X. 500
Normierte Verzeichnisdienste in Kommunikationssystemen

ETH

Eidgenössische
Technische Hochschule
Zürich

Departement Informatik
Fachgruppe
Kommunikationssysteme

Bernhard Plattner
Cuno Lanz
Andreas Zogg

X.500: Normierte Verzeichnisdienste in Kommunikationssystemen

Mai 1989

106

Adresse der Autoren:

plattner@inf.ethz.ch
lanz@inf.ethz.ch
zogg@inf.ethz.ch

Fachgruppe Kommunikationssysteme
ETH-Zentrum
CH-8092 Zürich, Switzerland

Zusammenfassung

Verzeichnisse sind spezialisierte, verteilte Datenbanken, in welchen Gegenstände aller Art, die in Kommunikationssystemen eine Rolle spielen, mit ihren Eigenschaften festgehalten sind. Verzeichnisdienste sind für die Entwicklung künftiger Kommunikationssysteme von zentraler Bedeutung, da deren Verfügbarkeit eine notwendige, jedoch nicht hinreichende Bedingung dafür ist, dass Kommunikationsfunktionen sinnvoll und komfortabel genutzt werden können. In diesem Beitrag diskutieren wir die Anforderungen, welche an Verzeichnisdienste gestellt werden und geben als Schwerpunkt einen Überblick über die Normen für Verzeichnisdienste in offenen Systemen (X.500), wie sie Ende 1988 verabschiedet wurden. Wir wenden uns sowohl an Leser, die sich generell über normierte Verzeichnisdienste informieren wollen, als auch an Fachleute, welche einen erleichterten Zugang zu den Normen suchen.

Schlüsselwörter: Verzeichnis, Verzeichnisdienst, OSI-Anwendungsdienst, X.500, ISO 9594, verteilte Datenbanken, Namen.

Abstract

Directories are specialized distributed databases, which serve to store data about communication entities. Directory services are important for future communication systems, since their availability is a necessary however not sufficient condition for the provision of easy-to-use communication services. In this paper we discuss the requirements for directories and review the new standards that govern directories in open systems (X.500). The paper is meant to be both an introduction for the reader generally interested in the topic and a help and entry point for those who plan to study standards documents.

Keywords: directory, directory service, OSI application service, X.500, ISO 9594, distributed databases, names.

Inhaltsverzeichnis

1. Einführung

1.1. Eigenschaften von Verzeichnisdiensten

Heutige und vermehrt noch künftige Kommunikationssysteme sind komplexe Gebilde mit weltweiter Ausdehnung, die aus einer grossen Anzahl von *Gegenständen* bestehen. Unter Gegenständen verstehen wir die Bestandteile eines Kommunikationssystemes wie Rechner, Prozesse, Anwendungen, Teilnehmer etc. Damit diese in verteilten Anwendungen referenziert werden können, werden sie mit *Namen* versehen. Ein Name identifiziert einen Gegenstand, d.h. er bezeichnet ihn eindeutig. Ein Gegenstand kann durchaus mehrere Namen haben. Da Namen oft von menschlichen Benutzern verwendet werden, müssen sie *benutzerfreundlich* sein: Ein Name eines Gegenstandes sollte mit einer grossen Wahrscheinlichkeit aufgrund bekannter Eigenschaften desselben erraten werden können. Namen sollten zudem eine möglichst lange Gültigkeit aufweisen.

Für die Erstellung von Beziehungen zwischen Gegenständen benötigt ein Kommunikationssystem jedoch *Adressen.* Diese bezeichnen die *Position eines Gegenstandes* bezogen auf die Architektur des Systemes. Da sich die Adresse eines Gegenstandes aus der konkreten Realisierung eines Kommunikationssystemes ableitet, ist sie oft für menschliche Benutzer nicht geeignet. Die Adresse eines Benutzers des OSI-Netzwerkdienstes beispielsweise ist eine Grösse mit maximal 40 dezimalen Ziffern. Wenn an der Netztopologie Änderungen vorgenommen werden, besteht die Möglichkeit, dass deswegen einem Teil der Gegenstände neue Adressen zugeteilt werden müssen. Adressen sind somit als Namen ungeeignet, obwohl auch sie Gegenstände eindeutig bezeichnen.

Aus diesen einleitenden Bemerkungen leitet sich direkt die Hauptaufgabe von *Verzeichnisdiensten* ab: Sie ordnen einem *Namen* eines Gegenstandes eine Menge von Werten zu. Diese Menge umfasst nicht bloss die Adresse, sondern grundsätzlich alle wissenswerte Information in Form von Text, Sprache, Bilder, etc. Mögliche Anwendungen sind die Speicherung von Passwörtern, die Verteilung von Chiffrierschlüsseln und die Verwaltung der Daten für die Verrechnung von Leistungen. Dass mit dieser Definition zusätzliche Anforderungen entstehen liegt auf der Hand. So stellt sich sofort die Frage, ob die gespeicherte Information, das sogenannte *Verzeichnis,* schützenswert ist und deshalb nur berechtigten Zugriffen durch autorisierte Teilnehmer ausgesetzt werden darf.

Der Verzeichnisdienst umfasst Operationen für die *Abfrage* und *Mutation* des Verzeichnisses. Die Abfrageoperationen können grob in zwei Klassen aufgeteilt werden. *White Pages*-Abfragen liefern die zu einem oder mehreren gegebenen Namen gespeicherte Information, und *Yellow Pages*-Abfragen liefern die Namen derjenigen Gegenstände, welche den in der Abfrage definierten Kriterien entsprechen. Operationen für die Mani-

pulation des Verzeichnisses umfassen das Einfügen, Entfernen und Abändern von Einträgen. Eine weitere Kategorie von Operationen ist für die Verwaltung des Verzeichnissystemes notwendig.

1.2. Anforderungen an Verzeichnissysteme

Als *Verzeichnissystem* bezeichnen wir ein reales System im Sinne von OSI, welches das Verzeichnis verwaltet und den Verzeichnisdienst erbringt. Mit der Bereitstellung rechnergestützter Kommunikationsdienste - und allgemein in Rechnernetzen - müssen auch Verzeichnisdienste für Rechneranwendungen zugänglich sein. Da moderne Kommunikationssysteme verteilte Anwendungen sind, die von einer grossen Anzahl von öffentlich-rechtlichen und privaten Organisationen betrieben werden, sind auch Verzeichnissysteme verteilt.

Effiziente, verteilte Datenbanken sind auch heute noch Gegenstand der Forschung, d.h. befriedigende Lösungen sind noch nicht bekannt. Während Verzeichnissysteme in einigen Aspekten als verteilte Datenbanken betrachtet werden können, erlauben sie jedoch Vereinfachungen, welche sie einerseits als Forschungsgegenstand besonders interessant machen, anderseits auch eine effiziente Realisierung ermöglichen.

Mit verteilten Datenbanken haben Verzeichnissysteme folgende Eigenschaften gemeinsam: Die Daten werden verteilt und aus Gründen der schnellen Zugreifbarkeit repliziert. An die Verfügbarkeit werden in beiden Fällen hohe Erwartungen gestellt. In bezug auf die Konsistenz der gespeicherten Daten sind die Anforderungen jedoch unterschiedlich: Operationen auf verteilten Datenbanken müssen immer konsistente Daten garantieren; dagegen sollen solche auf Verzeichnissystemen mit einer hohen Wahrscheinlichkeit korrekte Daten liefern. Fehler sind tolerierbar, da Verzeichnissysteme viel häufiger abgefragt als mutiert werden und die gelieferten Daten für Nachfolgeoperationen, etwa zum Erstellen einer Verbindung zu einem Kommunikationspartner mit einer erfragten Adresse, verwendet werden und fehlerhafte Angaben bei dieser Gelegenheit festgestellt werden können. Verzeichnissysteme weisen eine *beschränkte Konsistenz der Daten* auf, die so definiert ist, dass nach Änderungen des Verzeichnissystemes dieses längstens für ein gegebenes Zeitintervall inkonsistent sein darf.

Verzeichnissysteme sind keine Erfindung der neuesten Zeit; Beispiele für seit längerer Zeit in Betrieb stehende Systeme sind:

- Das *DARPA Domain Name System* [Mockapetris 84], ein Verzeichnissystem für das sogenannte ARPA Internet, das von der DARPA (Defence Advanced Research Projects Agency) unter Schirmherrschaft des U.S. Verteidigungsministeriums getragen wird und einen Zusammenschluss verschiedener Rechnernetze darstellt [Quart. et al. 86].

- *Grapevine* [Birrel et al. 82] ist ein verteiltes System, das im Xerox Palo Alto Research Center entwickelt wurde und heute bei der Xerox Corporation als unternehmensinternes System in Betrieb steht. Grapevine ist in erster Linie ein elektronisches Mitteilungssystem, welches eine sichere Übermittlung von Meldungen garantiert sowie Sender wie Empfänger von Meldungen authentifiziert. Daneben hat es die Funktion eines Verzeichnisdienstes. Die Umgebung von Grapevine bilden etwa 1500 Rechner auf etwas mehr als 50 lokalen Netzen, die zum sogenannten Xerox Research Internet zusammengekoppelt sind.
- *The Clearinghouse* [Oppen et al. 83] ist ein Verzeichnissystem, das aus der Verzeichnisdienstkomponente von Grapevine entstand. Viele seiner Konzepte sind eine Weiterentwicklung von Ideen, die im "Grapevine Registration Service" realisiert sind. Auch The Clearinghouse wurde bei Xerox entwickelt.

1.3. Übersicht über die X.500-Empfehlungen

Verzeichnisdienste waren und sind Gegenstand der Normierung: Arbeitsgruppen der ISO[1], in Zusammenarbeit mit der IEC[2], diskutieren gegenwärtig Entwürfe für den internationalen Standard 9594 [ISO/IEC 88]. Die CCITT[3] hat an ihrer Plenarversammlung Ende 1988 Verzeichnisdienste unter der Bezeichnung *X.500* [CCITT 89a] als Empfehlung verabschiedet.

Im Kontext des *Referenzmodells für die Verbindung offener Systeme (OSI)* des ISO ist der OSI-Verzeichnisdienst als eine Sammlung von *Application Service Elements (ASE)* spezifiziert und somit in die gegenwärtig stark vorangetriebene Normierung der Anwendungsschicht integriert.

Die CCITT-Empfehlungen der Serie X.500 enthalten eine Übersicht über die Konzepte und das Modell eines standardisierten Verzeichnisdienstes (X.500), eine Beschreibung des Informationsmodells (X.501, X.520, X.521), eine abstrakte Dienstspezifikation (X.511) und eine Beschreibung der zu verwendenden Protokolle (X.519). Aspekte der Verteilung eines Verzeichnissystemes werden in der Empfehlung (X.518) diskutiert. Da die Normierung von Verzeichnisdiensten im Kontext der OSI-Normierung zu sehen ist, sind zusätzlich zahlreiche Bezüge auf neuere Normen der Anwendungsschicht notwendig, insbesondere auf diejenigen der Serie X.200 [CCITT 89b].

---

1 International Standard Organization

2 International Electrotechnical Commission

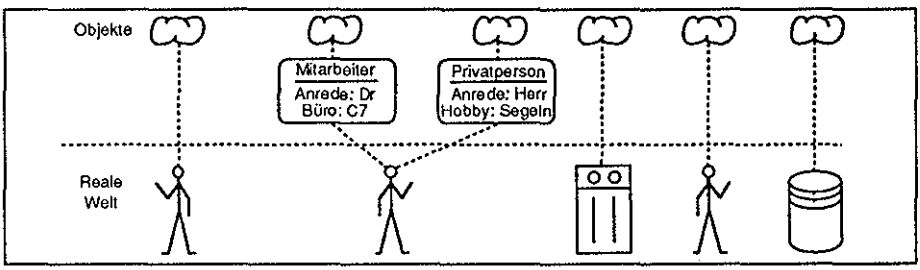
3 Comité Consultatif International Téléphonique et Télégraphique

2. Informationsmodell

Der Verzeichnisdienst benötigt zur Erbringung einer Dienstleistung eine Informationsgrundlage, die wir als *Verzeichnis* oder *Informationsbasis (Directory Information Base, DIB)* bezeichnen. Das *Informationsmodell* definiert eine Abbildung der "Gegenstände der realen Welt" auf das Verzeichnis. Diese Abbildung erfolgt in zwei Schritten. Zuerst modellieren wir die reale Welt, danach erfassen wir die relevante Information in adäquaten Datenstrukturen.

2.1. Modellierung der realen Welt

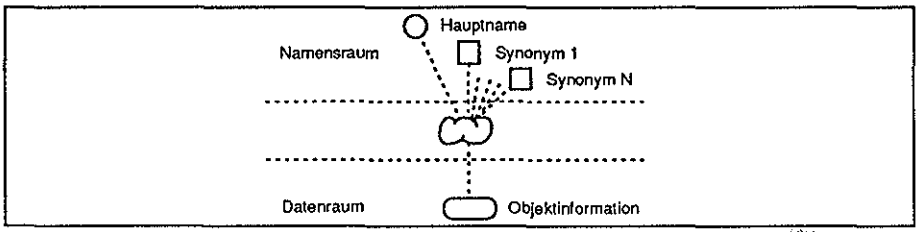
Im Umfeld von Kommunikationssystemen sind unter anderem Rechner, Prozesse, Anwendungen, Teilnehmer etc. die *Gegenstände der realen Welt*, welche für den Verzeichnisdienst auf *Objekte* abgebildet werden (Figur 2-1).



Figur 2-1: Modellierung der realen Welt durch Objekte

Da die Modellierung eines Gegenstandes vom Kontext abhängt, in dem er gesehen wird, kann er durch mehrere Objekte repräsentiert werden. Figur 2-1 zeigt beispielsweise einen Teilnehmer, der durch zwei Objekte mit verschiedener *Objektinformation* modelliert ist. Innerhalb einer Arbeitsumgebung wird er als *Mitarbeiter* und ausserhalb davon als *Privatperson* gesehen. Beim Mitarbeiter kann das Büro und bei der Privatperson das Hobby ein relevantes Merkmal sein.

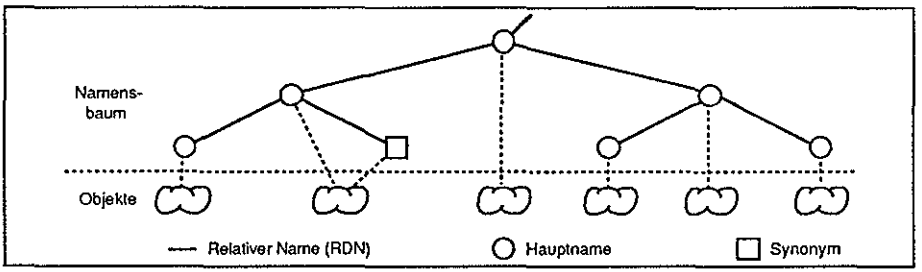
Jedes Objekt enthält neben der Objektinformation einen oder mehrere *global eindeutige Namen (Distinguished Names)*, über die es identifiziert wird. Wir verwenden dafür den Begriff *Name*. Den Namen der Ersterfassung eines Objektes bezeichnen wir mit *Hauptname*, die anderen mit *Synonym (Alias Name)*. Obwohl der Name und die Objektinformation ein Objekt beschreiben, werden *Namensraum* und *Datenraum* klar auseinandergehalten (Figur 2-2).



Figur 2-2: Unterscheidung zwischen Namensraum und Datenraum der Objekte

Sowohl der Name als auch die Objektinformation gliedern sich in *Attribute*. Attribute sind die elementaren Informationseinheiten. Ein Attribut besteht aus einem *Attributstyp (Typ)*, der unter anderem den Namen des Attributs, Gleichheitskriterien bei Vergleichen, den Wertebereich, etc. festlegt und einem oder mehreren *Attributswerten (Werten)*.

Gemäss der empfohlenen *Namenskonvention (Naming Convention)* gehen die Namen der Objekte durch Anfügen von Namenskomponenten auseinander hervor, so dass der Namensraum eine hierarchische Struktur bildet. Die Namenskomponenten nennt man dabei *relativ eindeutige Namen (Relative Distinguished Names, RDNs)* und die dabei entstehende Baumstruktur (Figur 2-3) *Namensbaum (Naming Tree)*. Jeder Knoten des Namensbaumes repräsentiert einen Namen, wobei die Wurzel den leeren Namen besitzt. Ein Sohnknoten erhält seinen Namen durch Anfügen eines RDNs zum Namen des Vaterknotens. Die verschiedenen Namen der Sohnknoten eines Knotens unterscheiden sich demnach nur im letzten RDN. Synonyme können nur durch Blattknoten des Namensbaumes repräsentiert werden.

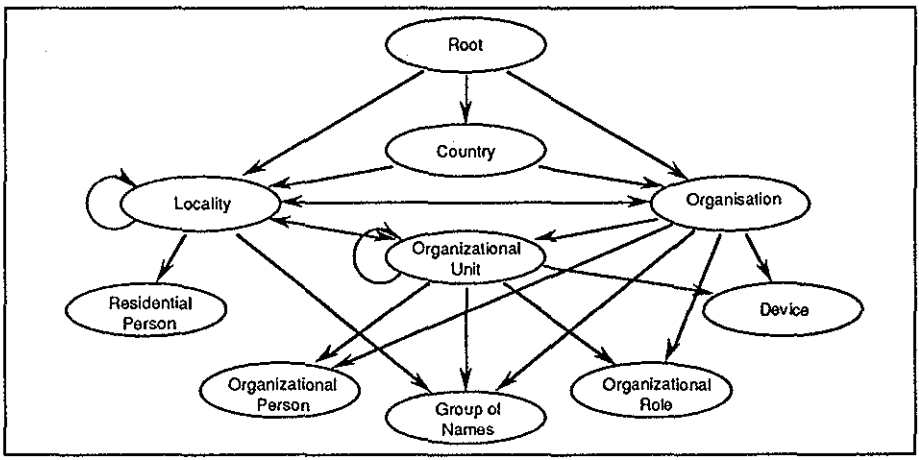


Figur 2-3: Teil des Namensbaumes

Die Objekte werden in *Objektklassen (Object Classes)* eingeteilt, die untereinander eine baumartige Beziehungsstruktur bilden, da jede Objektklasse Subklasse genau einer Superklasse ist. Die Zugehörigkeit eines Objektes zu einer Objektklasse bestimmt für dessen Eintrag die Typen, die zum Namen gehören und diejenigen, die zusätzlich zu den obligatorischen Typen der Superklasse notwendig oder optional sind. Diese hierarchische Einteilung erlaubt einen schrittweisen Aufbau komplexerer Einträge aus einfacheren. Wenn man beispielsweise wie in Figur 2-1 Teilnehmer als Mitarbeiter oder als

Privatperson modelliert, so könnte man eine Objektklasse Teilnehmer mit den beiden Subklassen Mitarbeiter und Privatperson definieren. Typen wie Anrede, die generell für die Beschreibung eines Teilnehmers benützt werden, könnten in der Superklasse festgelegt werden, so dass in den Subklassen nur noch die spezifischen Typen Büro respektive Hobby zu definieren wären.

Die *DIT-Struktur*[4] *(DIT Structure)* kontrolliert die Beziehungen *verwandter Objekte.* Zwei Objekte sind verwandt, wenn zwischen den durch sie repräsentierten Gegenstände der realen Welt eine Beziehung besteht. Die Beziehung zwischen zwei Gegenständen A und B kann mehrere Bedeutungen haben: A ist lokalisiert in B, A ist Bestandteil von B, A ist Mitglied von B, A ist Eigentum von B, A wird verwaltet von B, etc. Die DIT-Struktur ist ein gerichtetes Netz, dessen Knoten die Objektklassen sind. Die Pfeile verbinden Objektklassen mit verwandten Objekten. Die DIT-Struktur liegt ausserhalb der Empfehlungen. Figur 2-4 zeigt eine mögliche Ausprägung der DIT-Struktur.

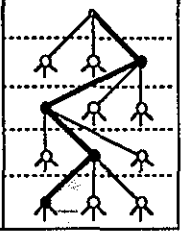


Figur 2-4: Mögliche DIT-Struktur

Die DIT-Struktur schränkt die erlaubten Ausprägungen des Namensbaumes ein. Eine mögliche Variante ist in der Figur 2-5 dargestellt. Der jeweils schwarz gefärbte Knoten jeder Hierarchiestufe des Namensbaumes ist in den drei anderen Spalten näher beschrieben. Für jeden dieser Knoten ist die zugehörige Objektklasse notiert. Die beiden mittleren Spalten bezeichnen einerseits den RDN, der den Knoten von seinen Bruderknoten unterscheidet und anderseits den Namen, der durch Aneinanderhängen der RDNs entsteht, die zu diesem Knoten führen. Die verwendeten Abkürzungen stehen für die folgenden Typenbezeichnungen: C steht für CountryName, O für OrganizationalName, OU für OrganizationalUnitName, CN für Common Name und SN für SurName.

---

[4] Die Motivation für das Akronym DIT erfolgt im nächsten Unterkapitel.

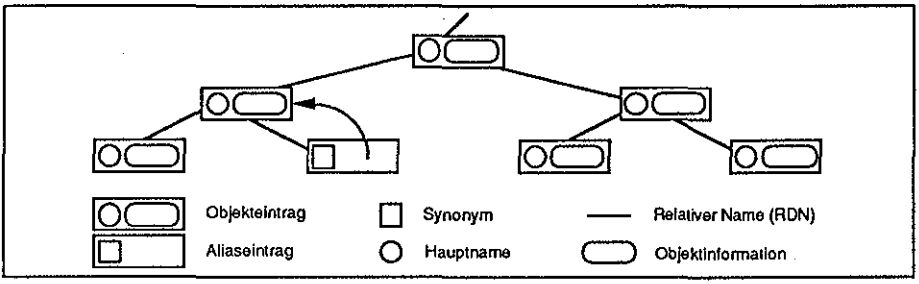
Objektklasse	RDN	Name	Namensbaum
Root	{}	()	
Country	{C = 'ch'}	({C = 'ch'})	
Organization	{O = 'eth'}	({C = 'ch'}, {O = 'eth'})	
Organizational Unit	{OU = 'abt IIIc'}	({C = 'ch'}, {O = 'eth'}, {OU = 'abt IIIc'})	
Organizational Person	{CN = 'lanz', SN = 'cuno'}	({C = 'ch'}, {O = 'eth'}, {OU = 'abt IIIc'}, {CN = 'lanz', SN = 'cuno'})	

Figur 2-5: Beispiel eines Namensbaumes

2.2. Datenstrukturen

Im folgenden werden die Datenstrukturen behandelt, welche der Repräsentation des vorangehend diskutierten Realitätsmodells dienen. Die zu erfassende Information kann in das *Verzeichnis (DIB)* und das *Verzeichnisschema (Directory Schema)* unterteilt werden. Das Verzeichnisschema enthält die vollständige Definition der Objektklassen und der Beziehungsstruktur unter den Objekten. Eine adäquate Datenstruktur zur Repräsentation des Verzeichnisschemas ist in den X.500 Dokumenten nicht enthalten.

Das Verzeichnis wird in einer einzigen globalen Datenstruktur gespeichert. Da die Objekte durch ihre Namen identifiziert werden, kann das Verzeichnis direkt aus dem Namensbaum abgeleitet werden. Es weist deshalb auch eine Baumstruktur auf, die mit *Directory Information Tree (DIT)* bezeichnet wird. Jeder Knoten des DIT (aussser der Wurzel) repräsentiert dabei einen Eintrag. Entsprechend der Unterteilung in Hauptnamen und Synonyme unterscheidet man zwischen Objekt- und Aliaseinträgen (Figur 2-6). Ein Objekt wird durch genau einen Objekteintrag und eventuell mehrere Aliaseinträge repräsentiert. Der Objekteintrag verbindet den Hauptnamen eines Objektes mit der zugehörigen Objektinformation. Ein Aliaseintrag hingegen ordnet einem Synonym den zugehörigen Hauptnamen zu und referenziert somit indirekt den entsprechenden Objekteintrag. Da Synonyme nur durch Blattknoten des Namensbaumes repräsentiert werden, sind Aliaseinträge immer Blattknoten des DIT.

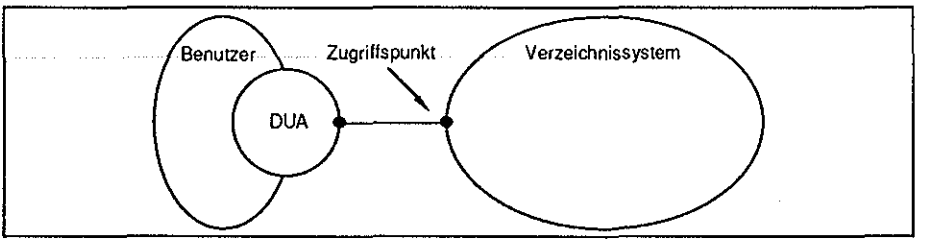


Figur 2-6: Teil des DITs

Der DIT kann unter Einhaltung von *Konsistenzbedingungen* verändert werden. Diese sind im Verzeichnisschema formuliert und besagen beispielsweise, dass kein Objekteintrag existieren darf, der nicht mit der entsprechenden Objektklassendefinition in Einklang steht. Eine weitere Konsistenzbedingung, die durch die Figur 2-6 sehr schön veranschaulicht wird, verlangt, dass keine Aliaseinträge ohne den entsprechenden Objekteintrag existieren dürfen.

3. Dienstleistungen aus der Sicht des Benutzers

Das Verzeichnissystem ist aus der Sicht des Benutzers ein unstrukturiertes System, das seine Dienste in Form von *Operationen* über *Zugriffspunkte* nach aussen anbietet. Ein Benutzer, sei dies eine Person oder ein Prozess, nimmt diese Operationen mit Hilfe eines *Benutzeragenten (Directory User Agent, DUA)* in Anspruch (Figur 3-1). Die Kommunikation zwischen DUA und Verzeichnissystem ist verbindungsorientiert. Für den Auf- und Abbau der Verbindung stehen die Operationen *Bind* und *Unbind* zur Verfügung.



Figur 3-1: Zugriff auf das Verzeichnissystem

3.1. Ports und ihre Operationen

Die Operationen, die das Verzeichnissystem zur Erbringung seines Dienstes anbietet, können durch ihre Eigenschaften klassifiziert werden. Einerseits unterscheidet man zwischen abfragenden und mutierenden Operationen; anderseits können sich Operationen auf einen einzelnen Eintrag oder auf Eintragsgruppen beziehen. Diese beiden unabhängigen Charakterisierungsmerkmale ergeben vier Operationsklassen. Da die Empfehlung keine mutierenden Operationen auf Eintragsgruppen vorsieht, verbleiben noch drei Klassen. Eine Operationsklasse bezeichnet man als *Port*. Ein Zugriffspunkt zum Verzeichnissystem muss nicht notwendigerweise alle drei Ports unterstützen. Die DUAs, die mit eingeschränkten Zugriffspunkten verbunden sind, können nur einen reduzierten Dienst anbieten. DUAs sind demnach anwendungsspezifisch konfigurierbar.

Tabelle 3-1 zeigt die Klasseneinteilung und benennt die drei Ports. Der *readPort* enthält Abfrageoperationen für einen Eintrag, der *searchPort* solche für Eintragsgruppen und der *modifyPort* Mutationsoperationen für einen Eintrag. Es wird offengelassen, ob in Zukunft für die Erweiterung der Funktionalität von Verzeichnisdiensten weitere Ports hinzukommen. So könnten zusätzliche Ports Operationen der vierten Klasse abdecken oder Abfragen und Mutation des Verzeichnisschemas erlauben.

	Abfragen	Mutieren
Einzeleintrag	readPort	modifyPort
Eintragsgruppe	searchPort	

Tabelle 3-1: Charakterisierung der Ports

Die nachfolgende Liste zeigt die Aufteilung der Operationen auf die drei Ports.

- Der *readPort* enthält die Operationen *Read, Compare* und *Abandon.* Bei Read und Compare wird der gewünschte Eintrag über den Namen des gesuchten Objektes identifiziert.
  - *Read* gibt den Inhalt eines Objekteintrages zurück. Der Benutzer kann spezifizieren, welche Attribute relevant sind und gezeigt werden sollen. Mit Read kann beispielsweise von einer Person, von der man den Namen kennt, die für elektronische Post gültige Adresse abgefragt werden.
  - *Compare* gibt einen boolschen Wert zurück, der aussagt, ob ein eingegebener Attributswert eines Objekteintrages mit dem entsprechenden Wert im Verzeichnis übereinstimmt. Compare ermöglicht unter anderem den Mechanismus zur Benutzerauthentifizierung durch das Testen von Schlüsseln oder Passwörtern.
  - Zum readPort gehört auch die Operation *Abandon.* Sie erlaubt Abfrageoperationen zu beenden, falls man an deren Resultat nicht mehr interessiert ist oder die Abfrage zu lange dauert. Es existiert jedoch keine Möglichkeit, Mutationen rückgängig zu machen.
- Der *searchPort* besteht aus den Operation *List* und *Search.* Die gewünschte Eintragsgruppe ist ein Teilbaum des DITs und wird über den Namen seiner Wurzel identifiziert. Der Eintrag dieser Wurzel heisst *Basiseintrag.*
  - *List* gibt alle Einträge aus, die sich im DIT hierarchisch unmittelbar unter dem Basiseintrag befinden. Mit dieser Operation kann man also eine Liste der Söhne eines Eintrages generieren. Unerwünschte Einträge können ausgeblendet werden. Durch wiederholte Ausführung dieser Operation kann ein Teilbaum des DITs gezielt traversiert werden.
  - *Search* beschränkt sich nicht nur auf die Söhne, sondern durchsucht den ganzen Unterbaum des Basiseintrages. Auch bei Search können unerwünschte Einträge ausgeblendet werden. Diese Operation eignet sich für Yellow Pages-Abfragen.
- Der *modifyPort* enthält die Operationen *AddEntry, RemoveEntry, ModifyEntry* und *ModifyRDN.* Der gewünschte Eintrag wird über seinen Namen identifiziert.

- *AddEntry* und *RemoveEntry* dienen dem Hinzufügen und Löschen ganzer Einträge. Sie können nur auf Blatteinträge des DITs angewendet werden. Damit kann zum Beispiel in einer Organisation ein neuer Mitarbeiter erfasst oder ein ehemaliger gelöscht werden.
- *ModifyEntry* erlaubt bei beliebigen Einträgen das Hinzufügen und Löschen von Attributstypen, sowie das Hinzufügen, Löschen und Ändern von Attributswerten.
- *ModifyRDN* ermöglicht die Änderung der letzten Namenskomponente des Namens eines Eintrages. Auch hier muss der Eintrag ein Blatteintrag des DITs sein. Die Unterscheidung zwischen ModifyEntry und ModifyRDN reflektiert die klare Trennung von Daten- und Namensraum.

3.2. Operationsparameter und ihre Informationstypen

Jede Operation besitzt einen Namen und die drei Parameter *Argument*, *Resultat* und *Fehler*. Argument ist der Eingabeparameter, Resultat und Fehler sind die Ausgabeparameter. Im Argument wird eine Operation näher beschrieben; Resultat bzw. Fehler zeigen ein erfolgreiches bzw. ein erfolgloses Beenden einer Operation an.

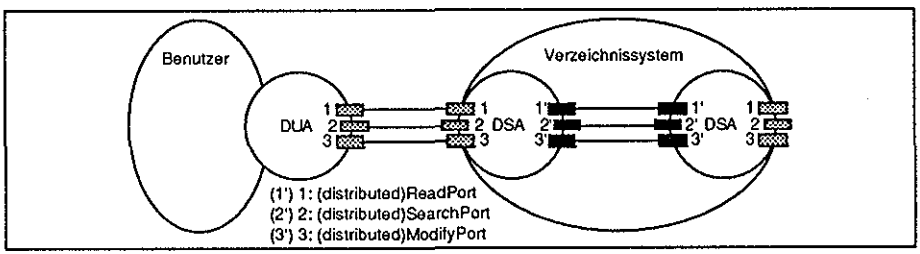
Die Struktur der drei Parameter ist durch sogenannte *Informationstypen* beschrieben. Die wichtigsten Informationstypen sind *Echtheitszeugnisketten* (Certification Path), *Beglaubigungen* (Credentials), *Filter*, *Steuerelemente* (Service Controls) und die *Eintragsinformation* (Entry Information). Auf die beiden ersten Informationstypen wird im Kapitel 6 über die Sicherheit eingegangen. Filter erlauben, die Resultatsmenge a priori einzuschränken, indem zusätzliche Bedingungen an die gesuchten Einträge geknüpft werden. Mit den Steuerelementen kann man Abfragen in bezug auf die folgenden Kriterien spezifizieren: Zeitlimite, Umfangslimite, Priorität der Abfrage, Einschränkung auf das lokale Verzeichnis, Erlaubnis, sich auf Kopien abzustützen, etc. Die Eintragsinformation schliesslich definiert die Struktur der Ausgabe, wie sie im Resultatsparameter zurückgegeben wird.

4. Aspekte der verteilten Architektur

Die Diskussion in den vorangehenden Kapiteln beschränkt sich auf diejenigen Aspekte, welche aus der Sicht der Benutzer von Bedeutung sind. Die folgenden Kapitel diskutieren nun Konzepte und Modelle, die bei der Realisierung von Verzeichnisdiensten eine wichtige Rolle spielen.

4.1. Funktionelle Verteilung

Damit der Zugang zum Verzeichnissystem von verschiedenen Orten möglich ist und unabhängig von diesen ein umfassender Dienst angeboten werden kann, müssen Verzeichnisdienste als geographisch verteilte Systeme realisiert werden. Ein verteiltes System in diesem Sinne ist ein Verbund von kooperierenden Komponenten, die gemeinsam einen Dienst erbringen. Die Komponenten des Verzeichnissystemes heissen *Directory Service Agents (DSA).* Sie erlauben einen unabhängigen Zugang von verschiedenen Orten und erbringen gemeinsam den gewünschten Dienst (Figur 4-1). Wie weiter unten erläutert wird, speichert und verwaltet jeder DSA nur einen Teil des Verzeichnisses. Um Benutzeraufträge zu beantworten, die sich auf Informationen beziehen, welche auf anderen DSAs gespeichert sind, müssen die DSAs untereinander vernetzt sein. Nur so kann ein DSA einen Benutzerauftrag in Teilaufträge zerlegen und andere DSAs mit deren Bearbeitung beauftragen.



Figur 4-1: Komponenten des Verzeichnissystems

Ein DSA erbringt seinen Dienst gegenüber DUAs und DSAs. Deshalb offeriert ein DSA die Operationen, die er den DUAs anbietet, in analoger Form auch anderen DSAs. Die zu Bind und Unbind analogen Operationen für den Auf- und Abbau von Verbindungen unter DSAs heissen *DSABind* und *DSAUnbind.* Die Operationen für die DSAs untereinander sind zu *distributedPorts* zusammengefasst. Ein searchPort beispielsweise bietet den DUAs die Operationen List und Search an und ein *distributedSearchPort* anderen DSAs die Operationen *DistributedList* und *DistributedSearch.* Die Operationen der distributedPorts benötigen als Eingabeparameter einige zusätzliche Informationstypen,

um die bei der Verteilung entstehenden Probleme in den Griff zu bekommen, d.h. die entsprechenden Operationen unterscheiden sich nur syntaktisch und nicht semantisch.

Wichtige Informationstypen sind *Originator*, *Operation Progress* und *Trace Information*.

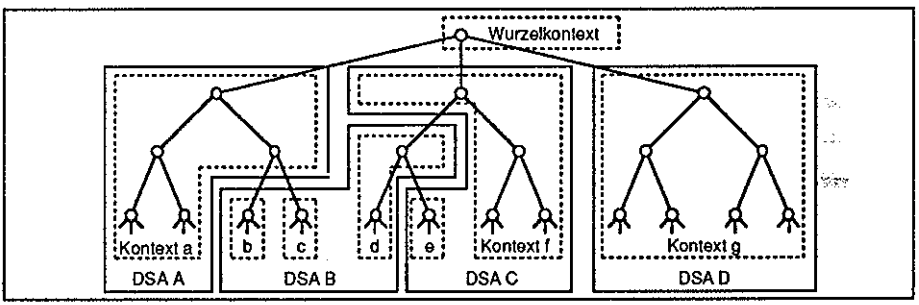
- *Originator* gibt über den Auftraggeber einer Operation genauere Auskunft, damit die ermittelten Teilresultate korrekt an diesen zurückgesendet werden können.
- Anhand von *Operation Progress* können sich DSAs, die mit Teilaufgaben von Operationen betraut sind, über deren Bearbeitungszustand informieren. Der Auftraggeber kann sich damit die Pendenz einer Operation merken.
- *Trace Information* hält den Weg eines Benutzerauftrages innerhalb des Verzeichnissystemes fest und detektiert somit Verarbeitungszyklen oder Inkonsistenzen im DIT.

4.2. Datenverteilung

Während sich der Verzeichnisdienstbenutzer nicht für die Datenverteilung interessiert, stellt sich vom Standpunkt des Verzeichnissystemes die Frage, wie man das Verzeichnis partitioniert, die dabei entstehenden Partitionen den DSAs zuteilt und das Wissen über die Verteilung verwaltet.

4.2.1. Partitionierung des DITs in Kontexte

Ein *Kontext (Namenskontext)* ist ein sich nicht zwingend bis zu den Blattknoten erstreckender Unterbaum des DIT. Den Namen des Wurzeleintrages dieses Unterbaumes bezeichnet man mit *Präfix (Kontextpräfix)*. Die Kontexte dienen einer hierarchischen Partitionierung des DITs, d.h. sie überdecken ihn vollständig, überlappen sich jedoch nicht (Figur 4-2). Ein Kontext wird von genau einem DSA verwaltet, jedoch kann ein DSA mehrere Kontexte verwalten. Der Kontext mit dem leeren Präfix heisst *Wurzelkontext*, seine hierarchisch unmittelbar untergeordneten Kontexte sind *First Level-Kontexte (FL-Kontexte)* und die DSAs, die solche Kontexte speichern *First Level-DSAs (FL-DSAs)*. In der Figur 4-2 sind a, f und g FL-Kontexte und A, C und D FL-DSAs.



Figur 4-2: Beispiel der Zerlegung eines hypothetischen DITs

Aufgrund bilateraler Absprachen, die ausserhalb der Empfehlungen liegen, können Teile des DITs redundant gespeichert werden. Die Empfehlungen treffen dazu zwei Vorkehrungen. Erstens können Aufträge spezifizieren, ob sich ihre Beantwortung auf Kopien stützen darf und zweitens geben Antworten an, ob sie aufgrund von Originalen oder Kopien entstanden sind.

4.2.2. Referenzen für die Kontextverteilung

Das Wissen über die Verteilung der Kontexte wird mittels *Referenzen* ausgedrückt. Eine Referenz gehört zu einem Kontext und bringt diesen zu einem anderen Kontext in eine hierarchische Beziehung. Es werden fünf Referenztypen unterschieden.

- *Unterreferenzen (Subordinate References)* werden für unmittelbar untergeordnete Kontexte verwendet, von denen man den RDN kennt, der vom eigenen Kontext zum untergeordneten führt. Sie bestehen aus diesem RDN, sowie aus dem Namen und der Präsentationsadresse des DSAs, der diesen Kontext speichert.
- *Nichtspezifische Unterreferenzen (Non-Specific Subordinate References)* bestehen aus dem Namen und der Präsentationsadresse des DSAs, der einen unmittelbar untergeordneten Kontext speichert, von dem man den RDN nicht kennt.
- *Überreferenzen (Superior References)* enthalten den Namen und die Präsentationsadresse des DSAs, der den unmittelbar übergeordneten Kontext speichert. FL-Kontexte und der Wurzelkontext besitzen keine Überreferenz und alle anderen genau eine.
- *Querreferenzen (Cross References)* dienen Optimierungszwecken. Sie setzen sich aus dem Präfix eines Kontextes, der nicht unmittelbar über- oder untergeordnet ist, sowie aus dem Namen und der Präsentationsadresse des DSAs zusammen, auf dem der entsprechende Kontext gespeichert ist.
- *Interne Referenzen (Internal References)* dienen dem Auffinden der Einträge innerhalb eines Kontextes und bestehen aus dem RDN dieser Einträge, sowie aus einem Verweis auf die lokale Datenbasis. Letzterer liegt ausserhalb der Empfehlungen. Für jeden Eintrag existiert genau eine interne Referenz. Damit der RDN zur Identifikation eines Eintrages genügt, muss der lokale Aufbau des DITs innerhalb des Kontextes bekannt sein. Die Speicherung dieses Wissens liegt leider auch ausserhalb der Empfehlungen.

Das Verzeichnissystem muss so organisiert sein, dass jeder DUA (über seinen verbundenen DSA) auf jeden beliebigen Eintrag zugreifen kann. Deshalb muss jeder DSA anhand der Referenzen seiner Kontexte entscheiden können, auf welchem DSA ein Eintrag gespeichert ist. Da ein Kontext wegen seiner internen Referenzen weiss, welche Einträge er selber speichert, muss ein DSA nicht von jedem Eintrag, sondern lediglich

von jedem Kontext den Speicherort kennen. Eine Referenz zu einem gesuchten Kontext muss nicht zwingend explizit vorhanden sein, sondern kann auch durch einen *Referenzpfad (Reference Path)* konstruiert werden. Um garantieren zu können, dass jederzeit ein Pfad gefunden wird, muss ein DSA mindestens eine Überreferenz und alle (nichtspezifischen) Unterreferenzen kennen. Ein DSA kann zusätzlich beliebig viele Querreferenzen speichern, um die Leistungsfähigkeit zu vergrössern.

Auf diese Weise existiert immer ein Pfad, der eventuell über den Wurzelkontext führt. Dieser spielt dabei eine Sonderrolle, da er nicht auf einem einzigen DSA gespeichert ist, sondern vielmehr auf jedem FL-DSA repliziert wird, so dass jeder FL-Kontext durch die Unterreferenzen des Wurzelkontextes alle anderen FL-Kontexte kennt. Das Wissen über FL-Kontexte wird demnach redundant gespeichert. Dieser Ansatz beruht auf der Annahme, dass ein weltweites System eine kleine Anzahl von FL-DSAs aufweisen wird, die zusammen einen leistungsfähigen Datendurchsatz, eine hohe Zuverlässigkeit und eine flexible, verteilte Autorität in der Verwaltung des ganzen Verzeichnisses aufweisen.

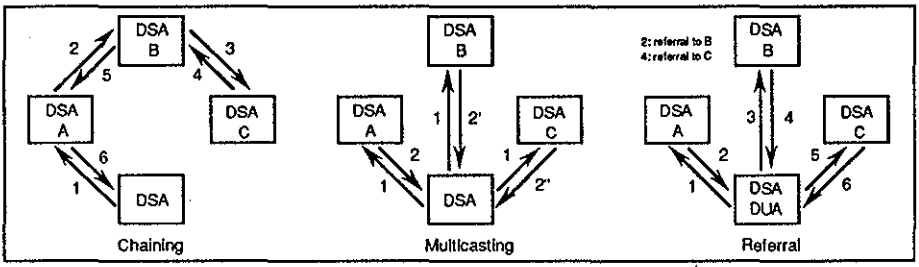
4.3. Operationelle Verteilung

Ein Benutzerauftrag kann oft nicht lokal behandelt werden, da die Information dazu auf mehreren DSAs verteilt ist. Er ist demnach keine atomare Einheit, sondern enthält mehrere Phasen der Bearbeitung. In diesen Fällen müssen gemäss den lokal vorhandenen Referenzen weitere DSAs einbezogen werden, um Teilaufträge weiterzuleiten. Aus den verschiedenen Referenztypen lassen sich mehrere Interaktionsmodi zwischen DSAs ableiten.

4.3.1. Interaktionsmodi zwischen DSAs

Man unterscheidet die drei Interaktionsmodi *Chaining*, *Multicasting* und *Referral* (Figur 4-3). Die Zahlen in der Figur geben den zeitlichen Ablauf an.

- Ein DSA benutzt *Chaining*, wenn er anhand seiner Unter-, Über- oder Querreferenzen schlüssig entscheiden kann, welcher DSA mit der Bearbeitung von Teilaufträgen in Anspruch genommen werden kann.
- *Multicasting* kommt dann zur Anwendung, wenn ein DSA aufgrund von nichtspezifischen Unterreferenzen nicht schlüssig festlegen kann, welcher DSA über die notwendigen Daten für die Bearbeitung von Teilaufträgen verfügt, oder wenn er im voraus weiss, dass mehrere DSAs involviert sind.
- Ein *Referral* ist immer eine Antwort auf einen Auftrag im Chaining- oder Multicasting-Modus, der nicht oder nur unvollständig verarbeitet werden konnte. Die Antwort im Referral-Modus enthält eine Referenz zu einem weiteren DSA, der dann im Chaining- oder Multicasting-Modus beauftragt werden kann. Im Referral-Modus kann interessanterweise auch ein DUA die Koordination übernehmen.



Figur 4-3: Interaktionsmodi zwischen DSAs

Die Wahl des Interaktionsmodus ist im allgemeinen den DSAs überlassen. Der Benutzer kann allerdings Chaining verbieten, so dass ein beauftragter DSA mit einem Referral oder einer Fehlermeldung antworten muss.

4.3.2. Phasen der Bearbeitung eines Benutzerauftrages

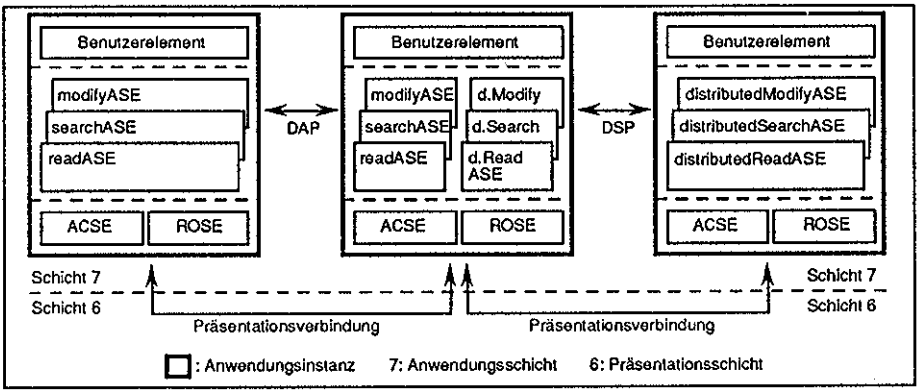
Die Bearbeitung eines Benutzerauftrages lässt sich chronologisch in die drei Phasen *Namensauflösung*, *Evaluation* und *Resultatekollektierung* unterteilen.

- Die *Namensauflösung (Name Resolution)* ermittelt für den Namen eines Eintrages anhand der verfügbaren internen Referenzen, ob der Eintrag lokal gespeichert ist. Ist dies der Fall, so ist die Namensauflösung beendet und der Auftrag kann in die nächste Phase eintreten, andernfalls wird mit den restlichen Referenzen derjenige DSA gesucht, welcher den Eintrag verwaltet.
- Die *Evaluation (Evaluation)* wird dann aufgerufen, wenn die Namensauflösung eine interne Referenz gefunden hat. Sie führt die eigentliche Operation aus. Da beispielsweise bei den Operationen *List* und *Search* mehrere Einträge betroffen sind, umfasst die Evaluationsphase für eine Operation unter Umständen mehrere DSAs. Jeder involvierte DSA liefert dabei Teilresultate für die weitere Verarbeitung.
- Sobald Teilresultate der Evaluation verfügbar sind, werden diese in der Phase der *Resultatekollektierung (Result Merging)* zu einem Gesamtresultat zusammengefügt und dieses dem Auftraggeber zurückgesendet.

5. Einbettung in das OSI-Modell

Kapitel 3 erklärt die Dienste, die das Verzeichnissystem einem Benutzer über einen DUA anbietet. Sind der DUA des Benutzers und der das Verzeichnissystem repräsentierende DSA in verschiedenen realen offenen Systemen[5] lokalisiert, so ist deren Kommunikation durch das *Verzeichniszugangsprotokoll (Directory Access Protocol, DAP)* definiert. Kapitel 4 erläutert diejenigen Aspekte, die für die Weiterleitung von Teilaufträgen unter DSAs wichtig sind. Wenn zwei kommunizierende DSAs in verschiedenen realen offenen Systemen lokalisiert sind, dann ist deren Kommunikation durch das *Verzeichnissystemprotokoll (Directory System Protocol, DSP)* definiert.

Im Sinne von OSI sind DUAs und DSAs Anwendungsprozesse. Der OSI-relevante Teil eines Anwendungsprozesses heisst *Anwendungsinstanz* (Application entity). Figur 5-1 zeigt den Aufbau dreier verzeichnisspezifischer Anwendungsinstanzen, wobei die linke zu einem DUA, die beiden anderen zu zwei verschiedenen DSAs gehören. Daraus folgt, dass die Kommunikation der linken und der mittleren Anwendungsinstanz durch das DAP, diejenige zwischen der mittleren und der rechten durch das DSP definiert ist. In beiden Fällen erfolgt die Verbindung über die Präsentationsschicht.



Figur 5-1: Einbettung in das OSI-Modell

Die Anwendungsinstanzen von DUA und DSA können grob in drei Teile aufgeteilt werden. Sie heissen: *Benutzerelement, verzeichnisspezifische Anwendungsdienstelemente* und *allgemeine Anwendungsdienstelemente.* Ein *Anwendungsdienstelement (Application Service Element, ASE)* besteht aus einer Menge von Funktionen, die den verbun-

---

[5] Ein *reales offenes System* ist ein reales System (dh. eine Menge von Rechnern, ihrer Programme, Peripheriegeräten etc.), das in bezug auf die Kommunikation mit anderen realen Systemen den Anforderungen der OSI-Empfehlungen genügt.

denen Anwendungsinstanzen eine bestimmte OSI-konforme Zusammenarbeit ermöglicht.

Das *Benutzerelement* ist die Schnittstelle zwischen der OSI-Umgebung und der Umgebung des realen Systemes. Hier stehen einem Benutzer die Verzeichnisdienstoperationen zur Verfügung.

Die *verzeichnisspezifischen ASEs* widerspiegeln die Portaufteilung und sind deshalb entsprechend benannt. Beim DAP heissen sie *readASE*, *searchASE* und *modifyASE*; beim DSP *distributedReadASE*, *distributedSearchASE* und *distributedModifyASE*. Da die verzeichnisspezifischen ASEs ihre ganze Funktionalität auf die Dienste der allgemeinen ASEs abbilden, sind das DAP und das DSP lediglich ein Satz von syntaktischen und semantischen Definitionen dieser Abbildung.

Allgemeine ASEs stellen für die verschiedensten Anwendungen unterstützende Dienste bereit. Die Agenten des Verzeichnisdienstes benützen zwei davon: *Association Control Service Element* (ACSE) und *Remote Operations Service Element* (ROSE). ACSE und ROSE erstellen, respektive verwenden über die Dienste der unterliegenden Schichten eine Präsentationsverbindung. Daraus folgt, dass sie für die eigentliche Kommunikation verantwortlich sind.

5.1. Association Control Service Element

Mit Association Control Service Element (ACSE) können logische Assoziationen (Verbindungen) zwischen einer anfragenden Anwendungsinstanz (*Initiator*) und einer antwortenden (*Responder*) aufgebaut (A-ASSOCIATE) und abgebaut (A-RELEASE) werden. Sind die beiden Instanzen miteinander verbunden, so ist ACSE bis zu einem Abbauwunsch transparent, d.h. es hat mit dem späteren Dialog zwischen den Anwendungsinstanzen nichts mehr zu tun. Zusätzlich können die Anwendungsinstanzen bzw. ACSE eine Assoziation abbrechen (A-ABORT, bzw. A-P-ABORT).

5.2. Remote Operations Service Element

Viele Anwendungen sind asymmetrisch in dem Sinne, dass Teile davon die Rolle eines *Client* und andere diejenige eines *Servers* übernehmen. Bei einem Verzeichnissystem ist Asymmetrie einerseits zwischen DUA und DSA und anderseits zwischen DSAs untereinander festzustellen. Remote Operations Service Element (ROSE) unterstützt das Client-Server Modell, indem es eine allgemeine Methode liefert, um Operationen auf fremden Anwendungsinstanzen ausführen zu lassen. Die aufrufende Anwendungsinstanz *(Invoker)* ruft dabei eine Operation auf, die sie auf einem anderen System ausgeführt haben möchte. ROSE sendet diese Operation zur ausführenden Anwendungsinstanz *(Performer)* und bringt sie dort zur Ausführung (RO-INVOKE). Das Resultat der Operation macht dann den umgekehrten Weg zurück (RO-RESULT). ROSE überträgt für

diesen Zweck den Informationsaustausch, der bei der lokalen Ausführung der Operation über die Ein- und Ausgabeparameter erfolgt wäre, auf die Argumente von RO-INVOKE und RO-RESULT. Der Performer kann einen entdeckten Fehler an den Invoker mit RO-ERROR zurückgeben. Zusätzlich können die Anwendungsinstanzen bzw. ROSE die Ausführung von Aufträgen verweigern (RO-REJECT-U, bzw. RO-REJECT-P).

ROSE unterteilt die auszuführenden Operationen in *Operationsklassen* und die Verbindungen zwischen den Anwendungsinstanzen in *Assoziationsklassen.*

Die Operationen werden nach der zeitlichen Abhängigkeit von Aufruf und Antwort und dem Antwortverhalten des Performers charakterisiert. Die Operationen des DAP und des DSP gehören dabei zur Operationsklasse 2, die vorschreibt, dass Invoker und Performer asynchron arbeiten und vom Performer verlangt, dass er sowohl Erfolg als auch Misserfolg einer ausgeführten Operation zurückmeldet.

Die Verbindungen werden danach klassifiziert, welche Anwendungsinstanz Operationen aufrufen darf. Das DAP benutzt Assoziationsklasse 1, die nur dem Initiator einer Verbindung erlaubt, Operationen aufzurufen. Da immer der DUA der Initiator ist, bedeutet dies, dass nur er Operationen auf dem DSA zur Ausführung bringen kann, nicht aber umgekehrt. Das DSP benutzt Assoziationsklasse 3, die sowohl dem Initiator als auch dem Responder erlaubt, Operationen aufzurufen.

6. Sicherheit

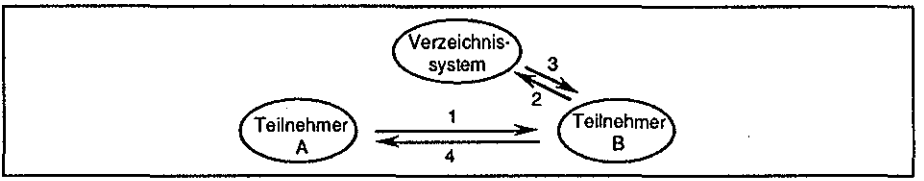
Für Verzeichnisdienste können in bezug auf die Sicherheit zwei Aufgabenbereiche unterschieden werden. Einerseits muss die Kommunikation eines Benutzers mit dem Verzeichnissystem sicher gestaltet werden, und anderseits kann der Verzeichnisdienst Mechanismen bereitstellen, die den Informationsaustausch unter beliebigen Teilnehmern eines Kommunikationssystemes sichern helfen.

- Bei der *Kommunikation mit dem Verzeichnissystem* müssen sich Benutzer und Verzeichnissystem gegenseitig authentifizieren, damit einerseits der Benutzer die Gewissheit erhält, auch tatsächlich mit dem gewünschten Diensterbringer in Verbindung zu stehen und anderseits das Verzeichnissystem die Zugriffsrechte auf das Verzeichnis kontrollieren und die beanspruchten Leistungen verrechnen kann. Daneben muss das Abhorchen, Duplizieren, Fehlleiten und Abändern von Information bei deren Austausch verhindert werden.
- Beim *Informationsaustausch unter beliebigen Teilnehmern* eines Kommunikationssystemes bestehen dieselben Sicherheitsansprüche, die mit den Begriffen *Teilnehmerauthentifizierung* und *Informationsauthentifizierung* zusammengefasst werden können. Das Verzeichnissystem kann dazu Passwörter, Beglaubigungen etc. speichern.

Der zweite Aufgabenbereich ist allgemeiner, da das Verzeichnissystem selber als ein Teilnehmer des Kommunikationssystemes aufgefasst werden kann. Das in den X.500 Empfehlungen definierte Sicherheitskonzept unterstützt diesen allgemeinen Fall. Es unterscheidet dabei zwischen zwei verschiedenen Sicherheitsstufen, die nun näher betrachtet werden.

6.1. Einfache Authentifizierung

Die *Einfache Authentifizierung (Simple Authentication)* ermöglicht nur die Teilnehmerauthentifizierung. Sie basiert auf Passwörtern, die im Verzeichnis gespeichert werden. Die Teilnehmer authentifizieren sich gegenseitig durch unverschlüsselte Übermittlung ihres Namens und ihres Passwortes. Die einfache Authentifizierung sollte sich deshalb auf die lokale Kommunikation beschränken. Die Figur 6-1 zeigt ein Beispiel, bei dem sich ein Teilnehmer A bei einem Teilnehmer B authentifiziert. A sendet seinen Namen und sein Passwort an B (1). Dieser kann mit der Operation Compare das erhaltene Passwort mit dem im Verzeichnis hinterlegten vergleichen (2). Nach Erhalt der Antwort vom Verzeichnis (3), kann B an A den Erfolg oder Misserfolg der Authentifizierung zurückmelden (4).



Figur 6-1: Mögliches Szenario bei schwacher Authentifizierung

6.2. Starke Authentifizierung

Erst die *Starke Authentifizierung (Strong Authentication)* unterstützt neben der Authentifizierung von Teilnehmern auch den Schutz der übermittelten Information. Es wird keine spezielle Verschlüsselungsfunktion vorgeschrieben, doch ist eine Technik empfohlen, die zur Familie der Public-Key Kryptosysteme gehört und in [Diffie 76] beschrieben ist. Diese Technik verwendet eine Funktionsklasse, die folgende Eigenschaften besitzt:

- Eine Funktion f aus dieser Klasse ist durch einen teilnehmerspezifischen Schlüssel k parametrisiert. Diese Parametrisierung wird durch Indexierung ausgedrückt: f_k.
- Jeder Teilnehmer i besitzt einen öffentlichen Schlüssel p_i und einen geheimen Schlüssel s_i. Daraus resultieren zwei Funktionen f_{p_i} und f_{s_i} zur Verschlüsselung.
- Ein Schlüssel eines Teilnehmers ist aus der Kenntnis des anderen nicht konstruierbar.
- Die beiden Funktionen f_{p_i} und f_{s_i} eines Teilnehmers sind einzeln nicht invertierbar, jedoch zueinander invers und kommutativ ($f_{p_i} \circ f_{s_i} = I = f_{s_i} \circ f_{p_i}$).

Aus diesen Eigenschaften lässt sich folgendes ableiten:

- Wird ein Text x mit f_{p_i} verschlüsselt, so kann er nur durch Teilnehmer i gelesen werden, denn nur er besitzt s_i, mit dem er $f_{s_i} \circ f_{p_i}(x) = x$ anwenden kann.
- Ein mit f_{s_i} verschlüsselter Text muss von Teilnehmer i stammen, denn nur er besitzt s_i.
- Falls vorausgesetzt wird, dass jeder Teilnehmer einen ausgezeichneten Namen besitzt und sein öffentlicher Schlüssel im Verzeichnis gespeichert ist, so ist jeder Teilnehmer durch den Besitz seines geheimen Schlüssels identifizierbar.

Das grösste Problem bei dieser Technik ist die Authentifizierung der öffentlichen Schlüssel. Damit ein Teilnehmer sicher sein kann, dass ein öffentlicher Schlüssel p_i eines Teilnehmers i echt ist, muss er sie von einer Quelle erhalten, der er vertraut und welche die Echtheit von p_i garantieren kann. Für diesen Zweck stellen sich vertrauende Teilnehmer i und j gegenseitig Echtheitszeugnisse $f_{s_i}(p_j)$ und $f_{s_j}(p_i)$ aus, die auch im Verzeichnis gespeichert werden. Mit $f_{s_i}(p_j)$ kann Teilnehmer j zu einem Teilnehmer k

(der in i vertraut und somit p_j besitzt) gehen und belegen, dass er wirklich Teilnehmer j ist. Dadurch ist das Problem der Authentifizierung von öffentlichen Schlüsseln auf das Finden einer lückenlosen *Echtheitszeugniskette (Certification Path)* reduziert.

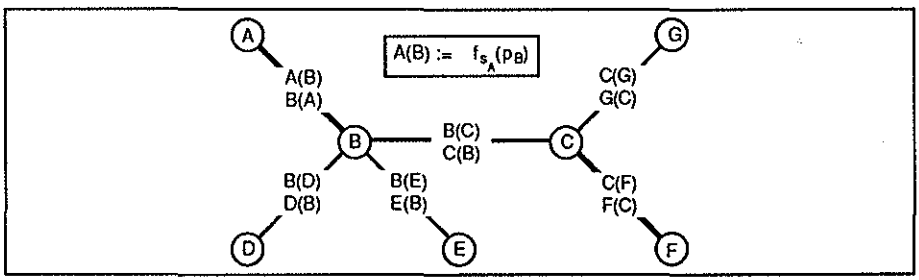
Figur 6-2 zeigt ein Netz von Vertrauensbeziehungen zwischen Teilnehmern eines Kommunikationssystemes. Die Knoten dieses Netzes sind Teilnehmer, die Kanten Vertrauensbeziehungen. Durch gegenseitiges Austauschen von Echtheitszeugnissen zwischen sich vertrauenden Teilnehmern können Echtheitszeugnisketten erstellt werden. Möchten beispielsweise die Teilnehmer A und F kommunizieren, so müssen sie sicher sein, dass sie korrekte öffentliche Schlüssel voneinander besitzen. Es ist ersichtlich, dass sich A und B, B und C und schliesslich C und F vertrauen. A kann deshalb die Zeugnisse A(B), B(C) und C(F) anfordern und daraus den öffentlichen Schlüssel von F folgendermassen berechnen:

$f_{pA}(A(B)) = f_{pA}(f_{sA}(pB)) = pB, \quad \Rightarrow \quad pB;$

$f_{pB}(B(C)) = f_{pB}(f_{sB}(pC)) = pC, \quad \Rightarrow \quad pC;$

$f_{pC}(C(F)) = f_{pC}(f_{sC}(pF)) = pF, \quad \Rightarrow \quad pF.$

Damit F den öffentlichen Schlüssel von A erhält, muss er sich die Zeugnisse F(C), C(B) und B(A) beschaffen und dann analog vorgehen. Als Nebenprodukt erhalten A und F auch gleich die öffentlichen Schlüssel aller Teilnehmer der Echtheitszeugniskette.



Figur 6-2: Vertrauensnetz für die starke Authentifizierung

7. Ausblick

Die vorangehenden Kapitel zeigen, dass ein Verzeichnissystem gemäss X.500 ein komplexes verteiltes System darstellt; entsprechend aufwendig werden auch konforme Implementationen sein. Trotzdem muss betont werden, dass X.500 nur ein Minimum der wünschbaren Konzepte und Funktionen umfasst. Im wesentlichen handelt es sich um

- eine Definition einer Struktur für einen hierarchischen Namensraum, der verschiedenen namensgebenden Instanzen Freiheit in bezug auf die konkrete Gestaltung desselben lässt,
- Richtlinien für eine sinnvolle Gestaltung dieses Namensraumes,
- eine Spezifikation von Abfrage-, Such- und einfachen Mutationsoperationen,
- eine Vorschrift über die Zusammenarbeit von DSAs, welche von verschiedenen Organisationen betrieben werden.

Zur Beantwortung der Frage, welche wünschbaren Funktionen in X.500 nicht definiert sind, lohnt es sich, einen Blick auf die Geschichte dieser Empfehlung zu tun. Dabei stellt man fest, dass in den Jahren 1984 bis 1986 Vorversionen diskutiert wurden, welche zusätzlich zu den Funktionen der heutigen Version weitere Eigenschaften aufwiesen:

- Der vorgeschlagene Namensraum hatte nicht die Struktur eines Baumes, sondern eines Graphen ohne Zyklen. Damit konnten Objekte (auch ohne den Alias-Mechanismus) mehrere Namen haben. Der Begriff des *Distinguished Name* ist eigentlich ein Relikt aus diesen Entwürfen und wurde früher verwendet, um einen von mehreren möglichen Namen besonders auszuzeichnen.
- Mit dem Konzept von *deskriptiven Namen* versuchte man, irgendwelche über Objekte gespeicherte Information zu deren Benennung zu verwenden. Ein deskriptiver Name für einen hypothetischen Mitbürger - als Menge von Attributen gegeben - könnte sein: *(AnzahlHaustiere = 3; Hobby = Segeln; Beruf = Elektriker)*. Es stellt sich natürlich sofort die Frage, ob diese Spezifikation überhaupt ein Name ist, d.h. ob sie genau ein Objekt bezeichnet. Effiziente, verteilte Algorithmen, die diese Frage beantworten können, sind noch ein Thema der Forschung.

  Die Attraktivität von deskriptiven Namen besteht darin, dass sie den Unterschied zwischen White- und Yellow-Pages-Diensten zum Verschwinden bringen.
- Unterstützung von replizierten Datenbeständen durch *Schattenkopien (Shadowing)*: Ein DSA konnte einen Teil des Datenbestandes eines anderen DSA zum Zweck einer lokalen Speicherung und damit eines rascheren Zugriffes anfordern und gleichzeitig den Lieferanten verpflichten, ihn über eventuelle Änderungen zu informieren.

- Mutationen beschränkten sich nicht auf das Entfernen oder Erzeugen von Blattknoten; vielmehr konnten beliebige Knoten des DIB-Graphen[6] entfernt oder neu eingefügt werden.
- Ein Konzept für die Vergabe von Zugriffsrechten auf Einträgen, basierend auf sog. *Zugriffskontrollisten (Access Control Lists)*.

Die Vorversionen scheiterten denn auch vor allem wegen ihrer Komplexität; nicht nur entstanden begründete Zweifel an der Implementierbarkeit derartiger Normen, man hatte insbesondere auch Schwierigkeiten, eine korrekte Definition der Semantik von komplexen Operationen (z.B. des Entfernens eines Knotens innerhalb des DIB-Graphen) zu finden. Dabei war eines der Hauptprobleme die Sicherung der Konsistenz des DIB-Graphen. Es ist jedoch zu erwarten, dass mit den Erfahrungen bei der Entwicklung der ersten Implementationen und dem Betrieb von Pilotsystemen einige der alten Ideen wieder aktuell werden und anlässlich einer Revision der Empfehlungen wieder diskutiert werden. Wir erwarten, dass vor allem bezüglich der Zugriffskontrolle und dem Zugang zu Informationen über das Verzeichnisschema Erweiterungen notwendig sein werden.

Den Autoren sind gegenwärtig eine kleine Zahl von Projekten bekannt, welche die Entwicklung von X.500-konformer Software zum Ziel haben. An der Hannover Messe (CeBIT '89) wurden denn auch erste Prototypen gezeigt. Es kann somit erwartet werden, dass in einem bis zwei Jahren eine Anzahl experimenteller und kommerzieller Systeme interessierten Anwendern zur Verfügung stehen wird. Dabei sollte nicht erwartet werden, dass die Verfügbarkeit einer Implementation allein schon zu einem korrekt funktionierenden weltweiten Verzeichnis führen wird; vielmehr werden die Hauptprobleme im organisatorischen Bereich, d.h. im Zusammenspiel zwischen den Betreibern der einzelnen DSAs, liegen.

Erste diesbezügliche Erfahrungen werden Pilotprojekte im universitären Rahmen, sowohl in Europa als auch in den USA, liefern. Zudem ist geplant, an wichtigen Messen (ähnlich wie bei der Einführung von X.400) sogenannte *Multivendor Shows* aufzubauen, in welchen X.500-Prototypen oder -Produkte verschiedener Hersteller gemeinsam ein Verzeichnissystem bilden.

Aus diesen Aktivitäten werden ebenfalls wertvolle Erkenntnisse betreffend das Testen von Implementationen in bezug auf ihre Normenkonformität gewonnen werden. Es ist anzunehmen, dass wie bei X.400 Testsuiten für X.500 entwickelt und normiert werden.

X.500 wird sich, trotz der hier genannten Einschränkungen, unzweifelhaft als ein wichtiges und unentbehrliches Bindeglied zwischen den verschiedenen OSI-Anwendungen beweisen.

---

[6] Man beachte, dass hier der Ausdruck DIT fehl am Platze wäre, da die DIB keine Baumstruktur aufwies.

A. Literaturverzeichnis

[Birrel et al. 82] Birrel A. D., Levin R., Needham R. M., Schroeder M. D.:
"Grapevine: An Exercise in Distributed Computing",
Communications of the ACM, 25 (4), April 1982.

[CCITT 89a] Comité Consultatif International Téléphonique et Télégraphique:
"Data communication networks: directory",
X.500: Overview of Concepts, Models and Services,
X.501: Models,
X.509: Authentication Framework,
X.511: Abstract Service Definition,
X.518: Procedures for Distributed Operation,
X.519: Protocol Specifications,
X.520: Selected Attribute Types,
X.521: Selected Object Classes,
Blue Book, Volume VIII, Fascicle VIII.8 (X.500 Series),
International Telecommunication Union, Geneva, 1989.

[CCITT 89b] Comité Consultatif International Téléphonique et Télégraphique:
"Data communication networks: Open Systems Interconnection (OSI)",
Blue Book, Volume VIII, Fascicle VIII.4 & VIII.5 (X.200 Series),
International Telecommunication Union, Geneva, 1989.

[Diffie 76] Diffie W., Hellman M.E.:
"New Directions in Cryptography",
IEEE Transactions on Information Theory, IT-22, No. 6, Nov. 1976.

[ISO/IEC 88] Internat. Standard Organization / Internat. Electrotechn.Commission:
"Information processing systems - Open Systems Interconnection - The Directory",
Draft International Standard ISO/IEC DIS 9594, 1988.

[Mockapetris 84] Mockapetris P. V.:
"The Domain Name System",
Computer-Based Message Services, Elsevier Science Publishers B.V., 1984.

[Oppen et al. 83] Oppen D. C., Dalal Y. K.:
"The Clearinghouse: A Decentralized Agent for Locating Named Objects in a Distributed Environment",
ACM Transactions on Office Automation Systems 1(3), July 1983.

[Quart. et al. 86] Quarterman J. S., Hoskins J. C.:
"Notable Computer Networks",
Communications of the ACM, Vol. 29, Nr. 10, Oct. 1986.

B. Figuren- und Tabellenverzeichnis

C. Stichwortverzeichnis