

Synchronisation von Verzeichnisdiensten
am Beispiel von Novell NDS und Microsoft ADS

DIPLOMARBEIT

zur Erlangung des akademischen Grades
Diplomingenieur (FH)
an der Fachhochschule für Technik und Wirtschaft Berlin

Fachbereich Ingenieurwissenschaften I
Studiengang Technische Informatik

Betreuer: Doz. Peter Puschmann
Dr. Thomas Schmidt
Eingereicht von: Taito Radtke im WS 2003/2004

Berlin, 12. Januar 2004

Inhaltsverzeichnis

1	Verzeichnisdienste	7
1.1	Evolution	7
1.2	Allgemeiner Aufbau von Verzeichnisdiensten	8
1.3	Arten von Verzeichnissen	10
1.4	Novell Directory Services (NDS)	11
1.4.1	Replikation der Verzeichnisdaten	12
1.4.2	Verzeichnisinterne Synchronisation	12
1.4.3	Partitionierung	12
1.4.4	Sicherheit	12
1.5	Active Directory Services (ADS)	13
1.5.1	Replikation und Partitionierung	14
1.5.2	Verzeichnisinterne Synchronisation	14
1.5.3	Sicherheit	14
1.6	Die Rolle von LDAP	15
1.6.1	Die Entstehung von LDAP	15
1.6.2	LDAP als Standalone-Server	16
1.6.3	Der LDAP Version 3 Standard	17
1.6.4	Vor- und Nachteile von LDAP	18
1.7	Zusammenfassung	18
2	Aspekte für den Abgleich von Verzeichnisdiensten	20
2.1	Grundlegende Anforderungen an einen Abgleichmechanismus	21
2.1.1	Heterogenität	21
2.1.2	Offenheit	21
2.1.3	Sicherheitsanforderungen	22
2.1.4	Skalierbarkeit	22
2.1.5	Fehlerverarbeitung	23
2.2	Synchronisationsmethoden zur Verteilung von Daten	24
2.2.1	Unidirektionale Synchronisation	24
2.2.2	Bidirektionale Synchronisation	25
2.3	Methoden zur Auslösung von Synchronisationsvorgängen	26
2.4	Zeitsynchronisation zur Ordnung von Änderungen	26
2.5	Granularität der Änderungserkennung	28
2.6	Methoden zum Zusammenfassen von Verzeichnisinformationen	29
2.6.1	Join – Zusammenfassung von Informationen	29
2.6.2	Virtuelles Verzeichnis – Zentrale Verlinkung von Daten	30

2.7	Funktionalitäten für einen Informationsabgleich	31
2.8	Passwortsynchronisation	33
2.9	Performanceanforderungen an Abgleichmechnismen	34
2.10	Administration der Synchronisation	34
2.11	Anforderungsprofil für eine Lösungsfindung	35
3	Lösungsansätze für die Verzeichnisdienstsynchronisation	37
3.1	Lose Offlinekopplung	37
3.2	Programmierschnittstellen und Entwicklungsumgebungen	39
3.2.1	LDAP C API	40
3.2.2	Perl	40
3.2.3	Java	40
3.2.4	ADSI	41
3.3	Microsoft Directory Synchronization Service (MSDSS)	42
3.4	Metaverzeichnisse	43
3.4.1	Aufbau eines Metaverzeichnisses	44
3.4.2	Metaverzeichnis als zentraler Informationsspeicher	44
3.4.3	Metaverzeichnis als Informationshändler	45
3.4.4	Produkte	45
3.4.5	Zusammenfassung	48
3.5	Directory Services Markup Language – die Zukunft?	49
3.6	Zusammenfassender Vergleich der Lösungsansätze	51
4	Nutzerdatenabgleich zwischen NDS und ADS am Beispiel FHTW	54
4.1	Ausgangslage	54
4.2	Zielstellung	57
4.3	Lösungskonzept	58
4.3.1	Wahl des zentralen Verzeichnisses und der Synchronisationslösung	58
4.3.2	DirXML als Synchronisationslösung	60
4.3.3	ADS Anforderungen	66
4.3.4	Attributzuweisung zwischen NDS und ADS	68
4.3.5	Synchronisationsfrequenz	69
4.3.6	Auswahl der zu synchronisierenden Objekte	70
4.3.7	Performanceanpassungen	70
5	Implementierung	72
5.1	Aufbau der Synchronisationsstruktur	72
5.2	Datenfluss	73
5.3	Filter	74
5.4	Regeln	75
5.4.1	SubscriberMatchingRule	75
5.4.2	SubscriberCreateRule	76
5.4.3	SubscriberPlacementRule	76
5.4.4	SubscriberMappingRule	77
5.4.5	Output Stylesheet	78
5.5	Konfiguration der Passwortsynchronisation	78

5.6	Sicherheit	79
6	Test	80
6.1	Testumgebung	80
6.2	Testergebnisse der Funktionstests	82
6.3	Testergebnisse der Passwortsynchronisation	83
7	Zusammenfassung	85
8	Ausblick	86
A	Eigenständigkeitserklärung	91

Abbildungsverzeichnis

2.1	Unidirektionale Synchronisation	25
2.2	Bidirektionale Synchronisation	25
2.3	Join Methode aus [TAHP03]	30
2.4	Virtual Directory aus [TAHP03]	31
3.1	JNDI Architektur [Sun03b]	41
3.2	Magic Quadrant for Metadirectory Products, 2H03 [Enc03]	49
4.1	Einrichtung von Nutzerkennungen im HRZ und FB1	56
4.2	Vereinfachung des Nutzermanagements mit Hilfe von DirXML	59
4.3	DirXML Architektur mit Remote Loader [Nov02]	60
4.4	DirXML Engine mit ihren Regelsätzen [Nov02]	62
4.5	DirXML Administrationsoberfläche iManager	64
5.1	Implementierung der DirXML Komponenten	73
5.2	DirXML Regeln in Publisher- und Subscriber-Kanal [Nov03d]	75
6.1	Aufbau der Testumgebung	80

Einleitung

Nach einer Studie von Forrester Research und der GartnerGroup benutzt das typische Fortune 1000 Unternehmen im Durchschnitt 181 Verzeichnisse [Zho99].

Heterogene Netzwerke mit verteilten Ressourcen befinden sich in jedem grösseren Unternehmen. Die Verwaltung von Ressourcen wird immer mehr über Verzeichnisdienste gesteuert. Dafür werden in vielen Unternehmen verschiedene Verzeichnisdienste verwendet. Dieser Umstand erzeugt einen enormen Kostenaufwand durch eine parallele Administration. Obwohl zumeist dieselben Identitätsinformationen in den verschiedenen Verzeichnisdiensten gespeichert werden, ist eine Migration der Daten in ein einziges Verzeichnis aufgrund der unterschiedlichen Anforderungen an die Verzeichnisdienste nur selten mit vertretbarem Aufwand möglich.

Eine Lösung des Problems stellt die Synchronisation von Daten zwischen unterschiedlichen Verzeichnissen dar. Sie kann helfen, unternehmensweite Nutzerdaten automatisch zu verteilen, abzugleichen und deren Administration zu vereinfachen. Mit Hilfe einer Synchronisation zwischen Verzeichnisdiensten kann die Informationsqualität gesteigert und der Weg zu einem einzigen unternehmensweiten Nutzerlogin geebnet werden.

Als weit verbreitete und in zahlreichen Unternehmen parallel eingesetzte Verzeichnisdienste stellt die Kombination der *Novell Directory Services* (NDS) und *Active Directory Services* ein repräsentatives Beispiel für Verzeichnisdienste mit einem Synchronisationsbedarf dar. Sie werden zumeist als Authentifizierungssysteme für IT-Netzwerke und ihre Ressourcen verwendet.

Diese Diplomarbeit beschäftigt sich mit dem Aufbau von Verzeichnisdiensten, sowie den Aspekten eines Abgleiches und stellt Lösungsansätze für eine Synchronisation der Verzeichnisdienste ADS und NDS vor. Im zweiten Teil wird an einem Beispiel eine Realisierung der Synchronisation von ADS und NDS vorgestellt. Dabei wird das Ziel einer „sanften“ Migration in eine gemeinsame Nutzer-Verwaltung am Beispiel der Anbindung eines Fachbereiches an das HRZ der FHTW Berlin erläutert.

1 Verzeichnisdienste

1.1 Evolution	7
1.2 Allgemeiner Aufbau von Verzeichnisdiensten	8
1.3 Arten von Verzeichnissen	10
1.4 Novell Directory Services (NDS)	11
1.4.1 Replikation der Verzeichnisdaten	12
1.4.2 Verzeichnisinterne Synchronisation	12
1.4.3 Partitionierung	12
1.4.4 Sicherheit	12
1.5 Active Directory Services (ADS)	13
1.5.1 Replikation und Partitionierung	14
1.5.2 Verzeichnisinterne Synchronisation	14
1.5.3 Sicherheit	14
1.6 Die Rolle von LDAP	15
1.6.1 Die Entstehung von LDAP	15
1.6.2 LDAP als Standalone-Server	16
1.6.3 Der LDAP Version 3 Standard	17
1.6.4 Vor- und Nachteile von LDAP	18
1.7 Zusammenfassung	18

Dieses Kapitel soll einen Überblick über Verzeichnisdienste geben. Die Erläuterung des Aufbaus von Verzeichnisdiensten ist nötig, um die Probleme einer Synchronisation von Verzeichnisdiensten zu verstehen. Weiterhin werden die in dieser Diplomarbeit besonders betrachteten Verzeichnisdienste ADS und NDS näher erläutert, um die wesentlichen Unterschiede zu erkennen.

1.1 Evolution

Die Kombination aus einem Verzeichnis, in dem Informationen abgespeichert werden, und einem Bereitstellungs- bzw. Verwaltungsdienst nennt man „Verzeichnisdienst“. Diese Form

der Informationsverwaltung ist schon seit Mitte der 80er Jahre bekannt und wurde zunächst nur für anwendungsabhängige Anforderungen konzipiert. Zu den ersten Verzeichnisdiensten gehörte der *Domain Name Service* (DNS). Er wird benutzt um Zuordnungen von DNS-Domännennamen zu IP-Adressen von Computern umzusetzen.

Bald darauf erkannte man den hohen Nutzen von Verzeichnisdiensten und es wurde Ende der 80er Jahre der erste Verzeichnisdienststandard mit dem Namen X.500 von der *International Telecommunications Union* (ITU) und der *International Organization for Standardization* (ISO) (früher als CCITT bekannt) entwickelt. Diese Standards ermöglichten die offene, anwendungsunabhängige Implementierung eines Verzeichnisses innerhalb eines Verzeichnisdienstes. Die technische Implementierung eines Verzeichnisdienstes wird von dem Standard jedoch nicht vorgegeben.

Die X.500 Standards verwenden verschiedene Modelle um die Verzeichnisstruktur und die Funktionalitäten eines Verzeichnisdienst aus verschiedenen Blickwinkeln zu beschreiben. Während die originalen 1988 verabschiedeten X.500 Spezifikationen nur das Directory Information Modell verwendeten, sind seit 1993 unter anderem folgende Modelle im X.501 Standard definiert.

- Directory Functional Model
- User Information Model
- Operational and Administrative Information Model
- DSA Information Model
- Directory Distribution Model
- Directory Administrative Authority Model
- Security Model

Die Architekturbasis der meisten aktuellen Verzeichnisdienste fundiert auf den X.500 Modellen. Auch für die Entwicklung von LDAP sind fast alle der X.500 Modelle übernommen worden, so dass jeder allgemeine Verzeichnisdienst direkt oder indirekt die X.500 Standards verwendet.

1.2 Allgemeiner Aufbau von Verzeichnisdiensten

Verzeichnisdienste bestehen aus einem Verzeichnis und Bereitstellungsdiensten. Die Struktur eines Verzeichnisses kann flach oder nach dem X.500 Modell hierarchisch organisiert sein und wird von einem zugehörigen *Schema* beschrieben. Das Schema des Verzeichnisbaumes

definiert alle existierenden *Objektklassen* eines Verzeichnisses. Das Schema definiert weiterhin die Einträge, die den, von den Objektklassen abgeleiteten, Objekten zugeordnet werden können und die Syntax für die Werte dieser Eigenschaften. Durch die Definition der Verzeichnisobjekte bestimmt das Schema die Struktur des Verzeichnisses. Das Verzeichnisschema kann demnach als die „Definition“ eines Verzeichnisses bezeichnet werden.

Die Schemainformationen beinhalten weiterhin Regeln, welche zur Bildung der detaillierten Objektdefinitionen verwendet werden. Diese Objektdefinitionen werden benutzt, um die Verzeichnishierarchie zu bilden und legen sowohl die Beziehung, als auch die Interaktionen zwischen den Objekten fest.

Ein *Namensraum* kann als Sammlung von Objekten bezeichnet werden, welche sich innerhalb eines *Containers* (ein Objekt, welches andere Objekte beinhalten kann) befinden und denselben Namenskonventionen folgen. Er ist ein abgegrenzter Bereich, in welchem standardisierte Namen zur symbolischen Darstellung von Verzeichnisobjekten verwendet werden können. In einem Namensraum wird an Hand von detaillierten Regeln festgelegt, wie Namen erstellt und verwendet werden.

Ein *Verzeichnisobjekt* ist eine Datenstruktur mit einer Menge von *Attributen*. Es repräsentiert einen Eintrag in einem Verzeichnis. Jedes Objekt wird von einer Objektklasse abgeleitet und erbt dessen spezifische Menge von Attributen (Einträge), von denen einige verbindlich, andere optional sein können. Jedes Attribut besitzt eine Syntax. Sie beschreibt den Typ der Informationen die in diesem Attribut gespeichert werden können. Die Attributwerte können verschiedene Informationen beinhalten, wie z.B. Nutzerdaten, Sicherheitsinformationen oder auch Bilder.

Ein Attribut des Objektes ist namensgebend und wird als *naming attribute* bezeichnet. Dieses Attribut stellt den eindeutigen Namen des Objektes – den *Relative Distinguished Name* (RDN) – dar. Er ist eindeutig in der jeweiligen Ebene der Hierarchie. Der *Distinguished Name* (DN) unterscheidet ein Objekt von allen anderen im gesamten Verzeichnisbaum und stellt den Namen des Objektes zusammen mit dem absoluten Pfad im Verzeichnis dar. Er ist der eindeutige Identifikationsschlüssel eines Objektes im Verzeichnisbaum.

Die Struktur der Informationen in einem Verzeichnis stellt sich als ein Baum dar. Dieser Baum wird als *Directory Information Tree* (DIT) oder auch Verzeichnisbaum bezeichnet. Der Verzeichnisbaum ist die hierarchische Anordnung von Verzeichnisobjekten in einem zusammenhängenden Namensraum. Containerobjekte formen die Äste des Baumes und fassen den Rest der Verzeichnisobjekte (Blattobjekte) zusammen. Die logische Struktur des Verzeichnisbaumes wird zur Darstellung und Administration der Objektsammlungen benutzt.

Alle Informationen eines Verzeichnisses werden in einer Datenbank – der *Directory In-*

formation Base (DIB) – gespeichert. Sie sind in der Regel auf Lesezugriffe und schnelle Suchen optimiert und unterstützen oft Indizes und Kataloge. Die Implementierungen der Speichertechniken sind sehr verschieden, sie sind nicht Bestandteil der X.500 Standards. Die Umsetzungen der Hersteller reichen von Textdateien (z.B. DNS) bis hin zu kommerziellen skalierbaren Datenbanken (z.B. Oracle Datenbank) als Speicher für Verzeichnisinformationen.

Die DIB kann in mehrere Datenbankteile aufgeteilt werden, welche *Partitionen* genannt werden. Das Teilen der DIB in Partitionen ist bei einigen Verzeichnisdiensten bis auf den Containerlevel möglich. Die *Partition root* oberste ist das Container Objekt, die Wurzel eines Verzeichnisses. Sie umfasst alle weiteren Unterbäume des Verzeichnisses solange kein anderes Container Objekt im Unterbaum als Wurzel für eine neue Partition eingerichtet wird. Verzeichnisdienste unterstützen meist Formen von Replikation um Redundanz und Fehlertoleranz zu erhöhen. Des Weiteren kann damit die Performance bei Suchenanfragen erhöht und das Datenvolumen bei der Replikation von Verzeichnisdaten gesteuert werden.

Ein Verzeichnisdienst kann auch bereitgestellten Dienste zur Unterstützung von Netzwerk-Clients, Administration, verzeichnisfähigen Anwendungen und Netzwerkdiensten umfassen. Allgemeine Verzeichnisdienste beinhalten Nutzer- und Gruppenmanagementfunktionen sowie robuste Sicherheitsdienste und unterstützen die Erweiterung ihres Schemas für anwendungsspezifische Anforderungen.

1.3 Arten von Verzeichnissen

Im Zuge der Entwicklung von Verzeichnisdiensten sind verschiedene Anwendungsformen entstanden. Sie lassen sich je nach Art der enthaltenen Objektklassen und den Managementfunktionalitäten (unterstützte Dienste) einteilen.

- **Netzwerkspezifische Verzeichnisdienste** Sie sind ausgelegt, die *Network Operation Systems* (NOS) Funktionen – wie Nutzerkonten, Sicherheit und Management der Netzwerkressourcen – verzeichnisorientiert anzubieten. In fast jedem NOS ist ein Verzeichnis enthalten, welches im Design teilweise an die X.500 Standards angelehnt ist. Beispiele für netzwerkspezifische Verzeichnisdienste sind der *Network Information Service* (NIS) und die *Active Directory Services* (ADS)¹ von Microsoft.
- **Allgemeine Verzeichnisdienste** bieten die größte Auswahl an Verzeichnisspezifischen Diensten und Funktionen. Sie erfüllen sowohl die Anforderungen großer Unternehmensdatenbestände als auch die von Netzwerkmanagementsystemen. Allgemein Verzeichnisse basieren gewöhnlich auf den X.500 Standards und unterstützen Stan-

¹Einordnung aufgrund seiner Plattformabhängigkeit (nur Windows 200x)

standardprotokolle und Dienste, wie z.B. LDAP und DNS. Ein Beispiel für einen allgemeinen Verzeichnisdienst sind die *Novell Directory Services* (NDS).

- **Metaverzeichnisdienste** besitzen Funktionen zum Managen und Integrieren von Informationen, welche in verschiedenen Verzeichnissen gespeichert sind. Um die Daten aus den verschiedenen netzwerkspezifischen und applikationsspezifischen Verzeichnissen abzurufen, besitzen Metaverzeichnisse Konnektoren, welche die Informationen nach unterschiedlichen Kriterien einsortieren können. Metaverzeichnisse werden im dritten Kapitel dieser Diplomarbeit näher erläutert.
- **Applikationsspezifische Verzeichnisse** speichern z.B. applikationsspezifische Nutzerdaten, Operationen die den Nutzern erlaubt sind und einige Konfigurationsdaten. Sie werden z.B. für Email- und Groupwareanwendungen benutzt. Ein Beispiel für einen applikationsspezifischen Verzeichnisdienst stellt der DNS dar.

1.4 Novell Directory Services (NDS)

Nach einem CEO von Novell sind die NDS „... a general-purpose directory service that manages discovery, security, storage, and relationships“. Novell hat viel Erfahrung auf dem Gebiet von Verzeichnisdiensten und bietet seine X.500-konforme NDS schon seit 1993 an. Während sich das Unternehmen früher eher auf die Verwaltung von Windows- oder DOS-Clients spezialisierte, nutzt es heute seinen Wissensvorsprung auf dem Gebiet der Verzeichnisdienste und hat das eigentliche Verzeichnis mit all seinen Funktionalitäten in den Mittelpunkt seiner Verkaufspolitik gestellt. Seit 1999 die Version 8.5 der NDS erschien, werden sie unter dem neuen Marketingnamen *eDirectory* vertrieben. Inzwischen ist auch die Portierung auf andere Systeme weiter fortgeschritten, so dass die eDirectory heute sowohl auf dem von Novell entwickelten Netzwerkbetriebssystem Netware als auch auf Windows NT/200x, Solaris, Linux und anderen Unixderivaten als Verzeichnisdienst benutzt werden kann.

Mit einer Nutzerbasis von über 140 Millionen Personen kann Novell einen großen Marktanteil bei Verzeichnisdiensten vorweisen. In vielen Unternehmen, so auch an der FHTW Berlin, werden die NDS vor allem für die Verwaltung von Windows-Clients verwendet. Auch von vielen Internetfirmen, z.B. Yahoo, werden die NDS eingesetzt, um die Nutzeraccounts ihrer Internetangebote zu verwalten.

Novell strebt in letzter Zeit verstärkt in die Richtung der offenen Betriebssysteme. So wurde am Ende des vergangenen Jahres unter anderem der deutsche Linuxdistributor Suse gekauft. Dabei verfolgt Novell das Ziel, die Verwaltung von Unix-basierenden Clients zu vereinfachen und die Weiterentwicklung der Unixvariante seines Verzeichnisdienstes voranzutreiben.

1.4.1 Replikation der Verzeichnisdaten

Die NDS unterstützen eine einfach zu administrierende Multi-Master Replikation. Die Platzierung und das Management sind sehr anpassungsfähig: so erlaubt die NDS-DIB 50 Repliken pro Partition und 250 Repliken pro Server. Ein automatischer Replikationsprozess sichert die Performance und Fehlertoleranz des Verzeichnisdienstes. Obwohl die NDS im Standardfall die Multi-Master Replikation benutzen, ist auch die Konfiguration hin zu einer Single-Master Umgebung mit Read-only Repliken möglich.

Die Replikationsgranularität kann über Filter gesteuert und verfeinert werden, um z.B. das Datenaufkommen und somit den Netzwerkverkehr für entfernte Replikationspartner weiter zu senken. Replikationsfilter können auf Objekt- und Attributebene getrennt für aus- oder eingehenden Replikationsverkehr erstellt werden.

1.4.2 Verzeichnisinterne Synchronisation

Für den Abgleich seiner Repliken nutzt die NDS-DIB eine transitive Synchronisation. Die Server können so in Replikationsringen organisiert und als eine „Kette“ synchronisiert werden. Diese kaskadierte Synchronisation verringert den Netzwerkverkehr zwischen den Repliken und ermöglicht einen Abgleich über verschiedene Netzwerkprotokolle (Multiprotokollrouting).

Die verzeichnisinterne Synchronisation der NDS ist priorisiert. Wichtige Updates, wie Sicherheitseinstellungen oder Passwörter, werden sofort an alle Repliken gesendet (FAST Synchronisation). Die größte Anzahl der Verzeichnisänderungen wird jedoch in größeren Zeitabständen, meist mit einer festen Frequenz, an die jeweiligen Synchronisationspartner verteilt.

1.4.3 Partitionierung

Bei der Partitionierung der Verzeichnisdatenbank sind kaum Grenzen gesetzt. Ein einzelnes Containerobjekt kann mehr als eine Milliarde Blattobjekte beinhalten. Die Referenzen zwischen den Teilen des Verzeichnisses, welche für die Inter-Server Operationen benötigt werden, werden automatisch bei der Partitionierung angelegt. Die Verzeichnisinformationen werden effizient gespeichert, so dass eine Partition mit einer Milliarde Objekten im Durchschnitt nur etwas mehr als ein Gigabyte an Speicherplatz belegt.

1.4.4 Sicherheit

Die Novell Directory Services unterstützen eine Anzahl von Nutzerbeglaubigungsmethoden. Dazu gehören SSL²-verschlüsselte Passwörter, X.509 Zertifikate oder Smart Cards. Novell's

²Secure Sockets Layer

Certificate Server, eine frei erhältliche Erweiterung der NDS, unterstützt und verwaltet PKI Zertifikate. Der Verzeichnisdienst bietet Zugangskontrollen und unterstützt die Delegation administrativer Rechte bis auf Attributebene. Weiterhin benutzt er ein System zum direkten Übertragen von Rechten, um die Zugangserlaubnis für jedes Objekt auslösen zu können.

1.5 Active Directory Services (ADS)

Microsoft brachte mit dem Betriebssystem *Windows 2000 Server* die erste Version eines X.500-konformen Verzeichnisdienstes mit dem Namen *Active Directory Services* auf den Markt. Im Gegensatz zur NDS sind Microsofts ADS derzeit lediglich für Windows-Systeme verfügbar. Sie sind ein netzwerkspezifischer Verzeichnisdienst und für die Verwaltung von verteilten Windows Netzwerken bestimmt.

Das Active Directory (AD) ist eine Mischung aus dem Windows NT Domänenmodell, dem DNS und dem LDAP-Protokoll mit vielen proprietären Abänderungen. Das Domänenmodell wird aus Gründen der Abwärtskompatibilität übernommen und bildet die Basis des ADS-Designs. Das X.500 Datenmodell wird vom Active Directory zum Strukturieren des Verzeichnisinhaltes, das LDAP Protokoll für den Zugriff auf Verzeichnisinformationen und DNS als Namensraummodell benutzt.

Microsoft erweitert die Funktionen des DNS, um damit auch die von Domänencontrollern angebotenen Dienste im Netzwerk verfügbar zu machen. Der Microsoft DNS-Dienst ist ein elementarer Bestandteil um die Funktionalitäten einer Domäne vollständig nutzen zu können.

Neben der Identifikation von Computernamen über DNS werden AD-Domänencontroller über spezielle bereitgestellte Dienste identifiziert, beispielsweise LDAP-Server [...] Domänencontroller und globale Katalogserver. Daher kann über einen DNS-Server mit Hilfe des Domänennamens und einer Dienstspezifikation ein Domänencontroller dieses Typs innerhalb der Domäne aufgefunden werden. [LB00, S.5]

Die ADS werden mit Hilfe von *Domänen* strukturiert. Die Merkmale einer Domäne im Kontext von ADS sind gemeinsame Sicherheits- und Administrationsdefinitionen für das jeweilige Windows-Netzwerk [LB00, S.3]. Die Grundstruktur einer ADS-Domäne ist, im Gegensatz zu den X.500 Standards, fest vorgegeben und nicht als Baum aufgebaut. Dies hat vor allen Dingen historische Gründe. Microsoft begründet es mit einer besseren Nachvollziehbarkeit des Wachstums von Windows Netzwerken [MR99, S.122]. Mehrere Domänen können zu einem Domänenbaum und mehrere Bäume zu einem *forest* zusammengefasst werden. Ein forest stellt die grundlegende Wurzel eines Active Directory Namensraum dar und ist in der Struktur oberhalb aller AD-Verzeichnisbäume angesiedelt. Diese „Wälder“ sind in

der Lage, ein allgemeingültiges Schema und Konfigurationsinformationen zu verteilen. Mit Hilfe von Vertrauensbeziehungen kann ein Datenaustausch zwischen den angeschlossenen Domänen realisiert werden.

Microsoft hat den heutigen Stellenwert von Verzeichnisdiensten erkannt und versucht durch stetige Weiterentwicklung der ADS seinen Marktanteil weiter auszubauen. Das wird durch die enorme Verbreitung von Windows-basierten Rechnern in vielen Unternehmen begünstigt.

1.5.1 Replikation und Partitionierung

Die ADS benutzen eine Multi-Master Replikation. Die Partitionierung des Verzeichnisses ist festgelegt – Partitionen werden automatisch auf Domänenebene angelegt und können auf keiner anderen Ebene erstellt werden. Ein Domänencontroller (Verzeichnisserver einer Domäne) kann und muss nur eine Replik der Domäne halten. Es existieren zwei unterschiedliche Replikatypen, wovon der eine alle Daten einer Partition (Full Replika) der andere Typ nur einen Teil der Partitionsdaten aller Domänenpartitionen (Partial Replika) enthält.

Die Active Directory benutzt ein „store-and-forward“ Mechanismus für die Replikation seiner Daten. Nachdem Objekte geändert wurden, werden die anderen Domänencontroller in der Domäne informiert, dass Änderungen durchgeführt wurden. Nach dieser Mitteilung nehmen die Zieldomänencontroller Kontakt zum Quelldomänencontroller auf und bekommen die geänderten Verzeichnisdaten.

1.5.2 Verzeichnisinterne Synchronisation

Für die Synchronisation der Repliken werden *Update Sequence Numbers* (USN) verwendet. Vor der Replizierung von Änderungen werden zwischen den Servern die USN³-Zähler verglichen und danach entschieden, ob und wie viele Informationen synchronisiert werden müssen. Ausserdem werden Zeitstempel benutzt, um Änderungskonflikte aufzulösen.

Die Synchronisation kann mit Hilfe von *sites* weiter angepasst werden. Sites definieren eine Menge von TCP/IP Subnetzen mit verschiedenen Konnektivitäten (z.B. schnelle LAN- oder langsame WAN-Verbindungen). Dieses sites-Objekte können zum Anpassen des Replikationsverkehrs benutzt werden.

1.5.3 Sicherheit

Die ADS unterstützen SSL, SASL⁴, X.509 Zertifikatmanagement und nutzen Kerberos für die Beglaubigung. Sicherheits- und Beglaubigungsregeln sowie Zugangskontrollen werden

³Das Prinzip der USN wird im zweiten Kapitel näher erläutert.

⁴Simple Authentication and Security Layer

auf der Domänenebene definiert und dort verwaltet. Zwischen verschiedenen Domänen können Vertrauensbeziehungen hergestellt werden. Für die Zugriffskontrollen und den Schutz der Verzeichnisobjekte werden *Access Control Lists* (ACL) verwendet.

1.6 Die Rolle von LDAP

Dieses Kapitel soll einen Überblick über das *Lightweight Directory Access Protocol* (LDAP) geben. LDAP ist sowohl als Protokoll, als auch als Server, aus dem heutigen Verzeichnisdienstumfeld kaum noch wegzudenken. Fast alle Verzeichnisdienste, so auch die ADS und NDS, unterstützen das „leichtgewichtige“ Zugriffsprotokoll und bieten somit eine standardisierte Schnittstelle für den Zugriff auf Verzeichnisinhalte an. Was liegt also näher, als LDAP für die Synchronisation zu nutzen?

1.6.1 Die Entstehung von LDAP

Als 1988 der erste Verzeichnisdienststandard mit dem Namen X.500 von der ISO⁵ und der ITU⁶ verabschiedet wurde, war zum Zugriff auf diesen Verzeichnisdienst ein *Directory Access Protocol* (DAP)⁷ vorgesehen. Als Protokoll des obersten Layers (Application Layer) im OSI-Referenzmodell⁸ ist DAP auf die kompletten darunterliegenden OSI-Layer angewiesen. Da in vielen Umgebungen aber nicht alle Schichten des OSI Modells verfügbar sind oder nur mit hohem Aufwand implementiert werden können, wurden wenige Jahre später einfachere Protokolle entwickelt. Parallel zu DAP entstanden so zwei Protokolle mit den Namen *Directory Assistance Service* (DAS) (beschrieben in RFC 1202 [Ros91]) und *Directory Interface to X.500 Implemented Efficiently* (DIXIE) (beschrieben in RFC 1249 [HSB91]). Beide Protokolle vereinfachten im Zusammenspiel den Zugriff auf Verzeichnisdienste des X.500 Standards.

After DIXIE and DAS showed the utility of a lighter-weight access protocol for X.500, the members of the OSI-DS Working Group within the IETF decided to join forces and produce a full-featured, lightweight directory access protocol for X.500 directories. Thus, LDAP was born. [TAHP03]

LDAP bringt im Gegensatz zum komplexeren X.500 DAP Protokoll eine Reihe von Vorteilen mit sich. Dazu gehören z.B. die gesteigerte Funktionalität und eine einfachere Datenrepräsentation (simple Textstrings). Vor allem aber läuft LDAP direkt über TCP und ist

⁵International Standard Organization

⁶International Telecommunications Union

⁷X.519 - Protocol Specifications

⁸reference model for open systems interconnection

somit nicht mehr von den vielen Schichten des OSI-Referenzmodells abhängig. Damit wird die Performance erhöht und die Implementierung ist um vieles einfacher.

Zunächst wurde LDAP nur zur Bereitstellung eines Zugriffs auf X.500 basierende Verzeichnisdienste verwendet. Die erste LDAP Implementation wurden an der Universität von Michigan entwickelt. Sie war klein, schnell und konnte auf vielen Plattformen benutzt werden. Mit Hilfe eines „ldapd“-Deamon, welcher als serverseitiger LDAP zu X.500 DAP Übersetzer agiert, war nun ein einfacher Zugriff auf die X.500 Verzeichnisse möglich.

1.6.2 LDAP als Standalone-Server

Die Vereinfachung des Verzeichniszugriffsprotokolles in Form von LDAP zog eine Veränderung des Übertragungsaufkommens nach sich. Der größte Teil der Zugriffe wurde nun über LDAP abgewickelt. Die Verzeichnisse blieben aber auf Basis von X.500 bestehen. Der serverseitige LDAP-Dienst (LDAP Proxie) übersetzte die Anfragen weiterhin in das X.500 konforme DAP Protokoll.

Um die Nachteile der durch das Kommunikationsprotokoll in Frage gestellten X.500 Verzeichnisse zu umgehen, wurde ein nativer LDAP Server mit dem Namen „slapd“ (standalone LDAP server) entwickelt. Das Basis-Schema wurde fast unverändert von X.500 übernommen.

Die erste Implementation, die den slapd-Server enthielt, wurde im Dezember 1995 von der Universität von Michigan freigegeben. Einer der bekanntesten Vertreter der komplett LDAP-basierenden Verzeichnisdienste ist heute die OpenLDAP Server Suite der OpenLDAP Foundation⁹.

LDAP ist also nicht nur als Kommunikationsprotokoll verfügbar, nach Verabschiedung des neueren LDAPv3 Standards existieren auch native LDAP Verzeichnisdienste. Sie sind nicht mehr auf die restriktiven X.500-Standardverzeichnisse angewiesen, sie können alleinstehend auf Basis der LDAP-RFCs und den X.500 basierenden LDAP Teilen betrieben werden. Heutzutage wird die Bezeichnung „LDAP“ mehr mit den Servern in Verbindung gebracht als mit dem eigentlichen Kommunikationsprotokoll.

Im Unterschied zu LDAP-Servern haben viele Unternehmen, wie z.B. Netscape, Sun Microsystems, Novell und Microsoft *LDAP-fähige* Verzeichnisdienste entwickelt. Sie basieren teilweise auf den älteren X.500 Standards sowie den LDAP Vorgaben, sind aber dennoch mit eigenen proprietären Verzeichnislösungen ausgestattet. Auch viele bedeutenden Applikationssoftwarehersteller unterstützen LDAP in ihren Client- und Serverprodukten.

⁹<http://www.openldap.org/>

1.6.3 Der LDAP Version 3 Standard

Der aktuelle LDAPv3-Standard wurde im Dezember 1997 von der IETF als 'Proposed Internet Standard' bestätigt. Die Verbesserungen im Gegensatz zum LDAPv2-Standard bestehen vor allem in folgenden Punkten (vergleiche [TAHP03]):

- **Internationalisierung**

Durch die Benutzung des universellen UTF-8 (UCS Transformation Format 8) Zeichensatzes, können alle mögliche Zeichen in fast jeder Sprache verwendet werden.

- **Referenzen**

Über einen Verweismechanismus sind die Server nun in der Lage, Anfragen mit Referenzierungen auf andere Server zu beantworten. Damit ist es möglich, einen Verzeichnisbaum zu partitionieren und auf mehrere Server aufzuteilen.

- **Sicherheit**

Die Unterstützung für *Simple Authentication and Security Layer* (SASL) und *Transport Layer Security* (TLS) wurde implementiert, um die Sicherheit von LDAP zu erhöhen.

- **Erweiterbarkeit**

LDAPv3 kann durch „Extensions“ und „Controls“ erweitert werden, um neue Funktionen zu unterstützen oder bestehende zu ergänzen.

- **Fähigkeiten und Schemaausgabe**

Alle LDAPv3-Server sind in der Lage, ihre unterstützten Protokolle, ihre Schemata sowie andere wichtige Informationen dem Client bekanntzugeben. Dadurch sind andere Server und Clients in der Lage, die LDAP Dienste besser und gezielter zu nutzen.

Eine Übersicht über die LDAPv3 Spezifikationen ist im RFC 3377 [HM02] enthalten. Momentan beschäftigt sich eine IETF Arbeitsgruppe mit Namen *ldapbis*¹⁰ mit der Weiterentwicklung dieser Standards.

Eine weitere IETF Arbeitsgruppe mit dem Titel *LDAP Duplication/Replication/Update Protocols* (LDUP) beschäftigt sich mit der Replikation zwischen LDAP Servern.

As LDAPv3 becomes more widely deployed, replication of data across servers running different implementations becomes an important part of providing a distributed directory service. However, the LDAPv3 community, to date, has

¹⁰<http://www.ietf.org/html.charters/ldapbis-charter.html>

focused on standardizing the client-server access protocol. This group was originally chartered to standardize master-slave and multi-master LDAPv3 replication [...]. [CA03]

Als einziges RFC Dokument dieser LDUP Arbeitsgruppe ist zur Zeit RFC 3384 [SWMH00] verfügbar. Es beschreibt die Anforderungen an eine Replikation zwischen LDAP Servern. In Zukunft sollte es möglich sein, mit Hilfe eines standardisierten Replikationsprotokolles Daten zwischen verschiedenen LDAP Servern zu replizieren.

1.6.4 Vor- und Nachteile von LDAP

Zusammenfassend lassen sich drei Einsatzarten von LDAP unterscheiden:

- LDAP als einfaches Zugriffsprotokoll für X.500 Verzeichnisse, realisiert über einen Protokollumsetzer auf dem Verzeichnisserver.
- LDAP als vollwertiger Verzeichnisdienst (Standalone-Server).
- LDAP als Schnittstelle für LDAP-fähige Verzeichnisdienste.

Viel Arbeit ist in die Entwicklung von LDAP geflossen und die Verbreitung von nativen LDAP-Servern schreitet dank des offenen Standard weiter voran. Es gibt viele Vorteile, die für die Benutzung von LDAP sprechen. Eine große Anzahl LDAP-fähiger Verzeichnisse und LDAP-verzeichnisfähiger Anwendungen existieren bereits. Eigene Verzeichnisanwendungen, die LDAP als Kommunikationsprotokoll verwenden, können mit vielen LDAP-kompatiblen Verzeichnissen benutzt werden. LDAP-Verzeichnisdienste sind preiswert in der Anschaffung, da frei verfügbar, und relativ einfach zu verstehen. Sie besitzen eine hohe Leistung, eine hohe Zuverlässigkeit und als Mehrzweckverzeichnisdienst sind sie für den Einsatz auf unterschiedlichen Gebieten geeignet.

Nachteile sind vor allem die nicht vorhandenen Replikationsstandards. Der OpenLDAP *stand-alone LDAP replication server* (slurpd) ist als eine einfache Lösung zur Umsetzung von Replikation zwischen LDAP Servern entwickelt worden. Er ist auf eine Änderungsdatei angewiesen, dessen Inhalt er an die Replikationspartner überträgt. Die Nachteile dieser Methode werden im Kapitel 2 erläutert.

1.7 Zusammenfassung

Verzeichnisdienste werden zum Speichern verschiedener Informationen genutzt. Ihre Datenspeicher sind leseoptimiert und können für einen Lastausgleich partitioniert werden. Aus Gründen der Redundanz und Fehlertoleranz können Verzeichnisdienste Kopien ihrer

Partitionen – Repliken – anlegen. Diese Repliken werden mit Hilfe von verschiedenen Synchronisationstechniken abgeglichen. Verzeichnisdienste sind verteilte Systeme und arbeiten nach dem Client-Server Prinzip.

Obwohl die meisten Verzeichnisdienste vom X.500 Standard abstammen, unterscheiden sich Aufbau und Mechanismen des internen Informationsaustausches. Eine Replikation von Daten zwischen verschiedenen Verzeichnisdiensten ist aufgrund der unterschiedlichen Replikationsmechanismen nicht möglich. Für eine Synchronisation von verschiedenen Verzeichnisdiensten können demnach keine internen Synchronisationsdienste benutzt werden.

Die Novell Directory Services und die Active Directory Services unterscheiden sich in ihren Replikations- und Partitionierungsmechanismen stark voneinander. Die NDS sind ein allgemeingültiger, die ADS ein netzwerkspezifischer Verzeichnisdienst. Während die NDS für verschiedene Anwendungen benutzt werden können, sind die ADS vor allem für die Verwaltung von Nutzerdaten und Windowsrechnern konzipiert.

Die Informationen der beiden verglichenen Verzeichnisdienste werden über unterschiedliche Verzeichnisschemata verwaltet und sind in verschiedenen Namensraumformaten organisiert. Die Erweiterung der Verzeichnisschemata ist in beiden Verzeichnisdiensten möglich.

Beide Verzeichnisdienste (NDS und ADS) stellen eine LDAPv3-Schnittstelle bereit. Es existieren jedoch momentan keine für eine Replikation über die LDAP-Schnittstelle benötigten Replikationsmechanismen. Eine IETF Arbeitsgruppe mit dem Namen LDUP hat sich mit den Anforderungen einer Replikation zwischen LDAP-Servern beschäftigt und sich zum Ziel gesetzt, in den nächsten Jahren einen Standard für einen Replikationsmechanismus zu entwickeln.

2 Aspekte für den Abgleich von Verzeichnisdiensten

2.1	Grundlegende Anforderungen an einen Abgleichmechanismus	21
2.1.1	Heterogenität	21
2.1.2	Offenheit	21
2.1.3	Sicherheitsanforderungen	22
2.1.4	Skalierbarkeit	22
2.1.5	Fehlerverarbeitung	23
2.2	Synchronisationsmethoden zur Verteilung von Daten	24
2.2.1	Unidirektionale Synchronisation	24
2.2.2	Bidirektionale Synchronisation	25
2.3	Methoden zur Auslösung von Synchronisationsvorgängen	26
2.4	Zeitsynchronisation zur Ordnung von Änderungen	26
2.5	Granularität der Änderungserkennung	28
2.6	Methoden zum Zusammenfassen von Verzeichnisinformationen	29
2.6.1	Join – Zusammenfassung von Informationen	29
2.6.2	Virtuelles Verzeichnis – Zentrale Verlinkung von Daten	30
2.7	Funktionalitäten für einen Informationsabgleich	31
2.8	Passwortsynchronisation	33
2.9	Performanceanforderungen an Abgleichmechnismen	34
2.10	Administration der Synchronisation	34
2.11	Anforderungsprofil für eine Lösungsfindung	35

Ziel eines Abgleiches von Verzeichnisdiensten ist die Schaffung eines konsistenten Informationsstandes zwischen den unterschiedlichen Verzeichnissen.

Da ein Abgleich von Verzeichnisdiensten, aufgrund des verschiedenartigen Aufbaus und der Nutzung unterschiedlicher interner Abgleichstechniken, mit verzeichnisdiensteigenen Mitteln nicht möglich ist, muss ein externer Synchronisationsmechanismus verwendet werden.

In diesem Kapitel werden Aspekte für einen Aufbau eines Synchronisationsmechanismus zwischen Verzeichnisdiensten erläutert.

Ein Synchronisationssystem zum Abgleichen von verschiedenartigen Verzeichnisdiensten stellt ein verteiltes System dar. Daraus lassen sich zunächst grundlegende Anforderungen an eine Synchronisationslösung ableiten.

2.1 Grundlegende Anforderungen an einen Abgleichmechanismus

Ein Abgleichmechanismus ermöglicht eine Kooperation zwischen den Verzeichnisdiensten. Als Kooperation versteht man den Austausch von Daten (Kommunikation), die Steuerung des zeitlichen Ablaufs der Interaktion und das ordnungsgemäße Zusammenspiel aller Beteiligten untereinander (Koordination) [Urb03]. Um eine Kooperation herzustellen, muss ein Abgleichmechanismus Heterogenität, Standardschnittstellen unterstützen. Außerdem muss er skalierbar sein, eine robuste Fehlerverarbeitung aufweisen und Sicherheitsmechanismen zum Schutz der Verzeichnisdaten und der Datenübertragung bereitstellen.

2.1.1 Heterogenität

Die Unterstützung von Vielfalt und Verschiedenartigkeit ist eine grundlegende Anforderung an einen Abgleichmechanismus. Während Verzeichnisdienste, ebenfalls verteilte Systeme, auch homogen aufgebaut sein können, muss ein Abgleichmechanismus, welcher als Vermittler zwischen verschiedenartigen Systemen auftritt, deren Heterogenität unterstützen. Er sollte mehrere Datenübertragungsprotokolle unterstützen, um Informationen zwischen den verschiedenen Datenquellen abgleichen zu können.

2.1.2 Offenheit

„Die Offenheit eines Computersystems ist die Eigenschaft, die festlegt, ob das System auf unterschiedliche Weisen erweiterbar ist und neu implementiert werden kann.“ [Cou01, S.35] Für die Verzeichnissynchronisation bedeutet Offenheit die Unterstützung weiterer Clients (Verzeichnisdienste). Eine Grundvoraussetzung dafür ist die Benutzung veröffentlichter Schnittstellen.

Ein Beispiel für solche Veröffentlichungen sind die „Request for Comment“ (RFC) Dokumente. Die Spezifikationen verschiedener Kommunikationsprotokolle wurden in dieser Serie veröffentlicht. Ein Beispiel ist das offene Verzeichnisdienstprotokoll LDAP, welches in vielen RFC-Dokumenten beschrieben wird (vergleiche Kapitel 1).

Erweiterbare verteilte Systeme werden auch *offene verteilte Systeme* genannt. Sie verfolgen zusammenfassend folgende Ziele:

- Verwendung veröffentlichter Schlüsselschnittstellen.
- Verwendung einheitlicher Kommunikationsmechanismen und veröffentlichte Schnittstellen für den Zugriff.
- Einsatz heterogener Hard- und Software, welche auch von unterschiedlichen Herstellern stammen kann.

2.1.3 Sicherheitsanforderungen

Verzeichnisdienste können viele sensible Informationen enthalten. Diese können innerhalb der Verzeichnisdienste durch Beglaubigung und Zugriffslisten ausreichend geschützt werden. Ein Abgleich von Informationen zwischen Verzeichnissen beinhaltet sowohl den Zugriff auf die Daten, als auch eine Übertragung der Informationen über ein Kommunikationsprotokoll. Die Sicherheitsanforderungen an einen Verzeichnisdienst müssen deshalb auch auf die Abgleichtechniken übertragen werden.

Probleme können durch die unterschiedlichen Sicherheitsanforderungen der Quellsysteme auftreten. So ist es bei einem Abgleich eines „sicheren“ und eines weniger „sicheren“ Systems denkbar, über einen Zugriff durch das „unsichere“ System mit Hilfe des Abgleichmechanismus in den Besitz sensibler Daten des vermeintlich „sicheren“ System zu gelangen. Ein Abgleich muss die unterschiedlichen Sicherheitsregeln der Quellsysteme berücksichtigen und die Möglichkeit des Unterlaufens dieser Regeln unterbinden.

Allgemein müssen folgende Aspekte berücksichtigt werden:

- **Vertraulichkeit:** Schutz gegen Offenlegung gegenüber nicht berechtigten Personen.
- **Integrität:** Schutz gegen Veränderung oder Beschädigung
- **Verfügbarkeit:** Schutz gegen Störungen der Methoden zum Ressourcenzugriff

Sicherheit für einen Abgleich kann erreicht werden, indem der Transport der Daten verschlüsselt wird und die Einträge der Verzeichnisse durch Zugriffslisten geschützt werden. Der gesicherte Zugriff eines Abgleichprozesses auf Verzeichnisinformationen kann über eine Beglaubigung des Prozesses an den Verzeichnissen und damit verbundenen sorgfältig ausgewählten Zugriffsrechten erreicht werden.

2.1.4 Skalierbarkeit

Die Skalierbarkeit drückt die Erweiterbarkeit eines Systems aus. Ein skalierbares Synchronisationssystem muss auch dann effektiv arbeiten, wenn die Anzahl der Ressourcen und Nutzer und somit das Datenaufkommen wesentlich steigt. Dafür muss ein Abgleichssystem erweiterbar sein. Die Erweiterung muss zu vernünftigen Kosten realisiert werden können.

Um Leistungsengpässe zu vermeiden, sollten Algorithmen von skalierbaren Synchronisationssystemen eine Dezentralisierbarkeit unterstützen. Die Prozesslast kann dadurch auf verschiedene Rechner verteilt werden. Eine Verteilung der Prozesslast bringt allerdings unter Umständen auch einen erhöhten Synchronisationsbedarf mit sich.

Eine Synchronisationstechnik für Verzeichnisdienste arbeitet eng mit den Verzeichnisdiensten zusammen und hängt somit auch teilweise direkt von ihrer Leistung ab.

Für eine Implementation können die bestehenden Skalierungsmechanismen der zu synchronisierenden Verzeichnisdienste genutzt werden. Wie im ersten Kapitel erläutert, verteilen Verzeichnisdienste Kopien ihrer Daten mit Hilfe von Repliken auf verschiedenen Verzeichnisdienstservern. Dem Abgleichmechanismus wird somit die Möglichkeit gegeben, parallel über verschiedene Server auf die Daten der Verzeichnisdienste zuzugreifen und somit skalierbar zu sein.

2.1.5 Fehlerverarbeitung

Ein elementarer Bestandteil zur Sicherung der Konsistenz der zu synchronisierenden Informationen ist eine Fehlerverarbeitung. Fehler können in den Verzeichnisdiensten oder in dem Synchronisationsmechanismus auftreten, die Folge sind verfälschte Informationen oder der komplette Abbruch der Informationsverarbeitung. Die Aspekte der Fehlerverarbeitung müssen einerseits die beteiligten Prozesse, andererseits die beteiligten Kommunikationskanäle berücksichtigen. Zu den Aspekten gehören:

- **Erkennung von Fehlern**

Als Fehlerquellen können sowohl die angeschlossenen Verzeichnisdienste, die Kommunikationskanäle als auch der Abgleichmechanismus selbst auftreten. Durch geeignete Methoden können einige Fehler erkannt werden. Teilweise sind diese Methoden in den Kommunikationsprotokollen bereits implementiert.

Verzeichnisdienste besitzen ein eigenes Fehlerverarbeitungssystem. Eine Abgleichmechanismus muss in der Lage sein, die durch die Verzeichnisdienste erkannten Fehler zu berücksichtigen.

Da ein Abgleichmechanismus Daten in den angeschlossenen Verzeichnissen verändert, kann er demnach die Konsistenz der Verzeichnisdienste gefährden. Durch den Abgleich können neue Fehler entstehen, welche durch die Veränderung der Informationen innerhalb des Abgleichmechanismus verursacht werden können. Diese Fehler müssen ebenfalls erkannt werden.

- **Tolerierung von Fehlern**

Es gibt Fehler, welche auch nach einer Erkennung nicht beseitigt werden können. Um einer „Verklebung“ durch zahlreiche Wiederholungen entgegenzuwirken, können

Fehler toleriert werden. Jedoch darf durch eine Tolerierung von Fehlern keine Fehlerfortpflanzung entstehen. Es bedarf demnach einer sensiblen Betrachtung der aufgetretenen Fehler.

Wenn beispielsweise ein Kommunikationspartner bei einer Synchronisation nicht erreichbar ist, kann dieser Zustand toleriert werden und der Synchronisationsprozess wird beendet. Üblicherweise erfolgt danach eine Ausgabe des Fehlers, um dem Benutzer eine Fehlerbeseitigung zu ermöglichen.

- **Regeneration nach Fehlern**

Die Konsistenz der Daten muss auch nach schwerwiegenden Fehlern gewährleistet sein. Dafür muss ein abgeglicherer Zustand zwischen den Verzeichnisdiensten auch nach einem Ausfall realisiert werden können. Um eine Regeneration zu ermöglichen, werden oft Änderungslogdateien verwendet. Im einfachsten Falle werden komplette Datenbestände abgeglichen und somit ein konsistenter Zustand wiederhergestellt.

- **Redundanz (Replikation)**

Um Dienste fehlertoleranter zu gestalten, werden redundante Komponenten verwendet. Bei einem Abgleich von Verzeichnisdiensten ist eine Redundanz meist auf der Seite der beteiligten Verzeichnisdienste gegeben. Sie replizieren ihre Daten innerhalb der eigenen Umgebung, um so Fehlern vorzubeugen. Der eigentliche Abgleichprozess kann aufgrund der verzeichnisinternen Datenreplikation auf mehreren Verzeichnisservern verteilt werden, um somit eine zusätzliche Redundanz zu erhalten.

2.2 Synchronisationsmethoden zur Verteilung von Daten

Synchronisation ist einfach gesagt das Kopieren geeigneter Daten von einer Datenquelle zu einem Ziel. Es gibt zwei Verteilungsmethoden, um Informationen zwischen Datenquellen abzugleichen.

2.2.1 Unidirektionale Synchronisation

Bei dieser Methode wird ein Ziel-Verzeichnisdienst mit Informationen einer Datenquelle bestückt. Änderungen an den abzugleichenden Einträgen im Verzeichnisbaum des Ziels sind nur dem Synchronisationsprozess erlaubt. Je nach Änderungsgranularität werden dabei der gesamte Inhalt des Verzeichnisses, nur ein Teil des Verzeichnisbaumes oder Änderungen seit dem letzten Update übermittelt.

Der Vorteil dieser Methode liegt in der einfachen Implementation, da geänderte Einträge im Ziel einfach gelöscht und neu angelegt werden. Daraus folgt jedoch der Nachteil, dass die

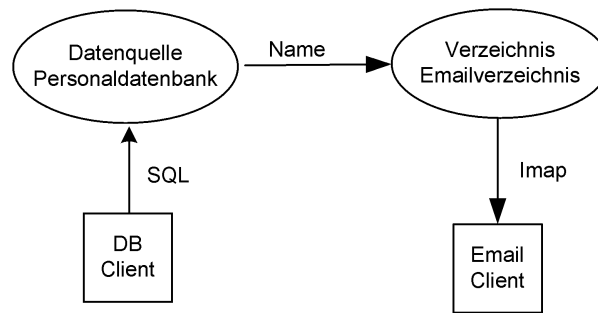


Abbildung 2.1: Unidirektionale Synchronisation

Administration der Daten nur im Quellverzeichnis (Schreibzugriff nur hier möglich) erfolgen darf.

Die unidirektionale Synchronisation wird oft verwendet, um z.B. Informationen aus Personaldatenbanken, in dem die meisten Nutzerdaten erstmalig erzeugt werden, in Verzeichnisdienste einzupflegen. Die Datenpflege und die Nutzung der Informationen können so voneinander getrennt werden.

2.2.2 Bidirektionale Synchronisation

Hier werden Datenänderungen in beiden Richtungen zwischen mehreren Datenquellen verbreitet, was ein Maximum an Flexibilität bietet. Es muss kein einzelner Eigentümer der Daten definiert werden, beide Datenquellen können Lese-/Schreibquelle der Daten bleiben und sind weiterhin getrennt voneinander administrierbar. Ein Nachteil der bidirektionalen Synchronisation besteht in der hohen Komplexität.

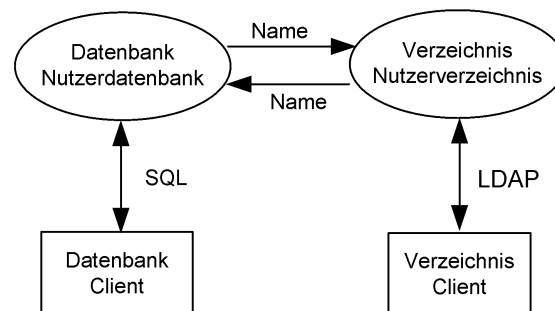


Abbildung 2.2: Bidirektionale Synchronisation

Da Änderungen auf beiden Seiten möglich sind, kann es zu Konflikten bei der Synchronisation kommen. Beispielsweise kann eine Änderung an demselben Attributwert eines Objektes in beiden beteiligten Datenquellen innerhalb eines Synchronisationszyklus durch-

geführt werden. Um dieses Problem zu umgehen können Zeitstempel verglichen werden, was eine Zeitsynchronisation bedingt.

Eine organisatorische Lösung des Problems, z.B. bei der Passwortsynchronisation, besteht in einer zentralen Änderung des sonst zu synchronisierenden Attributes. Das kann mit Hilfe einer zentralen Web-basierten Schnittstelle und einer Weiterleitung der Änderungen durch unidirektionale Synchronisation implementiert werden.

2.3 Methoden zur Auslösung von Synchronisationsvorgängen

Prinzipiell gibt es zwei Wege, einen Synchronisationsvorgang auszulösen:

- Bei der **ereignisgesteuerten** Auslösung bewirkt eine Änderung eines Eintrages im beteiligten Verzeichnisdienst ein sofortiges Auslösen eines Abgleichvorganges, der nur das betreffende Objekt bzw. dessen geänderte Attribute umfasst. Damit ist es möglich, das Übertragungsaufkommen an das eigentliche Informationsaufkommen anzupassen.
- Bei **zeitgesteuerten** Auslösemethoden wird der Datenbestand eines beteiligten Verzeichnisdienstes in regelmäßigen Abständen (Synchronisationsfrequenz) nach Änderungen durchsucht und diese (in Abhängigkeit von der unterstützten Änderungsgranularität) an die Teilnehmer weitergereicht. Das komplette Durchsuchen des Verzeichnisbaumes, auch bei nur wenigen Änderungen, wirkt sich nachteilig auf die absolut benötigte Zeit für den Abgleichvorgang aus. Der Vorteil dieser Methode liegt in der einfacheren Implementierung.

2.4 Zeitsynchronisation zur Ordnung von Änderungen

Bei verteilten Systemen laufen die Systemuhren der beteiligten Rechner aufgrund von unterschiedlichen Umwelteinflüssen und Materialeigenschaften nicht immer synchron. Sie besitzen keine gemeinsame Uhr. Ein ereignisbasierter Abgleich von Verzeichnisdaten (asynchroner Abgleich) ist nicht immer möglich, da nicht alle Verzeichnisdienste diese Form von Synchronisation unterstützen. Somit ist ein Abgleichmechanismus auf exakte Uhrzeiten angewiesen, um anhand von Zeitstempeln Änderungen der Daten ihrer Aktualität nach zu ordnen und Änderungskonflikte aufzulösen.

Es gibt verschiedene Möglichkeiten, die lokalen Uhren der Systeme zu synchronisieren. Einerseits kann der Abgleichmechanismus mit Hilfe von speziellen Nachrichten die Uhren untereinander abgleichen, andererseits kann das *Network Time Protocol* (NTP) für eine dezentrale Uhrensynchronisation benutzt werden.

Eine Betrachtung verzeichnisdienstinterner Synchronisationsmodelle soll helfen das Problem weiter zu veranschaulichen.

Verzeichnisrepliken werden zum großen Teil nur zeitgesteuert synchronisiert. Sie sind somit lose gekoppelt und gleichzeitige Änderungen derselben Attributwerte auf verschiedenen Repliken sind möglich. Es existieren drei unterschiedliche Methoden, um Änderungen abzugleichen:

- **Vergleich der Änderungszeit**

Eine Methode zum Auflösen von mehreren Änderungen am selben Objekt ist der Vergleich der Änderungszeit. Damit kann sichergestellt werden, dass die letzte Änderung in die Verzeichnisdatenbank geschrieben wird. Die Änderungszeit wird immer dann gesetzt, wenn Änderungen im Verzeichnis vorgenommen werden. Abhängig von der Änderungsgranularität werden diese Zeitstempelinträge auf den verschiedenen Ebenen (Attribut- oder Objektebene) vergeben. Für einen Vergleich der Änderungszeiten muss sichergestellt sein, dass alle Uhren der Verzeichnisserver eine abgegliche Zeit haben.

Ein Beispiel für diese Methode ist der Novell Verzeichnisdienst, welcher damit die Synchronisation seiner Repliken steuert.

- **Vergleich der Änderungssequenznummern**

Diese Methode benutzt eine andere Art von Ereignisstempeln, *Update Sequence Numbers (USN)*, um Verzeichnisänderungen zu ordnen. Dabei besitzt jeder Server einen internen Zähler. Wird ein Attribut verändert, so wird der Zähler des Servers um eins erhöht und bei dem Attribut abgespeichert. Weiterhin besitzt jedes Attribut noch einen weiteren eigenen Zähler, welcher bei einer Änderung ebenfalls erhöht wird.

Bei einem Abgleich der Repliken wird zunächst der Quellserverzähler mit dem abgespeicherten Wert auf dem Zielserver verglichen. Danach werden alle Attribute mit einem höheren Serverzähler als dem beim Replikationspartner gespeicherten untersucht. Abgeglichen werden nur Attribute, deren Attributzähler über dem Attributzähler des im Zielserver gespeicherten Attributes liegen. Bei einem erfolgreichen Abgleich der Daten wird auf dem Zielserver dann der Attributzähler des aktualisierten Attributes übernommen und der interne aktuelle Zählerstand des Zielservers beim Attribut gespeichert.

Nach Abschluss des Abgleiches tauschen die beiden Server ihre aktuellen Zählerstände aus und speichern sie für den nächsten Abgleich in lokalen Tabellen ab. Falls bei der Synchronisation Kollisionen (gleicher Attributzählerstand) auftreten, werden die

Zeitstempel der Einträge miteinander verglichen und die letzte Änderung wird abgespeichert.

Der Microsoft Verzeichnisdienst Active Directory benutzt zwei solcher Sequenznummern, um die Synchronisation seiner Repliken steuern zu können.

- **Benutzung einer Änderungsdatei**

Manche Verzeichnisdienste (z.B. LDAP) benutzen eine Änderungsdatei auf ihren Verzeichnisservern, in welche alle Änderungen am Verzeichnis in Reihenfolge des Auftretens eingetragen werden. Bei dieser Changelog-Methode spielt die Datenquelle ihren Änderungsdateiinhalt ab und überträgt die darin enthaltenen Änderungen zum Ziel.

Diese Methode ist einfach umzusetzen, jedoch sind die Änderungsdaten nach der Zeit ihres Auftretens geordnet. Bei mehrfachen Änderungen desselben Eintrages werden diese beim Ziel wieder nacheinander ausgeführt, obwohl bis auf die letzte Änderung alle weiteren hinfällig geworden sein können. Andererseits bietet diese Methode die Möglichkeit, alle Änderungen bis zu einem definierten Punkt (z.B. Auftreten eines Fehlers) wieder in das Verzeichnis einzuspielen.

2.5 Granularität der Änderungserkennung

Die Änderungsgranularität ist eine der wichtigsten Faktoren eines Abgleiches von Informationen. Durch eine hohe Auflösung lässt sich das zu übertragende Datenvolumen und demzufolge der Netzwerkverkehr enorm verringern. Die Frequenz des Abgleiches kann somit erhöht, die Performance gesteigert und dadurch die Informationsqualität verbessert werden. Weiterhin kann durch eine erhöhte Änderungsgranularität das Auftreten von Änderungskonflikten (hervorgerufen durch das gleichzeitige Ändern von Einträgen) minimiert werden.

Die Granularität beschreibt, welche Einträge eines Verzeichnisdienstes bei einer Änderung eines Attributwertes abgeglichen werden. Manche Verzeichnisdienste sind in der Lage, die Änderungen auf Attributebene zu erkennen, andere wiederum nur auf Objektebene.

Allgemein können drei verschiedene Auflösungsebenen unterschieden werden:

- **Verzeichnisebene**

Es existiert keine Änderungserkennung, vielmehr wird in großen Zeitabständen das komplette Verzeichnis abgeglichen. Für einen Synchronisationsmechanismus ist diese Abgleichform aufgrund des hohen Datenaufkommens nur bedingt einsetzbar. Sie wird vor allem zum Anlegen von Sicherheitskopien der Verzeichnisdatenbank, bei der Initialisierung des Abgleichvorganges oder im Fehlerfall für einen Neuaufbau der Datenbank eines Synchronisationspartners verwendet.

- **Objektinkrementeller Abgleich**

Dabei wird auf Objektebene untersucht, ob sich Einträge seit dem letzten Abgleich verändert haben. Ist dies der Fall, werden alle Attribute (auch die nicht geänderten) dieses Objektes abgeglichen. Der Vorteil ist die schnelle Änderungserkennung, da Änderungsabfragen nur auf Objektebene durchgeführt werden und somit die Abfragevorgänge schneller durchgeführt werden. Der Nachteil dieser Granularitätsstufe ist der unnötige Übertragungsverkehr, der durch den Transport von nicht geänderten Attributwerten entsteht.

Ein weiterer Nachteil sind Änderungen eines Objektes innerhalb der Verzeichnisdienste, auch bei für den Abgleich nicht relevanter, also von dem Abgleich ausgeschlossener, Attributwertänderungen. Ein Beispiel dafür ist das Setzen des Attributes „Logintime“ des Objektes „User“ bei jeder Beglaubigung an einem NDS-Verzeichnisserver.

- **Attributinkrementeller Abgleich**

Dabei werden Mechanismen zum Erkennen von Änderungen auf Attributebene benutzt. Das erlaubt folglich auch eine Synchronisation auf Attributebene. Der Vorteil besteht in der Minimierung des Übertragungsaufkommens und der Vermeidung unnötiger Änderungskonflikte.

In der Praxis ist es nicht einfach, Änderungen auf Attributebene auszuwerten, da diese nicht über alle Schnittstellen ausgegeben werden. Einfache Abgleichmechanismen arbeiten deshalb häufig nur mit objektinkrementellen Synchronisationstechniken.

2.6 Methoden zum Zusammenfassen von Verzeichnisinformationen

Bei der Synchronisation von mehreren Datenquellen entsteht der Wunsch, verwandte Informationen zusammenzuführen. Es gibt zwei Methoden, die sich vor allem bei der Zwischenspeicherung der Daten und somit der Eignung für eine Verzeichnissynchronisation unterscheiden. Während die erste Methode die Verzeichnisinformationen sowohl logisch als auch physikalisch (Zwischenspeicherung in der Verzeichnisdatenbank) zusammenführt, berücksichtigt die zweite Methode nur eine logische Zusammenfassung.

2.6.1 Join – Zusammenfassung von Informationen

Bei dieser Methode werden die zu synchronisierenden Informationen aus den beteiligten Datenquellen über Abgleichattribute zu logischen „Meta-“Objekten zusammengefasst und in

der Verzeichnisdatenbank (DIB) eines zentralen Verzeichnisses gespeichert. Das Abgleichattribut muss in den abzugleichenden Datenquellen vorhanden und jeweils eindeutig sein. Es entspricht dem Primärschlüssel von Datenbanktabellen.

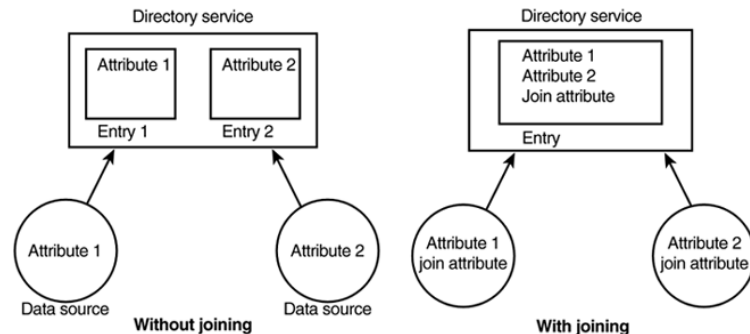


Abbildung 2.3: Join Methode aus [TAHP03]

Beispiele für geeignete eindeutige Abgleichattribute sind die Ausweisnummer eines Nutzers oder die Matrikelnummer eines Studenten. Falls kein eindeutiges Abgleichattribut vorhanden ist, können auch verkettete Attribute als solches benutzt werden. Eine Verknüpfung von Nachname, Geburtsdatum und Geburtsort eines Benutzers würde einem verketteten Abgleichattribut entsprechen.

2.6.2 Virtuelles Verzeichnis – Zentrale Verlinkung von Daten

Der Name *virtuelles Verzeichnis* beschreibt den Hauptunterschied zur Joinmethode. Ein zentrales Verzeichnis, welches als virtuelles Verzeichnis agiert, speichert keine Datensätze, sondern nur Verweise zu den Informationen der beteiligten Datenquellen in der eigenen Verzeichnisdatenbank. Bei einer Client-Abfrage an das virtuelle Verzeichnis wird diese Abfrage in die Protokolle der angeschlossenen Datenquellen übersetzt und dahin weitergeleitet. Die Antworten der Datenquellen werden auf dem Rückweg wieder in das Client-Protokoll übersetzt und an den Client ausgeliefert. Ein virtuelles Verzeichnis wird oft als Gateway zum Übersetzen von proprietären Datenbankprotokollen in ein einheitliches Clientprotokoll benutzt. Der Client braucht nur ein Übertragungsprotokoll zu benutzen, um mit vielen Datenquellen Kontakt aufzunehmen.

Die Vorteile liegen in der Aktualität der Daten und der Weiterleitung der Updateanfragen. Die aufwendigen Synchronisationsmethoden sowie eine Datenredundanz durch Zwischenspeicherung im zentralen Verzeichnis entfallen ebenfalls. Dem Vorteil der Datenaktualität stehen allerdings erhöhte Antwortzeiten gegenüber. Die Antwortzeiten des virtuellen Verzeichnisses sind nicht nur von der Antwortzeit der langsamsten Datenquelle abhängig. Hinzu kommen noch die Zeiten für die Übertragung einschließlich Protokollübersetzung und

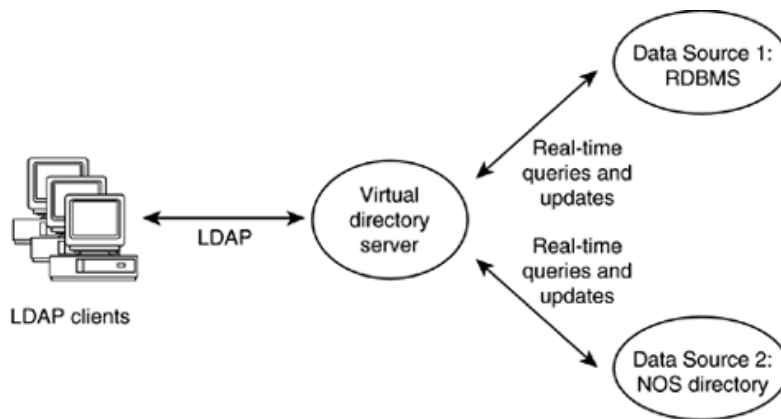


Abbildung 2.4: Virtual Directory aus [TAHP03]

Fehlerkontrolle.

Die in dieser Diplomarbeit betrachteten Verzeichnisdienste (NDS und ADS) werden meist als Ressourcenverwalter und Authentifizierungssysteme für Netzwerke verwendet. Die Verzeichnisdienste sind demnach selbst auf aktuelle Informationen angewiesen. Die angeschlossenen Client-Rechner greifen direkt auf die Verzeichnisse zu, um die Funktionalitäten der Verzeichnisdienste nutzen zu können. Es muss demnach eine Synchronisation der Verzeichnisdienste untereinander stattfinden. Mit Hilfe eines virtuellen Verzeichnis lässt sich ein Abgleich der Informationen in den Verzeichnissen jedoch nicht realisieren. Es existieren keine Änderungsdienste für die angeschlossenen Datenquellen.

2.7 Funktionalitäten für einen Informationsabgleich

Zu einer Synchronisationslösung für Verzeichnisdienste gehört auch ein flexibles Management der Verzeichnisinformationen. Dazu zählt das Zuordnen der Objekte und Attribute der abzugleichenden Verzeichnisdienste, sowie die Kontrolle der Informationsflüsse zwischen den Verzeichnissen. Des Weiteren ermöglicht das Filtern von Objekten und Attributen eine selektive Integration der Daten und reduziert das Übertragungsaufkommen der Synchronisation. Zur näheren Erläuterung dient die folgende Aufzählung von benötigten Funktionalitäten eines Informationsabgleiches:

- **Schema Mapping**

Eine Zuordnung der Schemata wird benötigt, wenn zwei Datenquellen verschiedene Attributnamen oder Datenfelder zum Abspeichern derselben Informationen nutzen.

Hier ein Beispiel, um das Problem besser zu verdeutlichen: Der Loginname eines Nut-

zers wird in den NDS unter dem Attributnamen „Loginname“, im Active Directory aber unter „sAMAccountName“ abgespeichert. Die Werte der beiden Einträge beinhalten dieselbe Information. Es muss also eine Schemazuordnung zwischen den beiden Attributen hergestellt werden.

Zum Steuern dieser Schemazuordnungen bieten einige Synchronisationslösungen (z.B. Metaverzeichnisse) Regeln an. Die Unterstützung reicht von einer festen eins-zu-eins (Attribut-Attribut oder Objekt-Objekt) bis hin zu einer flexiblen Zuordnung von beliebigen Attributen zu Attributketten.

- **Datenumwandlung**

Die Zuordnung von Attribut- oder Objektpaaren ist oft auch mit einer Datenumwandlung verbunden, da die Verzeichnisdienste dieselben Informationen in unterschiedlichen Datentypen ablegen.

Zum Beispiel kann die Bezeichnung eines Fachbereiches einerseits als Attributwert „fb1“ andererseits aber nur als Ziffer „1“ in den Verzeichnissen abgelegt sein. Für den Abgleich von zwei Verzeichnisdiensten müssen die Informationswerte zuerst in ein einheitliches Format überführt werden. Danach ist es dann möglich, einen logischen Abgleich der Attributwerte vorzunehmen.

- **Namensraumübersetzung**

Die Namensräume der verschiedenen Verzeichnisdienste unterscheiden sich oft hinsichtlich ihrer Struktur. Dabei befinden sich logisch verwandte Objekte zweier Verzeichnisdienste im einfachsten Falle nur an unterschiedlichen Stellen der Namensraumhierarchie. Oft jedoch sind die Namensräume beider Verzeichnisse unterschiedlich aufgebaut und müssen zueinander in Beziehung gebracht werden.

Beispielsweise besitzt ein Nutzerobjekt in dem FHTW eigenen NDS-Verzeichnisbaum den Namensraumeintrag „taito.users.fhtw“, im Active Directory-Verzeichnisbaum jedoch „taito.benutzer.fb1“. Eine Namensraumübersetzung muss in diesem Beispiel eine Zuordnung der beiden Kontexte vornehmen und diese ineinander überführen können.

- **Informationsfilterung**

Eine Synchronisation bezieht sich ausschließlich nur auf eine Auswahl von Objekten und Attributen zweier Verzeichnisdienste. Ziel einer Informationsfilterung ist die Auswahl relevanter und das Ausschließen für einen Abgleich unerheblicher Informationen zu steuern. Interessant ist dabei die Granularität der Informationsfilterung. Eine Filterung auf Attributebene ist wünschenswert, um das Übertragungsaufkommen zu mindern und unnötige Synchronisationsvorgänge zu vermeiden.

Bei einem Abgleich von Verzeichnisdiensten sind die Betriebsattribute eines jeden

(z.B. Laufwerksmapping in der jeweiligen Umgebung) meist irrelevant. Sie sollten von einer Synchronisation mit Hilfe von Filtern ausgeschlossen werden.

- **Beachtung der Informationsquellen**

Um die Konsistenz des Datenbestandes zu sichern und eine damit verbundene hohe Informationsqualität zu erreichen, ist eine Auswahl von Informationseigentümern bei einer Synchronisation unerlässlich. Die Wahl der Informationseigentümer hängt vom Ursprung der Informationen ab. Beispielsweise werden Mitarbeiterinformationen in einem Unternehmen als erstes in einer Personaldatenbank erzeugt. Für Attribute wie „Name“, „Jobtitel“ oder „Abteilung“ ist die Personaldatenbank Informationseigentümer und die Werte dieser Attribute dürfen nur dort verändert werden. Die E-Mailadresse eines neuen Mitarbeiters jedoch wird durch das Mailsystem erzeugt und kann danach in die Personaldatenbank eingepflegt werden. Das Mailsystem mit seinem Verzeichnis ist in diesem Falle also der Informationseigentümer des Attributes „E-Mailadresse“.

2.8 Passwortsynchronisation

Passwörter von Benutzern stellen ein besonderes Problem für eine Synchronisation von Verzeichnisdiensten dar. Die Passwörter werden in den Verzeichnissen nicht als Klartextpasswörter abgespeichert und sind somit nicht direkt synchronisierbar. Die Verzeichnisdienste führen eine irreversible Berechnung auf Grundlage des Klartextpassworts durch, das übertragen wird, wenn ein Passwort angelegt oder geändert wird. Der resultierende Hashwert wird in der Verzeichnisdatenbank abgespeichert. Wenn ein Benutzer sich einloggen möchte, wird die gleiche irreversible Berechnung noch einmal durchgeführt und das Resultat mit dem in der Verzeichnisdatenbank gespeicherten Wert verglichen. Für die Berechnung werden unterschiedliche Algorithmen verwendet, wodurch ein direkter Vergleich der Hashwerte ebenfalls unmöglich wird.

Es existieren verschiedene Lösungen für dieses Problem. Eine Möglichkeit ist, die Passwortänderungen (wie im Abschnitt bidirektionale Synchronisation erläutert) von einem zentralen Interface entgegenzunehmen und danach in die jeweiligen Verzeichnisdienste einzutragen. Damit wäre eine Synchronisation der Passwörter hinfällig, jedoch müssten die Benutzer damit vertraut gemacht werden.

Eine weitere Lösung ist die Weiterleitung der Passwortänderungen in den Client-Umgebungen. Dabei wird das Passwort nach der Eingabe parallel mit Hilfe der verzeichnisdienstspezifischen Algorithmen verschlüsselt und an die Verzeichnisse weitergeleitet.

2.9 Performanceanforderungen an Abgleichmechanismen

Die Verfügbarkeit und Leistung einer Abgleichtechnik hängt stark von dem abzugleichenden Datenumfang der beteiligten Verzeichnisdiensten ab.

Verzeichnisdienste stellen Redundanzen (z.B. Replikation der Verzeichnisdaten) bereit, um auch bei einem Ausfall eines Servers die enthaltenen Informationen publizieren zu können. Eine Abgleichtechnik sollte durch Skalierung in der Lage sein diese Redundanzen zu nutzen, um die eigene Performance zu erhöhen.

Ein weitere Faktor ist der Durchsatz. Er ist eine Kenngröße für die Geschwindigkeit, mit der Rechenarbeit erledigt werden kann und ist abhängig von der Verarbeitungs- und Datenübertragungsgeschwindigkeit. Für die Performance eines Abgleichvorganges ist der Durchsatz der Verzeichnisdienste, des Abgleichmechanismus und der Durchsatz des Netzwerkes entscheidend.

Die Wahl der *Synchronisationsfrequenz* wirkt sich direkt auf die Performance des Abgleiches aus. Sie ist sowohl vom erwarteten Datenübertragungsvolumen, als auch von der für einen Abgleich benötigten Zeit abhängig. So muss gewährleistet sein, dass ein Abgleichzyklus beendet ist, bevor der nächste gestartet wird. Das ist im besonderen bei zeitgesteuerten Synchronisationsvorgängen zu beachten. Bei ereignisgesteuerten Synchronisationsvorgängen muss ebenfalls sichergestellt sein, dass alle Änderungen innerhalb einer definierten Zeitspanne durchgeführt werden, da sie sonst schon durch neue Änderungen hinfällig geworden sein können.

Weitere Faktoren – wie z.B. Antwortzeiten und Bandbreite – der Verzeichnisdienste dürfen durch einen Synchronisationsvorgang nicht beeinträchtigt werden. Das heißt, die Leistung der Synchronisation darf nicht als Folge der engen Verbundenheit zu Lasten der beteiligten Verzeichnisdienste gehen.

2.10 Administration der Synchronisation

Die Benutzbarkeit einer Synchronisationstechnik hängt in starkem Maße von deren Administrationsmöglichkeiten ab. Der gesamte Synchronisationsvorgang sollte zentral anpassbar und nicht zu komplex aufgebaut sein. Es sollte ein Interface dezentral verfügbar sein, das alle Einstellungen ermöglicht. Dieses ist erforderlich, da die beteiligten Verzeichnisdienste von unterschiedlichen Administratoren gepflegt werden. Außerdem sollten unterschiedliche administrative Ebenen unterschieden werden. Beispielsweise sollte ein Quellverzeichnisadministrator nur den Informationsfluss seines Verzeichnisses steuern können, ein Synchronisationsadministrator jedoch auch in der Lage sein, Anpassungen an grundlegenden Einstellungen des Abgleichvorganges vorzunehmen.

In der Praxis haben sich vor allem Web-basierte Schnittstellen bewährt. Sie bieten bei

einigen Herstellern die Möglichkeit – abhängig vom eingeloggteten Nutzer – unterschiedliche Verwaltungsfunktionen darzustellen.

2.11 Anforderungsprofil für eine Lösungsfindung

Bei einem Abgleich der beiden Verzeichnisdienste NDS und ADS sind vor allem nutzerspezifische Daten relevant. Diese Daten sind in unterschiedlichen Verzeichnisbaumstrukturen mit voneinander unabhängigen Namensräumen organisiert. Die Schemata und somit der strukturelle Aufbau der Verzeichnisse unterscheidet sich voneinander. Die Daten befinden sich demnach in unterschiedlichen Hierarchien. Ein Abgleichsystem muss sowohl die Hierarchie, als auch die Schemata der Verzeichnisse erkennen und verwalten können.

Nicht alle Attribute der jeweiligen Verzeichnisdienste sind für einen Abgleich relevant (Betriebsattribute), so dass für eine Synchronisation Informationsfilter benötigt werden. Weiterhin muss ein Abgleichsystem die verschiedenen Eigentümer der Daten berücksichtigen.

Beide Verzeichnisdienste werden oft als Beglaubigungsstelle für PC Netzwerke verwendet. Ein Passwortabgleich ist daher gewünscht und muss bidirektional möglich sein. Das Abgleichsystem muss somit beide Arten der Passwortverschlüsselung der Verzeichnisdienste unterstützen, um eine Passwortsynchronisation zu ermöglichen.

Beide Verzeichnisdienste unterstützen intern eine Änderungsgranularität auf Attributebene. Ein Abgleichsystem sollte diese Änderungsgranularität übernehmen können, um das Datenübertragungsvolumen zwischen den Verzeichnisdiensten möglichst klein zu halten und unnötige Änderungskonflikte zu vermeiden.

Beide Verzeichnisdienste arbeiten intern mit verschiedenen Vergleichsmechanismen, um Konflikte bedingt durch gleichzeitige Änderungen aufzulösen. Der gemeinsame Nenner zum Vergleichen von Änderungen sind Zeitstempel. Sie werden sowohl in den ADS, als auch in den NDS benutzt. Ein Abgleichsystem muss in der Lage sein, diese Zeitstempel auszuwerten.

Eine Auslösung von Synchronisationsvorgängen sollte sowohl zeitlich (für den periodischen Abgleich großer Datenmengen) als auch ereignisbasiert (Passwortsynchronisation) möglich sein.

Die Administration des Synchronisationsvorganges muss dezentral und systemunabhängig verfügbar sein. Weiterhin sollten verschiedene Administrationsebenen (Synchronisationsadministrator, Verzeichnisdienstadministrator) unterstützt werden.

Für einen direkten Abgleich von einem ADS-Verzeichnis und einem NDS-Verzeichnis sind keine Zusammenfassungsmethoden notwendig. Die meisten Attribute der Objektklas-

sen können in einen direkten Zusammenhang gebracht werden. Für eine eventuelle spätere Erweiterung des Abgleichsystems durch die Anbindung weiterer Datenquellen sind Join-Methoden zum Zusammenfassen der Verzeichnisinformationen in einem zentralen Verzeichnis jedoch sinnvoll. Sie werden benötigt, um die Informationen der unterschiedlichen Datenquellen mit Hilfe von Abgleichattributen in Beziehung zu bringen und nur einen Objekteintrag für identische Informationen in einem zentralen Verzeichnis zu benötigen.

Ein Abgleichmechanismus muss trotz des geforderten Funktionsumfangs performant arbeiten. Er muss die Nebenläufigkeit von Verzeichnisdienst und Informationsabgleich auch bei hohen Synchronisationsfrequenzen sicherstellen.

Um auch bei grossen Datenübertragungsaufkommen und steigenden Nutzerzahlen noch performant arbeiten zu können, muss er skalierbar sein. Dabei sollte er die vorhandenen Skalierungsmechanismen der Verzeichnisdienste (interne Verzeichnisreplikation) nutzen, um so seine Prozesslast verteilen zu können und die Nebenläufigkeit von Verzeichnisdienst und Abgleich sicherzustellen.

Des Weiteren dürfen die verschiedenen Sicherheitsanforderungen der Verzeichnisdienste nicht unterlaufen werden. Daten, welche zwischen den Verzeichnisdiensten übertragen werden, müssen verschlüsselt werden.

Eine robuste Fehlerverarbeitung ist unabdingbar, um die Konsistenz der ADS und NDS nicht zu gefährden. Dabei müssen sowohl die Fehler der Verzeichnisdienste als auch die Fehler des Synchronisationsmechanismus ausgewertet und im Datenfluss berücksichtigt werden.

3 Lösungsansätze für die Verzeichnisdienstsynchronisation

- 3.1 Lose Offlinekopplung 37
- 3.2 Programmierschnittstellen und Entwicklungsumgebungen 39
 - 3.2.1 LDAP C API 40
 - 3.2.2 Perl 40
 - 3.2.3 Java 40
 - 3.2.4 ADSI 41
- 3.3 Microsoft Directory Synchronization Service (MSDSS) 42
- 3.4 Metaverzeichnisse 43
 - 3.4.1 Aufbau eines Metaverzeichnisses 44
 - 3.4.2 Metaverzeichnis als zentraler Informationsspeicher 44
 - 3.4.3 Metaverzeichnis als Informationshändler 45
 - 3.4.4 Produkte 45
 - 3.4.5 Zusammenfassung 48
- 3.5 Directory Services Markup Language – die Zukunft? 49
- 3.6 Zusammenfassender Vergleich der Lösungsansätze 51

Nachdem die Anforderungen an eine Synchronisation von Verzeichnisdiensten näher erklärt wurden, werden in diesem Kapitel die verschiedenen Lösungsansätze betrachtet.

3.1 Lose Offlinekopplung

Eine Synchronisation von Verzeichnisdiensten kann im einfachsten Fall über eine lose Offlinekopplung erreicht werden. Die Verzeichnisdaten des ersten Verzeichnisses werden in eine Datei geschrieben und in das Zielverzeichnis importiert. Als Austauschformate können z.B. kommasparierte Textdateien verwendet werden. Der Standard für den Austausch von Verzeichnisisnformationen auf Dateibasis ist das LDIF-Format.

Das *LDAP Data Interchange Format* (LDIF) wird von fast allen Verzeichnisdiensten unterstützt. Nicht nur die Verzeichnisdienste unterstützen den Import und Export von LDIF-Dateien, auch die meisten LDAP APIs können diese Daten lesen und schreiben. LDIF Dateien bilden die Verzeichnisinformationen in einem standardisierten Textformat (RFC 2849 [Goo00]) ab. Eine LDIF Datei kann eine Menge von Verzeichniseinträgen oder Änderungen an existierenden Verzeichniseinträgen enthalten. Beide zusammen dürfen nicht in derselben Datei enthalten sein. Gebräuchliche LDAP Operationen, wie „add“, „delete“, und „modify“ können zum Ändern von Verzeichnisinformationen in LDIF Dateien benutzt werden.

LDIF Beispiel

```
version:1
dn: cn=taito,o=FHTW
changetype: add
uid: taito
messageServer: cn=HRZT1,ou=HRZ,o=FHTW
sn: radtke
objectClass: person
objectClass: top
```

Für Realisierung einer LDIF Synchronisation muss der ausgewählte Verzeichnisdienstinhalt auf dem Quellserver in eine LDIF Datei exportiert, die Datei zu dem Zielsystem kopiert und danach in den Zielverzeichnisbaum importiert werden. [Goo00]

Microsoft bietet auf seinen Serverbetriebssystemen ein Kommandozeilenwerkzeug zum Import und Export von LDIF Dateien in oder aus dem Active Directory mit dem Namen *LDIFDE* an. Damit können die beschriebenen Funktionalitäten einer LDIF Schnittstelle benutzt werden.

Import Conversion Export Utility (ICE) Die Novell Directory Services gehen einen Schritt weiter. Mit dem *Import Conversion Export Utility* (ICE) können LDIF Dateien in jedes beliebige LDAP Verzeichnis importiert oder aus diesem exportiert werden. Ausserdem kann mit Hilfe dieses Werkzeuges eine Migration zwischen zwei LDAP Verzeichnissen aufgebaut werden.

ICE besteht aus drei Bausteinen. Eine Quellroutine liest die Daten aus den LDIF Dateien ein. Die eingelesenen Daten werden zu einer Engine, welche die Daten mit Hilfe von XML Regeln weiterverarbeiten kann, übertragen und danach an eine Zielroutine übergeben.

The Novell Import Conversion Export engine lets you specify a set of rules that describe processing actions to be taken on each record received from the source handler and before the record is sent on to the destination handler. These rules are specified in XML (either in the form of an XML file or XML data stored in the directory) and solve [...] problems when importing entries from one LDAP directory to another [...]. [Nov03e]

ICE beinhaltet vereinfachte Funktionalitäten für einen Informationsabgleich. Durch die Unterstützung von XML Regeln ist die Engine in der Lage, Unterschiede des Schemas, der Hierarchie oder fehlende Informationen auszugleichen. Es gibt drei verschiedene XML Regelsätze, welche dafür verwendet werden können.

- **Plazierungsregeln** setzen die Namensraumübersetzung (siehe Kapitel 2 Informationsabgleich) um.
- **Erstellungsregeln** vervollständigen Einträge um Informationen, welche zum Anlegen im Zielverzeichnis noch zusätzlich benötigt werden.
- **Schema Mapping Regeln** stellen die Verbindung zwischen den Quell- und Zieltributen her und erweitern, falls erforderlich, das Zielverzeichnis um die fehlenden Attribute.

Dieses Systemwerkzeug kann sowohl über das grafische Javainterface *ConsoleOne* oder über das Webfrontend *iManager* der NDS als auch in der Kommandozeile eines NDS-Verzeichnisservers benutzt werden. Es ist Bestandteil einer NDS-Installation und kann mit beliebigen LDAP-Datenquellen arbeiten.

Zusammenfassend eignen sich LDIF-Dateien zur Datensicherung oder zum einmaligen Abgleich von Verzeichnisdiensten. Die Umformungsregeln der ICE Umgebung bieten einfache Möglichkeiten zum Umwandeln von Informationen. Eine Synchronisation von Passwörtern ist auf der Basis einer Offlinekopplung nicht umzusetzen. Sowohl die LDIF-Schnittstelle selbst, als auch ICE, bieten keine Methoden zum Auflösen von Änderungskonflikten, keine zeitlichen oder ereignisorientierten Steuerungen oder Funktionalitäten zur Änderungserkennung.

3.2 Programmierschnittstellen und Entwicklungsumgebungen

Für Entwickler werden seitens der Verzeichnisdiensthersteller verschiedene *Software development kits* (SDK) angeboten. Diese benutzen verschiedene *Application Programming Interfaces* (APIs) und bieten so dem Entwickler eine Möglichkeit, über Funktionen auf die

Verzeichnisse zuzugreifen. Die Unterstützung der Entwicklungsumgebungen beschränkt sich bei den meisten APIs auf den Verzeichniszugriff, Funktionalitäten für einen Synchronisationsvorgang müssen komplett selbst implementiert werden.

3.2.1 LDAP C API

Für den Verzeichniszugriff über das LDAP Protokoll, gibt es seit 1995 die LDAP C API. Diese Schnittstelle wurde an der Universität von Michigan entwickelt und ist unter anderem im RFC 1823 [HS95] veröffentlicht. Darin beschrieben wird eine C Programierschnittstelle, welche in der Lage ist, Anfragen und Änderungen an einem LDAP Verzeichnis vorzunehmen.

Die originale LDAP API wurde für LDAP v.2 geschrieben und ist seit dem durch weitere Entwürfe dem neuen LDAP v.3 Standard angeglichen worden. Die Programierschnittstelle unterstützt synchrone und asynchrone Operationen. Sie folgt dem funktionalen Modell von LDAP und bietet entsprechende Funktionen (bind, search, add, delete, modify,..) an.

3.2.2 Perl

Der Zugriff auf Verzeichnisse mit Perl kann über die *Net::LDAP* Module erfolgen. Perl ist relativ einfach zu erlernen und ist als Skriptsprache zum Zusammenbringen von verschiedenen Informationen bestens geeignet.

Es existieren zwei Modulsammlungen zum Entwickeln von LDAP Programmen mit Perl. Die erste Modulsammlung ist *PerLDAP*¹, welche 1998 als Teil des Mozilla Projektes veröffentlicht wurde. *PerLDAP* ist rund um die LDAP C Api geschrieben und benutzt dessen Zugriffsmethoden. Ausserdem gibt es eine *Perl-LDAP*² Bibliothek, welche als objektorientierte Schnittstelle zu einem LDAP Server dienen kann. Diese Bibliothek ist komplett in Perl geschrieben und kommt ohne C Compiler oder Erweiterungen aus. Beide Modulsammlungen unterstützen die Basisoperationen von LDAP, als auch LDAP über *Secure Sockets Layer* (SSL) und den Import und Export von LDIF Dateien. Ausserdem sind Module zum Lesen und Schreiben von Daten in der *Directory Service Markup Language* (DSML) (siehe Abschnitt DSML) enthalten.

3.2.3 Java

Mehrere Entwicklungsumgebungen sind für den LDAP Verzeichniszugriff über Java verfügbar. Eine Schnittstelle wurde ursprünglich von Netscape entwickelt. Der zweite Weg für einen Zugriff über verschiedene Verzeichnisprotokolle ist das *Java Naming Directory Interface* (JNDI). Es wurde von SUN entwickelt und ist jetzt Bestandteil der *Java 2 Platform*

¹<http://www.perldap.org/>

²<http://perl-ldap.sourceforge.net/>

Standard Edition (J2SE). JNDI bietet eine einheitliche Schnittstelle für den Zugriff auf verschiedene Verzeichnisdienste. Die Architektur von JNDI ermöglicht den Zugriff auf die Verzeichnisdienste über verschiedene „Service Provider“. Der Vorteil der JNDI Schnittstel-

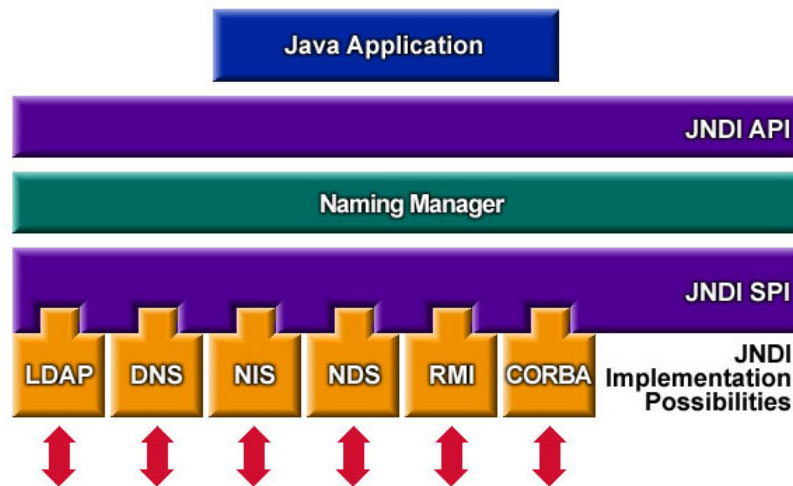


Abbildung 3.1: JNDI Architektur [Sum03b]

le gegenüber anderen LDAP SDKs, welche z.B. von SUN oder Novell angeboten werden, ist der höhere Abstraktionslevel. Dieser stellt die Funktionalität in einem breiten Feld von Verzeichnisumgebungen sicher.

Many benefits exist, with most relating to their close mapping to LDAP operations. However, this close mapping can mean that application written using [...] LDAP-specific APIs will not map well to web service or other non-LDAP directory access mechanisms as they surface. [...] JNDI faces no such limitations. vergl. [Don03, S.216]

3.2.4 ADSI

Das *Active Directory Service Interface* (ADSI) ist eine einfach zu nutzende Verzeichnisprogrammierschnittstelle basierend auf Microsoft's Component Object Model (COM) [ADS00]. Damit ist es möglich, mit jeder Anwendung, welche COM unterstützt, Verzeichnisinformationen zu bearbeiten.

ADSI ist laut Microsoft vor allem als einfache Schnittstelle für eine skriptgesteuerte Administration von Verzeichnissen gedacht. Ebenso wie JNDI bietet ADSI eine abstrakte, objektorientierte Schnittstelle zur Bearbeitung von Verzeichnisinhalten an. Diese orientieren sich an den Aufgaben der Administration und haben entsprechende Funktionalitäten, wie z.B. Anlegen von Nutzern oder das Druckermanagement. Die Schnittstelle unterstützt den

Security Account Manager (SAM) von Windows NT 4.0, die Active Directory Services, Novell NDS und LDAP-fähige Verzeichnisse.

3.3 Microsoft Directory Synchronization Service (MSDSS)

MSDSS ist eine Komponente des *Microsoft Service for Netware v.5*. Sie soll die Administration beider Verzeichnisse (ADS und NDS) vereinfachen und Änderungen zwischen den beiden Verzeichnissen synchronisieren [ADS00]. Die Synchronisation beschränkt sich zum einen auf die Verzeichnisdienste ADS und NDS und zum anderen auf die Unterstützung von Netware v4 und v5 als NDS-Serverbetriebssystem.

Das Produkt Microsoft Service for Netware v.5 ist parallel zum Erscheinen der Windows 2000 Serverfamilie veröffentlicht worden und stellt ein Migrations- und Synchronisationswerkzeug für den Umstieg von Netware auf Windows-basierte Serversysteme dar. So beinhaltet es neben einem Synchronisationsmechanismus für die Verzeichnisdienste auch ein Datenmigrationswerkzeug, um Dateien von einem Netwareserver in Richtung eines Windows-servers zu migrieren. Obwohl Microsoft von einer permanenten Synchronisation mit Hilfe von MSDSS nicht abrät, wird in den Dokumentationen die Synchronisation nur als Zwischenschritt einer kompletten Migration betrachtet.

Microsoft designed Windows 2000 Server and MSDSS to support ongoing mixed deployments, as well as to facilitate a complete conversion to the new operating system. You can use MSDSS synchronization to establish a long-term or permanent co-existence between Active Directory and your Novell directory. Establishing such a mixed environment lets you take advantage of many Active Directory features [...] without converting your entire existing network to the Windows 2000 platform. [...]

Typically, this entails running the two systems in parallel for weeks or months. During this time, you can perform migration tasks that are independent of MSDSS object migration and File Migration Utility file migration, such as replacing NDS-dependent applications with Active Directory-compatible applications. [MSD00b]

MSDSS unterstützt die uni- und bidirektionale Synchronisation. Auch Filter und Objektrelationen werden von MSDSS unterstützt. Diese Funktionalitäten sind jedoch sehr eingeschränkt implementiert. So kann z.B. eine unidirektionale Synchronisation nur in eine Richtung realisiert werden (ADS Änderungen werden in die NDS übertragen). Eine bidirektionale Synchronisation unterstützt für die NDS nur eine Änderungsgranularität auf der

Objektebene. Die Objektrelationen (Siehe Kapitel 2, Schema Mapping) sind nicht anpassbar und unterstützen nur drei Objektklassen (OU, User, Group) und einen kleinen Teil der verfügbaren Attribute.

Im MSDSS Paket enthalten ist auch ein „Password Management“. Damit können zwar keine Passwörter synchronisiert werden, jedoch Regeln für die Passwortgenerierung eines synchronisierten Nutzerobjektes definiert werden. So kann das Passwort automatisch mit dem Wert des Nutzernamens, einem leeren Wert³, einem Zufallswert oder einem zuvor konfigurierten festen Wert gesetzt werden.

Zusammenfassend eignet sich MSDSS nicht für die Synchronisation von Verzeichnisdiensten. Durch den bitteren Beigeschmack der vielen Beschränkungen eines eher für die Migration ausgelegten Synchronisationsmechanismus wird in dieser Diplomarbeit MSDSS in einem abschließenden Vergleich nicht berücksichtigt. Da MSDSS dennoch einen, wenn auch nur begrenzten, Ansatz für die Synchronisation von ADS und NDS aufzeigt, ist es an dieser Stelle der Vollständigkeit halber betrachtet worden.

3.4 Metaverzeichnisse

Metaverzeichnisse wurden entwickelt, um die Informationen der unterschiedlichen Verzeichnisse eines Unternehmens, welche oft überlappende Informationsdatensätze beinhalten, in Beziehung zu setzen und zusammenzufassen. Der Fokus von Metaverzeichnissen richtet sich dabei vor allem auf die Verwaltung von Identitätsinformationen. Es gilt neben der Minderung der kostenintensiven Mehrfachpflege von Informationen auch die Verfügbarkeit der Daten weiter zu erhöhen. Dafür unterstützen Metaverzeichnisse verschiedenste Kommunikationsprotokolle. Mit Hilfe eines Metaverzeichnisses ist es einem Unternehmen auch möglich, eine aggregierte Sicht auf alle Identitätsinformationen zu erstellen und zu verwalten.

Ein Metaverzeichnis integriert Informationen aus verschiedenen Datenquellen und benutzt dabei die im zweiten Kapitel näher erläuterten Synchronisations- und Zusammenfassungsmethoden. Es bietet ein vereinheitlichtes Verzeichnis, von dem aus der Administrator die Informationen pflegen und den Datenfluss zu den angeschlossenen Verzeichnissen steuern kann. Um den Informationsfluss steuern zu können, besitzen Metaverzeichnisse alle im zweiten Kapitel betrachteten Funktionalitäten für einen Informationsabgleich.

Ein Metaverzeichnis ermöglicht nicht nur die Synchronisation zwischen vielen angeschlossenen Verzeichnissen, Datenbanken oder anderen Anwendungen, es kann auch als zentraler

³Nutzer müssen beim erstmaligen Einloggen das Passwort ändern

Anfragepunkt für Clients dienen. Es erweitert die Verzeichnisfunktionalitäten, indem es Verzeichnisoperationen über diverse Namensräume unterstützt.

Auch eine Passwortsynchronisation der Nutzerobjekte in den beteiligten Verzeichnissen wird von vielen Metaverzeichnissen unterstützt.

3.4.1 Aufbau eines Metaverzeichnisses

Metaverzeichnisse bestehen im wesentlichen aus drei Komponenten: dem eigentlichen Verzeichnis, der *Meta Engine* und den Konnektoren. Das Meta-Verzeichnis ist ein universeller Namensraum mit allen ausgewählten Einträgen oder Verweisen auf die Einträge der Quellverzeichnisse. Die Meta Engine enthält die Integrations- und Synchronisationsprozesse. Diese regeln den Austausch von Informationen durch Kontrollieren und Filtern des Informationsflusses, beinhalten das Schema Mapping und unterstützen Sicherheitsmechanismen zum Absichern der Übertragungswege zwischen den Verzeichnissen.

Bei manchen Anbietern ist die Meta Engine eine eigene Applikation (Siemens DirX-metahub), bei anderen wird der als Verzeichnisdatenbank verwendete Verzeichnisdienst für diese Funktionalitäten miteinbezogen (Novell DirXML).

Die Konnektoren (*Namespace Connectors*) kommunizieren je mit einem anderen Typen von Verzeichnis- oder Datenbanknamensraum. Ein Konnektor ist für die Kommunikation mit einem einzigen verbundenen Verzeichnis verantwortlich und übersetzt die jeweiligen Protokolle und Namensräume für den Austausch von Informationen mit dem Metaverzeichnis. Die meisten Metaverzeichnisse bringen eine große Anzahl von Konnektoren mit und unterstützen auch die Entwicklung eigener.

Metaverzeichnisse unterscheiden sich hinsichtlich ihrer Informationsintegration. Diese basiert auf den im zweiten Kapitel genannten Zusammenfassungsmethoden für die Verzeichnissynchronisation.

3.4.2 Metaverzeichnis als zentraler Informationsspeicher

Bei einem als zentraler Informationsspeicher agierenden Metaverzeichnis, werden alle ausgewählten Informationen aus den angeschlossenen Verzeichnissen in den zentralen Metaverzeichnisbaum übernommen und in der zentralen Verzeichnisdatenbank abgespeichert. Die Daten können entweder direkt aus einem entfernten Verzeichnis kopiert oder durch Umwandlungsprozesse z.B. an den Metaverzeichnis-Namensraum angepasst und dann im zentralen Verzeichnis abgelegt werden.

Für die Speicherung der Daten werden Metaobjekte angelegt, die sich über die gesammelten Objektattribute der zugrundeliegenden Verzeichnisse formen. Die Objekte des Metaverzeichnisses sind weiterhin mit denen der Quellverzeichnisse assoziiert, um Änderungen

auf beiden Seiten abgleichen zu können. Dies ist notwendig, um den Datenstand des Metaverzeichnisses aktuell halten zu können.

Bei Client-Anfragen an das Metaverzeichnis werden die Antworten direkt aus der eigenen Verzeichnisdatenbank generiert. Das Metaverzeichnis tritt somit als Datenquelle auf. Um eine hohe Datenqualität des Metaverzeichnisses zu erreichen, müssen die Informationen mit den Quelleverzeichnissen entweder ereignisgesteuert oder, wenn das nicht möglich ist, mit einer hohen Synchronisationsfrequenz abgeglichen werden.

3.4.3 Metaverzeichnis als Informationshändler

Das ausschließlich logische Zusammenführen von Daten in einem Metaverzeichnis wird als *information brokering* bezeichnet. In dem Metaverzeichnis werden bei dieser Methode nur die Verweise auf die Attribute mit ihren Werten in ihren Ursprungsverzeichnissen gespeichert und keine Daten in den zentralen Verzeichnisspeicher repliziert. Bei Client-Anfragen agiert das Metaverzeichnis als Virtual Directory (vergl. Kapitel 2) und steuert den Informationsfluss zu den angeschlossenen Verzeichnissen. Es regelt alle Zugriffe und gibt die Ergebnisse an den Client weiter, welcher folglich nur mit dem Metadirectory kommunizieren muss, um an alle Daten der fernen Verzeichnisse zu gelangen. Das Metadirectory benutzt die in seinem Verzeichnisbaum gespeicherten Verweise, um das Ursprungsverzeichnis der gesuchten Daten zu ermitteln, leitet die Anfrage an dieses weiter und gibt das zurückkommende Ergebnis der Anfrage an den Client aus.

Der Verzeichnisbaum des Metaverzeichnisses enthält für jedes Objektattribut der angeschlossenen Verzeichnisse einen Alias-Eintrag. Es werden keine weiteren Informationen abgespeichert, so daß ein Metaverzeichnis, welches mit dieser Methode arbeitet, auf die Ursprungsverzeichnisse angewiesen ist. Ein Metaverzeichnis als Informationshändler, welches auch als serverbasiertes virtuelles Verzeichnis bezeichnet wird, ist als zentraler Gateway für Clients und deren Applikationen gedacht. Es bietet eine vereinheitlichte Verzeichnissicht auf die zusammengefassten Informationen mehrerer angeschlossener Verzeichnisse und Datenbanken, welche ihrerseits verschiedene Schemata und Namensräume benutzen können.

3.4.4 Produkte

Der Markt für Metaverzeichnisse ist überschaubar und hat durch die Beteiligung von Novell und Microsoft an mehr Aufmerksamkeit gewonnen. Microsoft übernahm dafür 1999 den Hersteller Zoomit und entwickelte die Metaverzeichnislösung unter dem Namen *Microsoft Metadirectory Service* (MMS) weiter. Im Juli 2003 wurde der Nachfolger *Microsoft Identity Integration Server 2003* (MIIS) vorgestellt. In Deutschland wird dieses Produkt frühestens

im Frühjahr 2004 erhältlich sein. Im Unterschied zu MMS (Active Directory als Datenspeicher) benutzt der MIIS einen SQL-Server als Informationsspeicher. Aufgrund der noch wenig verfügbaren Dokumentationen, fehlenden Marktvergleichen und seiner von Metaverzeichnissen abweichenden Architektur, wird der MIIS in dieser Diplomarbeit nicht betrachtet.

Novell entwickelte eine eigene Lösung, DirXML. Die Entwickler bezeichnen ihre Metaverzeichnislösung lieber als „data sharing and synchronisation service“ [Nov03d], sie bietet dennoch alle Bestandteile eines Metaverzeichnisses. Als grundlegender Verzeichnisdienst wird die eDirectory (neue Version der NDS) verwendet. Für die Verwaltung können demzufolge alle Administrationswerkzeuge der eDirectory benutzt werden. DirXML benutzt für die Kommunikation zwischen Konnektoren und der eigentlichen Engine – anders als alle anderen Hersteller – *eXtensible Markup Language* (XML)-Dokumente als Austauschformat für Informationen. Mit Einführung der neuen Netware 6.5 Serverbetriebssysteme von Novell, ist DirXML als kostenloses Starterpaket im Verzeichnisdienst enthalten.

Eigenschaften	Siemens DirXmetahub	Sun ONE Meta Directory	Microsoft MMS	CriticalPath Metadirectory	Novell DirXML
Verzeichnisse	LDAP basiert, DirX	Sun ONE Dir	Active Dir	ADS, eDirectory, CP Dir, IBM Dir, Sun ONE Dir	eDirectory
Join	•	•	•	•	•
Server- plattformen					
Windows 200x	•	○	•	•	•
Windows NT	•	•	○	○	•
Netware	○	○	○	○	•
Solaris	•	•	○	•	•
HP-UX	•	○	○	•	•
Linux	•	○	○	○	•
Administration					
Webinterface	○	○	•	○	•
Java GUI	○	•	○	○	•
GUI	•	-	•	•	•
Ereignismgmt					
Abfrage	•	•	•	•	•
Systemereignisse	•	•	○	•	•

Tabelle 3.1: Metaverzeichnisse im Vergleich

Ein weiterer Vertreter ist der Sun ONE Meta Directory Server, welcher als Nachfolger

des iPlanet/Netscape Directory Server gilt. Sun bietet, im Gegensatz zu den anderen Vertretern, nur wenige Konnektoren für etwaigige Zielsysteme an. In erster Linie werden die beiden „Direct Connectors“ [Sun03a] zum Anbinden von Datenquellen verwendet. Diese kommunizieren zum einen über LDAP und zum anderen über SQL⁴.

Siemens kann ebenso wie Critical Path auf einen großen Erfahrungsschatz auf dem Gebiet der Metaverzeichnislösungen verweisen.

Eine marktführende Stellung wird Critical Path zugesprochen, wobei für beide [CP und Siemens] Hersteller gilt, dass sie sich durch umfassende Lösungskompetenz auszeichnen. [Hol02]

Konnektoren	Siemens DirXmetahub	Sun ONE Meta Directory	Microsoft MMS	CriticalPath Metadirectory	Novell DirXML
Active Dir	•	•	•	•	•
Win NT	•	•	•	•	•
eDirectory	•	○	•	•	•
Unix	•	○	•	•	•
Linux	•	○	•	○	•
HP UX	•	○	•	○	•
Free BSD	•	○	•	•	•
Solaris	•	•	•	•	•
Exchange	•	○	•	•	•
Groupwise	•	○	•	•	•
Lotus Notes	•	○	•	•	•
SAP R/3	•	○	•	•	•
Peoplesoft	•	○	•	•	•
Oracle	•	•	•	•	•
Informix	•	○	•	•	•
Sybase	•	○	•	•	•
MS SQL	•	○	•	•	•
Cisco ACS	•	○	•	○	•
RADIUS-Server	•	○	•	•	•
HICOM (TK)	•	○	•	•	•
LDAP	•	•	•	•	•
SQL	•	•	•	•	•
ODBC	•	•	•	•	•
.csv Dateien	•	•	•	•	•
XML Interface	•	○	•	•	•
DSML Interface	•	○	•	-	• (extra)
LDIF Interface	•	○	•	•	•
weitere	-	univ. Text	Agent Toolkit	univ. Perl / Java	DriverKit

Tabelle 3.2: Konnektoren der Metaverzeichnisse (Auswahl)

Die Tabelle 3.1 auf der vorherigen Seite vergleicht die wichtigsten momentan am Markt vertretenen Produkte. Für eine spätere Implementierung sind dabei vor allem die Server-

⁴Structured Query Language

plattformen und die als Metaverzeichnis unterstützten Verzeichnisdienste von Interesse. Eine Übersicht der wichtigsten verfügbaren Konnektoren ⁵ stellt die Tabelle 3.2 auf der vorherigen Seite dar.

Der Radiant Logic RadiantOne VDS Server sei an dieser Stelle als ein Vertreter der Informationshändler erwähnt. Er setzt die Verlinkung von Datenquelleninhalten auf Attributebene in einem Metaverzeichnis um. Er kann auf den Windows Plattformen installiert werden und nutzt LDAP-basierte Verzeichnisse, um seine Verweise zu speichern. Die bereitgestellten Konnektoren berücksichtigen im besonderen die üblichen Datenbankprotokolle, ist er doch als LDAP-Gateway für Datenbanksysteme gedacht.

3.4.5 Zusammenfassung

Abschließend lassen sich die Vorteile der Metaverzeichnisse hervorheben. Die für eine Synchronisation notwendigen Änderungsdienste sind implementiert, so dass sich eine Synchronisation zwischen zwei, vor allem aber zwischen mehreren Datenquellen in kurzer Zeit realisieren lassen. Sie beinhalten komplexe Funktionalitäten, wie uni- und bidirektionale Synchronisationsmethoden, Änderungserkennungsmechanismen, Unterstützung von Änderungsgranularitäten auf der Attributebene und auch Passwortabgleichmechanismen. Die LDAP Unterstützung ist bei Metaverzeichnissen schon selbstverständlich, eine Unterstützung von DSML (siehe Abschnitt DSML) auch bei fast allen verfügbar.

Konnektoren für die gängigen Zielsysteme sind, mit Ausnahme von Sun, vorhanden, eine Entwicklung von eigenen Treibern ist bei allen Metaverzeichnissen, bis auf Siemens (durch externe Berater), möglich. Bei den, für eine Auswahl interessanten, erfolgreichen Implementationsbeispielen haben Critical Path und Siemens die Nase vorn, aber auch Novell DirXML ist schon in großen Projekten umgesetzt worden.

Die Abbildung 3.2 auf der nächsten Seite zeigt die verschiedenen Hersteller von Metaverzeichnissen. Für den Vergleich der Systeme benutzt Gartner folgende Bewertungskriterien (aus [Enc03]):

- Marketing and product strategies
- Third-Party software market acceptance
- Packaging, branding and Distribution
- Architectural flexibility
- Product reliability and scalability

⁵Weiterführender Artikel siehe [Sti03]

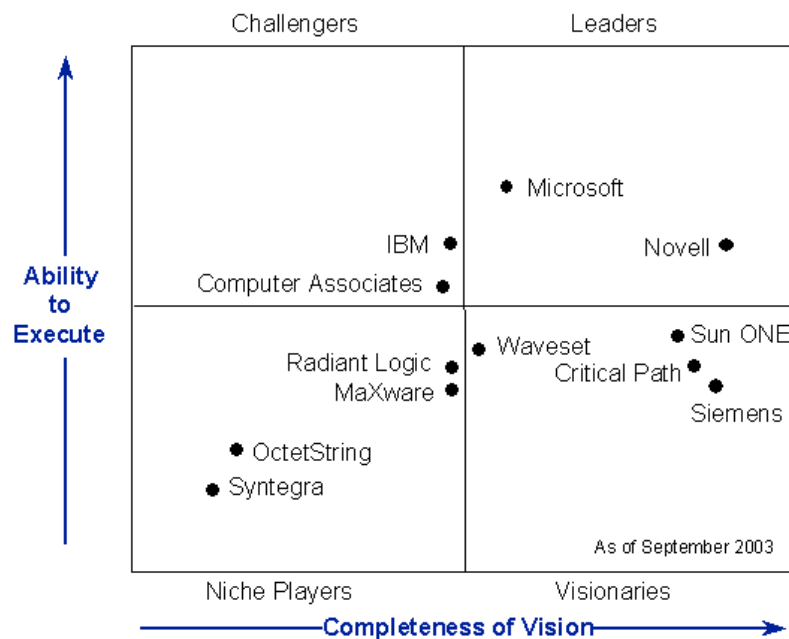


Abbildung 3.2: Magic Quadrant for Metadirectory Products, 2H03 [Enc03]

- Investment resources and commitment

Gartner sieht das Novell DirXML Produkt als Marktführer und das schon seit zwei Jahren. Microsoft konnte in diesem Jahr durch die Erneuerung seines Metaverzeichnisproduktes in den „Leader“-Quadranten aufschliessen.

3.5 Directory Services Markup Language – die Zukunft?

Ende 1999 wurden die erste Version (DSML 1.0) eines neuen Verzeichnisdatenaustauschformats fertiggestellt. Dieser relativ neue Standard mit dem Namen *Directory Services Markup Language* (DSML) wurde von Bowstreet unter Mitwirkung von IBM, Microsoft, Novell, Oracle, iPlanet, Critical Path und weiteren entwickelt. DSML 1.0 wurde bei verschiedenen Standardisierungsgremien, darunter auch bei der *Organization for the Advancement of Structured Information Standards* (OASIS)⁶, eingereicht und ist heute als DSML 2.0 Spezifikation verfügbar.

DSML 1.0 describes directory contents in XML. DSML 2.0 will give companies standardized XML format to access directory information wherever it exists on the Internet and provide a common XML description for manipulation of directory information such as queries, comparisons, updates, additions

⁶<http://www.oasis-open.org/>

and deletions. XML-based applications will consume the DSML information in business webs. [Bow00]

Fast alle Verzeichnisdiensthersteller haben eine Erweiterung für DSML 2.0 für ihre Produkte entwickelt, einige Analysten sprechen sogar schon von dem Nachfolger von LDIF als Verzeichnisaustauschformat.

DSML Code Beispiel

```
<dsml:entry dn="cn=taito,o=FHTW">
  <dsml:objectclass>
    <dsml:oc-value>top</dsml:oc-value>
    <dsml:oc-value>person</dsml:oc-value>
  </dsml:objectclass>
  <dsml:attr name="cn">
    <dsml:value>taito</dsml:value>
  </dsml:attr>
  <dsml:attr name="sn">
    <dsml:value>radtke</dsml:value>
  </dsml:attr>
  <dsml:attr name="uid">
    <dsml:value>taito</dsml:value>
  </dsml:attr>
  <dsml:attr name="messageServer">
    <dsml:value>cn=HRZT1,ou=HRZ,o=FHTW</dsml:value>
  </dsml:attr>
</dsml:entry>
```

DSML benutzt die *eXtensible Markup Language* (XML)⁷ zum Beschreiben der Verzeichnisisinformationen. Mit Hilfe von „Document Type Definitions“ (DTD) ist es möglich die Vielzahl der Schemarepräsentationen von Verzeichnisdaten ineinander zu überführen.

DSML Leverages LDAP: LDAP provides a means for accessing directory information. DSML provides the means for reading and understanding directory content. Today, without the DSML standard, developers who want to use directory policy and profile information must write code for each directory they want to support. Many use LDAP access protocols to generate LDIF or XML documents that can drive customization within applications such as electronic

⁷<http://www.w3.org/XML/>

commerce. But such proprietary solutions impede inter-company integration, making it expensive and time-consuming to build and manage such systems as supply chain management and distribution channel management. With a DSML standard, any XML-based application will be able to leverage directory information expressed as XML.[Oas02]

Für die Übertragung der DSML-Dateien werden in vielen Implementationen *Simple Object Access Protocol* (SOAP)⁸-Nachrichten und als Übertragungsprotokoll HTTP verwendet. DSML-Daten können damit über das Internet auf Port 80 „firewallfreundlicher“ zu anderen DSML-Applikationen übertragen werden.

DSML ist vielversprechend. Die Vorteile gegenüber dem LDIF-Format sind enorm, ist es doch nun einfacher möglich, die Verzeichnisdaten mit Hilfe von XML-Stylesheets und DTDs in andere Formate zu überführen. Novell hat seine DirXML Lösung, welche ebenfalls mit XML Standards arbeitet, den Entwicklern von DSML zur Verfügung gestellt.

3.6 Zusammenfassender Vergleich der Lösungsansätze

Eine Lösung für fast alle Probleme der Verzeichnissynchronisation bieten nur Metaverzeichnisse. Sie sind grosse Softwarepakete, welche für jede Form von Synchronisation zwischen Verzeichnisdiensten, Datenbanken und anderen Anwendungen geeignet sind. Metaverzeichnisse erfüllen alle Anforderungskriterien, wenn auch teilweise mit Hilfe weiterer Module (z.B. Passwortsynchronisationserweiterungen). Sie bieten viele erweiterte Funktionalitäten, die über eine Lösung einfacher Synchronisationsprobleme hinausreichen. So bieten einige Produkte eigene Workflowverwaltungswerkzeuge an, um möglichst alle Probleme eines Nutzermanagements in großen Unternehmen abzudecken.

Den grössten Vorteil gegenüber allen anderen Lösungsansätzen stellt die fertige Implementation der benötigten Funktionalitäten dar. Metaverzeichnisse müssen für einen Einsatz „nur“ noch installiert und konfiguriert werden.

Allerdings sind Metaverzeichnisse vor allem für die Synchronisation mehrerer Datenquellen ausgelegt. Ihre Anschaffung ist meist mit hohen Kosten verbunden und eine Schulung der Administratoren ist aufgrund ihrer Komplexität angebracht. Bei zunehmender Zahl angeschlossener Datenquellen steigt jedoch die Ersparnis der Entwicklungszeit und somit auch der Kosten im Gegensatz zu anderen Lösungen.

Wie auch bei den anderen Lösungsansätzen ist eine detaillierte Auseinandersetzung mit den zu synchronisierenden Quellen nötig. Ein Metaverzeichnis kann nur so gut arbeiten, wie es auf die einzelnen Systeme abgestimmt wurde. Des Weiteren benötigen Metaverzeichnisse

⁸<http://www.w3.org/TR/SOAP/>

ein zentrales Verzeichnis zum Abgleichen der Informationen. Diese Aufgabe kann jedoch im besten Falle durch ein vorhandenes Verzeichnis übernommen werden.

Ein selbstprogrammiertes Synchronisationssystem kann meist nur einen Teil der Anforderungen erfüllen. Zu komplex sind die einzelnen Probleme, so z.B. die Implementierung eines bidirektionalen Synchronisationsvorgangs oder einer Passwortsynchronisation. Auch die Entwicklung einer Administrationsoberfläche und die Anbindung von Anwendungen mit eigenen Kommunikationsprotokollen stellt eine enorme Herausforderung dar.

Eine bessere Alternative bietet der Austausch von Verzeichnisinformationen über LDIF. Zwar ist hier nur eine Synchronisation zwischen LDAP-fähigen Verzeichnissen möglich, dennoch bietet Novell mit seinem ICE-Werkzeug Möglichkeiten für ein, über XML anpassbares, Informationsmanagement an. Ein Nachteil der LDIF-basierten Lösungen ist jedoch das fehlende Ereignismanagement. Für eine bessere Handhabung müsste eine Administrationsoberfläche implementiert werden. Die direkte Synchronisation von Passwörtern ist nicht möglich.

DSML bietet alle Funktionalitäten einer LDIF-Schnittstelle verbunden mit den Vorteilen einer XML-Dokumentenverarbeitung. Durch die Abbildung der Verzeichnisschemata in DTD's ist eine Formatanpassung und Datentransformation ohne programmiertechnischen Aufwand möglich. Aufgrund des offenen Ansatzes wird sich diese Möglichkeit zum Synchronisieren von Verzeichnisinformationen schnell verbreiten. Jedoch erfordert eine DSML-Implementierung weitere Komponenten (z.B. Web-Services), welche auch administriert werden müssen.

Die Auswahl einer Synchronisationslösung für den Abgleich von ADS und NDS hängt also stark von den gegebenen Anforderungen ab. Die in dieser Diplomarbeit erarbeiteten Anforderungen werden momentan nur von Metaverzeichnissen erfüllt. Wenn allerdings auf eine Passwortsynchronisation verzichtet werden kann, in absehbarer Zeit keine weitere Synchronisation mit anderen Verzeichnisdiensten geplant ist und auch die Anforderungen an die Informationsaktualität nicht groß sind, können Synchronisationsmechanismen auf Basis von LDIF-Dateien eine sinnvolle Alternative zu den Metaverzeichnissen darstellen. Diese kann sowohl über selbstgeschriebene Programme als auch mit Hilfe der LDIF-Programme der Verzeichnisdiensthersteller implementiert werden.

Abzuwarten bleibt auch, wie sich DSML im Verzeichnisumfeld durchsetzen wird. Beide Verzeichnisdienste bieten schon jetzt eine Unterstützung für den DSML2 Standard an. Dabei sollte jedoch nicht unerwähnt bleiben, dass dafür auf Windows-basierten Serversys-

temen (ADS) der durch zahlreiche Sicherheitslücken bekannte Internet Information Server benötigt wird.

4 Nutzerdatenabgleich zwischen NDS und ADS am Beispiel FHTW

4.1 Ausgangslage	54
4.2 Zielstellung	57
4.3 Lösungskonzept	58
4.3.1 Wahl des zentralen Verzeichnisses und der Synchronisationslösung	58
4.3.2 DirXML als Synchronisationslösung	60
4.3.3 ADS Anforderungen	66
4.3.4 Attributzuweisung zwischen NDS und ADS	68
4.3.5 Synchronisationsfrequenz	69
4.3.6 Auswahl der zu synchronisierenden Objekte	70
4.3.7 Performanceanpassungen	70

In diesem Kapitel wird das Konzept eines Synchronisationsmechanismus zwischen NDS und ADS am Beispiel der FHTW Berlin diskutiert. Dafür wird zunächst die Ausgangslage betrachtet, um an Hand abgeleiteter Zielkriterien ein Lösungskonzept zu entwickeln.

4.1 Ausgangslage

An der FHTW existiert ein campusweites Computernetzwerk. Daran sind Computerlabore der Fachbereiche und zentraler Einrichtungen angeschlossen. Die meisten, jetzt angeschlossenen Computerlabore, sind jedoch vor dem Zusammenschluss dieses Computernetzwerkes entstanden und benutzen die in der Zeit vor dem Zusammenschluss entstandenen eigenen, unabhängigen Accountverwaltungssysteme. Studenten der FHTW wollen und müssen die IT-Einrichtungen der FHTW zunehmend strukturübergreifend nutzen und werden dabei durch die Vielzahl von Einzellösungen im besonderen bei der Accountverwaltung behindert.

Das Hochschulrechenzentrum (HRZ), Betreiber der zentralen Infrastruktur des FHTW-Netzes, stellt einige Dienste zentral bereit. Dazu gehören neben einem Mailservice, Einwahl-

Diensten, Zugang zum Wireless-LAN und weiteren Diensten auch das Betreiben eigener PC-Säle und die Betreuung ausgewählter Labore anderer Fachbereiche.

Für eine Benutzung dieser Dienste und Umgebungen werden vom HRZ eingerichtete Logins benötigt. Im HRZ selbst gibt es zwei Plattformen mit jeweils eigenständiger Accountverwaltung. Zum einen wird eine NIS-Domäne betrieben, welche die Authentifizierung für alle Unix-basierten Systeme übernimmt, zum anderen ein Novell-Verzeichnisdienst für die Verwaltung und Authentifizierung der Windows-basierten PCs. Die NIS-Domäne wird zur Zeit durch einen LDAP-basierten Verzeichnisdienst (OpenLDAP) abgelöst. Alle Studenten der Fachhochschule können Logins beantragen und die Rechner und Dienste des Rechenzentrums benutzen.

Der Fachbereich 1 der FHTW benutzt zur Administration seiner Windows- und auch Unix-Clients das Active Directory von Microsoft. Damit wird der Zugriff für Professoren, Dozenten, Mitarbeiter und Studenten gesteuert. Nutzeraccounts benötigen vor allem Studenten des Fachbereiches, um an rechnergestützten Vorlesungen und Laboren teilnehmen zu können.

Die Administration der beiden Verzeichnisdienste ADS (FB1) und NDS (HRZ) ist organisatorisch und personell voneinander unabhängig und soll auch in Zukunft nicht zentralisiert werden. Probleme bereitet die aus der getrennten Administration hervorgehende unterschiedliche Entwicklung der Accountdaten im Rechenzentrum und im Fachbereich 1 der FHTW.

Um die Problematik der getrennten Verwaltung von Accountdaten aufzuzeigen, wird im Weiteren der organisatorische Ablauf der Logineinrichtung geschildert.

Im Hochschulrechenzentrum der FHTW existiert für die Einrichtung von Logindatensätzen eine Logindatenbank. Die Logindatensätze in dieser zentralen Datenbank wurden in der Vergangenheit mit Hilfe von schriftlichen Anträgen der Nutzer initiiert und über eine Abfrage auf eine gefilterte Ansicht der Immatrikulationsdatenbank vor dem Anlegen in der Logindatenbank geprüft. Seit Anfang dieses Semesters wird für jeden neu immatrikulierten Studenten der FHTW automatisch ein Logindatensatz angelegt, wodurch ein schriftlicher Antrag entfällt.

Die von der Datenbank in eine Textdatei exportierten Login-Datensätze werden über die UIMPORT¹-Schnittstelle in die NDS importiert. Danach sind die neuen Logins einsatzbereit und können, nach Ausgabe einer Loginbestätigung in schriftlicher Form, von den Studenten benutzt werden. Die Loginbestätigung ist momentan noch notwendig, um dem

¹NDS-spezifische Schnittstelle zum Importieren von Verzeichnisisinformationen

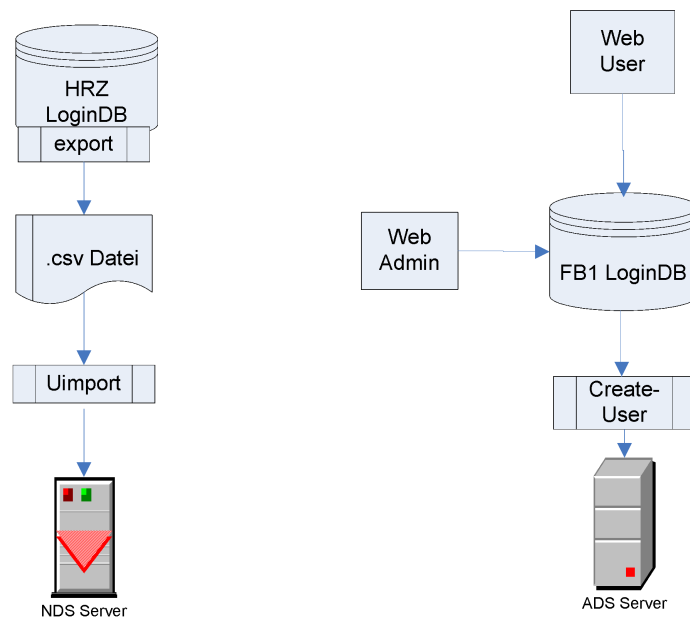


Abbildung 4.1: Einrichtung von Nutzerkennungen im HRZ und FB1

Benutzer seinen Loginnamen und das Passwort für die erstellten Accounts (Novell- und Unixaccount) mitzuteilen.

Die Verwaltung der Logindaten des Active Directory im Fachbereich 1 wird über eine dort existierende Datenbank gesteuert. Es existiert ein Webinterface, welches für lokale (im Fachbereich) Studenten, Professoren, Mitarbeiter, Dozenten und studentische Hilfskräfte gedacht ist. Dieses Webinterface ist nötig, um einen Account im Fachbereich beantragen zu können.

Alle Datensätze, welche vom Webinterface in der Fachbereichsdatenbank erzeugt wurden, werden von dem ADS-Administrator über ein gesondertes Webinterface überprüft und danach für den Import in das Active Directory gekennzeichnet. Für den eigentlichen Import der Accountdaten von der Fachbereichsdatenbank in die ADS wird eine ADSI-basierte Routine verwendet. Danach sind die eingerichteten Logins verfügbar und können nach Erhalt einer Loginbestätigung des Fachbereiches verwendet werden.

Die Accountdaten beider Verzeichnisdienste (NDS im HRZ und ADS im FB1) sind nach der Einrichtung vollkommen unabhängig voneinander. Die Loginnamen der Studentenaccounts sind aber in beiden Verzeichnissen identisch. Damit sollte eine Vereinfachung für die Studenten geschaffen werden, welche sich nur noch einen Loginnamen merken müssen. Das führt sowohl bei Studenten als auch bei den Administratoren zu Missverständnissen. Auf

Seiten der Studenten entsteht der Eindruck eines einzigen Accounts mit identischem Passwort. Auf der administrativen Seite gestaltet sich eine Hilfe bei Problemen mit Accountdaten schwierig, da die Benutzer den Geltungsbereich ihrer Accountdaten nicht unterscheiden können. Dadurch entsteht eine erhöhte Anzahl eigentlich unnötiger Helpdesk-Anfragen, was sich wiederum nachteilig auf die Arbeit der Administratoren auswirkt.

4.2 Zielstellung

Das Ziel soll im Allgemeinen ein Modell einer „sanften“ Migration in eine FHTW-weite gemeinsame Nutzer-Verwaltung/Authentifizierungsinfrastruktur am Beispiel der Verbindung eines Fachbereiches an das HRZ sein. Auf die Beibehaltung der unabhängigen Administrationmöglichkeit wird besonderer Wert gelegt.

Das Ziel eines Abgleiches der beiden Verzeichnisdienste NDS und ADS im speziellen ist die Vereinfachung der Verwaltung von Studentenaccountdaten. Ausserdem soll für die Studenten ein einheitlicher Account für beide Systemwelten angestrebt werden.

Die Ziele im Einzelnen:

- **Z1:** Wahl eines zentralen Verzeichnisses und Einrichtung einer offenen Synchronisationstechnik für eine spätere Anbindung weiterer Verzeichnisdienste.
- **Z2:** Entlastung der Administration durch Zusammenfassung und zentralisiertes Management der studentischen Accountdaten mit automatisierten Einrichtungsabläufen.
- **Z3:** Sicherung der bestehenden Funktionalität beim Management von nichtstudentischen Accountdaten.
- **Z4:** Gesicherter Austausch der Nutzerdaten zwischen den Verzeichnissen.
- **Z5:** Bidirektionale Synchronisation des Passwortes als Vereinfachung für Nutzer.

Um die Zielstellung erfüllen zu können, bedarf es einiger Abgrenzungskriterien:

- **K1:** Es werden keine Gruppenzugehörigkeiten **synchronisiert**, die Benutzung von Gruppen erfolgt in beiden Systemen mit unterschiedlicher Zielstellung.
- **K2:** Für die Synchronisation von Nutzerdaten werden nur Studentenaccounts berücksichtigt. Sie besitzen in beiden Systemen eine einheitliche Kennung (Matrikelnummer).
- **K3:** Studentische Nutzerstammdaten dürfen nur über die NDS (autoritäre Datenquelle) geändert werden, da diese ihre Daten von der RZ-Logindatenbank bekommt,

welche wiederum ihren Datenstamm aus der eigentlichen Datenquelle – der Immatrikulationsdatenbank – bezieht.

- **K4:** Die in der Active Directory des FB1 für die Verwaltung von Unix-basierten Clients benutzten Attribute werden in diesem Lösungskonzept nicht berücksichtigt.
- **K5:** Betriebsattribute der ADS, welche für einen Betrieb eines ADS-Accounts notwendig sind, werden nicht über den Abgleichmechanismus erzeugt.

4.3 Lösungskonzept

4.3.1 Wahl des zentralen Verzeichnisses und der Synchronisationslösung

Für die Wahl eines zentralen Verzeichnisses gemäss des ersten Zielkriteriums (**Z1**) ist der Nutzerdatenumfang der Verzeichnisse ausschlaggebend. Da in den Novell-Verzeichnisdienst des RZ's Accounts für alle Studenten aller Fachbereiche angelegt werden, beinhaltet er auch Nutzerdaten der Studenten des Fachbereiches 1. Das Active Directory des Fachbereiches wiederum verwaltet parallel nur studentische Nutzerdaten des Fachbereiches. Ein weiterer Faktor ist die organisatorische Nähe sowie die direkte Kopplung zwischen den RZ-NDS und der RZ-Logindatenbank. Das zentrale Verzeichnis, über dem die Informationen der Logindatenbank synchronisiert werden, ist somit das Novell-Verzeichnis des Rechenzentrums.

Aus der Betrachtung der Lösungsansätze für die Synchronisation der Verzeichnisdienste ergaben sich Vorteile durch die Benutzung der Metaverzeichnislösungen. Für die Auswahl einer Metaverzeichnislösung ist die Frage der Unterstützung der bestehenden Verzeichnisse ausschlaggebend. Soweit es möglich erscheint, sollte eine zusätzliche Implementation eines neuen Verzeichnisses vermieden und die Integration der bestehenden angestrebt werden. Durch die Auswahl des NDS-Verzeichnisses als zentrales Verzeichnis werden nur zwei Produkte der im dritten Kapitel vorgestellten Metaverzeichnisse weiter betrachtet. Das wäre zum einen das Critical PATH Metadirectory und Novells DirXML, die als einzige im Metaverzeichnisvergleich die NDS als zentrales Verzeichnis unterstützen.

Als weiteres Auswahlkriterium wird die Einfachheit des Aufbaus und der Kostenaufwand durch externe Berater hinzugenommen. Novell DirXML wird im Gegensatz zu dem anderen Vertreter ohne spezielle Berater vertrieben, was wiederum für eine einfachere Implementation einer Metaverzeichnislösung auf Basis von DirXML spricht. Weiterhin sind DirXML Dokumentationen frei verfügbar und sehr ausführlich. Die erforderlichen Kenntnisse für eine Implementierung bauen stark auf dem Verzeichnismangement des zugrunde liegenden Verzeichnisdienstes (NDS) auf. DirXML bietet weiterhin ausreichend Konnektoren zur Anbindung weiterer bestehender Verzeichnisdienste (z.B. NIS oder LDAP) an.

Durch die Wahl des zentralen Verzeichnisses und der Synchronisationstechnik ergibt sich ein neuer Ablauf der Accounteinrichtung für Studenten. Dieser Ablauf wird im Bild 4.2 dargestellt.

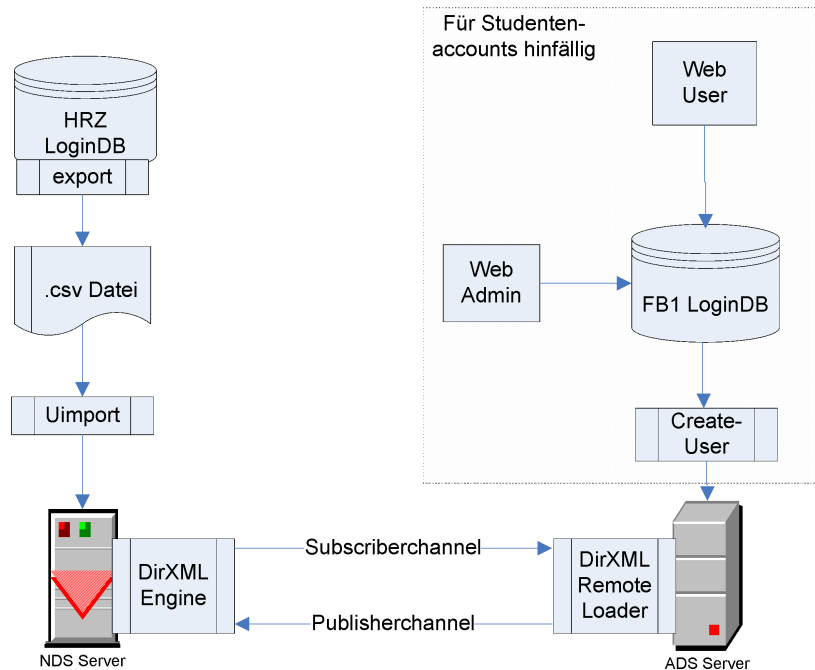


Abbildung 4.2: Vereinfachung des Nutzermanagements mit Hilfe von DirXML

Die Verwaltung des Lebenszyklus eines Studentenaccounts (Einrichtung, Modifikation und Löschung) wird in diesem Lösungskonzept komplett vom zentralen Verzeichnis und seinem Synchronisationsmechanismus (DirXML) gesteuert. Die Administratoren des Fachbereiches werden entlastet, Zielkriterium **Z2**.

Da das Lösungskonzept in dieser Diplomarbeit in erster Linie Studentenaccounts berücksichtigt, muss auch eine Möglichkeit zur Verwaltung von zusätzlichen Accounts im Active Directory gemäß dem Zielkriterium **Z3** gesichert sein. Die Einrichtung und Verwaltung der nichtstudentischen Datensätze kann weiterhin mit Hilfe des Webinterfaces und der Fachbereichsdatenbank erfolgen.

DirXML unterstützt einen sicheren Datenaustausch zwischen den Verzeichnisdiensten (**Z4**) durch Verschlüsselung der Kommunikationswege mit Hilfe von SSL. Mit Hilfe des im DirXML-Softwarepaket enthaltenen *PasswordSync*-Programms kann ein bidirektionaler Passwortabgleich zwischen den beiden Verzeichnissen gemäß **Z5** realisiert werden.

Die Zielkriterien werden von dem dargestellten Lösungskonzept vollständig erfüllt. Für dessen detaillierte Erarbeitung werden zunächst das Produkt selbst und danach die Daten-

abhängigkeiten betrachtet. Aus den Datenabhängigkeiten lassen sich dann Regeln für eine spätere Implementierung ableiten.

4.3.2 DirXML als Synchronisationslösung

DirXML ist zur Zeit in der Version 1.1a verfügbar und soll im Frühjahr 2004 durch eine neue Version mit dem Namen *Nsure Identity Manager 2* abgelöst werden. In der folgenden Übersicht wird der Marketingname eDirectory statt des sonst in dieser Diplomarbeit benutzten Namens NDS für den Novell Verzeichnisdienst verwendet. Das soll zum Einen dem Leser den Umgang mit DirXML Dokumenten (dort wird der Begriff eDirectory verwendet) erleichtern, zum Anderen die Versionsabhängigkeit des Produktes darstellen.

Architektur

Die DirXML Technologie besteht aus drei wesentlichen Komponenten. Dazu gehören das eDirectory (NDS), die DirXML Engine und das DirXML Driver Shim. Hinzu kommt noch der Remote Loader, welcher eine räumliche Trennung von Driver Shim und Driver Engine ermöglicht.

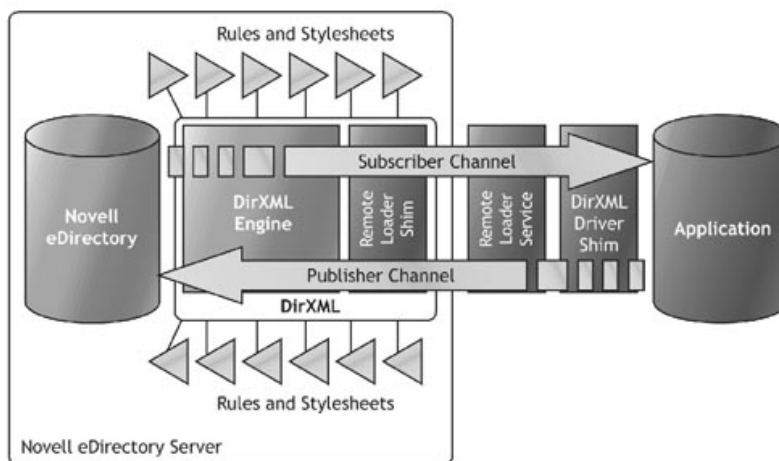


Abbildung 4.3: DirXML Architektur mit Remote Loader [Nov02]

- **Novell eDirectory**

Der Verzeichnisdienst agiert als Datenspeicher einer DirXML Installation und stellt somit das zentrale Verzeichnis einer „Hub-and-Spoke“-Architektur dar. Synchronisationsvorgänge der angeschlossenen Datenquellen laufen über diesen „Hub“. Jede der angeschlossenen Datenquellen kann mit einer beliebigen Anzahl anderer angeschlossener Datenspeicher synchronisiert werden.

Das eDirectory wird von der DirXML-Installation auch als Speicherplatz für Konfigurationsdaten und für die Unterstützung von Zugriffskontrollen verwendet. Dafür wird das Schema des Verzeichnisdienstes erweitert und Klassendefinitionen für Konfigurationsobjekte hinzugefügt. Weiterhin wird das Verzeichnis so erweitert, dass jedes Objekt ein *DirXML association*-Attribut erhält. Es ist ein Multiwertattribut und kann die korrespondierenden Attribute der zu synchronisierenden Datenquellen abspeichern. Dadurch können eindeutige Abgleichattribute, welche normalerweise bei beiden Synchronisationspartnern vorhanden sein müssten, entfallen.

- **DirXML Engine**

Die DirXML Engine ist die Software, welche die Synchronisation von Informationen zwischen dem eDirectory und den angeschlossenen Datenquellen steuert. Diese Engine kann auf einem oder mehreren Servern eines eDirectory Verzeichnisbaumes installiert werden und einen oder mehrere Treiber beinhalten. Die Synchronisation zwischen DirXML und den Datenquellen kann uni- oder bidirektional durchgeführt werden. Dafür werden zwei gerichtete Datenkanäle verwendet. Der *Publisher*-Kanal überträgt die Daten der Datenquellen zur eDirectory und der *Subscriber*-Kanal steuert den Datenfluss von der eDirectory zu den Datenquellen. Der Datenfluss wird kanalabhängig über Filter und Regeln gesteuert. Filter beinhalten eine Liste von Objektklassen und Attributen, welche im Verarbeitungsprozess von der DirXML-Engine berücksichtigt werden. Regeln sind die eigentlichen Transformationen der XML-Dokumente, welche von der DirXML Engine ausgeführt werden.

- **DirXML Driver Shim**

Der Driver Shim ist ein Programm, welches zur Kommunikation mit den angeschlossenen Datenquellen verwendet wird. Es kann in Java oder C++ geschrieben sein und kommuniziert auf der einen Seite mit der jeweiligen Datenquelle verständlichen Schnittstellenmethoden, auf der anderen Seite mit der DirXML Engine über XML-Dokumente mit Parametern. Je nach Ereignismanagement der Datenquelle (ereignis- oder zeitgesteuert) werden die Änderungsereignisse über den Driver Shim an die DirXML Engine übertragen.

- **Remote Loader**

Der Remote Loader ermöglicht eine getrennte Platzierung des Driver Shim, unabhängig von der DirXML Engine oder eDirectory Verzeichnisservern. Er besteht aus zwei Komponenten, dem *Remote Loader Shim* und dem *Remote Loader Service*. Der Remote Loader Shim ist Java-basiert und verbindet die DirXML Engine mit dem abgesetzten Driver Shim über den Remote Loader Service. Dieser Remote Loader Service läuft als Dienst oder Daemon auf dem Zielsystem, auf welchem sich die zu verbindenden

de Datenquelle befindet. Der Remote Loader Service und der Remote Loader Shim können über eine SSL-verschlüsselte TCP/IP-Verbindung miteinander kommunizieren.

Regeln und Filter in DirXML

Die in DirXML verfügbaren Regeln wandeln die Ereignisse des Einganges eines Kanals (Publisher oder Subscriber) in einen Satz von Befehlen für den Ausgang des Kanals um. Sie sind von einem Systemadministrator über eDirectory-Objekte konfigurierbar und können an die individuellen Anforderungen beliebig angepasst werden. Ausgeführt werden sie in der DirXML Engine. Alle Regeln können mit Hilfe von XSLT-Stylesheets implementiert werden. Sie sind jeweils getrennt für den Publisher- oder Subscriber-Kanal konfigurierbar.

DirXML uses XSLT to convert data and apply rules between the application, database, or directory and eDirectory. XDS is the XML vocabulary used by DirXML that can be transformed by a style sheet. [...] All rules can be implemented using XSLT style sheets, but rules that perform well-defined roles more commonly use an XML format specific to DirXML that more easily describes the transformation needed. [Nov03a, S.21]

Eine Übersicht der in DirXML verwendeten Regeln und Stylesheets ist in der Abbildung 4.4 dargestellt.

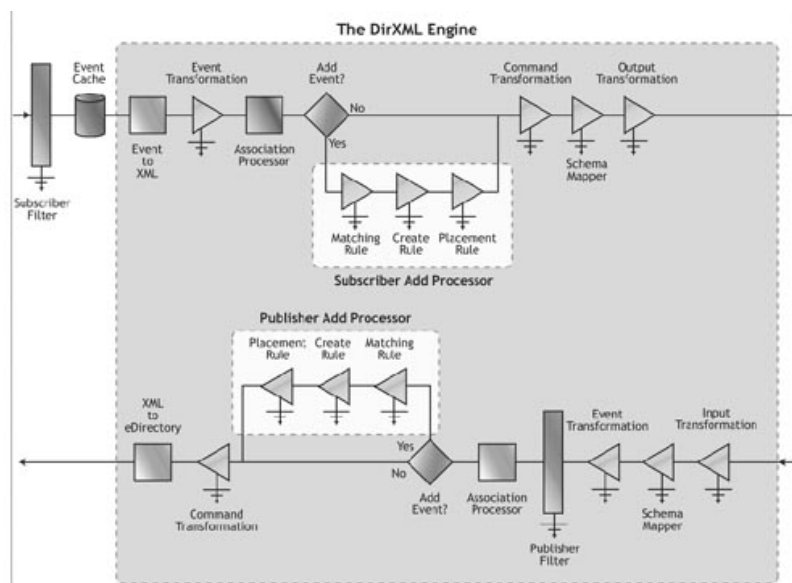


Abbildung 4.4: DirXML Engine mit ihren Regelsätzen [Nov02]

Es existieren folgendende Regeln zum Anpassen eines Datenflusses:

- **Schema Mapping** Eine XML-basierte Regel, welche eins-zu-eins Verbindungen von Objektklassen und Attributen zwischen dem eDirectory und den Datenquellen herstellt.
- **Object Matching** Eine XML-basierte Regel, welche die Kriterien für eine Assoziation zwischen eDirectory Objekten und den bestehenden Objekten der verbundenen Datenquelle definiert.
- **Object Create** Eine XML-basierte Regel, welche eine Definition der zu erfüllenden Anforderungen enthält, bevor ein Objekt erstellt werden kann.
- **Object Placement** Eine XML-basierte Regel, welche die Kriterien zum Platzieren eines neu zu erstellenden Objektes bestimmt.
- **Event Transformation** Ein XSLT-Stylesheet, in dem definiert wird, wie verschiedene Ereignisse des Eingangs in Aktionen des Ausgangs umzuwandeln sind.
- **Command Transformation** Mit diesem XSLT-Stylesheet können Aktionen umgewandelt oder verändert werden.
- **Data Transformation** Dieser XSLT-Stylesheet steuert die Formatumwandlung der in den XML-Dokumenten vorhandenen Objektklassen und Attribute.

Filter legen die Objektklassen und deren zugehörige Attribute fest, die für die DirXML-Verarbeitung berücksichtigt werden. Nur Ereignisse, welche im Zusammenhang mit den im Filter konfigurierten Objektklassen und Attribute stehen, können den Filter passieren. Damit helfen Sie die Last der DirXML-Engine zu verringern und Dateigentümeranforderungen umzusetzen. Sie werden getrennt nach Publisher- und Subscriber-kanal konfiguriert. Sie sind allerdings nicht in der Lage, Attributwerte auszuwerten und somit eine Selektion von Accountdaten zu steuern.

Weitere Funktionalitäten

DirXML stellt weitere wichtige Funktionalitäten gemäß dem Anforderungsprofil von Abgleichsystemen bereit. Dazu gehört eine Fehlerverarbeitung, welche ihre Logdateien auf die Komponenten aufteilt und so eine Fehlersuche erheblich erleichtert. Außerdem kann über einen *Report and Notification Service* (RNS) der DirXML-Status sowie Ereignisse der verbundenen Datenquellen auf verschiedene Art und Weise (z.B. zentrales Syslog, E-Mail, Web) weitergegeben werden.

Die Administration der DirXML-Engine sowie der Treibereinstellungen kann sowohl über den iManager (Abbildung 4.5) als auch über die ConsoleOne (JavaGui) erfolgen.

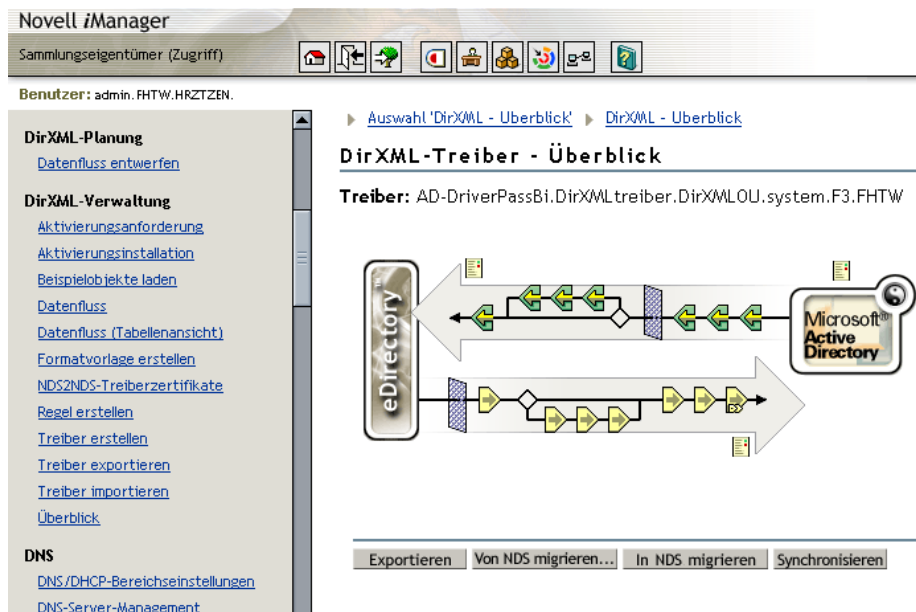


Abbildung 4.5: DirXML Administrationsoberfläche iManager

Durch die Aufteilung des Bearbeitungsprozesses der DirXML Dokumente in einzelne Regeln, können die Rechte für die Administration auf Regelebene verteilt werden.

Der iManager ist dezentral verfügbar und benötigt zur Benutzung lediglich einen aktuellen Browser. Die Verbindungen werden mit Hilfe von HTTPS verschlüsselt übertragen. Für die Administration von DirXML enthält der iManager einige Wizards, die die Konfiguration der Regeln und Filter erleichtern.

DirXML Treiber für Active Directory

Novell bietet vorkonfigurierte Treiber (Konnektoren) für die Anbindung von Datenquellen und Verzeichnisdiensten an. Mit Hilfe des Active Directory Treiber werden die DirXML Funktionalitäten für die ADS umgesetzt. Er wird auf einem Domänenserver oder einem Domänenrechner (Windows 2000) installiert und über den Remote Loader die Kommunikation mit der DirXML-Engine hergestellt. Der Remote Loader kann zur Lastverteilung, abhängig vom DirXML-Treiber, ebenfalls auf einem normalen Client-Rechner der ADS-Domäne installiert werden. Die Konfiguration erfolgt über die eDirectory-Administrationsoberflächen ConsoleOne oder iManager.

Der Treiber unterstützt uni- und bidirektionale Synchronisation zwischen Objekten der eDirectory und der ADS. Objekte können an jeder Stelle in der Verzeichnishierarchie platziert werden. Mit Hilfe des Treibers können fast alle Objektklassen (z.B. Benutzer, Gruppen,

Container) und deren Attribute synchronisiert werden. Auch selbst erstellte Objektklassen und Attribute können abgeglichen werden.

Um Änderungen der Active Directory zu erkennen, wird in einer definierbaren Synchronisationsfrequenz das *DirSync Control* der ADS benutzt [Kuo02, S.417]. Es ermöglicht eine Suche nach allen geänderten Attributen seit der letzten Anfrage. Synchronisationsvorgänge können demnach nur zeitgesteuert auf der Seite des Active Directory ausgelöst werden. Das Dirsync Control kann weiterhin nur zur Änderungsabfrage einer Domäne verwendet werden. Für jede weitere Domäne muss ein neuer DirXML-Treiber verwendet werden.

Für die Verschlüsselung von Datenübertragungen zwischen der DirXML Driver Engine und dem Active Directory Server verwendet der Treiber eine SSL-Verschlüsselung.

Passwortsynchronisation

Für die Passwortsynchronisation bietet der Active Directory Treiber eine Erweiterung, *PasswordSync*, an. Damit können Passwörter bidirektional abgeglichen werden.

With PasswordSync, a user is required to remember only a single password to log in to any of these systems. Administrators can manage passwords in the system of their choice. Any time a password is changed in one of these environments, it will be updated in all of them. [Nov03f]

PasswordSync macht sich den Umstand zu Nutzen, dass Windowsrechner in einer Domäne bei Passwortänderungen ihre Passwörter in Klartext an den Domänencontroller senden. Dieser verschlüsselt die Passwörter und legt sie im Active Directory ab. Der PasswordSync Dienst fängt die Passwortänderungen (mit Klartextpasswort) mit Hilfe eines Passwortfilters ab und gibt sie an einen Agenten weiter.

Der Passwortfilter für die eDirectory benutzt den *Novell Client*, um das Passwort von dort aus an einen Agenten weiterzuleiten. Im Gegensatz zu Microsoft-Client leitet der Novell Client Passwörter nicht in Klartext an andere Netzwerkressourcen weiter. Der Password-Sync Filter für eDirectory ist eine Erweiterung des Novell Client Softwarepaketes und ist ab der Version 4.81 (Win2000 / NT) im Client enthalten.

Obwohl das Synchronisieren der Passwörter bestehender Accounts möglich ist, gibt es Nachteile, welche auf der Architektur des PasswordSync-Programmes beruhen. So ist es nicht möglich Passwörter direkt nach dem Anlegen von Nutzeraccounts zu synchronisieren. Dies ist durch die fehlende Assoziation mit dem erst später durch DirXML erzeugten ADS-Account begründet.

Eine Passwortsynchronisation findet erstmalig beim Ändern des Passwortes nach hergestellter Assoziation zwischen ADS- und NDS-Nutzerobjekt statt.

Als Lösungsmöglichkeit schlägt Novell die Einrichtung mit einem Initialpasswort bei der eigentlichen Nutzeraccounterzeugung vor. In einem folgenden zweiten Durchgang wird das Passwort dann auf seinen richtigen Wert gesetzt. Eine weitere Möglichkeit, so Novell, ist eine Aufklärung der Nutzer, dass das Passwort erst nach einem erstmaligen Ändern mit dem jeweiligen ADS-Account synchronisiert wird [Nov03c].

4.3.3 ADS Anforderungen

Um die vorhandenen Datenabhängigkeiten zu veranschaulichen, werden alle momentan verwendeten Attribute, welche zum Anlegen eines Accounts im Active Directory benötigt werden, betrachtet. Daraus lassen sich später Synchronisationspartner ableiten und Erstellungsregeln entwickeln.

Attribut)	Beispielwert
sAMAccountName	s0282129
cn	s0282129
displayName	Taito Radtke
sn	Radtke
givenName	Taito
department	TI2002WS
memberof	CN=Studenten,OU=Gruppen,DC=fb1,DC=fhtw-berlin,DC=de
memberof	TI2002WS,OU=Gruppen,DC=fb1,DC=fhtw-berlin,DC=de
mail	s0282129@fhtw-berlin.de
userPrincipalName	s0282129@fb1.fhtw-berlin.de
homeDrive	I
homeDirectory	\\fb1-user\s0282129
scriptPath	logon.bat
userAccountControl	544

Tabelle 4.1: Momentan verwendete ADS Attribute

Die in der Tabelle 4.1 aufgeführten Attribute werden aus der Datenbank des Fachbereiches importiert. Dabei gelten folgende Bildungsregeln für die Attribute:

- Als Loginnamen des Benutzers wird das Attribut „sAMAccountName“ verwendet. Er wird aus der Matrikelnummer des Studenten mit führendem „s0“ gebildet und ist gleich dem ADS-Nutzernamenattribut „cn“.
- Die Attribute mit den Namen „displayName“, „sn“ und „givenName“ entsprechen dem kompletten Namen, dem Nachnamen und dem Vornamen eines Studenten.
- Der Wert des Attributes „department“ wird aus dem Immatrikulationsdatum und dem Studiengang gebildet. Der Wert „TI2002WS“ setzt sich demnach aus der Kurzform des Studienganges Technische Informatik und dem Kürzel „WS“ für Wintersemester, sowie dem Jahr der ersten Immatrikulation „2002“ zusammen.

- Jeder Student wird Teilnehmer einer weiteren Gruppe mit dem „**department**“-Attributwert als Namensgeber. Diese Zuggruppe befindet sich in der ADS-Verzeichnisbaumhierarchie an der gleichen Stelle wie die alle Studenten umfassende Gruppe „CN=Studenten,OU=Gruppen,DC=fb1,DC=fhtw-berlin,DC=de“.
Der Name der Gruppe ergibt sich demnach aus „CN=**department**,OU=Gruppen,DC=fb1,DC=fhtw-berlin,DC=de“.
Die Gruppenzugehörigkeiten der Zuggruppen werden im FB1 für E-Maillisten verwendet.
- Die E-Mailadresse mit dem Attributnamen „**mail**“ entspricht der E-Mailadresse eines Studenten an der FHTW und folgt der Bildungsregel „**cn**@fhtw-berlin.de“.
- Das Attribut „**userPrincipalName**“ wird vom ADS als ein DNS-konformes Namensattribut (siehe Kapitel 1 ADS) von Verzeichnisobjekten verwendet. Der Attributwert wird mit Hilfe des ADS-Domänennamens „fb1.fhtw-berlin.de“ und dem Wert des Attributes „**cn**“, also „**cn**@fb1.fhtw-berlin.de“ gebildet.
- Der Wert des Attributes „**homeDrive**“ wird für die Laufwerksbindungen der persönlichen Studentenverzeichnisse auf dem Dateiserver verwendet und entspricht immer dem Buchstaben „I“. Das persönliche Studentenverzeichnis auf dem Dateiserver ist in dem Attribut „**homeDirectory**“ abgebildet. Die feste Bildungsregel für dessen Wert lautet „\\fb1-user**cn**“.
- Die Attribute „**scriptPath**“ und „**userAccountControl**“ werden auf feste Werte entsprechend den Beispielen der Tabelle gesetzt. Damit wird einerseits der Name des Loginskriptes angegeben, mit dem zweiten Attribut die Passwortregeln (Benutzer kann Passwort ändern, Konto ist aktiv und Passwortänderung nicht verlangt) in das ADS geschrieben.
- Das Passwortattribut wird beim Importvorgang im Klartextformat aus der Fachbereichsdatenbank ausgelesen und über das Skript mit Hilfe der ADSI-Schnittstelle verschlüsselt abgespeichert.

Nachdem die momentan benutzten Attribute des Active Directory betrachtet wurden, werden im nächsten Schritt Synchronisationsattribute des NDS bestimmt. Auf eine Betrachtung der NDS-Attribute wird verzichtet, da das NDS-Verzeichnis als zentrales Verzeichnis der Metaverzeichnislösung DirXML arbeitet. Für den Fall, dass die vorliegenden Attribute des NDS-Verzeichnisses nicht ausreichen, werden Vorschläge zum importieren weiterer Attribute aus der Logindatenbank des Rechenzentrums gemacht.

4.3.4 Attributzuweisung zwischen NDS und ADS

Die Attributzuweisung zwischen dem NDS- und dem ADS-Verzeichnis teilt sich in mehrere Bereiche. Es existieren Attribute, welche direkt verbunden werden können. Andere Attribute wiederum sind im NDS-Verzeichnis nicht verfügbar und müssen nachträglich aus der RZ-Logindatenbank importiert oder mit Hilfe vorhandener Attribute erzeugt werden. Dazu gehören auch Betriebsattribute, welche nur für die Einrichtung der Active Directory Accountdaten verwendet werden. Sie werden mit keinem NDS-Attributwert verknüpft.

Direkt abzugleichende Attribute

NDS (HRZ)	ADS (FB1)	Bemerkungen	Sync.form
CN	cn		Uni (NDS)
CN	sAMAccountName		Uni (NDS)
CN	userPrincipalName	Wertumwandlung	Uni (NDS)
Full Name	displayName		Uni (NDS)
Surname	sn		Uni (NDS)
Given Name	givenName		Uni (NDS)
Login Disabled	userAccountControl	Wertumwandlung	Bi

Tabelle 4.2: Abzugleichende Attribute der Verzeichnisdienste

Die Tabelle 4.2 zeigt die direkt verbundenen Attributpaare beider Verzeichnisdienste. Das NDS-Verzeichnis ist Dateneigner für alle Attribute, welche über eine unidirektionale Synchronisation abgeglichen werden. Somit darf die NDS als einziger Verzeichnisdienst Änderungen an den Attributwerten vornehmen.

Ein weiteres Attribut, welches bidirektional synchronisiert werden muss, ist das Passwort. Dieses ist, wie im zweiten Kapitel erwähnt, jedoch in unterschiedlichen Formaten in den Verzeichnisdiensten abgespeichert und wird mit Hilfe des PasswordSync-Werkzeuges von DirXML abgeglichen. Beide Verzeichnisdienste können dieses Attribut ändern, ohne die Konsistenz zu gefährden.

Weitere benötigte Attribute

Wie die vorherige Betrachtung der ADS-Attributanforderungen zur Erstellung eines Benutzeraccounts ergeben hat, werden noch weitere Attribute benötigt. Diese besitzen jedoch keine Synchronisationspartner im NDS-Verzeichnis, können aber durch die vorhandenen Bildungsregeln aus NDS-Attributen in DirXML erzeugt werden. Sie sind in der Tabelle 4.3 dargestellt.

Aufgrund der gewünschten getrennten Administration der Betriebssystemumgebungen wird von einer späteren Implementierung dieser Betriebsattribute jedoch Abstand genommen. Die Administration des Active Directory kann diese Attribute mit Hilfe von ADSI

benutztes NDS-Att.	ADS (FB1)	Bildungsregel
CN	mail	CN@fhtw-berlin.de
-	homeDrive homeDirectory scriptPath	„I“ \\fb1-user\CN „logon.bat“

Tabelle 4.3: Weitere Betriebsattribute des ADS mit Bildungsregel

Skripten selbst erzeugen. Für die Erkennung von Accountdaten, welche in der Active Directory durch DirXML angelegt wurden, wird das `description`-Attribut der ADS beim Anlegen neuer Nutzerobjekte mit dem Wert „DirXML-Sync“ gefüllt.

Ein weiteres, noch unberücksichtigtes, Attribut ist das `department`-Attribut der ADS. Es wird für die Bezeichnung des Studiengangszuges, dem ein Student angehört, verwendet. Die Bildung dieses Attributes ist auf das Immatrikulationsdatum angewiesen. Das Immatrikulationsdatum ist jedoch in der RZ-Logindatenbank nicht enthalten und kann demzufolge nicht in der NDS erzeugt werden. Dieses Problem kann nicht unmittelbar gelöst werden. Eine Überprüfung der Datenquelle (Immatrikulationsdatenbank) der RZ-Logindatenbank auf geeignete Nutzereigenschaften muss erfolgen und nach Absprache mit den Datenschutzbeauftragten diese Einträge in die RZ-Logindatenbank übertragen werden.

4.3.5 Synchronisationsfrequenz

Die meisten Nutzerdaten werden am Anfang eines Semesters in grossen Blöcken eingerichtet. In diesem Zeitraum werden die Accountdaten einmal täglich von der RZ-Logindatenbank in die NDS eingepflegt. Die unidirektional zu synchronisierenden Nutzattribute werden nach der Einrichtung kaum oder meistens gar nicht geändert. Eine Synchronisationsfrequenz von einem Tag sollte demzufolge für die unidirektional abzugleichenden Attribute ausreichen.

Jedoch sieht das Lösungskonzept eine Passwortsynchronisation und den bidirektionalen Abgleich von Accountsperrungen vor. Der Abgleich dieser Attribute ist als zeitkritisch anzusehen. Da die Active Directory, im Gegensatz zum NDS-Verzeichnis, nur zeitgesteuert nach Änderungen abgefragt werden kann, ist die Synchronisationsfrequenz für eine Weiterleitung der Änderungen ausschlaggebend.

Die Benutzung der Ressourcen ist teilweise standortunabhängig möglich, es muss demnach eine relativ hohe Synchronisationsfrequenz gewählt werden. So kann es vorkommen, dass ein Benutzer sein Passwort in der ADS-Umgebung des Fachbereiches 1 ändert und wenige Zeit später auf den FTP-Server des NDS-Verzeichnisservers im HRZ zugreifen will. Eine Synchronisationsfrequenz von 10 Minuten wird für die erste Implementierung empfohlen. Sie sollte jedoch im weiteren Betrieb überprüft und den Erfahrungen nach angepasst werden.

4.3.6 Auswahl der zu synchronisierenden Objekte

In der Hierarchie des NDS-Verzeichnisbaumes wird keine Ordnung der Nutzerobjekte nach Fachbereichen vorgenommen. Es befinden sich alle Nutzerdaten in einem Containerobjekt. Eine Auswahl der mit dem Fachbereich 1 zu synchronisierenden Nutzerdaten muss demnach anhand von Attributen realisiert werden. Momentan existiert kein Attribut, mit dem eine Unterscheidung nach Fachbereichen möglich ist.

Die RZ-Logindatenbank beinhaltet für jeden Studenten seine Studiengangsnummer und den zugehörigen Fachbereich. Dieses Fachbereichsattribut der Logindatenbank kann in die NDS importiert werden. Im NDS-Verzeichnis würde sich dafür das noch nicht verwendete Locationattribut (Attributname: L) der Objektklasse „User“ als Speicherort für den Studiengang anbieten. Der Inhalt des Wertes kann eine 128 Zeichen umfassende Zeichkette beinhalten, was für diese Zwecke ausreichend ist.

Ein weitaus einfachere Möglichkeit ist eine Gruppe, in welcher alle zu synchronisierenden Nutzerobjekte Mitglied sind. In einem Nutzerobjekt der NDS wird diese Gruppenzugehörigkeit in dem `Group Membership`-Attribut gespeichert. Mit Hilfe der Create Rule im Subscriber-Kanal kann nun das Gruppenattribut überprüft und eine Erstellung eines Nutzeraccounts im Active Directory initiiert werden. Um die Gruppenzugehörigkeit im NDS-Verzeichnis anzulegen, wird das bereits erwähnte Fachbereichsattribut der RZ-Logindatenbank bei der Nutzeinrichtung ausgewertet.

4.3.7 Performanceanpassungen

DirXML ist für eine Synchronisation von Objekten auf eine Read/Write-Replik der Partition angewiesen, in der die zu synchronisierenden Benutzerdaten enthalten sind. In der Standardeinstellung werden alle Änderungen der durch DirXML verwendeten Partition nach abzugleichenden Objekten hin untersucht. Dadurch kann eine unnötige Last seitens der DirXML-Engine entstehen. Um den Fokus auf Änderungsereignisse an synchronisationsrelevanten Daten einzuschränken, kann ein „Scope Filtering“ benutzt werden [Nov03a]. Damit ist es möglich, nur Änderungen relevanter Containerobjekte im Verzeichnisbaum zu berücksichtigen.

Die NDS des HRZ sind in verschiedene Partitionen aufgeteilt. Eine davon umfasst den Container „USERS.FHTW“. In diesem Containerobjekt sind alle Nutzerobjekte (ca. 9.000), sowohl die für einen Abgleich mit dem FB1 relevanten, als auch die irrelevanten gespeichert. Die Anzahl der Nutzerobjekte, welche für einen Abgleich in Frage kommen, beträgt ca. 1000. Es werden demnach Änderungsereignisse an 8000 Objekte umsonst untersucht. Die DirXML-Filter können keine Attributwerte auswerten, wodurch die Selektierung erst in der DirXML-Engine über die Erstellungsregel umgesetzt werden kann.

Eine Lösung wäre die nachträgliche Feingliederung der NDS-Hierarchie im Nutzerdatencontainer. Es müssten für jeden Fachbereich ein eigenes Containerobjekt erstellt und die jeweiligen Nutzerdaten in diese neu eingeordnet werden. Die Vorgang ist sehr aufwendig, stellt allerdings die beste Lösung für das genannte Problem dar. Des Weiteren könnte auch die Gruppenzugehörigkeit der zu synchronisierenden Nutzerobjekte entfallen.

Eine weitere Performanceanpassung wird über die Platzierung der DirXML-Komponenten erreicht. Der Aufbau wird im Implementierungskapitel erläutert.

5 Implementierung

5.1	Aufbau der Synchronisationsstruktur	72
5.2	Datenfluss	73
5.3	Filter	74
5.4	Regeln	75
5.4.1	SubscriberMatchingRule	75
5.4.2	SubscriberCreateRule	76
5.4.3	SubscriberPlacementRule	76
5.4.4	SubscriberMappingRule	77
5.4.5	Output Stylesheet	78
5.5	Konfiguration der Passwortsynchronisation	78
5.6	Sicherheit	79

Die Implementierung stellt die Umsetzung des Lösungskonzeptes dar. Gemäß dem Lösungskonzept wird die Synchronisation wichtiger NDS-Nutzobjektattribute von im Fachbereich 1 studierenden Personen in die ADS umgesetzt. Weiterhin wird die Implementierung der Passwortsynchronisation zwischen NDS (HRZ1) und ADS (FB1) erläutert.

Weitere Bestandteile der Implementierung sind die notwendigen Erweiterungen der Betriebssysteme sowie der strukturelle Aufbau der Synchronisationskomponenten. Sie werden der Vollständigkeit halber ebenfalls erläutert.

5.1 Aufbau der Synchronisationsstruktur

Der Aufbau der Synchronisationsstruktur und die Platzierung der einzelnen DirXML Komponenten ist in der Abbildung 5.1 auf der nächsten Seite dargestellt.

Diese Verteilung der Komponenten soll die bestehenden Verzeichnisdienstserver vor weiterer Last bewahren und die Performance des Synchronisationsvorganges erhöhen. Die DirXML-Engine wird auf einen separaten NDS-Verzeichnisserver installiert. Dieser erhält nur eine Read/Write-Replik der Partition, die den Container „USERS.FHTW“ umfasst.

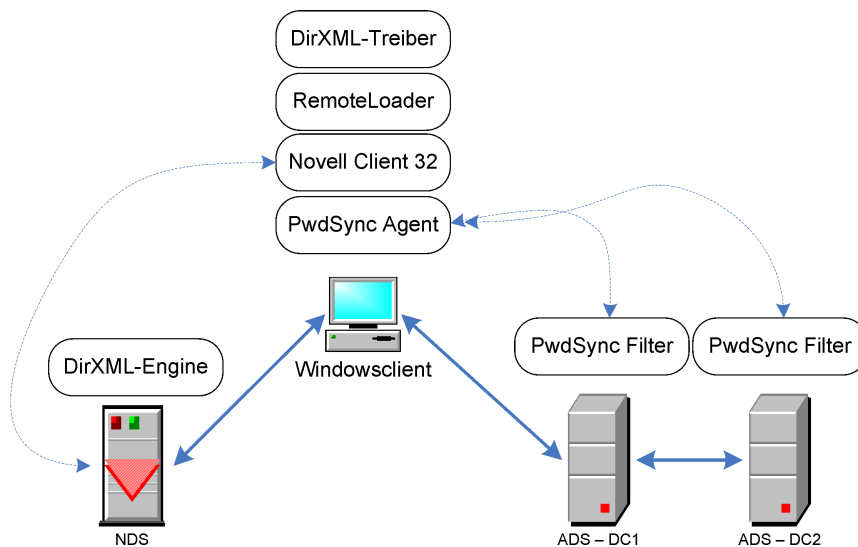


Abbildung 5.1: Implementierung der DirXML Komponenten

Auf einem Windows-Client (Windows 2000 Professional erforderlich) der ADS-Domäne werden der DirXML AD Treiber, der Remoteloader, der Novell Client 32 sowie der PasswordSync Agent installiert. Damit entlastet der Windows-basierte Rechner die vorhandenen Domänencontroller. Die Domänencontroller werden dadurch, bis auf die PasswordSync Filter, von weiteren Softwareinstallationen verschont.

Für eine Installation von DirXML ist ein Update der NDS des HRZ auf die aktuelle Version eDirectory 8.6 notwendig. Damit verbunden ist eine Erweiterung des Verzeichnisseschemas.

Obwohl DirXML auch für die bestehende Netwareversion der NDS-Verzeichnisse (5.1) verfügbar ist, wird eine Installation auf einem Netwareserver der aktuellen Version 6.5 empfohlen. Diese Empfehlung resultiert aus der Tatsache, dass DirXML im Netware-Serverpaket der Version 6.5 kostenlos als sogenanntes „StarterPack“ enthalten ist. Das DirXML-Starterpack umfasst neben dem in der Implementierung verwendeten Treiber für das Active Directory auch einen LDAP- und einen Textdateitreiber. Eine spätere Anbindung weiterer Datenquellen (z.B. RZ-Logindatenbank) über DirXML ist demnach möglich.

5.2 Datenfluss

Für die Synchronisation der Verzeichnisse werden nur die Attribute der Objektklasse „User“ berücksichtigt, da nur Nutzerdaten synchronisiert werden sollen. Der Datenfluss beruht auf den im Lösungskonzept (Kapitel 4) diskutierten Attributen.

In der Tabelle 5.1 auf der nächsten Seite sind die Attribute und die Synchronisationkanäle

NDS (HRZ)	ADS (FB1)	Publisher	Subscriber
CN	cn		•
CN	sAMAccountName		•
CN	userPrincipalName		•
Full Name	displayName		•
Surname	sn		•
Given Name	givenName		•
Title	title		•
Login Disabled	userAccountControl	•	•
	description		↗

Tabelle 5.1: Datenfluss zwischen NDS und ADS

dargestellt. Unidirektional zu synchronisierende Attribute werden durch den Subscriber-Kanal vom Active Directory Treiber empfangen.

Bidirektional abzugleichende Attribute müssen sowohl im Subscriber-Kanal als auch im Publisher-Kanal berücksichtigt werden.

Das `description`-Attribut soll später für eine Kennzeichnung der durch DirXML angelegten Objekte in der ADS verwendet werden. Das Attribut wird nicht in der Schema Mapping Regel berücksichtigt, es wird im Subscriber-Kanal der DirXML-Engine erzeugt. Der Datenfluss des Attributes ist in der Tabelle als ein gerichteter Pfeil (↗) dargestellt.

Zu den im Lösungskonzept näher erläuterten Attributen kommt noch ein weiteres Attribut hinzu. Aus Gründen der Vollständigkeit wird das zum Namen gehörende `Title`-Attribut in eine Synchronisation miteinbezogen. Es wird momentan nicht bei Studenten verwendet, eine Synchronisation ist aufgrund einer späteren Benutzung dennoch sinnvoll.

5.3 Filter

Für die Implementierung der Filter und Regeln wird der vorkonfigurierte bidirektionale Active Directory Treiber verwendet. Er beinhaltet eine Anzahl von Zuweisungen zwischen Nutzerattributen der Verzeichnisdienste und deren teilweise benötigten Datentransformationen. Er unterstützt das Anlegen, Löschen und Modifizieren von Nutzerobjekten.

Für die Umsetzung des Lösungskonzeptes werden Anpassungen am Publisher- und Subscriber-Filter durchgeführt. Die Filter haben die Aufgabe nur Attributänderungsereignisse der, in den jeweiligen Regeln, verwendeten Attribute zur DirXML-Engine passieren zu lassen. Gemäß des Datenflusses ergeben sich folgende Filterregeln:

- **Filter des Publisher-Kanal**

Erlaubte Attribute: `cn`, `userAccountControl`, `nadLoginName`

Erlaubte Objektklassen: `User`

- **Filter des Subscriber-Kanal**

Erlaubte Attribute: CN, Full Name, Surname, Given Name, Title, Login Disabled

Erlaubte Objektklassen: User

Der Filter des Publisher-Kanals beinhaltet, zusätzlich zu den im Datenfluss erläuterten Attributen, noch das Attribut `nadLoginname`. Es wird für die Passwortsynchronisation verwendet.

5.4 Regeln

Um dem Leser ein besseres Verständnis für den Aufbau der Kanäle und den damit verbundenen Regeln zu verschaffen, dient die Abbildung 5.2.

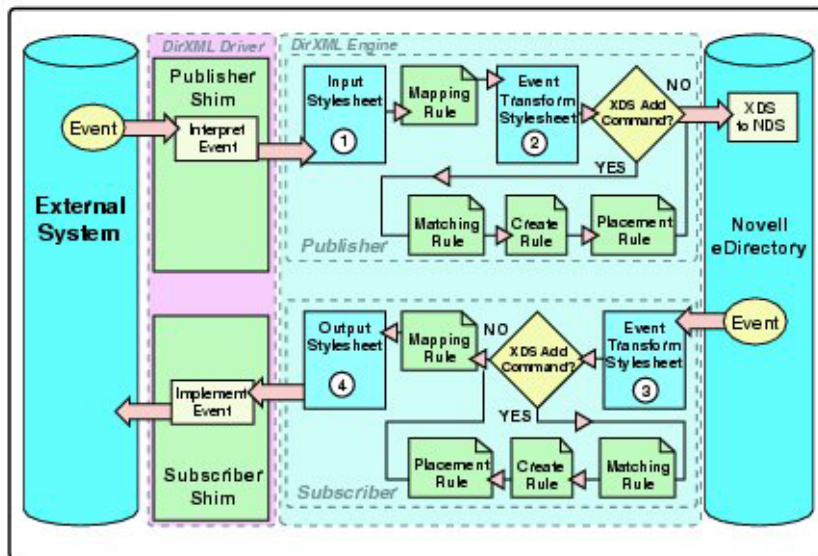


Abbildung 5.2: DirXML Regeln in Publisher- und Subscriber-Kanal [Nov03d]

Die Erläuterung der Regelimplementierung erfolgt am Beispiel des Subscriber-Kanals. Für eine Realisierung des Datenflusses werden folgende Regeln und Stylesheets verwendet:

5.4.1 SubscriberMatchingRule

Bestehende Nutzerobjekte in beiden Verzeichnisdiensten werden über einen Vergleich des CN-Attribute in Beziehung gebracht. Das CN-Attribut enthält in beiden Verzeichnisdiensten denselben Wert („s0“ + Matrikelnummer an der Hochschule) und ist eindeutig.

Vergleichsattribut: CN

SubscriberMatchingRule

```
<matching-rules>
  <matching-rule description="Users">
    <match-class class-name="User"/>
    <match-attr attr-name="CN"/>
  </matching-rule>
</matching-rules>
```

5.4.2 SubscriberCreateRule

Mit dieser Regel werden die, für eine Erstellung eines Nutzerobjektes (ADS), notwendigen Attribute im NDS-Verzeichnis abgefragt und somit ihr Vorhandensein überprüft. Weiterhin wird an dieser Stelle die Gruppenzugehörigkeit überprüft und danach bestimmt ob das Objekt in der ADS neu angelegt werden soll.

Abgefragte NDS Attribute: CN, GivenName, Surname, Full Name, Group Membership (Wert: „SyncFB1.USERS.FHTW“)

SubscriberCreateRule

```
<create-rules>
  <create-rule class-name="User" description="Users">
    <required-attr attr-name="CN"/>
    <required-attr attr-name="Given Name"/>
    <required-attr attr-name="Surname"/>
    <required-attr attr-name="Full Name"/>
    <match-attr attr-name="Group Membership">
      <value type="string">SyncFB1.USERS.FHTW</value>
    </match-attr>
  </create-rule>
</create-rules>
<create-rules>
```

5.4.3 SubscriberPlacementRule

Die Nutzerobjekte beider Verzeichnisdienste sind jeweils alle innerhalb eines Containerobjekts untergebracht. Die Pfade der Verzeichnisdienste sind eindeutig und müssen ineinander überführt werden. Sie werden für die Platzierung der Nutzerobjekte in den Verzeichnisdienstbäumen und somit zur Bildung des Distinguished Name verwandt.

NDS Pfad: .OU=USERS.OU=FHTW

ADS Pfad: ,OU=Benutzer,DC=fb1,DC=fhtw-berlin,DC=de

SubscriberPlacementRule

```
<placement-rules dest-dn-format="ldap" src-dn-format="slash">
  <placement-rule description="General">
    <match-class class-name="User"/>
    <match-path prefix="FHTW\USERS"/>
    <placement>CN=<copy-attr attr-name="CN"/>
      ,OU=Benutzer,DC=fb1,DC=fhtw-berlin,DC=de
    </placement>
  </placement-rule>
</placement-rules>
```

5.4.4 SubscriberMappingRule

Mit Hilfe dieser Regel werden die Objekt- und Attributrelationen zwischen beiden Verzeichnisdiensten hergestellt. Um Attributrelationen von Nutzerdaten herstellen zu können, wird zunächst eine Objektrelation der Klasse `User` angelegt. Die verbundenen Attribute sind in der Tabelle 5.2 dargestellt.

NDS (HRZ)	ADS (FB1)
User	user
CN	cn
CN	sAMAccountName
CN	userPrincipalName
Full Name	displayName
Surname	sn
Given Name	givenName
Login Disabled	userAccountControl
Title	title

Tabelle 5.2: Mapping Rule

Ein exemplarischer Auszug der Regel ist im folgenden Quellcode dargestellt.

SubscriberMappingRule (Auszug)

```
<attr-name-map>
  <class-name>
    <nds-name>User</nds-name>
    <app-name>user</app-name>
  </class-name>
```

```
<attr-name class-name="User">
  <nds-name>CN</nds-name>
  <app-name>userPrincipalName</app-name>
</attr-name>
<attr-name class-name="User">
  <nds-name>Login Disabled</nds-name>
  <app-name>userAccountControl</app-name>
</attr-name>
```

5.4.5 Output Stylesheet

Im Output Stylesheet des Subscriber-Kanals werden letzte Änderungen an den XML Dokumenten vorgenommen. Er wird vor allem für die Datenumwandlung verwendet. Außerdem bietet er die Möglichkeit Attribute ausserhalb der Schema Mapping Regel hinzuzufügen. Das `description`-Attribut soll nur in Richtung ADS ausgegeben werden und wird mit einem festen Wert belegt. Dafür wird das XML Dokument in dieser Instanz nach einem „add user“ Befehl durchsucht, und bei Erfolg das zusätzliche Attribut eingefügt.

Einzufügendes Attribute: `description` („DirXML-Sync“)

Output Stylesheet

```
<xsl:template match="add[@class-name='user']">
  <xsl:copy>
    <xsl:apply-templates select="node()|@*" />
    <add-attr attr-name="description">
      <value type="string">DirXML-Sync</value>
    </add-attr>
  </xsl:copy>
</xsl:template>
```

5.5 Konfiguration der Passwortsynchronisation

Für die Synchronisation des Passwortes wird DirXML als Übertragungsweg benutzt. Die Installation und Konfiguration der Passwortsynchronisation wird jedoch extra vorgenommen.

Bei der ersten Installation wird das Schema der eDirectory erweitert. Dabei werden neue Objektklassen für die Speicherung der Konfigurationsdaten der PasswordSync Filter und

des Agenten angelegt. Der Agenten und die, auf den Domänencontrollern, verwendeten Filter werden über die Systemsteuerung des Windows-Clients konfiguriert.

Während der PasswordSync Agent auf einem Windows 2000 Rechner installiert werden kann, müssen die PasswordSync Filter auf allen Domänenservern eingerichtet werden. Nur damit ist es möglich, die Passwortänderungen der Windowsrechner abzufangen und an die NDS weiterzuleiten.

5.6 Sicherheit

Für die Wahrung der Sicherheitsanforderungen wird mit Hilfe des DirXML Active Directory Treiber die Verschlüsselung der Datenübertragung mit SSL eingerichtet. Für die Sicherung des Zugriffes auf die Verzeichnisdaten müssen in den Verzeichnissen folgende Vorkehrungen getroffen werden:

NDS

Innerhalb des Novell-Verzeichnisdienstes werden Änderungen an den Verzeichnisobjekten, die durch einen Synchronisationsvorgang entstehen, durch das DirXML-Treiberobjekt durchgeführt. Die Rechtevergabe für den Treiber erfolgt über ein Nutzerobjekt, dem der Treiber sicherheitsrechtlich gleichgestellt wird. Um Änderungen an unbeteiligten Objekten zu vermeiden, werden Änderungsrechte nur für die Mitglieder des Synchronisationsgruppenobjekts (Auswahlkriterium für abzugleichende Nutzer) erteilt.

ADS

Innerhalb der ADS-Domäne werden Änderungen an den Verzeichnisdaten von dem Remote Loader entgegengenommen und von dem DirXML Treiber durchgeführt. Dieser benutzt dafür ein Nutzerobjekt der Active Directory, dass mit den benötigten Rechten ausgestattet sein muss. Die erforderlichen Rechte zum Anlegen von Nutzern können nicht eingeschränkt werden. Eine Begrenzung der Rechte auf den Benutzercontainer, in dem die zu synchronisierenden Nutzerobjekte angelegt werden, sollte jedoch durchgeführt werden.

6 Test

Um einen Teil der im Lösungskonzept diskutierten Funktionen zu testen und die Einhaltung der Anforderungen zu überprüfen, wurde eine Testumgebung aufgebaut. Durch unterschiedliche Testszenarien wurden sowohl die DirXML-Synchronisation mit den implementierten Regelsätzen und Filtern, als auch die Passwortsynchronisation getestet.

6.1 Testumgebung

Im Gegensatz zur Implementierung wurde für die Testumgebung eine andere Verteilung der Komponenten gewählt. Der Aufbau der Testumgebung und die Platzierung der Komponenten ist in Abbildung 6.1 dargestellt.

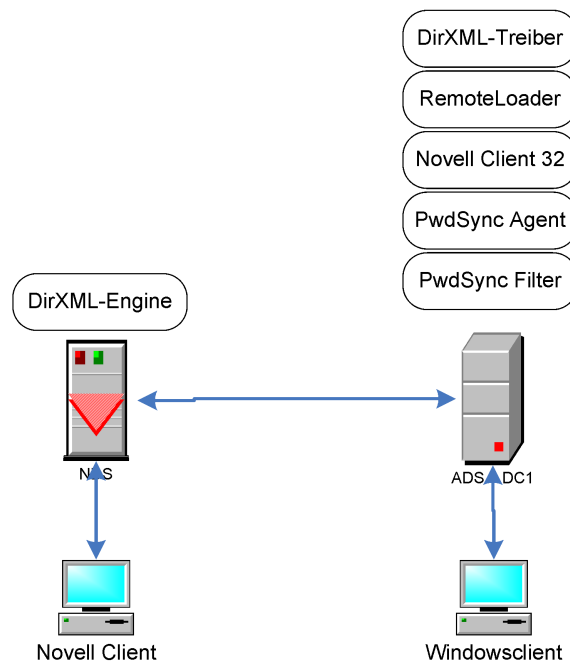


Abbildung 6.1: Aufbau der Testumgebung

Der Testaufbau gliederte sich in zwei Teile. Die Windowsdomäne bestand aus einem Domänencontroller und einem Domänen-Client. Das Novell Netzwerk bestand aus einem NDS-Verzeichnisserver und einem Windows-basierten Novell-Client.

Domänencontroller (ADS)

- Betriebssystem: Windows 2000 Advanced Server, ServicePack 4
- DirXML Komponenten: DirXML AD Treiber, Remoteloader, PasswordSync Agent, PasswordSync Filter, Novell Client 32 V4.9
- Weitere zusätzliche Komponenten: Microsoft DNS Server, WINS Server
- Hardware: PC Pentium 4 / 2,4 GHz, 512MB RAM

Domänen-Client (ADS)

- Betriebssystem: Windows 2000 Profesional SP2
- Netzwerk-Client Programm: Client für Microsoft-Netzwerke
- Hardware: PC Athlon / 600 MHz, 128MB RAM

NDS-Verzeichnissever

- Betriebssystem: Netware 6.5 (Evaluierungsversion), NDS 10510.64 / eDirectory 8.7.1
- DirXML Komponenten: DirXML-Engine
- Weitere zusätzliche Komponenten: iManger 2.0 (Apache, MySQL, Perl, Tomcat)
- Hardware: PC Pentium 4 / 1,8 GHz, 1024MB RAM

NDS-Client

- Betriebssystem: Windows 2000 Profesional SP4
- Netzwerk-Client Programm: Novell Client 32 V4.83
- Hardware: PC Athlon / 1,6 GHz, 256MB RAM

Verwendete DirXML Versionen

- DirXML-Engine: Version 1.1a (DirXML Starterpack für Netware 6.5)
- DirXML Active Directory Treiber: Version 2.0a (enthalten im DirXML Starterpack für Netware 6.5)
- PasswordSync: Version 1.0 (enthalten im DirXML Starterpack für Netware 6.5)

Für die Administration des DirXML-Treibers und der eDirectory wurden sowohl der iManager, als auch die ConsoleOne benutzt. Für die Verwaltung der ADS wurde die Microsoft Management Console verwendet. Um die einzelnen Abläufe von DirXML zu untersuchen, wurde das NDS-eigene Werkzeug *dstrace* und das Trace-Fenster des Remote Loader verwendet.

6.2 Testergebnisse der Funktionstests

In unterschiedlichen Testszenarien und auf die Regeln zugeschnittenen Einzeltests wurden die Funktionen der in der Implementierung erläuterten Regeln überprüft. Im einzelnen wurden unter anderen Tests zum Anlegen neuer und bestehender Nutzerobjekte (Überprüfung der Erstellungs- und Übereinstimmungsregeln), sowie Löschen und Modifizieren (auch gleichzeitiges Modifizieren in ADS und NDS) von Nutzerobjekten (Überprüfung der Filter und Schemaregeln) durchgeführt. Weiterhin wurden Tests zum Umbenennen und von Nutzerobjekten (Überprüfung des Assoziationsattributes) durchgeführt. Die Tests verliefen erfolgreich, die Synchronisation der Nutzerdaten wurde erwartungsgemäß von DirXML realisiert.

In einer weiteren Testgruppe wurde DirXML auf seine Performance hin untersucht. Dabei wurden in einem Verzeichnis 700 Nutzerobjekte „gleichzeitig“ angelegt und danach die Synchronisation gestartet. Zur Überprüfung der Performance wurden die Zeiten der Synchronisationsläufe aufgenommen und die Serverlast der Verzeichnisse server beobachtet. So dauerte das Anlegen der 700 Nutzerobjekte beim Synchronisationspartner 1:30 min (gerundet). Änderungen eines Attributwertes bei 700 Nutzern wurden in 40 Sekunden synchronisiert und das Löschen der Nutzerobjekte in beiden Verzeichnissen (initiiert durch das Löschen der Nutzerobjekte im NDS-Verzeichnis) und dauerte 1:10 min. Die Last des ADS-Verzeichnisses server stieg dabei nur um ca. 25%, die Last des NDS-Verzeichnisses server (DirXML-Engine) jedoch um bis zu 80%.

Die gemessenen Werte sollen nur als Beispiele dienen, sind sie doch unter anderem stark von der eingesetzten Hardware und der Netzwerkanbindung abhängig. Die beiden Verzeichnisse server waren über eine 100Mb Verbindung miteinander verknüpft.

In einer späteren Produktivumgebung sind eine solch hohe Zahl von „gleichzeitigen“ Änderungen zwar nicht zu erwarten, dennoch sollte ein extra NDS-Verzeichnisse server (vergleiche Implementierung) für die DirXML-Engine zur Verfügung gestellt werden. Damit wird sichergestellt, dass Änderungen möglichst zeitnah verarbeitet werden.

6.3 Testergebnisse der Passwortsynchronisation

Die Überprüfung der Passwortsynchronisation erfolgte in einer eigenständigen Gruppe von Tests. Es wurden verschiedene Szenarien der Passwortänderung und Passworteinrichtung für neue Nutzerobjekte untersucht. Zusammenfassend hier ein Überblick der Ergebnisse:

- Eine Abgleich der Passwörter ist erst nach erfolgter Assoziation der Nutzerobjekte (durch DirXML) zwischen ADS und NDS möglich.
- Die in den Verzeichnissen abgespeicherten Passwörter können **nicht** synchronisiert werden. Passwörter werden nur bei Passwortänderungen (nach erfolgter Assoziation der Nutzerobjekte) durch den Passwortagenten synchronisiert.
- Es werden alle Passwortänderungen berücksichtigt, welche nicht über den iManger oder der ConsoleOne (auf dem Server) durchgeführt werden. Die Änderungen dieser Programme werden direkt zum NDS-Verzeichnis übertragen. Im Gegensatz dazu werden Passwortänderungen durch die Microsoft Management Console auf einem ADS-Domänencontroller berücksichtigt.
- Der Passwortagent registriert Passwortänderungen aller Nutzer in den Verzeichnisdiensten und versucht diese abzugleichen. Bei fehlender Assoziation verwirft der PasswordSync Agent nach einem definierbaren Wiederholungszeitraum die Passwortänderung.
- Vor der Installation bestehende Nutzerobjekte können nachträglich automatisch mit Hilfe von DirXML verknüpft werden.

Aufgrund der Ergebnisse sind Änderungen an den bestehenden Einrichtungsroutinen der RZ-NDS unumgänglich. Momentan werden Nutzerobjekte zusammen mit den zugehörigen Passwörtern eingerichtet. Bei einer Implementierung von DirXML wären zum Zeitpunkt der Einrichtung der NDS-Nutzerobjekte noch keine korrespondierenden ADS-Nutzerobjekte vorhanden. Demnach existieren auch noch keine Assoziationen, welche für eine Passwortsynchronisation benötigt werden. DirXML kann die NDS-Nutzerobjekte erst nach der Einrichtung weiterverarbeiten, das Passwort ist zu diesem Zeitpunkt aber bereits im Verzeichnis gespeichert.

Diesem Problem kann durch eine veränderte Einrichtungsroutine begegnet werden. In einem ersten Durchgang der Einrichtung werden die Nutzerobjekte mit einem zufälligen Initialpasswort eingerichtet. Da DirXML die NDS-Daten ereignisgesteuert verarbeitet, werden kurze Zeit später die Nutzerobjekte in der ADS angelegt und die Assoziation zwischen den beiden Nutzerobjekten hergestellt. In einem zweiten Einrichtungsdurchgang können dann die „richtigen“ Passwörter den NDS-Nutzerobjekten zugetragen werden. Durch die nun

bestehende Verknüpfung von ADS- und NDS-Nutzerobjekten werden die Passwörter in beiden Systemen eingerichtet. Die Zeitspanne zwischen dem ersten und zweiten Durchgang der Nutzereinrichtung sollte nicht weniger als dem doppelten der Synchronisationsfrequenz betragen. Zwar verarbeitet der DirXML Treiber die Änderungen der NDS ereignisgesteuert, eine definierte Zeitspanne, in der alle Änderungen verarbeitet worden sind, kann jedoch nicht bestimmt werden.

7 Zusammenfassung

Ziel dieser Diplomarbeit war die Betrachtung und Umsetzung einer Synchronisation von Verzeichnisdiensten am Beispiel von Novell NDS und Microsoft ADS.

Im ersten Teil wurden zunächst die grundlegenden Eigenschaften der als Beispiel verwendeten Verzeichnisdienste betrachtet. Daraus ging hervor, dass es zu diesem Zeitpunkt nicht möglich ist, Verzeichnisdienste über die intern verwendeten Replikationsmechanismen zu synchronisieren. Zur Lösung der Problematik wurden Aspekte einer Synchronisation zwischen Verzeichnisdiensten betrachtet und Lösungsansätze dargestellt.

Am Beispiel der FHTW wurde im zweiten Teil der Diplomarbeit ein Lösungskonzept für die Synchronisierung der Nutzerdaten von ADS und NDS diskutiert. Aufgrund der vorher betrachteten Aspekte und der Anforderungen des Beispiels sowie der Vorteile von Metaverzeichnissen wurde DirXML als offene Synchronisationslösung ausgewählt. Das Ziel eines Abgleiches der beiden Verzeichnisdienste NDS und ADS zur Vereinfachung der Verwaltung von Studentenaccounts wurde anschliessend in der Implementierung umgesetzt.

Die Implementierung mit Hilfe von DirXML umfasste sowohl die Platzierung der DirXML Komponenten als auch die benötigten Regeln und Filter, um einen Informationsfluss zwischen den Verzeichnisdiensten herzustellen. Eine Testumgebung diente der Überprüfung der implementierten Funktionen. Auf die Besonderheiten der Passwortsynchronisation wurde in eigenständigen Testszenarien eingegangen und Lösungen für aufgetretene Probleme aufgezeigt.

Abschliessend betrachtet stellt DirXML die Grundlage für den Aufbau eines zentralen Verzeichnisses bereit. Die Implementierung ist einfach umzusetzen und realitätsnah gestaltet. Einzig die Passwortsynchronisation erfordert Änderungen an den momentan vorhandenen Organisationsabläufen einer Nutzereinrichtung. Der Lebenszyklus von Nutzerobjekten in den beiden verwendeten Verzeichnisdiensten kann nun zentral gesteuert werden.

8 Ausblick

Durch die immer grösser werdende Zahl von eingesetzten Verzeichnisdiensten mit zumeist identischen Informationen wird die Frage der Synchronisation der Verzeichnisdienste immer bedeutender. Sie bietet Unternehmen ein Einsparpotential durch die Vereinfachung von Administration und Informationsflüssen und der Schaffung einer besseren Informationsqualität.

Solange kein standardisierter Replikationsmechanismus für den Abgleich von Verzeichnisdaten verfügbar ist, wird der Markt für Metaverzeichnisse sich vergrössern. Dabei zeichnet sich der Trend der Integration von Metaverzeichnisfunktionalitäten in normale Verzeichnisdienste ab. So ist DirXML von Novell seit neuestem ein integraler Bestandteil der eDirectory.

Abzuwarten bleibt auch die Entwicklung von DSML. Abhängig von der Unterstützung durch Verzeichnisdienste und anderen Applikationen wird sich DSML aufgrund des offenen XML-Ansatzes weiter durchsetzen.

Die in der Diplomarbeit vorgestellte Implementierung auf Basis eines Metaverzeichnisses enthält Möglichkeiten für Erweiterungen. So kann mit Hilfe eines Textkonnektors für DirXML die zentrale RZ-Logindatenbank an das Metaverzeichnis angebunden werden. Auch die Synchronisation weiterer Nutzerobjekte, wie z.B. Professoren und Mitarbeiter des Fachbereiches 1, ist möglich. Als Vergleichsattribut dieser bestehenden Nutzerobjekte kann dafür z.B. der Vor- und Nachname verwendet werden.

Weiterhin könnte der Filter des implementierten Active Directory Treiber durch die Auswertung der Änderungsereignisse im Transformation Stylesheet des Subscriberkanals verbessert werden. Somit müssten nicht alle Nutzerobjekte des zu synchronisierenden NDS-Containers auf einen Abgleichbedarf in der DirXML-Engine untersucht werden.

Mit dem Ziel einer zentralen Verwaltung von Nutzerdaten kann das NDS-Verzeichnis im Rechenzentrum die Anbindung weiterer bestehender Datenquellen, wie z.B. des LDAP-Verzeichnisses im Rechenzentrums oder auch der Immatrikulationsdatenbank, umgesetzt werden.

Literaturverzeichnis

- [ADL00] Step-by-Step Guide to Bulk Import and Export to Active Directory / Microsoft. 2000. – Techinfo. <http://www.microsoft.com/windows2000/techinfo/planning/activedirectory/bulksteps.asp>
- [ADS00] Active Directory Interoperability and Metadirectory Overview / Microsoft Corporation. 2000. – White Paper. <http://www.microsoft.com/windows2000/techinfo/howitworks/activedirectory/identity.asp>
- [Bow00] BOWSTREET: Sun Microsystems, other tech pioneers advance DSML in directory-enabled e-business products / bowstreet. 2000. – Press Release
- [BS02] BETH SHERESH, Doug S.: *Understanding Directory Services*. Indianapolis : System Research Corporation, 2002
- [CA03] CHRIS APPLE, John S.: LDAP Duplication/Replication/Update Protocols (ldup) Charter / Internet Engineering Task Force (IETF). 2003. – Arbeitsgruppeninhalt. <http://www.ietf.org/html.charters/ldup-charter.html>
- [Car98] CARR, Allen: *Endlich Nichtraucher!* München : Wilhelm Goldmann Verlag, 1998
- [Cou01] COULOURIS, Kindberg: *Verteilte Systeme – Konzepte und Design*. München : Addison-Wesley, 2001
- [CP03] Critical Path Meta-Directory Server / Critical Path. 2003. – Produktübersicht. <http://www.cp.net/solutions/wirelessSP/dataDirectoryIntegration/metaDirectoryServer.jsp>
- [Don03] DONLEY, Clayton: *LDAP Programming, Management and Integration*. Greenwich : Manning, 2003
- [Enc03] ENCK, John: Magic Quadrant for Metadirectory Products, 2H03 / Gartner Research. 2003. – Research Note. Nr. M-21-0058

- [Goo00] GOOD, G.: *RFC 2849: The LDAP Data Interchange Format (LDIF)*. Juni 2000. – <http://www.ietf.org/rfc/rfc2849.txt>
- [Gre02] GREENBLATT, Bruce: *LDAP-ENABLED APPLICATIONS with Microsoft's Active Directory and Novell's NDS*. Upper Saddle River, New Jersey : Prentice Hall, 2002
- [Han98] HANNOVER, RRZN / U.: *Windows 2000 Server Netzwerkadministration*. Nackenheim : Herdt-Verlag für Bildungsmedien GmbH, 1998
- [HM02] HODGES, J. ; MORGAN, R.: *RFC 3377: Lightweight Directory Access Protocol (v3): Technical Specification*. September 2002. – <http://www.ietf.org/rfc/rfc3377.txt>
- [Hol02] HOLLÄNDER, Markus: *Meta Directory – Die Königsklasse auch von Microsoft? / ComConsult*. 2002. – White Paper
- [HR83] HÄRDER, Theo ; REUTER, Andreas: *Principles of Transaction-Oriented Database Recovery*. In: *ACM Computing Surveys* 15 (1983), Nr. 4, S. 287–317
- [HS95] HOWES, T. ; SMITH, M.: *RFC 1823: The LDAP Application Program Interface*. August 1995. – <http://www.ietf.org/rfc/rfc1823.txt>
- [HSB91] HOWES, T. ; SMITH, M. ; BEECHER, B.: *RFC 1249: DIXIE Protocol Specification*. August 1991. – <http://www.ietf.org/rfc/rfc1249.txt>
- [Kla01] KLASSEN, Norbert. *Directory Services for Linux in comparison with Novell NDS and Microsoft Active Directory*. August 2001
- [Kuo02] KUO, Peter: *Novell's Guide to DirXML*. Provo : Novell Press, 2002
- [LB00] LAURA BURRIS, ...: *Microsoft Windows 2000 Server – Die technische Referenz: Verteilte Systeme*. Unterschleißheim : Microsoft Press Deutschland, 2000
- [Loc93] LOCKEMANN, Krumm: *Telekommunikation und Datenhaltung*. München : Hanser Verlag, 1993
- [Mic03] Microsoft Corporation: *Polling for Changes Using the DirSync Control*. Oktober 2003. – http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ad/ad/polling_for_changes_using_the_dirsync_control.asp
- [MR99] MATTHIAS REINWARTH, Klaus S.: *Verzeichnisdienste in unternehmensweiten TK- und DV-Infrastrukturen*. Berlin : VDE Verlag, 1999

- [MSD00a] MSDSS Deployment: Implementing Synchronization and Migration / Microsoft Corporation. 2000. – White Paper
- [MSD00b] MSDSS Deployment: Understanding Synchronization and Migration / Microsoft Corporation. 2000. – White Paper
- [Nov02] NOVELL: Novell DirXML – Technical White Paper / Novell. 2002. – White Paper
- [Nov03a] Novell: *DirXML 1.1a Administration Guide*. Juni 2003. – http://www.novell.com/documentation/lg/dirxml11a/pdfdoc/dirxml11a_admin_guide.pdf
- [Nov03b] Novell: *DirXML Driver for Active Directory 2.0a*. September 2003. – http://www.novell.com/documentation/lg/dirxmldrivers/pdfdoc/active_directory_driver_implementation.pdf
- [Nov03c] Novell: *DirXML Starter Pack Installation Guide for Netware 6.5*. Juli 2003. – <http://www.novell.com/documentation/lg/dirxmlstarterpack/pdfdoc/jetset.pdf>
- [Nov03d] Novell: *eDirectory for the Beginner*. Oktober 2003. – <http://developer.novell.com/education/tutorials/edirectory/edirec24.htm>
- [Nov03e] Novell: *Novell Import Conversion Export Utility*. Oktober 2003. – <http://www.novell.com/documentation/lg/dirxml10/index.html?page=/documentation/lg/dirxml10/taoenu/data/a5hgmmu.html>
- [Nov03f] Novell: *Understanding PasswordSync*. Oktober 2003. – <http://www.novell.com/documentation/lg/pwdsync10/passsync/data/adtyhxw.html>
- [Oas02] OASIS: Directory Services Markup Language (DSML) / OASIS. 2002. – Technical Report. <http://www.oasis-open.org/cover/dsml.html>
- [Ros91] ROSE, M. T.: *RFC 1202: Directory Assistance service*. Februar 1991. – <http://www.ietf.org/rfc/rfc1202.txt>
- [Sti03] STIEL, Hadi: Verzeichnisintegration per Metadirectory. In: *Network Computing* (2003), Nr. 20, S. 46–53
- [Sun03a] Sun: *iPlanet Meta-Directory v 5.0 Configuration and Administration Guide*. Oktober 2003. – <http://docs.sun.com/source/816-6093-10/connrule.htm>
- [Sun03b] Sun: *JNDI Service Providers*. Oktober 2003. – <http://java.sun.com/products/jndi/serviceproviders.html>

- [SWMH00] STOKES, E. ; WEISER, R. ; MOATS, R. ; HUBER, R.: *RFC 3384: Lightweight Directory Access Protocol (version 3) Replication Requirements*. Oktober 2000.
– <http://www.ietf.org/rfc/rfc3384.txt>
- [TAHP03] TIMOTHY A. HOWES PH.D., Gordon S. G.: *Understanding and Deploying LDAP Directory Services*. Boston : Addison-Wesley, 2003
- [Urb03] URBAN, Prof. Dr. R.: *Betriebssysteme – Verteilte Systeme*. Westfälische Hochschule Zwickau (FH), Oktober 2003
- [VH94] VASSOS HADZILACOS, Sam T.: *A Modular Approach to Fault-Tolerant Broadcasts and Related Problems* / Department of Computer Science, Cornell University, Ithaca NY. 1994. – Technical Report TR94-1425
- [Zho99] ZHOU, Tao: *Directory Integration and the Metadirectory*. In: *Windows & .net Magazine* (1999), Juli

A Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Berlin, 12. Januar 2004

Taito Radtke