitel 3.3 die SAMBA-Funktionalität näher betrachtet wird, folgt zunächst ein kleiner Einschub über den Kommunikationsablauf zwischen einem SAMBA-Server und einem SAMBA-Client.

3.2.1 Kommunikation zwischen SAMBA-Server und -Client

SAMBA realisiert den Zugriff auf die Shares eines SAMBA-Servers mittels des Client-Server Konzepts. Dazu wird zwischen dem SAMBA-Server und einem Client immer dann eine sog. **Session** eröffnet, wenn ein Clientprozeß mit einem Serverprozeß Verbindung aufnimmt. Jeder Client bekommt für eine Session eine eigene Prozeßinstanz des Servermoduls von SAMBA (smbd) zugewiesen. Dieser Serverprozeß bearbeitet anschließend alle Aufträge, die vom Client während einer Session angefordert werden. Wenn alle Aufträge eines Clients abgeschlossen worden sind, endet auch die jeweilige Instanz des Serverprozesses (siehe hierzu [Tri95d]).

3.3 Smbclient

Smbclient ist Teil des SAMBA-Suites und ist ein Client-Programm. Es definiert eine ftp-ähnliche Benutzerschnittstelle, mit der man sich einerseits eine Aufzählung aller, von einem SMB-Rechner zur Verfügung gestellter Shares, anzeigen lassen kann. Andererseits können verschiedene Operationen auf diese Shares angewendet werden. Beispiele dieser Operationen sind in den folgenden Unterkapiteln zu sehen.

3.3.1 Benutzung von smbclient

Das Clientprogramm smbclient bietet eine Eingabeschnittstelle für den Benutzer an. Der Aufruf von smbclient wird, hier vereinfacht dargestellt, wie folgt durchgeführt:

$> smbclient servicename (-Optionen)

- **smbclient** eröffnet eine Session zu einem SAMBA-Server.
- **servicename** ist der Name der angeforderten Share eines bestimmten Peer-Rechners.

Der Server antwortet daraufhin mit einem Request nach einem Paßwort. Wird es richtig eingegeben, erscheint das SAMBA-Server-Prompt (vgl. ftp-Prompt). Dem Benutzer von smbclient stehen nun ca. 30 Operationen in Form von einzelnen Befehlen zur Verfügung, mit denen die ausgewählte Share bearbeitet werden kann. Es werden hier als Beispiele nur die Befehle behandelt, die zu einem späteren Zeitpunkt in Kapitel 6 in Zusammenhang mit Dolphin ihre Verwendung finden.

Bei der Abbildung 3-2 handelt es sich um einen Auszug aus der Liste der Dateien und Unterverzeichnisse der Share dzik auf dem Rechner hpsp1, die mit dem Befehl ls ausgegeben wird.

$> smbclient hpsp1 dzik
$> Password: secret
smb> ls

```
.cshrc              HR     2761  Fri May 19 09:55:07 1995
.tcshrc              H     2791  Wed Feb 28 12:57:27 1996
.login              HR     2526  Sat May 27 15:48:27 1995
.logout             HR       12  Tue Jul 30 12:24:50 1991
.mailr              HR       74  Fri Nov  6 14:10:18 1992
.Xauthority          H      680  Thu Feb 29 07:40:28 1996
pub                  D        0  Tue Dec  5 13:43:28 1995
Scripten             D        0  Tue Feb 27 11:16:11 1996
Steuer.tex                 5328  Mon Feb 26 11:26:48 1996
Mail                 D        0  Wed Jun 14 13:15:12 1995
News                 D        0  Thu Aug  3 15:53:59 1995
Autosaves            D        0  Thu Feb 29 07:46:58 1996
```

Abbildung 3-2: Die Ausgabe von ls der Share hpsp1 dzik

Die hier dargestellte Share ist ein Dateidienst, der Zugriff auf das Homeverzeichnis des Benutzers dzik gewährt. Die ausgegebene Liste enthält neben dem Namen der Datei außerdem noch weitere Angaben zu den einzelnen Dateien und Unterverzeichnissen. Zum einen wird mit Großbuchstaben die Art der Datei bzw. die Rechte, die auf dieser Datei gesetzt sind angezeigt. Ein D bezeichnet z.B. ein Verzeichnis. Kein Buchstabe bedeutet, daß es sich um eine Datei handelt, die nur Lese- und Schreibrechte besitzt. Es werden zudem Aussagen über die Größe der Dateien und Verzeichnisse in Byte und das Datum des letzten schreibenden Zugriffs, bzw. der Erstellung gemacht.

Mit den Befehlen mkdir und rmdir kann man neue Unterverzeichnisse erstellen oder wieder löschen. Mit rm werden Dateien gelöscht. Mit quit beendet man die SAMBA-Session.

Handelt es sich bei der angeforderten Share um eine Druckerwarteschlange[4] für einen bestimmten Drucker, so wird man vom SAMBA-Server mit der entsprechenden Druckerwarteschlange verbunden. Mit dem Befehl queue werden alle noch nicht gedruckten Druckaufträge ausgegeben. Mit cancel kann man einzelne Druckaufträge wieder aus der Druckerwarteschlage löschen.

Für weitere Informationen sowie eine komplette Aufzählung aller Befehle siehe [Tri95b].

Informationen über die einzelnen Shares eines SMB-Rechners können, wie eben gezeigt, über die Benutzerschnittstelle von smbclient abgefragt bzw. verändert werden. Die dabei verwendeten Befehle, sowie die ermittelten Daten, werden in Kapitel 6 eine wichtige Rolle spielen. Dort wird Dolphin mit Hilfe von Shellskripten die vorhandenen SAMBA-Befehle einsetzen, um

[4] Aufruf von smbclient mit der Option -P

- managementrelevante Informationen über die einzelnen Peer-Rechner abzufragen (z.B. wieviel Plattenkapazität besitzt eine Share),
- sinnvolle Modifikationen der Peer-Rechner zu ermöglichen (z.B. löschen einer core-Datei).

3.3.2 smbclient -L ohne Ausgabe einer Browseliste

Ruft der Befehl smbclient einen SMB-Server Rechner (z.B. hpsp1) mit der Option -L auf, so bekommt man als Antwort eine Liste der gemeinsam benutzbaren Shares des Rechners zurück.

Als Beispiel für diese Ausgabe soll Abbildung 3-3 dienen. In dieser Abbildung sind zusätzlich alle Zeilen[5] markiert, deren Inhalt in Kapitel 6 als managementrelevante Informationen von Dolphin verwendet.

Die Form der Liste ist aufgeteilt in:

- Die aktuelle Zeit des Servers.
- Die Zeitzone, in der sich der Server befindet.
- Einer Zeile, in der der aufgerufene Servername (hpsp1), der aufrufende Benutzername (dzik), die Workgroup (LANGROUP) und der Domainname (LANGROUP) stehen.
- einer tabellarisch eingeteilten Aufzählung der Dienste, auf die man mit dem SMB-Protokoll zugreifen kann. Ihr Aufbau hat folgendes Aussehen:
  - Dienste (Sharenames),
  - der Typbezeichnungen der Dienste,
  - und der dazugehörigen Kommentare.

[5] mit einem „<–“

$> smbclient -L hpsp1

```
Server time is  Mon Aug 14 14:34:34 1995        <-
Timezone is UTC+2.0                             <-

Server=[hpsp1] User=[dzik] Workgroup=[LANGROUP] <-
Domain=[LANGROUP]                               <-

Sharename      Type      Comment
---------      ----      -------
homes          Disk      Home Directories  <-
printers       Printer   All Printers
cdrom          Disk      CD-ROM - Laufwerk an /mnt/cdrom
pc_wr          Disk      PC-Software (Read/Write)
isar           Disk      ISAR-Archiv
openview       Disk      HP-OUA Dateien und WWW-Seiten
gp             Printer   Drucker gp
IPC$           IPC       IPC Service (Samba 1.9.13)
dzik           Disk      Home Directories
        ...
This machine does not have a browse list        <-
```

Abbildung 3-3: Die Ausgabe von smbclient -L

- und als letztes erscheint der Hinweis, ob die angesprochene Maschine (hpsp1) eine Browse-Liste besitzt oder nicht (hpsp1 besitzt in diesem Beispiel keine Browse-Liste).

3.3.3 smbclient -L mit Ausgabe einer Browseliste

Pchegering2 ist ein „Browsemaster" und besitzt eine Liste aller SMB-Rechner im Netz. Diese **Browseliste** wird dynamisch von SAMBA verwaltet und dem aktuellen Stand automatisch angeglichen. Mit dem Aufruf

$> smbclient -L pchegering2

wird folgende Browseliste ausgegeben:

```
This machine has a browse list:           <-

Server               Comment
---------            -------
HPSP1                Samba 1.9.1.13    <-
PCHEGERING2          S. McBrumm
PCHEGERING4          MNM-Team
```

Abbildung 3-4: Die Browseliste von Pchegering2

4 Die Benutzeroberfläche von Dolphin

Die grafische Benutzerschnittslle von Dolphin setzt sich aus mehreren funktionalen Bestandteilen zusammen, die in unterschiedlichen Fenster untergebracht sind.

Es folgt eine Übersicht der wichtigsten Benutzerwerkzeuge von Dolphin (siehe hierzu auch [Fed95]).

4.1 Der System Manager

Der System Manager hat die Aufgabe, diejenigen Geräte grafisch abzubilden, die in der Konfigurationsdatei „domainConfig" angegeben sind. In der Abbildung 6-1 ist ein Bildshirmabzug des System Managers zu sehen, in Abbildung 6-2, die dazugehörige Konfigurationsdatei.

Das Fenster des System Managers ist zweigeteilt. Im oberen Teil werden die Hauptdomäne und die dazugehörigen Subdomänen angezeigt. Die Informationen über die Domänenstruktur liest der System Manager direkt aus der Konfigurationsdatei. IFI_LMU ist Hauptdomäne (siehe Abbildung 10 und 11). In den nachfolgenden Zeilen werden sowohl NM[6], als auch CIP als Subdomänen von IFI_LMU angegeben. NM enthält einige Geräte in seiner Domäne: die Unix-Workstations „hpsp1" und „hpheger4", den Router „brokl3", den Laserjetdrucker „gutenberg" und einen PC „pchegering4". Im System Manager werden die Geräte, die von einer (Sub-)Domäne verwaltet werden, im unteren Teilbereich des Fensters abgebildet.

Vom System Manager aus, können mit Hilfe der Maussteuerung alle anderen Fenster über das Menü **Model Writers** geöffnet werden:

- den Model Editor
- den Knowledge Base Browser
- den Workspace
- den Icon Browser usw.

Ab der Dolphin-Version 14.18 wird der System Manager unter der Bezeichnung „Desktop" im Hauptfenster „VisualWorks" im Drop-down-Menü **Dolphin** aufgerufen. Die Werkzeuge der folgenden Kapitel werden ebenfalls über das Drop-down-Menü des VisualWorks-Fensters aufgerufen.

[6]steht für Netz-Management

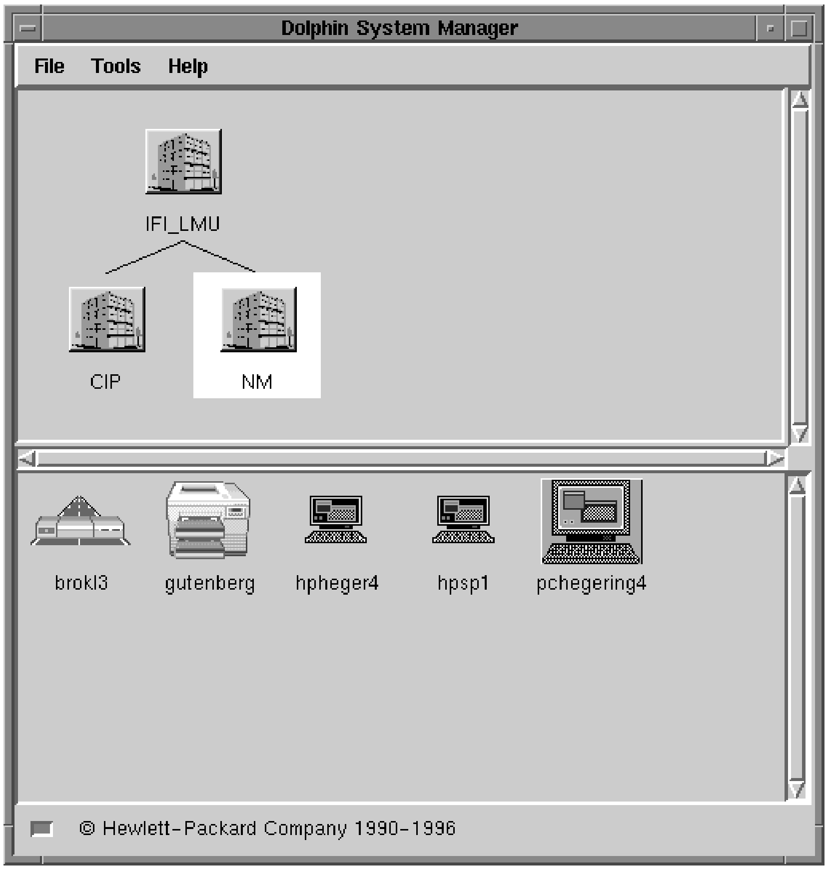

Abbildung 4-1: Der Dolphin System Manager

```
Domain:IFI_LMU::
Domain:NM:IFI_LMU:(MANAGES HpuxWS hpsp1)
                   (MANAGES HpuxWS hpheger4)
                    (MANAGES LaserJet4 gutenberg)
                     (MANAGES HpRouter brokl3)
                      (MANAGES DesktopPC pchegering4)
Domain:CIP:IFI_LMU:
```

Abbildung 4-2: Beispiel einer Konfigurationsdatei des System Managers

4.2 Der Model Editor

Der Model Editor ist die Bearbeitungsschnittstelle für die Models. Mit seiner Hilfe kann ein Administrator eigene Models erstellen, indem er Objekte, Queries oder Actions definiert (siehe auch Kapitel 2.1).

Nachdem das jeweilige Model in Dolphin eingefügt worden ist, können seine Objektklassen instanziiert werden. Dazu bewegt man die Maus auf das Feld Models und drückt die mittlere Maustaste. Es öffnet sich ein Pull-down-Menü. Durch das Anklicken der Option **file in** öffnet sich das Fenster „Dolphin File List on models". Hier kann man beliebige Models, die im Unterverzeichnis /models als Datei mit dem Suffix „*.m" abgespeichert sind, nach Dolphin importieren. Man erreicht das durch einfaches Anklicken einer Model-Datei mit der linken Maustaste. Im unteren Bereich des Fensters erscheint der Inhalt der markierten Model-Datei. Man bewegt die Maus auf diesen Bereich, drückt die mittlere Maustaste und aktiviert „file it in" (siehe Anhang). Dolphin importiert auf diese Weise das markierte Model und die darin enthaltenen Objektklassen werden gemäß ihrer Modellierung in die Vererbungs- und Enthaltenseinhierarchie von Dolphin aufgenommen. Erst danach sind die neuen Strukturen (Objekte, Queries, Actions usw.) Dolphin bekannt und können ab diesem Zeitpunkt instanziiert werden. Umgekehrt kann man mit der Option **remove** ausgewählte Models aus Dolphin wieder entfernen.

Klickt man mit der linken Maustaste ein beliebiges Model an, so erhält man im unteren Editorfeld die Importliste der Models. Mit Hilfe der übrigen Pull-down-Menüs werden auch konkrete Objekte, Queries, Events, Actions und Attribute des angewählten Models angezeigt.

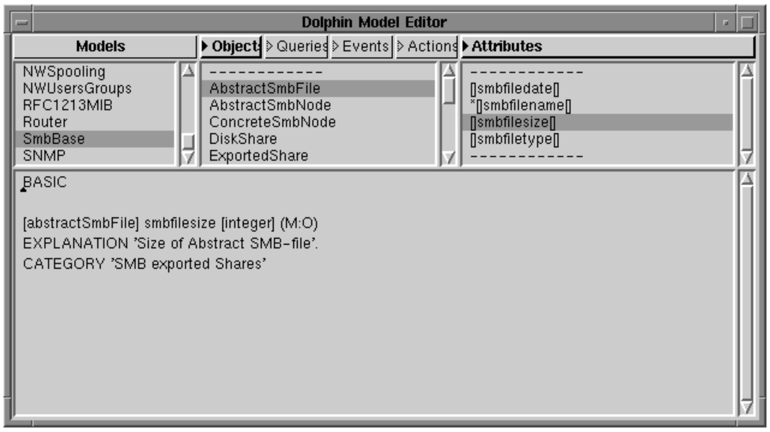

Abbildung 4-3: Der Dolphin Model Editor

Dolphin bietet im Model Editor die Möglichkeit an, die Vererbungshierarchie in einer Grafik baumartig anzuzeigen. Die Grafik erscheint, wenn man eine Objektklasse eines beliebigen Models mit der Maus einmal anklickt und anschließend die mittlere Maustaste gedrückt hält. Es öffnet sich ein Drop-down-Menü, in dem man sich wahlweise für die Objekthierarchie der markierten Objektklasse oder für die gesamte Objekthierarchie entschieden kann (vergleiche [Sch95] und Kapitel 2.1).

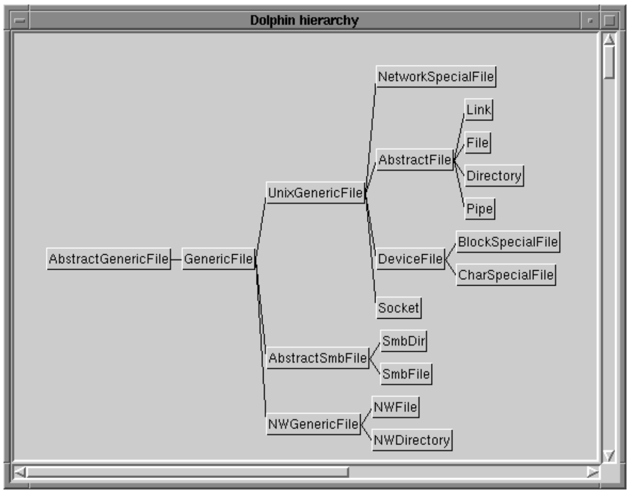

Abbildung 4-4: Ein Ausschnitt aus der Vererbungshierarchie in Dolphin

4.3 Der Workspace

Der Workspace stellt einen virtuellen Arbeitsplatz dar, in dem der Administrator durch objektbezogene Operationen Instanzen von Objektklassen erstellen kann. Die Onjektinstanzen sollen reale Objekte repräsentieren, wie z.B. Rechner, Drucker oder Router (siehe Kapitel 2.1 und Kapitel 2.2). Auf dem Workspace erscheint für jedes neu instanziierte Objekt ein Symbol in Form eines Icons (siehe Kapitel 4.5). Man erstellt ein neues Objekt durch das Menü **Contents/Create**. Hier können alle Objekte instanziiert werden, die in der Vererbungshierarchie als Blätter vorkommen, also nicht in der Enthaltenseinhierarchie einer anderen Objektklasse als innnerer Knoten vorkommen (siehe hierzu Abbildung 4-4).

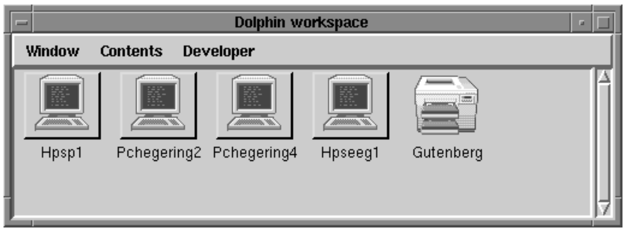

Abbildung 4-5: Der Dolphin Workspace

Einen Überblick der Eigenschaften von realen Ressourcen, die ab dem Zeitpunkt ihrer Erstellung im Workspace auch in der Fact Base von Dolphin gespeichert sind, kann man durch einen doppelten Mausklick auf das entsprechende Icon erhalten. Beinhaltet die Fact Base (siehe Kapitel 2.2.1) noch keine Informationen über die erstellte Objektinstanz, so nimmt Dolphin Kontakt zu dem realen Objekt auf und beginnt darüber Informationen zu sammeln. Die verschiedenen Arten der Kommunikation zwischen Dolphin und den einzelnen Netzressourcen wurde bereits in Kapitel 2.2 erläutert. Hervorgehoben soll an dieser Stelle nur werden, daß Dolphin seine Informationen über den Query-Mechanismus aus den Netzressourcen ableitet. Näheres zum Querymechanismus findet man in Kapitel 6.5. Mit Actions, die unter dem Menü **Contents** in der Kopfleiste des Workspace zu finden sind, kann der Benutzer direkten Einfluß auf die im Workspace dargestellten Objekte nehmen.

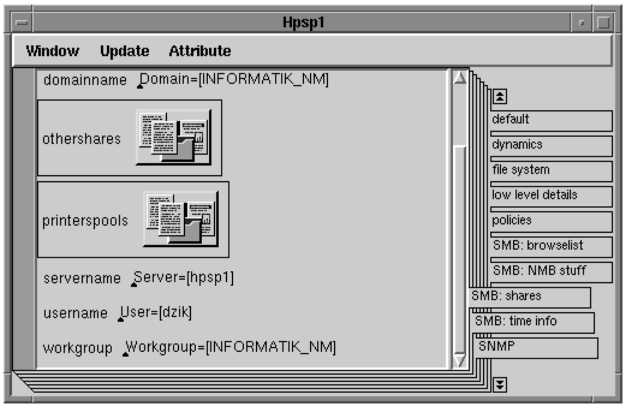

Abbildung 4-6: Der Dolphin Workspace: Hpsp1

Klickt man mit der linken Maustaste zweimal z.B. auf das Icon des erstellten Objekts Hpsp1, so erscheint ein neues Fenster mit der Aufschrift „Hpsp1“ (vgl. auch Abbildung 4-6). In diesem Fenster befinden sich, nach Registern geordnet, alle Attribute der Objektinstanz Hpsp1.

4.4 Das System Transcript

Das System Transcript zeigt u.a. den Kommunikationsverlauf zwischen Dolphin und den einzelnen Netzressourcen an. Im Falle eines Fehlers können hier ebenfalls Fehlermeldungen abgelesen werden, die Rückschlüsse auf die möglichen Fehlerursachen zulassen.

Das folgende Beispiel in Abbildung 4-7 zeigt den Kommunikationsverlauf einer „Shellskript“-Query zwischen Dolphin und der Unix-Workstation hpsp1. Man aktiviert die Query, indem man im Workspace eine Objektinstanz für den Rechner hpsp1 erstellt und diese durch einen doppelten Mausklick aktiviert.

Es erscheint ein Fenster mit der Balkenüberschrift „hpsp1“ (siehe hierzu auch Kapitel 4.3).

Wird mit der linken Maustaste auf das Register „SMB: Shares“ geklickt, so zeigt das System Transcript die Querynamen und die dazugehörigen

Shellskripten an, die Dolphin ausführt, um die benötigten Wertbelegungen der Attribute von hpsp1 zu bekommen[7].

```
getdomainname::COMMAND-get://'hpsp1'/'home/usr/stud/dzik/
Scripten/getdomainname hpsp1'

getservername::COMMAND-get://'hpsp1'/'home/usr/stud/dzik/
Scripten/getservername hpsp1'
```

Abbildung 4-7: Beispiel für eine System Transcript Ausgabe

Die Ausgabe bedeutet in diesem Fall, daß die Queries „getdomainname" und „getservername" aktiviert werden. Diese rufen wiederum Shellskripten auf, deren Namen hier nur zufällig mit den Querynamen übereinstimmen. Der gerufene Rechner ist hpsp1 (siehe hierzu auch Kapitel 6.6).

4.5 Der Icon Browser

Der Icon Browser ordnet den unterschiedlichen Objektinstanzen eines Models Icons zu. Hier kann man beliebige *.icn-Dateien laden und den Objektklassen aller Models zuordnen. Dabei beachtet Dolphin auch die Vererbungskriterien. Besitzt ein Objekt kein eigenes Icon, so weist ihr der Icon Browser automatisch das Icon der jeweiligen Oberklasse zu. Zu beachten bleibt bei der Vergabe eigener Icons nur, daß die Icons repräsentativ für die ihnen zugehörigen Objekte sein sollten.

[7] nur für den Fall, daß die Fact Base noch keine konkreten Wertbelegungen über das Objekt besitzt!

5 Das Modell SmbBase

Das Modell SmbBase ist im Rahmen dieses Fortgeschrittenenpraktikums entwickelt worden und stellt den zentralen Punkt dieser Arbeit dar. Bevor im nächsten Kapitel die Modellierungssprache von Dolphin anhand von SmbBase näher untersucht wird, soll dieser Abschnitt einen Einblick in die Objekthierarchie dieses Models gewähren. Es folgen zwei grafische Darstellungen, in denen verschiedene Ansätze der Objekthierarchie SmbBase gezeigt werden.

5.1 Die Objekthierarchie von Dolphin: Top-down-Ansatz

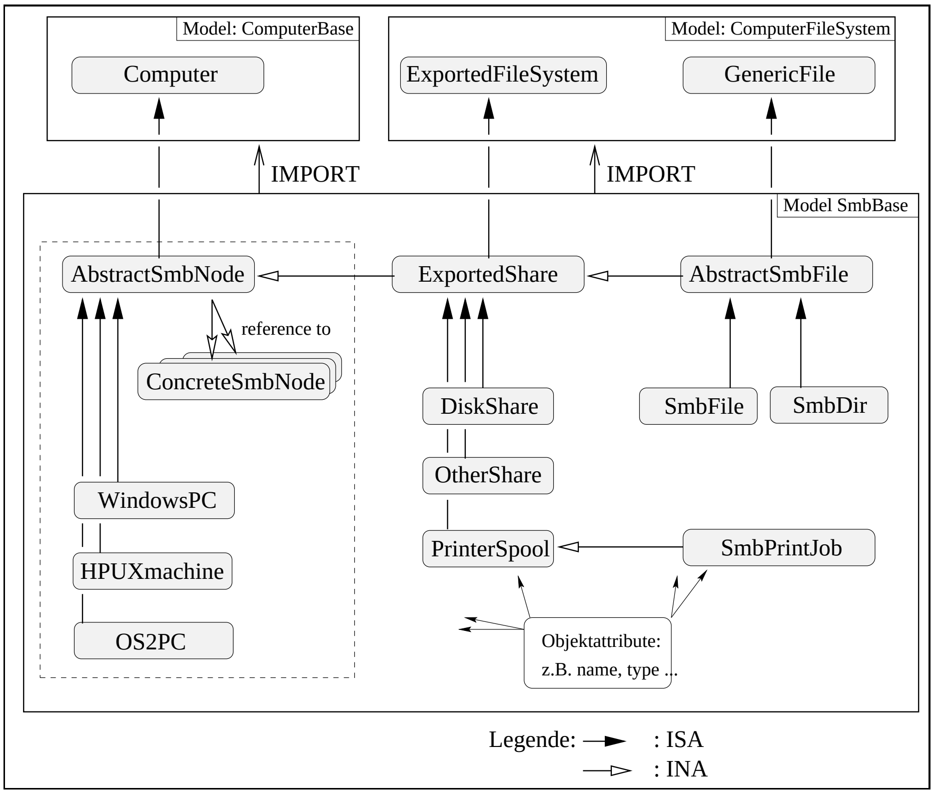

Abbildung 5-1: Die Objekthierarchie von Dolphin: Top-down-Ansatz

Die Abbildung 5-1 zeigt die Objekthierarchie gemäß eines Top-down-Ansatzes. Abbildung 5-2 stellt den Bottom-up-Ansatz dar, wie er in diesem Fortgeschrittenenpraktikum implementiert worden ist. Auf die Gründe für die Diskrepanz zwischen den beiden Ansätzen wird in diesem Kapitel eingegangen. Die Struktur des Models SmbBase wird in Kapitel 5.2 besprochen.

Der Unterschied zwischen Abbildung 5-1 und 5-2 ist auf den gestrichelten Rahmen begrenzt. Mit reference to möchte man erreichen, daß alle

SMB-Rechner, die in der Browseliste eines beliebigen SMB-Rechners vorkommen, als Referenzen auf die einzelnen SMB-Rechner in die Fact Base von Dolphin aufgenommen werden.

Dolphin erhält über die in Kapitel 3.3.3 besprochene Funktion smbclient -L immer eine aktuelle Liste aller SMB-Recher im Netz.

Mit dieser Information und einer entsprechenden Implementierung ist Dolphin in der Lage, mit einer einzigen Instanz eines SMB-Rechners durch den Benutzer (siehe Kapitel 4.3), alle anderen Peer-Rechner[8] automatisch als eigenständige Objektinstanzen zu erstellen. Der Benutzer von Dolphin muß weder wissen, wie viele und welche Peer-Rechner im Netz vorkommen, noch muß er zu einem späteren Zeitpunkt in den Instanziierungsprozeß eingreifen. Als Ergebnis erhält man von Dolphin auf dem Workspace eine vollständige Übersicht aller Peer-Rechner, dargestellt durch entsprechende Referenzsymbole.

Über die angezeigten Referenzicons kann der Administrator nun beliebig zwischen den verschiedenen Peer-Rechnern hin- und herwechseln[9] und so, „aus der Sicht der jeweiligen Peer-Rechner“ heraus, seine Managementaufgaben erfüllen.

Jede Instanz der Objektklasse ConcreteSmbNode ist demnach vollkommen eigenständig und besitzt eine eigene, gemäß Abbildung 5-1, vollständige Enthaltenseinhierarchie. Abbildung 5-2 unterscheidet sich in dieser Hinsicht. Der Bottom-up-Ansatz bietet zwar auch die Möglichkeit alle SMB-Rechner des Netzverbundes anzuzeigen. Allerdings ist man nicht mehr in der Lage, zwischen den einzelnen Peer-Rechnern zu „wechseln“ und sich ihre Attributwerte aus ihrer „eigenen“ Sicht anzeigen zu lassen. Im Vergleich zu Abbildung 5-1 besitzen die Objektinstanzen von ConcreteSmbNode aus Abbildung 5-2 keine eigenen Enthaltenseinhierarchien.

Der Top-down-Ansatz ist mit der derzeitig verfügbaren Version von Dolphin allerdings nicht zu realisieren. Da es sich bei Dolphin um einen Plattformprototyp handelt und es bis jetzt noch keine geeigneten Mechanismen zur Darstellung von Referenzen auf andere Objektklassen gibt, bleibt es abzuwarten, ob dieser Ansatz in der Zukunft verwirklicht werden kann. Bis dahin muß der Bottom-up-Ansatz verwendet werden, der sich an der Modellierungsmächtigkeit der gegenwärtigen Dolphinversion orientiert und Thema des folgenden Kapitels ist.

[8] sofern es sich dabei auch um SMB-Rechner handelt

[9] durch doppelten Mausklick auf die jeweiligen Icons

5.2 Die Objekthierarchie von Dolphin: Bottom-up-Ansatz

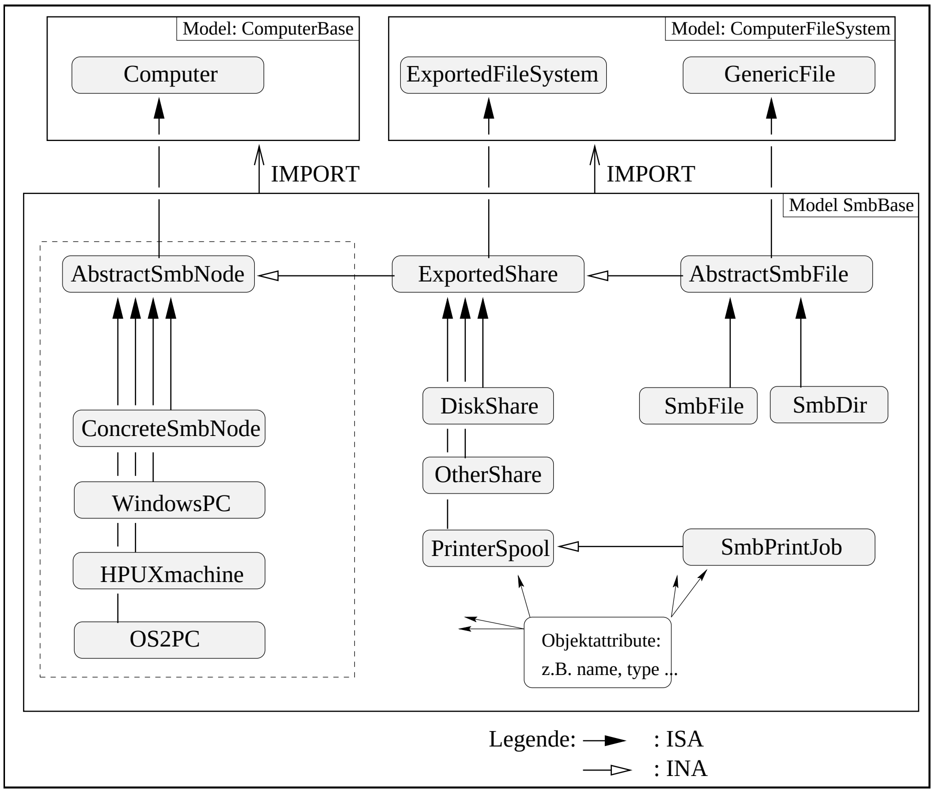

Abbildung 5-2: Die Objekthierarchie von Dolphin: Bottom-up-Ansatz

In Abbildung 5-2 ist die Vererbungs- und Enthaltenseinhierarchie des Models SmbBase dargestellt, wie sie im Model SmbBase implementiert worden ist. Die Objektklassen AbstractSmbNode, ExportedShare und AbstractSmbFile leiten sich von Objektklassen ab, deren Models nach SmbBase importiert werden. Sie werden im Sinne einer Verfeinerung der Objekteigenschaften mit zusätzlichen Attributen versehen. Damit wird eine Zuordnung von managementrelevanten Informationen möglich, die durch die SAMBA-Software zur Verfügung gestellt werden (siehe hierzu Kapitel 3.2 und Kapitel 6).

Die von diesen drei Objektklassen abgeleiteten Objekte (z.B. SmbFile, DiskShare oder WindowsPC) stellen weitere Verfeinerungen in der Vererbungshierarchie des Models SmbBase dar. Die Enthaltenseinhierarchie ergibt sich aus der Struktur, der von SAMBA ermittelbarer Informationen. Zur Verdeutlichung soll folgendes Beispiel genügen: Ein SMB-Rechner bietet anderen Rechnern Zugriff auf seine Shares an. Da jeder SMB-Rechner bestimmte Shares besitzt, werden sie in Dolphin entsprechend modelliert und mit dem INA-Konstrukt in die Enthaltenseinhierarchie eines SMB-Rechners aufgenommen. Das Objekt ExportedShare wird also zu einem Bestandteil von dem Objekt AbstractSmbNode (siehe auch Kapitel 6).

Das nächste Kapitel behandelt anhand des Models SmbBase die syntaktischen und semantischen Spezifikationen der Modellierungssprache von Dolphin. Zudem werden auch die Mechanismen ausführlich behandelt, mit denen Dolphin managementrelevante Informationen sammelt, diese in die Fact Base aufnehmen und sie anschließend auf seiner Benutzeroberfläche ausgeben kann.

6 Die Modellierungssprache von Dolphin

6.1 Der Aufbau von Objektklassen in einem Model

Die Notation von Objekten und ihren Attributen wird in der Abbildung 6-1 an einem Beispiel gezeigt und anschließend erläutert. Der Aufbau der Objektdeklarationen kann ebenfalls anhand von Abbildung 2-1 in Kapitel 2.1 nachvollzogen werden.

```
OBJECT ExportedShare ISA ExportedFileSystem INA AbstractSmbNode
DESCRIPTION 'Shares exported by a SMB node'
BASIC

[exportedShare] name [string] (M:O)
EXPLANATION 'Name of the exported SMB share' .
CATEGORY 'SMB shares'
.
[exportedShare] description [string] (M:O)
CATEGORY 'SMB shares'
.
[exportedShare] type [string] (M:O)
EXPLANATION 'Type of the exported SMB share' .
CATEGORY 'SMB shares'
.

UI
DERIVED
MAP
VIRTUAL
IDENT '[]name[]'
```

Abbildung 6-1: Das Objekt ExportedShare aus dem Model SmbBase

Die hier abgebildete Objektklasse ExportedShare gehört zu dem Model SmbBase.

- In der ersten Zeile steht der Name der Objektklasse, dessen erster Buchstabe per Konvention ein Großbuchstabe sein muß. ExportedShare erbt wegen des ISA-Konstrukts[10] die Attribute von ExportedFileSystem.

  ExportedFileSystem wird dabei in der Importzeile mit dem Model ComputerFileSystem in SmbBase importiert (siehe Abbildung 6-1 und und Abbildung 2-2 in Kapitel 2.1).

[10]ausgesprochen „is a“

Das Konstrukt INA[11] bedeutet, daß ExportedShare in der Enthaltenseinhierarchie zu einem Bestandteil eines AbstractSmbNode wird.

ExportedShare kann also nicht direkt vom Administrator erstellt werden. Vielmehr soll es als Objekt in einem AbstractSmbNode zusätzliche Möglichkeiten schaffen, Informationen über die Shares eines SMB-Rechners zu liefern. Die Instanziierung dieser Objektklasse findet zeitgleich mit der Instanziierung eines AbstractSmbNodes statt (siehe auch Kapitel 2.1.2 und 3.3.1).

- In der DESCRIPTION-Zeile steht eine kurze Beschreibung der Objektklasse.
- Unter dem Stichwort BASIC folgen die einfachen Attribute der Objektklasse.
- Die letzten Zeilen der Objektklasse bestehen aus verschiedenen Konstrukten, die in diesem Fortgeschrittenenpraktikum nicht näher behandelt werden (siehe hierzu [Pel95] und Kapitel 2.1.3).

  Im Falle von IDENT handelt es sich um ein Identifikationskonstrukt, das von Dolphin benötigt wird, wenn es mehrere Instanzen der gleichen Objektklasse erstellt. Dolphin kann so sicher gehen, auf welches konkrete Objekt es sich im einzelnen bezieht[12].

  In der Abbildung 6-1 werden die Instanzen des Objekts ExportedShare eindeutig durch den Wert des Attributs name identifiziert. Da jede Share in der Realität auch tatsächlich einen eindeutigen Namen besitzt, ist eine genaue Identifikation der einzelnen Instanzen von ExportedShare durch Dolphin gewährleistet.

6.2 Objekte aus dem Model SmbBase

Im Model SmbBase befinden sich Objektklassendefinitionen, die speziell auf die managementrelevanten Daten zugeschnitten worden sind, die durch die SAMBA-Software ermittelt werden können (siehe hierzu Kapitel 3.3). Einige dieser Objekteklassen sind z.B.:

[11] ausgesprochen „in a“
[12] vgl. Primärschlüssel bei relationalen Datenbanken

- **ExportedShare**
- **AbstractSmbNode**
- **AbstractSmbFile**

Eine genau Aufzählung aller Objektklassen befindet sich im Anhang.

Durch Instanziierung der jeweiligen Objektklassen und durch Belegung der Attribute mit den konkreten Werten, werden den Objektklassen Managementobjekte aus der realen Welt zugeordnet. Es entsteht auf diese Weise ein repräsentatives Abbild der realen Welt.

Der Rahmen für diese Abbildung wird durch die Genauigkeit und den Aufwand der Modellierung, sowie durch die Mächtigkeit der Modellierungssprache begrenzt.

In den folgenden Unterkapiteln werden einzelne Objektklassen näher untersucht.

6.2.1 ExportedShare

ExportedShare stellt einen gemeinsam benutzbaren Dienst eines SMB-Rechners in einem Peer-to-Peer-Netz dar. Diese Shares können von einem Unix-Rechner, auf dem die SAMBA-Software installiert ist, durch „smbclient -L rechnername“ angezeigt werden (siehe Kapitel 3.3 und 3.3.1).

Wie Dolphin die Objektinstanzen von ExportedShare mit den konkreten Werten initialisiert, zeigt Kapitel 6.4.

Zu beachten bleibt, daß ExportedShare die Attributeigenschaften von ExportedFileSystem erbt. Angezeigt wird dies durch das Konstrukt ISA.

Zur Verdeutlichung der Schreibweise zeigen die Abbildungen 6.2, 6.3 und 6.4 Definitionen von Objektklassen aus dem Model SmbBase.

```
OBJECT ExportedShare ISA ExportedFileSystem
                                INA AbstractSmbNode
DESCRIPTION 'Shares exported by a SMB node'
```

Abbildung 6-2: Der Definitionskopf der Objektklasse ExportedShare

6.2.2 AbstractSmbFile

AbstractSmbFile erbt die Eigenschaften der Objektklasse GenericFile und wird in die Enthaltenseinhierarchie der Objektklasse

Exportedshare eingefügt. Letztendlich bedeutet das, daß ein AbstractSmb-Node mehrere ExportedShares besitzt, die wiederum können mehrere Objektinstanzen vom Typ AbstractSmbFiles besitzen[13].

```
OBJECT AbstractSmbFile ISA GenericFile INA ExportedShare
DESCRIPTION 'Files and Directories in a Smb-Node'
```

Abbildung 6-3: Der Definitionskopf der Objektklasse AbstractSmbFile

(siehe hierzu auch Abbildung 5-2 in Kapitel 5.2)

Die restliche Syntax von AbstractSmbFile verhält sich analog zu ExportedShare in Abbildung 6-1.

6.2.3 AbstractSmbNode

Die Objektklasse AbstractSmbNode stellt einen beliebigen SMB-Rechner dar. Sie erbt die Eigenschaften der Objektklasse Computer aus dem importierten Model ComputerDynamics.

```
OBJECT AbstractSmbNode ISA Computer
```

Abbildung 6-4 Der Definitionskopf der Objektklasse AbstractSmbNode

AbstractSmbNode ist im Model SmbBase Oberklasse für eine Reihe anderer Objektklassen, wie z.B. WindowsPC oder HPUXmachine. Diese abgeleiteten Objektklassen sind Blätter im Objekthierarchiebaum und können also im Workspace durch den Benutzer von Dolphin direkt instaziiert werden (siehe Kapitel 4.3). WindowsPC oder HPUXmachine spiegeln also nach ihrer Instanziierung die einzelnen SMB-Rechner der realen Welt wieder.

Bei den Attributen von AbstractSmbNode handelt es sich um Eigenschaften der SMB-Rechner, die mit einer Query unter der Verwendung eines Shellskripts und der SAMBA-Software aus diesen Rechnern gewonnen werden können (siehe Kapitel 3.3 und Kapitel 6.5).

[13] Die Einschränkung nicht mehr als eine Abstufung in der Enthaltenseinhierarchie zuzulassen, ist mit der neuen Version Dolphin 14.18 aufgehoben worden.

6.3 Der Aufbau von Objektattributen

6.3.1 Das Attribut „name“

Objektattribute beschreiben, wie in Kapitel 2.1.3 bereits erwähnt worden ist, die Eigenschaften ihrer Objektklassen.

Attribute setzten sich aus mehreren Argumenten zusammen, wobei das erste Argument immer die Objektklasse sein muß, auf das sich dieses Attribut bezieht. Die mengenmäßige Beziehung zwischen den Argumenten wird durch einen Klammerausdruck festgelegt. Man spricht in diesem Zusammenhang auch von Attributrelationen.

```
BASIC

[exportedShare] name [string] (M:O)
EXPLANATION 'Name of the exported SMB share' .
CATEGORY 'SMB shares'
.
```

Abbildung 6-5: Das Attribut „name“ aus der Objektdeklaration von ExportedShare

Die Attributrelation aus Abbildung 6-5 besteht aus ihrem Namen und zwei Argumenten. Die Argumente werden in eckige Klammern gefaßt. Anhand dieser Attributrelationen kann der Dolphinmodul „**Inference Engine**“ feststellen, ob die Argumente in der Fact Base bereits mit Werten belegt sind oder funktionale Typüberprüfungen durchführen[14] (siehe hierzu auch [Fed95] und Kapitel 6.5, in dem die Aufgaben der Inference Engine erörtert werden).

Die Inference Engine akzeptiert die Relation name aus Abbildung 6-5 des Objekts ExportedShare nur, wenn das zweite Argument ein Objekt vom Typ String ist. String ist in diesem Falle eine Core Object Klasse (siehe auch Kapitel 2.1.2).

Wenn Dolphin mit Hilfe von Queries alle Attribute einer Objektklasse aus der realen Welt abfrägt, kann es sich also während des Abfrageprozesses vergewissern, ob die gesammelten Daten auch mit ihren Attributdefinitionen übereinstimmen. Ist das nicht der Fall, bricht Dolphin den Abfrageprozeß mit einer Fehlermeldung ab.

Das Schlüsselwort BASIC drückt aus, daß es sich hier um einfache Attribute handelt.

In der runden Klammer steht die Art der Beziehung der Obektklasse

[14] mittels Pattern Matching,

ExportedShare zur Objektklasse String[15]. Hier bedeutet (M:O), daß jede Instanz der Objektklasse ExportedShare nur einen Namen besitzen kann. Speziell diese Eigenschaft ermöglicht es Dolphin, alle verschiedenen Instanzen der gleichen Objektklasse ExportedShare zu unterscheiden (siehe hierzu Kapitel 6.1).

Die Zeile EXPLANATION beinhaltet beschreibende Angaben über das Attribut. In den Debuggern von Dolphin und im Workspacefenster erscheinen diese Erläuterungen bei Bedarf und tragen auf diese Weise zum besseren Verständnis der jeweiligen Attributwerte bei.

CATEGORY ist der Name des Registers im Workspace, unter dem die dazugehörigen Attributwerte zu finden sind. Wird das Register mit der linken Maustaste angewählt, werden alle Attributwerte dieses Registers auf einmal angezeigt. Fehlt der Registername, setzt Dolphin das Attribut automatisch in das Register Default (siehe hierzu auch Kapitel 4.3).

6.3.2 Das Attribut „isBrowseMaster“

```
[abstractSmbNode] isBrowseMaster (M)
EXPLANATION [abstractSmbNode], (' is '|' is not '),
                                'a Browsemaster'.
CATEGORY 'Browselist'
.
```

Abbildung 6-6: Das Attribut „isBrowseMaster“

Diese Attributrelation besitzt nur ein Argument. Auch in den runden Beziehungsklammern steht nur ein “M“. Die Explanation-Zeile sagt aus, daß die Objektklasse AbstractSmbNode entweder ein „Browsemaster“ ist oder nicht[16]). Das „| “-Zeichen bedeutet in der Modellierungssprache ein logisches „oder“. Trifft diese Eigenschaft isBrowseMaster in der Realität zu (nicht zu), so wird dieses Attribut im Workspace normal (rot) dargestellt. Wird mit der linken Maustaste anschließend auf das Attribut geklickt, erscheint der EXPLANATION-Text „[abstractSmbNode] is “ oder „ is not a Browsemaster“.

Die eigentliche Information, ob es sich bei einem SMB-Rechner um einen „Browsemaster“ handelt oder nicht, wird durch eine Query von Dolphin ermittelt, die zu diesem Attribut gehört. Dieser Mechanismus wir in Kapitel 6.4.2 und Kapitel 6.5 erläutert. Je nach ermitteltem Wahrheitswert, nimmt auch das Attribut isBrowseMaster den entsprechenden Wert an.

[15] Namen von Objektklassen müssen bei Attributdefinitionen mit einem Kleinbuchstaben beginnen

[16] Ein SMB-Rechner ist ein Browsemaster, wenn er eine Browseliste aller SMB-Rechner im Netz besitzt (siehe Kapitel 3.3.3

6.3.3 Die Attributrelation „[]files[]type[]“

Das Attribut []files[]type[] besitzt drei Argumente:

```
[diskShare] files [smbFile] type [string] (O:M:M)
EXPLANATION 'shows certain type of AbstractSmbFiles'.
CATEGORY 'SMB: properties'
```

Abbildung 6-7: Das Attribut „[]files[]type[]“

Dieses Attribut gehört zu der Objektklasse Diskshare, das sich von der Oberklasse ExportedShare ableitet (siehe Abbildung 5-2 in Kapitel 5.2). Mit diesem Attribut werden vom Modellierer notwendige Verfeinerungen der Oberklasse durchgeführt. Auf diese Weise kann Dolphin die von SMB-Rechnern angebotenen Shares in verschiedenen Klassen einteilen, da sie sich inhaltlich in Datei- oder Druckerdienste unterscheiden.

Handelt es sich bei einer Share um einen Dateidienst, so wird die Objektklasse Diskshare instanziiert. Bei einem Druckerdienst wird hingegen die Objektklasse PrinterSpool instanziiert[17].

Die Attribute der Objektklasse Diskshare beschreiben die erwähnten Unterschiede zur Objektklasse PrinterSpool. Zu einer Instanziierung kommt es, wenn der Benutzer von Dolphin sich eine Übersicht der „Disk-Shares“ eines SMB-Rechners anzeigen lassen will. Der dazugehörige Querymechanismus wird in Kapitel 6.4.3 behandelt.

[17] Die Bezeichnung PrinterSpool ist in Zusammenhang mit SMB-Shares schlecht gewählt. Besser wäre stattdessen PrinterShare

6.4 Der Aufbau von Queries in Dolphin

6.4.1 Die Query „getservername“

Die Query in Abbildung 6-8 heißt getservername und hat die Aufgabe, den Namen eines SMB-Rechners zu ermitteln und an Dolphin zu übergeben:

```
QUERY getservername [domainName]
COMMAND ::= '$DOLPHIN/getservername 'domainName
TARGET  LOCAL [AbstractSmbNode m].
VALID FOR 600.
PRE ::= [m]name[domainName].
RETURN ::= printableString[' ']<ps>
                            {[m] servername [ps]} TOEOL.
```

Abbildung 6-8: Die Query „getservername“

- In der ersten Zeile steht der Name der Query. Außerdem steht in eckigen Klammern die globale Variable domainname. DomainName stellt einen Eingabeparameter dar, der von Dolphin mit einem realen Objekt identifiziert werden kann. In diesem Fall nimmt domainName den Namen eines SMB-Rechners an, der vom Administrator durch einen doppelten Mausklick im Workspace ausgewählt wird (siehe Kapitel 4.3).
- Die nächste Zeile ist die Kommandozeile COMMAND. Sie beinhaltet einen Verzeichnispfad, an dessen Ende ein Shellskriptaufruf steht. Hier kann auch ein SNMP-Request oder auch ein RPC stehen. Die Eigenschaft, mit mehrere Methoden mit Netzressourcen zu kommunizieren, ist bereits in Kapitel 2.2 behandelt worden. Dolphin ersetzt domainName mit dem Namen des ausgewählten Objekts und das Shellskript verwendet ihn als Eingabeparameter. Die Ausgabe des Shellskripts wird im Anschluß mit Hilfe der RETURN-Zeile von Dolphin interpretiert und entsprechend weiterverarbeitet.
- Als nächstes folgt die TARGET-Zeile. Sie drückt aus, daß es sich bei dem angesprochenen Rechner, um einen SMB-Rechner (AbstractSmbNode) im Netz handelt. Das Konstrukt LOCAL bedeutet, daß der Befehl in der COMMAND-Zeile auf dem Rechner ausgeführt wird, auf dem Dolphin ausgeführt wird.

  Für den AbstractSmbNode wird zusätzlich eine neue globale Variable m definiert.

- Die VALID-Zeile gibt an, wie lange die durch die Query ermittelten Daten in der FactBase gespeichert bleiben (siehe hierzu Kapitel 2.2.1).

  Für diese temporäre Einschränkung gibt es wichtige Gründe. Speichert Dolphin die von Queries gesammelten Daten fest ab, so veraltet die Datenbank durch die häufig vorkommenden Änderungen in der realen Welt ziemlich schnell.

  Nach Ablauf der vorgegebenen Frist, erlischt im Workspace von Dolphin die Farbgebung der dargestellten Attribute und wird einheitlich grau. Attribute, die nicht auf der Benutzeroberfläche von Dolphin dargestellt werden, sind verloren. Erst durch Anklicken der Option Update all im Pull-Down-Menü Update des Workspace, werden die entsprechenden Queries erneut aktiviert, die Attributwerte werden ein weiteres mal gesammelt und für die Dauer des VALID-Wertes angezeigt.

  Auf diese Weise wird versucht, Speicherplatz einzusparen und die Netzlast in Grenzen zu halten. In der Abbildung 6-8 werden die Querydaten für 600 Sekunden in der FactBase gespeichert. Je nach Attributeigenschaft und der Erfahrung eines Systemverwalters muß sorgfältig abgewogen werden, welche Zeit letztlich eingestellt werden soll.

- Die nächste Zeile ist optional. Sie wird aus Sicherheitsgründen verwendet, damit es nicht zu Fehlern während des Ablaufs eines Query-Prozesses kommen kann.

  Das Schlüsselwort PRE steht für PRECONDITION. Eine Query wird nur dann ausgeführt, wenn diese Vorbedingung erfüllt ist. Ansonsten bricht Dolphin die Query ab.

  In diesem Fall wird von der Inference Engine das Attribut name überprüft. Existiert in der Fact Base eine Instanz eines AbstractSmbNodes mit dem Namen domainName, kann die Query ausgeführt werden. Ansonsten wird, noch bevor Dolphin das Netz mit seiner Queryabfrage belastet, der Queryprozeß abgebrochen, und es erscheint eine Fehlermeldung im System Transcript.

- In der RETURN-Zeile wird festgelegt, in welcher Form Dolphin die durch das Shellskript angeforderten Informationen in seine Fact Base aufnehmen wird. Die RETURN-Zeile gibt insbesondere eine Grammatik an, mit der Dolphin in der Lage ist, die vom Shellskript ermittelten Daten syntaktisch und semantisch zu interpretieren:
  - Bei der Ausgabe des Shellskripts wird es sich um einen String handeln, der in der Dolphingrammatik mit printableString bezeichnet wird. Im Grunde handelt es sich hier um eine einfache Typdefinition. In den spitzen Klammern „< >“ befindet sich eine

lokale Variable ps. Der Wert dieser Variable wird von Dolphin in die Fact Base als Wert des Attributs servername aufgenommen.

- Der in den eckigen Klammern und den einfachen Hochkommata eingeschlossene Leerzeichen (' ') sagt aus, daß der String auch aus mehreren, durch einfache Leerzeichen getrennte Wörter bestehen kann. Fehlt diese Klammer, wird nur eine Zeichenfolge bis zum ersten Leerzeichen als Wert der Variable ps zugewiesen.
- In den einfach geschweiften Klammern „{ }“ steht die eigentliche Zuweisung, in der dem Attribut servername ein konkreter Wert zugewiesen wird (siehe auch Kapitel 6.5).
  Die Variable m stellt den durch domainName ausgewählten Rechner dar. Sie ist gleichzeitig das erste Argument des Attributs. Das bedeutet, daß das Attribut servername des SMB-Rechners m mit dem Wert ps initialisert wird (siehe Kapitel 2.2.1). Dabei durchläuft ps eine Typüberprüfung durch die Inference Engine. Bei einem Typfehler (ps ist kein String) erscheint eine Fehlermeldung im System Transcript.
- TOEOL heißt „To End Of Line“ und bedeutet, daß die Ausgabe höchstens eine Zeile lang sein darf.

Die Fact Base wird auf diese Weise um den Wert des Attributs servername erweitert, und im Workspace erscheint der ermittelte Name des Servers.

Darüber hinaus muß vom Modellierer sichergestellt werden, daß das aufgerufene Shellskript korrekte Ergebnisse liefert. Im vorangegangenem Beispiel muß das Shellskript den richtigen String des Servernamens finden und ihn Dolphin übergeben (siehe hierzu die im Anhang beigefügten Shellskripten[18]).

[18] Mit dem Aufruf von smbclient und dem Programm awk wird der Servername ermittelt und ausgegeben.

6.4.2 Die Query „isBrowseMaster“

Diese Query soll mit Hilfe eines geeigneten Shellskripts entscheiden, ob es sich bei einem Rechner auch tatsächlich um einen „Browsemaster“ handelt oder nicht:

```
QUERY isbrowsemaster [domainName]
COMMAND ::= '$DOLPHIN/isbrowsemaster 'domainName
TARGET  LOCAL [BrowseMaster m].
VALID FOR 180.
PRE ::= [m]name[domainName].
RETURN ::= (0 {[m] isBrowseMaster} | 1)
                                {{[m] isBrowseMaster}} ToEOL.
```

Abbildung 6-9: Die Query „isBrowseMaster“

Bis auf die RETURN-Zeile hat sich im Vergleich zur Query getservername nichts verändert.

Auch hier wird in der COMMAND-Zeile ein Shellskript aufgerufen, das sich um die Beschaffung der benötigten Information kümmert. Stellt das Shellskript fest, daß es sich bei dem ausgewählten SMB-Rechner um einen Browsmaster handelt, übergibt es Dolphin den Wert „0“. Falls nicht, werden vom Shellskript keine Daten an Dolphin übergeben (siehe hierzu das Kapitel 3.3.1, Kapitel 6.3.2, Kapitel 6.5 und die im Shellskripten im Anhang).

Die eigentliche Zuweisung an das Attribut isBrowseMaster steht wiederum in einfachen geschweiften Klammern. Die Definition des Attributs ist bei der Objektklasse AbstractSmbNode zu finden.

Dolphin vergleicht das Ergebnis des Shellskripts mit der Zahl „0“, die sich in der Attributdefinition befindet (siehe hierzu Kapitel 6.3.2). Ist der Rechner [m] kein Browsemaster und als Ergebnis wird gar keine Ausgabe geliefert, wird gemäß Attributdeklaration der Text

„[browsemaster] is not a Browsemaster“,

rot untermalt, ausgegeben. Ist das Ergebnis des Shellskripts eine „0“, so wird der Text

„[browsemaster] is a Browsemaster“

ausgegeben. Das Queryergebnis wird natürlich ebenfalls in die Fact Base aufgenommen.

Bei den doppelt geschweiften Klammern „{{ }}“ handelt es sich um die sog. **Completeness Facts**. Completeness Facts sind Hinweise des Modellierers an Dolphin, daß die Query bereits alle in der realen Welt möglichen Wertbelegungen von Attributen erfaßt, die in den doppelt geschweiften Klammern angegeben sind. Dolphin stellt seine Suche nach weiteren Queries daraufhin ein, durch die es weitere Informationen aus der realen Welt ermitteln möchte. Die Bedienzeit kann auf diese Weise verkürzt werden.

6.4.3 Die Query „getExportedDiskShares“

In Abbildung 6-10 ist die Query getExportedDiskShares zu sehen. Sie liefert alle Shares eines SMB-Rechners als Objektinstanzen.

```
QUERY   getExportedDiskShares [domainName]
COMMAND ::= '$DOLPHIN/getdiskshares ' domainName
TARGET  LOCAL [AbstractSmbNode m].
VALID FOR 6000.
DECLARE [DiskShare s].
PRE ::= [m]name[domainName].
RETURN ::= shareLines {{[DiskShare m:_]name[_],
                         [DiskShare m:_]type[_],
                          [DiskShare m:_]description[_],
                           [m]diskshares[m:_]}}.
     shareLines   ::= shareLine+.
     shareLine ::= name type description ToEOL.
     name     ::= NEW[s] printableString<sn>
                     {[m:s] name [sn], [m] diskshares [m:s]}.
     type   ::= printableString<t> {[m:s] type [t]}.
     description   ::= printableString[' ']<d>
                     {[m:s] description [d]}.
```

Abbildung 6-10: Die Query „getExportedDiskShares“

Die meisten Teilkonstrukte sind in Kapitel 6.4.1 und 6.4.2 bereits behandelt worden.

Neu hinzugekommen ist die Zeile DECLARE. Hier wird die globale Variable s vom Typ der Objektklasse DiskShare definiert. In der RETURN-Zeile wird diese Variable benötigt werden.

In der RETURN-Zeile stehen wiederum die Regeln zum Parsen der Shellskriptausgabe in Form einer Grammatik. Die Ausgabe des aufgerufenen Shellskripts wird demzufolge mehrere Zeilen beinhalten, die sich wiederum in mehrere Informationseinheiten aufteilen.

- Falls der ausgewählte SMB-Rechner mehr als nur einen Dateidienst anbietet, müssen selbstverständlich entsprechend viele Instanziierungen der Objektklasse DiskShare durchgeführt werden. Dieser Tatsache wird in Dolphin mit shareLine+ Rechnung getragen.
- Die shareLine selbst teilt sich in drei weitere Bestandteile: name, type und description.

- Für jede vom Shellskript aus übergebene Zeile wir also ein neues Objekt vom Typ DiskShare instanziiert. Dies geschieht durch New[s]. Die Anzahl neu-instanziierter Objekte hängt also dynamisch von der Shellskriptausgabe ab. Die Objektinstanzen bekommen mit dem Attribut name sofort einen eindeutigen Namen zugewiesen (siehe Kapitel 6.1).
- Bei type und description handelt es sich jeweils um ein einfaches Attribut, das bereits bei der Query getservername behandelt worden ist.
- Die Completeness Facts stehen wieder in geschweiften Klammern, diesmal bereits am Beginn der RETURN-Zeile.

6.4.4 Die Query „getSmbDiskShareFiles“

In Abbildung 6-10 ist die Query getSmbDiskShareFiles zu sehen. Sie entspricht vom prinzipiellen Aufbau der Query getExportedDiskShare und liefert alle Dateien einer Diskshare eines ausgewählten SMB-Rechners. Da Objektklassen vom Typ SmbFile bereits die zweite Abstufung in der Enthaltenseinhierarchie der Objektklasse von AbstractSmbNode darstellen, enthält die Abbildung eine kompliziertere Syntax. Als kurzes Beispiel sei die Objektinstanz SmbFile aus Abbildung 5-2 genannt, die zur Enthaltenseinhierarchie der Objektinstanz ExportedShare gehört, welche wiederrum zur Enthaltenseinhierarchie eines AbstractSmbNodes gehört. Die Syntax der Variablen dieser Objektinstanzen hat dann folgendes Aussehen [m:exp:smbf].

Zum besseren Verständnis tragen Abbildung 5-2 in Kapitel 5.2 und das Model SmbBase selbst bei (siehe hierzu das Model SmbBase im Anhang).

```
QUERY getSmbDiskShareFiles [domainName] [diskShare]
COMMAND ::= '$DOLPHIN/getfiles ' domainName ' ' diskShare
TARGET  LOCAL [AbstractSmbNode m].
VALID FOR 6000.
DECLARE [SmbFile smbf] [DiskShare exp].
PRE ::= [m]name[domainName], [m:exp]name[diskShare].
RETURN ::= garbage lines {{[SmbFile m:exp:_]smbfilename[_],
                            [m:exp]files[m:exp:_]type[_],
                             [SmbFile m:exp:_]smbfiletype[_],
                              [SmbFile m:exp:_]smbfilesize[_],
                               [SmbFile m:exp:_]smbfiledate[_]}}.
       garbage ::= ToEOL.
       lines ::= line+.
       line ::= first second third fourth ToEOL.
       first ::= NEW[smbf] printableString<ps>
                  printableString<typ>
                  {[m:exp:smbf] smbfilename [ps],
                   [m:exp] files [m:exp:smbf] type [typ]}.
       second ::= printableString<abft>
                   {[m:exp:smbf] smbfiletype [abft]}.
       third ::= number<n> {[m:exp:smbf] smbfilesize [n]}.
       fourth ::= printableString[' ']<fd>
                   {[m:exp:smbf] smbfiledate [fd]}.
```

Abbildung 6-10: Die Query „getSmbDiskShareFiles"

6.5 Die Funktionsweise von Queries

Der Funktionsablauf einer Query gestaltet sich vereinfacht wie folgt , siehe hierzu auch [Pel95]:

- Der Administrator will die Attributeigenschaften einer Objektklasse abfragen (durch Doppelklick der linken Maustaste auf das jeweilige Objekt im Workspace).
- Stellt das Dolphinmodul **Inference Engine** fest, daß noch keine Informationen über diese Objektinstanz vorhanden sind, aktiviert er das Modul **Query Engine**. Die Inference Engine ist u.a. auch für Typüberprüfungen zuständig.
- Die Query Engine muß sich nun um die Beschaffung aller Attributwerte kümmern, die das gewählte Objekt beschreiben. Mit Hilfe der Attributnamen sucht er alle Queries ab und führt diejenigen Queries aus, die einen der gesuchten Attributnamen in der RETURN-Zeile aufweisen, d.h.

1. Die Query Engine durchsucht zuerst alle RETURN-Zeilen nach den Namen der Attribute, die für eine lückenlose Erfassung der Eigenschaften eines realen Objekts notwendig sind.
2. Anschließend ermittelt er die Namen aller in Frage kommender Queries.
3. Wird in einer Query durch doppelte geschweifte Klammern eine Completeness Fact explizit durch den Modellierer angegeben, bricht die Query Engine ihre Suche nach weiteren Queries ab.

- Dann werden von ihm alle in Frage kommenden Queries ausgeführt, indem jeweils zuerst die PRECONDITION-Zeilen überprüft werden. Erst im Anschluß darauf werden die jeweiligen RETURN-Zeilen abgearbeitet.
- Die Inference Engine führt alle notwendigen Typüberprüfungen aus und fügt, falls keine Fehler aufgetreten sind, die neuen Informationen in die Fact Base ein.

6.6 Von Queries benutzte Shellskripten

Die verwendeten Shellskripten haben zum großen Teil einen gemeinsamen Aufbau, der in der Abbildung 31 zu sehen ist:

```
#!/bin/sh

progname=`basename $0`
tmpfile=/tmp/$progname.$$
awkfile=$tmpfile.awk

smbclient -L "$1" > $tmpfile


cat <<. > $awkfile
/^\$/ &&  fourthemptyline=="yes" { exit }
/^\$/ &&  thirdemptyline=="yes" { fourthemptyline="yes"; next}
/^\$/ &&  secondemptyline=="yes" { thirdemptyline="yes"; next}
/^\$/ &&  firstemptyline=="yes" { secondemptyline="yes"; next}
/^\$/ { firstemptyline="yes"; next }
{print \$1 }
.
/usr/local/lmu/bin/gawk -f $awkfile $tmpfile

rm -f $awkfile
rm -f $tmpfile
```

Abbildung 6-11: Das Shellskript „getbrowseentry"

- Zuerst werden zwei temporäre Dateien erzeugt, die bei der Namensgebung ihre Prozeßnummer mit einbeziehen, so daß sie eindeutig identifizierbar werden. Sie werden im /tmp-Verzeichnis untergbracht bis sie nach Abarbeitung des Shellskripts wieder gelöscht werden.
- In den awkfile wird das jeweilige awk-Programm eingefügt.
- In den tmpfile kommt die Ausgabe von „smbclient -L $1". $1 ist in diesem Beispiel die globale Variable domainName, die von Dolphin an das Shellskript übergeben wird.
- Mit den emptyline-Abfragen werden die Leerzeilen der Ausgabe von smbclient -L durch awk gelöscht. Gleichzeitig wird in den Variablen (firstemptyline usw.) festgehalten, in welcher Leerzeile man sich in der Ausgabe von smbclient -L befindet. Awk gibt alle Strings, durch Leerzeichen getrennt, zeilenweise aus.
- Nun werden die mit awk präparierten Ausgaben vom Query Engine nach Dolphin übernommen.

Die Shellskripten sind ebenfalls im Anhang beigefügt. Es wird empfohlen, diese Shellskripten zumindest einmal durchzulesen, da ihre Funktionalität als Beschaffungswerkzeug von Informationen über SMB-Rechner für Dolphin von großer Bedeutung ist.

6.7 Actions

Actions sind zum Teil bereits in Kapitel 2.2.1 erwähnt worden. Mit ihnen kann ein Systemverwalter konkret Einfluß auf seine Netzressourcen nehmen. Im Rahmen dieses Fortgeschrittenenpraktikums sind ein paar Actions implementiert worden. Abbildung 6-12 zeigt eine Beispiel-Action:

```
ACTION rmSMBfile [domainName] [diskShare] [smbFile]
COMMAND ::= '$DOLPHIN/rmsmbfile ' domainName ' ' diskS-
hare ' ' smbFile
TARGET LOCAL [AbstractSmbNode m].
DECLARE [DiskShare ds]  .
PRE ::= [m]name[domainName], [m:ds]name[diskShare],
         [m:ds]files[ds:smbFile]type['file'].
POST ::= ~[m:ds]files[ds:smbFile]type['file'].
```

Abbildung 6-12: Die Action „rmSMBfile“

Auf die Syntax der Action wird hier nicht eingegangen.

Eine Action kann vom Systemverwalter im Workspace aufgerufen werden, wenn bestimmte Bedingungen erfüllt sind. Stimmen die Bedingungen mit der PRECONDITION-Zeile einer Action überein, erkennt das der Inference Engine und startet automatisch die entsprechende Action (siehe hierzu [Pel95] und [Fed95]). Das aufgerufene Shellskript in der Kommandozeile muß wiederum dafür sorgen, daß die Bedingungen nach der Ausführung der Action (POST-Zeilenbedingungen) erfüllt sind. Ansonsten kommt es zu einer Fehlermeldung.

Zu den von Dolphin aufrufbaren Actions gehören z.B. das Erstellen (create) und das Löschen (delete) von Objekten.

Die für das Model SmbBase implementierten Actions befinden sich ebenfalls im Anhang.

7 Zusammenfassung und Ausblick

Dieses Fortgeschrittenenporaktikum vermittelt einen Einblick in den Entwurf eines Models, das von der Plattform Dolphin zu Managementzwecken eines PC-Peer-to-Peer-Netzes verwendet werden kann. In diesem Zusammenhang wurde das Model SmbBase implementiert.

In der Ausarbeitung wurden am Anfang zunächst das Informations- und Kommunikationsmodell von Dolphin erläutert.

Es wurde bemerkt, daß von PC-Peer-to-Peer-Rechnern auf der Anwendungsschicht heutzutage häufig das SMB-Protokoll verwendet wird, mit denen sich die Peer-Rechner gegenseitig Zugriff auf ihre Shares gewähren.

Damit auch die auf einem Unix-Rechner residente Plattform Dolphin auf von PC-Peer-Rechnern angebotenen Datei- und Druckerdienste zugreifen kann, ist die SAMBA-Software analysiert worden. Sie bietet neben der Möglichkeit für Unix-Rechner auf die Shares von PC-Peer-Rechnern zuzugreifen, auch eine Schnittstelle an, mit der managementrelevante Informationen über die Peer-Rechner abgefragt werden können. Diese Funktionalität ist in dem Model SmbBase eingefügt worden. Zu diesem Zweck sind auch mehrere Shellskripten entwickelt worden, mit denen Dolphin in die Lage versetzt wird, SAMBA-Befehle aufzurufen.

Um den Umgang mit der Plattform Dolphin zu erleichtern, ist ein eigenes Kapitel der grafischen Benutzeroberfläche von Dolphin gewidmet worden.

Das letzte Kapitel gibt anhand von Auszügen aus dem Model SmbBase Beispiele für die Verwendung der Modellierungssprache von Dolphin an.

8 Anhang

Das in diesem Fortgeschrittenenpraktikum entwickelte Model SmbBase wird wie folgt in Dolphin installiert:

Jede Dolphin-Version besitzt neben der Image-Datei ein Unterverzeichnis „models/“. In dieses Unterverzeichnis werden alle Models, die nach Dolphin importiert werden sollen, als ASCII-Dateien abgelegt. Alle Model-Dateien müssen zusätzlich mit dem Suffix .m versehen werden.

Das Model SmbBase ist als ASCII-Datei „SmbBase.m“ abgespeichert und wird unter dem oben erwähnten Unterverzeichnis abgelegt.

Mit Hilfe des Model Editors wird das Model SmbBase nach Dolphin einfügt. Dieser Vorgang wird in Kapitel 4.2 erläutert.

Es muß zusätzlich die Pfadvariable „$DOLPHIN“ für die von den Queries und Actions verwendeten Shellskripten gesetzt werden. Die Pfadvariable verweist auf ein Verzeichnis, in dem sich die einzelnen Shellskripten befinden[19].

Für einige Actions muß eine Kennung in den Shellskripten angegeben werden, um sich in Dolphin für die Benutzung der Actions zu authorisieren. Das dazugehörige Paßwort muß ebenfalls anstelle von „secret“ im Klartext in das jeweilige Shellskript eingesetzt werden. Nur auf diese Weise können die Actions von Dolphin ausgeführt werden.

Für dieses Fortgeschrittenenpraktikum ist die Dolphinversion 14.18 verwendet worden. Die Konfiguration von Dolphin muß bei der Verwendung von Queries und Actions aus dem Model SmbBase beachtet werden. Im Drop-down-Menü „Dolphin“ im VisualWorks-Fenster können die notwendigen Einstellungen vorgenommen werden. Das „Config“-Register muß auf „System manager“ eingestellt werden, das „Remsh“-Register muß in der „User-“ als auch in der „Actor“-Zeile die eigene Kennung enthalten.

Im World-Wide-Web ist die Dokumentation zu diesem Fortgeschrittenpraktikum unter
http://www.nm.informatik.uni-muenchen.de/Forschung/dolphin_index.html
zu finden.

Es folgt nun ein Ausdruck des Models SmbBase und im Anschluß darauf ein Ausdruck von ausgesuchten Shellskripten, die im Model SmbBase ihre Verwendung finden.

[19] In diesem Fortgeschrittenenpraktikum verweist die Pfadvariable $DOLPHIN auf das Verzeichnis /users/stud/dzik/Skripten.

Literatur

[Fed95] Adam Barry Feder. *A Dolphin Model Development Environment*. MIT, http://www.nm.informatik.uni-muenchen.de/Forschung/dolphin_index.html, 1995.

[Gro90a] Network Working Group. *RFC1155, Structure and Identification of Management Information for TCP/ IP-based Internets*. http://www.internic.net/rfc/, 1990.

[Gro90b] Network Working Group. *RFC1157, A Simple Network Management Protocol (SNMP)*. http://www.internic.net/rfc/, 1990.

[Gro91] Network Working Group. *RFC1212, Concise MIB Definitions*. http://www.internic.net/rfc/, 1991.

[HA93] H.-G. Hegering and S. Abeck. *Integriertes Netz– und Systemmanagement*. Addison-Wesley, 1993.

[Kun95] Michael Kunze. *Vereinte Stärken*. Heise Verlag, April 1995.

[Mit95] Prof. Dr. Bernhard Mitschang. *Skript zur Vorlesung Datenbanken I*. Technische Universitaet Muenchen, 1995.

[Pel95] Adrian Pell. *Model Language Specification*. http://www.nm.informatik.uni-muenchen.de/Forschung/dolphin_index.html, 1995.

[Sch95] Hans Schlenker. *Fortgeschrittenenpraktikum: Management des LRZ-Systemmanagementagenten mit Dolphin*. Institut fuer Informatik der Ludwig-Maximilians-Universitaet Muenchen, Lehrstuhl Hegering, http://www.nm.informatik.uni-muenchen.de/Forschung/dolphin_index.html, 1995.

[Tri95a] Andrew Tridgell. *Manual Page nmbd(8)*. http://www.mzc.cornell.edu/msccf/software/samba/samba.intro.html, 1995.

[Tri95b] Andrew Tridgell. *Manual Page smbclient(1)*. http://www.mzc.cornell.edu/msccf/software/samba/samba.intro.html, 1995.

[Tri95c] Andrew Tridgell. *Manual Page smb.conf(5)*. http://www.mzc.cornell.edu/msccf/software/samba/samba.intro.html, 1995.

[Tri95d] Andrew Tridgell. *Manual Page smbd(8)*. http://www.mzc.cornell.edu/msccf/software/samba/samba.intro.html, 1995.