Bachelorarbeit

Effiziente Software-Verteilung in heterogenen Umgebungen

von Daniel Schreiner

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Daniel Schreiner

Effiziente Software-Verteilung in heterogenen Umgebungen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Angewandte Informatik
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.-Ing. Martin Hübner
Zweitgutachter : Prof. Dr. rer. nat. Gunter Klemke

Abgegeben am 28.01.2008

Thema der Bachelorarbeit

Effiziente Software-Verteilung in heterogenen Umgebungen

Stichworte

Desktop-Management, Software-Management, Asset-Management, Web-Service, Axis, WSS4J

Kurzzusammenfassung

Die Kosten einer Netzwerk-Arbeitsstation über ihre Lebenszeit übersteigen durch vorgenommene administrative Tätigkeiten die Anschaffungskosten um ein Vielfaches. Durch immer kürzere Innovations-Zyklen in der Software-Entwicklung steigt der administrative Aufwand durch die Aktualisierung der installierten Software stetig. Um diesen zu senken, ist eine Werkzeug-gestützte Verwaltung der Software-Bestände nötig. Da jedes Computer- und Netzwerk-System individuell aufgebaut ist, müssen sich die Verwaltungs-Werkzeuge an die Systeme anpassen können.

In dieser Arbeit werden bestehende Architekturen und Systeme zur Verwaltung von Arbeitsstationen untersucht und anhand der Ergebnisse und der beispielhaften Anforderungen einer mittelständischen Firma ein Konzept und eine Beispiel-Implementation für ein System zur Software-Verteilung und Inventar-Erfassung erstellt.

Title of the paper

Efficient Software-Deployment in heterogeneous Environments

Keywords

Desktop-Management, Software-Management, Asset-Management, Web-Service, Axis, WSS4J

Abstract

Throughout the life cycle of a networked computer administrative measures result in a total running cost that hugely exceeds the cost of purchase.

Due to ever shorter innovation cycles in software development the time and effort spent on keeping the installed software up-to-date rises continually. A tool-based management of the softwary inventory is necessary to decrease this effort. These management tools need to adapt to the system at hand since, typically, every two systems differ in their configuration.

In this work we survey the existing architecture and administration tools. Building on this survey and the requirements of a medium-sized company we develop an outline and a proof-of-concept implementation of a tool that deploys software across a network and takes stock of the installed software and hardware.

Danksagung

An dieser Stelle möchte ich meiner Mutter und Katrin für das „Mitfiebern“ danken und dafür, das ihr immer da seid, wenn man euch braucht. Einen weiteren Dank gebührt meiner Mathematik- und LaTeX-Hotline Henning, da ich ohne dich nicht an diesem Punkt angelangt wäre.

Des weiteren möchte ich Prof. Dr.-Ing. Hübner für seine wertvollen Anregungen und die hervorragende Kommunikation danken.

Inhaltsverzeichnis

1. Einleitung **7**
- 1.1. Motivation . . . 7
- 1.2. Ziele der Arbeit . . . 7
- 1.3. Aufbau der Arbeit . . . 8

2. Grundlagen **9**
- 2.1. Der Management-Begriff . . . 9
- 2.2. Sichten auf das Management . . . 10
- 2.3. Das Management-System . . . 16

3. Anforderungsanalyse **19**
- 3.1. System-Umgebung . . . 19
- 3.2. Stand des Desktop-Managements . . . 20
- 3.3. Anwendungsfälle . . . 22
- 3.4. Fachliche Anforderungen . . . 25
- 3.5. Technische Anforderungen . . . 26

4. Architekturen und Lösungen **29**
- 4.1. Management-Architekturen und -Modelle . . . 29
- 4.2. Marktübersicht . . . 41

5. Design **45**
- 5.1. Organisationsmodell . . . 45
- 5.2. Kommunikationsmodell . . . 46
- 5.3. Funktionsmodell . . . 47
- 5.4. Informationsmodell . . . 53

6. Realisierung **57**
- 6.1. Eingesetzte Technologien . . . 57
- 6.2. Implementation des Agenten . . . 63
- 6.3. Implementation des Gateways . . . 70
- 6.4. Implementation des Informationsmodells der Management-Datenbank . . . 79

7. Schluss **81**
7.1. Zusammenfassung 81
7.2. Ausblick 82

A. Screenshots der entwickelten Anwendung **84**

B. Code-Beispiele **95**

C. Installationshinweise **99**

D. Konfigurationshinweise **101**

E. Inhalt der beigefügten CD **105**

Literaturverzeichnis **106**

Verzeichnis der RFCs und Standards **109**

Glossar **113**

Abbildungsverzeichnis **118**

1. Einleitung

1.1. Motivation

Diese Arbeit entsteht auf Anregung eines mittelständischen Unternehmens der Bauingenieur-Branche. Dieses wird im weiteren Verlauf der Arbeit als *Bau GmbH* bezeichnet. Es besteht aus drei voneinander unabhängigen Standorten. Der Gründungsstandort Hamburg beschäftigt mittlerweile 120 Ingenieure, Konstrukteure und kaufmännisches Personal. Dazu kommt noch ein IT-Team bestehend aus 3 Mitarbeiter, die für die Planung, Modernisierung und den reibungslosen Betrieb der bestehenden IT-Infrastruktur verantwortlich sind.

Das System besteht aus ca. 120 Arbeitsstationen, die mit Microsoft Windows XP Professional als Betriebssystem arbeiten, und ca. 20 Servern die hauptsächlich mit einem auf Linux basierenden Betriebssystem arbeiten. Das System besteht sowohl aus einem statischen Netzwerk, als auch aus Clients, die dynamisch (z. B. Baustellen-PCs, Notebooks, Arbeitsplätze bei Kunden im Fall von Langzeit-Projekten) angebunden sind. Das Netzwerksystem besteht aus einer Domänen-Struktur, die durch Samba-Server realisiert wird. Der Verzeichnis-Dienst wird durch OpenLDAP-Server bereitgestellt.

Die Vernetzung der Standorte wird vorangetrieben und die Aktualisierungs-Zyklen der eingesetzten Software werden kürzer. Die Anzahl der eingesetzten Technologien steigt. Damit wächst der Umfang der Aufgaben des IT-Teams stetig. Deshalb wird das Ziel verfolgt, die aufgewandte Zeit für die allgemeinen operativen Tätigkeiten auf ein Minimum zu reduzieren. Hierfür werden geeignete Werkzeuge gesucht.

1.2. Ziele der Arbeit

Der primäre Angriffspunkt ist eine möglichst vollständige Unterstützung des Desktop-Managements der Clients. Diese soll vornehmlich die Aktualisierung der eingesetzten Softwarepakete übernehmen und die Administratoren bei der Inventarisierung der Clients unterstützen. Änderungen an der Konfiguration des Clients müssen leicht und übersichtlich vorgenommen werden können und der Erfolg der vorgenommenen Änderungen soll überwacht werden können. Dabei soll der gesamte Lebenszyklus des Clients abgedeckt werden,

d. h. der Installations-Stand soll möglichst von der erstmaligen Installation des Betriebssystems über die Installation der Anwendungssoftware bis hin zur Ausmusterung nach Erreichen seiner Abschreibungszeit überwacht und kontrolliert werden können.

1.3. Aufbau der Arbeit

In dieser Arbeit soll analysiert werden, durch die werkzeuggestützte Unterstützung welcher der beschriebenen Aufgaben, in Hinsicht auf die *Bau GmbH*, der grösste Nutzen entsteht. Des weiteren soll eine Analyse einiger existierender Softwarelösungen deren Vor- und Nachteile aufzeigen. Schlussendlich soll eine eigene Implementierung der analysierten Kernkomponenten erstellt werden, die die gefundenen Schwachstellen der kommerziellen Lösungen umgeht, bzw. verbessert.

Die Arbeit gliedert sich dabei in folgende Bereiche:

Kapitel 2 : In Kapitel 2 wird ein Überblick über die Konzepte gegeben, die sich hinter dem Begriff *Management*, bzw. *Management-System* verbergen. Des weiteren wird eine Einordnung des Themas dieser Arbeit vorgenommen.

Kapitel 3 : In Kapitel 3 werden Hintergrundinformationen über die System-Landschaft der *Bau GmbH* gegeben und anhand der Problemstellung die Anforderungen an eine eigene Lösung ermittelt.

Kapitel 4 : In Kapitel 4 werden einige Architekturen, die für die Lösung der Problemstellung konzipiert wurden, beschrieben. Eine kurze Marktübersicht stellt bestehende Produkte vor.

Kapitel 5 : In Kapitel 5 wird ein Design zur Erstellung einer eigenen Lösung vorgestellt.

Kapitel 6 : In Kapitel 6 wird die Realisierung eines Prototypen nach dem im Kapitel 5 beschriebenen Design beschrieben.

Kapitel 7 : In Kapitel 7 wird letztendlich eine Zusammenfassung der Arbeit und ein Ausblick auf mögliche Weiterentwicklungen gegeben.

2. Grundlagen

Die Verwaltung heterogener Computer-Netzwerke besteht aus verschiedenen Aspekten, die eine Strukturierung der in ihr enthaltenen Aufgaben erfordert. In diesem Kapitel werden zunächst die zugrunde liegenden Begriffe und Konzepte vorgestellt, die für die Klassifizierung der Verwaltung von heterogenen Computer-Netzwerken entworfen worden sind. Anschließend werden die Struktur und die typischen Komponenten eines Verwaltungs-Systems aufgezeigt.

2.1. Der Management-Begriff

Für das weitere Vorgehen ist es nötig, zu klären was man unter dem Management-Begriff versteht. Es muss geklärt werden, was verwaltet werden kann und welche Voraussetzungen dafür nötig sind.

Das Management vernetzter Systeme bezeichnet die systemweite Verwaltung der in ihm enthaltenen Komponenten. Die Komponenten können dabei sowohl physikalisch erfassbar und messbar(wie z. B. Hardware-Komponenten und Datenverkehr), als auch abstrakt(wie z. B. Benutzer, Software, Dienste) sein. Für eine möglichst umfassende Verwaltung müssen alle im System vorhandenen Ressourcen durch das Management abgebildet werden können. Verwaltet werden dabei die Lokalisierung, die Konfiguration, die eingesetzten Dienste und Übertragungsprotokolle, die Zugriffsrechte, die Überwachung der Funktionalität und die Wiederherstellung der Funktionalität im Fehlerfall. Die Verwaltung soll dabei gliederbar sein, d. h. die vorhandene Domänen-Struktur muss abgebildet und erweitert werden können(z. B. durch Aspekte der Organisations-Struktur, Rollen oder geographische Strukturen). Es müssen strukturierte Gruppen gebildet werden, denen unterschiedliche, klar definierte, Aufgaben zugeteilt werden können. Die gebildeten Teilaspekte müssen über wohldefinierte Schnittstellen in der Lage sein, die produzierten und erhaltenen Informationen untereinander aufzuteilen, s. d. das System ganzheitlich agieren kann.

Der Begriff der Verwaltung bezieht sich nicht nur auf die technische Infrastruktur, sondern umfasst auch den organisatorischen Aspekt und damit die vollständige Eingliederung in die betriebliche Struktur. So müssen z. B. Management-Prozesse gebildet werden, um die Geschäftsprozesse zu unterstützen. Dafür muss für die Verwaltung eine personelle Rollen-

und Aufgabeneinteilung stattfinden, die die bestehende Struktur der Organisation in der Verwaltung widerspiegelt.

Das Ziel des Managements ist es, die Effizienz der betrieblichen Datenverarbeitungs-Ressourcen zu maximieren und den erbrachten Zeitaufwand der mit dem Management betrauten Mitarbeiter zu minimieren.

Die Komplexität des zu verwaltenden Systems ist abhängig von verschiedenen Faktoren, wie z. B. den Anforderungen an die Verfügbarkeit, den Kosten, der geforderten Flexibilität des Systems in Bezug auf Änderungen und der Komplexität der eingesetzten Komponenten, d. h. deren Heterogenität. Dies muss durch das Verwaltungssystem erfasst werden können. Mit steigender System-Komplexität, steigt daher auch die Komplexität, bzw. die Anforderungen, an die Verwaltung und deren Bedienungspersonals (Hegering et al., 1999, S. 86f).

2.2. Sichten auf das Management

Die Komplexität der Anforderungen an die Verwaltung und deren Abbildung auf die Unternehmensstruktur erfordern eine Strukturierung, bzw. Klassifizierung, der einzelnen abzudeckenden Teilbereiche auf die im folgenden eingegangen werden soll.

Das integrierte Management lässt sich horizontal und vertikal in mehrere Ebenen einteilen und in die Unternehmensstruktur eingliedern(Abbildung 2.1). Umso mehr Schichten durch ein Management-System erfasst werden können, desto integrierter ist es. Die Teilbereiche haben dabei unscharfe Grenzen, die sich je nach betrieblichem Umfeld, Detail-Tiefe der Strukturierung und Umfang der angebotenen Dienste verschieben können. So kann ein Systemdienst sowohl lokal oder verteilt realisiert sein; er fällt damit in die Bereiche des System-Managements oder des Anwendungs-Managements(s. u.). Daher differieren die Inhalte und der Umfang der vorgenommenen Strukturierungen in der Literatur. Im folgenden wird sich an der Struktur orientiert, die Hegering et al. (1999) vornimmt.

In den nächsten Abschnitten sollen die Aufgaben und Unterteilungen der einzelnen Schichten genauer beschrieben werden.

2.2.1. Ressourcen-basierte Sicht

Die vertikale Gliederung des integrierten Management(Abbildung 2.1) versucht durch ihre Schichten, die verwalteten Ressourcen zu bündeln. In der vertikalen Gliederung beschreibt das Enterprise-Management die unternehmensweiten Geschäftsprozesse und leitet daraus

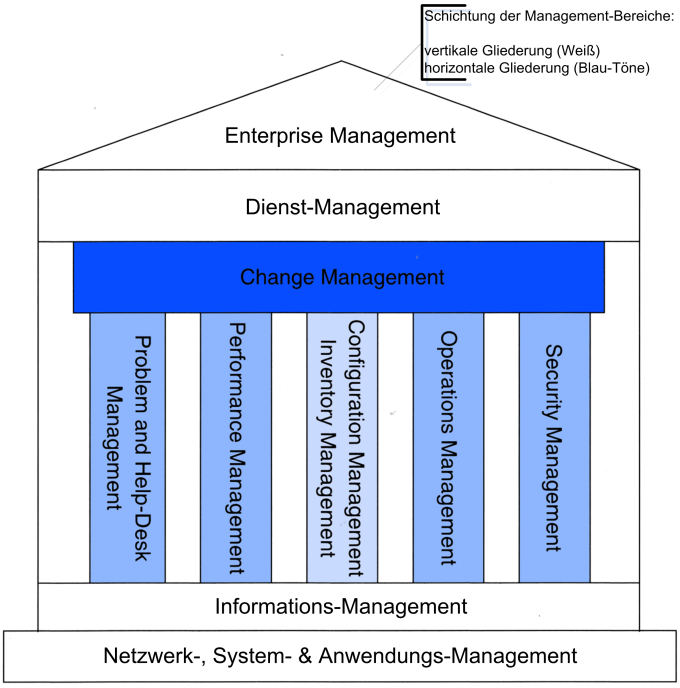


Abbildung 2.1.: Schichtung des Managements(Hegering et al. (1999), abgeändert)

Vorgaben für die betrieblichen Dienste(z. B. Vorgaben für Schnittstellen zu externen Dienstleistern) ab, es bildet damit die Vorgaben für die darunterliegenden Schichten aus. Das Dienste-Management ist für die Verwaltung und Aufstellung von verteilten Diensten zuständig(z. B. das Bereitstellen von Verzeichnis-Diensten). Das Anwendungs-Management verwaltet die eingesetzten verteilten Anwendungen. Für die Verwaltung der bereitgestellten und erarbeiteten Datenbestände ist das Informations-Management zuständig. Das System-Management befasst sich mit der Verwaltung der Ressourcen der Endgeräte(z. B. Software-Verteilung, Lizenzkontrolle, etc.) und das Netzwerk-Management bezieht sich auf die Verwaltung der Netzwerkkomponenten, also der eingesetzten Switches, Router, Übertragungswege, Übertragungsprotokolle, etc..

2.2.2. Funktions-basierte Sicht

Die horizontale Gliederung(Abbildung 2.1) erstellt eine funktionale Bündelung aller verwalteten Ressourcen. Im Vergleich mit der Ressourcen-basierten Sicht sind die einzelnen Teilbereiche daher angewendet auf die zu verwaltenden Ressourcen auf alle Schichten der vertikalen Gliederung gleichermaßen anwendbar. Sie werden unter dem Oberbegriff Änderungs-Management(Change-Management) zusammengefasst. In den folgenden Unterkapiteln werden ihre Aufgaben genauer beschrieben.

Sicherheits-Management

Das Sicherheits-Management(Security-Management) dient der Analyse, Strukturierung, Umsetzung und Überwachung der zu schützenden Bereiche innerhalb eines Betriebes. Das zentrale Ziel ist, die Vertraulichkeit, Integrität und Verfügbarkeit der Bereiche sicher zu stellen.

Zu den Aufgaben des Sicherheits-Managements gehört zunächst das Auffinden der schützenswerten Ressourcen und deren möglichen Bedrohungen. Dies geschieht durch Risiko- und Bedrohungs-Analysen. Die gefundenen Risiken(z. B. Vertraulichkeitsverlust durch passive und aktive Angriffe, Datenverlust durch Funktionsstörungen, Naturkatastrophen, etc.) werden darauffolgend durch die Definition von Sicherheitsregeln minimiert. Schließlich werden die erarbeiteten Sicherheitsregeln(z. B. Passwort-Regeln, Zugriffskontrollen, Verschlüsselung des Netzwerkverkehrs oder der gespeicherten Daten) unternehmensweit in die vorhandenen Dienste integriert und deren Einhaltung überwacht.

Operations-Management

Das Operations-Management bezieht sich auf Vorgänge der kurzfristigen administrativen Tätigkeiten, d. h. Tätigkeiten, die ohne grösseren Planungs- oder Test-Aufwand vorgenommen werden. Es umfasst die Vergabe von Betriebsmitteln(z. B. Kontingentierung von Speicherplatz), die Pflege der Daten in Verzeichnisdiensten(z. B. Namensänderungen, Zuweisen von Benutzerrechten) und die, falls dieses in der Unternehmensstruktur vorgesehen ist, Faktorisierung der entstandenen Kosten auf die entsprechenden Kostenstellen.

Leistungs-Management

Das Leistungs-Management(Performance-Management) dient der Festlegung und Implementation von Anforderungen an die zu erbringende Leistung des Systems. Des weiteren muss es die Erbringung der Leistung überwachen und bei Unterschreitung einer definierten Mindestleistung Benachrichtigungs- und Eingriffsmöglichkeiten zur Verfügung stellen.

Hierfür müssen zunächst die zu verwaltenden Ressourcen und die Mindestanforderung an deren Leistung identifiziert werden. Dann müssen Parameter für die physikalische Messung der geforderten Leistung definiert werden. Daraufhin müssen geeignete Maßnahmen implementiert werden, die die geforderte Leistung aus Sicht der jeweiligen Anwender sicherstellen(Quality of Service - QoS) und bei Unterschreitung zur Wiederherstellung der Leistung führen. Anschließend muss die geforderte und tatsächlich erbrachte Leistung kontinuierlich

gemessen und verglichen werden(z. B. durch Auswertungen von Log-Dateien, Performance-Messungen, etc.). Bei der Unterschreitung der geforderten Leistung müssen die vorher definierten Maßnahmen zur Wiederherstellung der Mindestleistung eingeleitet werden.

Problem- und Helpdesk-Management

Wenn unerwünschte Systemereignisse entstehen, müssen diese zur Sicherung der Systemintegrität möglichst zeitnah erkannt, lokalisiert und behoben werden. Das Problem- und Helpdesk-Management definiert Regeln, Prozesse und Werkzeuge für das Erkennen und Lokalisieren von abnormem Systemverhalten und spezifiziert Maßnahmen zur Wiederherstellung eines normalen Systembetriebes (Hegering et al., 1999, S. 79).

Zunächst müssen Regeln für die Verfügbarkeit der einzelnen System-Komponenten, die sog. Service Level Agreements(SLAs), aufgestellt werden. Diese definieren, welche Komponenten für wie lange nicht verfügbar sein dürfen. Sie nehmen also eine Priorisierung der Komponenten vor, indem sie die einzelnen System-Komponenten nach ihrer maximalen Ausfallzeit ordnen. Die maximal erlaubte Ausfallzeit nimmt dabei reziproportional zu der Einstufung, wie geschäftskritisch die Komponente ist, ab. Die definierten Regeln müssen daraufhin durch geeignete Maßnahmen in dem System umgesetzt werden. Dies geschieht z. B. direkt durch den Anwender oder durch Einführung von Werkzeugen zur Überwachung der Verfügbarkeit des Systems(z. B. Netzwerküberwachungs- und Hardwareüberachungs-Werkzeuge). Diese müssen die erkannten Probleme, möglichst noch vor einem Totalausfall, an eine geeignete Stelle weiterleiten. Im Falle einer zentralen Ansprechstelle(Single Point of Contact - SPOC), ist dies der Helpdesk, ansonsten ist dies ein dafür zuständiger Techniker oder dessen Abteilung. Die beauftragte Instanz ist dann angehalten, den Dienst innerhalb der SLA wiederherzustellen oder, wenn dieses nicht möglich ist, das Problem an eine entsprechende, vom Problem-Management vorgegebene, übergeordnete Stelle zu weiter zu geben. Man spricht dann von einer Eskalation des Problems. Alle erkannten Probleme müssen in einer Datenbank, dem Trouble-Ticket-System, gespeichert werden. Die gespeicherten Daten umfassen die Problemstellung, die involvierten Ressourcen, eine Historie der vorgenommenen Aktionen und die letztendliche Problemlösung.

Konfigurations- und Inventar-Management

Das Konfigurations-Management(Configuration-Management) ist verantwortlich für die Aufstellung, Änderung und Kontrolle der Konfiguration der System-Komponenten. Als Konfiguration des Systems bezeichnet man, abhängig vom Kontext, entweder eine Beschreibung des Systems und seiner Bestandteile, den Vorgang der Änderung von Einstellungen an einer

Komponente oder dessen Ergebnis, d. h. die Menge von Parametern, die die Einstellung einer Komponente definieren (Hegering et al., 1999, S. 76f).

Das Konfigurations-Management hat u. a. die Aufgabe, die Lokalisation der Konfiguration festzulegen. In einem vernetzten System kann es von großem Vorteil sein, wenn die Konfiguration zentral gespeichert, verwaltet und geändert werden kann. Änderungen werden somit z. B. durch einmalige Ausführung auf alle angeschlossenen Arbeitsstationen propagiert(z. B. Gruppenrichtlinien im Active Directory oder Thin-Clients, die auf einen zentralen Terminal-Server zugreifen). Des weiteren ist der Speicherort der Konfiguration von Bedeutung. Dies hat sowohl Relevanz im Hinblick auf die Verfügbarkeit(z. B. bei zentraler Konfiguration im Falle eines Netzwerk-Ausfalls), als auch auf die Sicherheit der Konfiguration(z. B. erhöhte physikalische Zugriffssicherheit durch Lagerung in einem verschlossenen Server-Raum). Zur Planung gehört außerdem die Festlegung der Konfigurationsart. Man unterscheidet zwischen statischer Konfiguration, bei der Änderungen an der Konfiguration nicht ohne Neuinitialisierung der entsprechenden Komponente übernommen werden, oder dynamischer Konfiguration, die die entsprechenden Änderungen ohne Neuinitialisierung übernehmen.

Das Konfigurations-Management muss Prozesse und Werkzeuge zur Verfügung stellen, die Änderungen der Konfiguration ermöglichen, also die Konfigurationsparameter ändern können. Für die Kontrolle der Konfiguration und der vorgenommenen Konfigurationsänderungen müssen Werkzeuge zur Verfügung stehen, die die Konfigurationsparameter abfragen können. Sowohl die vorhandene Konfiguration, als auch die vorgenommenen Änderungen an ihr müssen in einer Konfigurations-Datenbank gespeichert werden. Damit wird das Führen einer Historie, sowie die Erstellung von Visualisierungen und Dokumentationen des vernetzten Systems unterstützt.

Die komplexen Aufgaben des Konfigurations-Management lassen sich in Unterbereiche aufteilen. Das Anwendungs-Management(Application-Management) dient der Konfiguration der Anwendungs-Ressourcen, also deren Installation, Deinstallation und Konfiguration. Das Inventar-Management(Inventory-Management) dient der Auflistung der vorhandenen Ressourcen und deren Betriebsparameter. Eine spezielle Form des Inventar-Managements ist durch das Asset-Management gegeben, welches zusätzlich Informationen betreffend betriebswirtschaftlicher Aspekte(z. B. Anschaffungskosten, Betriebskosten, Abschreibungszeiten, etc.) der Komponenten bereitstellt.

2.2.3. Andere Sichten

Neben den bereits vorgestellten Sichten, existieren noch diverse andere Sichtweisen, die sich zumeist auf Teilaspekte des Managements fokussieren. Zwei davon sollen an dieser Stelle kurz vorgestellt werden, da sie für die Lösung der Problemstellung hilfreich erscheinen.

Anwendungs-Management

Eine andere Sicht des Managements liefert das Anwendungs-Management(Application-Management), wie es in Baron et al. (2002) beschrieben ist. Es stellt eine Verknüpfung der Funktions-basierten Sicht mit verschiedenen betriebswirtschaftlichen Aspekten dar und dient der Verwaltung von Anwendungen über ihren Lebenszyklus(Lifecycle-Management). Also von der Entwicklung und den Test über die Integration bis hin zur Weiterentwicklung oder der Ausmusterung der Anwendungs-Komponente(Abbildung 2.2). Die Sichtweise hilft bei der Ermittlung der Gesamtkosten einer Anwendung über ihre Lebenszeit.

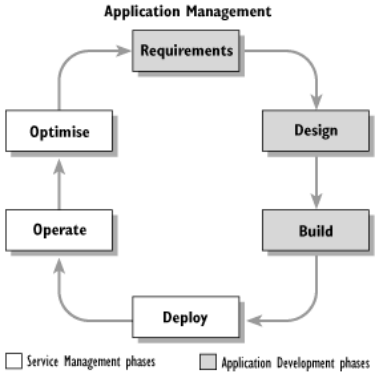


Abbildung 2.2.: Application-Management(aus Baron et al. (2002))

Desktop-Management

Der Begriff Desktop-Management definiert eine Bündelung für verschiedene administrative und operative Aufgaben, die während der Lebenszeit eines Netzwerk-Clients auftreten. Es ist eine Untermenge des PC-Lifecycle-Management(Abbildung 2.3).

Das Desktop-Management definiert eine Sicht auf das funktionsbasierte Management in Abhängigkeit zum Endgerät. Damit wird ermöglicht, dass die Management-Funktion auf die darunterliegenden Geschäftsprozesse abgebildet werden kann. Das Ziel des Desktop-Managements ist dabei die möglichst vollständige Verwaltung des Netzwerk-Clients über seine Lebenszeit, um dessen Gesamtkosten (Total Cost of Ownership - TCO) zu minimieren.

Die Aufgaben auf den Clients umfassen die Installation des Betriebssystems und dessen Aktualisierung, die Installation von Softwarepaketen, die Inventarisierung auf Hardware- und Software-Ebene, die Überwachung der Betriebsbereitschaft, die Erstellung von Nutzungsstatistiken und Möglichkeiten zum ferngesteuerten Eingriff im Fehlerfall.

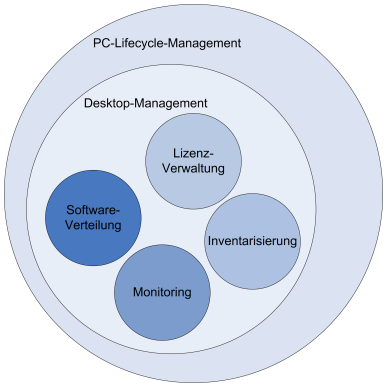


Abbildung 2.3.: PC-Lifecycle-Management

2.3. Das Management-System

Ein Management-System ist ein System, dass alle oder einen Teil der zu verwaltenden Aspekte integriert und über eine einheitliche Sicht zur Verfügung stellt. Durch Abbildung der Management-Prozesse, Integration von Diagnose-Funktionen und deren übersichtlicher Darstellung werden die mit der Aufgabe betrauten Mitarbeiter bei ihrer Arbeit unterstützt und angeleitet. Teilaspekte können automatisiert werden.

Ein Management-System besteht aus mehreren Komponenten denen unterschiedliche Rollen zugeordnet sind(Abbildung 2.4):

1. Der *Management-Controller* dient dem, mit dem Management beauftragten, Mitarbeiter zur Anzeige der erhobenen Informationen. Weiterhin lassen sich durch Eingaben über ihn die Management-Funktionen steuern.
2. Der *Management-Agent* erfasst Management-Informationen an der zu verwaltenden Ressource, bzw. gibt Management-Informationen an sie weiter. Er kann daher, je nach verwalteter Ressource, unterschiedlich beschaffen sein, muss aber die benötigten Schnittstellen zu den anderen Komponenten besitzen.
3. Der *Management-Server* oder *-Gateway* ist die zentrale Komponente des Management-Systems. Er übernimmt die Steuerung und Kontrolle der Management-Vorgänge. Die Steuerungsbefehle erhält der Management-Server vom Controller. Die Ausführung der Befehle erfolgen über den Management-Agent. Die Steuerungsbefehle und Ergebnisse der Ausführung werden in der Management-Datenbank gespeichert. Zur besseren Integration des Management-Systems in die Systemlandschaft sind Schnittstellen zu evtl. anderen bestehenden Management-Systemen nötig.

4. Die *Management-Datenbank* speichert alle für das Management relevanten Informationen. Diese umfassen sowohl vom Management-Agent erhobene Daten(z. B. Inventar-Informationen), als auch vom System bereitgestellte Steuer-Daten(z. B. Installations-Anweisungen).

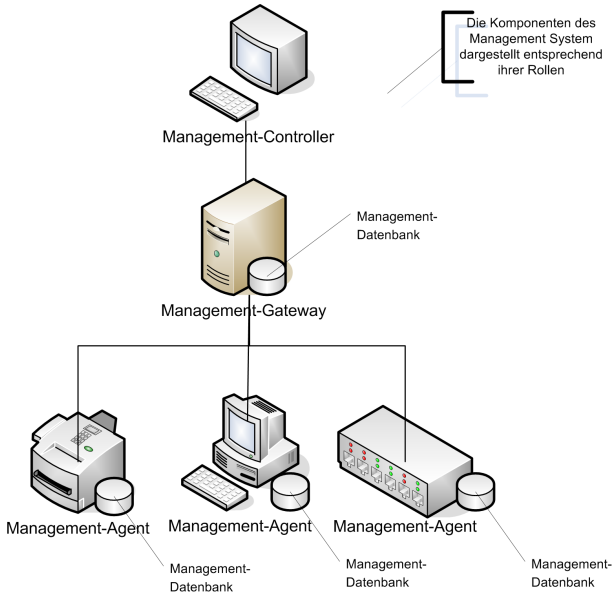


Abbildung 2.4.: Rollen des Management-System

Die Komponenten lassen sich durch Ausbildungen von Organisationsmodellen(s. Kap. 4.1.1) im System anordnen. Somit lassen sich Domänen-Strukturen abbilden und Hierarchien aufbauen, z. B. durch den Einsatz von Management-Agenten die Sub-Agenten abfragen(Abbildung 2.5).

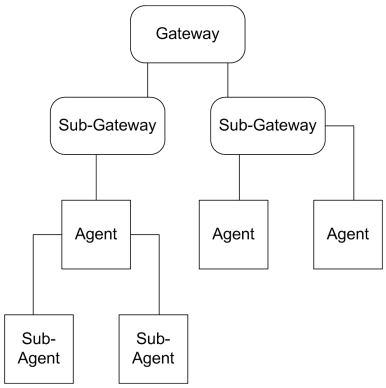


Abbildung 2.5.: Domänen-Bildung im Management-System

In einem vernetzten System spielt die Datenhaltung und -erfassung des Management-Systems

eine besondere Rolle. Sie kann zentral oder dezentral erfolgen und sollte, abhängig von den erhobenen Daten, funktional getrennt werden. So werden z. B. Inventar-Informationen in Inventar-Datenbanken und Konfigurations-Informationen in Konfigurations-Datenbanken gespeichert.

Bei einer dezentralen Datenhaltung befinden sich die Management-Datenbanken direkt, d. h. lokal, an den zu verwaltenden Komponenten. Bei der zentralen Datenhaltung werden die Management-Datenbanken räumlich getrennt zu der zu verwaltenden Komponente vorgehalten, meist an einem zentralen Ort. Beide Speicherarten bergen Vor- und Nachteile.

Die lokale Speicherung hat den Vorteil, dass die abgefragten Informationen stets aktuell sind, da sie bei der Abfrage lokal abgeglichen werden können. Nachteilig ist, dass die Informationen nicht abgefragt werden können, wenn die Komponente über das Netzwerk nicht erreichbar ist(z. B. Notebooks, ausgeschaltete Arbeitsstationen).

Die zentrale Speicherung birgt den Vorteil, dass die Informationen, unabhängig von der Verfügbarkeit der betreffenden Komponente, abgefragt werden können. Nachteilig ist hier, die nicht sichergestellte Aktualität der abgerufenen Informationen.

In der Praxis hat sich daher eine Mischform der Speicherung bewährt. Die Daten werden sowohl lokal als auch zentral vorgehalten(Abbildung 2.4). Ein regelmäßiger Abgleich der Daten sichert dann eine größt mögliche Aktualität der zentralen Datenbank, wodurch bei einzelnen Management-Aufgaben eine wiederholte Abfrage der lokalen Datenbasis und damit eine unnötige Netzwerkbelastung ausbleiben kann.

3. Anforderungsanalyse

Nachdem in dem letzten Kapitel der theoretische Hintergrund der Verwaltung von heterogenen Systemen aufgezeigt wurde, werden in diesem die Anforderungen an ein Management-System(s. Kap. 2.3) beschrieben. Da die Anforderungen anhand technischer und organisatorischer Rahmenbedingungen aufgezeigt werden müssen, sollen sie am Beispiel der *Bau GmbH* ermittelt werden und bilden die Grundlage für das zu entwickelnde System. Hierfür wird zuerst die zu Grunde liegende System-Umgebung beschrieben. Anschließend wird der momentane Stand des Desktop-Managements und dessen Anwendung analysiert. Aus den gewonnenen Erkenntnissen werden dann technische und fachliche Anforderungen an ein zu entwickelndes Management-System ermittelt.

3.1. System-Umgebung

Wie bereits in Kap. 1.1 beschrieben wurde, handelt es sich bei dem vernetzten System der *Bau GmbH* um ein heterogenes System, bestehend aus ca. 120 Arbeitsstationen, die mit Microsoft Windows XP Professional als Betriebssystem ausgerüstet sind, und ca. 20 Servern, die als Betriebssystem hauptsächlich Linux(Debian Sarge und SLES 10), aber auch Microsoft Windows XP Professional oder Microsoft Windows 2000 Server einsetzen. Die Domänen-Struktur wird durch Samba-Server realisiert und durch einen Verzeichnis-Dienst auf Basis von OpenLDAP-Servern bereitgestellt. Das Netzwerk besteht hausintern aus einem kabelgebundenem System. Die Arbeitsstationen sind mit einer Übertragungsgeschwindigkeit von 100MBit an die Switches angebunden, die Server und Switches untereinander arbeiten mit einem Gigabit-Backbone. Externe Arbeitsstationen, die sich bei Kunden oder auf Baustellen befinden, werden über das Internet durch VPN der Zugang zum Netzwerk gewährt. Notebooks realisieren dies über UMTS und VPN.

Die Arbeitsstationen sind, je nach Funktion des benutzenden Mitarbeiters, heterogen in der Hardware- und Software-Konfiguration. Eine Änderung der System-Konfiguration zu einem Terminal-Server-System, welches die Konfiguration homogenisieren würde, ist nicht möglich, da die eingesetzte Konstruktions-Software zu hohe Anforderungen an die Grafikleistung und die Statik-Software zu hohe Anforderungen an die Rechenleistung stellt, um wirtschaftlich auf Terminal-Server-Systemen eingesetzt werden zu können.

Die administrative und operative Verwaltung des Systems erfolgt durch drei Mitarbeiter. Um die benötigte Zeit für die Planung der Einführung neuer Technologien und der Erweiterung des Systems zu erhalten und die Kosten für den Betrieb des Systems zu senken, ist es geboten, die aufgewendete Arbeitszeit für die operative Verwaltung des Systems zu minimieren. Diese wird zur Zeit zu einem großen Teil für das Desktop-Management aufgewendet, insbesondere für die Software-Verteilung und die Asset-Verwaltung. Die Firma Attachmate verweist auf eine Gartner-Studie, die zu dem Ergebnis kommt, dass 80% der Gesamtkosten einer Arbeitsstation nach ihrer Anschaffung entstehen(attachmate, 2007, S. 2 nach Silver (2005)). Attachmate ist in dieser Hinsicht zwar nicht als objektiv anzusehen, da sie selber, als Hersteller von Desktop-Management Produkten, ein wirtschaftliches Interesse am Ergebnis der Studie hat. Des weiteren werden keine genaueren Angaben über die Konfiguration der Arbeitsstationen gemacht. Allerdings decken sich die Beobachtungen tendenziell mit den Ergebnissen der Studie. Das primäre Ziel ist es also, das Desktop-Management so umzugestalten und zu integrieren, dass der zeitliche und personelle Aufwand reduziert wird.

Um eine Verbesserung des Desktop-Managements in dieser Hinsicht zu erlangen, muss zunächst der Ist-Zustand des Systems betrachtet werden. Dieses geschieht im nächsten Abschnitt.

3.2. Stand des Desktop-Managements

Die Verwaltung der Arbeitsstationen erfolgt nicht durch ein integriertes Management-System, d. h. die einzelnen Teilaufgaben(s. Kap. 2.2.3) erfolgen durch manuelle Eingriffe, bei denen der verwaltende Mitarbeiter nicht oder nur in Teilbereichen durch das System unterstützt wird.

Die Installation des Betriebssystems wird entweder, bei grösseren Chargen an neu-angeschafften Arbeitsstationen, über das Imaging, d. h. spiegeln, bereits installierter Arbeitsstationen oder durch individuelle Installationen vorgenommen. Dies erfordert einen erträglichen Zeitaufwand und erfordert nicht die dauerhafte Aufmerksamkeit des Mitarbeiters. Die Aktualisierung des Betriebssystems erfolgt durch dessen interne Aktualisierungs-Mechanismen über einen WSUS-Server.

Die Überwachung der Betriebsbereitschaft und die Erstellung von Nutzungsstatistiken wird nicht auf den Arbeitsstationen vorgenommen. Im Falle eines Defektes oder einer Fehlfunktion meldet sich der Anwender in der Regel bei den IT-Mitarbeitern, die die Störung dann entweder über ein Fernwartungsprogramm oder vor Ort beseitigen.

Die Inventarisierung auf Hardware- und Software-Ebene, sowie der Assets, erfolgt manuell über die Pflege von verschiedenen Tabellen und Dokumenten. Aufgrund der manuellen Bearbeitung beschränken sich die enthaltenen Informationen auf die wesentlichen Attribute, ohne

große Detail-Tiefe zu erlangen. Problematisch an der Methodik ist die schwierige Synchronisation und hohe Fehleranfälligkeit der Aktualisierung der vorhandenen Inventarisierungs-Informationen. Eine fehlende Detail-Tiefe der Informationen bewirkt, dass häufig die benötigte Information nicht oder nur unvollständig vorliegt und durch alternative Quellen beschafft werden muss.

Die Installation und Aktualisierung von Softwarepaketen erfolgt auf verschiedene Arten. Umfangreiche Pakete werden manuell installiert. Kleinere Softwarepakete werden, abhängig von der Verbindungsgeschwindigkeit, d. h. z. B. nicht bei Verbindungen über UMTS, über ein Anmelde-Script installiert. Gleiches gilt für Konfigurationsänderungen. Diese Methodik birgt mehrere Probleme und Risiken:

1. Hoher personeller und zeitlicher Aufwand durch manuelle Installationen
2. Belastung der Mitarbeiter durch manuelle Installationen oder lange Anmeldezeiten
3. Neue Software- und Konfigurations-Pakete werden erst bei Neuanmeldung ausgerollt. So werden z. B. gesperrte Arbeitsstationen oder Notebooks, die nur in den Schlafmodus versetzt werden, nicht über Konfigurationsänderungen informiert.
4. Schlechte und unübersichtliche Integrationsmöglichkeiten für Ausnahmen und Abbildung der Domänen-bezogenen Richtlinien
5. Unübersichtlicher Installationsstand und dadurch resultierende Mängel im Lizenz-Management
6. Unübersichtliche Versionsinformationen der eingesetzten Software (Stand des Patch-Management)
7. Ungenügende Überwachungs- und Report-Möglichkeiten für Skript-gesteuerte Installationen.

Die Anforderungen an die Lizenzverwaltung sind weniger hoch. Das Betriebssystem und die Standard Anwendungen(z. B. Microsoft Office Produkte) sind alle über Volumen-Lizenzen gedeckt, spezialisierte Anwendungen (z. B. CAD- und Statik-Programme) werden über sog. Dongel-Server mit Netzwerk-Lizenzen versorgt. Dadurch lassen sich Unter-Lizenzierungen, d. h. es stehen weniger Lizenzen zur Verfügung als Lizenz-pflichtige Software eingesetzt wird, vermeiden. Ein Problem stellen Über-Lizenzierungen dar, d. h. die vorhandenen Lizenzen werden nicht ausreichend genutzt und verbrauchen daher unnötig Betriebsmittel. Um dieses Problem zu beheben, müssten Nutzungs-Statistiken der einzelnen Software-Pakete aufgestellt werden.

Um eine Verbesserung des Desktop-Managements zu erlangen, muss mindestens das Asset- und Software-Management durch eine werkzeuggestützte Lösung ersetzt werden. Die Lösung

soll die o. g. Schwachstellen eliminieren und damit die Effizienz des Desktop-Managements erhöhen.

3.3. Anwendungsfälle

Für eine fachlich sinnvolle Einordnung der zu entwickelnden Lösung müssen zunächst die Prozesse herausgestellt werden, die die Lösung unterstützen soll. Diese sollen im folgenden beschrieben werden. Die Prozesse betreffen die Konfigurationsänderungen an Hardware und Software. Die Konfiguration stellt dabei, wie in Kap. 2.2.2 beschrieben, sowohl eine Beschreibung der Menge der Komponenten, als auch derer Parameter dar.

3.3.1. Operative Software-Konfiguration

Zunächst soll der Prozess der operativen Software-Konfiguration(Abbildung 3.1) beschrieben werden. Der Prozess bezieht sich auf ein einzeln auftretendes Ereignis. Er ist also im operativen und nicht im administrativen Bereich anzusiedeln. Eine fehlerhafte Software-Konfiguration kann sowohl ein nicht oder zuviel vorhandenes Software-Paket bedeuten, ein Software-Paket dessen Abhängigkeiten zu anderen Software-Paketen nicht erfüllt werden, als auch ein Software-Paket dessen Einstellungsparameter zu einem fehlerhaften Verhalten führt.

Wenn während des Betriebes ein Fehler an der Software-Konfiguration festgestellt wird, müssen zunächst die entsprechenden Rahmenbedingungen ermittelt werden. Diese werden aus den Inventar-Datenbanken(z. B. installierte Software-Version, Menge der installierten Software-Pakete zur Ermittlung der Abhängigkeiten, etc.) und den Konfigurations-Datenbanken(z. B. Einstellungsparameter, die zum Fehler führten, Menge der Abhängigkeiten zu anderen Software-Paketen) entnommen. Wenn eine Behebung des aufgetretenen Fehlers durch Rekonfiguration nicht möglich ist, muss durch administrative Entscheidungen eine Veränderung der Software-Konfiguration(z. B. durch Anschaffung neuer Software-Pakete) erwirkt werden. Dies ist ein eigenständiger Prozess, der hier nicht detaillierter betrachtet werden muss. Wichtig ist hier ausschließlich, dass die neue Konfiguration in die entsprechenden Datenbanken aufgenommen werden muss. Ist eine Rekonfiguration möglich, so müssen ebenfalls die Konfigurationsänderungen in die entsprechenden Datenbanken eingetragen werden. Dieser Vorgang wird solange wiederholt, bis das System wieder in einem fehlerfreien Zustand ist.

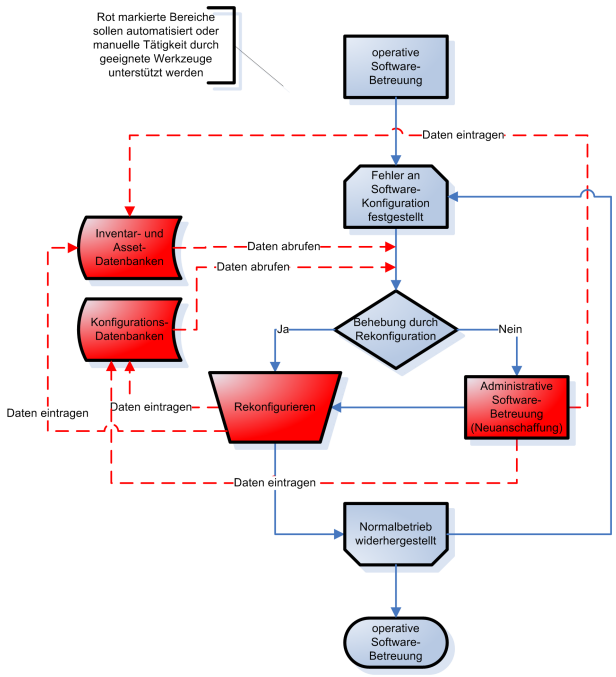


Abbildung 3.1.: Operative Software-Konfiguration

3.3.2. Operative Hardware-Konfiguration

Der Prozess der operativen Hardware-Konfiguration(Abbildung 3.2) bezieht sich auf Konfigurationsänderungen im operativen Bereich, d. h. er findet Anwendung bei individuell auftretenden Anforderungen an Konfigurationsänderungen, bedingt durch Anwenderwünsche(z. B. der Wunsch nach einem Zweit-Monitor) oder Störungen der Hardware.

Wenn ein Fehler an der Hardware-Konfiguration festgestellt wird, muss zunächst entschieden werden, ob es sich um einen Hardware-Defekt, d. h. Totalausfall der Komponente oder Konfigurationsfehler im Sinne der Menge an Komponenten, handelt oder der Fehler durch eine einfache Rekonfiguration, d. h. Einstellung der Konfiguration im Sinn der Menge an Einstellungsparametern, behoben werden kann(s. Kap. 2.2.2). Handelt es sich um einen Defekt, so müssen die entsprechenden Daten der Hardware-Komponente ermittelt werden. Diese werden den Inventar- und Asset-Datenbanken entnommen. Mit diesen Daten ist es dann möglich, die Prozesse anzusprechen, die, entweder durch Reparatur oder Neuanschaffung, zu einer korrekten Konfiguration führen. Diese Prozesse sollen hier nicht genauer beschrieben werden, ihre Ergebnisse, eine eventuell neu angeschaffte oder durch Reparatur veränderte Komponente, müssen in die Inventar- und Asset-Datenbanken aufgenommen werden. Nachfolgend muss die Konfiguration im Sinne der Konfigurations-Datenbank wiederhergestellt werden, bzw. Änderungen an der Konfiguration in der Konfigurations-Datenbank gespeichert werden.

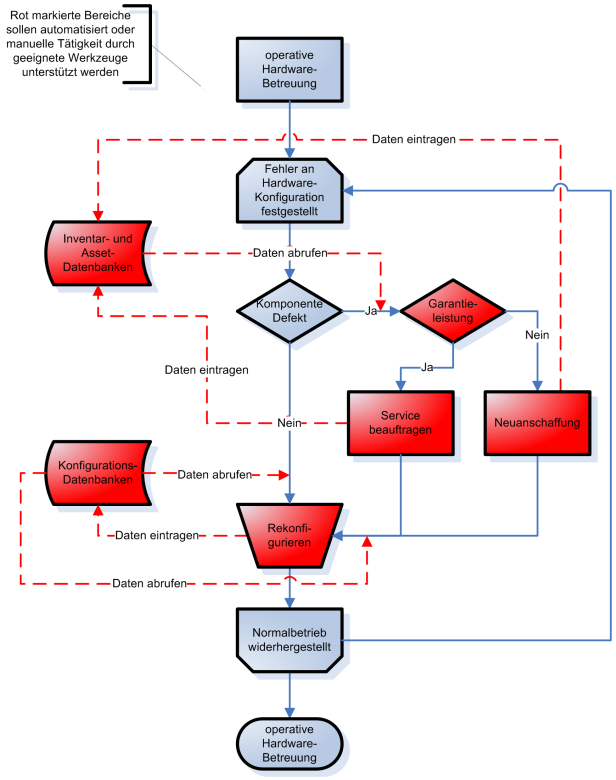


Abbildung 3.2.: Operative Hardware-Konfiguration

3.4. Fachliche Anforderungen

Im folgenden soll aufgezeigt werden, welche spezifischen fachlichen Anforderungen sich für eine Lösung ergeben. Die Anforderungen werden aus den ermittelten Mängeln und Prozessen ermittelt. Sie werden dabei auf die einzelnen Verwaltungskomponenten aufgeteilt, s. d. das Design einer späteren Lösung modular abgestimmt werden kann.

3.4.1. Anforderungen an das Asset-Management

Die Anforderungen an das Asset-Management beziehen sich hauptsächlich auf das Sammeln und zur Verfügung stellen von Informationen und deren statistischer Auswertung. Die mit dem Management beauftragten Mitarbeiter sollen bei ihren Entscheidungen unterstützt werden. Die Sammlung und Auswertung der Datenbasis muss dabei weitgehend automatisiert, d. h. ohne Eingriff der Mitarbeiter, erfolgen und auf Anforderung die, für die Unterstützung der Prozesse benötigten, Ansichten produzieren. Dazu gehören:

- die Übersicht über die eingesetzten Komponenten, d. h. eingesetzte Hard- und Software. Die gesammelten Informationen sollten dabei detailliert sein.
- die Übersicht über Rechnungs-, Garantie- und Reparatur-Informationen
- die Pflege geeigneter Kennzeichnungen(z. B. Führung einer internen Bezeichnung, bzw. Asset-Nr.)
- Such- und Report-Funktionen(z. B. das Auflisten aller Komponenten, die eine bestimmte andere Komponente beinhalten, etc.)

3.4.2. Anforderungen an das Software-Management

Die primären Anforderungen an das Software-Management liegen bei einer einfachen und übersichtlichen Zuteilung der Berechtigungen für die betreffenden Software-Pakete. Dafür muss es in der Lage sein, die vorhandene Domänen-Struktur und die darauf abgebildeten Gruppen nachzubilden. Ausnahmen davon sollten durch zusätzliche Berechtigungen auf ebenso einfache Weise abbildbar sein, s. d. die Paket-Verteilung individuell anpassbar wird.

Aufgrund der vorgenommenen Berechtigungen muss das System in der Lage sein, die entsprechenden Software-Pakete auf die Zielsysteme auszurollen. Das Ausrollen und dessen Erfolg muss überwacht werden und in einer Historie vom System gespeichert und dargestellt werden können. Außerdem sollten die Erfolge als System-übergreifender Report angezeigt werden können. Dies dient neben der Erfolgskontrolle auch dem Lizenz-Management, da es eine Überprüfung der Volumen-Lizenzen erlaubt.

Eine weitere Anforderung besteht neben der Berechtigung zur Installation auch in der Kontrolle der Häufigkeit der Installation. So erfordert z. B. das Ausrollen eines Software-Paketes im Erfolgsfall eine einmalige Installation. Die Installation von Richtlinien oder Einstellungsparametern im Sinne der Konfiguration erfordern allerdings mitunter mehrmalige Installationen, um ihre Beständigkeit zu sichern.

Des weiteren muss das Software-Management gewährleisten, das bestehende Abhängigkeiten zwischen Software-Paketen berücksichtigt werden. So können z. B. Pakete existieren, für die, als Voraussetzung zur Installation, andere Pakete installiert sein müssen. Andere Pakete wiederum schließen das Vorhandensein bestimmter Pakete aus.

3.5. Technische Anforderungen

Neben den fachlichen gibt es auch eine Reihe technischer Anforderungen an die Komponenten(s. Kap. 2.3) des Management-Systems, auf die im folgenden eingegangen werden soll.

3.5.1. Management-Controller

Der Management-Controller ist die Schnittstelle zwischen den mit dem Management beauftragten Mitarbeitern und dem Management-System. Dementsprechend sollte er aus Aspekten der Usability leicht und auf bekannte Art bedienbar sein.

Des weiteren ist es vom Vorteil, wenn die Ausführung des Management-Controller nicht auf einzelne Arbeitsplätze beschränkt ist. Der mit dem Management beauftragte Mitarbeiter soll also von beliebigen Arbeitsplätzen Zugriff auf den Management-Controller haben. Dies erfordert für die Verbindung zwischen Management-Controller und -Gateway eine Zugangskontrolle, s. d. nur die berechtigten Mitarbeiter Zugriff auf das Management-Gateway erhalten.

Aufgrund der Ausrüstung der Arbeitsstationen muss der Management-Controller unter dem Betriebssystem Microsoft Windows XP Professional ausführbar sein. Eine leichte Portierbarkeit zu anderen Plattformen wäre aber, im Fall einer Änderung der Beschaffenheit des Systems, von Vorteil.

3.5.2. Management-Agent

Der Management-Agent bildet die Schnittstelle zwischen der zu verwaltenden Ressource und dem Management-Gateway. Er befindet sich lokal auf der Ressource, d. h. auf den zu

verwaltenden Arbeitsstationen, und muss in der Lage sein, mit dem Management-Gateway zu kommunizieren.

Um die oben beschriebenen fachlichen Anforderungen, d. i. das Sammeln von Informationen für das Asset-Management und das Entgegennehmen von Installationsbefehlen und deren Ausführung für das Software-Managements, umsetzen zu können, muss der Management-Agent dauerhaft und mit erhöhten, d. h. administrativen, Rechten ausgeführt werden. Dies setzt besondere Anforderungen an die Sicherheit voraus. So darf der einfache Systembenutzer keinen vollwertigen Zugang zu dem Agent haben, d. h. er darf ihn z. B. nicht beenden oder seine Konfigurationsparameter verändern können. Es wäre aber vorteilhaft, wenn der Benutzer trotzdem in der Lage ist, bestimmte Funktionen(z. B. der Abruf eines Status oder eine erzwungene Aktualisierung) aufzurufen. Des weiteren muss die Kommunikation zum Management-Gateway gesichert sein, s. d. die gesammelten Informationen nicht ungeschützt über das Netzwerk verbreitet werden. Sowohl Management-Gateway, als auch Management-Agent, müssen ihre Authentizität gegenseitig nachweisen können, s. d. dem Gateway keine gefälschten Informationen geliefert werden und dem Agent keine unauthorisierten Installationsbefehle erteilt werden können.

Das Sammeln von Informationen für das Asset-Management erfordert eine Konfigurations- und Erweiterungsmöglichkeit des Agents, s. d. die zu sammelnden Informationen flexibel gestaltet werden können.

Die gesammelten Informationen und Ergebnisse der Instalationen müssen auch in dem Fall, dass der Management-Agent seine Verbindung zum Gateway verliert, persistent gespeichert werden können, um bei einer Wiederherstellung der Verbindung die Daten erneut übermitteln zu können.

3.5.3. Management-Datenbank

Die Anforderungen an die Management-Datenbank bestehen in ihrer Fähigkeit die Daten des Management-Systems aufzunehmen, persistent zu speichern und auf Anforderung an das Management-Gateway auszuliefern. Die Datenbank sollte Transaktionssicherheit aufweisen, s. d. beispielsweise nebenläufige Schreibvorgänge nicht zu Datenverlusten führen können(vgl. (Heuer und Saake, 2000, S. 497)).

Die Sicherheitsanforderungen an die Management-Datenbank erfordern Zugriffskontrolle auf die hinterlegten Daten, s. d. nur authorisierte Zugriffe erfolgen. Wenn sich die Management-Datenbank räumlich getrennt vom Management-Gateway befindet, eine Kommunikation zwischen ihnen also über das Netzwerk stattfindet, ist eine verschlüsselte Kommunikation und eine Authentizitäts-Prüfung anzuraten.

3.5.4. Management-Gateway

Das Management-Gateway dient, wie in Kap. 2.3 beschrieben, der Verarbeitung der erhobenen Daten und Steuerungsbefehle. In der Systemarchitektur ist es somit als Server anzusehen. Aufgrund der, in Kap. 3.1 beschriebenen, Systemstruktur ist daher eine, vom Betriebssystem unabhängige, Einsatzfähigkeit des Management-Gateways anzuraten. Es muss mindestens unter einem auf Linux basierenden Betriebssystem arbeiten.

Um die Portabilität der angebundenen Management-Agenten zu fördern, sollten die Schnittstellen zu den Agenten auf einer standardisierten und plattformunabhängigen Middleware beruhen. Die Schnittstellen zu der angebundenen Management-Datenbank und zur Integration anderer Management-Systeme sollten ebenfalls offen implementiert werden, s. d. der Zugriff auf die dahinterliegenden Komponenten flexibel und austauschbar gestaltet werden kann.

Die Anforderungen an die Sicherheit liegen so, wie es für die Interoperabilität zwischen den o. g. Komponenten des Management-Systems und dem Management-Gateway beschrieben wurden. Zusätzlich kommt noch die Implementation eines Schutzes gegen Ausfalls hinzu, da das Gateway als zentrale Komponente des Systems anzusehen ist und es dadurch als systemkritischer Bestandteil angesehen werden muss.

4. Architekturen und Lösungen

Es existieren zahlreiche Modelle für Management-Architekturen, die in verschiedenen Management-Systemen zur Anwendung kommen. In diesem Kapitel werden zunächst die für Management-Systeme entwickelten Architekturen und Technologien vorgestellt und ihre Anwendbarkeit auf das System beschrieben. Dann werden einige ausgewählte bestehende Management-Systeme vorgestellt und auf ihre Konformität mit den Anforderungen geprüft.

4.1. Management-Architekturen und -Modelle

Die grundsätzlichen Ziele der Architekturen bestehen in der Vereinheitlichung und Kooperationsfähigkeit aller Management-Werkzeuge, in der Automatisierung ihres Betriebes und in deren simplen und fehlertoleranten Bedienbarkeit. Die Probleme, die dabei auftreten, betreffen die Darstellung und Übertragung der erhobenen Management-Informationen(Terplan, 1995, S. 39f). Lösungen müssen also durch Informations-, Organisations-, Funktions- und Kommunikations-Modelle gebildet werden(Hegering et al., 1999, S. 100). Die Architekturen, die sich dabei durchgesetzt haben, beruhen grundsätzlich auf zwei verschiedenen Prinzipien:

- Ihr Einsatz ist Ressourcen-schonend und die Komponenten arbeiten mit einem geringen Befehlssatz und kommen mit wenigen Datentypen aus. Sie sind zwar unabhängig und universell einsetzbar, haben aber durch den reduzierten Funktionsumfang Nachteile in der Handhabung und Funktionstiefe. Architekturen, die auf diesem Prinzip beruhen, werden hauptsächlich für die Verwaltung von Netzwerk-Komponenten(z. B. Switches, Router, Bridges, etc.), Embedded-Devices und anderen Komponenten, die geringe Ressourcen zur Verfügung haben, genutzt.
- Ihr Einsatz ist nicht Ressourcen-abhängig, daher ist eine umfassende, objektorientierte Darstellung der verwalteten Komponenten möglich. Kommunikation kann über verschiedene Protokolle erfolgen. Erweiterungen können dynamisch eingebunden werden und es existieren verschiedene Sicherheitsprotokolle. Architekturen, die auf diesem Prinzip beruhen, werden für die Verwaltung komplexer Komponenten, wie z. B. für Arbeitsstationen und Server, oder Strukturen benötigt.

Die Verwaltung von heterogenen Umgebungen erfordert herstellerunabhängige Architekturen. Durch offen verfügbare Spezifikationen müssen verbindliche Standards geschaffen werden, die eine weite Verbreitung und Implementation finden und dadurch eine Komponenten- und Plattform-unabhängige, einheitliche Verwaltung ermöglichen. Die Standardisierung kann dabei auf zwei verschiedenen Arten erfolgen:

1. Standardisierung der Schnittstellen der Management-Architektur
2. Standardisierung der Eingangs genannten Teilmodelle, d. h. der Informations-, Abstraktions-, Funktions- und Kommunikations-Modelle

Die am weitesten verbreiteten Architekturen und Modelle werden von unabhängigen Organisationen oder Konsortien entwickelt und gepflegt. Diese sind z. B. *SNMP*(Simple Network Management Protocol), dass durch die Internet Engineering Taskforce (IETF) entwickelt wird, *DMI*(Desktop Management Interface), sowie *WBEM*(Web Based Enterprise Management) und dessen Informationsmodell *CIM*(Common Information Model), welche durch die Distributed Management Task Force(DMTF) entwickelt werden. Weitere Modelle sind das *CMIP/S*(Common Management Information Protocol/Service), welche durch ISO-OSI entwickelt werden, und *OMA*(Object Management Architecture), das auf CORBA aufsetzt und welches durch die Object Management Group(OMG) entwickelt wird.

Im folgenden soll auf einige dieser Architekturen und Modelle genauer eingegangen werden, so dass ihr Aufbau und ihre Funktionsweisen für die Entwicklung des Designs genutzt werden können. Zunächst müssen allerdings die o. g. Teilmodelle der Management-Architekturen betrachtet werden, s. d. die Struktur der Architekturen genauer definiert werden können.

4.1.1. Teilmodelle

Das Modell einer Management-Architektur setzt sich aus mehreren Teilmodellen zusammen. Für die Analyse der Struktur einer Management-Architektur ist es also nötig, die Teilmodelle zu untersuchen. Die Beschaffenheit und die Aufgaben der Modelle werden im folgenden beschrieben.

Informationsmodell

In einer heterogenen Systemlandschaft kann sich die Darstellung, d. h. Kodierung, von Informationen zwischen den einzelnen System-Komponenten unterscheiden(z. B. die Kodierung von Datums-Angaben). Es ist daher zwingend erforderlich die Datendarstellung des Management-Systems zwischen den Systemen-Komponenten zu normalisieren. Die Darstellung besteht dabei aus einer eindeutigen Identifikationsmöglichkeit des Datensatzes,

der Beschreibung des Datentypes, seiner annehmbaren Werte und ausführbaren Operationen, Informationen zu den Zugriffsmöglichkeiten auf den Datensatz und Informationen zu Beziehungen zu anderen Datensätzen. Die zusammengefassten Beschreibungen eines Datensatzes des Management-Systems nennt man ein Management-Objekt(MO). Die Menge aller Management-Objekte einer Komponente des Management-Systems nennt man Management-Informations-Basis(MIB). Das Informationsmodell schafft eine einheitliche Definition der Syntax zur Beschreibung von Management-Objekten. Außerdem wird die Strukturierung der Management-Objekte und deren Instantiierungs- und Abstraktions-Möglichkeiten festgelegt.

Organisationsmodell

Das Organisationsmodell dient der Abbildung der Management-Architektur auf die Struktur des Unternehmens. Es ermöglicht Gruppen-, Domänen- und Rollen-Zugehörigkeiten auf topologischer, funktionaler und organisatorischer Ebene. Das Organisationsmodell definiert dabei die hierarchische Strukturierung der einzelnen Komponenten des Management-Systems. Die Strukturierung kann auf gleichberechtigter Basis(Peer-to-Peer), auf einer Client/Server-Basis oder auf einer Mischform, bei der einige Komponenten gleichberechtigt und andere als Auftraggeber oder Auftragnehmer dienen, bestehen. Die Struktur muss nicht statisch sein, d. h. die Rollen der Komponenten können sich im Hinblick auf ihre Kooperationsform während ihres Bestehens ändern, um sich an neue Ereignisse und Aufgaben anzupassen.

Funktionsmodell

Das Funktionsmodell nimmt eine funktionale Gliederung der Management-Objekte nach den in Kap. 2.2 definierten Sichten vor. Dies geschieht durch Abbildung der Funktionen auf das Nachrichten-Protokoll der Management-Objekte und Festlegung geeigneter Schnittstellen.

Die Aufteilung der Management-Objekte gemäß ihrer Funktion ermöglicht die logische Trennung der Komponenten und ist somit die Grundlage für die Bildung eines Organisationsmodells.

Kommunikationsmodell

Eine Management-Architektur besteht aus verschiedenen Komponenten deren Lokalisation in dem System mit hoher Wahrscheinlichkeit dezentral angelegt ist. Es wird ein Modell benötigt, das die Kommunikation und die beteiligten Akteure beschreibt. Dies geschieht durch das Kommunikationsmodell.

Das Modell muss also ein Protokoll spezifizieren, das die Syntax und Semantik der übermittelten Nachrichten und die beteiligten Akteure beschreibt. Außerdem muss das Protokoll durch das Modell in die vorhandene Kommunikationsinfrastruktur eingegliedert werden, also z. B. eine Einordnung nach der OSI-Kommunikationsarchitektur, wie sie in ISO/IEC 7498-1 beschrieben ist, vornehmen. Des weiteren muss das Modell die Form der Kommunikation zwischen den Akteuren beschreiben. Mögliche Kommunikationsformen sind:

synchron Die Reihenfolge der Nachrichten zwischen den Akteuren ist festgelegt, d. h. die Kommunikation erfolgt nach einem festgelegten Frage/Antwort-Schema. Diese Art der Kommunikation wird für Aufträge und Statusabfragen an Management-Objekte angewendet.

asynchron Die Reihenfolge der gesendeten und empfangenen Nachrichten zwischen den Akteuren ist nicht relevant und muss deswegen nicht festgelegt werden. Diese Kommunikationsform wird in einer Management-Architektur z. B. für das Übermitteln von Ereignissen(Events), also z. B. Fehler- oder Erfolgsmeldungen, genutzt.

gerichtet Bei der gerichteten Kommunikation ist der Empfänger der Nachrichten festgelegt. Angewendet auf das Kommunikationsmodell einer Management-Architektur bedeutet dies, dass der Empfänger ein eindeutig definiertes Management-Objekt oder eine andere Architektur-Komponente ist.

ungerichtet Bei der ungerichteten Kommunikation ist der Empfänger nicht festgelegt. Diese Kommunikationsform wird im Zusammenhang mit der *asynchronen* Kommunikation in den Kommunikationsmodellen genutzt, wenn Management-Objekte Ereignisse melden müssen ohne den genauen Empfänger zu kennen. Als Transport-Protokoll bietet sich hier das, in der OSI-Kommunikationsarchitektur auf der Transportschicht spezifizierte, User Datagram Protocol(UDP, RFC768) an, da es asynchrone, ungerichtete Kommunikation ermöglicht.

4.1.2. Internet-Management Architektur

Die Internet-Management Architektur ist eine offene Architektur, deren Entwicklung durch das Internet Architecture Board(IAB) und die Internet Engineering Taskforce(IETF) geleitet wird. Sie wird häufig auch nach ihrem Kommunikationsprotokoll *SNMP*(Simple Network Management Protocol) benannt und liegt derzeit in der dritten Version(SNMPv3, RFC3410 bis RFC3418) vor. Das Ziel bei der Entwicklung war es, eine Architektur zu schaffen, die sich zur Verwaltung aller, über die Netzwerk-Architektur des Internets verbundenen, Komponenten eignet. Sich also in die OSI-Kommunikationsarchitektur eingliedern lässt. Bei der Entwicklung musste also besonders auf die leichtgewichtige, d. h. Ressourcen-schonende, und vielseitige Einsetzbarkeit geachtet werden. Der Einsatzbereich erstreckt sich von der Verwaltung

von Netzwerk- und Telekommunikations-Komponenten über die Verwaltung von Peripherie-Komponenten, wie z. B. Drucker, bis hin zur Verwaltung von Arbeitsstationen und Servern. Da die Internet-Management-Architektur eine der am häufigsten eingesetzten Architekturen ist, soll sie in diesem Abschnitt etwas genauer betrachtet werden.

Die erste Version von SNMP(SNMPv1, RFC1155 bis RFC1157, RFC1213) hat ungenügende Spezifikationen im Hinblick auf Sicherheitsanforderungen. Authentifizierung und Verschlüsselung ist nicht oder ungenügend implementiert und Authorisierung erfolgt über Klartext-Kennwörter, den sog. Community-Strings. Zusätzlich ist die Integrität der Übertragung großer Datenmengen ungenügend gesichert. Dies machte die Weiterentwicklung zu *SNMPv2* nötig. SNMPv2 existiert in diversen Ausprägungen(SNMPv2p, SNMPv2u, SNMPv2c), die sich im Implementierungsumfang, hauptsächlich im Hinblick auf die Sicherheitsfunktionen(RFC1441 bis RFC1452, RFC1902 bis RFC1908) oder Funktions-Erweiterungen, wie dem RMON(Remote-Monitoring, RFC2819 und RFC4502), unterscheiden. Die dritte Version, SNMPv3, wurde nötig, um die verschiedenen Implementierungsversuche von SNMPv2 zu vereinheitlichen und dabei die Interoperabilität der verschiedenen Vorgängerversionen der Architektur zu bewahren. Sie besteht aus mehreren Subsystemen, die modular aufgebaut sind und u. a. die Übersetzung der verschiedenen Nachrichtenformate, Protokolle und Sicherheitssysteme der Vorgängerversionen vornehmen können. Außerdem wird ein neues Nachrichtenformat und erweiterte Spezifikationen für die Zugriffskontrolle eingeführt(RFC3411). Die am weitesten verbreitete Version ist z. Zt. SNMPv2, gefolgt von SNMPv1 bei Komponenten mit leistungsschwacher Hardware. Der Einsatz von SNMPv3 beschränkt sich fast ausschließlich auf den Einsatz in Management-Controller/Gateways, die aufgrund ihrer Beschaffenheit per se die gestiegenen Anforderungen an die Ressourcen erfüllen können.

Im folgenden wird die Implementation der Teilmodelle der Internet-Management Architektur kurz beschrieben. Dabei wird auch auf die Änderungen zwischen den einzelnen Versionen der Architektur eingegangen, s. d. ein grundlegender Eindruck über deren Entwicklung entsteht.

Organisations- und Funktionsmodell

Die Internet-Management Architektur kennt in ihrer ursprünglichen Fassung zwei Arten von Akteuren: Die *Agenten*, die den in Kap. 2.3 beschriebenen Management-Agenten entsprechen, und den *Managern*, die einer Mischform aus Management-Gateway und -Controller entsprechen. Manager und Agent bilden in SNMPv1 eine flache Hierarchie, in der der Agent auf Anfragen eines oder mehrerer Manager antwortet. Sie bilden also eine Client/Server-Struktur. Mit Einführung der Manager-Manager-Kommunikation, der Proxy- und Sub-Agenten und den erweiterten Zugriffskontroll-System in SNMPv2 und SNMPv3 wird die flache Hierarchie aufgebrochen(Abbildung 4.1) und ermöglicht über Zugriffskontroll-Richtlinien(Policies) die Bildung von Domänen-Strukturen.

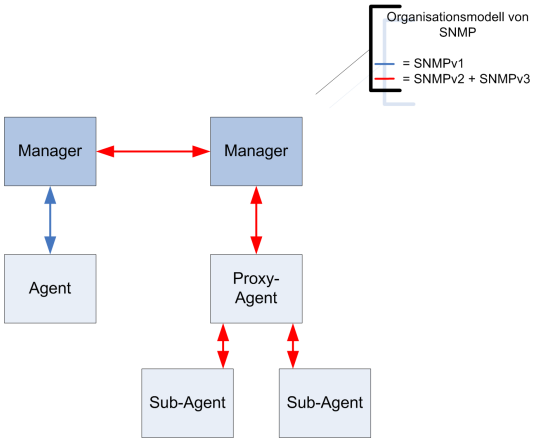


Abbildung 4.1.: Organisationsmodell der Internet-Management Architektur

Informationsmodell

Das Informationsmodell der Internet-Management Architektur wird *SMI*(Structure of Management Information) genannt. Nach einem ersten Entwurf(SNMPv1-SMI, RFC1155), wurde mit der Entwicklung von SNMPv2 auch die Entwicklung eines erweiterten Informationsmodells(SNMPv2-SMI, RFC1902) nötig, welches die entstandenen Änderungen an der Datenstruktur repräsentieren kann.

Das Datenmodell ist einfach gehalten. Agenten und Manager besitzen eine Datenbank(MIB (s. Kap. 4.1.1)) mit einer fest vorgegebenen Menge an repräsentierbaren Informationen. Dem Manager dient die Datenbank zur Abfrage und Anzeige der Informationen, dem Agenten zur Erfassung und, ab SNMPv2, auch zur Vorverarbeitung der erfassten Informationen.

Die Menge an erfassbaren Informationen werden in einer hierarchischen Baumstruktur geordnet, die durch eine ASN.1-kodierte(Abstract Syntax Notation One, ISO/IEC 8824-1) Benennung der in ihr spezifizierten Management-Objekte entsteht. Jeder Knoten verwaltet als Besitzer die Struktur seines Unterbaumes. Die Management-Objekte werden durch die Blätter des Baumes repräsentiert. Die Baumstruktur definiert einen Namensraum, der jedem Knoten und Management-Objekt im Baum einen eindeutigen Namen zuweist, der *OID*(Objekt Identificator). In der Internet-Management Architektur sind eine Reihe von Knoten definiert, die die wichtigsten funktionalen und organisatorischen Gliederung der Unterbäume vornehmen:

- *Standard-MIB*(`OID: .1.3.6.1.2.1`): Die Standard-MIB gibt eine Menge von Gruppen vor, die eine Grundmenge von Management-Objekten zur System-Verwaltung(z. B. der Netzwerk-Schnittstelle) beinhaltet. Die aktuelle Standard-MIB für SNMPv2-SMI, die *MIB-2*, ist in RFC1213 definiert. Die in 4.2 dargestellten Gruppen der Standard-MIB sind die

in SNMPv1-SMI definierten Gruppen. SNMPv2-SMI fügt weitere Gruppen hinzu und verlagert andere Gruppen. So wird z. B. die Gruppe *snmp*(`OID: .1.3.6.1.2.1.11`) nach *snmpMIB*(`OID: .1.3.6.1.6.3.1`) ausgelagert. Verschiedene Erweiterungen fügen neue Gruppen hinzu, z. B. *rmon*(`OID: .1.3.6.1.2.1.16`, RFC2819 und RFC4502).

- *Experimental-MIB*(`OID: .1.3.6.1.3`): Die Experimental-MIB beinhaltet MIB-Spezifikationen, die zur Standardisierung vorgesehen sind.
- *Private-MIB*(`OID: .1.3.6.1.4.1`): Die Private-MIB dient als Namensraum für Hersteller-abhängige MIB-Spezifikationen. Die Registrierung[1] und Verwaltung[2] von Namensräumen im Bereich der Private-MIB wird von der Internet Assinged Numbers Authority(IANA) vorgenommen.
- *security*, *snmpv2*, *mail* wurden durch SNMPv2-SMI hinzugefügt und repräsentieren die erweiterten Aspekte von SNMPv2.

Eine Übersicht über den Registrierungsbaum wird in Abbildung 4.2 gegeben.

Für das System muss eine Grundmenge an Management-Objekten festgelegt werden, die sog. *Internet-MIB*. Es existieren diverse RFCs, die, herstellerunabhängig, verschiedene MIBs zu den zu verwaltenden Komponenten spezifizieren(z. B. RFC1213, RFC2790, RFC3805, etc.). Außerdem besteht unter der Private-MIB die Möglichkeit, Hersteller-abhängige MIBs zu integrieren. Jeder Agent besitzt eine *Agenten-MIB*, die eine Untermenge der Internet-MIB darstellt und für den Agenten die Menge an erfassbaren und erfassten Informationen spezifiziert. Die Management-Objekte werden nicht objektorientiert behandelt, d. h. es existiert keine Vererbung zwischen den Objekten und die Management-Objekte haben kein eigenes Nachrichten-Protokoll. Sie existieren höchstens als einmalige Instanz der Internet-MIB in jeder Agenten-MIB. Dafür wird dem Agenten durch die Internet-MIB eine Objekt-Definition vorgegeben. Sie spezifiziert:

1. Name und OID des Vater-Knotens des Management-Objekts
2. Die erlaubte Zugriffsart(*not-accessible*, *read-only*, *read-write* und *accessible-for-notify*, *read-create* in SNMPv2-SMI)
3. eine Beschreibung
4. Syntax in Form eines ASN.1-Datentyps. Mögliche Datentypen sind für SNMPv1-SMI *Counter*, *Gauge*, *IpAddress*, *Network Address*, *Opaque*, *Time Ticks*, *INTEGER*, *NULL*, *OCTET STRING*, *OBJECT IDENTIFIER*. SNMPv2-SMI fügt die Datentypen *Unsigned32*, *Counter64* und *Nsap Address* hinzu. Mehrdimensionale Objekte, z. B. Routing-Tabellen, werden durch *SEQUENCE OF* oder *SEQUENCE* gebildet.

[1] `http://pen.iana.org/pen/PenApplication.page`
[2] `http://www.iana.org/assignments/enterprise-numbers`

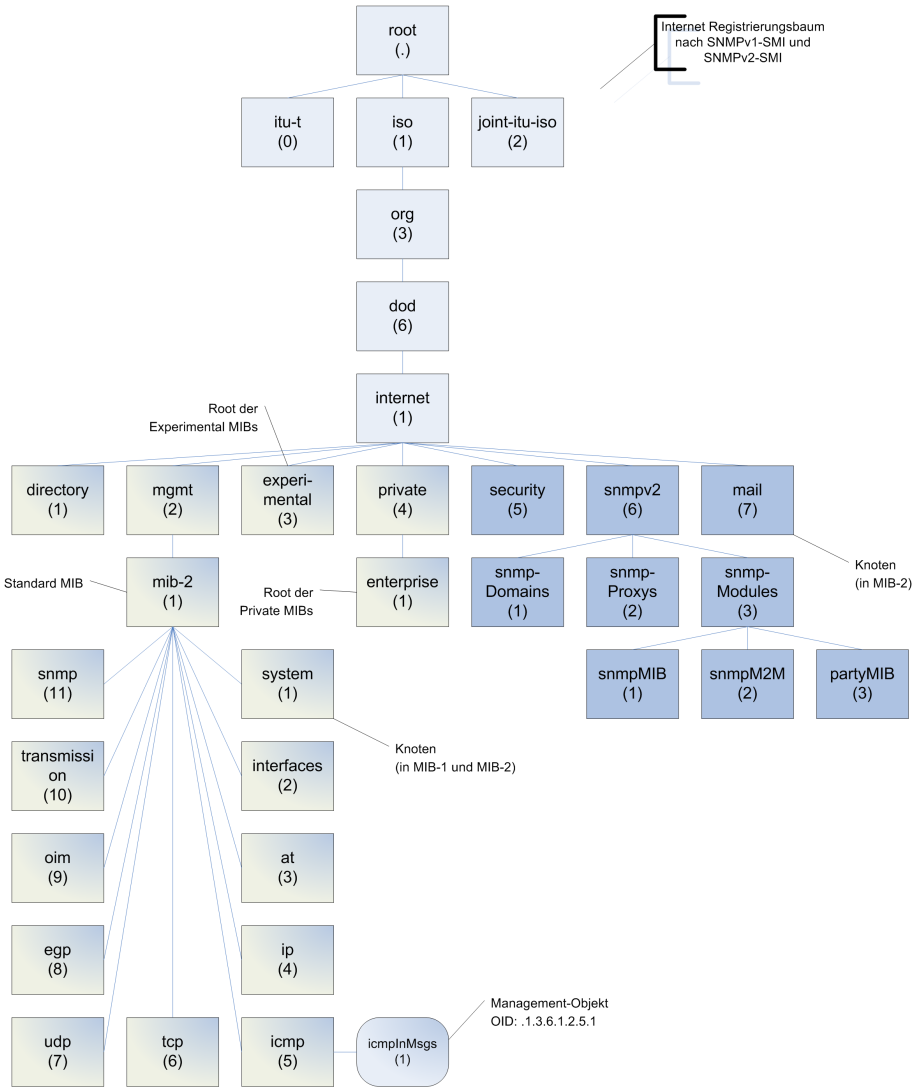


Abbildung 4.2.: Internet-Registrierungsbaum für SNMP-SMI

5. Informationen über den Gültigkeitsbereich des Management-Objektes (*current*, *deprecated*, *obsolete*)

Die Management-Datenbank (s. Kap. 2.2.3) liegt bei der Internet-Management Architektur als verteiltes System vor. Die Management-Informationen befinden sich in den Agenten, die Beschreibung der Informationen liegt global vor. Der Manager erhält die Management-Informationen auf Anfrage von dem entsprechenden Agenten.

Die starre Baumstruktur und die nicht-objektorientierte Implementierung der Management-Objekte in SMI ermöglichen zwar eine sehr zielgerichtete Auswahl an Attributen für die Verwaltung, bewirken aber ebenso eine verminderte Abstrahierungsmöglichkeit der Management-Objekte auf ihre zu verwaltende Ressource. So können sich z. B. die Management-Objekte, deren Bündelung eine Ressource beschreibt, in unterschiedlichen Unterbäumen des Registrierungsbaumes befinden.

Kommunikationsmodell

Die Internet-Management Architektur setzt in ihrer ursprünglichen Fassung das ihren Namen bestimmende Simple Network Management Protocol(SNMPv1, RFC1157) ein. Es basiert auf einem simplen Request-Response-Verfahren, bei dem der Manager eine Anfrage an den Agenten schickt, die der Agent mit einer geeigneten Nachricht beantwortet. Eine Ausnahme bildet ein sog. *Trap*, den ein Agent ohne vorherige Aufforderung an einen Manager schicken kann. Der Manager kann dann durch die Initialisierung einer geeigneten Abfrage auf den Trap reagieren. Traps dienen also dem Agenten dazu, den Manager auf bestimmte Ereignisse(z. B. Fehler, Änderungen der Netzwerk-Konfiguration, etc.) aufmerksam zu machen. Die Kommunikation erfolgt asynchron, d. h. der Manager kann nach Übermittlung seiner Anfrage weiterarbeiten und muss nicht auf die Antwort des Agents warten. Das Protokoll kennt folgende Nachrichten:

1. *get-request*: Anforderung einer Instanz eines Management-Objektes durch den Manager an den Agent. Der Manager übergibt zur Identifikation die OID des Objektes und ggf. einen Index, falls es sich um eine Tabelle handelt. Der Agent sendet den Wert der Instanz in einer Response-Nachricht an den Manager(s. u.).
2. *get-next-request*: Anforderung nach Durchwanderung der Agenten-MIB durch den Manager an den Agent. Der Manager übergibt zur Identifikation mindestens eine OID einer Objektinstanz oder eines Objekttyps. Der Agent antwortet dem Manager mit einer Response-Nachricht(s. u.), die den Wert und die OID der, in lexikalischer Ordnung, nächsten vorhandenen Objekt-Instanz enthält.

3. *get-bulk-request*: Diese Nachricht dient dem Abruf mehrerer Datensätze durch eine einzige Anforderung und ist eine Erweiterung von SNMPv2. Sie wurde benötigt um die Netzwerkbelastung beim Abruf großer Tabellen(z. B. Routing-Tabellen) zu verringern.
4. *set-request*: Anforderung des Managers an den Agent eine Instanz eines Management-Objektes mit einem Wert zu belegen. Der Manager übermittelt die OID der Instanz und den neuen Wert. Der Agent antwortet mit einer Response-Nachricht die den Erfolg der Operation beschreibt.
5. *response*: Eine Response-Nachricht stellt eine Antwort eines Agenten an den Manager dar.
6. *trap*: Benachrichtigung eines Agenten an einen Manager über ein eingetretenes Ereignis. Diese Nachricht wurde in SNMPv2 in *snmpv2-trap* umbenannt.
7. *inform-request*: Dies stellt eine Nachricht für die Manager-Manager-Kommunikation dar und wurde mit der Entwicklung von SNMPv2 eingeführt.

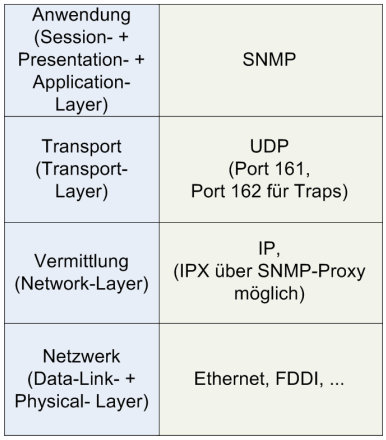


Abbildung 4.3.: SNMP im OSI-Referenzmodell

SNMP ist in das OSI-Referenzmodell eingegliedert(Abbildung 4.3). Zum Übertragen der Nachrichten wird UDP(RFC768) genutzt. Das Request-Response-Verfahren kommuniziert dabei auf Port 161 und die Traps auf Port 162. Nachteilig an der Verwendung von UDP ist, dass das Protokoll verbindungslos ist, d. h. weder der Manager noch der Agent können sich darauf verlassen, dass ihre Anfrage oder Antwort erhalten wurde.

Ein großer Nachteil von SNMPv1 waren die unzureichenden Sicherungsfunktionen. Das Protokoll verwendet den sog. *Trivial Authentification Algorithm*. Dabei werden bei jeder Anfrage des Managers ein Passwort, ein sog. *Community String*, im Klartext übermittelt. Der Agent gleicht den Community-String bei Empfang der Nachricht gegen einen bei ihm gespeicherten String

ab und führt die Anfrage bei Erfolg aus, die Antwort wird ungesichert an den Manager geschickt. Die Daten der SNMP-Pakete enthalten keinen Zeitstempel oder Sequenz-Nummern. Daraus ergeben sich mehrere Probleme. Zum einen ist der Community String nicht gegen das Abhören des Netzwerkverkehrs geschützt, d. h. ein abgehörtes SNMP-Paket ermöglicht einem Angreifer den Zugriff auf den Agent. Zum anderen erfolgt keine Authentifikations-Prüfung des Managers oder, bei der Antwort, des Agenten. Der Agent nimmt also alle Pakete an, die einen korrekten Community String aufweisen. Der Manager nimmt wiederum alle Antworten an, welches ihn der Gefahr von Wiedereinspielungs-Angriffen aussetzt, oder ihn bei Übermittlung von gefälschten Trap-Nachrichten zu ungewollten Reaktionen verleiten kann. Die diversen Ausprägungen von SNMPv2(s. o.) versuchten diese Schwachpunkte zu beheben, indem sie zum einen eine optionale Authentifizierung über den MD5-Algorithmus(Message Digest 5, RFC1321) und eine Verschlüsselung über DES-CBC(Data Encryption Standard - Cipher Block Chaining) einführten. Des weiteren wurde das Protokoll noch um einen Zeitstempel-Dienst erweitert(RFC1445 bis RFC1447). Die verschiedenen Ausprägungen von SNMPv2 und der häufig gleichzeitige Einsatz mehrerer Protokoll-Versionen machten es notwendig, das SNMPv3 in der Lage ist, zwischen den verschiedenen Protokoll-Versionen zu übersetzen. Dies wurde durch einen modularen Aufbau und entsprechende Übersetzungseinheiten gewährleistet. Die Zugangssicherung wird um den SHA-Algorithmus(Secure Hash Algorithm, FIPS180-2) erweitert.

4.1.3. Web-based Enterprise Management

Das Web-based Enterprise Management(WBEM) wurde ursprünglich durch einem Zusammenschluss von einigen großen Firmen der IT-Branche entworfen. Später wurden seine Kernkomponenten der Distributed Management Taskforce(DMTF) zur Weiterentwicklung übergeben, s. d. diese, durch einen offenen Standardisierungs-Prozess eine breitere Implementierung in den bestehenden Systemlandschaften finden. Da WBEM nicht so leichtgewichtig und Ressourcen-schonend konzipiert ist wie SNMP, hat sich seine Implementation bislang auf leistungsfähige Systeme, wie Desktop oder Server, beschränkt. Die Kernkomponenten des WBEM bestehen aus

- einem Kommunikationsmodell, das auf HTTP aufsetzt und den Abruf von Verwaltungs-Befehlen über XML-Dokumente(s. DSP0201) und die Anfragen durch eingebettete Befehle in HTTP-Headern beschreibt(s. DSP0200).
- einem Informationsmodell, dem Common Information Model(CIM), das eine einheitliche Struktur für die Instrumentierung der Management-Objekte vorgibt(s. DSP0004).

Im CIM werden die Strukturen der Managed-Objects, ähnlich wie in den MIBs von SNMP(s. Kap. 4.1.2), durch eine Beschreibungssprache in sog. MOF-Dateien(Managed Object Format) beschrieben. Eine prinzipielle Beschreibung des Aufbaus einer Objekt-Klasse

ist in Abbildung 4.4 dargestellt. Im Unterschied zu SNMP lassen sich die beschriebenen Managed-Objects aber nicht nur als einmalige Instanz instrumentieren, sondern werden Objekt-orientiert behandelt. Die Objekte lassen sich dabei in eine Vererbungs-Hierarchie eingliedern, aus verschiedenen anderen Objekten zusammensetzen und beliebig häufig als unabhängige Objekte instrumentieren. Die Möglichkeiten der Vererbung und Komposition gewährleisten eine flexible Wiederverwertbarkeit der definierten Strukturen und bieten somit einen deutlichen Vorteil gegenüber der starren Baumstruktur von SNMP-MIBs. Die definierten Objekte der MOF-Dateien lassen sich in drei Klassen gliedern:

1. Das *Core Model* stellt einen Grundstock an benötigten Klassen und deren Attributen und Operationen für die Definition eines Managed-Objekts zur Verfügung.
2. Das *Common Model* stellt erweiterte Klassen zur Verfügung, die zwar unabhängig zu der darunter liegenden Plattform sind, die Managed-Objects aber in Ihrem Informationsgehalt und Nachrichten-Protokoll verfeinern.
3. Das *Extension Model* stellt Plattform-abhängige Klassen zur Verfügung(z. B. das Win32-Schema für Microsoft Betriebssysteme)

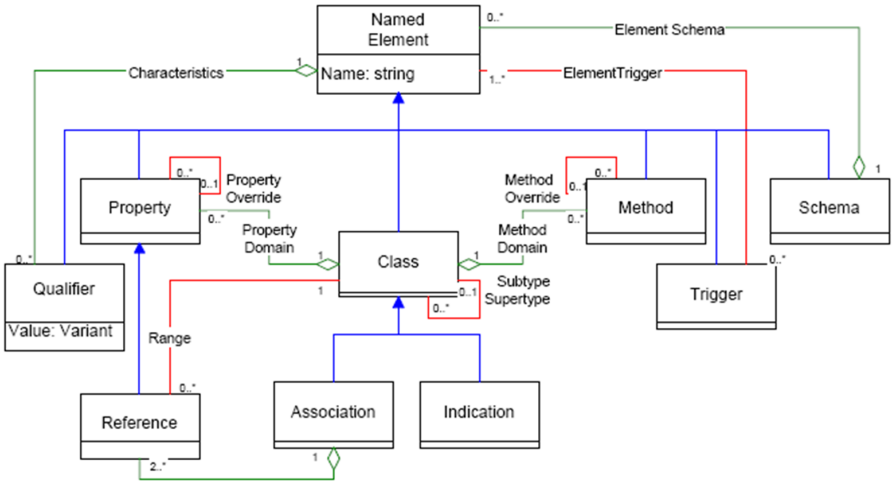


Abbildung 4.4.: Struktur des CIM Meta-Schemas(aus DSP0004)

Der WBEM-Agent auf einer verwalteten Ressource dient der Kommunikation mit den Managed-Objects über HTTP. Für deren Instrumentierung greift er entweder als sog. *Provider*, ähnlich einem Proxy, auf andere lokale Management-Systeme, z. B. SNMP oder DMI, zu oder er fragt als sog. *CIMOM*(CIM Object Manager) seine interne Management-Datenbank ab. Die Ergebnisse werden entsprechend an die anfragende Stelle weitergeleitet(Hegering et al., 1999, S. 232).

Microsoft hat WBEM unter dem Namen *Windows Management Instrumentation*(WMI) in seine Betriebssysteme integriert. Sie stellen dabei sowohl einen CIMOM als Dienst bereit, als auch einen Provider in Form eines Kernel-Treibers. Daneben bestehen auch einige Open-Source-Implementationen, wie z. B. *OpenWBEM*[3], *WBEM Source*[4] und *OpenPegasus*[5].

4.2. Marktübersicht

Nachdem mit SNMP und WBEM zwei Architekturen zur Verwaltung heterogener Computer-Netzwerke vorgestellt worden sind, sollen in diesem Abschnitt einige ausgewählte Produkte kurz vorgestellt werden, die die Architekturen zur Verwaltung einsetzen. Die Produkte sollen dabei u. a. auf ihre Anwendbarkeit, also auf die in Kap. 3 beschriebenen Anforderungen, untersucht werden.

4.2.1. Enteo v6 Client Suite

Bei der Enteo Suite handelt es sich um mehrere Desktop-Management Produkte, die zu einem Produkt zusammengefasst wurden. Der Schwerpunkt liegt dabei auf dem Software-Management(*enteo v6 NetInstall*), das mit umfangreichen Funktionen zur Paketierung, zum Testen und zur Richtlinien-gesteuerter Installation ausgestattet ist(Real-World Labs (2007)). Das Modul zum Inventar-Management(*enteo v6 Inventory*) nutzt Microsofts Implementierung des WBEM, Windows Management Instrumentation(WMI), zur Inventarisierung der Arbeitsstationen und SNMP zur Erkennung der Netzwerk-Umgebung. Des weiteren sind Module zur Betriebssystem-Deployment(*enteo v6 OS Deployment*), Patch-Management(*enteo v6 Patch Management*) und Fernsteuerung(*enteo v6 Remote*) vorhanden. Die Enteo Suite besteht aus einer Client/Server Architektur. Der Server nutzt eine Microsoft SQL-Datenbank zur Datenhaltung. Die Enteo Suite kann ausschließlich Windows Arbeitsstationen verwalten, das Server-System erfordert mindestens einen Windows-Server. Die Konfiguration des Systems erfolgt per Web-Browser oder über eine Management-Konsole, die auf Windows-basierten Systemen ausgeführt werden kann. Der Server benötigt keinen Verzeichnis-Dienst, sondern erkennt die verwalteten Arbeitsstationen anhand der Anfragen der dort gestarteten Clients. Diese können dann durch die Administrations-Konsole in Gruppen eingeteilt werden und somit die Domänen-Struktur abbilden. Ist ein Verzeichnis-Dienst in Form eines Active Directory vorhanden, kann dieser zur Unterstützung der Domänen-Bildung genutzt werden.

[3] `http://openwbem.sourceforge.net/`
[4] `http://http://wbemservices.sourceforge.net/`
[5] `http://http://www.openpegasus.org/`

Durch die Beschränkung auf Windows-basierte Systeme in den System-Anforderungen[6], eignet sich die Enteo Client Suite nicht für die in Kap. 3.1 beschriebene System-Umgebung. Das Vorhandensein eines Active Directories ist zwar nicht zwingend erforderlich und eine Installation auf einem einfachen, nicht Domänen-kontrollierenden, Windows-Server möglich, aber die doppelte, da nicht automatisierte, Pflege der Domänen-Struktur würde einen erheblichen Administrations-Aufwand erfordern.

4.2.2. Novell ZENworks v7 Suite

Novell ZENworks Suite ist eine Sammlung einzelner Management-Produkte, die über eine gleichförmige Speicherung der Management-Informationen und eine einheitliche Oberfläche, z. B. der *Novell ConsoleOne* oder über einen Web-Server, administriert werden können. Die Datenhaltung erfolgt in einem Datenbank-System von Sybase. ZENworks kann zur Integration in die bestehende Domänen-Struktur auf ein Active Directory oder ein Novell eDirectory, der LDAP-Implementation von Novell, als Verzeichnis-Dienst zurückgreifen. Die Desktop-Management-Komponente, ZENworks Desktop Management, implementiert u. a. Funktionen zum Software-Management, Sichern der Arbeitsstationen und Fernsteuerung. Neben dem Desktop-Management werden zusätzlich Module für das Server-Management, Handheld-Management, Patch-Management, Asset-Management, etc. angeboten. Die Server-Dienste sind in unterschiedliche Komponenten nach ihrer Funktion aufgespalten und können teils auf Linux-, teils Windows-Server Systemen installiert werden. Auf den verwalteten Arbeitsstationen muss ein Dienst installiert werden, der die Verbindung mit dem Server herstellt.

Novell ZENworks bietet durch seinen modularen Aufbau viele unterschiedliche, frei konfigurierbare Varianten zur Verwaltung von Computer-Netzwerken an. Da seine Server-Dienste teils auf Linux-, teils auf Windows-Servern installiert werden, und es Clients für fast alle Plattformen, u. a. auch Handhelds, gibt, eignet es sich zur Verwaltung von heterogenen Netzwerken. Um die in Kap. 3.4 spezifizierten Anforderungen zu erfüllen, müssen die Module für das Desktop- und Asset-Management erworben werden. Die Systemvoraussetzungen[7] für das Asset-Management Modul in der beschriebenen Größe der Systemumgebung, verlangen eine Installation auf einem auf Windows-basierten System, während das Desktop-Management Modul auf einem Linux-Server installiert wird. Durch die Bereitstellung mehrerer Systeme ist ein größerer Administrationsaufwand gegeben, der die Vorteile des Management-Systems negieren könnte. Ob dieser Aufwand, z. B. durch Nutzung virtueller Maschinen, gesenkt werden kann, bedarf eines Tests.

[6] http://www.enteo.com/de/v6-netinstall-sr_de,251940,47.html
[7] http://www.novell.com/documentation/zam75/pdfdoc/am75install/asset_management_installation_guide.pdf

4.2.3. Microsoft System Management Server 2003

Der Microsoft System Management Server(SMS) bietet eine Management-System für Server und Arbeitsstationen, die mit Microsofts Betriebssystemen arbeiten. Er bietet dabei Funktionen zur Software-Verteilung, Inventarisierung und System-Diagnose. Spezielle Funktionen für das Sicherheits- und Lizenz-Management, wie z. B. das *Application-Management*, das Regeln zur Benutzer-basierten Genehmigung von ausführbaren Anwendungen, sowie zur Fernsteuerung und zur Netzwerk-Überwachung implementiert, runden das System ab. Der SMS integriert sich nahtlos in ein bestehendes Active Directory und lässt sich zu einem Verband von mehreren kooperierenden Servern zusammenschließen(vgl. Kaczmarek (2004)). Die verwalteten Arbeitsstationen werden mit einem Dienst ausgestattet, der die Kommunikation mit dem SMS steuert und dessen Aufträge abarbeitet. Für die Steuerung der verwalteten Arbeitsstation nutzt der Dienst die Windows Management Instrumentation(WMI). Da dies eine Microsoft eigene Implementation von WBEM ist, ist der Einsatz auf Plattformen limitiert, die auf Microsoft Betriebssysteme basieren. Der SMS benötigt den Microsoft SQL-Server zur Speicherung der erhobenen Daten. Die Konfiguration des SMS wird über eine Management-Konsole vorgenommen.

Da die Systemvoraussetzungen[8] für den Betrieb des SMS und der verwalteten Arbeitsstationen eine reine Windows-Plattform und das Vorhandensein eines Active Directory als Verzeichnis-Dienst vorschreibt, ist SMS nach der in Kap. 3.1 beschriebenen System-Umgebung nicht einsetzbar. Es eignet sich dafür durch seine vollständige Unterstützung der Microsoft-spezifischen Verwaltungs-Schnittstellen hervorragend zur Verwaltung von homogenen, auf Microsoft Betriebssystemen basierenden, Computer-Netzwerken.

4.2.4. Andere

Neben den vorgestellten kommerziellen Produkten existieren noch diverse andere(z. B. Intel LANDesk Management Suite, ScriptLogic Desktop Authority). An dieser Stelle sollen noch zwei Open-Source-Projekte vorgestellt werden, die vielversprechend erscheinen.

Ein Projekt ist *opsi*[9]. opsi dient dem Software-Management, dem Inventar-Management und dem OS-Deployment, d. h. der automatischen Auslieferung von Betriebssystemen. Es besteht aus einem Client/Server-System. Der Server wird auf einem Debian Linux installiert und nimmt Anfragen von dem Client entgegen. Der Client ist auf den zu verwaltenden Arbeitsstationen installiert und nimmt Änderungen in der Software-Konfiguration bei der Benutzer-Anmeldung entgegen und führt diese aus.

[8] http://technet.microsoft.com/de-de/sms/bb676790(en-us).aspx
[9] http://www.opsi.org/

OCSInventory NG[10] ist ein System für Software-Distribution und Inventar-Management. Es besteht aus einer Client/Server-Architektur, die sowohl für Windows, als auch für Linux Systeme implementiert ist. Das System wurde ursprünglich für das Sammeln von Inventar-Installationen konzipiert und bietet hierfür eine gute und übersichtliche Administrationsoberfläche. Die Software-Verteilung ist eine neuere Funktion, die dabei aber schon die wichtigsten Funktionen integriert. So ist u. a. eine Paketierung und Priorisierung der zu installierenden Programme möglich. Das System besteht aus einem Client auf den verwalteten Arbeitsstationen, der über SOAP mit dem Server kommuniziert. Die Anzeige der Administrationsoberfläche erfolgt über einen Web-Server. Zur Datenhaltung benutzt der Server eine *MySQL*-Datenbank.

[10] `http://www.ocsinventory-ng.org/`

5. Design

Zur Erfüllung der beschriebenen Anforderungen ist ein Prototyp eines Management-Systems zu entwickeln. Der Prototyp soll aus zwei Komponenten bestehen:

1. Eine Client-Komponente, die die in Kap. 3.5.2 beschriebenen Funktionen des Management-Agents übernehmen soll.
2. Eine Server-Komponente, die die in Kap. 3.5.4 beschriebenen Funktionen des Management-Gateways übernehmen soll. Die Schnittstellen zu der Management-Datenbank und dem Management-Controller sollen dabei so erstellt werden, dass das Management-Gateway sich in bereits bestehende Lösungen integrieren lässt. Die zusätzliche Implementation von Komponenten zur Eingabe von Steuerungs-Befehlen und Ansicht der Verarbeitungs-Ergebnisse, sowie der Datenhaltung, also entfallen kann.

In diesem Kapitel soll das Design der zu entwickelnden Komponenten des Prototypen und ihrer Kooperation beschrieben werden.

5.1. Organisationsmodell

Die beiden Komponenten des Management-Systems, der Agent und das Gateway, sollen als Client/Server-System implementiert werden(Abbildung 5.1). Das Gateway nimmt, in der Rolle des Servers, Anfragen vom Agenten bzgl. Änderungen der Software-Konfiguration entgegen. Des weiteren wird ihm die Inventar-Konfiguration der verwalteten Arbeitsstation vom Agenten übermittelt. Die Hierarchie ist flach aufgebaut. Jedem Agenten ist über seine Konfiguration genau ein Gateway zugeteilt. Dadurch können mehrere Gateways, z. B. zur Lastverteilung oder zur Abbildung von Domänen-Strukturen, innerhalb eines Netzwerkes existieren. Die Konfiguration der Gateways wird über deren Benutzerschnittstelle gesteuert, auf die die Controller zugreifen können. Die Controller haben kein fest zugeordnetes Gateway und können so die Konfiguration aller Gateways steuern.

Das Organisationsmodell ähnelt damit dem in Kap. 4.1.2 vorgestellten Organisationsmodell der Internet-Management Architektur. Im Gegensatz zu SNMP wird die Nachrichtenübermittlung im Prototyp durch den Agenten initiiert. Der Agent kommuniziert in konfigurierbaren, periodischen Abständen mit dem Gateway. Zusätzlich besteht eine manuelle Möglichkeit,

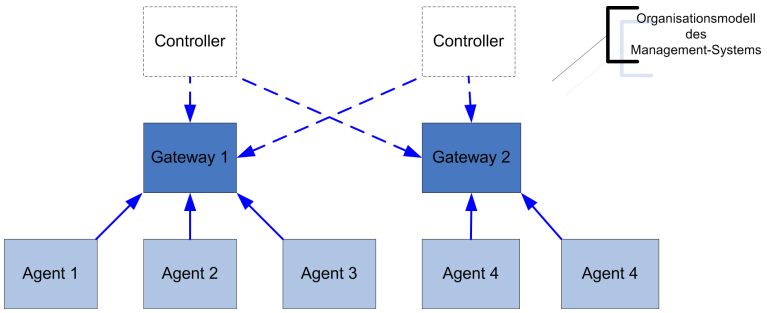


Abbildung 5.1.: Organsiationsmodell des Prototypen

die Kommunikation zu veranlassen. Hierdurch können Benutzer die Kommunikation forcieren und kurzfristig vorgenommene Konfigurationsänderungen im Sinne des Helpdesk-Managements(s. Kap. 2.2.2) erhalten.

Sollte das Gateway, z. B. bei Notebooks, die aus dem Netzwerk entfernt wurden, vom Agenten nicht erreicht werden können, findet keine Kommunikation statt. SNMP würde in diesem Fall durch das Suchen der Agenten, dem sog. *pollen*, das Netzwerk für die verbleibenden Arbeitsstationen unnötig belasten.

Eine Kommunikation zwischen verschiedenen Gateways, also eine Synchronisation der Management-Daten, ist nicht vorgesehen. Die Management-Daten, werden, wie in Kap. 6.1.4 beschrieben, in einer externen Datenbank gespeichert. Diese bietet i. a. von sich aus performante Synchronisationslösungen an(vgl. z. B. Radtke (2004)). Durch die Nutzung der selben Management-Datenbank oder durch deren Synchronisation mit der Management-Datenbank eines anderen Gateways lässt sich so ein gleichförmig konfiguriertes System erstellen.

5.2. Kommunikationsmodell

Das Kommunikationsmodell des Prototypen beschreibt die Kommunikation zwischen dem Gateway und dem Agenten. Wie in Kap. 5.1 beschrieben wurde, wird die Kommunikation mit dem Gateway durch den Agenten veranlasst. Entsprechend dem Funktionsmodell(s. Kap. 5.3) muss er dabei folgende Anfragen an das Gateway stellen können:

1. Übermittlung der ermittelten Inventar-Informationen
2. Anfrage nach ausstehenden Änderungen in der Software-Konfiguration
3. Übermittlung der Ergebnisse von vorgenommenen Änderungen in der Software-Konfiguration.

Das Gateway seinerseits, muss die empfangenen Nachrichten verarbeiten und beantworten indem es:

1. die Inventar-Informationen einem Management-Objekt zuordnet und in einer geeigneten Weise speichert, s. d. eine Anzeige und Auswertung der Informationen möglich ist.
2. die ausstehenden Änderungen in der Software-Konfiguration eines Management-Objektes ermittelt und dem anfragenden Agenten übermittelt. Um den Netzwerkverkehr nicht unnötig zu belasten, sollen dabei nur die Informationen übertragen werden, die der Agent für die entsprechende Änderung der Konfiguration benötigt.
3. die Ergebnisse von vorgenommenen Änderungen in der Software-Konfiguration müssen einem Management-Objekt zugeordnet und in einer geeigneten Weise gespeichert werden, s. d. das Gateway bei späteren Anfragen entscheiden kann, welche Änderungen ausstehen, welche erfolgreich vollzogen sind und welche fehlerhaft vollzogen wurden.

5.3. Funktionsmodell

Die Funktionskomponenten des Systems bestehen aus Agent und Manager. Beide Komponenten dienen dem Software- und Asset-Management. Der Agent hat dabei die Aufgabe, Inventar-Informationen zu sammeln, an das Gateway auszuliefern und die erhaltenen Installationsaufträge auszuführen. Die Aufgabe des Gateways besteht in dem Entgegennehmen der vom Agenten gelieferten Arbeitsergebnisse, deren Auswertung und Visualisierung. Des weiteren ist es für die Bestimmung neuer Arbeitsaufträge für den Agenten verantwortlich. Im folgenden werden die Aufgaben der Funktionskomponenten dargestellt.

5.3.1. Agent

Der Agent besteht aus mehreren funktionalen Komponenten. Eine schematische Gliederung ist in Abbildung 5.2 dargestellt. Er lässt sich zunächst einmal in drei voneinander unabhängige Bereiche unterteilen:

1. Inventar-Erfassung
2. Software-Verteilung
3. Benutzer-Interaktion

Jede Teilkomponente ist dabei für die Erfüllung ihres abgegrenzten Aufgabenkomplexes zuständig und unabhängig von den jeweils anderen Teilkomponenten. Sie haben Zugriff auf die benötigten Schnittstellen. Sie bieten Zugang zur lokalen Datenhaltung für Management-Informationen und Konfiguration des Agenten, zur Datenübertragung an das Gateway und zum Abruf der lokalen Informationsquellen für Inventar-Informationen.

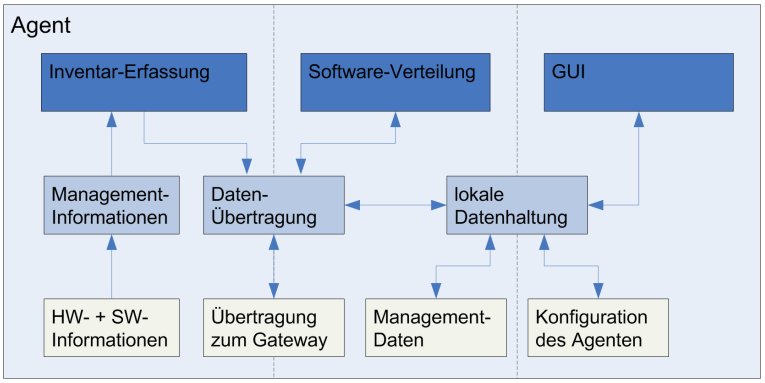


Abbildung 5.2.: Komponenten des Agenten

Die *Inventar-Erfassung* ist für das Sammeln und Bereitstellen der Inventar-Informationen zuständig. Für das Sammeln der Inventar-Informationen muss diese Teilkomponente Zugriff auf die Maschinen- und Betriebssystem-spezifischen Informationsquellen haben. Da der Zugriff auf diese Datenbanken abhängig vom eingesetzten Betriebssystem ist (s. Kap. 6.1.2), müssen die Daten über eine Schnittstelle abgerufen werden und in eine geeignete Darstellung(s. Kap. 5.4) gebracht werden. Die Informationsquellen bieten u. u. Inventar-Informationen an, die sich zur Laufzeit ändern können(z. B. über gestartete Prozesse), also dynamisch sind, und solche, die sich zur Laufzeit nicht ändern(z. B. Informationen über das verwertete Mainboard), also statisch sind. Aus Performance-Gründen bietet es sich daher an, die statischen Informationen von den dynamischen zu trennen. Während der Abruf der dynamischen Daten in regelmäßigen Intervallen erfolgen muss, kann sich der Abruf der statischen Daten auf einen einmaligen Abruf beschränken. Der Datenabruf darf dabei die Betriebsbereitschaft des Agenten nicht beeinträchtigen. Er sollte also, bei längeren Operationen, in einem eigenständigen Thread erfolgen. Um einen heterogenen Einsatz des Agenten zu ermöglichen, ist eine Implementation der Schnittstelle für die wichtigsten Betriebssysteme und Informationsquellen nötig. Die gesammelten Informationen müssen dem Agenten durch die Inventar-Erfassung zum Abruf bereitgestellt werden. Dieser muss daraufhin die Daten an das Gateway weiterleiten und deren korrekte Übertragung überprüfen.

Die *Software-Verteilung* ist für die Ausführung von Software-Installationen zuständig. Hierfür muss sie vom Agenten die empfangenen Software-Pakete zugeteilt bekommen. Sie muss

daraufhin einen Installations-Prozess starten, der die im Paket spezifizierten Konfigurationsänderungen an dem Software-Bestand vornehmen kann. Der Prozess muss eigenständig ablaufen und darf den Agenten auch bei längeren Installationen nicht blockieren. Er muss dabei in der Lage sein, die systemspezifischen Eigenschaften des Software-Paketes zu erkennen und sie den entsprechenden Systemprogrammen für die Installation zuzuführen. Er muss die Ausführung des Prozesses überwachen und dem Agenten Rückmeldung über den Status und die Korrektheit der Ausführung der Installation geben. Die Rückmeldung muss von dem Agenten an das Gateway weitergeleitet werden, s. d. dieses den Software-Stand des Agenten überwachen kann und dem mit dem Management beauftragten Mitarbeiter Rückmeldungen über eventuelle Installations-Fehler geben kann. Die Installations-Ergebnisse dürfen auch bei einem Verlust der Verbindung zum Gateway nicht verloren gehen und müssen in diesem Fall in einer lokalen Management-Datenbank zwischengespeichert werden. Bei Wiederherstellung der Gateway-Verbindung ist so eine erneute Übermittlung möglich.

Die *Benutzer-Interaktion* dient der Anzeige von Status-Meldungen(Abbildung A.1, A.4), dem Senden von Befehlen an den Agenten(Abbildung A.2) und der leichten Konfiguration des Agenten(Abbildung A.3). Um zwei möglichen Rollen, die administrative und die normale Rolle, der interagierenden Benutzer abzubilden, muss die Interaktions-Komponente des Agenten eine Möglichkeit anbieten, den normalen Benutzer von einem Administrator zu unterscheiden(Abbildung A.2). Der normale Benutzer soll nur die Möglichkeit haben, bestimmte Status-Informationen einzusehen und eine sofortige Synchronisation mit dem Gateway zu veranlassen. Dies unterstützt den Administrator bei der Leistung von telefonischer Hilfestellung und ermöglicht es dem Benutzer, Änderungen an seiner Software-Konfiguration zeitnah zu erhalten. Die administrative Benutzerschnittstelle bietet erweiterte Konfigurationsmöglichkeiten, z. B. die Arbeit des Agenten zu stoppen oder zu unterbrechen. Sie muss daher entsprechend vor dem Zugriff durch einfache Benutzer geschützt werden. Die vorgenommenen Änderungen an der Konfiguration müssen über eine Schnittstelle zur lokalen Datenhaltung in eine Konfigurations-Datenbank geschrieben werden.

5.3.2. Gateway

Um die Heterogenität des Gateways in der Systemlandschaft zu fördern, besteht es aus mehreren Teilkomponenten, die über Schnittstellen auf ihre funktionalen Komponenten zugreifen. Diese können können auf einfache Art ersetzt werden. Somit bleibt die Funktionalität des Systems auch bei einer Änderung der Systemlandschaft erhalten. Eine schematische Darstellung der Teilkomponenten und ihrer angestrebten Implementationen für den Prototypen sind in Abbildung 5.3 dargestellt.

Die *Kommunikations-Komponente* stellt Funktionen bereit, um die externen Komponenten

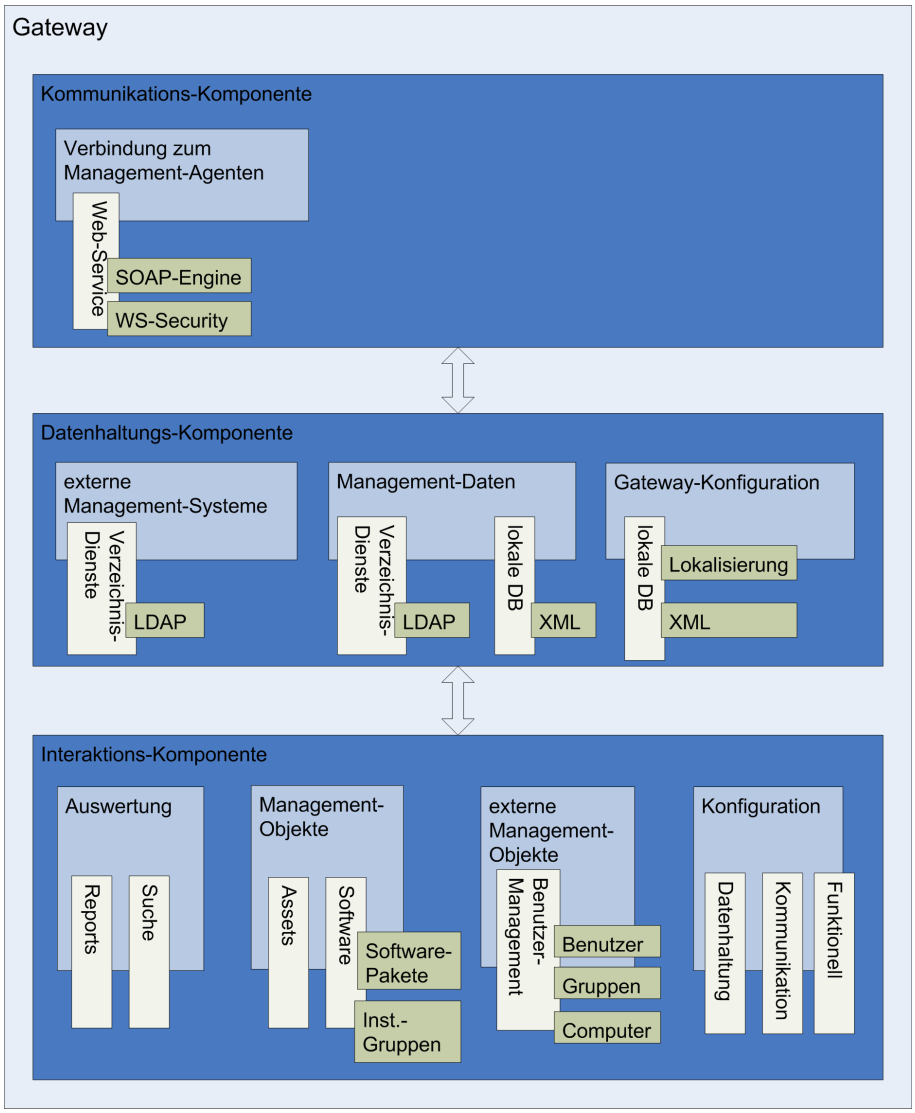


Abbildung 5.3.: Komponenten des Gateways

des Management-Systems, also den Agenten, in das System einzugliedern. Ihre Aufgaben liegen in

1. der Entgegennahme von Inventar-Informationen und deren Zuordnung zu einem Management-Objekt.
2. der Entgegennahme von Ergebnissen der Software-Verteilung und deren Zuordnung zu einem Management-Objekt.
3. dem Übermitteln von Aufträgen zur Software-Verteilung an ein Management-Objekt. Hierfür muss eine Auswertung erfolgen, deren Ergebnis die Berechtigung des Management-Objektes auf bestimmte Software-Pakete und deren Notwendigkeit zur Verteilung bestimmt. Um die Netzwerkbelastung zu minimieren, müssen die dabei übermittelten Informationen auf ein Mindestmaß reduziert werden(s. Kap. 5.4).

Die entgegen genommenen Informationen müssen in der Management-Datenbank(s. u.) gespeichert werden, s. d. sie für spätere Auswertungen bereitstehen. Sollten die Informationen nicht zugeordnet oder gespeichert werden können, ist eine entsprechende Reaktion erforderlich. Die Informationen müssen also, je nach Relevanz, verworfen oder dem Agenten die Notwendigkeit einer erneuten Übermittlung signalisiert werden. Für eine einfache Konfiguration der Kommunikations-Komponente ist es erforderlich, dass deren Konfigurations-Parameter in einer Konfigurations-Datenbank abgelegt sind.

Die *Datenhaltungs-Komponente* ist für die Bereitstellung der vom System verarbeiteten Daten konzipiert. Sie muss dabei die Daten aus verschiedenen Datenbanken beziehen und speichern können. Um das Management-System in einer heterogenen Umgebung einsetzen zu können, muss die Schnittstelle zu den Datenbanken offen und flexibel gestaltet sein, so dass ein individueller Angleich vorgenommen werden kann.

1. Zugriff auf externe Management-Systeme: Die Datenhaltung muss die Daten der Benutzer-Verwaltung von einer externen Datenquelle beziehen können. Diese Daten werden für die Zuordnung von Berechtigungen und der Abbildung der Organisationsstruktur benötigt(s. u.). Die Implementation soll einen Zugriff auf LDAP-Server bieten können, da sie von den meisten größeren Systemumgebungen als Verzeichnis-Anbieter für die Benutzer-Verwaltung genutzt werden(z. B. NDS, ADS, OpenLDAP). Die Unterschiede zwischen den einzelnen LDAP-Implementationen besteht in unterschiedlich implementierten Schemata, d. h. unterschiedlich strukturierten Daten. Die in diesem Prototyp implementierte Variante soll auf die Schemata, die in RFC4510, RFC4519 und RFC4524 spezifiziert sind, aufbauen und primär das OpenLDAP-System unterstützen. Da diese Schemata auch von ADS- und NDS-Systemen zu großen Teilen unterstützt werden((Larisch, 2003, S. 113)), ist eine Portierung ohne größeren Zeitaufwand möglich(Zörner (2005)). Die ausgelesenen Informationen sollen dabei Benutzer-, Computer- und Gruppen-Objekte umfassen, um die Organisationsstruktur auf Berechtigungen

abbilden und Inventar-Informationen zuordnen zu können. Bei der Implementation der Management-Datenbank des Prototypen soll auf die oben genannte LDAP-Schnittstelle zurückgegriffen werden. Durch seine Eigenschaften, Daten strukturiert abzulegen, persistent zu speichern und diese später performant und sicher(RFC4513) zugänglich zu machen, ist LDAP als Datenspeicher geeignet. Die Datensicherheit ist durch die Möglichkeit der Replikation der Daten gegeben(Radtke (2004), RFC4533). Die Benutzung der LDAP-Schnittstelle ermöglicht es, die externen System-Anforderungen für den Prototypen des Management-Systems klein zu halten. Die Implementation muss dahingehend erweitert werden, dass sie die Management-Objekte der Software- und Asset-Verwaltung zu lesen und zu speichern vermag(s. Kap. 5.4).

2. Zugriff auf Management-Daten: Das Gateway muss Zugriff auf die Management-Datenbank haben, um die erhobenen Management-Informationen zur Auswertung auszulesen und sie persistent speichern zu können.

3. Zugriff auf die Gateway-Konfiguration: Das Gateway muss für seine Komponenten verschiedene Konfigurations-Parameter verwalten, s. d. sich die Konfiguration des Gateways über die Interaktions-Komponente vornehmen lassen kann. Die Konfiguration muss dafür lokal von dem Gateway in einer strukturierten Form gespeichert und ausgelesen werden können.

Die *Interaktions-Komponente* stellt die Benutzer-Schnittstelle zur Interaktion mit dem, mit dem Management beauftragten, Mitarbeiter zur Verfügung. Ihre Aufgabe besteht in der Darstellung von

1. der *Konfiguration* des Gateways. Die Konfiguration des Gateways muss von einem Management-Controller angezeigt und geändert werden können(Abbildung A.15). Die Konfiguration umfasst Parameter der Kommunikations- und Datenhaltungs-Komponente. Weiterhin sollen Einstellungen der Programm-Logik vorgenommen werden können. Dabei soll der Benutzer durch entsprechende Hilfestellung angeleitet werden.

2. den erhobenen *Management-Daten*. Zur Analyse und Übersicht muss der Benutzer Zugriff auf die erhobenen Management-Daten haben. Die Ergebnisse der Inventarisierung und Software-Verteilung, sollen dem Benutzer dargestellt werden und ihn in die Lage versetzen, sich jederzeit einen einfachen Überblick über den Zustand des Systems zu verschaffen. Die Darstellung muss die Software-Pakete und ihre Installations-Berechtigungen, d. h. die Systeme, an die die Auslieferung der Software-Pakete erfolgen soll oder bereits ausgeliefert sind, anzeigen. Die Installations-Berechtigungen sollen dabei über eine Auswahl der von der Benutzer-Verwaltung gelieferten Gruppen erfolgen. Des weiteren muss der Benutzer in der Lage sein, neue Konfigurationen, d. h. neue Software-Pakete(Abbildung A.11) und deren Installations-Berechtigungen(Abbildung A.10) zu erstellen und bereits Vorhandene zu bearbeiten.

Die Anzeige der Asset-Informationen muss so erfolgen, dass der Benutzer die angeforderten Informationen schnell und übersichtlich dargestellt bekommt und er mit ihnen gleichzeitig die Ergebnisse der Software-Installationen einsehen kann, s. d. er sich einen Überblick über die Konfiguration jedes einzelnen Systems machen kann. Zusätzlich muss die Einpflege geeigneter Informationen zu Garantie-Zeiten, Herstellern oder Händlern und der Vergabe von internen Bezeichnungen für die Komponenten möglich sein(Abbildung A.8, A.9).

3. den externen Management-Daten, d. h. der Benutzer-Verwaltung. Die Daten der Benutzer-Verwaltung müssen für Analysen erfasst und angezeigt werden(Abbildung A.7). Zur besseren Integration in bestehende Verwaltungs-Systeme sollten die Einträge auch bearbeitet werden können und Berechtigungs-Gruppen(Abbildung A.6) erstellt werden können, s. d. man die Zugriffsrechte auf neue Software-Pakete flexibel konfigurieren und erstellen kann.

4. der *Auswertung* der gesammelten Management-Daten. Die Auswertung umfasst das Erstellen von Reports und Suchen nach bestimmten Informationen, s. d. der Benutzer zum Einen gezielt Informationen von dem System erfragen kann(Abbildung A.13, A.14), ohne dass er Kenntnis von dem Standort der Informationen hat und zum Anderen vom System automatisch auf bestimmte Rahmenbedingungen und System-Konfigurationen hingewiesen wird. Hierzu gehören z. B. fehlerhaft ausgeführte oder noch ausstehende Installations-Anforderungen(Abbildung A.12).

Die Ansichts-Komponenten werden modular aufgebaut. Dies erleichtert die Darstellung und ermöglicht eine leichte Veränderung und Austauschbarkeit der angezeigten Darstellungs- und Design-Elemente und damit auch die einfache Veränderungen der abgebildeten Arbeits-Prozesse.

Da die Konfiguration des Management-Systems und seiner verwalteten Komponenten ein sicherheitskritischer Bereich ist und vor unauthorisiertem Zugriff geschützt werden muss, werden die entsprechenden Anzeige-Komponenten in einem geschlossenen Benutzer-Bereich integriert.

5.4. Informationsmodell

In diesem Abschnitt wird das Design der für die Management-Objekte verwendeten Daten-Struktur beschrieben. Dafür ist die Betrachtung der Daten-Strukturen in den verschiedenen Komponenten des Management-Systems nötig.

1. Der Agent hält nur die für die Erfüllung seiner Aufgaben benötigten Informationen vor. Er erhebt zum Einen Inventar-Informationen, die er vollständig an das Gateway übermitteln

muss, zum Anderen erhält er vom Gateway Informationen zu Software-Paketen, die er ausführen soll. Er beantwortet die Aufträge mit entsprechenden Status-Meldungen.

2. Das Gateway arbeitet mit detaillierten Informationen zu den Management-Objekten, um ein hohes Abstraktionsniveau zu erreichen. Es muss also alle vom Agenten erhobenen Informationen verarbeiten und speichern können. Des weiteren muss es die Datenstrukturen der angebundenen externen Management-Systeme implementieren und sie zur Auswertung mit den eigenen Management-Objekten verknüpfen können. An den Agenten übertragene Aufträge müssen dort nicht in der gleichen Abstraktionstiefe vorliegen. So werden z. B. Attribute, die dem Gateway bei einer Auswahl der Software-Pakete für die Verteilung zu den Agenten unterstützen, für die Funktionalität des Agenten nicht benötigt. Es muss also eine Umwandlung vor der Übertragung erfolgen.

3. Die Daten-Strukturen des Gateways müssen in der Management-Datenbank gespeichert werden. Hierfür ist eine geeignete Abbildung der Management-Objekte auf die unterstützten Strukturen der Datenbank nötig.

Zur Erstellung des Informationsmodells, ist zunächst eine Identifizierung aller abzubildenden Objekte vorzunehmen. Danach ist eine Einteilung der abzubildenden Management-Objekte in Klassen möglich. Hierdurch können Aussagen über Gemeinsamkeiten und mögliche Verallgemeinerungen getroffen werden. Durch Objekt-orientierte Implementation lassen sich die dargestellten Informationen abstrahieren.

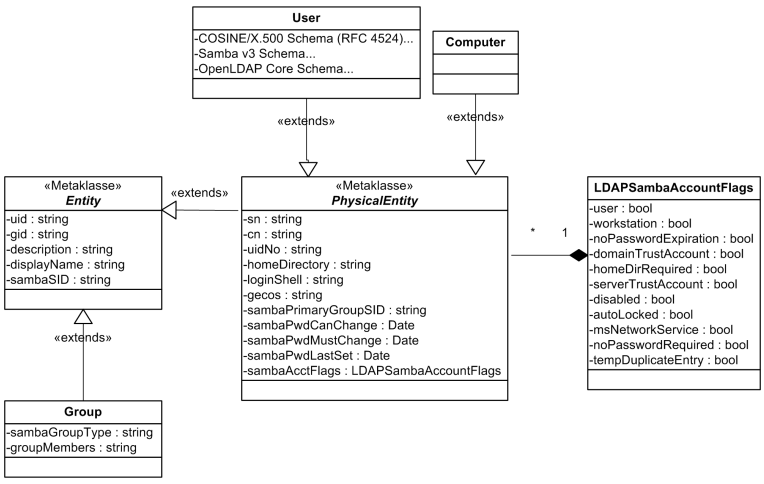


Abbildung 5.4.: Modell der Benutzer-Verwaltung

Für die Benutzer-Verwaltung werden die Klassierungen an die hierfür eingesetzten Bezeichnungen der Samba und ADS(Active Directory Services) angelehnt(s. S. 51). Es handelt sich hierbei um Benutzer(User), Computer und, als ein Container um beide zu bündeln, um Gruppen(Groups). Die Analyse ihrer Attribute in den o. g. Implementationen ergibt, das sie

eine Untermenge der in RFC2307, RFC4519 und RFC4524 spezifizierten Attribute bilden. Die Implementationen von Samba und dem Active Directory von Microsoft unterscheiden sich dabei in der Menge der implementierten Attribute. Das Design des Prototypen soll sich an der, in Abschnitt 3.1 vorgestellten, Systemstruktur orientieren. Daher werden die Daten anhand der Attribut-Menge der Samba-Implementierung, wie in Abbildung 5.4 dargestellt, strukturiert. Das *Entity*-Objekt bildet die grundlegende Menge von Attributen an. Das Attribut *uid* dient dabei als eindeutiger Identifizierungs-Schlüssel für ein Objekt. *Groups* können in ihrem *groupMembers* Attribut mehrere uid-Attribute speichern und so verschiedene Objekte bündeln. *User* und *Computer* werden über eine weitere Menge an gemeinsamer Attribute zu *PhysicalEntities* zusammengefasst, in der Informationen zu dem genutzten Benutzer-Konto gespeichert werden. Zusätzlich besitzen die User noch diverse Attribute, die hauptsächlich Informationen zu Kontakt-Daten(z. B. E-Mail-Adressen, Telefon-Nummern) enthalten.

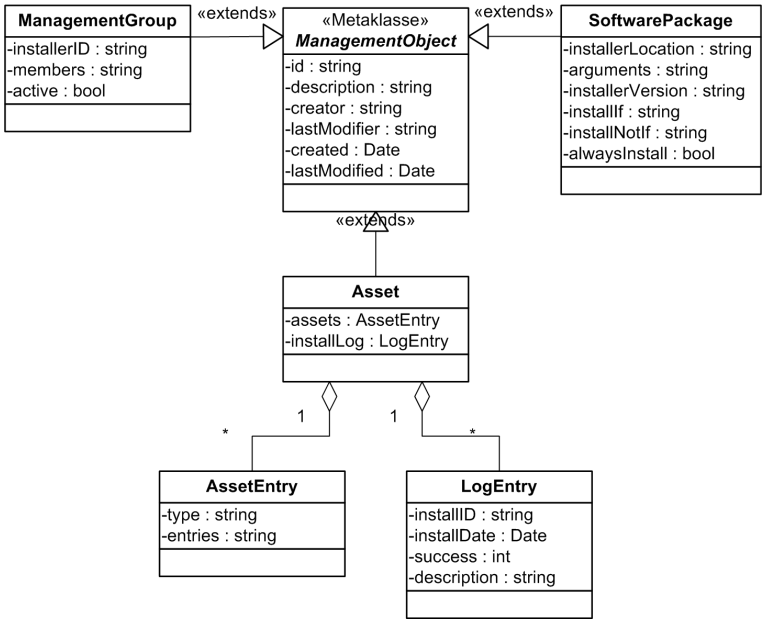


Abbildung 5.5.: Modell der Software- und Asset-Verwaltung

Die Software- und Asset-Verwaltung benötigt ein eigenes Daten-Modell(Abbildung 5.5), um ihre Funktionsmodelle abzubilden. Die Identifizierung der Klassen und ihrer Attribute ergibt folgende Komponenten:

- Das *ManagementObject* bildet die grundlegenden Attribute ab. Es enthält einen eindeutigen Identifizierungs-Schlüssel(*id*), eine Beschreibung(*description*) und Informationen zur Erstellung- und Änderungs-Verfolgung.
- Das *SoftwarePackage* speichert alle benötigten Informationen zu einem Installations-Paket. Dazu gehören der Speicherort(*installerLocation*), Argumente(*arguments*), die

bei der Installation angegeben werden sollen, Versions-Informationen(*installerVersion*) und Informationen, die Aufschluss über Abhängigkeiten(*installIf* für Pakete, die vor dem Installations-Paket erfolgreich installiert worden sein müssen , *installNotIf* für Pakete, die vor dem Installations-Paket nicht installiert sein dürfen) und über die Anzahl(*alwaysInstall*) der auszuführenden Installationen pro verwalteten Objekt geben.

- Die *ManagementGroup* verknüpft ein *SoftwarePackage*(*installerID*) mit Gruppen(*members*) aus der Benutzer-Verwaltung, um die Installations-Berechtigungen abzubilden. Des weiteren lässt sich die Freigabe der Installation steuern(*active*). So lassen sich Pakete erstellen, die noch nicht für den produktiven Gebrauch bestimmt sind, bzw. noch zurückgehalten werden sollen.
- Ein *AssetEntry* dient der Aufnahme einer Menge von Inventar-Informationen. Sie besteht aus einem Namen oder einer Typen-Beschreibung(*type*) und den Inventar-Informationen(*entries*). Auf diese Art lassen sich mehrere Attribute unter einer Typen-Klasse bündeln.
- Ein *LogEntry* dient der Beschreibung einer vollzogenen Installation. Es ist mit einem SoftwarePackage verknüpft(*installID*), speichert das Installations-Datum(*installDate*) und einen Status(*success*). Des weiteren lässt sich eine Beschreibung des Status(*description*) hinzufügen, welche genauere Informationen über den Ausgang der Installation liefert.
- Ein *Asset* nimmt alle vom Management-System gesammelten Informationen über einen Benutzer oder Computer auf. Es hält dafür Listen aller anfallender *LogEntry*s und *AssetEntry*s des Besitzers vor. Das *Asset* ist über seinen eindeutigen Schlüssel mit dem des Besitzers verknüpft.

Da die Kommunikation mit dem Agenten möglichst effizient und Ressourcen-schonend, d. h. mit geringer Netzwerkbelastung, verlaufen soll, werden die zur Installation benötigten Informationen, das sind die Attribute *id*, *installerLocation* und *arguments*, vor der Übermittlung aus dem *SoftwarePackage* extrahiert und übermittelt. Auf der Agenten-Seite werden sie nach der Ausführung mit zusätzlichen Werten(*success* und *description*) an das Gateway zurückgesendet. Dieses erstellt daraufhin aus den Werten einen *LogEntry*.

Um das Informationsmodell in der Management-Datenbank abzubilden, muss das Daten-Modell in ein Modell gewandelt werden, das eine Ablage der Informationen in der Datenbank ermöglicht. Die Objekte und deren Attribute müssen also in ein von der Datenbank darstellbares Format gewandelt werden.

6. Realisierung

Das im letzten Kapitel beschriebene Design dient als Grundlage zur Entwicklung eines Prototypen. In diesem Kapitel soll sein Aufbau und seine Funktionsweise beschrieben werden. Hierfür werden zunächst die in dem Prototypen eingesetzten Techniken und ihr Einsatzbereich aufgezeigt. Anschließend wird ein Überblick über den Aufbau des Prototypen gegeben und dessen Implementation beschrieben.

6.1. Eingesetzte Technologien

In diesem Abschnitt werden die eingesetzten Technologien und deren Aufgabe im Prototypen beschrieben. Die Beschreibung erfolgt zunächst anhand der Komponenten, d. h. des Agenten und des Gateways, des Prototypen. Nachfolgend wird auf die einzelnen externen Schnittstellen und ihre zur Implementation eingesetzten Technologien eingegangen.

6.1.1. Gateway

Der Prototyp soll die Inventar-Informationen und die Software-Konfiguration der Arbeitsstationen verwalten können und dem mit dem Management betrauten Mitarbeitern jederzeit einen Überblick über die entsprechenden Informationen liefern können. Dabei soll der Mitarbeiter unabhängig von seinem Arbeitsplatz Zugriff auf das System erlangen können. Hierfür soll eine Komponente des Systems als zentrales Management-Gateway implementiert werden, die Betriebssystem-unabhängig eingesetzt werden kann. Der Zugriff, d. h. die Eingabe von Steuerungs-Befehlen und der Abruf von Management-Informationen, auf das Management-Gateway soll einfach gehalten werden und keine gesonderten Anforderungen an das Betriebssystem und die installierten Anwendungen der zugreifenden Arbeitsstation stellen.

Für den beschriebenen Einsatz des Management-Systems wird eine Plattform benötigt, die Betriebssystem-unabhängig arbeitet oder für ein großes Spektrum an Betriebssystemen implementiert ist und deren Interoperabilität den entwickelten Prototyp unterstützt. Als Programmiersprache wurde daher *Java* gewählt.

Die vom Gateway erhobenen Daten des Management-Systems müssen verarbeitet, in der Management-Datenbank gespeichert und, in aufbereiteter Form, angezeigt werden. Zusätzlich müssen die beschriebenen technischen Anforderungen(s. Kap. 3.5) bzgl. der Sicherheit und der Anbindung an bestehende Management-Systeme erfüllt werden können.

Als Plattform für das Gateway wurde *Apache Tomcat* gewählt. Tomcat ist ein auf Java basierender Web-Server und Servlet-Container. Er bietet mit seiner Referenz-Implementation der Java Servlets und Java Server Pages(JSP) die Möglichkeit, HTML-Seiten dynamisch zu generieren und so die erhobenen Management-Daten aufzubereiten und anzuzeigen. Die Implementierung wird dabei durch den Einsatz von *MyFaces* unterstützt. MyFaces basiert auf JSF(Java Server Faces) und stellt ein Framework zur Verfügung, mit dem sich Ereignis-orientiert Anzeigeelemente erstellen lassen. Es benutzt durch Implementation eines Front-Controllers das Model2-MVC-Pattern(Model-View-Controller-Pattern) für Web-Anwendungen und erlaubt so eine logische Trennung der Anzeigeelemente von den angezeigten Daten und der dahinterstehenden Logik(Gamma et al. (2004)). Hierdurch lassen sich die Anzeige-Elemente modular gestalten, s. d. Änderungen an der Ansicht nicht zwangsläufig zu einer Änderung des Programm-Codes führen. Dies erleichtert die Darstellung und ermöglicht eine leichte Veränderung und Austauschbarkeit der angezeigten Darstellungs- und Design-Elemente und damit auch die einfache Veränderungen der abgebildeten Arbeits-Prozesse. Des weiteren erweitert MyFaces die Anzeige-Elemente durch zusätzliche Komponenten, die z. B. in dem *tomahawk*[1]-Paket implementiert sind.

Die Schnittstelle zur Verwaltung ist für den mit der Verwaltung beauftragten Mitarbeiter über einen Web-Browser über das HTTP-Protokoll Betriebssystem-unabhängig verfügbar. Die Benutzung des HTTP-Protokoll ermöglicht ohne Rekonfiguration des Netzwerkes die Anzeige der Informationen auch über Netzwerk-Grenzen hinaus, da HTTP-Pakete Paket-Filter, die an den Netzwerk-Grenzen installiert sind, meist ungehindert passieren können.

Aufgrund der Ziele des Systems, Applikationen auf die Arbeitsstationen auszurollen, muss dem Schutz des Gateways vor unberechtigten Zugriff besondere Beachtung geschenkt werden. Durch einen erfolgreichen Angriff auf das System ließen sich z. B. Trojaner auf allen Arbeitsstationen installieren. Für den Schutz der Kommunikation und der Betriebsbereitschaft stellt Tomcat mehrere Mechanismen zur Verfügung:

- Durch die Implementation von Funktionen zur Benutzeranmeldung wird die Authorisierung und Authentifizierung von Benutzern ermöglicht. Die Anbindung an bestehende Benutzer-Verwaltungssysteme ist, z. B. durch den Zugriff auf LDAP- oder AD-Verzeichnisse(Apache Software Foundation (2006a), Apache Software Foundation (2006b)), gegeben.

[1] `http://myfaces.apache.org/tomahawk/index.html`

- Durch die Implementation von SSL(Secure Socket Layer)ist eine Verschlüsselung der Kommunikation zwischen Controller, also dem Web-Browser, und dem Gateway möglich(Apache Software Foundation (2006c)).

Funktionen und Schnittstellen, die Tomcat nicht von sich aus bereitstellt, lassen sich im Rahmen der Möglichkeiten der Java Laufzeit-Umgebung realisieren.

6.1.2. Agent

Das Sammeln von Inventar-Informationen und die Konfiguration der Softwarepakete erfordert eine auf den Arbeitsstationen lokalisierte Komponente des Management-Systems, den Management-Agent. Obwohl z. Zt. alle Arbeitsstationen in der beschriebenen Systemhierarchie auf Windows-basierten Betriebssystemen arbeiten, soll der Agent Betriebssystemunabhängig sein oder seine Schnittstelle zum Management-Gateway so offen implementiert sein, dass sie auch für andere Plattformen implementiert werden kann. Dies soll eine spätere Ausdehnung der Verwaltung auf die Server oder einen Wechsel der Systemumgebung der Arbeitsstationen ermöglichen.

Daher soll der Agent ebenfalls in Java implementiert werden, das unter der aktuellen Version 6 der Laufzeit-Umgebung von Sun Microsystems zum Einsatz kommen soll. Version 6 bietet u. a. erweiterte Funktionalität im Umgang mit Netzwerk-Schnittstellen und Datei-Systemen(vgl. Sun Microsystems (2005)), die für die Funktionalitäten des Agents benötigt werden.

Zum Sammeln der Management-Informationen, ist es nötig, das der Agent Zugriff auf möglichst alle Management-Schnittstellen hat, die das darunterliegende Betriebssystem bietet. Die auf Windows XP Professional basierenden Arbeitsstationen besitzen, neben ihren internen Befehlen, als Management-Datenbanken u. a. eine CIM-Datenbank[2](s. Kap. 4.1.3) und die Registrierungs-Datenbank. Für auf Windows basierende Arbeitsstationen wird daher der Zugriff auf die Registrierungs-Datenbank durch *JNIRegistry*[3] realisiert. JNIRegistry greift über das Java Native Interface(*JNI*) auf die Registrierungs-Datenbank von Windows zu und ermöglicht das einfache Auslesen von Schlüsseln und deren Werte. JNI bietet in Java eine Schnittstelle zu nativen, also Plattform-spezifischen, Programm-Bibliotheken. Der Zugriff auf andere native Systemfunktionen, die Microsoft bereitstellt, wird durch *Jawin*[4] implementiert. Jawin stellt dabei eine Schnittstelle zum einfachen Zugriff auf Microsoft-COM(Component Object Model) und Funktionen der Dynamic Link Libraries(DLL) bereit, indem es die Funktionalität von JNI erweitert, s. d. der Zugriff auf native Systemfunktionen ohne die Erstellung

[2]Microsofts Management-Schnittstelle WMI ist nicht Kompatibel zu dem aktuellen CIM-Datenmodell der DMTF(s. VALUEADD\MSFT\MGMT\CIMV2R5\README.TXT auf der Windows XP Installations-CD)

[3]`http://www.trustice.com/java/jnireg/`

[4]`http://jawinproject.sourceforge.net/`

von nativen Programm-Code möglich ist(Halloway (2001)). Der Zugriff auf die CIM-Datenbank wird durch eine Schnittstelle zur WMI implementiert.

Obwohl Java prinzipiell eine Betriebssystem-unabhängige Implementierung erlaubt, bewirkt der Zugriff auf nicht standardisierte Schnittstellen, z. B. über einen Zugriff durch JNI, einen Verlust dieser Eigenschaft. Die Implementation der Datenerhebung kann also nicht Betriebssystem-unabhängig implementiert werden, sondern nur für die nötigen Systeme erstellt und entsprechend ihrer Nutzung delegiert werden.

6.1.3. Kommunikations-Schnittstelle

Für die Implementation der Kommunikations-Schnittstellen zwischen dem Manager und dem Agent in einem heterogenen System ist die Integration einer geeigneten Middleware notwendig. Die Spezifikationen der Middleware müssen offen verfügbar sein. Sie muss eine breite Integration in die bestehenden Systemlandschaften aufweisen und die technischen Anforderungen(s. Kap. 3.5) unterstützen.

Als Schnittstelle wurden Web-Services auf Basis von *Axis*[5](Apache eXtensible Interaction System) gewählt. Axis ist eine SOAP-Implementierung zum einfachen Erstellen von Web-Services und den dazugehörenden Clients. Es lässt sich nahtlos in den für das Gateway genutzten Servlet-Container integrieren. SOAP als Kommunikations-Protokoll gliedert sich auf der Anwendungs-Ebene in das OSI-Referenzmodell ein(Abbildung 6.1). Damit ist dem Prototypen eine breite Integrations-Basis in der Netzwerk-Architektur gegeben. Mit dem SOAP-Protokoll lassen sich Daten übermitteln und Nachrichten an entfernte Objekte senden. Die übermittelten Daten bzw. Nachrichten werden in einem standardisierten Format in XML-Dokumente gekapselt und mit entsprechenden Beschreibungen zur Dekodierung und Verwertung versehen. Die Semantik der Nachrichten wird dabei vom Applikations-Kontext bestimmt, d. h. sie kann abhängig vom gewählten Informationsmodell(s. Kap. 5.4) gewählt werden.

Das SOAP-Nachrichtenformat sieht einen festen, in Abbildung 6.2 dargestellten, schematischen Aufbau vor. Da SOAP-Nachrichten auf ihrem Weg vom Sender zum Empfänger mehrere Stationen passieren können, wird die Nachricht in einen *SOAP-Envelope* eingehüllt. Jede weitere Station, die die Nachricht passiert, kann der Nachricht weitere umhüllende Umschläge hinzufügen und der Nachricht somit Informationen hinzufügen ohne die ursprüngliche Nachricht zu verändern. Der Kopfteil der Nachricht, der *SOAP-Header*, enthält Informationen zur Verarbeitung der Nachricht, während sich die eigentlichen Nutzdaten und deren Beschreibung im *SOAP-Body* befinden. Hier können ebenfalls Fehlerinformationen in Form von *SOAP-Faults* übertragen werden.

[5] `http://ws.apache.org/axis/`

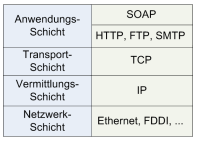


Abbildung 6.1.: SOAP im OSI-Referenzmodell

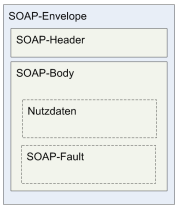


Abbildung 6.2.: SOAP-Nachrichtenformat

Durch die freie Semantik des Nachrichteninhaltes und die Möglichkeit Nachrichten an entfernte Objekte zu senden unterstützt SOAP damit die Erstellung eines Nachrichten-Protokolls für den Prototyp. Zudem lassen sich, durch eine geeignete Konfiguration der Schnittstelle, kompatible Clients auf Basis von .NET erstellen(Beyer et al., 2004, S. 491, 438f). Dies wird durch den Einsatz von *WSDL*, einer offenen, d. h. Plattform- und Protokoll-unabhängigen, Schnittstellen-Beschreibungs-Sprache, gefördert.

Axis bietet Mechanismen zur Vorverarbeitung der empfangenen und zu sendenden Nachrichten. Innerhalb von Axis werden die Nachrichten in Kettenform sequentiell durch die verschiedenen Verarbeitungsstationen geführt und entsprechend ihres Kontextes verarbeitet(Beyer et al. (2004)). Axis unterscheidet dabei, neben der grundlegenden Entscheidung, ob es sich um eine zu sendende oder zu empfangene Nachricht handelt, zwischen folgenden Ketten(Chains):

1. *Transport-Chain*: In der Transport-Kette werden Nachrichten abhängig von ihrem Transport-Protokoll verarbeitet.
2. *Global-Chain*: In der globalen Kette werden alle Nachrichten verarbeitet, die Axis passieren, unabhängig von dem benutzten Web-Service.
3. *Service-Chain*: Die Dienst-Kette bietet Nachrichtenverarbeitung auf Dienst-Ebene an, d. h. in ihr lassen sich Nachrichten abhängig von dem benutzten Web-Service verarbeiten.

Der Weg der Nachrichten durch die verschiedenen Verarbeitungsstationen ist in Abbildung 6.3 dargestellt.

Abbildung 6.3.: Nachrichtenverarbeitung der Axis Engine

Dies ermöglicht es, den Prototyp durch *Apache WSS4J*[6] zu schützen, indem es in die Dienst-Ketten für den zu erstellenden Service integriert wird. WSS4J erweitert die Funktionalität von Axis u. a. durch die Implementation von WS-Security(Web-Service Security, Nadalin et al. (2004b)) und X509- und Username-Token Profilen(Hallam-Baker et al. (2004), Nadalin et al. (2004a)). Somit ermöglicht es, SOAP-Nachrichten zu verschlüsseln, zu signieren, mit Zeitstempeln zu versehen und gegen Zugriffs-Listen abzugleichen und damit die Anforderungen im Bezug auf Integrität, Authentifizierung, Authorisierung und Geheimhaltung der Nachrichten sicherstellen.

6.1.4. Andere Schnittstellen

Um das Management-System in das unternehmensweite Netzwerk einzugliedern, benötigt es Schnittstellen zu den anderen eingesetzten Management-Systemen. Zum Abgleich der System-Benutzer wird ein Zugriff auf die entsprechenden Verzeichnis-Dienste, z. B. LDAP oder dessen Pendant in der Microsoft Systemumgebung, AD(Active Directory), benötigt. Die Java Laufzeit-Umgebung von Sun Microsystems bietet hierfür das *JNDI*(Java Naming and Directory Interface) an, mit dem sich diese Zugriffe realisieren lassen(Sun Microsystems (2007)).

Das Abfragen und Speichern von Management-Daten erfordert den Zugriff auf eine performante Datenbank. Denkbar sind hier z. B. Lösungen zur Anbindung an relationale Datenbanken, die in Java über *JDBC*(Java Database Connectivity) realisiert werden können. Eine andere, hier genutzte, Möglichkeit besteht aus der Nutzung der o. g. Verzeichnis-Dienste. Sie bieten zwar i. d. R. keine Transaktionssicherheit für ihre Operationen, bieten dafür aber die Möglichkeit eines schnellen und unkomplizierten Zugriffs auf Objekte. Dies wird durch den Einsatz von auf Filtern basierenden Suchfunktionen erreicht(Zörner, 2005, S. 62f). Eine weitere Möglichkeit bieten hier sog. *SearchControls*, die allerdings nicht standardisiert und daher von der gewählten Implementation des LDAP-Servers abhängig sind(Zörner (2005)).

Im nächsten Abschnitt wird die Implementation des Prototypen beschrieben. Hierfür wird eine Unterteilung in dessen einzelne Komponenten, dem Agenten und dem Gateway, vorgenommen. Für diese wird zunächst jeweils ein Überblick über den Aufbau in der Implementation

[6] `http://ws.apache.org/wss4j/`

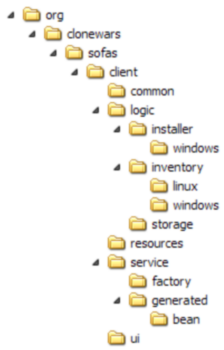
gegeben. Daraufhin wird eine konkrete Beschreibung derer funktionaler Bestandteile gegeben.

6.2. Implementation des Agenten

Der Agent ist auf den zu verwaltenden Arbeitsstationen lokalisiert und erfüllt im Management-System, wie in Kap. 5.3.1 beschrieben, unterschiedliche Aufgaben. Zum Einen dient er dem Sammeln von Inventar-Informationen, zum Anderen zur Installation von Software-Paketen. Die dabei gewonnenen Daten werden an das Gateway gesendet, während Arbeitsaufträge empfangen werden. Der Prototyp des Agenten ist in Java implementiert und nutzt zur Erfüllung seiner Aufgaben verschiedene Technologien(s. Kap. 6.1.2).

6.2.1. Überblick

Um die Implementation des Agenten zu beschreiben, soll zunächst ein Überblick über deren Aufbau gegeben werden.



Wie in Abbildung 5.2 visualisiert wurde, besteht der Agent aus verschiedenen Komponenten, die in unterschiedlichen Paketen angeordnet sind. Die nebenstehende Abbildung zeigt die Paket-Struktur des Agenten. Der Name des Basis-Pakets lautet `org.clonewars.sofas.client`. In ihm befindet sich die Anwendungs-Klasse des Agenten. Sie heißt `SofasClient`. Die einzelnen Komponenten verteilen sich auf die darunter liegenden Pakete:

`common`: Hier sind verschiedene Hilfsklassen abgelegt, die z. B. der Normalisierung von Werten und dem Zugriff auf das Datei-System dienen.

`logic`: In diesem Paket befinden sich die Implementationen der funktionalen Komponenten. Es handelt sich dabei um:

- `installer`: das Sub-System zur Installation von Software-Paketen.
- `inventory`: das Sub-System zur Erfassung von Inventar-Informationen.
- `storage`: die Komponente zur lokalen Datenhaltung.

`resourcen`: Hier befinden sich Ressourcen(z. B. Bilder, Konfigurations-Dateien, etc.), die der Agent zur Ausführung benötigt.

`service`: In diesem Paket liegen die Klassen, die für die Kommunikation mit dem Gateway erforderlich sind.

`ui`: Hier befinden sich Klassen zur Bereitstellung der Benutzer-Schnittstelle.

6.2.2. Zusammenhalt

In diesem Abschnitt wird die Kohärenz der Klassen des Agenten beschrieben. Ein schematischer Zusammenhalt der für die Umsetzung des in Kap. 5.3.1 beschriebenen Designs relevanten Klassen ist in Abbildung 6.4 dargestellt.

Die zentrale Implementations-Komponente ist dabei die Klasse `SofasClient`. Sie bindet zunächst eine Klasse zur lokalen Datenhaltung(`PersistanceHandler`), die die Konfiguration des Agenten bereitstellt. Die Konfiguration wird dabei über eine Schnittstelle(`IStorageProvider`) zur besseren Unterstützung der Modularität zugänglich gemacht. Nach dem Laden der Konfiguration werden die Subsysteme zur Software-Verteilung(`Installer`), zur Inventar-Erfassung(`Inventory`), zur Daten-Übertragung(`WSClient`) und das Benutzer-Interface(`SofasClientUI`) initialisiert.

Da das Installations-Subsystem abhängig von dem eingesetzten Betriebssystem ist, wählt der `Installer` hiernach seinen Provider aus. Der Provider muss die Schnittstelle `IInstallerProvider` implementieren und stellt die Funktionen zum Installieren eines Software-Paketes zur Verfügung. Da die Installation von Software-Paketen auf den mit Windows betriebenen Arbeitsstationen ausgeführt werden muss, ist hierfür ein `WindowsInstallerProvider` implementiert.

Das Subsystem für die Inventar-Erfassung ist abhängig vom Betriebssystem und den darauf verfügbaren Schnittstellen zu Inventar-Informationen. Daher wählt auch das `Inventory` seinen Provider abhängig vom ermittelten Betriebssystem aus. Er muss dafür die Schnittstelle `IInventoryProvider` implementieren. Es wurden Provider für Linux(`LinuxInventoryProvider`) und Windows(`WindowsInventoryProvider`) implementiert. Der Windows-Provider nutzt verschiedene Funktions-Klassen und hat durch sie Zugriff auf die Registrier-datenbank(`WindowsRegAccess`), System-Funktionen(`WindowsDLLAccess`) und die WMI(`WindowsWMIAccess`). Der Linux-Provider ist nach der in Kap. 3.1 aufgezeigten Systemlandschaft nicht zwingend benötigt und liefert nur wenige Informationen, die durch die Java-Umgebung bereitgestellt werden.

Die Daten-Übertragung zum Gateway wird vom `WSClient` vorgenommen. Dafür wird zur Konfiguration von Axis und WSS4J ein `WSSecurityEngineConfigurator` erstellt. Dieser wird zur Erzeugung eines Stubs(`SOFASServiceSoapBindingStub`) genutzt. Der Stub stellt den Client-seitigen Endpunkt der *Axis Engine* dar(Abbildung 6.3). Der Endpunkt wird dabei über die Schnittstelle `SOFASServerImpl` angebunden.

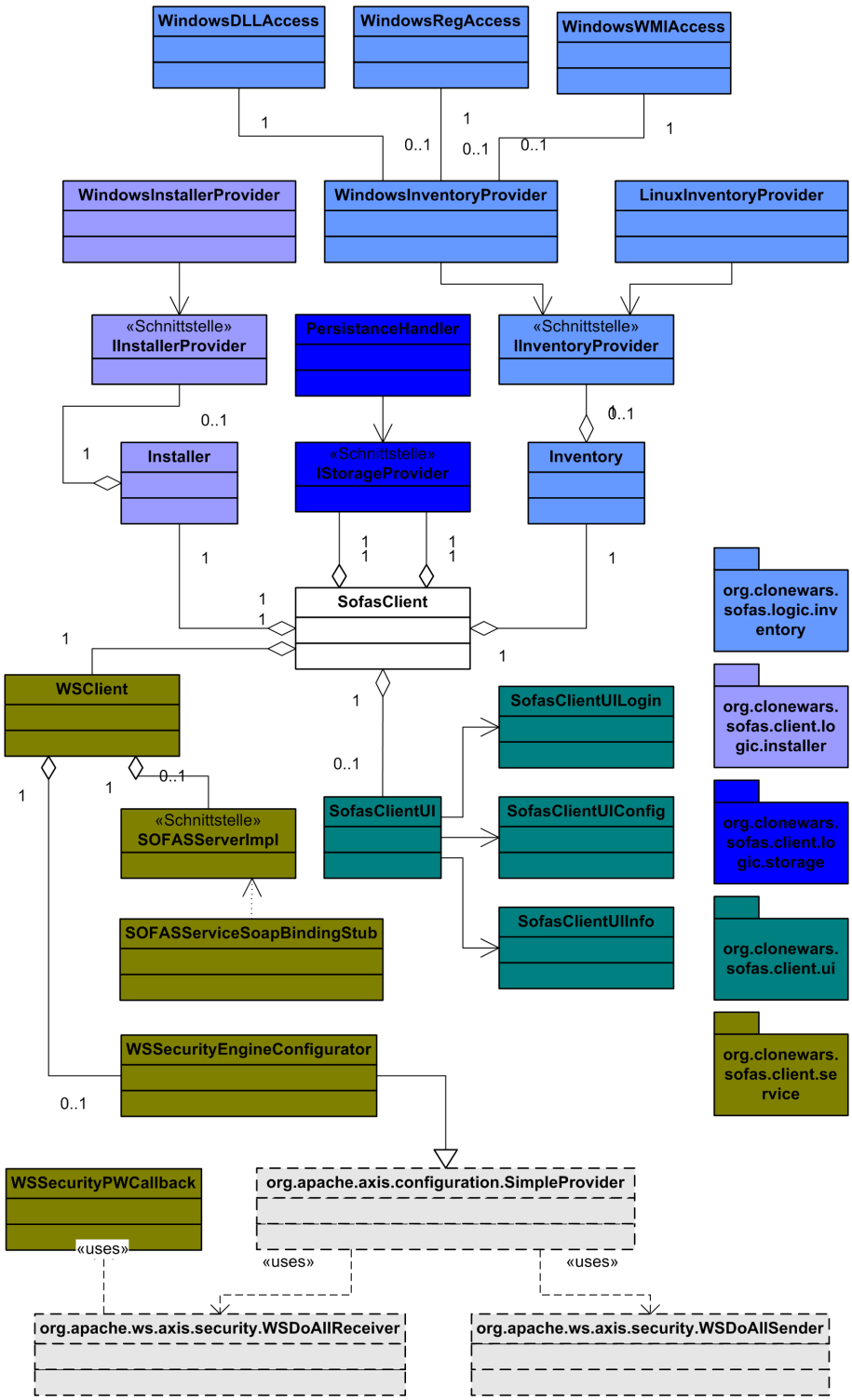


Abbildung 6.4.: Klassen-Diagramm des Agenten

Die Benutzer-Schnittstelle erstellt, sofern eine graphische Benutzeroberfläche vom Betriebssystem zur Verfügung gestellt wird, ein Symbol und ein Bedienungsmenü im System-Tray(Abbildung A.2). Je nach gewählter Funktion können durch das Menü die Fenster zur Konfiguration(`SofasClientUIConfig`, Abbildung A.3), zur Information(`SofasClientUIInfo`, Abbildung A.4) und zum Anmelden für den administrativen Modus(`SofasClientUILogin`, Abbildung A.2) erstellt werden.

6.2.3. Details

Nachdem im letzten Abschnitt der Zusammenhang der einzelnen Klassen beschrieben wurde, sollen in diesem Abschnitt genauer auf die Funktionen der einzelnen Klassen des Agenten eingegangen werden.

PersistanceHandler Der `PersistanceHandler` bietet Zugriff auf das lokale Dateisystem, um die Konfiguration des Agenten persistent zu speichern und bei Bedarf auszulesen. Die Konfigurations-Parameter werden dafür mit einem `XMLEncoder` strukturiert in eine XML-Datei serialisiert. Das Deserialisieren erfolgt entsprechend mit Hilfe eines `XMLDecoder`(s. a. Kap. D).

Die Konfiguration besteht zum Einen aus den Einstellungs-Parametern und zum Anderen aus dem Zustand, d. h. bei einem vorherigen Lauf gespeicherte Arbeitsergebnisse(z. B. Ergebnisse von Installationen, die noch nicht an das Gateway übertragen worden sind) und einigen statistischen Daten(z. B. den Erfolgs-Status der letzten Aktionen).

Installer Der `Installer` ist die Basis des Subsystems zur Software-Verteilung. Damit auch bei lang andauernden Installationen der Agent nicht blockiert wird, arbeitet der `Installer` als eigenständiger Thread. Aufträge zur Installation werden in einer Queue abgelegt, aus der sie nacheinander an den zuständigen `IInstallerProvider` abgegeben werden. Die Ergebnisse der Installationen, d. h. ihr Status und dessen Beschreibung(s. Kap. 5.4), werden ebenfalls in einer Queue zur Abholung für den `SofasClient` abgelegt.

WindowsInstallerProvider Der `WindowsInstallerProvider` implementiert die Schnittstelle `IInstallerProvider` und stellt Funktionen zur Installation von Software-Paketen für Windows-basierte Betriebssysteme bereit. Dafür prüft er die erhaltenen Installationsaufträge auf Konsistenz, d. h. es wird geprüft, ob das Installations-Programm an dem angegeben Standort, der `installerLocation`(s. Kap. 5.4), existiert und es sich dabei um ein für diesen Provider ausführbares Programm handelt. Bei positiven Befund wird das Programm an den entsprechenden Prozess-Starter übergeben. Der Prozess wird überwacht und das Ergebnis an den übergeordneten `Installer` zurückgeliefert.

Der implementierte `WindowsInstallerProvider` kann folgende Dateitypen bearbeiten:

- Registrierungs-Dateien(*REG*)
- Microsoft Installer(*MSI*, *MSP*)
- Skripte(*BAT*, *CMD*)
- sonstige ausführbare Dateien(*EXE*, etc.)

Inventory Das `Inventory` ist die Basisklasse des Subsystems zur Inventar-Erfassung. Sie arbeitet als eigenständiger Thread, um den Agenten nicht durch Verzögerungen während der Inventarisierung zu blockieren. Bei seinem Start sammelt es einmalig die statischen Inventar-Informationen, also diejenigen, die sich zur Laufzeit des Agenten nicht ändern. Danach werden in periodischen Abständen die dynamischen Inventar-Informationen, also diejenigen, die sich zur Laufzeit ändern, gesammelt. Die gesammelten Informationen werden für den Abruf durch den `SofasClient` bereitgestellt.

Das `Inventory` greift auf eine Klasse, die die Schnittstelle `IInventoryProvider` implementiert, zu, die es abhängig vom eingesetzten Betriebssystem auswählt. Sie stellt Funktionen zum System-spezifischen Zugriff auf Inventar-Funktionen bereit. Funktionen, die durch die Java Laufzeit-Umgebung System-übergreifend zur Verfügung gestellt werden, z. B. Informationen zu Datei-Systemen und Netzwerk-Eigenschaften, werden direkt in der Klasse `Inventory` bereitgestellt.

WindowsInventoryProvider Der `WindowsInventoryProvider` implementiert die Schnittstelle `IInventoryProvider` und stellt Funktionen zum Sammeln von Inventar-Informationen für auf Windows basierende Betriebssysteme zur Verfügung.

Die Abfrage der Inventar-Informationen werden dabei abhängig von der Informations-Quelle an Hilfs-Klassen(`WindowsDLLAccess`, `WindowsRegAccess`, `WindowsWMIAccess`) delegiert.

WindowsRegAccess Die Klasse `WindowsRegAccess` stellt Funktionen zum Zugriff auf die Registrier-Datenbank von Windows zur Verfügung. Hierfür benutzt sie die *JNIRegistry*-Bibliothek[7].

Sie bietet Möglichkeiten zur Auflistung der

- installierten Dienste und ihrer Start-Art
- installierten Software und ihre Version

[7] `http://www.trustice.com/java/jnireg/`

WindowsDLLAccess Die Klasse `WindowsDLLAccess` stellt Funktionen zum Zugriff auf System-Bibliotheken von Windows zur Verfügung. Sie benutzt hierfür die *jawin*-Bibliothek[8] und bietet Möglichkeiten zur Auflistung der

- ausgeführten Prozesse und ihrer PID(Process ID)
- verbundenen Netzwerklaufwerke
- Eventlog-Einträge

WindowsWMIAccess Die Klasse `WindowsWMIAccess` stellt Funktionen zum Zugriff auf das Windows Management Interface(WMI) zur Verfügung. Dies geschieht durch das Starten eines Prozesses, der die gewünschten Abfragen über ein WMI-Skript vornimmt und `WindowsWMIAccess` das Ergebnis übergibt. Die vorzunehmenden Abfragen sind in der Datei `WindowsWMIAccess.properties`, die sich im `resourcen`-Paket befindet(s. Kap. 6.2.1), frei konfigurierbar(s. Kap. D).

LinuxInventoryProvider Der `LinuxInventoryProvider` implementiert die Schnittstelle `IInventoryProvider` und stellt Funktionen zum Sammeln von Inventar-Informationen für auf Linux basierende Betriebssysteme zur Verfügung. Die Implementation umfasst dabei nur Inventar-Informationen, die die Java Laufzeit-Umgebung von sich aus zur Verfügung stellt(z. B. freier Speicher im Datei-System, Netzwerk-Konfiguration, Benutzername, etc.).

SofasClientUI Die `SofasClientUI` stellt die Schnittstelle zum Benutzer zur Verfügung. Dafür prüft sie, ob eine graphische Oberfläche vom Betriebssystem zur Verfügung gestellt wird. Sollte dies der Fall sein, so wird ein Symbol im System-Tray erstellt, das den Status des Agenten(s. Kap. A.1) anzeigt und über seine Menü-Struktur die Steuerung und Konfiguration des Agenten erlaubt(Abbildung A.2). Hierfür werden, abhängig von dem aufgerufenen Menüpunkt, die Klassen `SofasClientUILogin` zum Wechseln in den administrativen Modus, `SofasClientUIInfo` für Status-Informationen und `SofasClientUIConfig` zur Konfiguration der Agenten aufgerufen.

SofasClientUILogin Die Klasse `SofasClientUILogin` stellt ein Fenster zur Anmeldung bereit(s. Kap. A.2). Die erfolgreiche Anmeldung bewirkt einen Wechsel in den administrativen Modus, in dem der Benutzer sowohl die Konfiguration des Agenten ändern, als auch dessen Beendigung veranlassen kann. Der administrative Modus kann durch den Menü-Eintrag *Abmelden* im Menü des Tray-Symbols wieder verlassen werden.

Für die Anmeldung wird das eingegebene Passwort Base64-kodiert und dessen SHA1-Hashwert gegen einen in der Konfiguration gespeichertes Passwort abgeglichen.

[8] `http://jawinproject.sourceforge.net/`

SofasClientUIInfo Die Klasse `SofasClientUIInfo` stellt ein Fenster zur Anzeige von Status-Informationen bereit(s. Kap. A.4). Es werden Informationen über den Rechner-Namen und die IP-Adresse, der nächsten und letzten Kontaktaufnahme mit dem Gateway und deren Erfolgs-Status angezeigt. Die Informationen werden aus der Konfiguration des Agenten entnommen.

SofasClientUIConfig Die Klasse `SofasClientUIInfo` stellt ein Fenster zur Anzeige und Änderung von Konfigurations-Parametern bereit(s. Kap. A.3). Es kann die Netzwerkadresse des Gateways, das Aktualisierungs-Intervall des `SofasClient` und erweiterte Benachrichtigungs-Optionen bei Installationen für den Benutzer eingestellt werden. Des weiteren kann die Kontaktaufnahme mit dem Gateway deaktiviert werden. Dieses soll Administratoren bei manuellen operativen Tätigkeiten unterstützen.

Die Klasse `SofasClientUIInfo` hat lesenden und schreibenden Zugriff auf die Konfiguration des Agenten.

WSClient Die Klasse `WSClient` bietet dem `SofasClient` Zugriff auf die Implementation des Web-Service-Clients. Sie initialisiert Axis mit der angegebenen Konfiguration in Hinsicht auf die Zieladresse, also der Adresse des Gateways, und der konfigurierten Sicherheits-Einstellungen. Des weiteren stellt sie über eine Implementation der Service-Schnittstelle(`SOFASServerImpl`) einen Zugriff auf die Sende und Empfangs-Schnittstellen von Axis zur Verfügung.

Die übertragenden Daten werden von ihr entsprechend dem Informationsmodell, wie in Kap. 5.4 beschrieben, vor dem Versenden und nach dem Empfangen in die benötigten Daten-Strukturen gewandelt.

WSSecurityEngineConfigurator Die Klasse `WSSecurityEngineConfigurator` dient der Konfiguration der Axis Engine und von WSS4J, der Implementation von WS-Security(s. Kap. 6.1.3). Hierfür bildet die Klasse eine Erweiterung zu einem `org.apache.axis.configuration.SimpleProvider`, der, entsprechend der Konfiguration des Agenten, die benötigten Implementations-Klassen von WSS4J, `org.apache.ws.axis.security.WSDoAllReceiver` zum Empfang und `org.apache.ws.axis.security.WSDoAllSender` zum Senden, an die Dienst-Kette von Axis bindet(Abbildung 6.3).

WSSecurityPWCallback Für das Signieren, Verschlüsseln und Zuordnen von Benutzer-Token werden unterschiedliche Passwörter benötigt. Die Klasse `WSSecurityPWCallback` liefert das von WSS4J benötigte Passwort für die auszuführende Operation.

SOFASServiceSoapBindingStub Die Klasse `SOFASServiceSoapBindingStub` stellt die Implementation des Web-Service-Client dar. Dafür erweitert sie die Klasse

`org.apache.axis.client.Stub` und implementiert das Kommunikations-Interface des Gateways(`SOFASServerImpl`).

Die Klasse `SOFASServiceSoapBindingStub` wurde durch die Schnittstellen-Beschreibung(`SOFASService.wsdl`, s. Kap. 6.3.1) des Gateways mittels *WSDL2Java* automatisch erstellt.

SofasClient Die Klasse `SofasClient` ist die Basisklasse des Agenten und enthält seine ausführbare Methode. Als zentrale Klasse des Agenten verbindet sie dessen einzelnen Funktions-Komponenten.

Der `SofasClient` ist als Thread implementiert. So kann er die Ergebnisse der Inventarisierungen von `Inventory` periodisch Abfragen und an den `WSClient` zur Übertragung an das Gateway weiterreichen. Gleichzeitig kann er durch ihn Anfragen nach neuen Software-Paketen an das Gateway stellen lassen und diese an den `Installer` weiterreichen. Dessen Arbeitsergebnisse werden wiederum über den `WSClient` an das Gateway gesendet.

Für eine Rückkopplung mit dem Benutzer der Arbeitsstation werden Status-Änderungen vom `SofasClient` an die `SofasClientUI` gemeldet.

6.3. Implementation des Gateways

Das Gateway ist der zentrale Server im Management-System und dient zum Einen dem mit dem Management beauftragten Mitarbeitern als Ansichts- und Steuer-Schnittstelle und zum Anderen dem Agenten, indem das Gateway Management-Informationen entgegennimmt und Arbeitsaufträge erteilt.

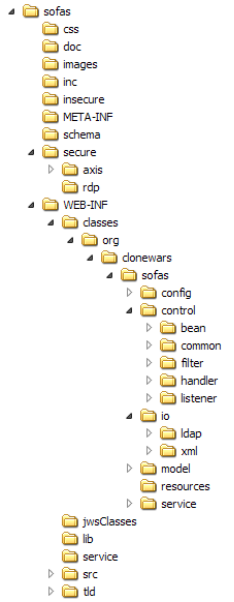
Er wird, wie in Kap. 6.1.1 beschrieben, als eine Anwendung im Kontext eines Tomcat-Servers ausgeführt und implementiert verschiedene Technologien, um seine einzelnen Module zu implementieren.

In diesem Abschnitt soll der Aufbau des Gateways beschrieben werden. Dazu wird zunächst ein Überblick über die Verzeichnis-Struktur der Komponenten gegeben. Anschließend werden der Zusammenhang der Komponenten und ihre Implementations-Details aufgeführt.

6.3.1. Überblick

In diesem Abschnitt soll ein Überblick über die Implementation des Gateways und seiner Komponenten gegeben werden. Dafür wird zunächst die Verzeichnis-Struktur und ihr Inhalt beschrieben.

Das Gateway besteht, wie in Abbildung 5.3 beschrieben, aus mehreren Komponenten. Um seine Implementation zu unterstützen werden unterschiedliche Technologien genutzt. Die verwendete Plattform ist dabei der Tomcat-Server. Die funktionalen Komponenten sind in Java implementiert(s. Kap. 6.1.1).



Tomcat erfordert eine zum Teil vorgeschriebene Verzeichnis-Struktur, um Anwendungen auf ihm auszuführen. Dazu gehört das Wurzelverzeichnis(*sofas*), welches den Anfang der Pfadangabe der URI der Anwendung wiederspiegelt und das Anwendungs-Verzeichnis(*WEB-INF*), das die Anwendung, benutzte Bibliotheken und Konfigurations-Dateien enthält. Alle anderen Verzeichnisse sind frei wählbar und hängen nur von den, in der Anwendung genutzten URIs und den gewählten Zugriffs-Berechtigungen ab. Die Zugriffsberechtigungen lassen sich über die Datei `WEB-INF/web.xml` konfigurieren(s. Kap. D). Diese Datei dient Tomcat zur Konfiguration der Anwendung, bzw. des Kontextes. In dem Verzeichnis `WEB-INF` befinden sich noch weitere Konfigurations-Dateien. Die Datei `faces-config.xml` dient der Konfiguration des Front-Controllers von MyFaces und `server-config.wsdd` der Konfiguration von Axis(s. Kap. 6.1.1).

Eine Übersicht der Verzeichnisse befindet sich in der nebenstehenden Abbildung. Ihr Inhalt wird im folgenden kurz beschrieben:

`sofas`: das Wurzelverzeichnis der Anwendung

`css`: Beschreibungs-Dateien für das Design der Anzeige-Elemente

`doc`: Dokumentationen des erzeugten Java-Codes

`images`: In der Anwendung genutzte Bild-Ressourcen

`inc`: Bausteine für die anzuzeigenden JSP-Seiten

`insecure`: ungesicherter Ablage-Bereich für JSP-Seiten

`META-INF`: Kontext-Informationen, die Tomcat optional einbindet

`schema`: die Schema-Definition für das Datenmodell der LDAP-Schnittstelle6.4

`secure`: gesicherter Bereich für JSP-Seiten, der durch die Tomcat-eigenen Sicherheits-Funktionen geschützt wird

- `axis`: Konfigurations-Zeiten von Axis(s. Kap. D)
- `rdp`: Fernsteuerungs-Applet(s. u.)

`WEB-INF`: Anwendungs-Bereich. Dieser Bereich ist nicht über den Browser erreichbar

- `classes`: Compilierte Anwendungs-Klassen(s. u.)
- `lib`: genutzte Programm-Bibliotheken
- `service`: Schnittstellen- und Implementations-Beschreibungs-Dateien des Web-Services
- `src`: Java Quell-Code
- `tld`: Beschreibungen von genutzten Anzeige-Elementen

Es folgt eine Beschreibung der Paketstruktur, d. h. des Aufbaus, der erstellten Java-Anwendung. Die Paket-Struktur ist in der obigen Grafik unterhalb des `WEB-INF/classes`-Verzeichnis aufgezeigt. Das Basis-Paket heißt `org.clonewars.sofas`. In ihm befinden sich die implementierten Klassen, die die Funktionalität der, in Abbildung 5.3 beschriebenen, Komponenten des Gateways repräsentieren.

- `config`: Klassen der Datenhaltungs-Komponente für die Gateway-Konfiguration
- `control`: Klassen für die Interaktions-Komponente und zum Zusammenspiel mit der darunter liegenden Plattform, d. h. Tomcat
  - `bean`: von MyFaces verwaltete Java-Beans
  - `common`: Hilfs-Klassen zur Daten-Normalisierung und Integritäts-Prüfung.
  - `filter, handler, listener`: Klassen zur Überprüfung von Anfragen und zur Reaktion auf bestimmte im System auftretende Ereignisse
- `io`: Klassen für die Datenhaltungs-Komponente zur Anbindung an externe Management-Systeme und zum Zugriff auf Management-Daten
  - `ldap`: Klassen für den Zugriff auf Verzeichnis-Dienste, die auf dem OpenLDAP-Server basieren
  - `xml`: Ein Test-Treiber der Management-Daten in XML-Dateien speichert, bzw. sie dort ausliest
- `model`: Implementation des, in Kap. 5.4 beschriebenen, Datenmodells
- `resources`: Konfigurations- und Lokalisierungs-Dateien(s. Kap. D)
- `service`: Klassen des erstellten Web-Services

6.3.2. Zusammenhalt und Aufbau

In diesem Abschnitt soll der Zusammenhalt und der Aufbau der einzelnen Funktions-Komponenten beschrieben werden. Aufgrund der Aufgaben des Gateways, lassen sich die Komponenten dabei in zwei voneinander unabhängige Bereiche einteilen:

1. Der Anzeige und Bearbeitung der Management-Informationen durch Visualisierung der Daten in entsprechenden Anzeige-Elementen durch die Interaktions-Komponente, im folgenden als *Benutzer-Interaktion* bezeichnet.
2. Der Sammlung, Zuordnung und Auslieferung der Daten durch Kommunikation mit dem Agenten durch die Kommunikations-Komponente, im folgenden als *Agenten-Interaktion* bezeichnet.

Die beiden Bereiche arbeiten dabei mit der gleichen Informations-Basis, der Datenhaltungs-Komponente, nutzen dabei aber unterschiedliche Schnittstellen als Werkzeug zur Auslieferung, bzw. Erhalt ihrer Arbeits-Aufträge. Im folgenden sollen zunächst der Aufbau der beiden o. g. Bereiche beschrieben werden. Nachfolgend wird der Aufbau der Datenhaltungs-Komponente erläutert.

Benutzer-Interaktion

Die Benutzer-Interaktion muss die erhobenen Informationen zur Visualisierung aufbereiten, s. d. der mit dem Management beauftragte Mitarbeiter mit übersichtlichen Anzeigen durch seine Arbeitsprozesse geführt wird. Neben der reinen Anzeige der Informationen müssen diese ebenfalls geändert werden können. Weiterhin muss die Konfiguration, das ist z. B. das Erstellen von Installations-Paketen und die Vergabe von Installations-Berechtigungen, des Management-Systems durch geeignete Visualisierungen erstellt, bzw. editiert werden können.

Das Gateway nutzt für diese Aufgaben die Funktionalität des Tomcat-Servers, Web-Seiten dynamisch zu erstellen. Ebenfalls unterstützt wird es durch den Einsatz des Front-Controllers von MyFaces. Dieser garantiert die Modularität der angezeigten Elemente durch Trennung der Programm-Logik von der Konfiguration der Anzeige-Elemente(s. Kap. 6.1.1), s. d. die Visualisierung späteren Bedürfnissen angepasst und erweitert werden kann. Die Konfiguration des Front-Controllers erfolgt in der Datei `WEB-INF/faces-config.xml`. Hier werden zum Einen die Beans, sog. Managed-Beans, registriert, die die Daten-Quelle für die Anzeige-Elemente darstellen, zum Anderen wird ein Navigations-Pfad durch die einzelnen Ansichten des Programms erstellt. Neben der automatischen Navigation aufgrund von aufgetretenen Ereignissen wird der Anwender in der Navigation durch ein Menü unterstützt(s. Abbildung A.5).

Da die Anwendung vor unauthorisierten Zugriff geschützt werden soll, wird sie mittels der, in Tomcat integrierten, Schutzmaßnahmen gesichert(s. Apache Software Foundation (2006b)), s. d. der Inhalt des Verzeichnisses `secure` erst nach einer Authentifizierung zugänglich gemacht wird. Hierfür wird dem Anwender ein Dialog gegeben dessen Konfiguration in Listing B.2 beschrieben ist. Zuvor wird dem Anwender eine Start-Ansicht präsentiert, in der er nur zu unkritischen Bereichen, z. B. der Dokumentation, Zugang hat.

Authentifiziert sich ein Anwender erfolgreich, so wird durch den Filter `DispatcherFilter` zunächst ein Zugang zu den Management-Daten und externen Management-Systemen in Form eines `IOController` aufgebaut und in die Sitzung des Benutzers gebunden, s. d. der Benutzer, während er angemeldet ist, anwendungsweit Zugriff auf die hinterlegten Daten hat. Weiterhin werden erweiterte Navigations-Möglichkeiten in dem o. g. Menü freigeschaltet. Er hat nun Zugang zu den Ansichten der einzelnen Management-Funktionen(s. Abb. A.6, A.8, A.10), den Berichten und der Suche(s. Abb. A.12), sowie der Konfiguration(s. Abb. A.15).

Um eine einfache Navigation durch die Management-Objekte zu ermöglichen, werden die Ansichten der Management-Funktionen jeweils durch eine Baum-Darstellung der vorhandenen Objekte, sowie einem Bereich für deren Inhalt, bzw. Details gebildet. Dafür wird die Ansicht von jeweils zwei Beans mit Daten versorgt:

1. Die Daten der Management-Objekte werden, abhängig von ihrem Typ, durch eine `AssetViewBean`, `MgmtViewBean` oder `ObjectViewBean` bereitgestellt. Diese enthalten Methoden zur Anzeige der Attribute der Management-Objekte(s. Kap. 5.4) und stellen bei Verknüpfungen zwischen verschiedenen Management-Objekten, z. B. bei der Anzeige von Mitgliedern einer Gruppe, Funktionen zur Delegierung der Ansichten bereit. Des weiteren ermöglichen Sie die Erstellung von neuen und Änderung der vorhandenen Management-Objekte.

2. Die Daten der Baum-Struktur werden, abhängig von dem Typ ihrer angezeigten Management-Objekten, durch eine `AssetTreeBean`, `MgmtTreeBean` oder `ObjectTreeBean` geliefert. Diese benutzen jeweils den, an den Kontext gebundenen, `IOController`, um die Management-Objekte der von der Datenhaltungs-Komponente zu erfragen. Wird ein Objekt in dem Baum ausgewählt, so wird dessen `id`(s. Kap. 5.4) in eine Variable des Kontextes geschrieben. Dies wird durch den `TreeAttributeListener` registriert und die entsprechende Bean zur Darstellung der Objekt-Details erhält eine Nachricht zur Aktualisierung ihres Daten-Bestandes.

Die `ReportBean` dient der vordefinierten Auswertung der vorhandenen Management-Objekte. Hierfür wertet fragt sie die vorhandenen Management-Objekte vom `IOController` ab und untersucht sie nach vordefinierten Kriterien. Dies können z. B. enthaltene Fehler-Meldungen bei Installations-Ergebnissen oder nicht genutzte Ressourcen sein. Da die Abfrage und Auswertung *aller* Management-Objekte aus Sicht der Anwendungs-Performance und des Ressourcen-Verbrauchs ungünstig ist, erfolgt die Erstellung nur auf explizite Anfrage.

Die Suche wird durch die `SearchBean` mit Daten versorgt. Dabei wird der Anwender bei der Auswahl der Such-Parameter unterstützt, indem ihm von der `SearchBean` die möglichen Ergebnisse stufenweise zur Auswahl zur Verfügung gestellt werden(Abbildung A.13). Die Management-Objekte, die sich in der Ergebnismenge befinden, werden dem Anwender auf der Such-Seite angezeigt und eine schnelle Navigation zu ihnen ermöglicht.

Die Parameter der Konfiguration werden durch die `ConfigBean` vorgehalten und können durch sie über die Konfigurations-Seite geändert werden. Sie benutzt den `PersistanceHandler` zum Zugriff auf die Konfigurations-Parameter, die in der Datei `WEB-INF/SOFAS.xml` hinterlegt sind(s. Kap. D).

Beispiel für modulare Erweiterungen der Benutzer-Interaktion Um die Modularität der Interaktions-Komponente aufzuzeigen wurde in die Visualisierung der Assets ein Anzeige-Element eingebunden, das den Aufruf eines Fernsteuerungs-Applets erlaubt, wenn der Computer, den das Asset repräsentiert, über das Netzwerk erreichbar ist. Das Desktop-Management wird dabei also um Funktionen des Helpdesk-Management erweitert.

Neben der Erstellung einer geeigneten Beschreibung der Ansicht in Form einer JSP oder Integration der Beschreibung in eine bestehende JSP(s. Listing B.4), muss, wenn sich die Erweiterung in einer eigenen JSP befindet, ein Navigations-Pfad zu ihr erstellt werden(s. Listing B.5). Für ein einheitliches Design der Ansichten wurden die Kopf- und Fuß-Zeilen der Seiten in dem Verzeichnis `inc`(s. Kap. 6.3.1) der Anwendung abgelegt, s. d. sie sich von dort einfach in neue JSPs integrieren lassen.

Als Fernsteuerungs-Applet wird der *Java Remote Desktop Protocol Client* von Elusiva[9] genutzt.

Agenten-Interaktion

Die Aufgaben der Agenten-Interaktion bestehen in der Kommunikation mit dem Agenten. Dem Gateway werden vom Agenten die Ergebnisse der Inventarisierung und der Installations-Aufträge übermittelt. Diese müssen einem Asset zugeordnet und in der Management-Datenbank gespeichert werden. Des weiteren muss es auf Anfrage des Agenten Installations-Aufträge übermitteln. Hierfür muss das Gateway durch gegebene Regeln eine Auswahl an Software-Paketen treffen und sie an den Agenten übermitteln.

Die Kommunikation mit dem Agenten wird mittels eines Web-Services auf Basis von Axis realisiert. Die Sicherung der Kommunikation erfolgt über WSS4J. Die hierfür benötigten

[9] `http://www.elusiva.com/opensource/`

Klassen befinden sich unter dem Paket `org.clonewars.sofas.service`. Ein schematischer Aufbau der Agenten-Interaktion ist in Abbildung 6.5 dargestellt.

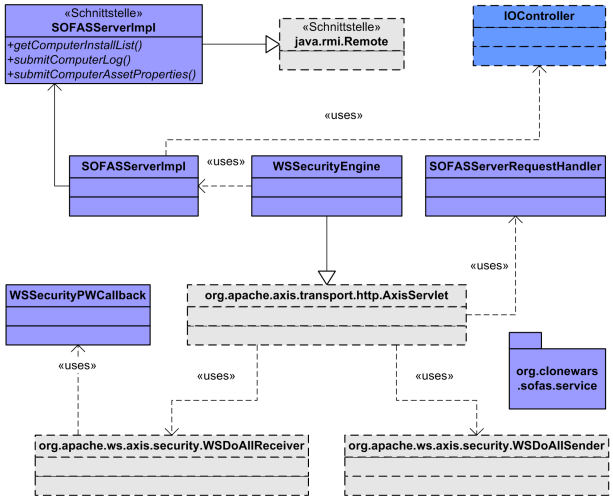


Abbildung 6.5.: Klassen-Diagramm des Gateways zur Interaktion mit den Agenten

Um Konfigurations-Parameter von Axis und WSS4J intern durch die Konfiguration des Gateways(s. Kap. D) ändern zu können, wurde das `AxisServlet`, das die Basis der Axis Engine(Abbildung 6.3) bildet, durch die Klasse `WSSecurityEngine` erweitert. Sie ermöglicht zum Einen die Konfiguration von WSS4J und prüft zum Anderen die Konsistenz der Gesamt-Konfiguration anhand der einzelnen Parameter. Da sie ein Servlet ist, muss sie im Anwendungs-Kontext registriert werden. Dies wird, wie in Listing B.3 dargestellt, in der Datei `WEB-INF/web.xml`.

Die Konfiguration der Axis Engine wird beim Starten aus der Datei `WEB-INF/server-config.wsdd` gelesen, bzw. beim Beenden dort gespeichert. In ihr werden u. a. die Handler definiert und an die Ketten der Nachrichtenverarbeitung von Axis gebunden(Abbildung 6.3). Die Handler `WSDoAllReceiver` und `SOFASServerRequestHandler` werden dabei in die Dienst-Kette der, an das `AxisServlet` gestellten, Anfragen und `WSDoAllSender` in die Dienst-Kette der, vom `AxisServlet` gesendeten, Antworten integriert.

Die Handler `WSDoAllSender` und `WSDoAllReceiver` sind die Komponenten von WSS4j, die entsprechend ihrer Konfiguration, die ein- und ausgehenden SOAP-Nachrichten ver- und entschlüsseln. Des weiteren werden die Nachrichten mit Zertifikaten, Zeitstempeln und Benutzernamen-Token in Senderichtung versehen, bzw. in Empfangsrichtung abgeglichen. Zum Erhalten der passenden Schlüssel nutzt der `WSDoAllReceiver` dabei die Klasse `WSSecurityPWCallback`.

Der Handler `SOFASServerRequestHandler` überprüft eingehende Anfragen auf erfolgreiche Authentifizierung. Daraufhin bindet sie die Datenhaltungs-Komponente in Form eines

`IOController` an die Anfrage, s. d. sie in späteren Bearbeitungsschritten Zugriff auf die Datenhaltung hat.

Die zentrale Implementation für die Interaktion mit dem Agenten wird in der Klasse `SOFASServerImpl` vorgenommen. Sie implementiert die gleichnamige Schnittstelle, die die Methoden des Web-Services spezifiziert. Die Methoden sind dabei:

- *getComputerInstallList*: Der Aufruf dieser Methode ermittelt die zu installierenden Software-Pakete(s. u.). Das Ergebnis wird als Array vom Typ `InstallObjectBean` an den Agenten übermittelt. Eine `InstallObjectBean` ist ein Container für die reduzierten Informationen eines `SoftwarePackage`(s. Kap. 5.4).

  Das Auswahlverfahren der Installations-Aufträge berücksichtigt mehrere Aspekte. Einschränkungen die in den `SoftwarePackage`, z. B. Abhängigkeiten von anderen Paketen, der mit dem Paket assoziierten `ManagementGroup`, z. B. den Aktivierungs-Status des Pakets, müssen ebenso berücksichtigt werden, wie die Möglichkeit, das das Paket bereits zur Installation an den Agenten ausgeliefert wurde. Die Berechtigung eines Agenten zur Installation erfolgt über den Abgleich, ob die Arbeitsstation des Agenten sich in einer der, in der `ManagementGroup` spezifizierten, Berechtigungsgruppen befindet.

- *submitComputerLog*: Durch diese Methode übermittelt der Agent ein Array der mit Installations-Ergebnissen versehenen o. g. `InstallObjectBean`s zurück an das Gateway. Das Gateway wandelt sie daraufhin in `LogEntry`s(s. Kap. 5.4) um und ordnet sie dem den Agenten entsprechenden Management-Objekt zu.

- *submitComputerAssetProperties*: Durch diese Methode übermittelt der Agent ein Array von ermittelten Inventar-Informationen in Form von `InventoryObjectBean`s an das Gateway. Das Gateway wandelt sie daraufhin in `AssetEntry`s(s. Kap. 5.4) um und ordnet sie dem den Agenten entsprechenden Management-Objekt zu.

Im Fehlerfall übertragen die Methoden eine geeignete Fehlermeldung in Form eines `AxisFault` an den anfragenden Agenten. Um die, vom Agenten übermittelten, Informationen einem Management-Objekt zuordnen und speichern zu können, sowie zur Ermittlung der an den Agenten zu übertragenden Daten, nutzt die Klasse `SOFASServerImpl` den an die Anfrage gebundenen `IOController`.

Datenhaltungs-Komponente

Die Datenhaltung bietet Zugriff auf die Benutzerverwaltung des Netzwerk-Systems, die Management-Datenbank und die Konfigurations-Datenbank des Gateways. Die Konfigurations-Datenbank des Gateways besteht aus einer XML-Datei(s. Kap. D) die durch die Klasse

PersistanceHandler, wie die auf S. 66 beschriebene, realisiert ist. Die Schnittstellen auf die externen Management-Systeme und die Management-Datenbank sind, wie in (s. Kap. 6.1.4) erläutert, durch den Zugriff auf OpenLDAP-Server realisiert, lassen sich aber auf einfache Art über die Konfigurations-Seiten des Gateways(Abbildung A.15) austauschen.

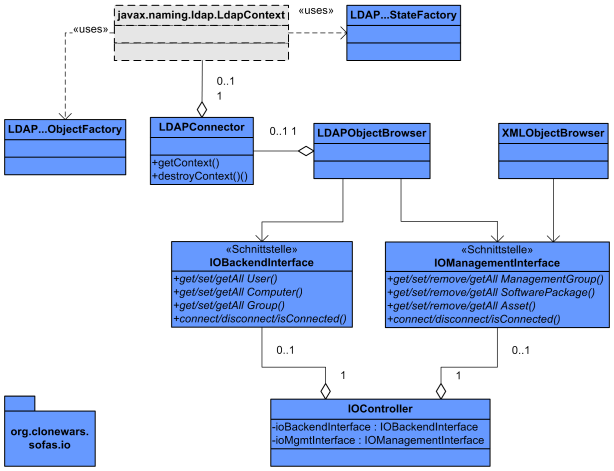


Abbildung 6.6.: Klassen-Diagramm des Gateways zur Datenhaltungs-Komponente

Die Basis-Klasse des Management-Systems für den Zugriff auf den LDAP-Server ist der IOController. Er bindet die implementierte Management-Datenbank und das externe Management-System über deren Schnittstellen IOBackendInterface und IOManagementInterface. Diese stellen Methoden zum Zugriff auf die verschiedene Objekte und zum Verbindungsaufbau mit den Daten-Banken zur Verfügung. Die tatsächlichen Implementations-Klassen der Schnittstellen werden dem IOController von der Konfiguration des Gateways vorgegeben. Der IOController stellt für den Prototypen Methoden zum Zugriff auf die Management-Objekte bereit. Zusätzlich bietet er Möglichkeiten zur Vorverarbeitung der abzurufenden Informationen an.

Da sowohl die Management-Datenbank, als auch die Benutzerverwaltung durch einen Zugriff auf einen OpenLDAP-Server realisiert werden soll, implementiert die Klasse LDAPObjectBrowser die Schnittstellen IOBackendInterface und IOManagementInterface und wird in der Ausgangs-Konfiguration(s. Kap. D) als Implementations-Klasse für die Verbindungen des IOController spezifiziert. Der LDAPObjectBrowser benutzt einen LDAPConnector(s. u.), um eine Verbindung zu den LDAP-Servern aufzubauen. Das LDAP bietet über wohldefinierte Filter eine performante Möglichkeit der Suche nach Einträgen(Zörner, 2005, S. 62f). Dieses macht sich der LDAPObjectBrowser über Such-Vorlagen zu Nutze, um angeforderte Objekte zu erhalten.

Der LDAPConnector stellt die Verbindung zum LDAP-Server her. Er nutzt dafür das Java Native Directory Interface(JNDI). Die Konfiguration der Schnittstelle, das betrifft hauptsächlich

Verschlüsselungs-, Authentifizierungs- und Kontext-Optionen, wird aus der Konfiguration des Gateways entnommen. Der `LDAPConnector` stellt einen `LdapContext` zur Verfügung, s. d. der `LDAPObjectBrowser` durch JNDI Zugriff auf den Server nehmen kann.

Um die im Management-System genutzten Objekte der Benutzer-Verwaltung, sowie des Software- und Asset-Managements(s. Kap. 5.4) zwischen dem Informationsmodell des Management-Systems und dem Daten-Modell des LDAP-Servers(s. Kap. 6.4) zu wandeln, bietet JNDI Fabrik-Klassen, sog. *State-* und *Object-Factory*s, an. Diese müssen bei JNDI registriert werden. JNDI prüft beim Abruf von LDAP-Einträgen und beim Speichern von Java-Objekten, ob eine registrierte Fabrik-Klasse auf die entsprechende Operation anwendbar ist. Die erste zutreffende Klasse wird daraufhin zur Wandlung genutzt. Für die Wandlung von LDAP-Einträgen zu Management-Objekten wurden die Klassen `LDAPGroupObjectFactory`, `LDAPManagementObjectFactory` und `LDAPUserObjectFactory` erstellt, für die gegenteilige Wandlung `LDAPGroupStateFactory`, `LDAPManagementStateFactory` und `LDAPUserStateFactory`. Die Registrierung der Factory-Klassen wird durch den `LDAPConnector` vorgenommen.

6.4. Implementation des Informationsmodells der Management-Datenbank

Wie in Kap. 5.3.2 beschrieben wurde, sollen die Management-Daten in einer Verzeichnis-Struktur eines LDAP-Servers abgelegt werden.

Um das Informationsmodell in der Management-Datenbank abzubilden, muss auch für den LDAP-Server ein Daten-Modell erstellt werden, das die Objekte und deren Attribute in die Verzeichnisse des LDAP-Server strukturieren kann. Hierfür ist eine Schema-Erweiterung erstellt worden, die in Anhang B.1 aufgezeigt wird. Ein Schema beschreibt, ähnlich der MIB-Dateien von SNMP und der MOF-Dateien von CIM, die Syntax und Semantik der abgelegten Daten auf einem LDAP-Server. Der Aufbau einer Schema-Definition ist in RFC4512 beschrieben.

Mit RFC2713 existiert ein Schema, um serialisierte Java Objekte in LDAP-Verzeichnissen zu speichern. Dies ermöglicht eine einfache Speicherung der Objekte. Allerdings liegen die gespeicherten Daten in diesem Fall weder in einer für den Menschen, noch für nicht-Java Programme lesbaren Form vor. Unter der Prämisse, eine Anwendung mit offenen Schnittstellen zu entwickeln, ist die Entwicklung eines eigenen Schemas also dennoch nötig.

Die implementierte Schema-Definition beschreibt dabei, analog zu dem entwickelten Informationsmodell(s. Kap. 5.4), drei ablegbare Objekte:

1. Ein *sofasSoftwarePackageContainer* dient der Aufnahme eines Objektes vom Typ *SoftwarePackage* und seiner Attribute.

2. Ein *sofasInstallerGroupContainer* dient der Aufnahme eines Objektes vom Typ *ManagementGroup* und seiner Attribute.
3. Ein *sofasAssetContainer* dient der Aufnahme eines Objektes vom Typ *Asset* und seiner Attribute.

7. Schluss

In diesem Kapitel werden zunächst die gewonnenen Erkenntnisse der Arbeit zusammengefasst. Abschließend wird ein Ausblick auf die Weiterentwicklung der Management-Systeme und des entwickelten Prototypen gegeben.

7.1. Zusammenfassung

Die Notwendigkeit einer Werkzeug-gestützten Verwaltung der System-Komponenten nimmt mit dem Umfang der System-Landschaft zu. Die Implementation eines Verwaltungs-Systems bewirkt dabei nicht nur durch Zeiteinsparungen einen betriebswirtschaftlichen Vorteil, sondern führt auch durch Vereinheitlichung der Prozesse zu einem in sich homogen agierenden Gesamt-System.

Die Mannigfaltigkeit der genutzten Hardware- und Software-Plattformen, sowie Dienste machen in den Verwaltungs-Systemen eine breite Integration der vorhandenen Management-Schnittstellen nötig. Dies erfordert standardisierte und offene Management-Architekturen. Ein einfacher und flexibler Aufbau sichert dabei eine breite Integration. Die Integration von Sicherheitskonzepten ermöglicht ihre Integration auch in kritischen Bereichen der Systemlandschaft.

Die Verwaltung kann auf mehreren Ebenen erfolgen, die durch verschiedene unterstützende Werkzeuge abgebildet werden können. Das Desktop-Management und mit ihm das Software- und Asset-Management bilden die Basis für die Verwaltung der Endgeräte auf der Benutzerseite des Netzwerk-Systems.

Bestehende Desktop-Management Systeme lassen sich durch starre Anforderungen an die verwalteten Endgeräte und das Netzwerk-System nicht immer auf allen verwalteten Komponenten einheitlich nutzen. Die fehlende Interoperabilität mit anderen vorhandenen Verwaltungs-Systemen wirkt dem Prinzip des integrierten und einheitlichen Management-Systems entgegen. Nicht vorhandene Modularität in dem Umfang der Verwaltung bewirken eine schlechte Anpassungsfähigkeit auf die individuellen Bedürfnisse an das Management-System.

Der entwickelte Prototyp versucht durch die Schaffung von Modularität in Darstellung, Funktion und angebundene Datenquellen, sowie Plattform-Unabhängigkeit, ein leicht anpassbares und erweiterbares Management-System zu bilden. Modularität lässt sich dabei durch Implementierung von Schnittstellen erzeugen. Plattform-Unabhängigkeit lässt sich nicht allein durch Nutzung von Plattform-unabhängigen Middlewares und Programmiersprachen erreichen, sondern muss in bestimmten Bereichen durch geeignete Wahl der benutzten Datentypen und Funktionen implementiert werden.

Der Prototyp beinhaltet Module für die Software-Verteilung und Inventar-Erfassung. Das Management-System wird dabei über eine Client/Server-Architektur realisiert. Die Kommunikation erfolgt auf Basis von Web-Services. Die Darstellung erfolgt auf Basis von dynamisch erzeugten HTML-Seiten, die dem Nutzer die Konfiguration des Systems ermöglichen, einen Überblick über den Installations-Stand der Endgeräte vermitteln und auf mögliche Fehler hinweisen.

Zur Erfassung der Management-Informationen, war es notwendig ein Daten-Modell zu schaffen, dass die Informationen für das Management-System und seine Daten-Bank abbilden kann. Hierbei wurde sich an den Daten-Modellen bereits bestehender Systeme orientiert und diese, in für die Erfüllung der Anforderungen benötigter reduzierter Form, übernommen. Dies beinhaltet u. a. Möglichkeiten zum Ausbilden von Abhängigkeiten und Berechtigungen.

7.2. Ausblick

Der entwickelte Prototyp vermag eine anfängliche Übersicht über die Leistungsfähigkeit von Software- und Inventar-Management-Systemen zu geben. Mit steigender Implementationstiefe, lassen sich die unterstützten Funktionen durch das modulare Konzept nahezu beliebig erweitern. Erste Angriffspunkte können in diesem Fall z. B.:

- der Ausbau des Daten-Modells sein, s. d. die verteilten Software-Pakete um weitere Attribute(z. B. Informationen über unterstützte Betriebssysteme, Zählungen der Installationen zur Implementation Lizenz-Verwaltungen, etc.) ergänzt werden.
- der Aufbau einer Kommunikations-Schnittstelle für die Gateway-Gateway-Kommunikation sein, s. d. verschiedene Gateways miteinander kommunizieren können und sich somit besser zur Abbildung von Domänen-Strukturen eignen.
- der Ausbau der Kommunikations-Schnittstelle zwischen Gateway und Agent sein, z. B. zur ferngesteuerten Rekonfigurationen des Agenten.
- die Implementation weiterer Anbindungen an Datenquellen sein. Dies können andere unterstützte Datenbanken oder weitere Anbindungen an Management-Systeme sein.

- die Einführung einer automatischen Deinstallation durch die Rücknahme einer Änderung der Software-Konfiguration sein.
- die Implementation weiterer Aspekte des Desktop-Management, z. B. des Lizenz-Management, sein.

Um das Problem der fehlenden Interoperabilität zwischen Management-Systemen und deren Anpassungsfähigkeit an individuelle Bedürfnisse zu lösen, müssen Management-Systeme i. A. durch die Einführung einer generellen Management-Plattform, die über offene Schnittstellen modular um Verwaltungsfunktionen erweitert werden kann, integriert werden. Dies kann durch einen konsequentem Aufbau nach dem SOA-Paradigma(Service orientated Architecture), d. h. in sich abgeschlossene, dynamisch angebundene Netzwerk-basierte Dienste mit offenen Schnittstellen, die Plattform-unabhängig interagieren, erreicht werden.

A. Screenshots der entwickelten Anwendung

Agent: Status-Anzeige

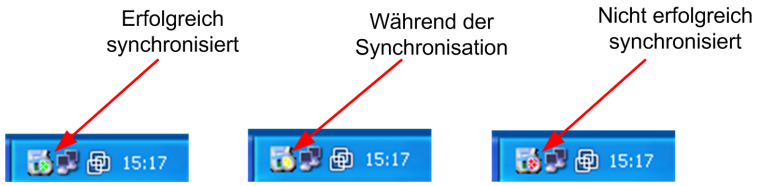


Abbildung A.1.: Agent: Status-Anzeige im System-Tray

Agent: Tray-Menü und Anmelde-Dialog

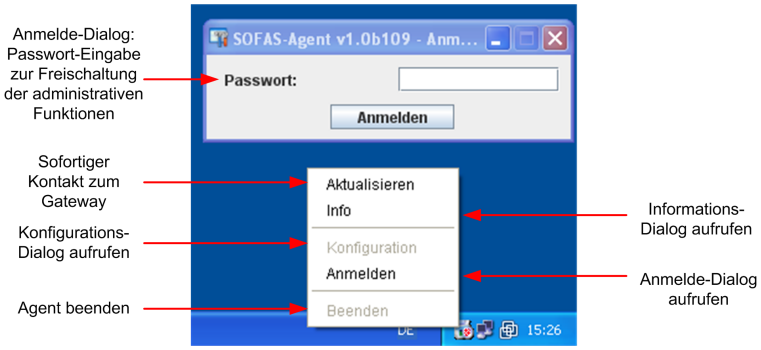


Abbildung A.2.: Agent: Tray-Menü und Anmelde-Dialog

Agent: Konfigurations-Dialog

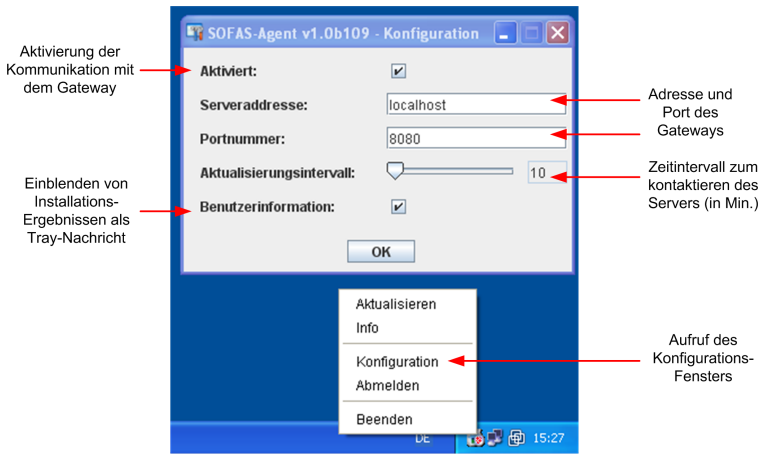


Abbildung A.3.: Agent: Konfigurations-Dialog

Agent: Informations-Dialog

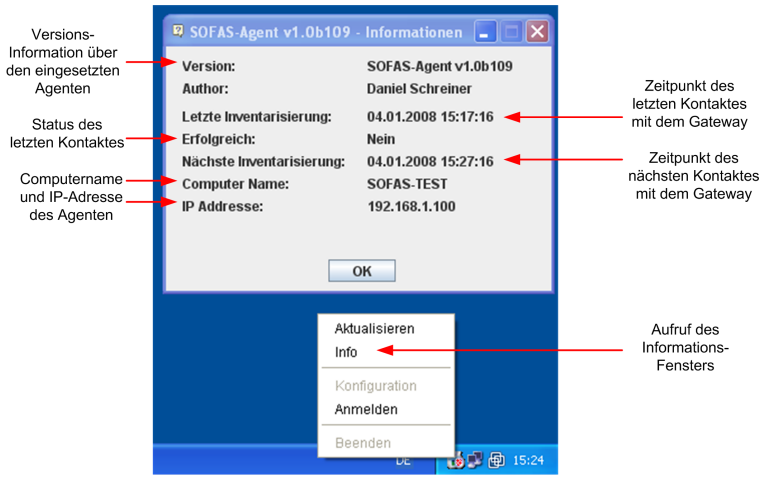


Abbildung A.4.: Agent: Informations-Dialog

Gateway: Menü-Struktur

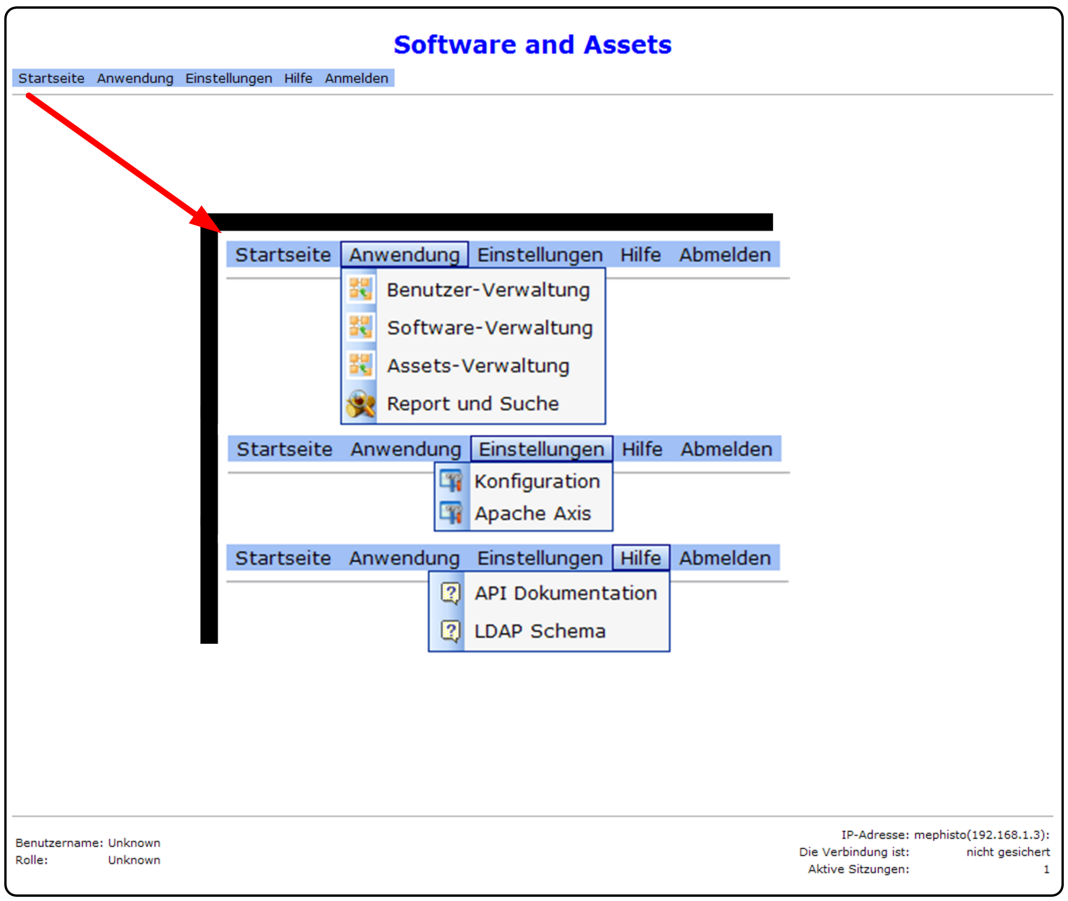


Abbildung A.5.: Gateway: Menü-Struktur

Gateway: Benutzer-Verwaltung

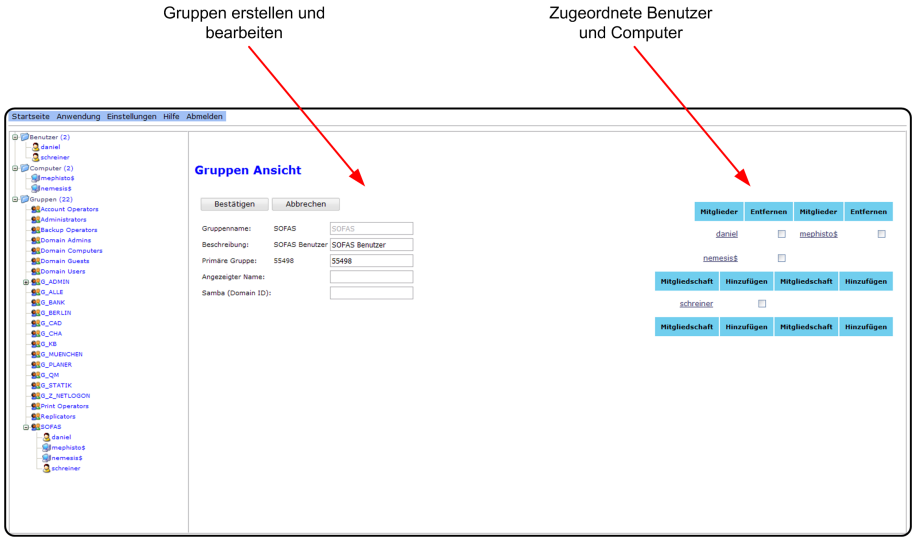


Abbildung A.6.: Gateway: Berechtigungs-Gruppen

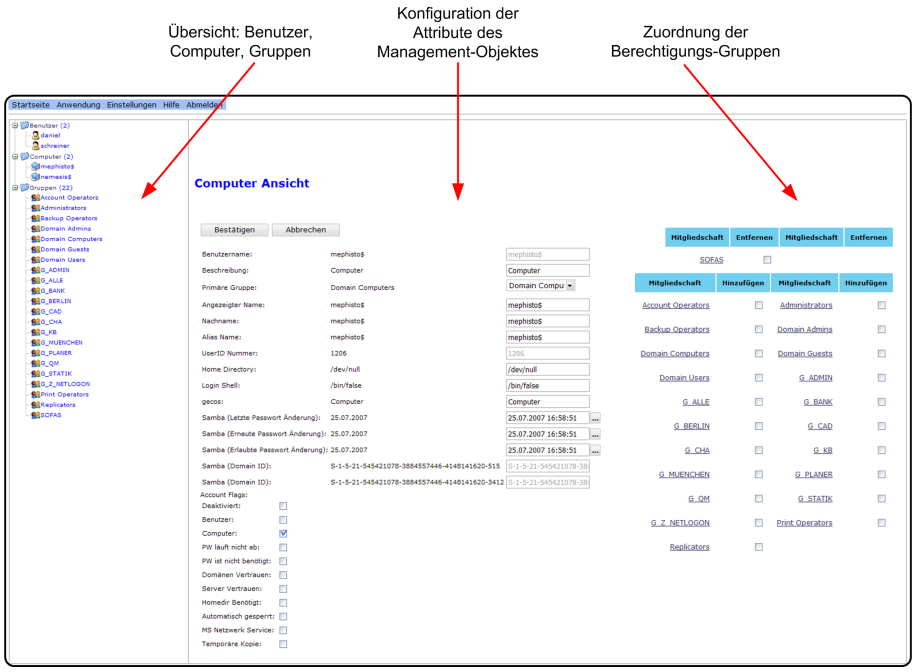


Abbildung A.7.: Gateway: Computer- und Benutzer-Anzeige

Gateway: Asset-Verwaltung

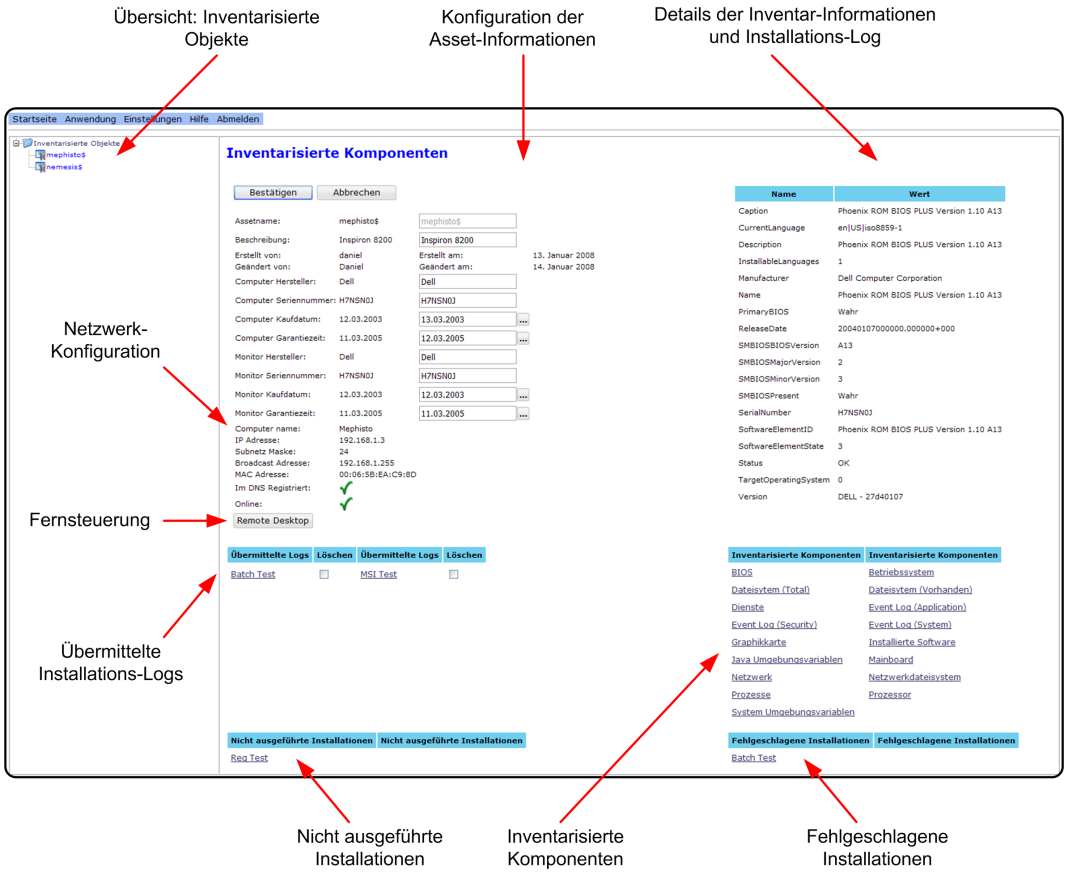


Abbildung A.8.: Gateway: Asset-Anzeige (Übersicht)

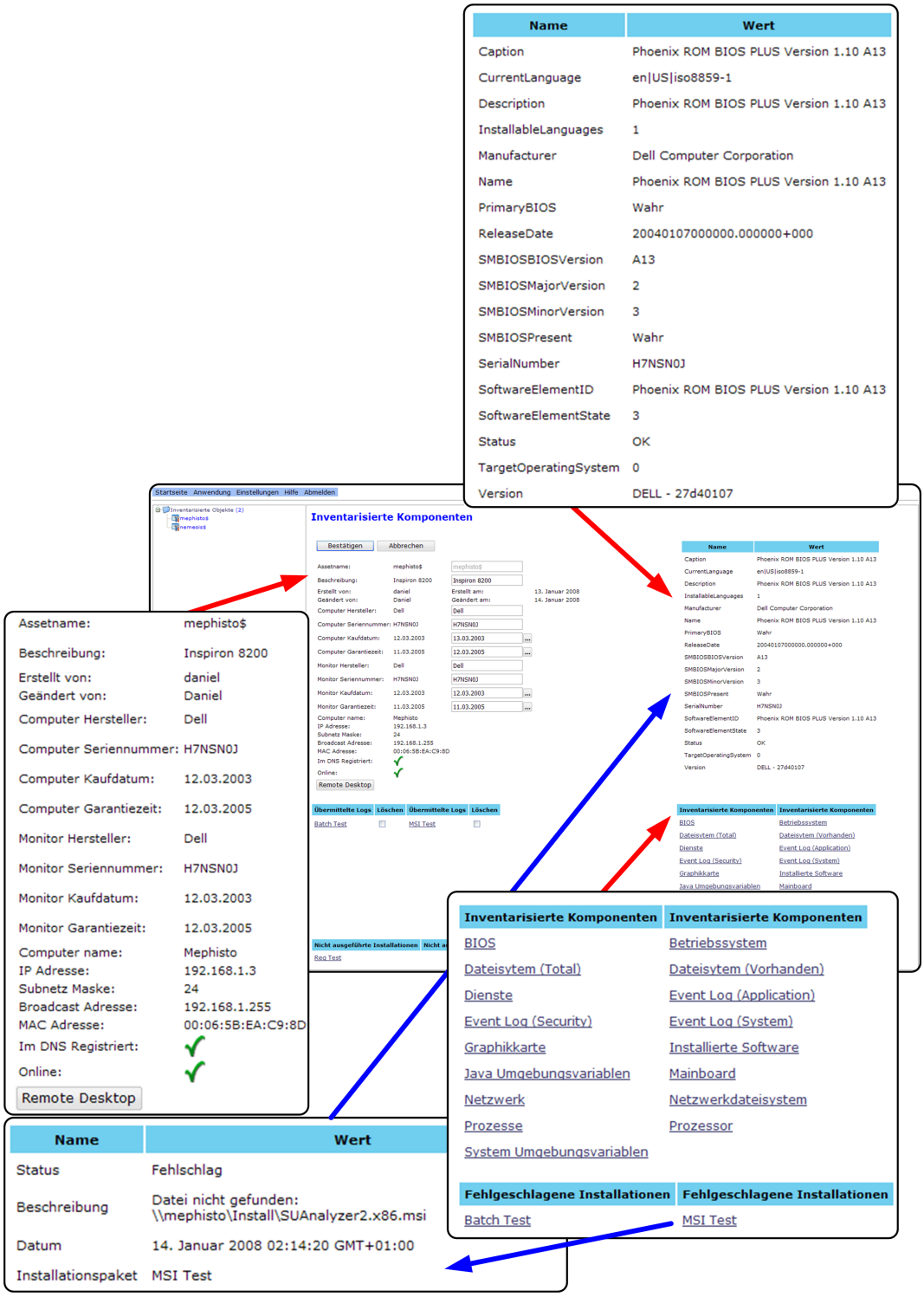


Abbildung A.9.: Gateway: Asset-Anzeige (Details)

Gateway: Software-Verwaltung

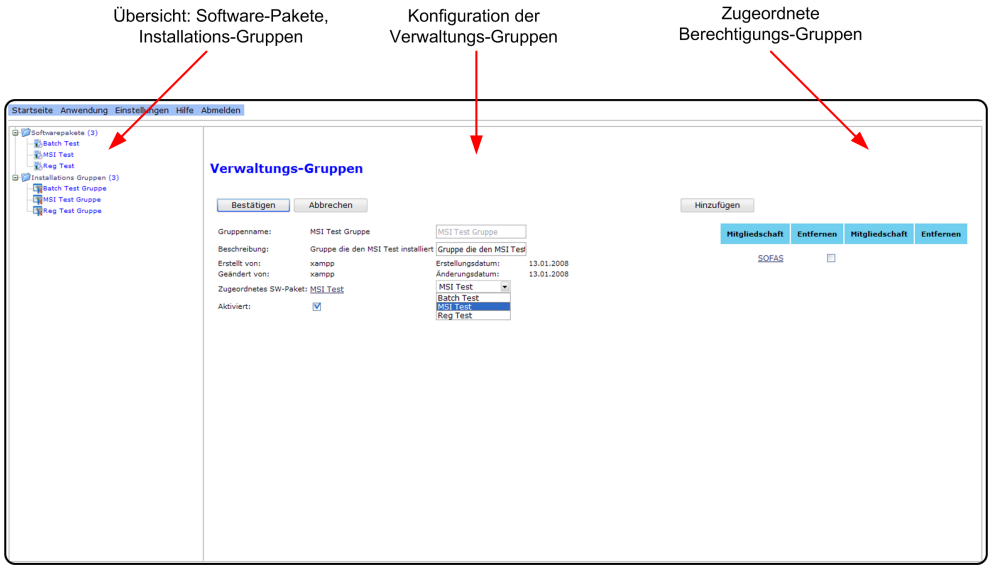


Abbildung A.10.: Gateway: Installations-Gruppen

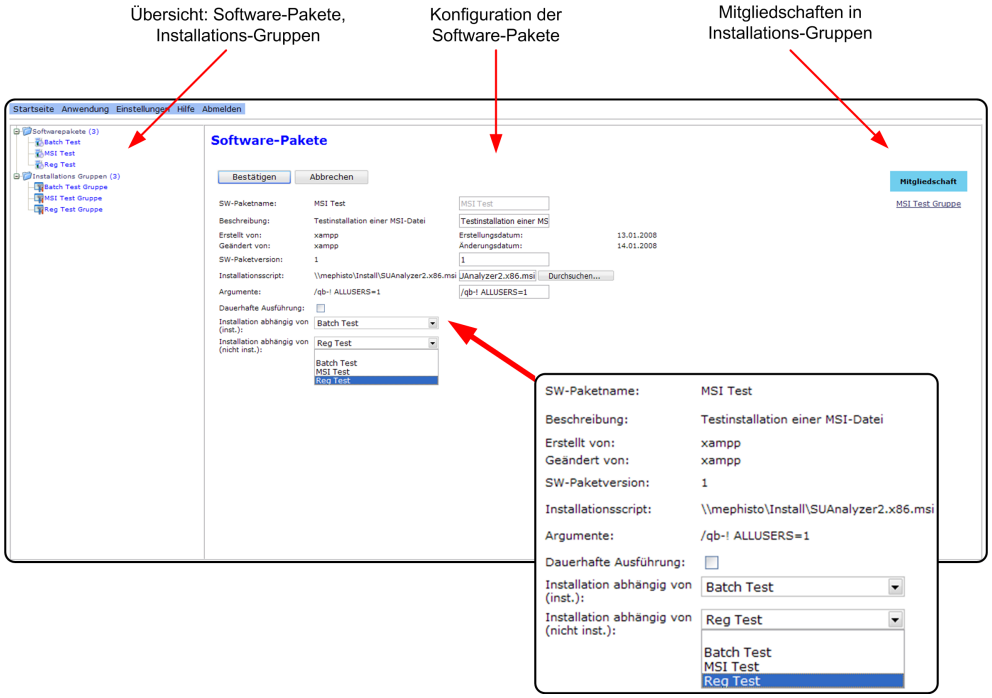


Abbildung A.11.: Gateway: Software-Pakete

Gateway: Reports und Suche

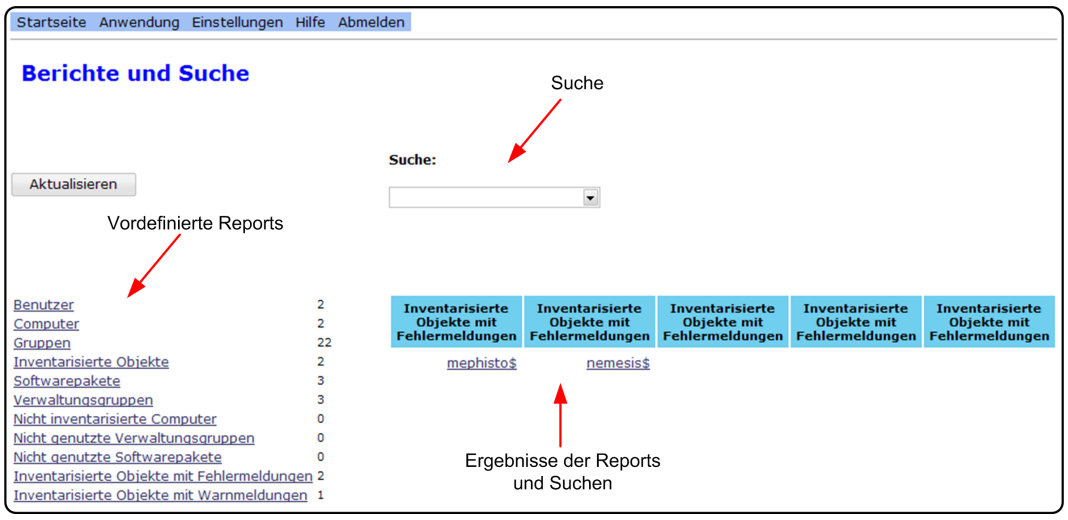


Abbildung A.12.: Gateway: Reports und Suche

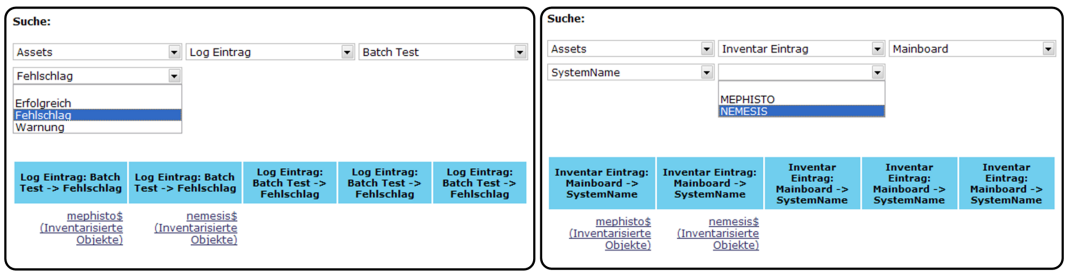


Abbildung A.13.: Gateway: Beispiel für die Suche

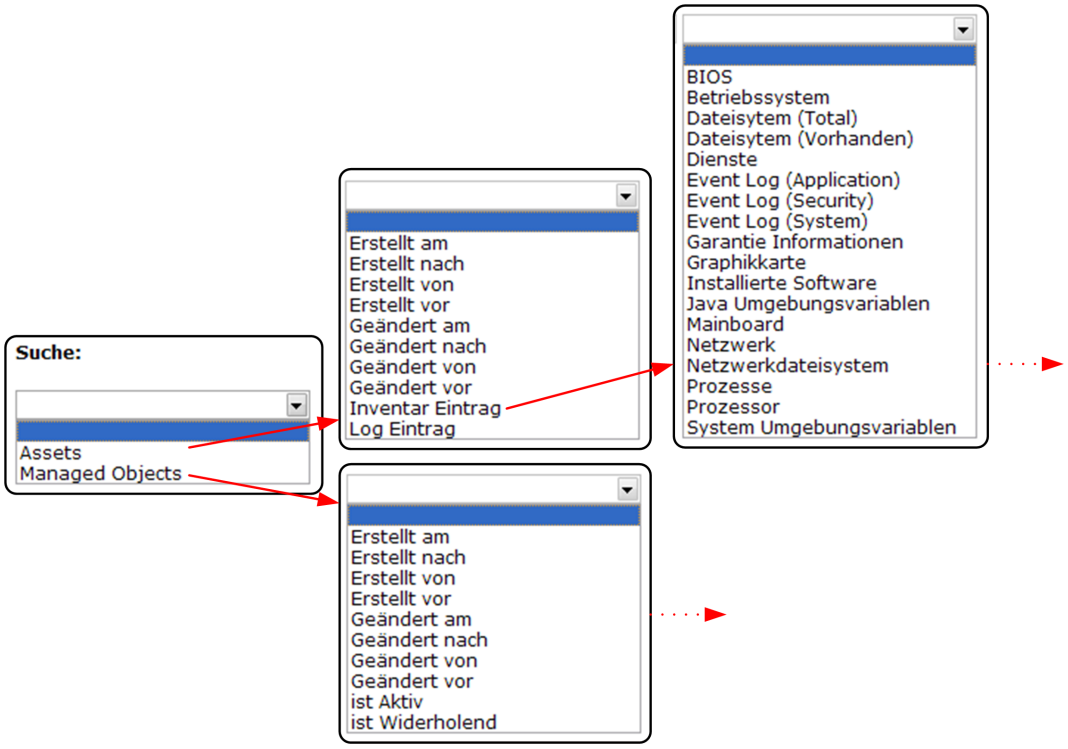


Abbildung A.14.: Gateway: Aufbau der Suche

Gateway: Konfiguration

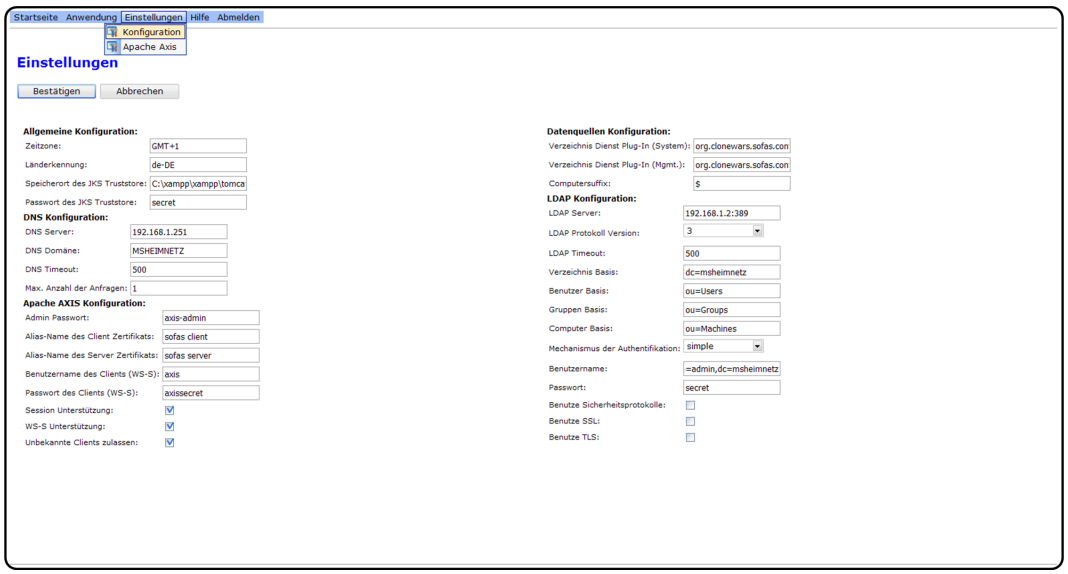


Abbildung A.15.: Gateway: Konfiguration

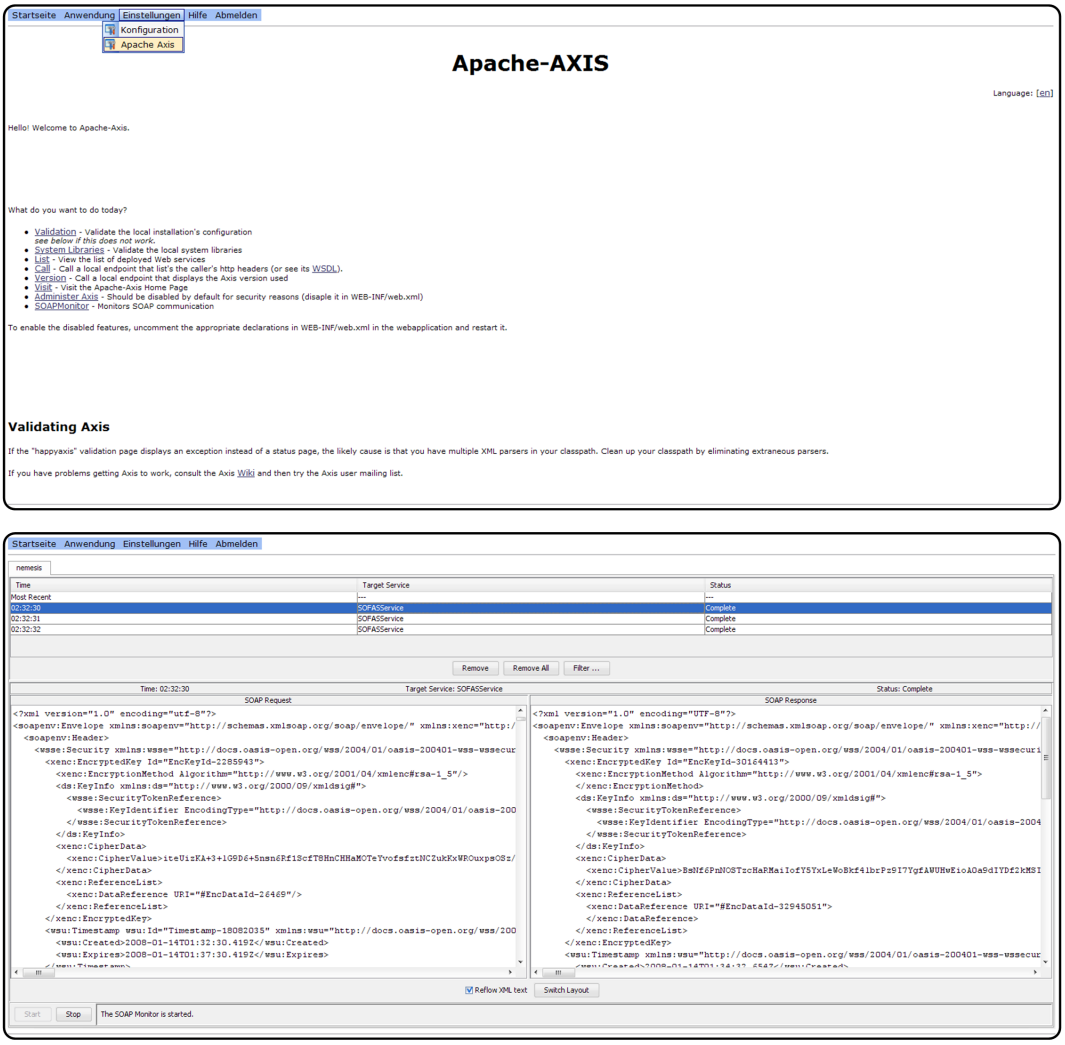


Abbildung A.16.: Gateway: Konfiguration(Axis und SOAP-Monitor)

B. Code-Beispiele

LDAP Schema Definition

Der in Listing B.1 gezeigte Code stellt die Definition des für den Prototypen erstellten Schema dar. Die Definition erfolgte nach RFC4512 und RFC4517. Der ASN.1-kodierter Namensraum `1.3.6.1.4.1.29611` ist bei der IANA reserviert und sollte daher nicht mit vorhandenen Schema-Definitionen kollidieren. Eine Veranschaulichungs-Beispiel ist in Abbildung B.1 zu sehen.

Listing B.1: LDAP-Schema Definition

```
1  ########################################################################
   ##                Attributes for Installgroups                        ##
   ########################################################################

   attributetype ( 1.3.6.1.4.1.29611.1.1 NAME 'sofasInstallerID'
6    DESC 'A unique id for the referenced installer'
     EQUALITY caseExactIA5Match
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

   attributetype ( 1.3.6.1.4.1.29611.1.2 NAME 'sofasMemberUid'
11   DESC 'Specifying the members of a install group'
     EQUALITY caseExactIA5Match
     SUBSTR caseExactIA5SubstringsMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

16 attributetype ( 1.3.6.1.4.1.29611.1.5 NAME 'sofasIsActive'
     DESC 'Specifying if this installgroup is activated'
     EQUALITY caseExactIA5Match
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

21 ########################################################################
   ##                Attributes for Softwarepackages                     ##
   ########################################################################

   attributetype ( 1.3.6.1.4.1.29611.1.6 NAME 'sofasInstallerLocation'
26   DESC 'URL specifying the location of a software bundle'
     EQUALITY caseExactIA5Match
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

   attributetype ( 1.3.6.1.4.1.29611.1.7 NAME 'sofasInstallerArguments'
31   DESC 'Specifying the installation arguments of a software bundle'
     EQUALITY caseExactIA5Match
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

   attributetype ( 1.3.6.1.4.1.29611.1.8 NAME 'sofasInstallerVersion'
36   DESC 'Version of a software bundle'
     EQUALITY numericStringMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.36 )

   attributetype ( 1.3.6.1.4.1.29611.1.9 NAME 'sofasIsAlwaysInstall'
41   DESC 'Specifying if this software bundle should be executed once or many'
     EQUALITY caseExactIA5Match
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

   attributetype ( 1.3.6.1.4.1.29611.1.3 NAME 'sofasInstallIf'
46   DESC 'Specifying which other packages must have to be installed first'
     EQUALITY caseExactIA5Match
     SUBSTR caseExactIA5SubstringsMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

```
attributetype ( 1.3.6.1.4.1.29611.1.4 NAME 'sofasInstallIfNot'
  DESC 'Specifying which other packages must not have been installed'
  EQUALITY caseExactIA5Match
  SUBSTR caseExactIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

#########################################################################
##                      Attributes for Assets                          ##
#########################################################################

attributetype ( 1.3.6.1.4.1.29611.1.10 NAME 'sofasAssets'
  DESC 'A number of assets. They have to be of type id!!value'
  EQUALITY caseExactIA5Match
  SUBSTR caseExactIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.29611.1.11 NAME 'sofasInstallLog'
  DESC 'A number of assets. They have to be of type id!!date_numeric'
  EQUALITY caseExactIA5Match
  SUBSTR caseExactIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

#########################################################################
##                          Common Attributes                          ##
#########################################################################

attributetype ( 1.3.6.1.4.1.29611.1.12 NAME 'sofasDescription'
  DESC 'Specifying a description of this object'
  EQUALITY caseExactIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.29611.1.13 NAME 'sofasCreator'
  DESC 'Specifying the userID of the Creator'
  EQUALITY caseExactIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.29611.1.14 NAME 'sofasCreated'
  DESC 'Specifying the date of creation'
  EQUALITY numericStringMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.36 )

attributetype ( 1.3.6.1.4.1.29611.1.15 NAME 'sofasModifier'
  DESC 'Specifying the userID of the last modifier'
  EQUALITY caseExactIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.29611.1.16 NAME 'sofasLastModified'
  DESC 'Specifying a time when this entry was last changed'
  EQUALITY numericStringMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.36 )

#########################################################################
##                           Object Classes                            ##
#########################################################################

objectClass ( 1.3.6.1.4.1.29611.1.17 NAME 'sofasSoftwarePackageContainer'
  DESC 'Container for a sofasSoftwarePackage'
  SUP top
  STRUCTURAL
  MUST ( cn $ sofasInstallerLocation $ sofasInstallerVersion )
  MAY ( sofasDescription $ sofasCreator $ sofasLastModified $ sofasCreated $
        sofasModifier $ sofasInstallerArguments $ sofasIsAlwaysInstall $
        sofasInstallIf $ sofasInstallIfNot) )

objectClass ( 1.3.6.1.4.1.29611.1.18 NAME 'sofasInstallerGroupContainer'
  DESC 'Container for a sofasInstallerGroup'
  SUP top
  STRUCTURAL
  MUST ( cn $ sofasInstallerID $ sofasIsActive)
  MAY ( sofasDescription $ sofasCreator $ sofasCreated $ sofasModifier $
        sofasLastModified $ sofasMemberUid ) )

objectClass ( 1.3.6.1.4.1.29611.1.19 NAME 'sofasAssetContainer'
  DESC 'Container for a sofasAsset'
  SUP top
  STRUCTURAL
  MUST ( cn )
  MAY ( sofasDescription $ sofasCreator $ sofasCreated $ sofasModifier $
        sofasLastModified $ sofasAssets $ sofasInstallLog ) )
```

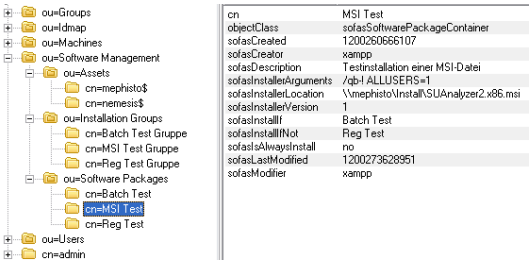


Abbildung B.1.: Beispiel für die Anwendung von Listing B.1

Konfigurations-Dateien

Listing B.2: Auszüge aus der Datei /WEB-INF/web.xml: Sicherheits-Einstellungen

```
1
    <security-constraint>
      <web-resource-collection>
        <web-resource-name>SOFAS User</web-resource-name>
        <url-pattern>/secure/*</url-pattern>
6       <http-method>GET</http-method>
        <http-method>POST</http-method>
      </web-resource-collection>
      <auth-constraint>
        <role-name>admin</role-name>
11    </auth-constraint>
    </security-constraint>


    <login-config>
16    <auth-method>FORM</auth-method>
      <realm-name>A Server Configuration Form-Based Authentication Area</realm-name>
      <form-login-config>
        <form-login-page>/insecure/login.jsf</form-login-page>
        <form-error-page>/insecure/loginError.jsf</form-error-page>
21    </form-login-config>
    </login-config>

    <security-role>
      <description>
26      The role that is required to log in to the Application: Adminrights
      </description>
      <role-name>admin</role-name>
    </security-role>
  </web-app>
```

Listing B.3: Auszüge aus der Datei /WEB-INF/web.xml: Konfiguration der SOAP Engine

```
  <!--    We use not the Standard servlet, instead we are usind a custum servlet
          defined below. This let's us configure the server on the fly...
    <servlet>
5     <servlet-name>AxisServlet</servlet-name>
      <display-name>Apache-Axis Servlet</display-name>
      <servlet-class>
          org.apache.axis.transport.http.AxisServlet
      </servlet-class>
10  </servlet>
  -->

    <servlet>
      <servlet-name>AxisServlet</servlet-name>
15    <display-name>Apache-Axis Servlet</display-name>
      <servlet-class>
          org.clonewars.sofas.service.axis.WSSecurityEngine
      </servlet-class>
    </servlet>
```

Beispiel für eine Erweiterung der Interaktions-Komponente

Listing B.4: Erweiterung einer JSP um ein zusätzliches Anzeige-Element

```
<t:commandButton
  rendered="#{assetViewBean.hostOnline && assetViewBean.networkInfoAvailable}"
  styleClass="button" value="#{messages['button_nettools_rdp']}"
  action="#{assetViewBean.actionRDP}"
  immediate="true" />
```

Listing B.5: Erweiterung der Datei /WEB-INF/faces-config.xml um einen zusätzlichen Navigationspfad

```
<navigation-rule>
  <description>
       Navigation for network tools.
  </description>
  <from-view-id>*</from-view-id>

  <navigation-case>
    <from-outcome>RDP_applet</from-outcome>
      <to-view-id>/secure/rdp/RDP.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

C. Installationshinweise

Systemvoraussetzungen

Systemvoraussetzungen für das Gateway:

- Sun Microsystems JRE v1.6+[1]
- Sun Microsystems Java Cryptography Extension (JCE)[2]
- Apache Tomcat v5.5.20[3]

Systemvoraussetzungen für den Agenten:

- Sun Microsystems JRE v1.6+
- Sun Microsystems Java Cryptography Extension (JCE)

Installation

Installationshinweise für das Gateway:

1. Installation der Sun Microsystems JRE v1.6.
2. Installation der Sun Microsystems Java Cryptography Extension. Beachten Sie dabei bitte die Hinweise in dem heruntergeladenen Archiv.
3. Installation und Konfiguration des Apache Tomcat v5.5.20[4].

[1] `http://java.sun.com/javase/downloads/`
[2] `http://java.sun.com/javase/downloads/`
[3] `http://tomcat.apache.org/download-55.cgi`
[4] `http://tomcat.apache.org/tomcat-5.5-doc/index.html`

4. Installation der Anwendung durch Aufruf der Tomcat Management-Seite(`http://<Server>:<port>/manager/html`). Unter der Option „Lokale WAR Datei zur Installation hochladen“ die mitgelieferte WAR-Datei(*sofas.war*, (s. Kap. E)) auswählen. Die Anwendung wird dann in dem Kontext *sofas* installiert und ist durch Anwählen der Adresse `http://<Server>:<port>/sofas/` erreichbar.
5. Für die Konfiguration des Gateways beachten Sie bitte Anhang D.

Installationshinweise für den Agent:

1. Installation der Sun Microsystems JRE v1.6.
2. Installation der Sun Microsystems Java Cryptography Extension. Beachten Sie dabei bitte die Hinweise in dem heruntergeladenen Archiv.
3. Kopieren Sie den Agenten(*SOFAS-Agent_onejar.jar*, (s. Kap. E)) in ein beliebiges Verzeichnis und wechseln Sie in einer Konsole in das entsprechende Verzeichnis.
4. Der Start den Agenten wird wird mit dem Befehl *java -jar SOFAS-Agent_onejar.jar* eingeleitet.
5. Für die Konfiguration des Agenten beachten Sie bitte Anhang D.

Hinweis: Die Installation des Agenten in der konzipierten Form macht nur Sinn, wenn er mit administrativen Rechten, automatisch und dauerhaft ausgeführt wird. Dies können Sie unter Windows-basierten Betriebssystemen durch den Einsatz eines Start-Skriptes und Hilfsprogrammen, wie z. B. *RunAs Professional*[5] erreichen. Eine andere Möglichkeit besteht in der Installation als Dienst. Hierfür existieren Programme, wie z. B. *JavaService*[6] oder *Java Service Wrapper*[7], die allerdings u. U. eine Anpassung des Programm-Codes erforderlich machen.

[5] `http://www.mast-computer.com/`
[6] `http://forge.objectweb.org/projects/javaservice/`
[7] `http://wrapper.tanukisoftware.org/doc/english/introduction.html`

D. Konfigurationshinweise

Sowohl der Agent, als auch der Manager, besitzen zahlreiche Einstellungsmöglichkeiten. Dabei wurden sie so entwickelt, das sich die am häufigsten genutzten Konfigurations-Parameter aus der Anwendung heraus konfigurieren lassen.

Konfiguration des Agenten

Der Agent ist mit einer Initial-Konfiguration versehen, die sich im `resources`-Paket(s. Kap. 6.2.1) befindet. Sie besteht aus den Dateien

1. `SOFASConfiguration.properties`: Hier befinden sich die allgemeinen Einstellungs-Parameter. Sie sind in der Datei durch Kommentare erklärt. Des weiteren enthält sie alle innerhalb der Anwendung auftretenden Text-Blöcke, s. d. sie zur Anpassung der Lokalisierung genutzt werden kann.
2. `WindowsWMIAccess.properties`: Hier befinden sich Optionen für die WMI-Schnittstelle des Inventarisierungs-System. Eine Erklärung der Parameter befindet sich innerhalb der Datei.
3. `SOFASKeystore.ks` und `cert.cer`: Diese Dateien dienen der Funktionalität der implementierten Sicherheits-Funktionen. Bei der Datei *SOFASKeystore.ks* handelt es sich um einen Java-Keystore, d. i. ein Container für öffentlich und private Schlüssel, sowie X.509-Zertifikate[1]. Er enthält den öffentlichen und privaten Schlüssel des Agenten und den öffentlichen des Gateways. Er wird für die Ver- und Entschlüsselung der SOAP-Nachrichten und der Überprüfung der Zertifikate benötigt. Die Datei *cert.cer* enthält das exportierte Zertifikat des Agenten. Sie wird für die Einsatzbereitschaft des Agenten nicht benötigt und wurde an dieser Stelle abgelegt, um ihren einfachen Import in andere Schlüssel-Container zu ermöglichen.

Wenn ein Benutzer den Agenten zum ersten Mal startet, wird die Initial-Konfiguration ausgelesen und, um eine individuelle Konfiguration zu ermöglichen, unter Windows in der Datei

[1] `http://java.sun.com/javase/6/docs/technotes/tools/windows/keytool.html`

`%APPDATA%\SOFAS-Agent\SOFAS-Agent_config.xml` abgelegt. Des weiteren wird die Datei `SOFASKeystore.ks` in das selbe Verzeichnis extrahiert und der Speicherort in der Konfiguration übernommen. In einer Linux-Umgebung werden die Dateien in `~/.SOFAS-Agent/` abgelegt. Bei jedem weiteren Start der Anwendung wird auf die extrahierten Konfigurations-Dateien zugegriffen.

Für eine individuelle Konfiguration, die über die Möglichkeiten der Konfiguration der graphischen Oberfläche hinaus geht, sollte der Agent also einmal gestartet und wieder beendet werden. Die Konfiguration kann dann in der o. g. Datei vorgenommen werden.

Für die Konfiguration des Agenten über die graphische Benutzer-Schnittstelle und zum Beenden des Agenten muss er in den administrativen Modus versetzt werden. Das Initial-Passwort, um in den administrativen Modus des Agenten zu gelangen, lautet *secret*. Für eine Änderung des Initial-Passwortes stellt der Agent einen Mechanismus bereit. Durch Aufruf von `java org.clonewars.sofas.client.common.CryptStore <Passwort>` wird ein neues Kennwort generiert, das in der Konfigurations-Datei(s. o.) hinterlegt werden kann.

Konfiguration des Gateways

Die Konfiguration des Gateway setzt sich aus mehreren Dateien zusammen. Sie besitzt, ähnlich wie der Agent, eine Initial-Konfiguration, die das Gateway beim ersten Start in einen konsistenten, aber nicht unbedingt funktionsfähigen Zustand versetzt.

Die Konfigurations-Dateien setzen sich zusammen aus folgenden Dateien, die sich in dem `resources`-Paket(s. Kap. 6.3.1) befinden:

1. `SOFASConfiguration.properties`: Hier befinden sich die allgemeinen Einstellungs-Parameter. Sie sind in der Datei durch Kommentare erklärt.
2. `JSPMessages.properties`: Diese Datei enthält die in den JSP-Dateien benötigten Textblöcke. Sie dient der Lokalisierung und der individuellen Anpassung der angezeigten Texte.
3. `BeanMessages.properties`: Diese Datei enthält die in den Java-Dateien benötigten Textblöcke. Sie dient ebenfalls der Lokalisierung und der individuellen Anpassung der angezeigten Texte.
4. `SOFASKeystore.ks` und `cert.cer`: Diese Dateien werden für die implementierten Sicherheits-Funktionen benötigt. Bei der Datei *SOFASKeystore.ks* handelt es sich um einen Java-Keystore, d. i. ein Container für öffentlich und private Schlüssel, sowie X.509-Zertifikate[2]. Er enthält den öffentlichen und privaten Schlüssel des Gateways

[2] `http://java.sun.com/javase/6/docs/technotes/tools/windows/keytool.html`

und den öffentlichen des Agenten. Er wird für die Ver- und Entschlüsselung der SOAP-Nachrichten und der Überprüfung der Zertifikate benötigt. Die Datei *cert.cer* enthält das exportierte Zertifikat des Gateways. Sie wird für die Einsatzbereitschaft des Gateways nicht benötigt und wurde an dieser Stelle abgelegt, um ihren einfachen Import in andere Schlüssel-Container zu ermöglichen.

Des weiteren existieren noch verschiedene andere Dateien, die der Konfiguration der eingesetzten Technologien(s. Kap. 6.1.1) dienen:

1. `/WEB-INF/faces-config.xml`: Diese Datei dient der Konfiguration von *MyFaces*. Hier lassen sich Einstellungen zum Lebenszyklus der Java-Beans machen und der Navigations-Verlauf der Anwendung steuern.
2. `/WEB-INF/server-config.wsdd`: Diese Datei konfiguriert das *AxisServlet*, d. h. die verwendete SOAP-Implementation.
3. `/WEB-INF/web.xml`: Diese Datei konfiguriert den Anwendungs-Kontext in *Tomcat*.
4. `/WEB-INF/SOFAS.xml`: s. u.

Um nach dem ersten Start der Applikation Zugriff zu erlangen, müssen in der Datei `/WEB-INF/web.xml` die Sicherheits-Einstellungen(Listing B.2) an die gegebenen Authentifizierungs-Mechanismen angepasst werden(vgl. Apache Software Foundation (2006b)).

Beim ersten Start der Anwendung werden die benötigten Konfigurations-Parameter aus der Datei `SOFASConfiguration.properties` ausgelesen. Sie stehen dann zur Verfügung, um in der Konfigurations-Ansicht(s. Kap. A.15) innerhalb der Application angepasst zu werden. Beim Beenden der Applikation werden die Konfigurations-Parameter in der Datei `/WEB-INF/SOFAS.xml` persistent gespeichert, wo sie beim nächsten Start wieder ausgelesen werden.

Falls die implementierte Schnittstelle zur Speicherung der Management-Daten auf einem OpenLDAP-Server genutzt werden soll, ist das dortige Bekanntmachen des erstellten Schemas(s. Kap. B.1) nötig. Beim ersten Aufruf werden dort die, in Abbildung B.1 dargestellten, Verzeichnisse zur Ablage der Managemen-Daten erstellt. Für den Testbetrieb wurde der Test-Treiber `org.clonewars.sofas.io.xml.XMLObjectBrowser` entwickelt, der die Management-Daten in einer XML-Datei ablegt. Vor der Benutzung des Test-Treibers ist ein Blick in dessen Dokumentation anzuraten.

Von der Applikation aufgerufene Applets, z. B. für den Zugang zu der Fernsteuerung(Abbildung A.8), wurden mit dem, in der Datei *SOFASKeystore.ks* enthaltenen, Zertifikat signiert. Das Zertifikat, bzw. die Signatur, ist für ein halbes Jahr gültig und muss dementsprechend nach Ablauf erneuert werden.

Hinweis: Da es sich bei dem Prototypen um eine Entwicklungs-Version handelt, enthält er einige Debug-Funktionen. Sowohl der Agent, als auch das Gateway sind mit Funktionen zum aufzeichnen von Log-Informationen ausgestattet. Hierfür liegen jeweils in der Wurzel des Paket-Baumes zwei Konfigurations-Dateien(`log4j.properties` und commons-logging.properties). Das Gateway ist so konfiguriert, das sich Axis aus der Ferne administrieren lässt. Dafür wurde in der Datei `/WEB-INF/server-config.wsdd` die Zeile

```
<parameter name="enableRemoteAdmin" value="true"/>
```

eingefügt. Des weiteren wurde die Axis-Konfigurationsseite und der SOAP-Monitor, ein Applet zur Visualisierung der gesendeten und empfangenen SOAP Nachrichten, in die Anwendung integriert(Abbildung A.16).

E. Inhalt der beigefügten CD

Zum Umfang dieser Arbeit gehört eine CD-ROM. Die Verzeichnisse enthalten folgende Bestandteile:

- *Arbeit*:
  - *thesis-view.pdf*: Dieser Text in einer Version, die sich zur Bildschirm-Ansicht eignet, d. h. die Graphiken sind mit 96dpi gerastert und die Farbe der Verweise ist blau.
  - *thesis-print.pdf*: Dieser Text in einer Version, die sich zum Ausdruck eignet, d. h. die Graphiken sind mit 300dpi gerastert und die Farbe der Verweise ist schwarz.
- *Code*:
  - *agent*: Der Quell-Code incl. Bibliotheken(`SOFAS-Agent_1.0.zip`), der compilierte Code ohne Bibliotheken(`SOFAS-Agent.jar`), sowie eine direkt ausführbare Version(`SOFAS-Agent_onejar.jar`) der Agenten-Komponente des Prototypen
  - *gateway*: Der Quell-Code(`SOFAS-Server_1.0.zip`), sowie eine auf dem Tomcat-Server ausführbare Version(`sofas.war`) des Gateways

Literaturverzeichnis

[Apache Software Foundation 2006a] APACHE SOFTWARE FOUNDATION: *The Apache Tomcat 5.5 Servlet/JSP Container - JNDI Resources HOW-TO*. 2006. – URL http://tomcat.apache.org/tomcat-5.5-doc/jndi-resources-howto.html. – Zugriffsdatum: 10.01.2008

[Apache Software Foundation 2006b] APACHE SOFTWARE FOUNDATION: *The Apache Tomcat 5.5 Servlet/JSP Container - Realm Configuration HOW-TO*. 2006. – URL http://tomcat.apache.org/tomcat-5.5-doc/realm-howto.html. – Zugriffsdatum: 10.01.2008

[Apache Software Foundation 2006c] APACHE SOFTWARE FOUNDATION: *The Apache Tomcat 5.5 Servlet/JSP Container - SSL Configuration HOW-TO*. 2006. – URL http://tomcat.apache.org/tomcat-5.5-doc/ssl-howto.html. – Zugriffsdatum: 10.01.2008

[attachmate 2007] ATTACHMATE: *Making PC Lifecycle Management Work for You - A Four Phase Approach*. 2007. – URL http://www.attachmate.com/NR/rdonlyres/C866F7E7-F96C-4360-B96D-693C974A08A7/0/070014_pclm_WP.pdf. – Zugriffsdatum: 20.12.2007

[Baron et al. 2002] BARON, Anthony ; CLARKE, Brett ; HERTROYS, Paul ; OOSTEROM, Norbert van et al.: *Best Practice: ITIL - The Key To Managing IT Services*. TSO for Office of Government Commerce (OGC), 2002. – URL http://www.ogc.gov.uk/guidance_itil.asp. – CD-ROM - Vers. 2.1

[Beyer et al. 2004] BEYER, Thomas ; FROTSCHER, Thilo ; TEUFEL, Marc ; WANG, Dapeng (Hrsg.): *Java Web Services mit Apache Axis*. Software & Support Verlag GmbH, 2004. – ISBN 3-935042-57-4

[Gamma et al. 2004] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*. Addison Wesley, 2004. – ISBN 3-8273-2199-9

[Hallam-Baker et al. 2004] HALLAM-BAKER, Phillip ; KALER, Chris ; MONZILLO, Ronald ; NADALIN, Anthony ; OASIS (Hrsg.): *Web Services Security: X.509 Certificate Token Profile*. 2004. – URL http://docs.oasis-open.org/wss/2004/

01/oasis-200401-wss-x509-token-profile-1.0.pdf. – Zugriffsdatum: 03.01.2008

[Halloway 2001] HALLOWAY, Stuart: *Jawin, An Open Source Interoperability Solution*. 2001. – URL http://www.onjava.com/pub/a/onjava/2001/11/14/jawin.html. – Zugriffsdatum: 10.01.2008

[Hegering et al. 1999] HEGERING, Heinz-Gerd ; ABECK, Sebastian ; NEUMAIR, Bernhard: *Integriertes Management vernetzter Systeme: Konzepte, Architekturen und deren betrieblicher Einsatz*. dpunkt Verlag, 1999. – ISBN 3-932588-16-9

[Heitlinger 1995] HEITLINGER, Paulo: *Netzwerk-Management: Komplettlösungen und Tools*. International Thomson Publishing, 1995. – ISBN 3-8266-0117-3

[Heuer und Saake 2000] HEUER, Andreas ; SAAKE, Gunter: *Datenbanken - Konzepte und Sprachen*. mitp-Verlag, 2000. – ISBN 3-8266-0619-1

[Kaczmarek 2004] KACZMAREK, Steven D.: *Microsoft System Management Server 2003*. Microsoft Press, 2004. – ISBN 0-7356-1888-7

[Larisch 2003] LARISCH, Dirk: *Windows Server 2003: Active Directory Services*. Galileo Press, 2003. – ISBN 3-89842-348-4

[Nadalin et al. 2004a] NADALIN, Anthony ; GRIFFIN, Phil ; KALER, Chris ; HALLAM-BAKER, Phillip ; MONZILLO, Ronald ; OASIS (Hrsg.): *Web Services Security: Username Token Profile 1.0*. 2004. – URL http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf. – Zugriffsdatum: 03.01.2008

[Nadalin et al. 2004b] NADALIN, Anthony ; KALER, Chris ; HALLAM-BAKER, Phillip ; MONZILLO, Ronald ; OASIS (Hrsg.): *Web Services Security: SOAP Message Security 1.04 (WS-Security 2004)*. 2004. – URL http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf. – Zugriffsdatum: 03.01.2008

[Radtke 2004] RADTKE, Taito: *Synchronisation von Verzeichnisdiensten*, Fachhochschule für Technik und Wirtschaft Berlin, Diplomarbeit, 2004

[Real-World Labs 2007] REAL-WORLD LABS: Die Rechner an die Leine nehmen. In: *Network Computing* (2007), Oktober, Nr. 17, S. 16–17

[Silver 2005] SILVER, Michael A. ; GARTNER INC. (Hrsg.): *Saving Money on PC Deployment*. 2005. – URL http://www.gartner.com/DisplayDocument?doc_cd=135813. – Zugriffsdatum: 20.12.2007

[Sun Microsystems 2005] SUN MICROSYSTEMS: *JDK Core Engineering Team: Core Java Technology Features*. 2005. – URL `http://java.sun.com/developer/technicalArticles/J2SE/Desktop/JavaSE6_build39.html`. – Zugriffsdatum: 02.01.2008

[Sun Microsystems 2007] SUN MICROSYSTEMS: *JNDI as an LDAP API*. 2007. – URL `http://java.sun.com/docs/books/tutorial/jndi/ldap/jndi.html`. – Zugriffsdatum: 10.01.2008

[Terplan 1995] TERPLAN, Kornel: *Client/Server-Management*. DATACOM Buchverlag GmbH, 1995. – ISBN 3-89238-124-0

[Zörner 2005] ZÖRNER, Stefan: *LDAP für Java-Entwickler: Eine praxisorientierte Einführung*. Software & Support Verlag, 2005. – ISBN 3-935024-72-8

Verzeichnis der RFCs und Standards

[DSP0004] DSP0004: *Desktop Management Task Force: Common Information Model(CIM) Infrastructure Specification (Version 2.3, Final)*. – URL http://www.dmtf.org/standards/published_documents/DSP0004V2.3_final.pdf. – Zugriffsdatum: 28.12.2007

[DSP0200] DSP0200: *Desktop Management Task Force: Operations over HTTP (Version 1.2, Final)*. – URL http://www.dmtf.org/standards/published_documents/DSP200.html. – Zugriffsdatum: 28.12.2007

[DSP0201] DSP0201: *Desktop Management Task Force: Representation of CIM in XML (Version 2.2, Final)*. – URL http://www.dmtf.org/standards/published_documents/DSP201.html. – Zugriffsdatum: 28.12.2007

[FIPS180-2] FIPS180-2: *Federal Information Processing Standard (FIPS) 180-2: Secure Hash Signature Standard*. – URL http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf. – Zugriffsdatum: 03.01.2008

[ISO/IEC 7498-1] ISO/IEC 7498-1: *Information Technology - Open System Interconnection - Basic Reference Model: The Basic Model*. – URL http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip. – Zugriffsdatum: 26.12.2007

[ISO/IEC 8824-1] ISO/IEC 8824-1: *ITU-T Rec. X.680: Abstract Syntax Notation One (ASN.1) - Specification of Basic Notation*. – URL http://www.itu.int/ITU-T/e-business/mou/MoUMG-standards.html. – Zugriffsdatum: 26.12.2007

[RFC1155] RFC1155: *M.T. Rose and K. McCloghrie: Structure and identification of management information for TCP/IP-based internets*. RFC 1155 (Standard). – URL http://www.ietf.org/rfc/rfc1155.txt

[RFC1157] RFC1157: *J.D. Case et al.: Simple Network Management Protocol (SNMP)*. RFC 1157 (Historic). – URL http://www.ietf.org/rfc/rfc1157.txt

[RFC1213] RFC1213: *K. McCloghrie and M. Rose: Management Information Base for Network Management of TCP/IP-based internets:MIB-II*. RFC 1213 (Standard) (Request for Comments). – URL `http://www.ietf.org/rfc/rfc1213.txt`. – Updated by RFCs 2011, 2012, 2013

[RFC1321] RFC1321: *R. Rivest: The MD5 Message-Digest Algorithm*. RFC 1321 (Informational). – URL `http://www.ietf.org/rfc/rfc1321.txt`

[RFC1441] RFC1441: *J. Case et al.: Introduction to version 2 of the Internet-standard Network Management Framework*. RFC 1441 (Historic). – URL `http://www.ietf.org/rfc/rfc1441.txt`

[RFC1445] RFC1445: *J. Galvin and K. McCloghrie: Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2)*. RFC 1445 (Historic). – URL `http://www.ietf.org/rfc/rfc1445.txt`

[RFC1447] RFC1447: *K. McCloghrie and J. Galvin: Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2)*. RFC 1447 (Historic). – URL `http://www.ietf.org/rfc/rfc1447.txt`

[RFC1452] RFC1452: *J. Case et al.: Coexistence between version 1 and version 2 of the Internet-standard Network Management Framework*. RFC 1452 (Proposed Standard) (Request for Comments). – URL `http://www.ietf.org/rfc/rfc1452.txt`. – Obsoleted by RFC 1908

[RFC1902] RFC1902: *J. Case et al.: Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)*. RFC 1902 (Draft Standard) (Request for Comments). – URL `http://www.ietf.org/rfc/rfc1902.txt`. – Obsoleted by RFC 2578

[RFC1908] RFC1908: *J. Case et al.: Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework*. RFC 1908 (Draft Standard) (Request for Comments). – URL `http://www.ietf.org/rfc/rfc1908.txt`. – Obsoleted by RFC 2576

[RFC2307] RFC2307: *L. Howard: An Approach for Using LDAP as a Network Information Service*. RFC 2307 (Experimental). – URL `http://www.ietf.org/rfc/rfc2307.txt`

[RFC2713] RFC2713: *V. Ryan and S. Seligman and R. Lee: Schema for Representing Java(tm) Objects in an LDAP Directory*. RFC 2713 (Informational). – URL `http://www.ietf.org/rfc/rfc2713.txt`

[RFC2790] RFC2790: *S. Waldbusser and P. Grillo: Host Resources MIB*. RFC 2790 (Draft Standard). – URL `http://www.ietf.org/rfc/rfc2790.txt`

[RFC2819] RFC2819: *S. Waldbusser: Remote Network Monitoring Management Information Base*. RFC 2819 (Standard). – URL `http://www.ietf.org/rfc/rfc2819.txt`

[RFC3410] RFC3410: *J. Case et al.: Introduction and Applicability Statements for Internet-Standard Management Framework*. RFC 3410 (Informational). – URL `http://www.ietf.org/rfc/rfc3410.txt`

[RFC3411] RFC3411: *D. Harrington et al.: An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks*. RFC 3411 (Standard). – URL `http://www.ietf.org/rfc/rfc3411.txt`

[RFC3418] RFC3418: *R. Presuhn: Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)*. RFC 3418 (Standard). – URL `http://www.ietf.org/rfc/rfc3418.txt`

[RFC3805] RFC3805: *R. Bergman et al.: Printer MIB v2*. RFC 3805 (Proposed Standard). – URL `http://www.ietf.org/rfc/rfc3805.txt`

[RFC3986] RFC3986: *T. Berners-Lee and R. Fielding and L. Masinter: Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986 (Standard). – URL `http://www.ietf.org/rfc/rfc3986.txt`

[RFC4502] RFC4502: *S. Waldbusser: Remote Network Monitoring Management Information Base Version 2*. RFC 4502 (Draft Standard). – URL `http://www.ietf.org/rfc/rfc4502.txt`

[RFC4510] RFC4510: *K. Zeilenga: Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map*. RFC 4510 (Proposed Standard). – URL `http://www.ietf.org/rfc/rfc4510.txt`

[RFC4512] RFC4512: *K. Zeilenga: Lightweight Directory Access Protocol (LDAP): Directory Information Models*. RFC 4512 (Proposed Standard). – URL `http://www.ietf.org/rfc/rfc4512.txt`

[RFC4513] RFC4513: *R. Harrison: Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms*. RFC 4513 (Proposed Standard). – URL `http://www.ietf.org/rfc/rfc4513.txt`

[RFC4517] RFC4517: *S. Legg: Lightweight Directory Access Protocol (LDAP): Syntaxes and Matching Rules*. RFC 4517 (Proposed Standard). – URL `http://www.ietf.org/rfc/rfc4517.txt`

[RFC4519] RFC4519: *A. Sciberras: Lightweight Directory Access Protocol (LDAP): Schema for User Applications*. RFC 4519 (Proposed Standard). – URL `http://www.ietf.org/rfc/rfc4519.txt`

[RFC4524] RFC4524: *K. Zeilenga: COSINE LDAP/X.500 Schema*. RFC 4524 (Proposed Standard). – URL `http://www.ietf.org/rfc/rfc4524.txt`

[RFC4533] RFC4533: *K. Zeilenga and J.H. Choi: The Lightweight Directory Access Protocol (LDAP) Content Synchronization Operation*. RFC 4533 (Experimental). – URL `http://www.ietf.org/rfc/rfc4533.txt`

[RFC768] RFC768: *J. Postel: User Datagram Protocol*. RFC 768 (Standard). – URL `http://www.ietf.org/rfc/rfc768.txt`

Glossar

.NET Eine von Microsoft entwickelte Softwareplattform

Active Directory Das Active Directory ist der Verzeichnis-Dienst der Windows Server-Landschaft(s. a. LDAP)

administrative/operative Tätigkeit Im Kontext der Systemadministration ist eine operative Tätigkeit, eine kurzfristige Tätigkeit am System, die ohne grösseren Planungs- oder Testaufwand erfolgt. Eine administrative Tätigkeit hingegen durchläuft alle vorgeschriebenen Phasen(d. h. Planung, Realisierung, Test) bis zur letztendlichen Implementation.

ADS (Active Directory Services): s. Active Directory

ASN.1 (Abstract Syntax Notation One): Eine standardisierte Beschreibungssprache für Datenstrukturen

Axis (Apache eXtensible Interaction System): SOAP Implementation der Apache Software Foundation.

CIM (Common Information Model): Das standardisierte Informationsmodell des WBEM

CIMOM (CIM Object Manager): Ist in der WBEM Architektur ein Vermittler für Anfragen. Er leitet sie an einen passenden Provider weiter.

CMIP/S (Common Management Information Protocol/Service): Ein auf dem OSI-ISO Protokoll-Modell aufbauendes Management-Protokoll. Haupteinsatzgebiet sind Telekommunikations-Netzwerke.

COM (Component Object Model): Eine, von Microsoft entwickelte, proprietäre Technologie zur Interprozess-Kommunikation

CORBA (Common Object Request Broker Architecture): Eine Spezifikation für eine Plattform-übergreifende, objektorientierte Middleware

DLL (Dynamic Link Libraries): Eine Programm-Bibliothek, die Fubktionen und Ressourcen zur Verfügung stellen kann

DMI (Desktop Management Interface): Ein standardisiertes Framework zur Erfassung von Inventar-Informationen

DMTF Distributed Management Task Force

DSL (Digital Subscriber Line): Breitband-Internetzugangs Technik

HTML (Hypertext Markup Language): Beschreibungssprache für Web-Seiten

HTTP (Hypertext Transfer Protocol): Ein zustandsloses Protokoll zur Übertagung von Daten

IAB Internet Activities Board

IETF Internet Engineering Taskforce

ISO International Standards Organization

ITIL Die IT Infrastructure Library ist ein Framework für die unternehmensweite Implementierung des IT Service-Managements. Es setzt sich zusammen aus Best-Practices die in Studien in privaten und staatlichen Organisationen erstellt wurden. Die ITIL wird unter Zusammenarbeit mit verschiedenen Großunternehmen durch das englische Office of Government Commerce(OGC) beständig weiterentwickelt.

JDBC (Java Database Connectivity): Eine Datenbankschnittstelle für Java

JNDI (Java Naming and Directory Interface): Eine Java-Schnittstelle zu Namens- und Verzeichnis-Diensten

JNI (Java Native Interface): Eine Schnittstelle zum Aufruf Plattform-spezifischer Bibliotheken aus der Java Laufzeit-Umgebung

JRE (Java Runtime Environment): Eine Umgebung, um Java-Programme auszuführen

JSF (Java Server Faces): Ein Framework zur Erstellung von Anzeige-Objekten in Web-Anwendungen auf Basis von Servlets und JSPs

JSP (Java Server Pages): Technik zur Einbettung von Java-Code in HTML-Seiten. Ermöglicht den dynamischen Aufbau der Ansicht-Elemente

LDAP Lightweight Directory Access Protocol. Es beschreibt die Abfrage und Modifikation von Informationen, die durch einen Verzeichnis-Dienst zur Verfügung gestellt werden. Die Daten sind dabei über Schemata, die eine Relation zwischen Objekt-Klassen und ihren Attributen definieren, streng geordnet. Die z. Zt. verwendete Protokoll-Version ist LDAP v3, sie ist definiert in RFC4510 bis RFC4524.

Management-Komponente Eine abgegrenzter Teil eines Management-Systems mit dezidierter Funktion.

Management-Object Eine verwaltete Ressource. Dies kann z. B. ein Computer oder dessen Grafikkarte sein, aber auch abstrakte Gebilde, wie ein Benutzer oder eine Richtlinie

Management-System Ein System, das die Verwaltung von Management-Objekten ermöglicht.

MD5 (Message Digest No. 5): Eine kryptologische Hash-Funktion(s. RFC1321)

MIB (Management Information Base): Syntaktische und semantische Beschreibung der Informationen, die zu einem Management-Objekt zur Verfügung gestellt werden können.

MO (Management-Objekt): s. Management-Objekt

MVC (Model-View-Controller): Das MVC-Muster ist eine Technik, um eine logische Trennung zwischen den Daten(Model), deren Anzeige(View) und der dahinterstehenden Logik(Control) zu erreichen.

MyFaces Apache MyFaces ist eine JSF-Implementation der Apache Software Foundation.

NDS (Novell Directory Services): Der Verzeichnis-Dienst von Novell Netware. Seit Netware 5.1 unterstützt er Abfragen über das LDAP. Neuere Versionen werden auch als *Novell eDirectory* bezeichnet.

OID (Object Identificator): Eine ASN.1 kodierter eindeutiger Identifikator für ein Objekt.

OMA (Object Management Architecture): Eine Spezifikation für eine Management-Architektur auf Basis von CORBA. Entwickelt von der OMG

OMG Object Management Group

OpenLDAP OpenLDAP ist eine Open-Source-Implementierung des Lightweight Directory Access Protocol(LDAP). Die Software Suite besteht aus einem LDAP-Service zur Veröffentlichung der Verzeichnisses, einem Replikations-Service zur Verteilung des Verzeichnisses im Netzwerk und verschiedenen kleineren Werkzeugen zur Wartung der Verzeichnis-Datenbank.

QoS Quality of Service. QoS beschreibt die Dienstgüte, d. h. es stellt eine Menge an Qualitätsanforderungen aus der Sicht des Dienstbenutzers an den Dienst. Um die geforderte Leistung erfolgreich zu erbringen, ist der Dienst zur Einhaltung der QoS verpflichtet. Ein Beispiel für QoS-Definitionen sind das Einhalten einer minimalen Übertragungsgeschwindigkeit oder Antwortzeit bei IP-Netzwerken.

RMON (Remote Monitoring Standard for SNMP-MIBs): Stellt Überwachungs- und Diagnose-Funktionen auf Basis einer MIB für SNMP zur Verfügung

Samba Samba ist eine Open-Source-Implementation des SMB/CIFS Protokoll und stellt Datei- und Druck-Dienste für Clients, die das SMB/CIFS-Protokoll implementieren(z. B. Microsoft Windows basiert), auf Unix- und Linux-Servern zur Verfügung. Es ermöglicht die Interoperabilität von Microsoft Windows und Unix/Linux in einem heterogenen Netzwerk.

Servlet Java-Klassen, die dynamisch HTML-Seiten generieren

SHA (Secure Hash Algorithm): Ein Gruppe standardisierter kryptologischer Hash-Funktionen(s. FIPS180-2)

SLA (Service Level Agreement): Ein Vertrag über Leistungseigenschaften, z. B. zugesicherte Reaktionszeiten

SMI (Structure of Management Information): Eine Regelsammlung für die Syntax der Beschreibung von Management-Objekten

SNMP (Simple Network Management Protocol): Ein Netzwerk-Management-Protokoll, manchmal auch dessen implementierende Internet-Management Architektur

SOAP (Simple Object Access Protocol): Ein Protokoll zur Übertragung von Daten oder Senden von Nachrichten an entfernte Objekte

Software-Paket Eine Anwendung, die in einer abstrahierten Form vorliegt und mit verschiedenen Meta-Daten verknüpft ist, s. d. sie durch einem geeigneten Interpreter der Meta-Daten auf einem System in einen ablauffähig konfigurierten Zustand gebracht werden kann.

SPOC (Single Point of Contact): Ein zentraler Ansprechpartner In der Terminologie des Helpdesk- und Problem-Managements

SSL (Secure Socket Layer): Ein transparentes Verschlüsselungs- und Transport-Protokoll

Tomcat (Apache Tomcat): Ein auf Java basierender Applications-Server.

UDP (User Datagram Protocol): Ein verbindungsloses Transport-Protokoll(nach OSI)

UMTS (Universal Mobile Telecommunications System): Ein Mobilfunkstandard der dritten Generation

URI (Uniform Ressource Identificator): Eine Zeichenfolge zur Identifikation einer Ressource(s. RFC3986

VPN (Virtual Private Network): Ein logisch abgetrenntes Netzwerk innerhalb eines öffentlichen Netzwerkes

WBEM (Web Based Enterprise Management): Eine von der DMTF standardisierte Management-Architektur

WMI (Windows Management Instrumentation): Microsofts Implementation des WBEM

WSDL Web-Service Description Language

WSS4J (Web-Service Security for Java): Implementation des WS-Security- und Token-Profils in Java

WSUS (Windows Server Update Services): Patch-Verteilungs-System für Microsoft Produkte

XML (Extensible Markup Language): Eine Beschreibungssprache für hierarchisch strukturierte Daten

Abbildungsverzeichnis

2.1. Schichtung des Managements(Hegering et al. (1999), abgeändert) 11
2.2. Application-Management(aus Baron et al. (2002)) 15
2.3. PC-Lifecycle-Management 16
2.4. Rollen des Management-System 17
2.5. Domänen-Bildung im Management-System 17

3.1. Operative Software-Konfiguration 23
3.2. Operative Hardware-Konfiguration 24

4.1. Organisationsmodell der Internet-Management Architektur 34
4.2. Internet-Registrierungsbaum für SNMP-SMI 36
4.3. SNMP im OSI-Referenzmodell 38
4.4. Struktur des CIM Meta-Schemas(aus DSP0004) 40

5.1. Organsiationsmodell des Prototypen 46
5.2. Komponenten des Agenten 48
5.3. Komponenten des Gateways 50
5.4. Modell der Benutzer-Verwaltung 54
5.5. Modell der Software- und Asset-Verwaltung 55

6.1. SOAP im OSI-Referenzmodell 61
6.2. SOAP-Nachrichtenformat 61
6.3. Nachrichtenverarbeitung der Axis Engine 62
6.4. Klassen-Diagramm des Agenten 65
6.5. Klassen-Diagramm des Gateways zur Interaktion mit den Agenten 76
6.6. Klassen-Diagramm des Gateways zur Datenhaltungs-Komponente 78

A.1. Agent: Status-Anzeige im System-Tray 84
A.2. Agent: Tray-Menü und Anmelde-Dialog 84
A.3. Agent: Konfigurations-Dialog 85
A.4. Agent: Informations-Dialog 85
A.5. Gateway: Menü-Struktur 86
A.6. Gateway: Berechtigungs-Gruppen 87
A.7. Gateway: Computer- und Benutzer-Anzeige 87

A.8. Gateway: Asset-Anzeige (Übersicht) 88
A.9. Gateway: Asset-Anzeige (Details) 89
A.10.Gateway: Installations-Gruppen 90
A.11.Gateway: Software-Pakete 90
A.12.Gateway: Reports und Suche 91
A.13.Gateway: Beispiel für die Suche 91
A.14.Gateway: Aufbau der Suche 92
A.15.Gateway: Konfiguration 93
A.16.Gateway: Konfiguration(Axis und SOAP-Monitor) 94

B.1. Beispiel für die Anwendung von Listing B.1 97

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung vom 22.11.2001 nach §22(4) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 28.01.2008	
Ort, Datum	Unterschrift