INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Diplomarbeit

Eine erweiterbare Management-Plattform für Hostvirtualisierungslösungen

Florian Bittner

Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Dr. Vitalian Danciu Nils gentschen Felde Tobias Lindinger
Abgabetermin:	14. Oktober 2008

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 14. Oktober 2008

. . . . . .
(Unterschrift des Kandidaten)

Abstract

Virtualisierung ist ein aktuelles Thema in Wissenschaft und Industrie. Aufgrund der Leistungssteigerungen der letzten Jahre u.a. bei Prozessoren, Hauptspeicher und Festspeicher, haben viele Rechnersysteme einen Leistungsüberschuss zu verzeichnen. Hostvirtualisierung nutzt diesen Überschuss und teilt die Ressourcen auf mehrere virtuelle Systeme auf. Diese Aufteilung setzt jedoch ein effizientes Management voraus, da es ansonsten u.U. zu Leistungsverlusten kommen kann, falls eine Ressource stark frequentiert wird.

Diese Arbeit zeigt die aktuell bestehenden Probleme beim Management virtueller Umgebungen am Beispiel der Lehr- und Forschungseinheit für Kommunikationssysteme und Systemprogrammierung der Universität München auf. Anhand dieser Defizite werden Anforderungen an eine Managementplattform aufgestellt um ein effizientes Management von Virtualisierungslösungen zu ermöglichen.

Diese Anforderungen werden genutzt um eine neue Managementplattform zu entwerfen. Für die Konzeption dieser Managementplattform wird auf die Vorgehensweisen zum Entwurf von generellen Managementsystemen zurückgegriffen. Vor allem die Managementteilmodelle, das Informationsmodell, das Organisationsmodell, das Kommunikationsmodell und das Funktionsmodell, sind ein zentraler Bestandteil für den Entwurf der Managementplattform.

Die Praxistauglichkeit der entworfenen Managementplattform wird schließlich anhand deren Implementierung aufgezeigt.

Inhaltsverzeichnis

1 Einleitung

Die Idee der Virtualisierung ist in der Informatik nicht neu sondern wird schon seit Jahrzehnten eingesetzt. Anfangs konnte man Virtualisierung nur in Rechenzentren nutzen, da nur hier die entsprechenden Kapazitäten verfügbar waren. Mittlerweile ist Virtualisierung jedoch auf jedem x86-System einsetzbar und dadurch ein aktueller Trend in Wissenschaft und Industrie. Möglich wurde diese Entwicklung durch das schnelle voranschreiten des IT-Sektors in den letzten Jahren. Die verfügbaren CPU-Leistungen, Speichergrößen und Datenübertragungsraten haben sich vervielfacht. Aus diesem Grund haben aktuelle Rechnersysteme zumeist einen Überschuss an Ressourcen zu verzeichnen. Will man nun diese überschüssigen Ressourcen effizient nutzen liegt es nahe mehrere Betriebssysteme anstatt nur einem pro System zu nutzen. Hier kommt nun die Virtualisierung ins Spiel. Es werden auf einem physischen System mehrere virtuelle Systeme etabliert, wobei sich jedes als eigenständiges System darstellt. Somit ist es möglich mehrere und auch verschiedene Betriebssysteme auf einem einzigen Rechnersystem zeitgleich zu nutzen. Die Vorteile der Virtualisierung sind u.a.:

- hohe Ressourcenausnutzung
- Isolation von Diensten
- gute Wirtschaftlichkeit, da weniger physische Maschinen gebraucht werden.

Demgegenüber steht jedoch der Nachteil, dass sich evtl. Geschwindigkeitsnachteile ergeben, da nun die vorhandenen Ressourcen unter mehreren (virtuellen) Systemen aufgeteilt werden müssen. Somit ist es unerlässlich die virtuellen Systeme effizient managen zu können. Hierzu werden auf diese Aufgaben zugeschnittene Managementlösungen benötigt. Die am Markt erhältlichen freien und proprietären Virtualisierungslösungen, Beispiele dafür sind XEN und VMware, bringen dafür ihre eigenen Managementtools mit.

1.1 Motivation und Ziele

In größeren Umgebungen sind zumeist verschiedene Virtualisierungslösungen im Einsatz, da jede ihre speziellen Vor- und Nachteile besitzt. So kann für jedes Szenario eine bestmögliche virtuelle Infrastruktur geschaffen werden. Diese Heterogenität stellt jedoch neue Herausforderungen an das Management dieser Lösungen. Vorhandene herkömmliche Managementtools sind nicht auf die Probleme der virtualisierten Umgebungen vorbereitet. Auf der anderen Seite sind die Managementtools der einzelnen Virtualisierungslösungen isoliert voneinander und in den wenigsten Fällen interoperabel. Es existieren zwar diverse übergreifende Managementlösungen für virtualisierte Umgebungen, diese sind jedoch auf bestimmte Szenarien (z.B. Webhosting) zugeschnitten und unterstützen nur eine geringe Zahl an Virtualisierungslösungen und Funktionalitäten. Ein weiteres Problem ist die teilweise stark eingeschränkte Möglichkeit diese Managementlösungen in die vorhandene IT-Infrastruktur einzu-

binden. Fehlt z.B. eine Anbindung an den eingesetzten Verzeichnisdienst, ist ein Single-Sign-On nicht möglich. Setzt man heutzutage Virtualisierung in größeren Umgebungen ein, hat man folglich vor allem mit der Interoperabilität der vorhandenen Managementlösungen und dem daraus resultierenden höheren Verwaltungsaufwand zu kämpfen.

Auch für die am Lehrstuhl eingesetzten Virtualisierungslösungen, v.a. XEN und VMware, existieren keine Managementlösungen, die den Anforderungen des Lehrstuhls genügen oder auf diese effizient zugeschnitten werden können. Die genannten Unzulänglichkeiten der Managementlösungen kommen auch hier zum tragen.

Ziel dieser Arbeit ist es, eine Managementplattform zu konzipieren, welche das genannte Problem der fehlenden Interoperabilität, und somit des hohen Verwaltungsaufwandes, löst. Dazu muss eine Erweiterbarkeit, v.a. in Hinsicht auf die Unterstützung von beliebigen Virtualisierungslösungen und die Anbindung verschiedener Backendkomponenten, z.B. Verzeichnisdienste, gegeben sein. Zusätzlich zur Konzeption soll eine Implementierung der Managementplattform erfolgen. Diese soll jedoch aufgrund des Umfangs einer solchen nur rudimentär erfolgen und sich auf die Virtualisierungslösungen Xen und VMware beschränken. Diese sollen über ein Webinterface gemanagt werden können. Die Authentifizierung der Benutzer soll gegenüber dem Verzeichnisdienst OpenLDAP erfolgen.

1.2 Vorgehen und Gliederung

Den Einstieg in diese Arbeit übernimmt das Kapitel Anforderungen. Hier wird anhand der Lehr- und Forschungseinheit für Kommunikationssysteme und Systemprogrammierung der Universität München, exemplarisch gezeigt, wie die aktuelle Managementsituation beim Einsatz virtueller Umgebungen aussieht. Dazu werden zuerst die eingesetzten virtuellen Umgebungen und die dazugehörige IT-Infrastruktur dargestellt. Anschließend werden die dafür genutzten Managementtools kurz zusammengefaßt. Anhand der daraus resultierenden Problematiken werden die Anforderungen an ein Managementsystem zusammengestellt. Diese bilden eine Grundlage für die in dieser Arbeit zu entwerfende Managementplattform.

Das nächste Kapitel, 'State of the Art', gibt einen kurzen Einblick in einige am Markt erhältliche Managementtools für Hostvirtualisierungslösungen. Diese werden mit den Anforderungen des Lehrstuhls verglichen und die dabei aufgetretenen Defizite erläutert.

Den größten Teil dieser Arbeit nimmt die Analyse und Modellierung der Managementplattform ein. Anhand der etablierten Management-Teilmodelle (Informations-, Organisations-, Kommunikations- und Funktionsmodell) werden die benötigten Bausteine für die Managementplattform erarbeitet und beschrieben. Dazu gehört u.a. eine Management Information Base (MIB), die zu unterstützende Topologie, benötigte Kommunikationsbeziehungen und eine kurze Beschreibung der Funktionsbereiche nach FCAPS, den Teilbereichen des Funktionsmodells.

Schließlich folgt der Entwurf und die prototypische Implementierung der entworfenen Managementplattform. Diese erfolgt allerdings nur rudimentär, da der gesamte Umfang den Rahmen dieser Arbeit sprengen würde.

Den letzten Teil dieser Arbeit bildet eine Schlussbetrachtung, welche die erreichten Funktionalitäten aufzeigt sowie mögliche Erweiterungen des Systems erläutert. Im Anhang sind schließlich Erfahrungen und Installationsanleitungen zur Implementierung zusammengetragen. Außerdem finden sich dort Tabellen und Beschreibungen, die eine informative Ergänzung zum Hauptteil dieser Arbeit bieten.

2 Anforderungen

Ziel dieses Kapitels ist es die Anforderungen an ein Managementsystem für Hostvirtualisierungslösungen zusammenzustellen. Als Basis dient hierbei die existierende virtuelle Infrastruktur am Lehrstuhl. Diese wird beschrieben und die aktuelle Managementsituation dargestellt. Die dabei auftretenden Defizite werden kurz erläutert und ergeben schließlich einen Anforderungskatalog für die zu entwerfende Managementplattform.

2.1 Virtuelle Umgebungen am Lehrstuhl

Zum Zwecke von Lehre und Forschung werden am Lehrstuhl diverse Serversysteme eingesetzt, u.a. Mailserver, Webserver und Monitoringsysteme. Um die Vorteile der Hostvirtualisierung zu nutzen wurden in den letzten Jahren einige dieser Serversysteme virtualisiert. Dadurch verbesserte sich die Ressourcenauslastung der Rechner und die neu zu beschaffende Hardware konnte zahlenmäßig gering gehalten werden.

Ein Beispiel ist das 2006 im Rahmen einer Diplomarbeit [Lin06] virtualisierte IT-Sicherheitspraktikum für Studenten. Die für das Praktikum genutze Infrastruktur, samt Client-Rechner, wurde im Rahmen der Arbeit virtualisiert, mit dem Ziel, das Praktikum hinsichtlich Teilnehmerzahl flexibler und kostenneutraler zu gestalten, sowie den Administrationsaufwand zu senken. Die teilnehmenden Studenten können seitdem auf „ihren“ Rechner z.B. vom CIP-Pool oder von zu Hause zugreifen. Die virtuelle Umgebung läuft momentan auf einem einzigen Server, dieser stellt auch die einzige zu beschaffende Hardwarekomponente im Rahmen des Praktikums dar.

Am Lehrstuhl werden verschiedene Virtualisierungslösungen eingesetzt um den teilweise stark differenzierten Anforderungen der Teilbereiche von Forschung und Lehre Rechnung zu tragen. Für das genannte Sicherheitspraktikum wird auf die Open-Source Lösung Xen gesetzt. Auch einige weitere Server setzen diese ein. Eine andere virtuelle Umgebung basiert auf Microsoft Virtual Server 2005, diese wird eingesetzt um den Anforderungen diverser Anwendungen aus der Windowswelt gerecht zu werden. Deren Nachfolger, MS Hyper-V, wird zu Testzwecken ebenfalls eingesetzt. In einigen Praktikas werden teilweise VMware Serverprodukte verwendet, z.B. der ESX-Server.

2.2 Aktuelle Situation beim Management der virtuellen Umgebungen

Das Management der virtuellen Umgebungen erfolgt momentan sehr heterogen. Die Virtualisierungslösungen von Microsoft bringen jeweils ihre eigenen Managementtools mit. Für die VMware-Serverprodukte gibt es ebenfalls eine eigene Managementlösung, das VMware Virtual Center. Die Open-Source Lösung Xen bringt Kommandozeilenprogramme zum Management mit, für das IT-Sicherheitspraktikum wurde im Rahmen der genannten Diplomarbeit ein eigenes Managementtool erstellt, das Virtual Machine Management Tool (VMMT).

Es wird somit aktuell am Lehrstuhl für jede Virtualisierungslösung ein eigenes Managementtool eingesetzt und der Verwaltungsaufwand ist entsprechend hoch (siehe Abbildung 2.1). Problematisch ist u.a. dass die einzelnen Tools teilweise stark plattformabhängig sind, z.B. kann das Management von VMware Server-Produkten mittels Virtual Center nur aus einer Windowsumgebung heraus erfolgen. Das bei den VMware Server-Produkten integrierte Webinterface bietet nur sehr rudimentäre Konfigurationsmöglichkeiten. Der im Rahmen des IT-Sicherheitspraktikums eingesetzte VMMT ist zwar über ein Webinterface plattformunabhängig erreichbar, bietet jedoch nur eine minimale Verwaltung der virtuellen Umgebung. Weitergehende Funktionen, z.b. Migration, sind damit nicht möglich. Auch eine Integration in die bestehende IT-Infrastruktur ist nicht bei allen Tools gegeben. Z.B. ist eine Anbindung des eingesetzten Verzeichnisdienstes zur Benutzerauthentifizierung nicht bei jeder Lösung möglich. Am Lehrstuhl wird momentan zur Authentifizierung NIS (Network Information Service, [nis08]) eingesetzt, eine Umstellung auf OpenLDAP [ope08] ist jedoch geplant. Ebenfalls sinnvoll wäre eine Anbindung an vorhandene Monitoringsysteme, dazu müsste eine Schnittstelle „nach außen“ bestehen.

Die aktuellen Probleme beim Management der eingesetzten Virtualisierungslösungen sind zusammengefaßt:

- Kein zentrales Management
- Authentifizierung der Benutzer erfolgt nicht anhand des bestehenden Verzeichnisdienstes
- Managementtools sind teilweise nicht plattformunabhängig bedienbar.
- Nicht jedes Managementtool unterstützt alle Funktionalitäten des entsprechenden Virtualisierers
- Anbindung des Managementtools an externe Anwendungen meist nicht möglich.

2.3 Wünsche an das Management von Virtualisierungslösungen seitens des Lehrstuhls

Die aktuelle Situation beim Management der Virtualisierungslösungen wurde aufgezeigt und die dabei auftretenden Probleme erläutert. Dieser Abschnitt ergänzt die Situationsbeschreibung um die Wünsche an eine zukünftige Managementlösung. Diese orientieren sich an den aufgezeigten Defiziten der aktuellen Lösungen. Eine grafische Darstellung dazu findet sich in Abbildung 2.2.

Zentrales Management aller eingesetzten Virtualisierungslösungen Die verschiedenen eingesetzten Virtualisierungslösungen sollen, anders als bisher, anhand eines Managementtools gemanagt werden können. Dabei ist es unerheblich, ob das Managementtools direkt auf die Ressourcen zugreift oder dazu eine andere Managementlösung, z.B. als Proxy, benötigt. Jedoch sollte zur täglichen Benutzung lediglich ein einziges Tool nötig sein.

Abbildbarkeit der existierenden Organisationsstruktur Die verschiedenen existierenden Benutzergruppen sollen sich in der Managementstruktur abbilden lassen. Dabei sollen

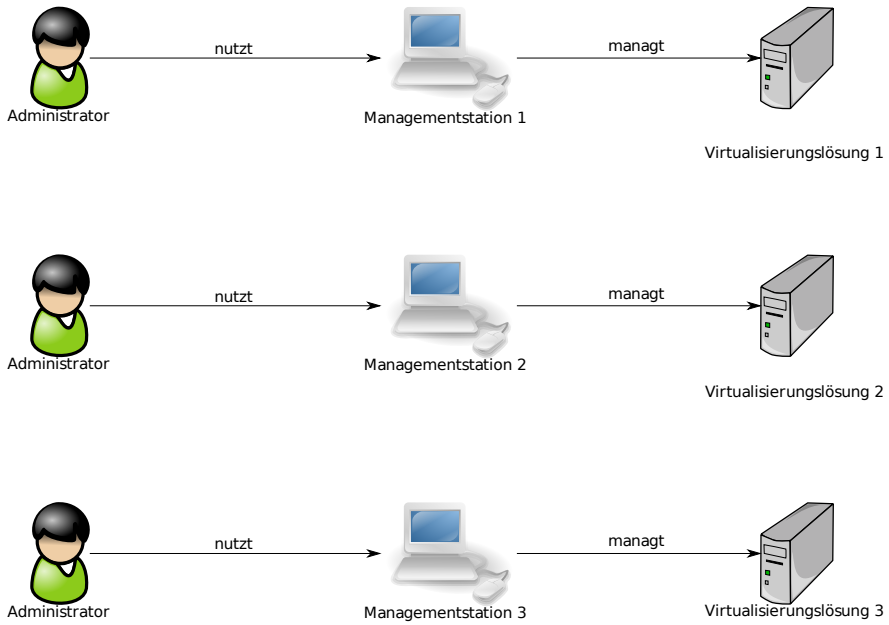

Abbildung 2.1: Die aktuelle Managementsituation

auch ineinander geschachtelte Gruppen möglich sein, womit eine Baumstruktur entsteht.

Plattformunabhängige Bedienbarkeit Aufgrund der verschiedenen Betriebssysteme, welche am Lehrstuhl eingesetzt werden, soll sich die Managementlösung plattformunabhängig bedienen lassen, z.B. durch ein Webinterface.

Integration in die bestehende Single-sign-on-Infrastruktur Um das bestehende Single-sign-on weiterzuführen soll die Authentifizierung der Benutzer durch den eingesetzten Verzeichnisdienst erfolgen. Generell ist eine Anbindungsmöglichkeit an verschiedene Verzeichnisdienste wünschenswert.

Die folgenden spezifischen Produkte, ermittelt aus der aktuellen IT-Infrastruktur des Lehrstuhls, sind zu unterstützen:

- Unterstützung von Xen, VMware ESX, sowie Virtual Server und Hyper-V von Microsoft
- Anbindung an den Verzeichnisdienst OpenLDAP
- Anbindung an eine Datenbank

2.4 Szenario für das Management virtueller Systeme am Lehrstuhl

Aus den vorherigen Abschnitten läßt sich die Kurzbeschreibung eines abstrakten Szenarios ableiten anhand dessen in den folgenden Abschnitten die Anforderungen an das Managementsystem aufgezeigt werden.

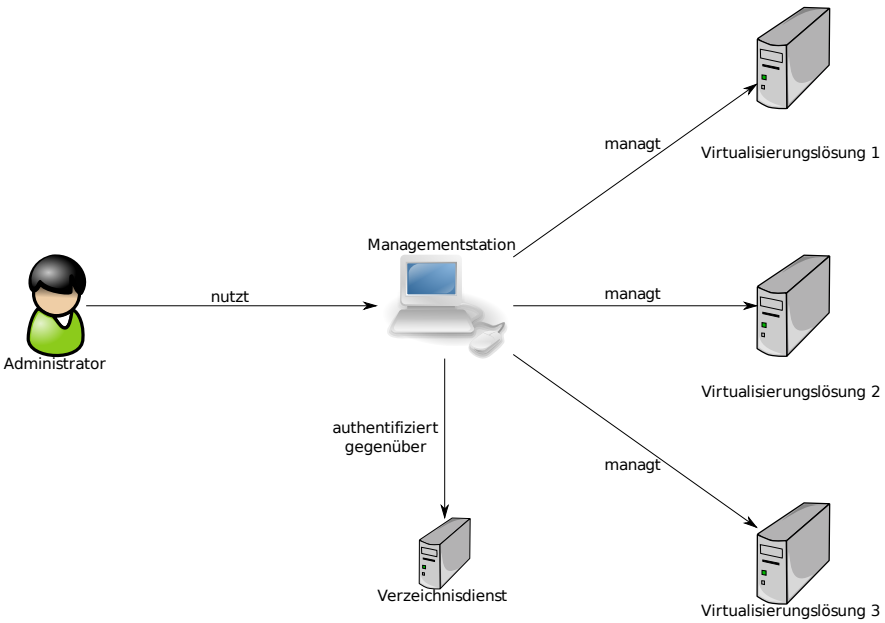

Abbildung 2.2: Die gewünschte Managementsituation

2.4.1 Kurzbeschreibung des Szenarios

Die zuständigen Administratoren des Lehrstuhls stellen Virtualisierungslösungen bereit. Verschiedene Nutzergruppen bekommen Zugriff auf Teile der Virtualisierungslösungen, das können Hostsysteme oder auch nur einzelne VMs sein. Die Nutzergruppen können wiederum Zuständigkeiten für einzelne VMs unter den Mitgliedern aufteilen, und evtl. dazu benötigte Untergruppen erstellen.

Die Verwaltung der eingesetzten Virtualisierungslösungen wird mittels eines zentrales Tools vorgenommen, die Nutzer greifen auf dieses betriebssytemunabhängig zu. Die Authentifizierung der Benutzer geschieht anhand des bereits vorhandenen Verzeichnisdienstes. Weitere vorhandene Systeme, z.B. ein Monitoringsystem, sind an das Managementsystem angebunden.

Die Ressourcen (Hosts und VMs) werden laufend überwacht. Ein Ausfall einer Ressource wird, wenn möglich, automatisch behoben. Kritische Systemzustände sollen ebenfalls, wenn möglich, automatisch bereinigt werden. Eine Benachrichtigung der zuständigen Stellen, sowie eine Speicherung der aufgetretenen Ereignisse und Aktionen, sollen vorgenommen werden.

Neue Ressourcen sollen, soweit möglich, durch ein Autodiscovery automatisch erkannt werden.

Verschiedene Speichertechnologien werden verwendet, u.a. Raid, LVM, iSCSI und Fiberchannel.

2.5 Anforderungen an das Management virtueller Systeme

Aus den Wünschen an das Management und das beschriebene Szenario ergeben sich die folgenden Anforderungen an das Management von Virtualisierungslösungen.

Abstraktion der Virtualisierungslösungen Der Wunsch nach einem einheitlichen Management aller eingesetzten Virtualisierungslösungen bedingt, dass Implementierung und Konzepte der einzelnen Virtualisierungslösungen vom Management abstrakt behandelt werden.

Mandantenfähigkeit Um die existierenden Organisationsstrukturen abbilden zu können muss das Management mandantenfähig sein, d.h. Teile der Ressourcen müssen eigenständig verwaltet werden können.

Reporting und Monitoring Im Rahmen der Überwachung und Kontrolle der eingesetzten Virtualisierer müssen geeignete Mechanismen vorhanden sein. Das System muss in der Lage sein Ereignisse und Managementinformationen zu verarbeiten und zu speichern.

Failover und Loadbalancing Das Managementsystem sollte es ermöglichen Ressourcen qualitativ zu überwachen. Ausgefallene Dienste sollen automatisch wiederhergestellt werden. Kritische Systemzustände, z.B. Überlastung eines Hosts, sollen, wenn möglich, automatisch bereinigt werden.

2.6 Anforderungen an die Managementarchitektur

Die Anforderungen an die Managementarchitektur werden anhand einiger exemplarischer Anwendungsfälle aufgezeigt, diese entstammen dabei dem aufgezeigten Szenario in Abschnitt 2.4. Zunächst werden die dazu benötigten Aktoren definiert, und anschließend die Anwendungsfälle erläutert.

2.6.1 Aktoren

Zur Beschreibung der Anwendungsfälle werden folgende Aktoren benötigt, die Reihenfolge orientiert sich dabei an der zunehmenden Berechtigung der Aktoren, d.h. die Berechtigungen sind kumulativ.

VM-N Der VM-Nutzer kontrolliert den Zustand seiner VM, d.h. er kann sie starten, stoppen u.ä.

VM-M Der VM-Manager konfiguriert die VMs, d.h. er legt z.B. die Menge des für die VM verfügbaren Hauptspeichers fest, oder fügt ihr virtuelle Festplatten hinzu. Er kann außerdem eine VM auf einen anderen Host migrieren.

Host-M Der Host-Manager konfiguriert Hosts, d.h. er legt u.a. fest welche Ressourcen für VMs zu Verfügung stehen oder welche VMs auf welchen Hosts laufen dürfen.

Sys-M Der System-Manager kann zusätzlich neue Hosts in das Managementsystem einbinden oder entfernen.

2.6.2 Anwendungsfälle

Die folgenden Anwendungsfälle stellen einen representativen Teilausschnitt der insgesamt anfallenden Anwendungsfälle dar.

Hinzufügen eines Hostsystems

Beschreibung: Der Sys-M fügt ein Hostsystem dem Managementsystem hinzu und nimmt eine grundlegende Konfiguration des Hosts vor

Vorbedingungen:

- Das Managementsystem ist bereit Anfragen entgegenzunehmen
- Der Sys-M ist am Managementsystem angemeldet

Primärszenario:

1. Das System wird vom Sys-M aufgefordert den Host hinzuzufügen
2. Das System überprüft die Berechtigung des Sys-M
3. Das System prüft ob eine Kommunikation mit dem Host möglich ist
4. Das System prüft ob der Host schon vorhanden ist
5. Das System fügt den Host hinzu
6. Die Parameter des neuen Hosts werden mit Standardwerten belegt
7. Das System vermerkt die Aktion
8. Das System meldet dem Sys-M die Ausführung

Nachbedingungen:

- Host ist im System vorhanden
- Host ist konfiguriert
- Aktion ist vermerkt

Sekundärszenarien:

- Sys-M ist nicht berechtigt einen Host hinzuzufügen
  1. Wie Primärablauf, aber bei 2. schlägt die Autorisierung fehl
  2. Das System vermerkt die fehlgeschlagene Autorisierung
  3. Rückmeldung an Sys-M

  Nachbedingungen:
  - Host ist dem System nicht hinzugefügt
  - Ereignis ist vermerkt
- Eine Kommunikation mit dem Host ist nicht möglich
  1. Wie Primärablauf, aber bei 3. schlägt die Kommunikation fehl
  2. Das System vermerkt die fehlgeschlagene Kommunikation
  3. Rückmeldung an Sys-M

  Nachbedingungen:
  - Host ist dem System nicht hinzugefügt
  - Ereignis ist vermerkt
- Host ist schon vorhanden
  1. Wie Primärablauf, aber bei 4. findet das System den Host im Verzeichnis
  2. Das System vermerkt das Ereignis
  3. Rückmeldung an Sys-M

  Nachbedingungen:
  - Host ist dem System nicht hinzugefügt
  - Ereignis ist vermerkt

Konfiguration eines Hostsystems

Beschreibung: Ein Hostsystem wird durch einen Host-M konfiguriert.

Vorbedingungen:

- Das Managementsystem ist bereit Anfragen entgegenzunehmen
- Der Host-M ist am Managementsystem angemeldet

Primärszenario:

1. Der Host-M fordert das System auf die gewünschten Parameter des Hosts zu setzen
2. Das System überprüft die Berechtigung des Host-M
3. Das System prüft ob die Parameter vorhanden sind
4. Das System prüft die gültigkeit der neuen Parameter innerhalb definierter Grenzen
5. Das System ändert die entsprechenden Parameter
6. Das System vermerkt die Aktion
7. Das System meldet dem Host-M die Ausführung

Nachbedingungen:

- Host ist konfiguriert
- Die Aktion ist vermerkt

Sekundärszenarien:

- Host-M ist nicht berechtigt den Host zu konfigurieren
  1. Wie Primärablauf, aber bei 2. schlägt die Autorisierung fehl
  2. Das System vermerkt die fehlgeschlagene Autorisierung
  3. Rückmeldung an Host-M

  Nachbedingungen:
  - Host ist nicht konfiguriert
  - Ereignis ist gemeldet
- Parameter nicht vorhanden
  1. Wie Primärablauf, aber bei 3. schlägt die Suche nach den Parametern fehl
  2. Das System vermerkt das Ereignis
  3. Rückmeldung an Host-M

  Nachbedingungen:
  - Host ist nicht konfiguriert
  - Ereignis ist vermerkt
- Neue Parameter nicht gültig
  1. Wie Primärablauf, aber bei 4. schlägt Prüfung der neuen Parameter fehl
  2. Das System vermerkt das Ereignis
  3. Rückmeldung an Host-M

  Nachbedingungen:
  - Host ist nicht konfiguriert
  - Ereignis ist vermerkt

Verschieben einer VM im laufenden Betrieb

Beschreibung: Eine VM wird durch einen VM-M im Betrieb auf einen anderen Host verschoben

Vorbedingungen:

- Das Managementsystem ist bereit Anfragen entgegenzunehmen
- Der VM-M ist am Managementsystem angemeldet
- Die VM existiert und läuft und kann im Betrieb verschoben werden
- Zielhost und VM sind vorhanden

Primärszenario:

1. Der VM-M fordert das System auf die VM auf einen Host zu verschieben
2. Das System überprüft die Berechtigung des VM-M
3. Das System prüft ob der Host erreichbar ist
4. Das System prüft ob die VM auf dem Host schon läuft
5. Das System prüft ob die VM auf den Host verschoben werden kann
6. Das System verschiebt die VM auf den Host
7. Das System speichert den neuen Ort der VM
8. Das System vermerkt die Aktion
9. Das System meldet dem VM-M die Ausführung

Nachbedingungen:

- Die VM läuft nicht auf dem Quellhost
- Die VM läuft auf dem Zielhost
- Die Aktion ist vermerkt

Sekundärszenarien:

- VM-M ist nicht berechtigt die VM zu verschieben
  1. Wie Primärablauf, aber bei 2. schlägt die Autorisierung fehl
  2. Das System vermerkt die fehlgeschlagene Autorisierung
  3. Rückmeldung an VM-M

  Nachbedingungen:
  - VM läuft auf Quellhost
  - VM läuft nicht auf Zielhost
  - Ereignis ist gemeldet
- Der Host ist nicht erreichbar
  1. Wie Primärablauf, aber bei 2. schlägt die Zustandsprüfung des Hosts fehl
  2. Das System vermerkt die Aktion
  3. Rückmeldung an VM-M

  Nachbedingungen:
  - VM läuft auf Quellhost
  - VM läuft nicht auf Zielhost
  - Ereignis ist gemeldet

- VM läuft schon auf Zielhost
  1. Wie Primärablauf, aber bei 4. erkennt das System, dass die VM schon auf dem Zielhost läuft
  2. Das System vermerkt das Ereignis
  3. Rückmeldung an VM-M

  Nachbedingungen:
  - VM läuft auf Quellhost
  - VM läuft nicht auf Zielhost
  - Ereignis ist gemeldet
- VM kann nicht auf den Zielhost verschoben werden
  1. Wie Primärablauf, aber bei 5. kann die VM nicht auf den Zielhost verschoben werden
  2. Das System vermerkt das Ereignis
  3. Rückmeldung an Host-M

  Nachbedingungen:
  - VM läuft auf Quellhost
  - VM läuft nicht auf Zielhost
  - Ereignis ist gemeldet

Steuern einer VM

Beschreibung: Eine VM wird durch einen Steuerbefehl in einen neuen Zustand überführt

Vorbedingungen:

- Das Managementsystem ist bereit Anfragen entgegenzunehmen
- Der VM-N ist am Managementsystem angemeldet

Primärszenario:

1. Der VM-N fordert das System auf die in einen neuen Zustand zu überführen
2. Das System überprüft die Berechtigung des VM-M
3. Das System prüft ob die VM existiert
4. Das System prüft ob die VM in den neuen Zustand überführt werden kann
5. Das System überführt die VM in den neuen Zustand
6. Das System speichert den neuen Zustand der VM
7. Das System vermerkt die Aktion
8. Das System meldet dem VM-N die Ausführung

Nachbedingungen:

- Die VM ist in den neuen Zustand überführt
- Die Aktion ist vermerkt

Sekundärszenarien:

- VM-M ist nicht berechtigt die Aktion auszuführen
  1. Wie Primärablauf, aber bei 2. schlägt die Autorisierung fehl
  2. Das System vermerkt die fehlgeschlagene Autorisierung

3. Rückmeldung an VM-N

Nachbedingungen:

- VM ist im ursprünglichen Zustand
- Ereignis ist gemeldet

- VM existiert nicht
  1. Wie Primärablauf, aber bei 3. erkennt das System, dass die VM nicht existiert
  2. Das System vermerkt das Ereignis
  3. Rückmeldung an VM-N

  Nachbedingungen:
  - VM ist im ursprünglichen Zustand
  - Ereignis ist gemeldet
- VM kann nicht in den neuen Zustand überführt werden
  1. Wie Primärablauf, aber bei 4. kann die VM nicht in den neuen Zustand überführt werden.
  2. Das System vermerkt das Ereignis
  3. Rückmeldung an VM-N

  Nachbedingungen:
  - VM ist im ursprünglichen Zustand
  - Ereignis ist gemeldet

Erstellen einer VM aus einer Vorlage

Beschreibung: Eine VM wird aus einer Vorlage erstellt

Vorbedingungen:

- Das Managementsystem ist bereit Anfragen entgegenzunehmen
- Der VM-M ist am Managementsystem angemeldet

Primärszenario:

1. Der VM-N fordert das System auf aus einer Vorlage eine neue VM zu erstellen
2. Das System überprüft die Berechtigung des VM-M
3. Das System prüft ob die Vorlage existiert
4. Das System erstellt die VM gemäß der Vorlage
5. Das System fügt die neue VM dem System hinzu
6. Das System vermerkt die Aktion
7. Das System meldet dem VM-M die Ausführung

Nachbedingungen:

- Die neue VM existiert
- Die neue VM ist dem System hinzugefügt
- Die Aktion ist vermerkt

Sekundärszenarien:

- VM-M ist nicht berechtigt die Aktion auszuführen

1. Wie Primärablauf, aber bei 2. schlägt die Autorisierung fehl
2. Das System vermerkt die fehlgeschlagene Autorisierung
3. Rückmeldung an VM-M

Nachbedingungen:

- Neue VM existiert nicht
- Ereignis ist gemeldet

- Vorlage existiert nicht
  1. Wie Primärablauf, aber bei 3. erkennt das System, dass die Vorlage nicht existiert
  2. Das System vermerkt das Ereignis
  3. Rückmeldung an VM-M

  Nachbedingungen:
  - VM ist im ursprünglichen Zustand
  - Ereignis ist gemeldet

Klonen einer VM

Beschreibung: Eine VM wird geklont

Vorbedingungen:

- Das Managementsystem ist bereit Anfragen entgegenzunehmen
- Der VM-M ist am Managementsystem angemeldet
- Die zu klonende VM existiert

Primärszenario:

1. Der VM-N fordert das System auf aus einer VM eine neue VM zu erstellen
2. Das System überprüft die Berechtigung des VM-M
3. Das System erstellt die neue VM
4. Das System fügt die neue VM dem System hinzu
5. Das System fordert den Benutzer auf die VM zu konfigurieren
6. Das System vermerkt die Aktion
7. Das System meldet dem VM-M die Ausführung

Nachbedingungen:

- Die neue VM existiert
- Die neue VM ist dem System hinzugefügt
- Die neue VM ist konfiguriert
- Die Aktion ist vermerkt

Sekundärszenarien:

- VM-M ist nicht berechtigt die Aktion auszuführen
  1. Wie Primärablauf, aber bei 2. schlägt die Autorisierung fehl
  2. Das System vermerkt die fehlgeschlagene Autorisierung
  3. Rückmeldung an VM-M

  Nachbedingungen:

  - Neue VM existiert nicht
  - Ereignis ist gemeldet
- Die Konfiguration der neuen VM schlägt fehl
  1. Wie Primärablauf, aber bei 5. schlägt die Konfiguration fehl
  2. Das System vermerkt das Ereignis
  3. Rückmeldung an VM-M

  Nachbedingungen:
  - VM hat dieselbe Konfiguration wie die geklonte
  - Ereignis ist gemeldet

2.6.3 Zusammenfassung der Managmentfunktionen

Aus den gezeigten Anwendungsfällen ergibt sich die folgende Aufstellung an zu unterstützenden Managementfunktionen:

- Hinzufügen/Entfernen von Ressourcen
- Konfigurieren der Ressourcen
- Monitoring der Ressourcen
- Dienstgüteüberwachung der Ressourcen
- Zustandsüberwachung der Ressourcen
- VM-Migration
- VM-Steuerung
- VM-Vorlagen
- Klonen von VMs
- Benutzer-/Gruppenverwaltung
- Benachrichtigungen
- Autodiscovery

2.6.4 Nichtfunktionale Anforderungen

Zusätzlich zu den beschriebenen funktionalen Anforderungen, die größtenteils Aspekte des Funktionsmodells betreffen, kann man den beschriebenen Anwendungsfällen weitere nichtfunktionale Anforderungen entnehmen.

Zugriffsberechtigungen für Ressourcen

In den Anwendungsfällen sind die Ressourcen, also Host und VM, zugriffsbeschränkt. Jede Aktion eines Nutzers hat, ungeachtet seiner Aktorenrolle, zuallererst eine Prüfung seiner Zugriffberechtigung zur Folge. Für die Managementarchitektur bedeutet dieser Sachverhalt, dass feingranulare Zugriffsrechte der Benutzer und Gruppen auf Ressourcen zu setzen sein müssen.

Einheitliche MIB

Die Abstraktion der Virtualisierungslösungen verlangt nach einer gemeinsamen Informationsbasis, d.h. die verschiedenen MIBs der Virtualisierer müssen durch eine einheitliche MIB abstrahiert werden.

Technische Anforderungen

Aus den beschriebenen Sachverhalten lassen sich auch einige technische Anforderungen an die Managementarchitektur ableiten.

Anbindung an Datenbanken Die Managementinformationen werden in einer Datenbank gespeichert.

Anbindung an Verzeichnisdienste Die Authentifizierung der Benutzer erfolgt gegen einen Verzeichnisdienst.

Plattformunabhängige Bedienbarkeit Die Managementarchitektur muss einen plattformunabhängigen Zugriff bereitstellen, z.B. durch ein Webinterface.

Skalierbarkeit in größeren Umgebungen In größeren Umgebungen soll die Skalierbarkeit gewährleistet sein.

2.7 Zusammenfassung

Die aktuelle Situation beim Management der am Lehrstuhl eingesetzten Virtualisierungslösungen wurde beschrieben und die dabei auftretenden Defizite dargestellt. Aus diesen wurde ein Wunschkatalog an ein zukünftiges Management erstellt und ein entsprechendes Szenario beschrieben.

Anschließend wurden daraus anhand einiger Anwendungsfälle die funktionalen Anforderungen an ein Managementsystem beschrieben und die damit verknüpften nichtfunktionalen Anforderungen dargestellt.

Eine Zusammenfassung der Anforderungen ist in Tabelle 4.1 zu finden.

Tabelle 2.1: Zusammenfassung der Anforderungen

Anforderungen an das Management
Abstraktion der Virtualisierungslösungen
Mandantenfähigkeit
Reporting und Monitoring
Failover und Loadbalancing
Anforderungen an die Architektur
Funktionale Anforderungen
Hinzufügen/Entfernen von Ressourcen
Konfigurieren der Ressourcen
Monitoring der Ressourcen
Dienstgüteüberwachung der Ressourcen
Zustandsüberwachung der Ressourcen
VM-Migration
VM-Steuerung
VM-Vorlagen
Klonen von VMs
Benutzer-/Gruppenverwaltung
Benachrichtigungen
Autodiscovery
Nichtfunktionale Anforderungen
Einheitliche MIB
Zugriffsberechtigung auf Ressourcen
Abstrakte Anbindung an Datenbanken
Abstrakte Anbindung an Verzeichnisdienste
Schnittstelle für externe Anwendungen
Plattformunabhängige Bedienbarkeit
Skalierbarkeit in größeren Umgebungen
Führen einer Datenhistorie

3 State of the Art

Dieses Kapitel zeig den aktuellen Stand der Managementtools für Hostvirtualisierungslösungen anhand einiger Beispiele auf. Zuerst werden die gewählten Tools kurz vorgestellt und anschließend anhand einer Tabelle gegenübergestellt.

3.1 Bestehende Managementtools

Es existieren zahlreiche Managementtools für Virtualisierungslösungen, sowohl im kommerziellen Umfeld wie auch als Freie Software. Die folgende Aufstellung zeigt anhand einiger Beispiele welchen Anforderungen diese Tools genügen. Der Fokus liegt hierbei auf den zu unterstützenden Virtualisierungstechnologien Xen und VMware. Als Beispiele wurde einmal die Managementsoftware von VMware gewählt, da sie weit verbreitet ist und viele Funktionalitäten unterstützt. Dazu kommen ein Tool für Xen sowie ein virtualisiererübergreifendes Tool.

Die Aufzählung der Funktionalitäten orientiert sich an den im vorangegangenen Kapitel erstellten Anforderungen.

3.1.1 VMware Virtual Center

Im kommerziellen Umfeld sind die VMware-Produkte (u.a. VMware ESX Server) die verbreitetsten Virtualisierungslösungen. Entsprechend umfangreich ist das dazugehörige Produkt VMware Virtual Center [vir08].

Virtual Center besteht aus verschiedenen Komponenten (Server, Database, Client, Agent und Web Access) welche flexibel zu einer Managementinfrastruktur zusammengefügt werden können. Es bietet eine einheitliche Sicht auf die verschiedenen Ressourcen, z.B. ESX Server und virtuelle Maschinen, und stellt deren gesamte Funktionalität zu Verfügung.

Virtual Center ist eine umfassende und bewährte Managementlösung, die jedoch nur für VMware-Virtualisierungslösungen genutzt werden kann. Es existieren momentan keine Erweiterungen, die es ermöglichen andere Produkte, z.B. Xen, anzubinden. Ein weiterer Nachteil ist die Beschränkung auf Windows-Systeme sowie die eingeschränkte Unterstützung von Verzeichnisdiensten, hier wird aktuall nur Active Directory unterstützt.

Folgende Funktionalitäten bietet Virtual Center u.a.:

- Unterstützte Virtualisierungslösungen sind alle VMware Produkte
- Authentifizierung gegenüber Active Directory
- Mandantenfähigkeit
- API für Managementanwendungen
- Webinterface

- Umfangreiches Reporting (Ereignisse, Alarme, Performanz)
- Historie
- VM-Konfiguration (online/offline)
- VM-Templates
- VM Migration (online, offline, physical2virtual, physical2physical, storage)
- VM Cloning
- Zugriff auf VM-Konsole (auch über Webinterface)
- Loadbalancing
- Failover

3.1.2 Enomalism

Enomalism [eno08] ist eine webbasierte Plattform für virtuelle Infrastrukturen. Enomalism steht unter einer Open-Source Lizenz (AGPL) und benötigt ein Linux-Betriebssystem. Virtualisierer bindet es mittels Libvirt [lib08] ein, dieses unterstützt u.a. Xen, Qemu und OpenVZ.

Sie bietet u.a. folgende Funktionalitäten:

- Unterstützung von Xen u.a. über libvirt
- Authentifizierung gegenüber LDAP
- API für Managementanwendungen
- Webinterface
- Reporting (nur Performanz)
- VM-Migration (online/offline, physical2virtual/virtual2virtual über externe Tools)
- VM-Templates
- Migration (offline)

3.1.3 Convirt

Convirt [con08], ehemals Xenman, ist ein Open-Source Projekt und bietet umfangreiche Möglichkeiten zur Verwaltung der Virtualisierungslösung Xen, eingeschränkt wird auch KVM [kvm08] unterstützt. Convirt benötigt ein Linux-Betriebssystem, stellt jedoch kein Webinterface zum plattformunabhängigen Management zu Verfügung.

Folgenden Funktionalitäten werden u.a. angeboten:

- Unterstützung von Xen, KVM
- VM-Konfiguration (online/offline)

- VM-Templates
- VM-Migration (online/offline, physical2virtual/virtual2virtual über externe Tools)
- Zugriff auf VM-Konsole

Die Roadmap des Projekts listet u.a. folgende Funktionalitäten:

- Authentifizierung gegenüber LDAP
- API für Managementanwendungen
- Webinterface

3.1.4 IBM Virtualization Manager

Virtualization Manager [ibm08] ist ein Plugin für IBM Director, einer Programmsuite für das Systemmanagement. Die Erweiterung fügt Unterstützung für virtuelle Systeme von VMware, Microsoft und für Xen (experimentell) hinzu. Damit können Hostsystems und VMs gesteuert und verwaltet werden, außerdem ist die manuelle und automatische Migration von virtuellen Maschinen möglich. Im Falle von VMware wird jedoch Virtual Center benötigt um eine Live-Migration durchführen zu können.

Die Funktionalitäten umfassen u.a.:

- Unterstützung für VMware-Produkte, Microsoft Virtual Server und Xen
- Webinterface (über Director)
- Reporting (Ereignisse, Alarme)
- VM Migration (online (nur VMware), offline, virtual2virtual)

3.2 Vergleich der Funktionalitäten der Managementtools

Die folgende Tabelle (3.1) gibt einen Überblick über die Funktionalitäten der vorgenannten Managementlösungen. Sie ist nach den aufgestellten Anforderungen aus dem vorherigen Kapitel gegliedert.

Tabelle 3.1: Vergleich der Managementtools mit den Anforderungen

	VC	E	C	VM
Funktionale Anforderungen				
Hinzufügen/Entfernen von Ressourcen	✓	✓	✓	✓
Konfigurieren der Ressourcen	✓	✓	✓	×
Monitoring der Ressourcen	✓	✓	×	×
Dienstgüteüberwachung der Ressourcen	✓	✓	×	×
Zustandsüberwachung der Ressourcen	✓	✓	×	×
VM-Migration	✓	(✓)	(✓)	(✓)
VM-Steuerung	✓	✓	✓	✓
VM-Vorlagen	✓	✓	✓	×
Klonen von VMs	✓	×	×	×
Benutzer-/Gruppenverwaltung	✓	✓	×	×
Benachrichtigungen	✓	✓	×	✓
Autodiscovery	×	×	×	×
Nichtfunktionale Anforderungen				
Einheitliche MIB	×	(✓)	(✓)	(✓)
Zugriffsberechtigung auf Ressourcen	✓	✓	×	×
Abstrakte Anbindung an Datenbanken	×	×	×	×
Abstrakte Anbindung an Verzeichnisdienste	×	×	×	×
Schnittstelle für externe Anwendungen	✓	✓	(✓)	✓
Plattformunabhängige Bedienbarkeit	✓	✓	(✓)	×
Skalierbarkeit in größeren Umgebungen	✓	×	×	×
Führen einer Datenhistorie	✓	×	×	×

Abkürzungen

VC	VMWare Virtual Center
E	Enomalism
C	Convirt
VM	IBM Virtualization Manager
(✓)	(eingeschränkt) unterstützt
×	nicht unterstützt

4 Analyse und Modellierung

In diesem Kapitel wird zuerst die Umsetzung der Anforderungen analysiert und erläutert. Anschließend wird daraus die Managementarchitektur generiert und die Managementteilmodelle beschrieben.

4.1 Ideen zur Umsetzung der Anforderungen

Dieser Abschnitt zeigt Ideen zur Umsetzung der Anforderungen in der Managementarchitektur auf.

4.1.1 Funktionale Anforderungen

Die Einordnung der funktionalen Anforderungen wird anhand der Beschreibung der Managementanwendungen einer Managementarchitektur nach [HAN99, S. 288 ff] vorgenommen. Darin werden die gebräuchlichsten Anwendungen beschrieben und ihre Aufgaben kurz erläutert.

In Teilen wird auch auf den Aufbau einer Managementplattform, beschrieben in [HAN99, S. 277 ff], eingegangen. Darin werden die Architektur einer Plattform sowie deren Bestandteile erläutert und deren Aufgabengebiete abgegrenzt.

Hinzufügen/Entfernen von Ressourcen Das Hinzufügen und entfernen von Ressourcen ist Bestandteil des *Konfigurationsmanagements* und wird in diesem vorgenommen. Änderungen werden gegenüber der Informationsverwaltung validiert.

Konfigurieren der Ressourcen Das Konfigurieren von Ressourcen ist ebenfalls Bestandteil des *Konfigurationsmanagements*. Auch hier werden die Änderungen mit der Informationsverwaltung abgeglichen.

Monitoring der Ressourcen Die *Leistungsüberwachung* nimmt das Monitoring der Ressourcen vor. Sie tätigt in definierten Abständen Messungen und gibt die Ergebnisse an das Informationsmodell zur Speicherung weiter.

Dienstgüteüberwachung der Ressourcen Die Dienstgüte der Ressourcen wird in der *Schwellwertüberwachung* definiert und auch überwacht. Abnormale Systemzustände werden an den *Ereignismanager* weitergeleitet. Weitere Anwendungen, z.B. ein *Loadbalancer*, können bei überschreiten von Schwellwerten eine Neukonfiguration des Managements vornehmen um wieder einen stabilen Systemzustand herzustellen.

Zustandsüberwachung der Ressourcen Die *Zustandsüberwachung* prüft in definierten Abständen den Zustand der Ressourcen und und generiert im Problemfall eine Ereignismeldung. Weitere Anwendungen, z.B. ein *FailoverManagement*, können ausgefallene Dienste oder Ressourcen wiederherstellen.

VM-Migration Die VM-Migration wird vom *Konfigurationsmanagement* durchgeführt. Auch hier werden die Änderungen mit der Informationsverwaltung abgeglichen.

VM-Steuerung Die VM-Steuerung ist ebenfalls Teil des *Konfigurationsmanagements*. Die möglichen Steueroptionen werden anhand eines VM-Zustandsmodells festgelegt.

VM-Vorlagen Im Rahmen des *Konfigurationsmanagements* können VM-Vorlagen erstellt werden. Diese definieren die Eigenschaften einer VM sowie deren Komponenten. Aus dieser abstrakten Beschreibung können schließlich spezifische VMs erstellt werden.

Klonen von VMs Das Klonen von VMs ist ebenfalls Teil des *Konfigurationsmanagements*. Die abstrakte Funktion „Klonen„wird durch eine spezifische Funktion ergänzt, welche das Klonen für die entsprechende VM-Technologie vornimmt.

Benutzer-und Gruppenverwaltung Das *Security Management* übernimmt die Aufgaben der Benutzer- und Gruppenverwaltung, d.h. verwalten der Benutzer und Gruppen, Zuordnung der Benutzer zu Gruppen, sowie Authentifizierung und Autorisierung.

Benachrichtigungen Der *Ereignismanager* übernimmt die zentrale Verarbeitung von Ereignismeldungen. Ereignisse werden an andere Applikationen übergeben, damit diese eine weitere Verarbeitung vornehmen können. Z.B. kann ein *Notificationmanagement* Benachrichtigungen per E-Mail an die zuständigen Benutzer senden, oder einen Eintrag in entsprechende Logfiles vornehmen.

Autodiscovery Der *Topologiemanager* nimmt ein periodisches Suchen nach neuen Ressourcen vor. Je nach Konfiguration können z.B. neue Ressourcen dem System hinzugefügt oder nur Benachrichtigungen ausgelöst werden.

4.1.2 Nichtfunktionale Anforderungen

Als Grundlage dient hierbei die Beschreibung einer Managementarchitektur nach [HAN99, S. 97 ff]. Darin wird der allgemeine Aufbau einer Managementarchitektur erläutert und deren wesentliche Bestandteile, v.a. die Managementteilmodelle, beschrieben.

Wie im vorherigen Abschnitt werden außerdem Teile der Beschreibung von Managementplattformen nach [HAN99, S. 277 ff] verwendet.

Abstraktion der Virtualisierungslösungen Innerhalb einer Managementplattform ist der *Kommunikationsbaustein* für die Kommunikation mit den Ressourcen zuständig. Um den Anforderungen gerecht zu werden muss innerhalb des Bausteins eine zusätzliche Abstraktionsschicht eingeführt werden. Diese versteckt die spezifischen Kommunikationsanforderungen einer Virtualisierungslösung vor den übrigen Teilen der Managementplattform.

Zugriffsberechtigung auf Ressourcen Feingranulare Zugriffsberechtigungen auf die Ressourcen, im Zusammenspiel mit einer Benutzer- und Gruppenverwaltung, sind wichtig um spezifische Organisationsstrukturen umsetzen zu können. Die Benutzer- und Gruppenverwaltung geschieht gegenüber einem Verzeichnisdienst, die dazugehörigen Zugriffsrechte dagegen werden in der angebundenen Datenbank gespeichert. Hier findet sich auch eine Zuordnung der Benutzer und Gruppen zu den Zugriffsrechten. Die Autorisierung der Benutzer geschieht durch das *Security Management*. Dieses bekommt die nötigen Informationen aus der Informationsverwaltung.

Abstrakte Anbindung an Datenbanken Ähnlich wie bei der Abstraktion der Virtualisierungslösungen wird innerhalb der *Informationsverwaltung* eine weitere Abstraktionsschicht eingezogen. Mittels Adaptern kann die Informationsverwaltung auf verschiedene, wenn nötig auch gleichzeitig auf mehrere, Datenbanken zugreifen.

Abstrakte Anbindung an Verzeichnisdienste Ebenso gestaltet sich die Anbindung der Verzeichnisdienste. Auch hier nehmen Adapter eine flexible Anbindung verschiedener Verzeichnsidienste vor.

Schnittstelle für externe Anwendungen Der Oberflächenbaustein einer Managementplattform stellt einen Zugangspunkt für Benutzer bereit. Durch eine Abstraktionsschicht kann hier die Anbindung verschiedener externer Anwendungen erreicht werden. Dazu werden *Zugangsadapter* eingeführt, welche verschiedene Protokolle zur Kommunikation mit der Managementplattform bereitstellen. Z.B. könnte über einen *SNMP-Zugangsadapter* ein Monitoring-Tool Informationen aus der Plattform beziehen.

Plattformunabhängige Bedienbarkeit Über die Protokolladapter kann ebenfalls ein Webinterface an die Plattform angebunden werden. Z.B. kann eine mittels PHP programmierte Webanwendung über Soap mit einem entsprechenden *Soap-Zugangsadapter* kommunizieren und dem Benutzer damit den Zugriff auf die Plattform bereitstellen.

Skalierbarkeit in größeren Umgebungen Um die Skalierbarkeit in größeren Umgebungen zu gewährleisten müssen u.a. Kommunikationswege verkürzt, die Anzahl der Kommunikationsverbindungen eingeschränkt und übertragene Daten minimiert werden. Folgende Maßnahmen sollen dazu beitragen (siehe dazu Abbildung 4.1).

Hierarchie aus Plattformen Eine Hierarchie aus interagierenden Managementplattformen soll die Kommunikationsverbindungen pro Plattform einschränken. Je nach Bedarf wäre es möglich weitere Plattformen dem System hinzuzufügen um die Last weiter zu verteilen.

Dynamische Funktionalitäten einer Plattform Zusätzlich könnten die Plattformen unterschiedliche Funktionalitäten erfüllen. So kann z.B. eine Plattform als Zwischenspeicher dienen und Informationen nur weiterleiten wenn diese gebraucht werden, z.B. durch einen Benutzerabruf, oder wenn Schwellerte überschritten werden. Die Abbildung zeigt eine Hierarchie aus Plattformen mit unterschiedlichen Rollen.

Zugriff auf Managementobjekte auf jeder Hierarchiestufe möglich Als weitere Maßnahme kann man Benutzerzugriffe auf tiefere Hierarchieschichten verschieben. Verwaltet ein Benutzer z.B. einen Teilbaum des Managementsystems, dann müssten alle abgerufenen Informationen bis zur Wurzel des Baums weitergereicht werden da sich die Benutzerschnittstelle hier befindet. Entsprechend ist dann die Auslastung der beteiligten Knoten, vor allem auf höheren Hierarchiestufen. Durch zusätzliche Benutzerschnittstellen auf tieferen Hierarchiestufen kann dieses Problem jedoch gemildert werden. Die Abbildung zeigt ein Beispiel mit Zugriffsmöglichkeiten auf verschiedenen Ebenen.

Führen einer Datenhistorie Das führen einer Historie kann durch Speicherung der anfallenden Managementinformationen erreicht werden. Jedoch ist hier auf Dauer eine Aggregationsfunktion nötig um die gespeicherten Informationen zu konsolidieren, da ansonsten der Umfang der Daten mit der Zeit zu groß wird.

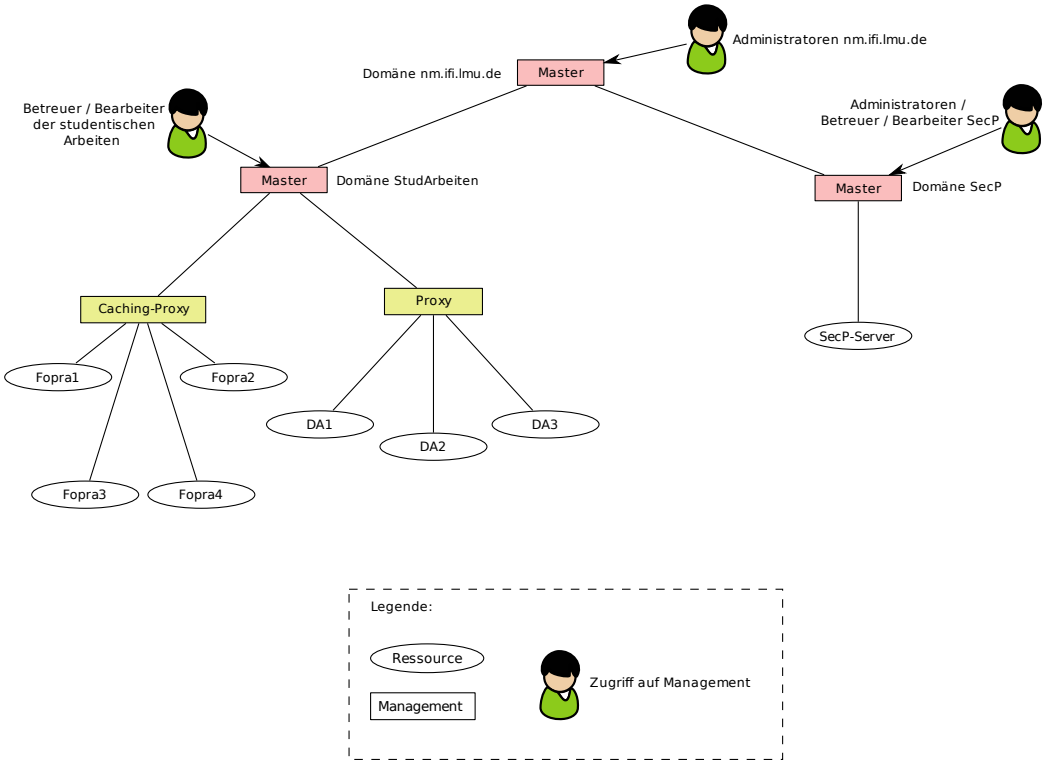

Abbildung 4.1: Beispielhierarchie mit verschiedenen Rollen und Zugriffsmöglichkeiten

4.2 Die Architektur der Managementplattform

Im letzten Abschnitt wurden die Ideen zur Umsetzung der Anforderungen skizziert und grob in die einzelnen Teile der Managementarchitektur nach [HAN99] eingeordnet. Dieser Abschnitt beschreibt nun das Gesamtmodell der sich daraus ergebenden Managementplattform.

4.2.1 Die Managementplattform

Die Managementplattform ist die zentrale Komponente des Managementsystems und besteht aus verschiedenen Infrastrukturbausteinen. Diese stellen die grundlegenden Dienste der Managementplattform zur Verfügung. Der Aufbau der Managementplattform ist angelehnt an die Beschreibung eines allgemeinen Managementsystems in [HAN99, S. 277ff] und ist dargestellt in Abbildung 4.2. Die Darstellung der Bausteine der Plattform erfolgt dabei gemäß der Abbildung vertikal absteigend.

Der Oberflächenbaustein

Der Oberflächenbaustein bietet einen Zugriff auf das System von ‘außen’. An dieser Schnittstelle werden alle relevanten Informationen zu Verfügung gestellt, die nötig sind, um das Managementsystem und die daran angebundenen Ressourcen managen zu können. Daran angebunden werden können u.a. grafische Benutzerschnittstellen und/oder Monitoringsys-

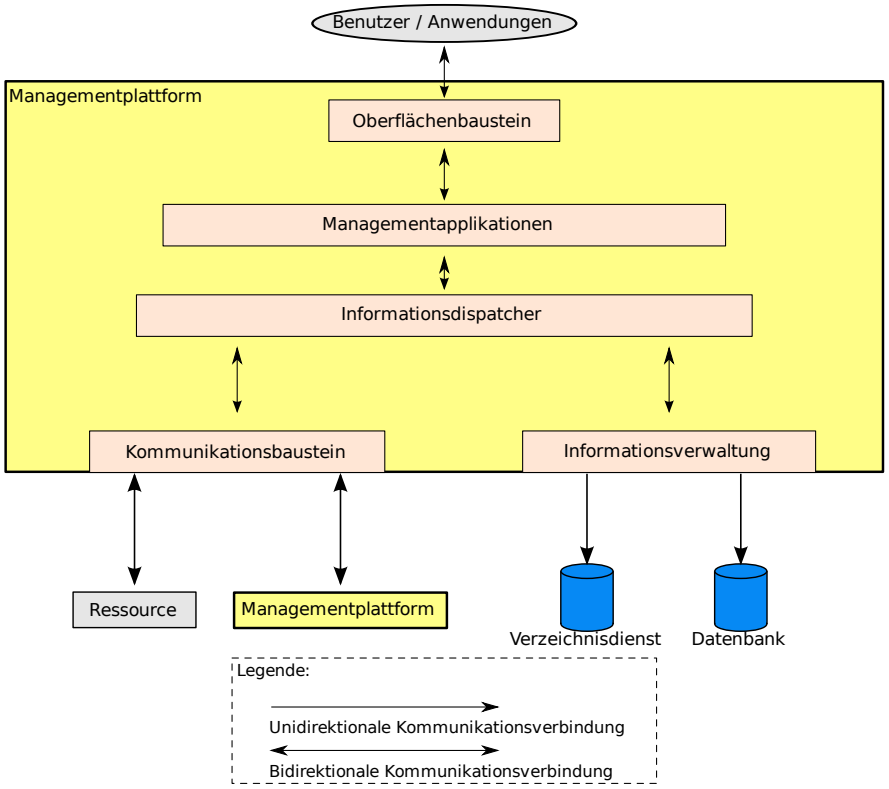

Abbildung 4.2: Der Aufbau der Managementplattform

teme. Um verschiedenen Systemen den Zugriff effizient zu ermöglichen ist der Zugriff nicht auf eine bestimmte Technik festgelegt, vielmehr können verschiedene Zugangspunkte mittels *Zugangsadapter* definiert werden. Der Oberflächenbaustein ist in Abbildung 4.3, mit verschiedenen Zugriffspunkten, für SNMP und SOAP, grafisch dargestellt.

Die Managementapplikationen

Ein weiterer Baustein sind die Managementapplikationen, dargestellt in Abbildung 4.4. Sie stellen die eigentlichen Funktionalitäten der Managementplattform dar. Sie sind für die Verarbeitung der gesammelten Informationen zuständig und enthalten u.a. Module zur Konfiguration der Ressourcen und zur Ereignisverarbeitung. Die benötigten Managementanwendungen, angelehnt an die in [HAN99, S. 288 ff] beschriebenen Applikationen, werden im Folgenden kurz beschrieben.

Konfigurationsmanager Der Konfigurationsmanager ist zuständig für die Konfiguration der Ressourcen. Er stellt die aktuellen Informationen über die Ressourcen zu Verfügung und bietet auch eine Konfigurationsänderung dieser an.

Leistungsmonitor Der Leistungsmonitor definiert die zu überwachenden Managementobjekte und legt das Intervall der Messungen fest. Er führt das Monitoring anhand der festgelegten Intervalle durch.

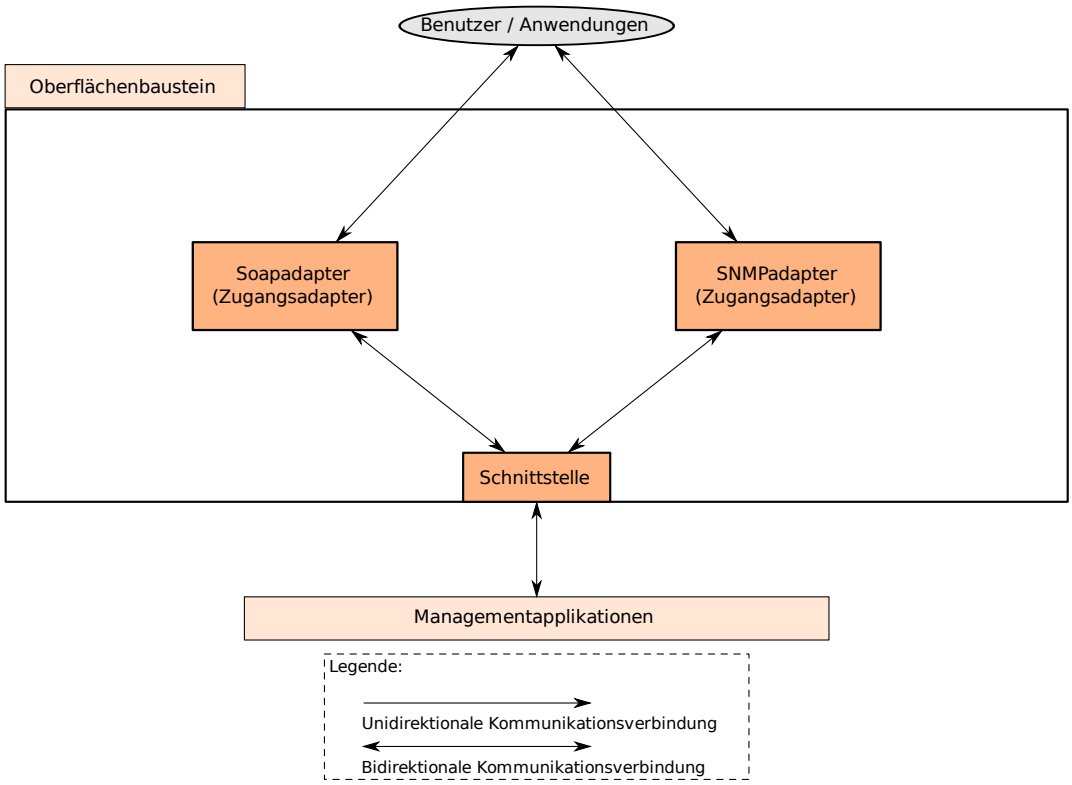

Abbildung 4.3: Verschiedene bereitgestellte Zugangspunkte durch Zugangsadapter

Schwellwertüberwachung Die Schwellwertüberwachung prüft die eingehenden Monitoringdaten gegen definierte Schwellwerte und löst bei Über- bzw. Unterschreitung eine Ereignismeldung aus. Die Schwellwerte können von berechtigten Benutzern für einzelne Managementobjekte getrennt vorgegeben werden. Somit ist eine feingranulare Überwachung der Ressourcen je nach Erforderniss möglich.

Ereignismanager Der Ereignismanager nimmt Ereignismeldungen entgegen und verteilt sie an die Managementapplikationen. Diese können die Informationen dann zu einer weiteren Verarbeitung nutzen, z.B. ein Loadbalancer die von der Schwellwertüberwachung generierten Events. Der Ereignismanager stellt verschiedene Klassen von Ereignismeldungen bereit, für die sich die Managementanwendungen registrieren können. Bei Erhalt eines Ereignisses sendet er dieses an die entsprechenden Empfänger.

Loadbalancer Der Loadbalancer prüft anhand gegebener Schwellwerte die Stabilität des Systems. Im Falle einer Überlastungssituation ergreift er geeignete Maßnahmen um die Stabilität des Systems wiederherzustellen, z.B. durch verschieben einer rechenintensiven VM.

Zustandsüberwachung Der Zustandsmonitor ergänzt die Überwachung der Ressourcen durch regelmäßige Abfragen des aktuellen Zustands. Eine Zustandsänderung, z.B. Ausfall eines Systems, generiert eine Ereignismeldung.

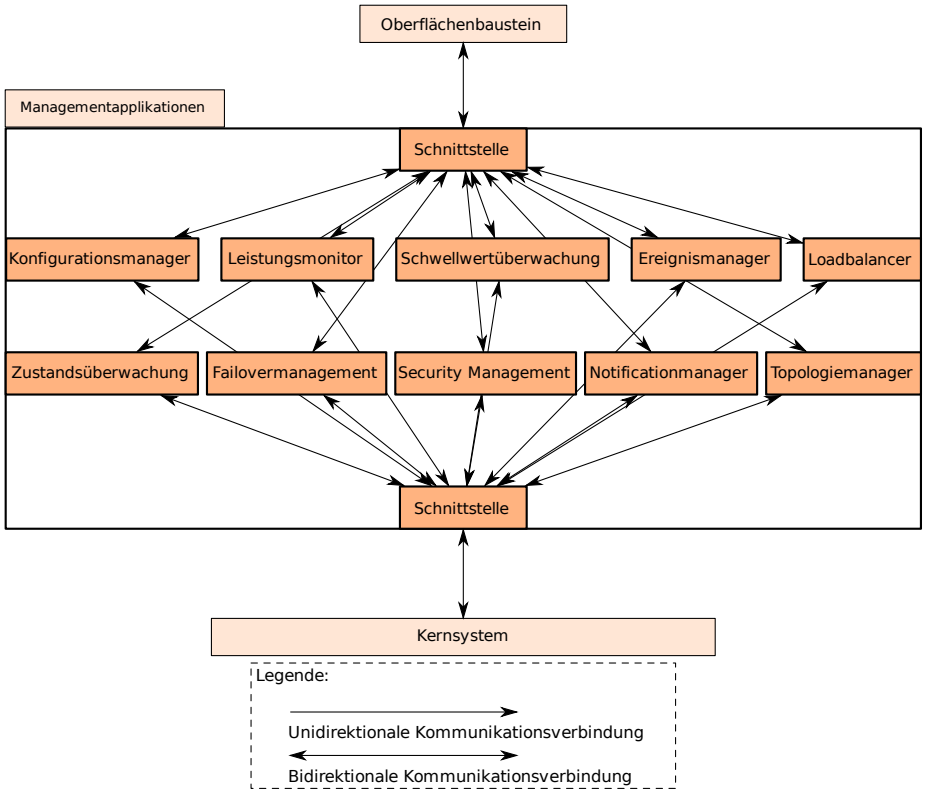

Abbildung 4.4: Die Managementapplikationen

Failovermanagement Das Failovermanagement reagiert auf Meldungen der Zustandsüberwachung und startet z.B. eine abgestürzte VM neu.

Security Management Das Security Management ist für sicherheitsrelevante Aufgaben wie z.B. Authentifizierung und Autorisierung der Benutzer und Ressourcen zuständig.

Notificationmanager Der Notificationmanager stellt verschiedene Benachrichtigungen bereit, z.B. per E-Mail oder durch einen Logfile-Eintrag.

Topologiemanagement Das Topologiemanagement ermöglicht ein weitgehend automatisiertes Auffinden von Ressourcen durch Autodiscovery, sowie eine daraus generierte Übersicht über die Topologie des Managementsystems.

Das Kernsystem als Informationsdispatcher

Das Kernsystem dient als Schaltzentrale zwischen den Bausteinen Managementapplikationen, Kommunikationsbaustein und Informationsverwaltung. Empfangene Managementinformationen leitet es an die entsprechenden Bausteine weiter. Im Rahmen der vorgestellten dynamischen Funktionalitäten einer Plattform (siehe 4.1.2) nimmt das Kernsystem eine zentrale Rolle ein. Anhand der übernommenen Funktionalität der Plattform, also ihrer Rolle, leitet es die empfangenen Daten an die dazugehörigen Zielbausteine weiter. Es übernimmt

also Aufgaben eines Informationsdispatchers und wird auch im folgenden so bezeichnet. Der Informationsdispatcher ist in Abbildung 4.5 grafisch dargestellt. Die Rollen des Informationsdispatchers werden im Rahmen des Organisationsmodell (Abschnitt 4.3.3) dargestellt.

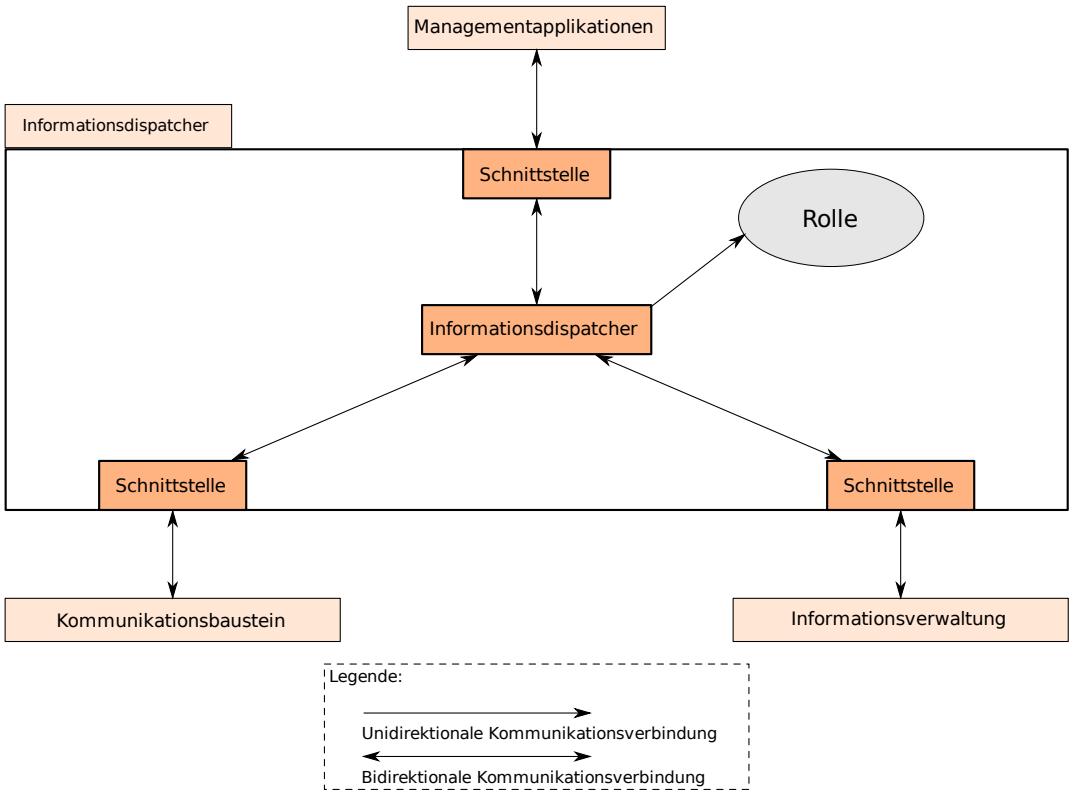

Abbildung 4.5: Der Informationsdispatcher

Die Informationsverwaltung

Die Speicherung und Verwaltung der Managementinformationen übernimmt die Informationsverwaltung (siehe Abbildung 4.6). Sie ist mit verschiedenen Backendsystemen verbunden, z.B. mit einer Datenbank zur Speicherung der Informationen und einem Verzeichnisdienst zur Authentifizierung der Benutzer. Um Flexibilität zu gewährleisten sind zwischen der Informationsverwaltung und den Backendsystemen Adapter eingebaut. Diese bilden eine Abstraktionsschicht und und ermöglichen somit eine einfache Austauschbarkeit der Backendsysteme. Mittels *Datenbankadapter* können verschiedene Datenbanken, wenn nötig auch gleichzeitig, an das System angebunden werden. Die *Verzeichnisdienstadapter* ermöglichen die Anbindung verschiedener Verzeichnisdienste. Die Informationsverwaltung enthält auch ein Modell der virtuellen Umgebungen, dieses wird im Informationsmodell (Abschnitt 4.3.1) eingehender dargestellt. Die Informationsverwaltung stellt den übrigen Infrastrukturbausteinen die vorgehaltenen Managementinformationen zur Verfügung.

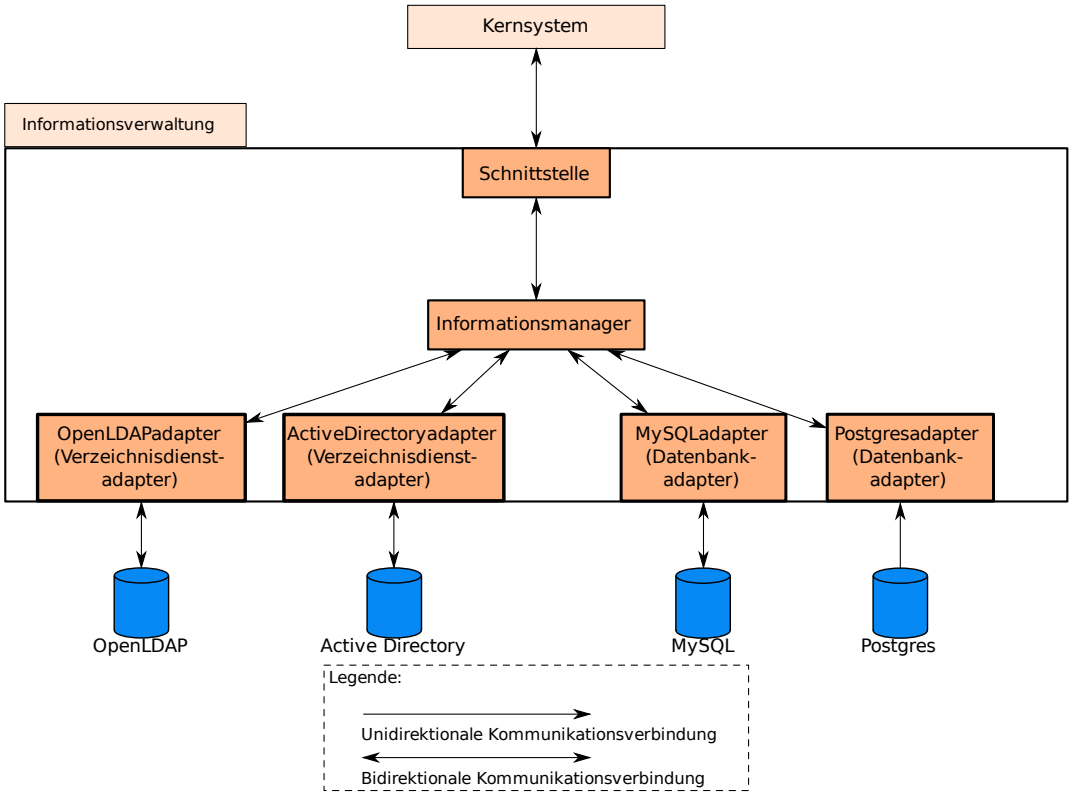

Abbildung 4.6: Anbindung diverser Backendkomponenten durch Adapter

Der Kommunikationsbaustein

Der Kommunikationsbaustein bildet die Schnittstelle zu den Ressourcen und anderen Managementplattformen. Mittels verschiedener Adapter werden die Ressourcen und Plattformen an die Managementplattform angebunden. Abbildung 4.7 zeigt den Kommunikationsbaustein mit integrierten *Ressourcenadaptern*, welche die Anbindung an Xen, Libvirt und VMware ESX vornehmen, sowie einem *Managementadapter*, welcher die Kommunikationsverbindung mit anderen Plattformen bereitstellt.

4.3 Modellierung der Managementteilmodelle

In diesem Abschnitt werden die Managementteilmodelle anhand der Informationen den vorangegangenen Abschnitten modelliert.

4.3.1 Informationsmodell

Das Informationsmodell liefert einen Beschreibungsrahmen für die Managementobjekte auf denen das Management operiert. Beim Management von Virtualisierungslösungen stehen dabei vor allem zwei Objekte im Vordergrund, der Host und die VM. Beide bestehen aus weiteren Komponenten welche für das Management von Bedeutung sind. Ein weiteres wichtiges Managementobjekt ist die Managementplattform selbst, auch sie wird in die Modellierung

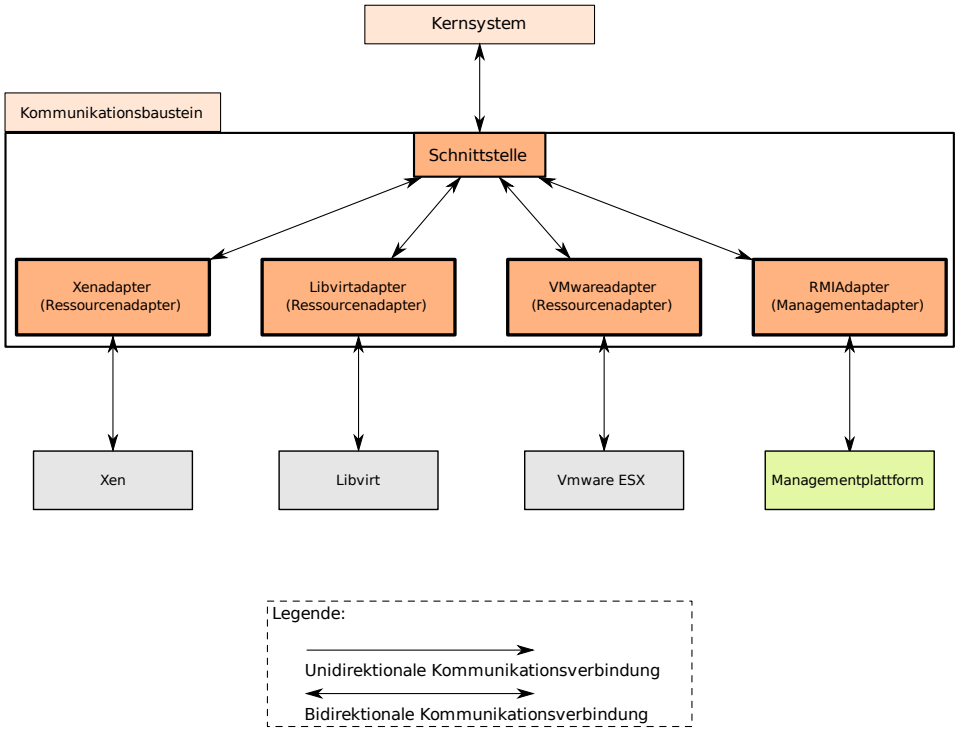

Abbildung 4.7: Kommunikationsadapter nehmen die Anbindung an verschiedene Ressourcen vor

mit einbezogen. Als Beschreibungssprache wird UML benutzt, da es verbreiteter Standard ist.

Benötigte Managementobjekte

Es existieren einige Beschreibungsrahmen für Computersysteme, z.B. CIM [cim08] und SID [sid08]. Vor allem CIM bietet hier eine durchdachte Struktur an brauchbaren Objekten an, jedoch ist diese sehr komplex. Außerdem ist nur ein kleiner Teil der dort definierten Klassen überhaupt für die benötigte MIB relevant. Die direkte Übernahme des CIM-Modells ist aus diesen Gründen nicht praktikabel.

Die vorhandenen Modelle der verschiedenen Virtualisierungslösungen dagegen sind besser an die Erfordernisse angepaßt, bieten jedoch nur eine produktspezifische Sicht auf die Managementobjekte.

Aus diesem Grund wurde ein naiver Ansatz mit diesen Modellen kombiniert. Der naive Ansatz besteht aus einer Betrachtung der Virtualisierungslösungen aus der Sicht eines Managementsystems. Dieses “sieht“erstmal nur Host und VMs. Im nächsten Schritt werden die einzelnen managementrelevanten Elemente sichtbar aus denen beide aufgebaut sind.

Host und VM bestehen aus weiteren, managementrelevanten Komponenten. Das sind Prozessor, Hauptspeicher, Festspeicher und Netzkarten. Durch diese Komponenten läßt sich die Last des Gesamtsystems beschreiben. Prozessor- und Speicherauslastung geben an, wie weit ein Host mit den darauf laufenden VMs ausgelastet ist. Der Festspeicher gibt an, ob noch

weitere VMs erzeugt werden können. Und die Netzkarten geben die aktuelle Netzlast, erzeugt der VMs, wider. Aus den Leistungsdaten dieser vier Komponenten kann man die Stabilität des Gesamtsystems erkennen und bei Überlastsituationen entsprechend agieren.

Zusammenfassung der benötigten Komponenten:

- Managementplattform
- Host
- VM
- Prozessor
- Hauptspeicher
- Netzkarten
- Festspeicher

Die Management Information Base

Anhand der erforderlichen Komponenten aus dem vorherigen Abschnitt, werden die benötigten Klassen definiert und deren Eigenschaften festgelegt.

Die zentralen Klassen Die zentralen Klassen stehen im Mittelpunkt der MIB. Alle anderen Klassen sind deren Subklassen und/oder mit diesen verbunden. Sie sind dargestellt in 4.8.

ManagedElement Das Wurzelelement ist *ManagedElement* und von ihm leiten sich alle anderen Elemente ab. Der Parameter *id* stellt einen systemweit eindeutigen Identifikator dar.

Managementplatform Die Managementplattform managt belibig viele Systeme. Sie besitzt einen übergeordneten Manager sowie belibig viele untergeordnete Agenten.

System Ein System stellt sowohl ein phyisches als auch ein virtuelles System dar, diese werden dann explizit durch die Vererbung auf die Elemente *PhysicalSystem* und *VirtualSystem* dargestellt. Ein System besitzt diverse beschreibende Parameter, u.a. *isHost*, welcher angibt ob das System als Host fungiert. Im Zusammenspiel mit der Assoziation *active_vms* sind damit geschachtelte virtuelle Umgebungen (VM-in-VM) möglich.

Ist das System ein Host kommen auch noch weitere Parameter ins Spiel. *virt_host_type* gibt den Typ des Virtualisierers an, z.B. Xen, *virt_host_version* ergänzt diese Information um die Version, z.B. 3.2 für Xen 3.2. *max_vms* gibt die maximale Anzahl an VMs an die auf dem Host gleichzeitig laufen dürfen. Der Parameter *reserved_ram* bezeichnet die Menge an Hauptspeicher in bytes, welche für spezielle VMs reserviert ist, also normalen VMs nicht zugewiesen werden darf. Die Attribute *min_vcpus* und *host_max_vcpus* geben an wieviele virtuelle CPUs ein Host maximal und minimal zur Verfügung stellen darf. Ähnlich die Attribute , *host_min_ram* und *host_max_ram*, sie geben entsprechend die minimalen und maximalen Werte für die VMs nutzbaren Hauptspeicher an. *migration_enabled* aktiviert die Migration für dieses System, ein Host kann bei deaktivierter

Migration keine VMs zu sich hin- oder wegmigrieren. Eine VM darf nicht verschoben werden.

Die Attribute *os_name* und *os_version* beschreiben das auf dem System laufende Betriebssystem, z.B. „Debian „und “5.0“für die Linuxdistribution Debian in Version 5.

Als Beschreibungsattribute dienen *name* und *description*, hier kann man für ein System beliebige Werte eintragen.

Verknüpft ist ein System mit einer beliebigen Anzahl an *Devices* und *Filesystemen*.

PhysicalSystem Ein physiches System hat als zusätzliches Attribut *location*, hier kann man dn aktuellen Standort der Maschine festlegen.

VirtualSystem Eine VM hat, als Gegenstück zu einem Host, die Attribute *virt_client_type* und *virt_client_version*, diese geben entsprechend den Typ und die Version des Virtualisierungstyps der VM an. *use_reserved_ram* gibt an ob die VM reservierten Hauptspeicher nutzen darf. *dynamic_cpu* gibt an ob die Anzahl der virtuellen CPUs der VM dynamisch ist. Die Anzahl der beim starten der VM zugewiesenen CPUs gibt *boot_cpu_count* an. dynamic_ram gibt an ob der Hauptspeicher einen statischen Wert, dann gilt *vm_min_ram*, annimmt oder eben dynamisch in den Grenzen *vm_min_ram* und *vm_max_ram* zugewiesen wird.

VMState Eine VM ist immer mit einem Zustand verknüpft. Folgende Zustände kommen dabei in Frage (siehe auch Zustandsdiagramm in Abbildung 4.14):

running Die VM läuft.

stopped Die VM ist ausgeschalten.

paused Die VM ist pausiert.

migrating Die VM migriert gerade.

suspended Die VM ist suspendiert.

Device Ein Device stellt ein physisches oder virtuelles Gerät dar, das Attribut *isVirtual* nimmt die Unterscheidung vor. *device* gibt das dazugehörige logische Gerät an, z.B. eth0 bei Linux für die erste Netzkarte.

LogicalFile Stellt eine abstrakte Datei, und damit auch ein Verzeichnis, dar. Das Attribut *name* gibt den Namen der Datei bzw. des Verzeichnisses an.

Filesystem Stellt ein Dateisystem dar. Das Attribut *size* gibt die Größe des zu Verfügung stehenden Speicherplatzes an. *mount-path* gibt den Ort der Einbindung in das System an, z.B. „/var“unter Linux oder „C:„unter Windows.

StatisticalData Die statistischen Informationen zu den einzelnen Objekten werden hier erfasst. *time* gibt den Zeitpunkt der Datenerfassung an.

QosConfig Die Schwellwerte werden in diesen Klassen Objekten zugeordnet.

Die Subklassen von Device Die Subklassen von Device stellen die einem System zuzuordnenden Geräte dar (Abbildung 4.9).

Cpu Cpu stellt einen physischen oder virtuellen Prozessor dar. Als beschreibendes Attribute dient *modell*, z.B. „AMD Athlon(tm) 64 X2 Dual Core Processor 4800+„. Die aktuelle Geschwindigkeit des Prozessors wird durch *speed* angegeben, *features* zeigt dessen technischen Möglichkeiten auf, um z.B. bei einer Migration feststellen zu können ob diese durchgeführt werden kann, da z.B. eine auf einem 64-Bit-System laufende VM nicht ohne weiteres auf ein 32-Bit-System verschoben werden kann.

VirtualCpu Eine virtuelle CPU. Die Assoziation *runningOn* gibt an, auf welchem Prozessor des Hostsystems dieser virtuelle Prozessor gerade läuft.

PhysicalCpu Eine physiche CPU.

Ram Stell den Hauptspeicher des Systems dar. *size* gibt dessen verfügbare Größe an.

NetworkInterface Stellt eine Netzanbindung dar. *speed* gibt die maximale Übertragungsgeschwindigkeit an, z.B. "100MBit". *ipv4* beinhaltet ein oder mehrere IPv4-Adressen im CIDR-Format. *ipv6* ist entsprechend das IPv6-Äquivalent. *mac* gibt die MAC-Adresse an.

VirtualNic Stellt ein virtuelles Netzinterface dar

PhysicalNic Stellt ein physisches Netzinterface dar.

Media Ein an das System angebundenes Medium, z.B. Festplatte, CDROM u.ä., ist eine Unterklasse von Media. Dieses besteht aus beliebig vielen Partitionen *size* gibt die Größe des verfügbaren Speicherplatzes an.

HardDisk Stellt eine Festplatte dar. Falls es eine virtuelle Festplatte ist , bezeichnet die Assoziation *contains virtual harddisk* die beinhaltende Image-Datei.

VirtualHardDisk Stellt eine virtualle Harddisk dar.

PhysicalHardDisk Stellt eine physische Harddisk dar.

StorageExtent Ein StorageExtent stellt eine Partition auf abstrakter Ebene dar. Es beinhaltet durch die Assoziation *has filesystem* höchstens ein Dateisystem. Durch die Assoziation *based on extent* kann eine belibig tiefe Schachtelung herbeigeführt und u.a. Raid und LVM beschrieben werden. *size* gibt die Größe des zu Verfügung stehenden Speicherplatzes an.

Die Subklassen von StorageExtent Die Subklassen von StorageExtent stellen verschiedene Arten von angebundenen Speichern dar (Abbildung 4.10).

LogicalDisk Stellt logische Partitionen dar, die auf anderen Partitionen angesiedelt sind, z.B. Raid über mehrere Partitionen.

DiskPartition Kann sowohl eine Partition direkt auf einer Festplatte (DiskPartition) oder auf einem anderen StorageExtent sein, z.B. eines Software-Raids.

Raid Stellt einen Raidverbund dar.

LVM Beschreibt ein LVM.

NetworkEnabledDevice Stellt eine netzwerkbasierte LogicalDisk dar. Die Assoziation *remoteServiceHostedOn* zeigt auf das System mit welchem eine Verbindung für diesen Dienst benötigt wird.

iScsi Stellt eine iSCSI-Disk dar.

FiberChannel Stellt eine Fiberchannel-Disk dar.

Die Subklassen von Filesystem und LogicalFile Die Subklassen von Filesystem und LogicalFile stellen die in einem System verfügbaren Dateisysteme sowie deren Inhalte dar (Abbildung 4.11).

RemoteFilesystem Stellt ein angebundenes entferntes Dateisystem dar.

NFS Das Network Filesystem wird durch die Serveradresse (*nfs_server*) und den Pfad des freigegebenen Verzeichnisses (*nfs_path*) dargestellt.

LocalFilesystem Stellt ein lokal auf dem System angesiedeltes Dateisystem dar. Ausprägungen davon können z.B. Ext3 oder NFS sein. Ein Filesystem liegt entweder auf einem StorageExtent oder in einem FilesystemImage, z.B. einem ISO-Image. Es besteht aus belibig vielen LogicalFiles.

Ext3 Stellt das Ext3-Dateisystem dar.

XFS Stellt das XFS-dateisystem dar.

Directory Stellt ein Verzeichnis dar und beinhaltet über die Assoziation *contains* belibig viele LogicalFiles.

DataFile Stellt alle Dateien außer Verzeichnis dar.

ImageFile Ein ImageFile beinhaltet ein Dateisystem oder ein Abbild einer Festplatte.

DiskImage Beinhaltet das Abbild einer Festplatte, z.B. ein VMware-Image.

VMwareImage Stellt ein VMware-Image dar.

FilesystemImage Beinhaltet das Abbild eines Dateisystems, z.B. ein ISO-Image.

IsoImage Stellt ein ISO-Image dar.

Die Subklassen von StatisticalData und QosConfig Die Subklassen von StatisticalData stellen die Monitoringdaten für die zu überwachenden Objekte dar. Die Subklassen von QosConfig beschreiben die Schwellwerte dieser Objekte (Abbildung 4.12).

NicData Beinhaltet die Messdaten einer Netzkarte. *in_kb* Gibt die Größe der empfangenen Daten in kbytes an, *out_kb* die der gesendeten.

RamData Beinhaltet die Messdaten des Hauptspeichers. *usage* gibt die gemessene Auslastung in kbytes an.

CpuData Beinhaltet die Messdaten einer Cpu. *usage* gibt die gemessene Auslastung in % an.

SystemData Beinhaltet die Messdaten eines Systems. *cpu_usage* gibt die gemessene Prozessorlast des Systems an.

StorageExtentData Beinhaltet die Messdaten eines StorageExtents. *read_kb* gibt die gemessene Auslastung bei Lesevorgängen in kbytes an, *write_kb* entsprechend bei Schreibvorgängen. *usage* gibt den verfügbaren Speicher in kbytes an.

FilesystemData Beinhaltet die Messdaten eines Dateisystems. *usage* gibt den verfügbaren Speicher in kbytes an.

MediaData Beinhaltet die Messdaten eines Mediums. *read_kb* und *write_kb* stellen die Leseraten bzw. die Schreibraten dar. *usage* beschreibt die Auslastung des Speichers.

QosNicConfig Beschreibt die Schwellwerte eines Netzinterfaces. *max_in_kb* und *min_in_kb* beschreiben die Schwellwertgrenzen der eingehenden Daten, *max_out_kb* und *max_in_kb* die der ausgehenden Daten.

QosRamConfig Beschreibt die Schwellwerte eines Ram-Objekts. *min_usage* und *max_usage* geben die Schwellwertgrenzen der Speicherauslastung an.

QosCpuConfig Beschreibt die Schwellwerte einer CPU. *min_usage* und *max_usage* geben die Schwellwertgrenzen der CPU-Auslastung an.

QosSystemConfig Beschreibt die Schwellwerte eines System-Objekts. *cpu_min_usage* und *cpu_max_usage* geben die Schwellwertgrenzen der Gesamt-CPU-Auslastung des Systems an.

QosStorageExtentConfig Beschreibt die Schwellwerte eines StorageExtents. *min_usage* und *max_usage* geben die Schwellwertgrenzen der Speicherplatzauslastung an.

QosFilesystemConfig Beschreibt die Schwellwerte eines Filesystems. *min_usage* und *max_usage* geben die Schwellwertgrenzen der Speicherplatzauslastung an.

QosMediaConfig Beschreibt die Schwellwerte eines Mediums. *min_usage* und *max_usage* geben die Schwellwertgrenzen der Speicherplatzauslastung an. *max_read_kb* und *min_read_kb* geben die Schwellwerte für den Lesezugriff an, *max_write_kb* und *min_write_kb* die für den Schreibzugriff.

Benutzer, Gruppen und ACLs Die folgenden Klassen stellen die Benutzer, Gruppen und Zugriffsrechte auf Objekte dar (Abbildung 4.13).

ManagingEntity Stellt eine managende Entität dar.

User Stellt einen Benutzer dar. Die Attribute *username*, *firstname*, *lastname* und *description* beschreiben den Benutzer.

Group Stellt eine Benutzergruppe dar. Die Attribute *name* und *description* beschreiben diese.

ACL Stellt ein Zugriffsrecht dar.

SystemACL Beschreibt das Zugriffsrecht auf ein System. *system_id* gibt an für welches System diese ACL gilt. *isSystemOwner* gibt an ob die damit verbundene ManagingEntity der Eigentümer des Systems ist. *canShutdown*, *canStart*, *canMigrate*, *canPause* und *canSuspend* geben die Steuerungsmöglichkeiten für das angegebene System an. *canEdit* gibt an ob die Gruppe bearbeitet werden darf. *canDelete* gibt das Recht zum löschen der Gruppe.

GroupACL Beschreibt das Zugriffsrecht auf eine Gruppe. *group_id* gibt an für welche Gruppe diese ACL gilt. *cannAddUsers* gibt das Recht Benutzer hinzuzufügen, *canRemoveUsers* erlaubt es Benutzer aus der Gruppe zu entfernen. *canEditUsers* erlaubt es die Rechte der Benutzer für diese Gruppe zu ändern. *isGroupOwner* gibt an ob die ManagingEntity der Eigentümer der Gruppe ist. *canCreateUsers* un *canCreateGroups* geben an ob Benutzer bzw. Gruppen innerhalb dieser Gruppe erstellt werden dürfen. *canEditGroups* und *canDeleteGroups* geben an ob Untergruppen bearbeitet oder gelöscht werden dürfen.

PlatformACL Beschreibt das Zugriffsrecht auf eine Plattform. *platform_id* gibt die Plattform an für die diese ACL gilt. *canAddResource* gibt an ob Ressourcen zur Plattform hinzugefügt werden dürfen. *canRemoveResource* und *canDeleteResource* geben an ob Ressourcen der Plattform bearbeitet oder entfernt werden dürfen.

GeneralACL Beschreibt allgemeine Zugriffsrechte. *canAddPlatform* gibt an ob Plattformen zum Managementsystem hinzugefügt werden dürfen. *isAdmin* gibt an ob die ManagingEntity als Administrator agieren darf, d.h. diese hat uneingeschränkte Zugriffsrechte.

4.3.2 Funktionsmodell

Im Rahmen des Funktionsmodells wird der Gesamtkomplex des Managements von Virtualiserungslösungen in verschiedene Funktionsbereiche gegliedert. Anhand der im Kapitel 2 definierten Managementfunktionen werden diesen Funktionen zugeordnet und anschließend deren Aufrufkonventionen festgelegt.

Die Gliederung in Funktionsbereiche wird nach FCAPS, siehe dazu [HAN99, S.75 ff], vorgenommen. Diese fünf Bereiche überdecken alle benötigten Managementfunktionen.

Konfigurationsmanagement

Das Konfigurationsmanagement ist zuständig für die Beschreibung der Konfiguration des Systems, den Vorgang des Konfigurierens, sowie das Ergebnis eines Konfigurationsvorgangs.

Benötigte Interaktionen

Setzen von Konfigurationsparametern Das Setzen von Konfigurationsparametern bei Managementobjekten ist eine zentrale Aufgabe des Konfigurationsmanagements. Mit jeder Änderung wird das Managementsystem in einen neuen Zustand überführt.

Abfragen der aktuellen Konfigurationsparameter Die aktuellen Konfigurationsparameter eines Managementobjekts müssen abgefragt werden können.

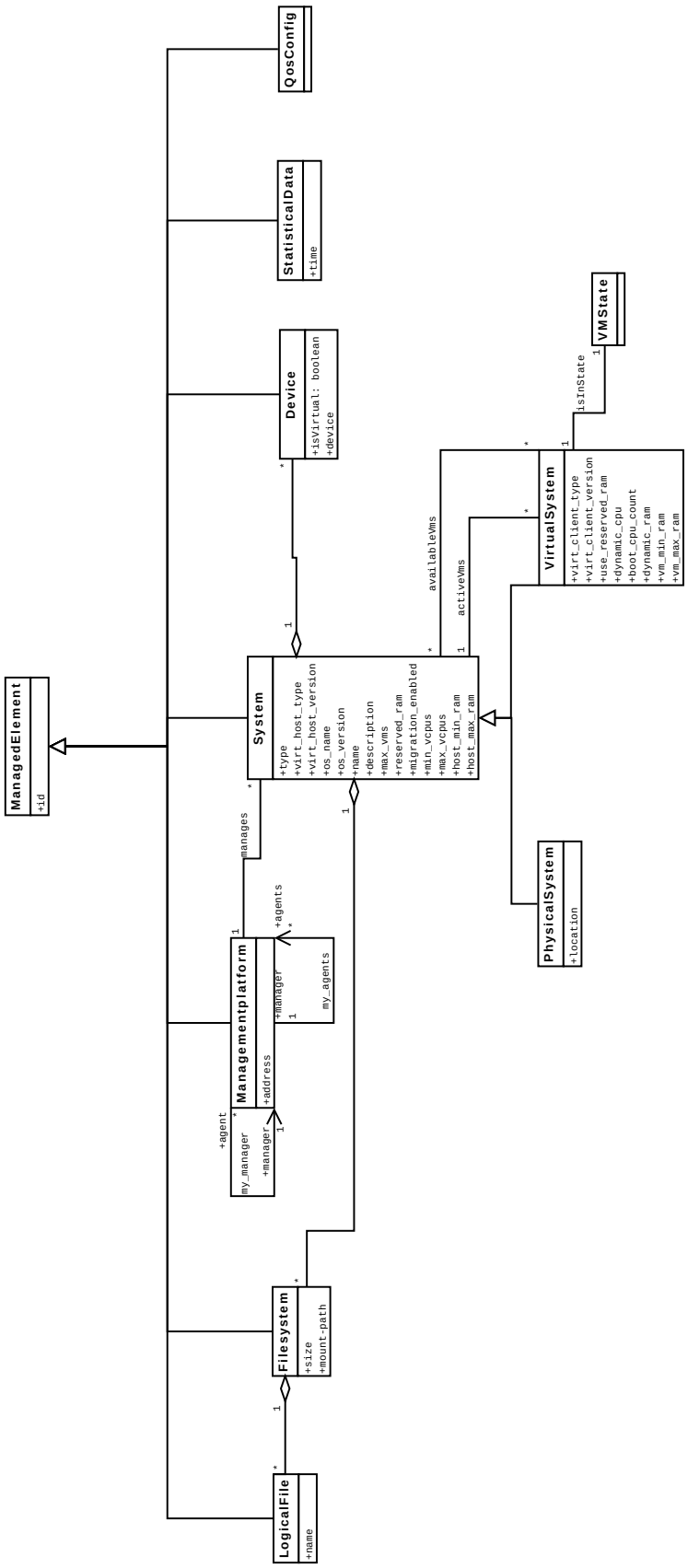

Abbildung 4.8: UML-Darstellung der MIB - Basisobjekte

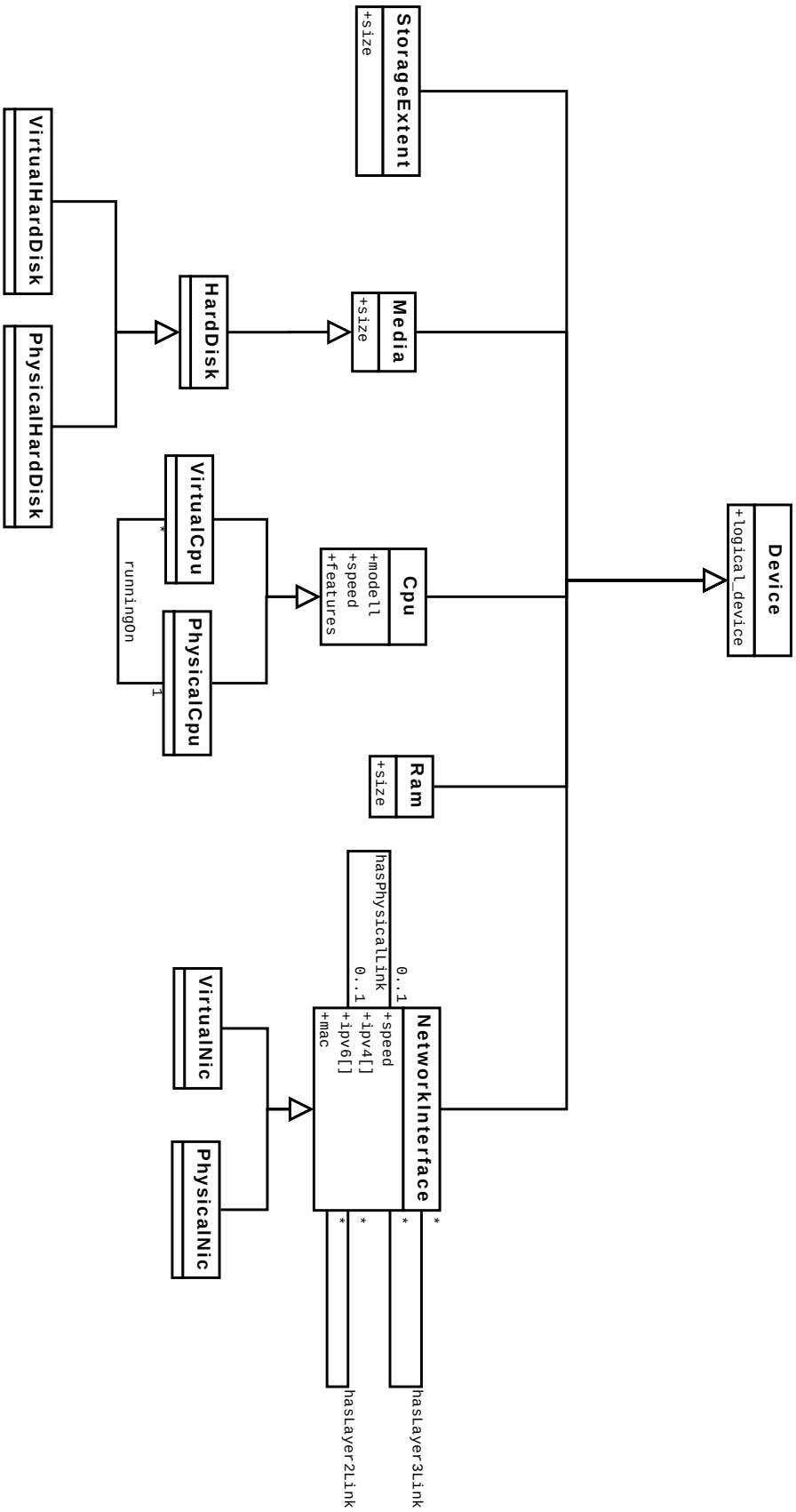

Abbildung 4.9: UML-Darstellung der MIB - Devices

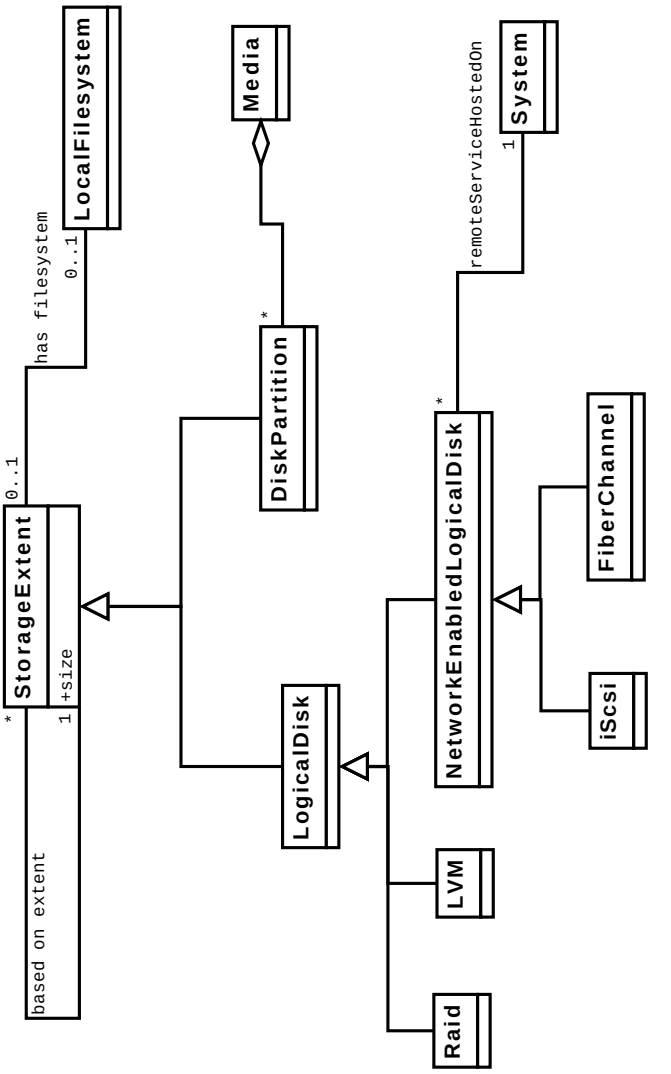

Abbildung 4.10: UML-Darstellung der MIB - StorageExtent

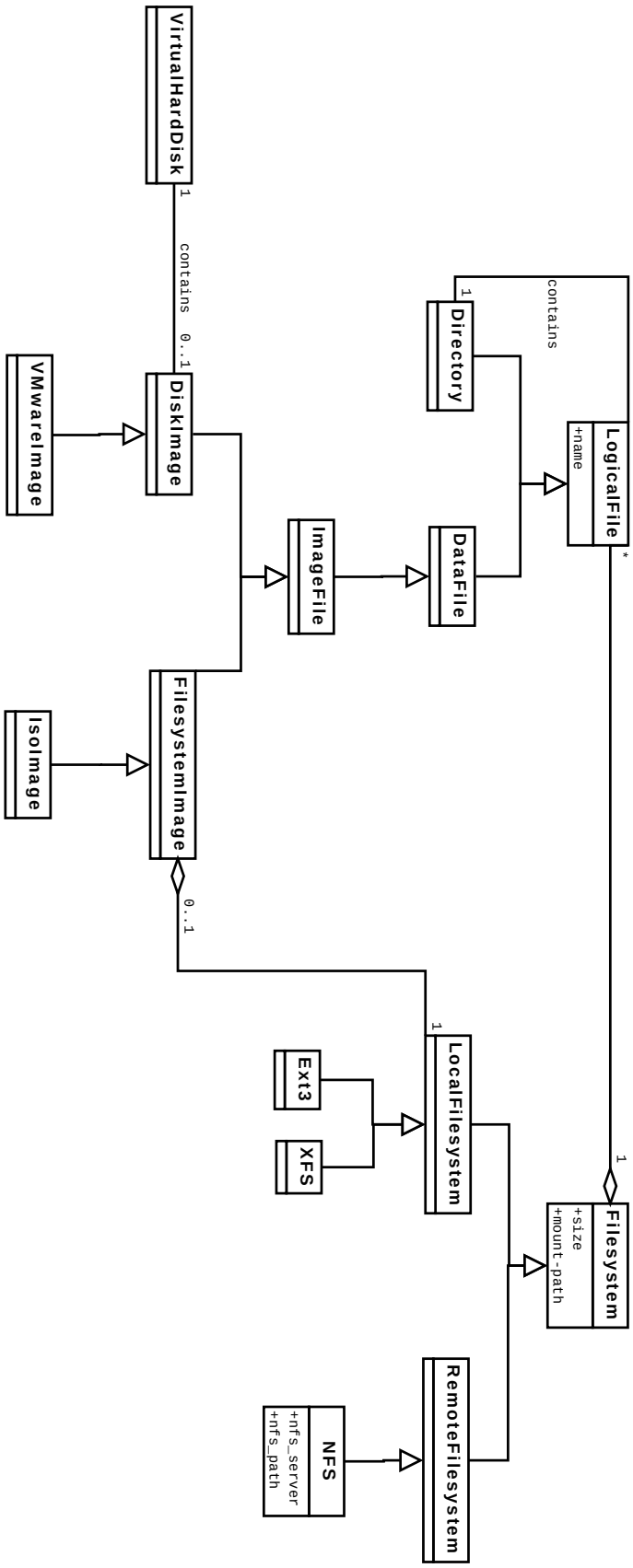

Abbildung 4.11: UML-Darstellung der MIB - Filesystem

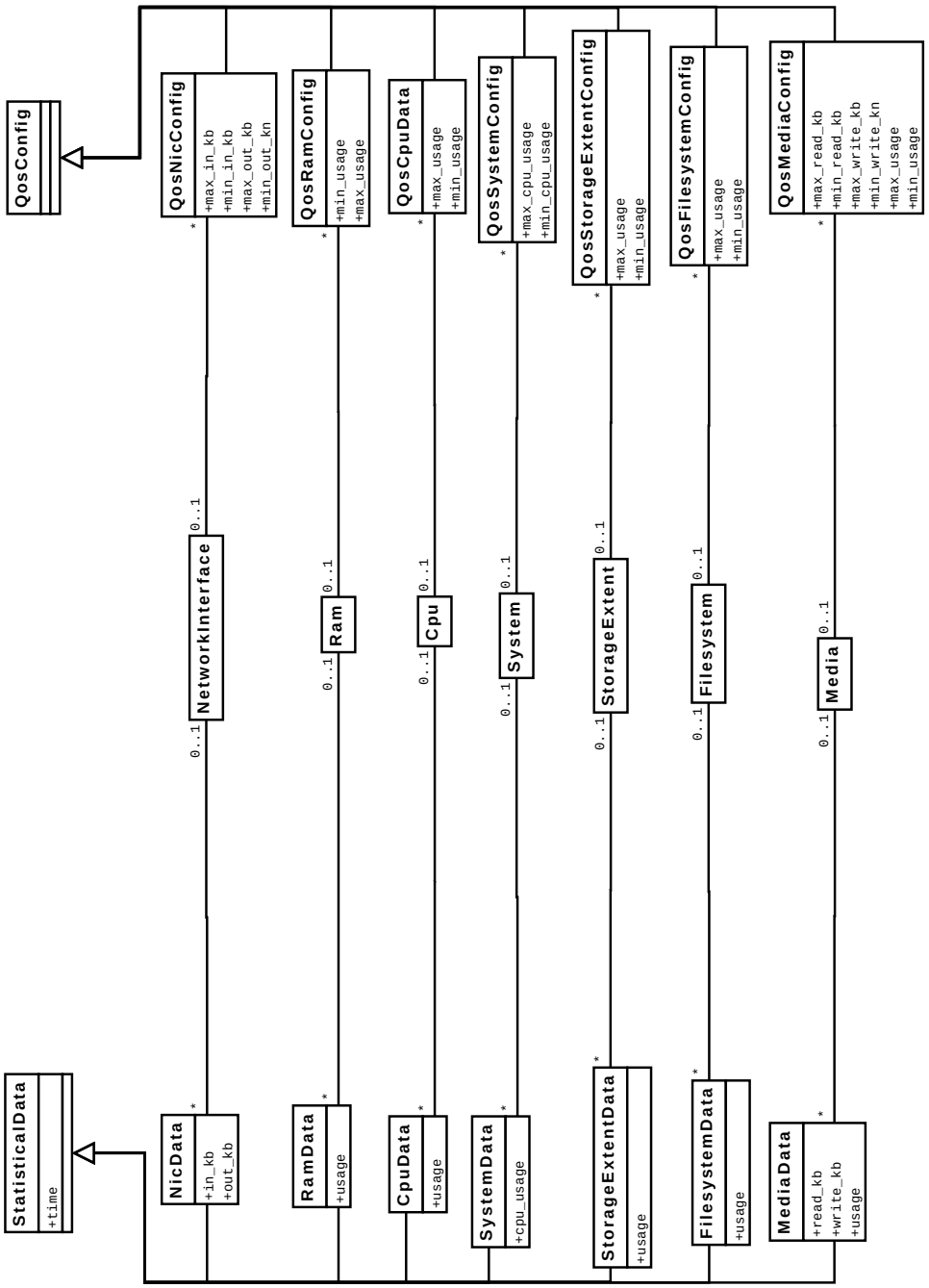

Abbildung 4.12: UML-Darstellung der MIB - StatisticalData und QosConfig

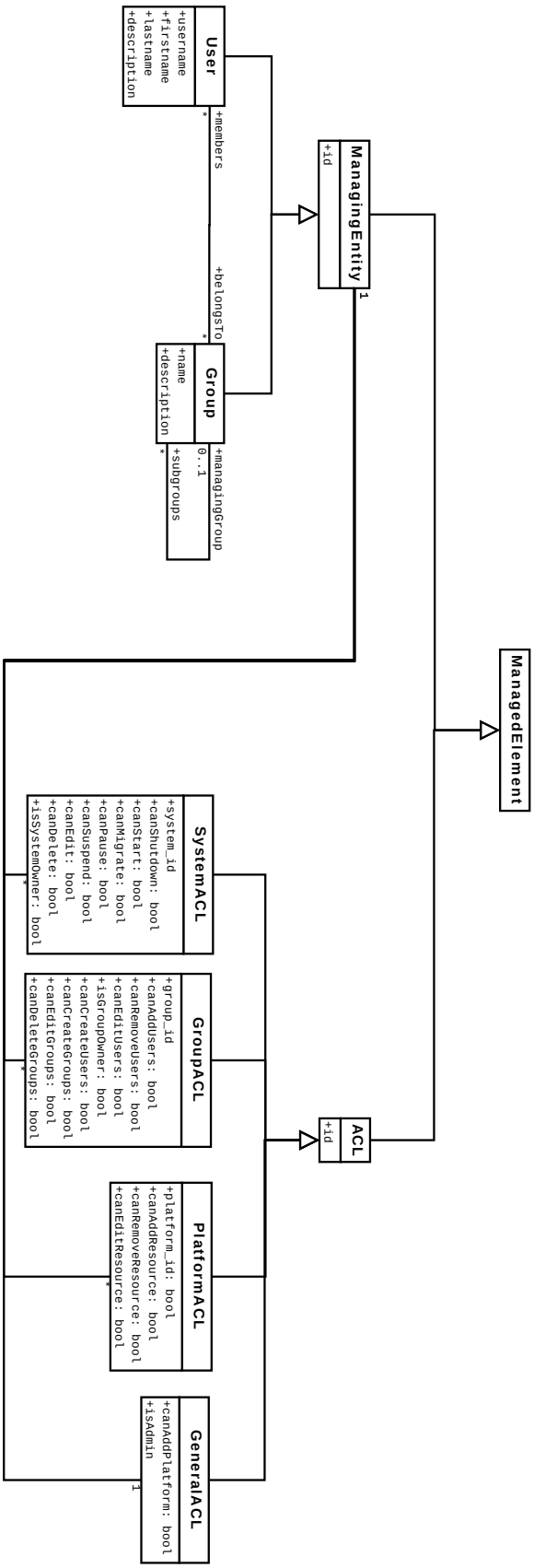

Abbildung 4.13: UML-Diagramm der ACLs

Abfrage der aktuellen Systemkonfiguration Die aktuelle Beschreibung des Systems muss abgefragt werden können. Dies beinhaltet die verfügbaren Managementobjekte und deren Beziehungen untereinander.

Hinzufügen/Entfernen von Managementobjekten Managementobjekte müssen dem System hinzugefügt bzw. von diesem entfernt werden können.

Migration einer VM Das Konfigurationsmanagement muss die Migration einer VM unterstützen.

Steuerung einer VM Die Steuerung der VMs, also das starten, stoppen, usw. muss möglich sein.

Verwalten einer VM-Vorlage Das Konfigurationsmanagement muss die Verwaltung von VM-Vorlagen, also erstellen, bearbeiten und löschen, unterstützen

Erstellen einer neuen VM Eine neue VM muss erstellt werden können, z.B. aus einer Vorlage, durch klonen oder manuell.

Konfiguration des Autodiscovery Das Autodiscovery muss konfiguriert werden können, das schließt vor allem die Definition des Suchbereichs sowie das Intervall der automatischen Suche mit ein.

Zuordnung der Managementobjekte Die Managementobjekte *Konfigurationsmanager* und *Topologiemanager* sind dem Konfigurationsmanagement zuzuordnen.

Konfigurationsmanager Der Konfigurationsmanager ist für alle Aufgaben des Konfigurationsmanagements zuständig, allein das Autodiscovery sowie die Generierung der Topologie obliegt dem Topologiemanager.

Topologiemanager Der Topologiemanager ist für das automatische Auffinden von Ressourcen, sowie für die Generierung der Topologie des Managementsystems zuständig.

Beschreibung der Funktionen

Setzen von Konfigurationsparametern Das Konfigurationsmanagement stellt die Methode *setConfig(elem_id, config)* bereit um die Konfiguration des durch *elem_id* angegebenen Objekts zu ändern. *config* ist eine Instanz derselben Klasse wie das angegebene Objekt und enthält die neue Konfiguration, die von diesem übernommen werden soll.

Abfragen der aktuellen Konfigurationsparameter Die Abfrage der aktuellen Konfiguration eines Managementobjekts übernimmt die Methode *getConfig(elem_id)*. Diese gibt das angegebene Objekt zurück.

Abfrage der aktuellen Systemkonfiguration Die Methode *getTopology()* des Topologiemanagers gibt die aktuelle Systemkonfiguration zurück. Darin enthalten sind alle verfügbaren Managementobjekte und deren Beziehungen.

Hinzufügen/Entfernen von Managementobjekten Der Konfigurationsmanager stellt die Methode add(objekt, elem_id) bereit um eine neue Instanz zum System hinzuzufügen. *elem_id* bezeichnet hierbei das Objekt zu dem das Neue hinzugefügt werden soll.

Migration einer VM Eine VM kann mittels einer der Methoden *migrateOnline(vm_id, target_host)*, *migrateOffline(vm_id, target_host)* und *migrateStorage(vm_id, target_dir)* verschoben werden. *vm_id* definiert dabei die VM, *target_host* den Zielhost und *target_dir* das Zielverzeichnis der entsprechenden Migration.

Steuerung einer VM Die Methoden *vmStart(vm_id)*, *vmStop(vm_id)*, *vmPause(vm_id)* und *vmSuspend(vm_id)* dienen der Steuerung der angegebenen VM. Das Zustandsdiagramm einer VM ist in Abbildung 4.14 dargestellt.

Verwalten der VM-Vorlagen Zur Verwaltung der VM-Vorlagen stehen die Methoden *getTemplates()* zur Auflistung aller verfügbaren Vorlagen, *getTemplate(template_id)* zur Anforderung einer Vorlage, sowie *setTemplate(template_id, newTemplate)* zur Änderung einer Vorlage zu Verfügung. *newTemplate* ist ein neues Templateobjekt, deren Eigenschaften vom alten Objekt übernommen werden. Das Erstellen eines neuen Templates geschieht mittels *addTemplate(template)*, *template* beinhaltet ein neues Vorlagenobjekt. Ein Vorlagenobjekt ist die Instanz eines *VirtualSystem*, jedoch mit Attribut *isTemplate=true*, sowie die dazugehörigen Komponenten.

Erstellen einer neuen VM Eine neue VM kann mittels verschiedener Methoden erstellt werden. *createVmFromTemplate()* erstellt eine VM aus einer Vorlage. *createVmFromClone(vm_id)* erstellt eine VM aus einer exisiterenden mit Id *vm_id* mittels klonen. *createVm()* erstellt eine neue unkonfigurierte VM.

Konfiguration des Autodiscovery Das Autodiscovery kann über die Methode *setIntervall(secs)* zeitlich gesteuert werden. secs gibt den Abstand zwischen zwei Suchläufen in Sekunden an. *getIntervall()* liest diesen Wert aus und liefert ihn als Rückgabewert.

Fehlermanagement

Das Fehlermanagement hat die Aufgabe des Entdeckens, Eingrenzens und Behebens von abnormalen Systemzuständen.

Benötigte Interaktionen Die folgenden Interaktionen mit dem Fehlermanagement werden benötigt:

Aktivieren und deaktivieren der Zustandsüberwachung für Ressourcen Die Zustandsüberwachung der Ressourcen muss aktiviert und deaktiviert werden können. Dies ist vor allem in Wartungszeiträumen sinnvoll, in denen Systeme temporär gestoppt werden müssen.

Definition eines Fehlerzustandes Um einen Fehlerzustand festzustellen muss definiert sein, wann ein Fehlerzustand vorliegt. Die Dauer eines abnormalen Systemzustands bis zum Auslösen eines Fehlerzustandes muss vom Benutzer konfigurierbar sein.

Reaktion auf Fehlerzustände Ist ein Fehlerzustand eingetreten muss das System sinnvoll darauf reagieren, für jede Ressource müssen entsprechende Reaktionen konfigurierbar sein.

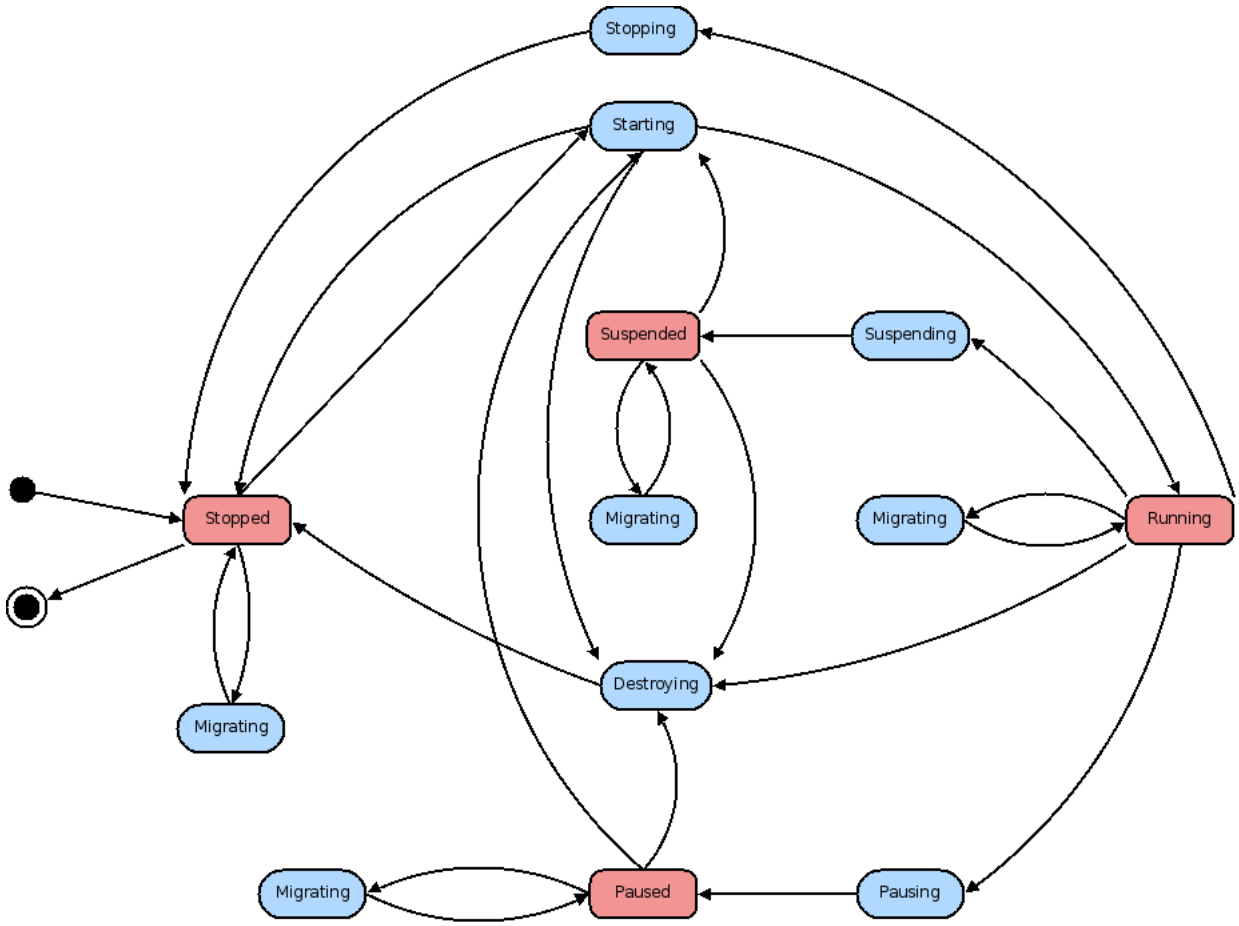

Abbildung 4.14: Zustandsdiagramm einer VM

Zuordnung der Managementobjekte Zum Fehlermanagement gehören die Managementobjekte Zustandsüberwachung und Failover.

Beschreibung der Funktionen

Aktivieren und deaktivieren der Zustandsüberwachung für Ressourcen Das Fehlermanagement stellt die Methode *activateStateCheck(system_id,bool)* zur Aktivierung (bool=true) und Deaktivierung (bool=false) der Zustandsüberwachung der mit *system_id* angegebenen Ressource zu Verfügung.

Definition eines Fehlerzustandes Über die Methode *setTimeout(secs)* lässt sich die Dauer in Sekunden einstellen, nach der ein abnormaler Systemzustand in einen Fehlerzustand übergeht.

Reaktion auf Fehlerzustände Die Reaktion auf einen Fehlerzustand lässt sich über die Methode *onFail(system_id,reaction)* einstellen. system_id gibt das System an für das diese Konfiguration gilt. Der Parameter *reaction* stellt dabei eine der folgenden Möglichkeiten dar:

- *1* bedeutet VM auf anderem Host neu starten

- *2* bedeutet VM auf selbem Host neu starten

Leistungsmanagement

Das Leistungsmanagement dient der Leistungsüberwachung der Ressourcen. Hier werden Dienstgüten festgelegt und die Ressourcen daraufhin überwacht.

Benötigte Interaktionen Die folgenden Interaktionen mit dem Leistungsmanagement werden benötigt:

Festlegen von Abfrageparametern zu Managementobjekten Um eine Leistungsüberwachung durchzuführen muss das Leistungsmanagement das Festlegen relevanter Parameter ermöglichen, z.B. das Abfrageintervall für eine Ressource.

Aktivieren und Deaktivieren der Leistungsüberwachung für ein MO Für einzelne Ressourcen muss die Überwachung aktiviert und deaktiviert werden können, dies ist Beispielsweise für Wartungsfälle relevant, in denen die Systemlast generell ansteigt.

Festlegen von Dienstgüten zu Managementobjekten Das Leistungsmanagement muss die Möglichkeit bieten den Managementobjekten Dienstgüten zuzuordnen, welche mit den Leistungsdaten verglichen werden können um Leistungsengpässe erkennen zu können.

Abfrage der festgelegten Dienstgüten Für die Benutzer des Systems muss die Möglichkeit bestehen die festgelegten Dienstgüten abzufragen.

Löschen der Dienstgüten eines Objekts Die einem MO zugeordneten Dienstgüten müssen gelöscht werden können.

Benachrichtigung über eingehende Leistungsdaten Um Leistungsengpässe zu erkennen muss das Leistungsmanagement auf eingehende Leistungsdaten reagieren und diese mit den festgelegten Dienstgüten abgleichen.

Konfiguration der Reaktion auf Verletzung der Dienstgüten Bei Verletzung von Dienstgüten muss festgelegt werden wie das Leistungsmanagement darauf reagieren soll.

Zuordnung der Managementobjekte Dem Leistungsmanagement sind die Managementobjekte *Leistungsüberwachung*, *Schwellwertüberwachung* und *Loadbalancer* zuzuordnen.

Leistungsüberwachung Die Leistungsüberwachung übernimmt die Messung der Leistungsparameter der MOs.

Schwellwertüberwachung Die Schwellwertüberwachung prüft die Leistungsdaten gegen die definierten Dienstgüten.

Loadbalancer Der Loadbalancer reagiert auf Verletzungen der Dienstgüte und stellt einen stabilen Systemzustand wieder her.

Beschreibung der Funktionen Um die dargestellten Interaktionen zu ermöglichen benötigt das Leistungsmanagement entsprechende Funktionen. Diese werden im Folgenden anhand der Interaktionen beschrieben.

Festlegen von Abfrageparametern zu Managementobjekten Die Leistungsüberwachung stellt die Methode *setQuery(elem_id, intervall)* zur Konfiguration der Leistungsüberwachung eines Mos zur Verfügung. Der Parameter *elem_id* gibt an, um welches Element es sich handelt. Das Element kann jedes Element im Rahmen der MIB sein, zu dem eine Assoziation zu einem StatisticalData-Objekt existiert. *intervall* gibt die Zeit eines Überwachungsintervalls in Sekunden an.

Aktivieren und Deaktivieren der Leistungsüberwachung für ein MO Die Leistungsüberwachung stellt die Methode *setQueryEnabled(bool)* zur Aktivierung (Wert true) bzw. Deaktivierung (Wert false) der Leistungsüberwachung eines Elements zur Verfügung. Wird die Überwachung eines Systems aktiviert/deaktiviert, dann wird auch die Überwachung aller Komponenten aktiviert/deaktiviert, aus dem dieses besteht.

Festlegen von Dienstgüten zu Managementobjekten Die Schwellwertüberwachung stellt die Methode *setQos(elem_id, QosConfig)* bereit, wobei elem_id für das gewünschte Element und die Klasse QosConfig für eine Dienstgütekonfiguration laut MIB steht. Die Attribute einer Qos-Instanz müssen entsprechend des gewählten Elements gesetzt werden.

Abfrage der festgelegten Dienstgüten Die Abfrage der Dienstgüte eines Elements stellt die Schwellwertüberwachung über die Methode *getQos(elem_id)* bereit. *elem_id* bezeichnet die id des gewünschten Elements.

Löschen der Dienstgüten eines Objekts Die Schwellwertüberwachung stellt die Methode *removeQos(elem_id)* bereit um für das Objekt mit Id *elem_id* die Dienstgütekonfiguration zu entfernen.

Benachrichtigung über eingehende Leistungsdaten Um eine Verletzung der Dientgüte eines Objekts feststellen zu können muss die Schwellwertüberwachung informiert werden wenn neue Messdaten zur Verfügung stehen. Dazu nimmt die Methode *notify(elem_id)* die Id des Objekts entgegen für das neue Messdaten eingegangen sind.

Reaktion auf Verletzung der Dienstgüten Liegt eine Verletzung der Dienstgüte vor muss der Loadbalancer benachrichtigt werden. Dazu stellt er die Methode *notify(elem_id)* zur Verfügung. *elem_id* bezeichnet die Id des Elements welches die Dienstgüteverletzung verursacht.

Abrechnungsmanagement und Benutzerverwaltung

Das Verwalten der Benutzer und Gruppen sowie die Zuordnung der Ressourcen zu diesen ist Teil des Accounting.

Benötigte Interaktionen Die folgenden Interaktionen werden benötigt:

Verwalten der Benutzer und Gruppen Die Verwaltung der Benutzer und Gruppen schließt das Hinzufügen, Ändern und Entfernen von Benutzer und Gruppen , sowie die Zuordnung von Benutzern zu Gruppen, ein.

Zuordnung der Ressourcen Zurodnen von Ressourcen zu Benutzer und/oder Gruppen.

Zuordnung der Managementobjekte Die Funktionalität zur Verwaltung der Benutzer und Gruppen sowie die Autorisierung dieser obliegt dem *SecurityManager*. Der *AccessManager*, als Teil des Oberflächenbausteins, überprüft jeden Zugriff auf das System anhand des SecurityManagers. Erst bei Freigabe durch diesen können die weiteren Managementapplikationen genutzt werden. Die Autorisierung der Benutzer und Gruppen erfolgt anhand deren zugeordneter ACLs. Eine Übersicht über die verfügbaren ACLs ist in Abbildung 4.13 dargestellt.

Beschreibung der Funktionen Die folgenden Funktionen werden zur Umsetzung der Interaktionen benötigt:

Verwalten der Benutzer und Gruppen Zur Verwaltung der Benutzer stellt der SecurityManager verschiedene Methoden bereit. *getUserlist()* gibt die verfügbaren Benutzer als User-Objekte zurück. *getUser(user_id)* fordert das Benutzerobjekt mit der angegebenen *user_id* an. *addUser(user)* fügt das übergebene Benutzerobjekt dem System hinzu. *deleteUser(user_id)* löscht den angegebenen Benutzer. *changeUser(user)* führt Änderungen an einem Benutzer durch, *user* enthält das Benutzerobjekt mit den Änderungen. Die Verwaltung der Gruppen erfolgt analog zu den Benutzern über die Methoden *getGrouplist()*, *getGroup(group_id)*, *addGroup(group)*, *deleteGroup(group_id)* und *changeGroup(group)*. Dazu kommen Methoden zum Verwalten der Mitglieder. *addUserToGroup(user_id, group_id)* fügt den angegebenen Benutzer der angegebenen Gruppe hinzu. removeUserFromGroup(user_id, group_id) entfernt einen Benutzer aus einer Gruppe.

Zuordnung der Ressourcen Die Zuordnung der Ressourcen erfolgt über die Methode *setOwner(id, system_id)*, sie legt die ManagingEntity mit Id *id* fest zu der das System mit Id *system_id* organisatorisch gehört.

Sicherheitsmanagement

Die Funktionalität der Zuordnung von Berechtigungen sowie die Autorisierung der Benutzer obliegt dem *SecurityManager*. Der *AccessManager*, als Teil des Oberflächenbausteins, überprüft jeden Zugriff auf das System anhand des SecurityManagers. Erst bei Freigabe durch diesen können die weiteren Managementapplikationen genutzt werden. Die Autorisierung der Benutzer und Gruppen erfolgt anhand deren zugeordneter ACLs. Eine Übersicht über die verfügbaren ACLs ist im Informationsmodell in Abbildung 4.13 dargestellt.

Benötigte Interaktionen

Zuordnung zu Berechtigungen auf Ressourcen von Benutzern und Gruppen Im Rahmen der Autorisierung wird eine Zuordnung der Berechtigungen auf Ressourcen zu Benutzern und Gruppen benötigt.

Zuordnung der Managementobjekte Der *SecurityManager* übernimmt diese Aufgaben.

Beschreibung der Funktionen Die folgenden Funktionen werden zur Umsetzung der Interaktionen benötigt:

Zuordnung zu Berechtigungen auf Ressourcen von Benutzern und Gruppen Die Zuordnung der ACL-Objekte zu den ManagingEntities übernehmen die Methoden *addAcl(id, acl)*, *removeAcl(id,acl_id)* und *changeAcl(id,acl)*. *id* bezeichnet in allen Fällen die ManagingEntity, also den Benutzer oder die Gruppe. *acl* ist ein ACL-Objekt und *acl_id* die Id desselbigen. Die Methoden *getSystemAclList(id)*, *getGeneralAcl(id)*, *getPlatformAclList(id)* und *getGroupAclList(id)* geben zur ManagingEntity mit Id *id* die entsprechende ACL-Objekte zurück. Einzelne Objekte können mit *getSystemAcl(id,system_id)*, *getPlatformAcl(id, platform_id)* und *getGroupAcl(id, group_id)* gesucht werden.

4.3.3 Organisationsmodell

Das Organisationsmodell beschreibt die topologische und funktionelle Anordnung des Managementsystems sowie die Kooperationsform der beteiligten Akteure.

Topologische Anordnung Als topologische Anordnung für das Managementsystem wurde das hierarchische Modell gewählt. Somit steht eine Managementplattform an der Spitze der Hierarchie und managt alle weiteren Managementplattformen und Ressourcen, siehe Abbildung 4.15. Gleichzeitig kann die Kontrolle über Teilbäume teilweise abgegeben werden um Mandantenfähigkeit zu erreichen. Dieses hierarchische Modell entspricht u.a. der am Lehrstuhl vorherrschenden Organisationsform beim Management der virtuallen Systeme.

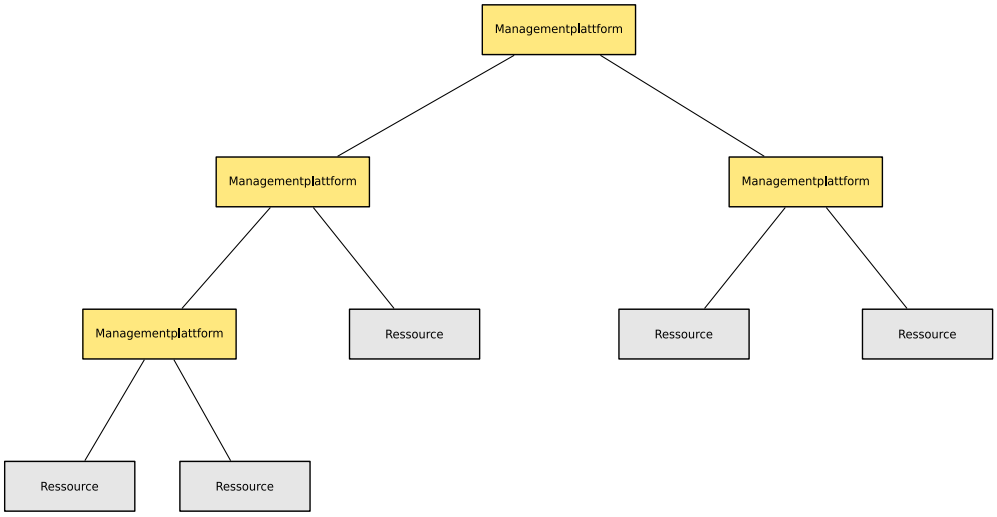

Abbildung 4.15: Beispiel einer Hierarchie aus Managementplattformen und Ressourcen

Rollen und Kooperationsform Als Kooperationsform wurde das Manager-Agent Modell gewählt. Der Manager fordert bei diesem Modell Informationen vom Agenten an oder weist diesen an eine bestimmte Operation auszuführen. Die Managementplattform ist hier Manager, Agent oder beides, je nachdem auf welcher Ebene in der Hierarchie sie sich befindet. Im Normalfall ist eine Managementplattform jedoch immer ein Manager, da sie entweder eine

Managementplattform oder eine Ressource unter sich hat. Alle Plattformen unter der Obersten sind auch immer Agenten, ebenso alle Ressourcen. Dies ist in Abbildung 4.16 grafisch dargestellt.

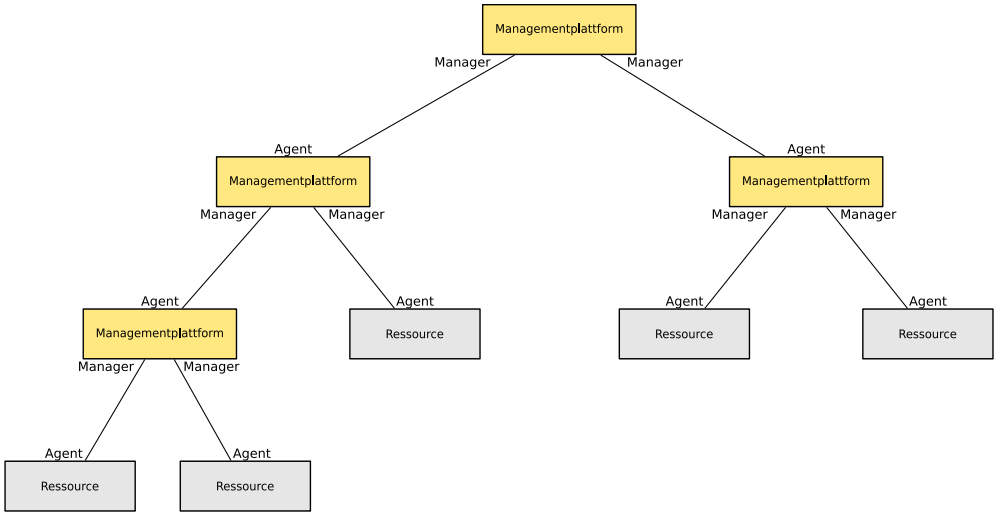

Abbildung 4.16: Beispielhierarchie mit Rollenverteilung Manager-Agent

Domänenmodell Aus organisatorischer Sicht können Ressourcen zu eigenständig verwalteten Einheiten zusammengefaßt werden. Aus diesem Grund wird ein einfaches Domänenmodell unterstützt. Eine Domäne ist ein Teilbaum mit einem Domänenmaster und beliebig vielen Domänenmitgliedern. Ein Domänenmaster kan wiederum ein Mitglied sein, eine Schachtelung der Domänen ist damit möglich. Aus technischer Sicht sind Domänenmaster Nachrichtengrenzen. Nachrichten einer Domäne gelangen nicht über deren Grenzen hinaus. Jedoch kann ein Domänenmaster Daten eins untergeordneten Domänenmasters abfragen, z.B. zu Abrechnungszwecken.

Abbildung 4.17 zeigt ein Beispiel anhand des Lehrstuhls. Die Domäne *nm.ifi.lmu.de* stellt die übergeordnete Domäne des Lehrstuhls dar. Dieser untergeordnet ist einmal die Domäne *StudArbeiten*, welche Projekte im Rahmen von Fopras und Diplomarbeiten beinhaltet. Diese Domäne besteht aus mehreren Managementplattformen, zwei die direkt in Kontakt mit den Ressourcen stehen, sowie der Domänenmaster, welcher die Daten beider untergeordneter Plattformen sammelt und diese auch steuert.

Die weitere untergeordnete Domäne ist *SecP*, welche die virtuelle Infrastruktur des IT-Sicherheitspraktikums beinhaltet.

Funktionelle Anordnung Die funktionelle Anordnung der Managementplattformen ist dynamisch. Die Funktionalitäten einer Plattform können, wie in Abschnitt 4.1.2 vorgeschlagen, verändert werden. Abbildung 4.18 zeigt eine Beispielhierarchie mit verschiedenen Rollen und Zugriffsmöglichkeiten auf verschiedenen Ebenen. Im folgenden werden diese Rollen definiert und beschrieben. Die Reihenfolge der folgenden Beschreibungen orientiert sich am zunehmenden Funktionsumfang der Rollen.

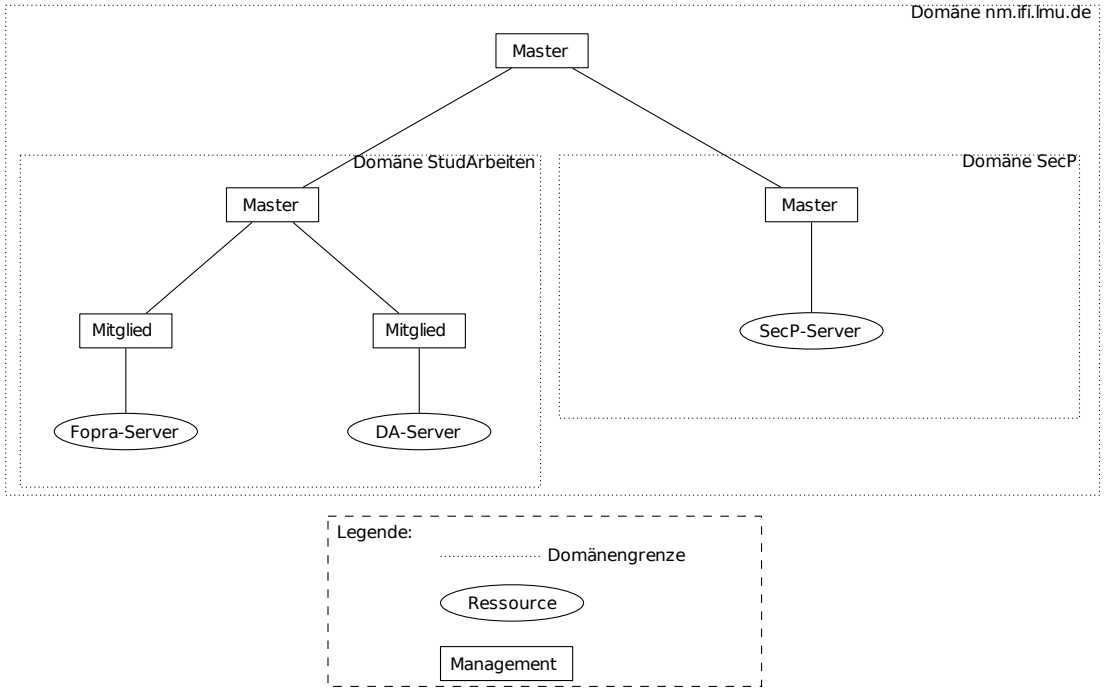

Abbildung 4.17: Beispiel einer Domänenhierarchie

Proxy Die Proxy-Rolle beschreibt einen Manager mit minimaler Funktionalität. Es sind nur diejenigen Elemente aktiv welche für die Interaktion mit Ressourcen und untergeordneten Managern benötigt werden (siehe Abbildung 4.19). Insgesamt sieht das Modell zwei Kommunikationsbeziehungen vor, zwischen den Ressourcen und dem übergeordneten Manager, sowie zwischen untergeordneten und übergeordneten Managern. Im ersten Fall schickt der hierarchisch darüberliegende Manager Steuerbefehle an den Proxy, wobei diese an den Proxy selbst oder an die angebundenen Ressourcen gerichtet sein können. In die andere Richtung werden Managementinformationen und Events an diesen übertragen. Die zweite Kommunikationsbeziehung beschränkt sich auf ein Weiterleiten der eingehenden Daten an den entsprechenden Empfänger. Dies ist von den hierarchisch tiefergelegenen Managern zum höherstehenden Manager (Managementinformationen und Events) als auch umgekehrt (Steuerinformationen) möglich. Aktive Managementapplikationen sind der Zustandsmonitor zur Überwachung der Ressourcen, der Ereignismanager um Ereignisse verarbeiten zu können, sowie der Leistungsmonitor um Messungen auf den Ressourcen durchzuführen. Die Informationsverwaltung beinhaltet in dieser Konfiguration nur allgemeine Informationen über die angebundenen Ressourcen und Manager, z.B. Identifikation und Adressen, eine Anbindung an externe Datenquellen wird nicht benötigt. Weitergehende Daten sind in einem hierarchisch höhergelegenen Manager gespeichert.

Caching-Proxy Der Caching-Proxy stellt eine funktionelle Erweiterung des Proxy dar. Eingehende Daten werden nicht unbedingt sofort an den Empfänger weitergeleitet, sondern

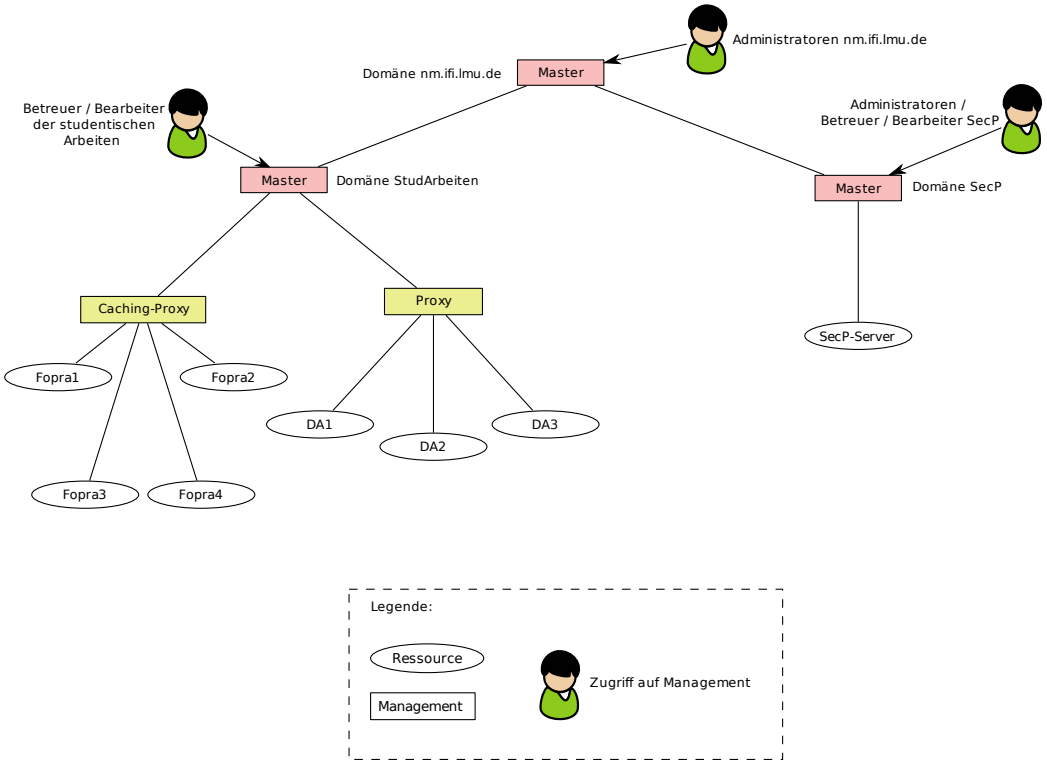

Abbildung 4.18: Beispielhierarchie mit verschiedenen Rollen und Zugriffsmöglichkeiten

können zwischengespeichert werden. Zwischengespeicherte Daten müssen jedoch verarbeitet werden um kritische Zustände in den Managementinformationen zu erkennen. Aus diesem Grund wird als weitere Managementapplikation eine Schwellwertüberwachung benötigt, wie in Abbildung 4.20 dargestellt. Kritische Daten müssen sofort an den übergeordneten Manager weitergeleitet werden, evtl. mit weiteren Daten aus dem Zwischenspeicher.

Master Der Master enthält ein eigenes Datenmodell der angebundenen Ressourcen und untergeordneten Manager. Er bildet die Wurzel eines eigenständig verwalteten Teilbaums. An die übergeordnete Instanz werden, wenn vorhanden, nur rudimentäre Daten, z.B. Abrechnungsinformationen, übertragen. Der Master besitzt vollständige Kontrolle über die von ihm verwalteten Objekte. Über die aktivierte Benutzerschnittstelle kann der Master separat verwaltet werden. Die Authentifizierung und Autorisierung erfolgt über einen angebundenen Verzeichnisdienst. Die Managementinformationen werden in der angebundenen Datenbank gespeichert. Der Master kann die Rollen seiner untergeordneten Objekte selbst festlegen, auch ein untergeordneter Master ist möglich. Die grafische Darstellung der Rolle ist in Abbildung 4.21 dargestellt.

4.3.4 Kommunikationsmodell

Das Kommunikationsmodell erläutert die Konzepte zum Austausch von Managementinformationen. Zunächst werden die Akteure sowie deren Kommunikationsbeziehungen darge-

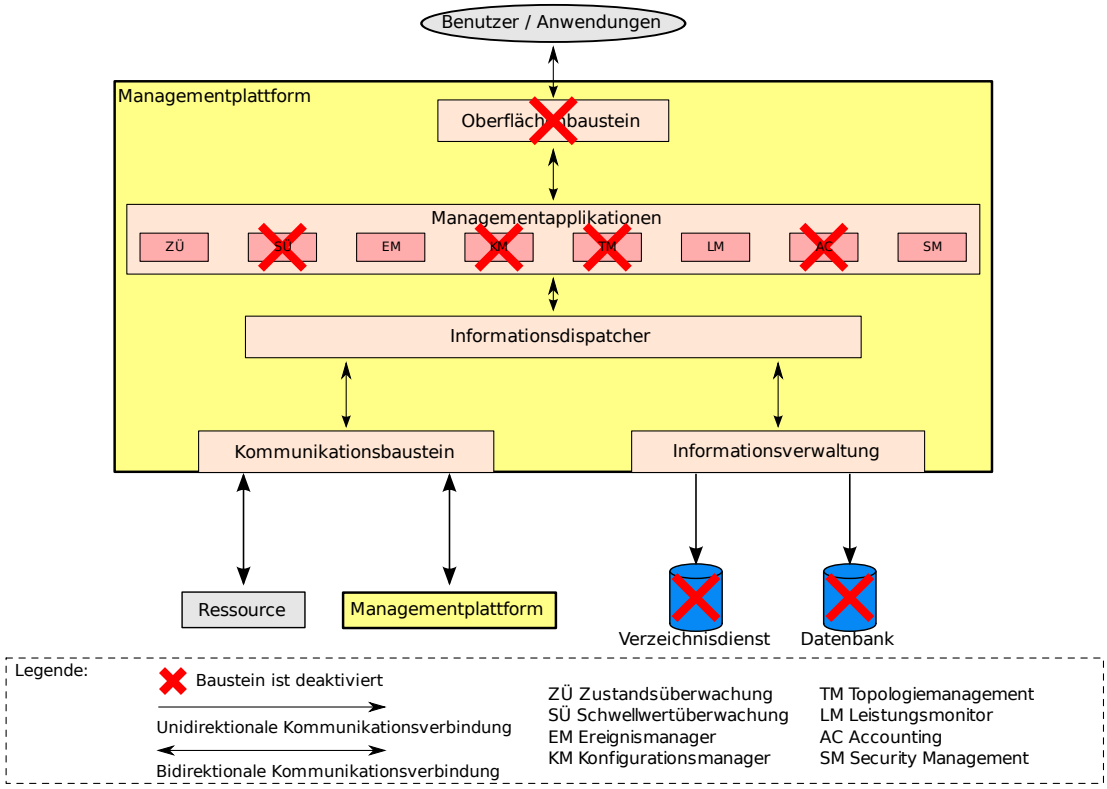

Abbildung 4.19: Managementplattform mit Rolle Proxy

stellt. Anschließend werden die Kommunikationsmechanismen zum Austausch der Managementinformationen festgelegt.

Kommunikationsbeziehungen

Die benötigten Managementinformation, also Monitoringinformationen, Steuerbefehle und Events, werden grundsätzlich zwischen Managementplattform und Ressourcen oder zwischen Managementplattformen übertragen.

Monitoringinformationen gehen ursprünglich von einer Ressource aus und werden dann zur zuständigen Managementplattform weitergeleitet, auf diesem Weg findet sowohl Kommunikation Ressource-Plattform statt, als auch eine zwischen Plattformen, wenn die Zielplattform nicht die Plattform mit der Ressourcenanbindung ist (siehe Abbildung 4.22). Der zweite Teil der Kommunikation (Plattform-Plattform) kann verzögert erfolgen, falls die Daten in einem zwischenliegendem Knoten gecached werden und erst auf Anforderung oder nach Intervallen weitergeleitet werden.

Steuerinformationen gehen diesen Weg in die entgegengesetzte Richtung, der Unterschied ist hierbei, dass diese für eine Ressource oder für eine Plattform bestimmt sind. Steuerinformationen für eine Plattform legen z.B. Caching-Policies fest. Der Weg der Steuerinformationen ist in Abbildung 4.23 dargestellt.

Events nehmen denselben Weg wie Monitoringinformationen. Jedoch kann hier der Erstellungsort auch eine Plattform sein, in dem Fall fällt natürlich der Wegeabschnitt Ressource-

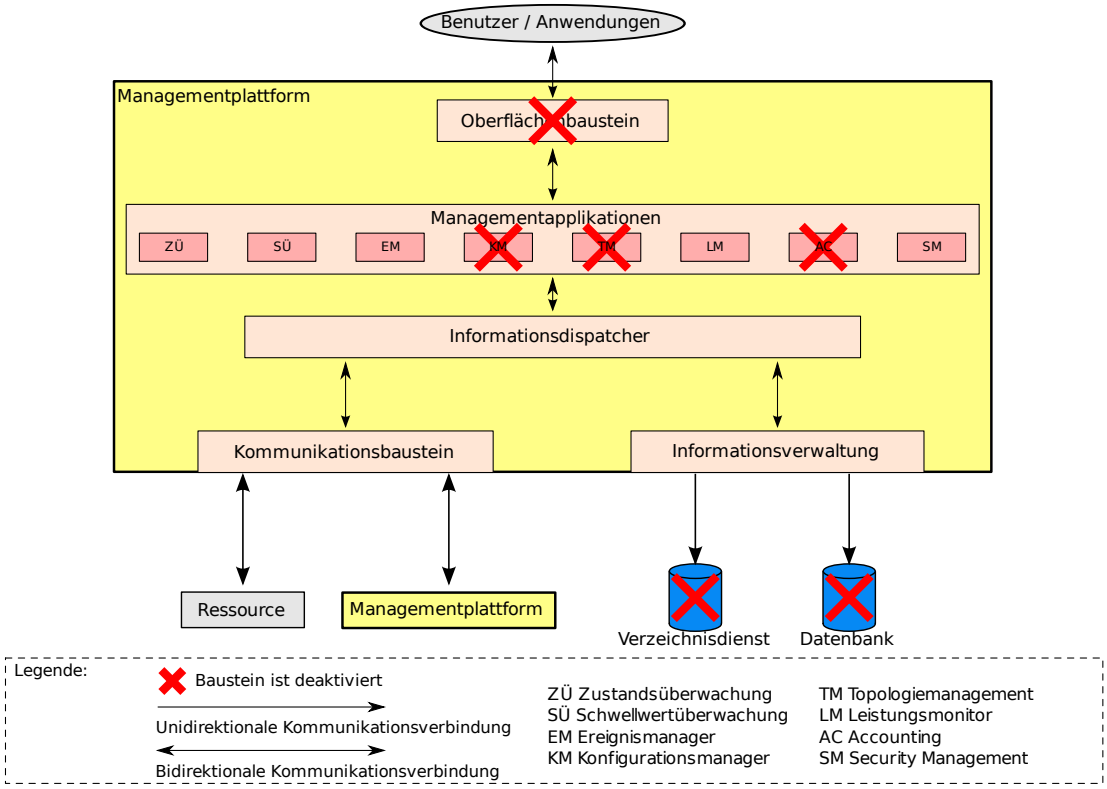

Abbildung 4.20: Managementplattform mit Rolle Caching-Proxy

Plattform weg.

Kommunikationsmechanismen

Das Kommunikationsmodell basiert, wie im Organisationsmodell in Abschnitt 4.3.3 beschrieben, auf dem hierarchischen Manager-Agenten-Modell. Folglich können die kommunizierenden Partner wechselnde Rollen einnehmen, sowohl die Agenten-, wie auch die Manager-Rolle. Manager fordern Informationen von den Agenten per *Request* an und Agenten antworten mit einer *Response* darauf.

Die folgenden Interaktionen werden hierbei verwendet:

- Request

  Get Fordert Managementinformationen an

  Set Sendet Steuerbefehle

  Add Fügt Managementobjekte hinzu

  Remove Entfernt Managementobjekte

- Response

  Update Sendet Managementinformationen

  Error Sendet Fehlermeldungen

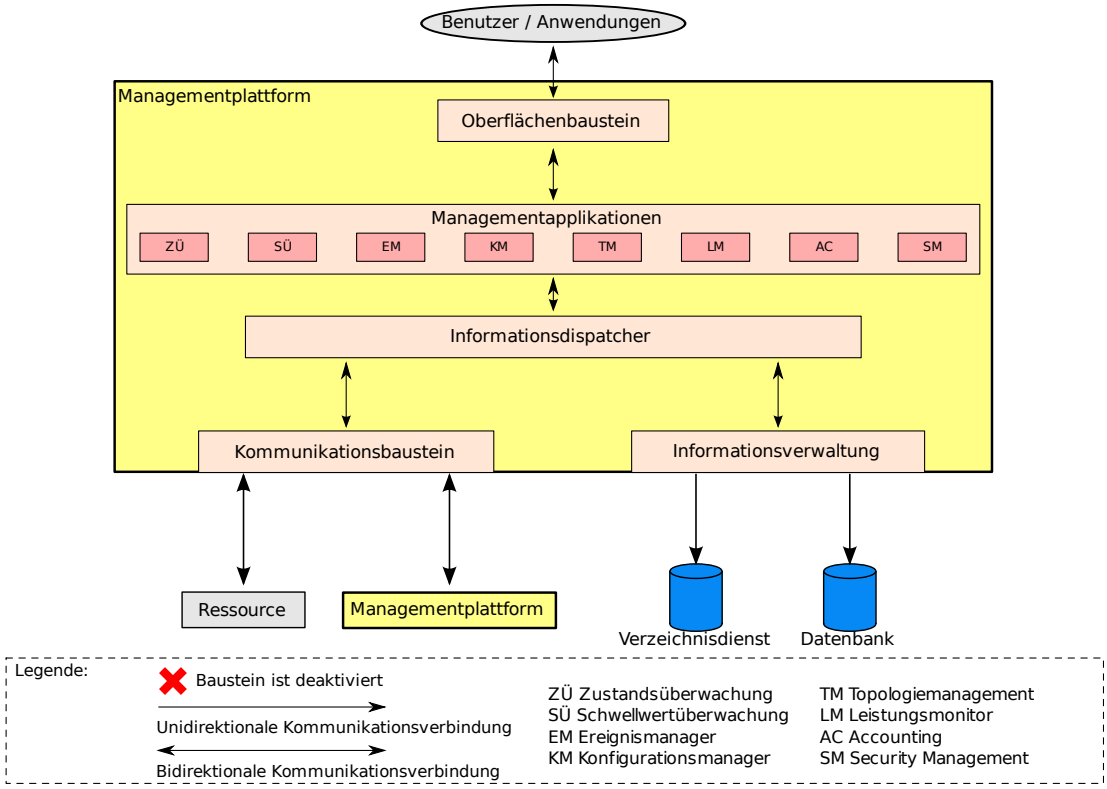

Abbildung 4.21: Managementplattform mit Rolle Master

Definition der Protokoll-Datenstrukturen

Die Managementinformationen werden zum Transport in spezielle Nachrichten verpackt, diese stellen die Protokoll-Datenstrukturen dar. Eine Gesamtübersicht einschließlich aller Unterklassen ist in Abbildung 4.24 dargestellt.

Die Basisklasse Der Transport der Managementinformationen findet über standardisierte Nachrichten statt. Die Basisklasse ist *Message*, von Ihr werden alle weiteren Klassen abgeleitet. Sie besitzt die Attribute *source_id* und *target_id*, zur Identifikation des Quell-, bzw. Zielelements, sowie das Attribut *timestamp*, welches die Erstellungszeit der Nachricht beinhaltet.

Die Interaktionsklassen Die im Unterabschnitt 4.3.4 aufgezeigten Interaktionen finden sich in den folgenden Transportnachrichtenklassen wieder.

Get Get-Nachrichten fordern Managementinformationen an. Es existieren zwei Unterklassen:

GetMonitoringUpdate Fordert ein Update der Monitoringinformationen an. Diese Nachricht ist an eine Ressource gerichtet und wird vom zuständigen Domänenmaster angefordert. Eventuelle Zwischenplattformen leiten diese Nachricht direkt weiter. Diese

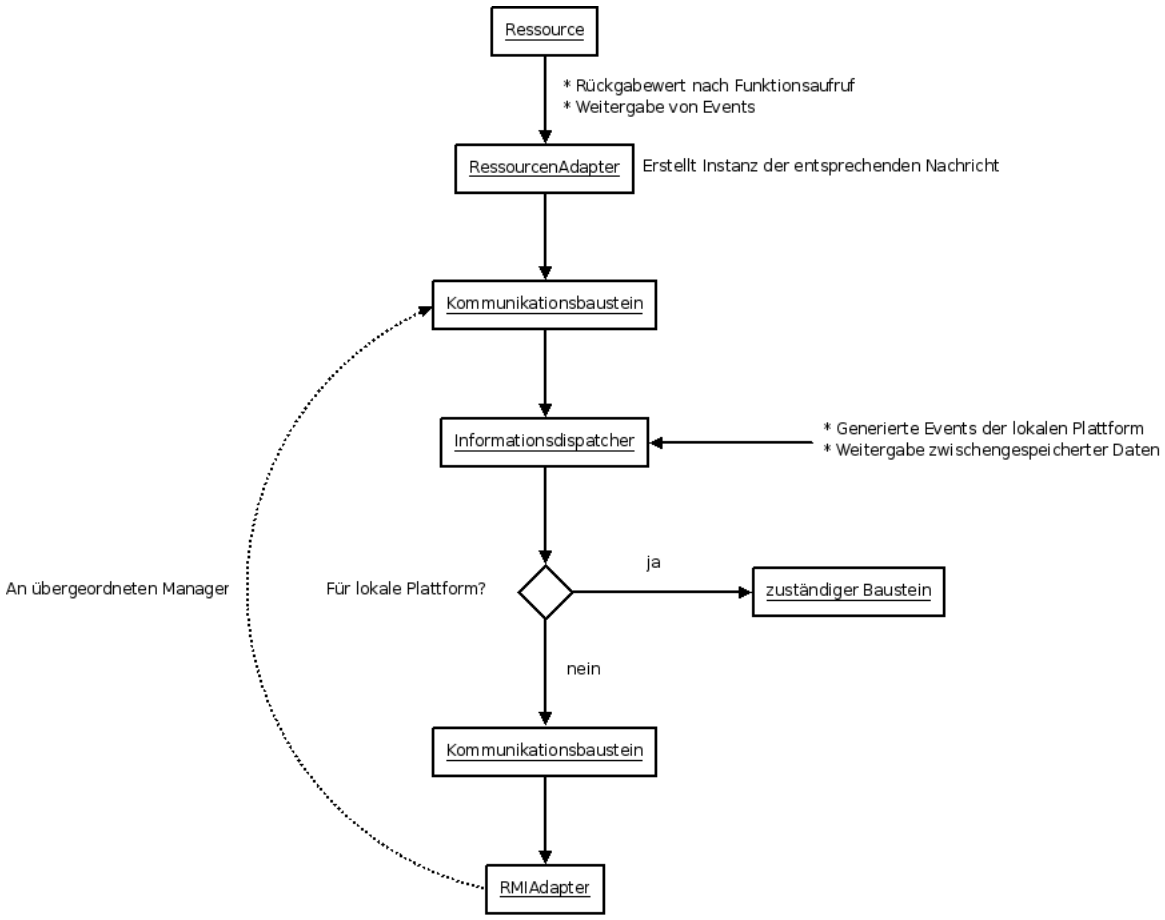

Abbildung 4.22: Weg der Monitoringinformationen und Events

Klasse ist wiederum unterteilt in verschiedene Klassen, welche das Zielelement bestimmen, von welchem die Monitoringdaten benötigt werden.

GetConfigUpdate Fordert die aktuelle Konfiguration eines Managementobjekts an. Diese Nachricht ist an eine Plattform gerichtet, entweder an den Informationsdispatcher zur Abfrage der aktuellen Konfiguration oder an eine Managementapplikation.

Set Set-Nachrichten beschreiben Konfigurationsänderungen. Die Unterklassen orientieren sich dabei an der Spezifikation des Informationsmodells.

SetPlatformConfig Ändert die Konfiguration einer Plattform (z.B. der Managementapplikationen). Diese Klasse ist entsprechend der Applikationen weiter unterteilt (siehe Abbildung).

SetPlatformState Ändert die Rolle einer Plattform.

SetVmState Verursacht einen Zustandsübergang bei einer VM.

SetSystemConfig Ändert die Konfiguration eines Systems.

SetDeviceConfig Ändert die Konfiguration eines Devices.

SetFilesystemConfig Ändert die Konfiguration eines Filesystems.

SetLogicalFileConfig Ändert die Konfiguration eines LogicalFiles.

Add Fügt ein Managementobjekt hinzu. Die Unterklassen spezifizieren das hinzuzufügende Managementobjekt näher. Das Attribut *config* beinhaltet eine Konfiguration für das neue Objekt.

AddSystem Fügt ein System hinzu. Die Unterklassen spezifizieren ob es sich um ein physisches System oder ein virtuelles handelt.

AddDevice Fügt ein Device hinzu. Fügt einem System ein Device hinzu. Das Attribut *system* gibt das System an.

AddFilesystem Fügt ein Filesystem hinzu. Das Attribut *system* gibt das System an.

AddLogicalFile Fügt eine Datei hinzu. Das Attribut *filesystem* gibt das Filesystem an.

AddPlatform Integriert eine neue Plattform in das System.

Remove Remove-Nachrichten entfernen ein Managementobjekt aus dem System. Die Unterklassen sind analog zu den Add-Nachrichten definiert und werden nicht explizit erläutert.

Update Update-Nachrichten transportieren Monitoringinformationen oder Informationen über erfolgte Konfigurationsänderungen. Entsprechend sind zwei Unterklassen vorhanden:

MonitoringUpdate Transportiert Monitoringinformationen. Hat ihren Ursprung in einer Ressource, kann jedoch auch von einer Plattform gesendet werden, falls diese die Informationen zwischenspeichert.

ConfigUpdate Beinhaltet eine erfolgte Konfigurationsänderung und ist eine direkte Antwort auf eine SetConfig-Nachricht.

Error Beinhaltet Fehlermeldungen auf Get-, Add- oder Remove-Nachrichten. Das Attribut *description* erläutert das aufgetretene Problem näher. *message* beinhaltet die auslösende Nachricht.

4.4 Abgleich mit den Anforderungen

Der Vergleich des Modells mit den in Kapitel 2 aufgestellten Anforderungen ist in Tabelle 4.1 dargestellt. Es zeigt sich, dass die gewünschten Anforderungen von den erarbeiteten Modellen komplett unterstützt werden.

Tabelle 4.1: Abgleich mit den Anforderungen

Anforderung	erfüllt
Funktionale Anforderungen	
Hinzufügen/Entfernen von Ressourcen	✓
Konfigurieren der Ressourcen	✓
Monitoring der Ressourcen	✓
Dienstgüteüberwachung der Ressourcen	✓
Zustandsüberwachung der Ressourcen	✓
VM-Migration	✓
VM-Steuerung	✓
VM-Vorlagen	✓
Klonen von VMs	✓
Benutzer-/Gruppenverwaltung	✓
Benachrichtigungen	✓
Autodiscovery	✓
Nichtfunktionale Anforderungen	
Einheitliche MIB	✓
Zugriffsberechtigung auf Ressourcen	✓
Abstrakte Anbindung an Datenbanken	✓
Abstrakte Anbindung an Verzeichnisdienste	✓
Schnittstelle für externe Anwendungen	✓
Plattformunabhängige Bedienbarkeit	✓
Skalierbarkeit in größeren Umgebungen	✓
Führen einer Datenhistorie	✓

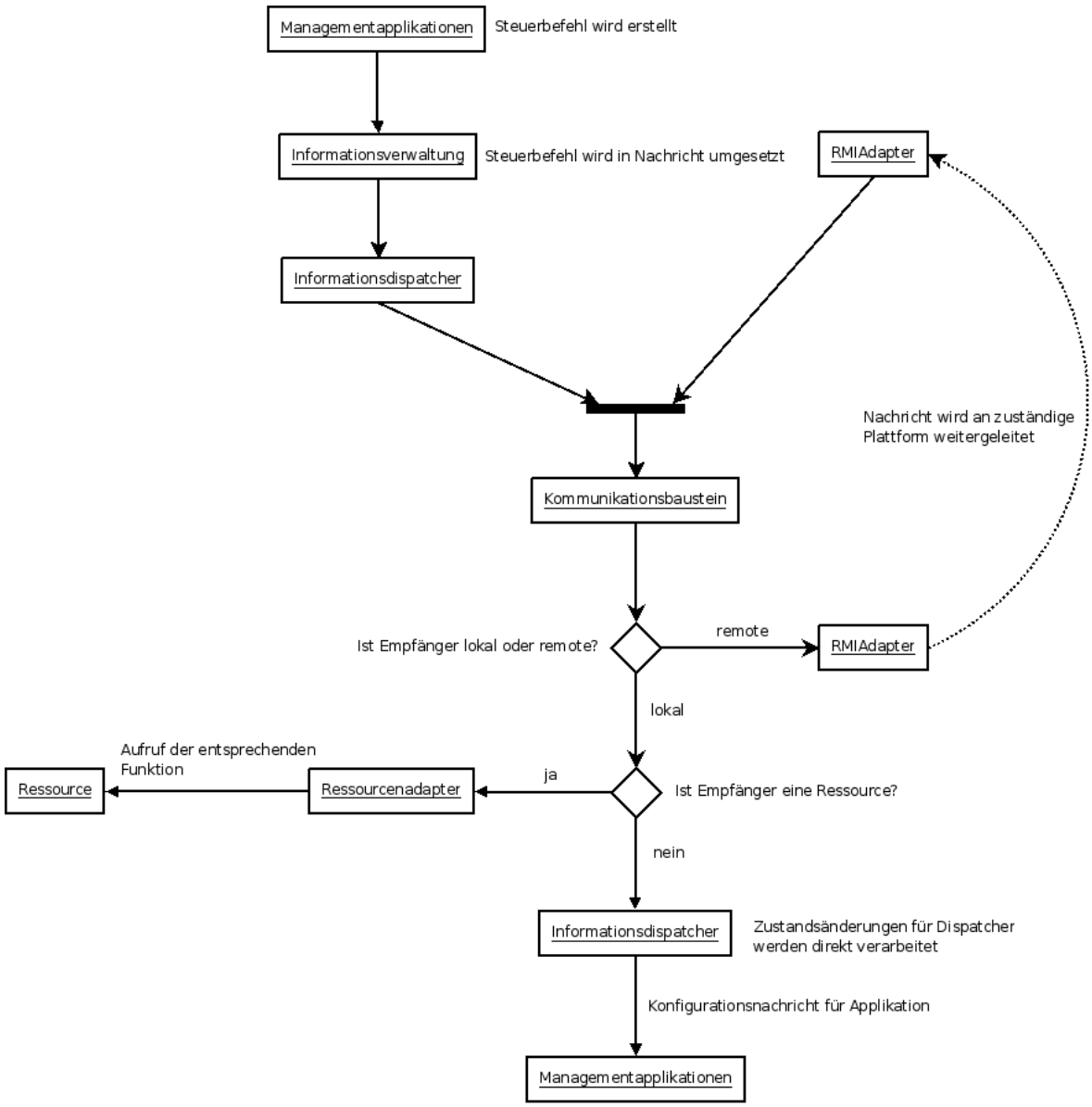

Abbildung 4.23: Weg der Steuerbefehle

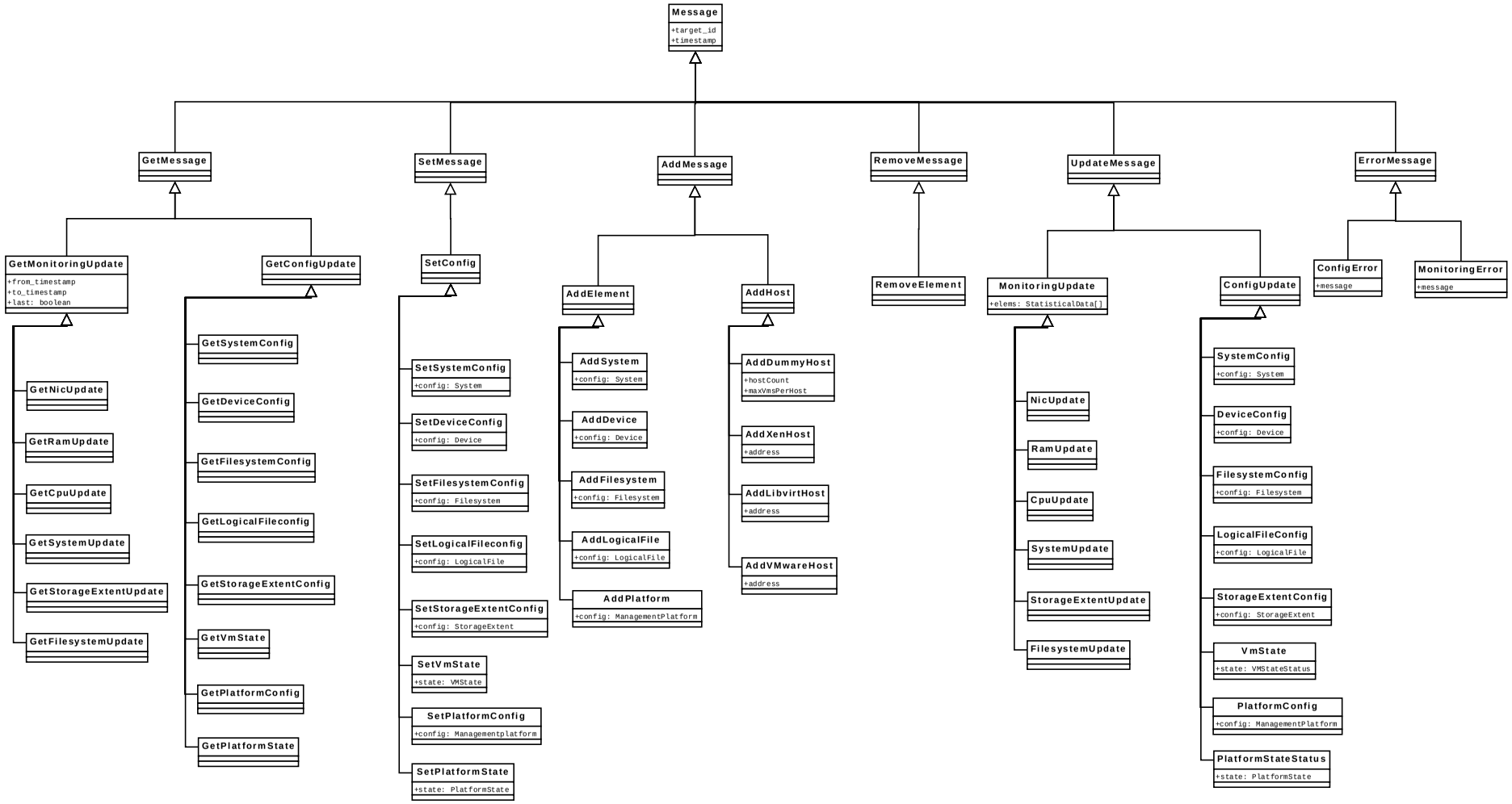

Abbildung 4.24: UML-Diagramm der Messages-Klasse

5 Entwurf

In diesem Kapitel wird anhand der entwickelten Modelle die Managementplattform entworfen. Die Modellierungssprache wird weiterhin UML sein.

5.1 Einschränkungen

Die folgenden Einschränkungen wurden beim Entwurf gemacht um den Zeitrahmen dieser Arbeit nicht zu sprengen.

Authentifizierung / Autorisierung

Die Authentifizierung sowie die Autorisierung der Benutzer wird im Entwurf nur rudimentär berücksichtigt, d.h. die dazu nötigen Objekte sind vorhanden, aber deren Funktionalität wird nicht genutzt. Dies kann zu einem späteren Zeitpunkt jedoch nachgerüstet werden.

5.2 Entwurf der Plattform

Zuerst wird die Plattform in einer Gesamtsicht mittels eines UML-Klassendiagramms, zur besseren Übersicht allerdings ohne Attribute und Methoden, dargestellt. Die einzelnen Komponenten der Plattform werden anschließend detaillierter in Diagrammen mit Fokus auf den Nachrichtenfluss aufgezeigt.

5.2.1 Gesamtübersicht

In Abbildung 5.1 ist das Modell der Plattform dargestellt, es orientiert sich an den Modellen aus Kapitel 4. Die zentrale Komponente ist der *InformationDispatcher*, der die Nachrichtenvermittlung zwischen den Komponenten *ApplicationManagement*, *InformationManagement* und *CommunicationManagement* vornimmt. Diese Kapseln jeweils den Zugriff auf ihren Infrastrukturbaustein und verdecken dessen innere Funktionsweise. Dies entspricht der Funktionsweise des Entwurfsmusters *Fassade*, bei dem die Subsystemklassen durch die Fassadeklasse verdeckt werden (siehe dazu [fas08]).

Eine weitere Kapselung findet zwischen den externen Zugangspunkten und den Managementapplikationen statt. Hier dient das *PlatformInterface* als Schnittstelle zu den Applikationen. Jeder Zugriff auf diese muss über sie erfolgen.

5.2.2 Der InformationDispatcher

Der Informationdispatcher verteilt die empfangenen Nachrichten an die angebundenen Komponenten. Dies erfolgt je nach aktueller Rolle des Dispatchers. Abbildung 5.2 zeigt den Fluss der Nachrichten. Die Methoden *receiveFromApp()*, *receiveFromComm()* und *receiveFromInfo()* nehmen die Nachrichten der entsprechenden Komponenten entgegen, leiten diese an den

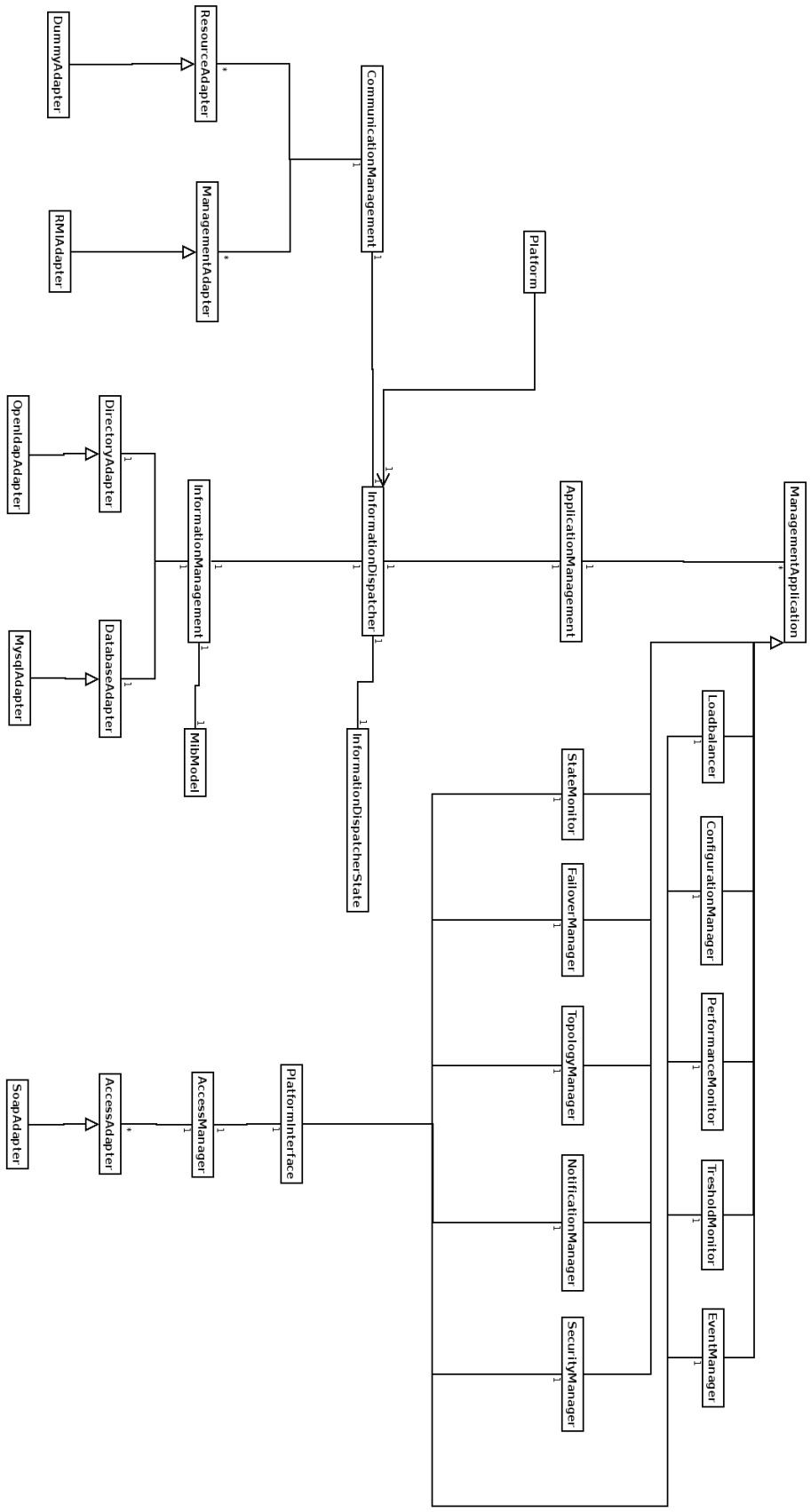

Abbildung 5.1: UML-Diagramm der Plattform

InformationDispatcherState weiter. Dieser stellt die aktuelle Rolle des Dispatchers dar und verteilt die Nachrichten gemäß dieser weiter an die angebundenen Komponenten.

5.2.3 Die Informationsverwaltung

Die in Abbildung 5.3 dargestellte Informationsverwaltung beinhaltet ein im Hauptspeicher gehaltenes Modell der MIB, sowie mehrere Anbindungen an Datenbanken und Verzeichnisdienste, Die Klasse *InformationManagement* dient als Schnittstelle und nimmt Anfragen von anderen Komponenten entgegen. Zum Empfang von Nachrichten steht die Methode *receiveFromDispatcher()* zu Verfügung. Die so empfangenen Nachrichten werden verarbeitet und die entsprechende Funktion ausgeführt, z.B. Hinzufügen einer neuen VM oder hinzufügen eines neuen Devices zu einer VM. Zum einfacheren Zugriff auf die verwalteten Elemente führt die Informationsverwaltung Listen über diese.

5.2.4 Die Managementapplikationen

Die Interaktionen der Managementapplikationen sind in Abbildung 5.4 dargestellt. Die Schnittstelle bildet das *ApplicationManagement*, welches Ereignisse vom InformationDispatcher entgegennimmt und an den *EventManager* weiterleitet. Dieser verteilt die eingehenden Nachrichten an die für diese Nachrichtenklasse mittels registerForEvents(Object, MessageClass) registrierten Applikationen.

Der *ConfigurationManager* bildet für die Managementapplikationen die Schnittstelle zur Informationsverwaltung. Über ihn können Managementinformationen gelesen, sowie Konfigurationsänderungen vorgenommen werden.

Die beiden Applikationen *StateMonitor* und *TresholdMonitor* empfangen Nachrichten über neue Monitoringdaten. Diese Monitoringdaten werden auf abnormale Systemzustände bzw. Schwellwertüberschreitungen untersucht. Dazu können beide Applikationen auf die in der Informationsverwaltung vorgehaltenen Daten über den *ConfigurationManager* zugreifen. Werden kritische Werte festgestellt wird eine Meldung an den EventManager gesendet. Diese werden dann vom *Loadbalancer* bzw. *FailoverManager* empfangen und weiterverarbeitet.

Der *NotificationManager* nimmt die Nachrichten entgegen und schreibt diese in ein Logfile und/oder sendet eine E-Mail-Benachrichtigung. Dieses Verhalten ist vom Benutzer je Nachrichtenklasse einstellbar über die Methoden *setLog(MessageClass, enabled)* und *setMail(MessageClass, e-mail, enabled)*. *MessageClass* bezeichnet die Nachrichtenklasse, *enabled* zeigt an ob diese Regel aktiviert ist und *e-mail* bezeichnet die E-Mail-Adresse des Empfängers der Nachricht.

Der *PerformanceMonitor* stösst Messungen an den Ressourcen an. Der *ConfigurationManager* generiert die entsprechenden Nachrichten und übergibt sie dem Dispatcher.

Der *TopologyManager* führt das Autodiscovery durch und übergibt gefundene Ressourcen dem *ConfigurationManager*. Dieser generiert die dazugehörige Nachricht und versendet diese.

Der *SecurityManager* interagiert direkt mit der Informationsverwaltung und führt Authentifizierung und Autorisierung durch.

5.2.5 Der Kommunikationsbaustein

Die Klasse *CommunicationManagement* bildet die Schnittstelle zwischen dem Dispatcher und den Adaptern zu den externen Komponenten (siehe Abbildung 5.5). Die Methoden *receiveFromDispatcher()*, *receiveFromResource()* und *receiveFromExtern()* nehmen die Nach-

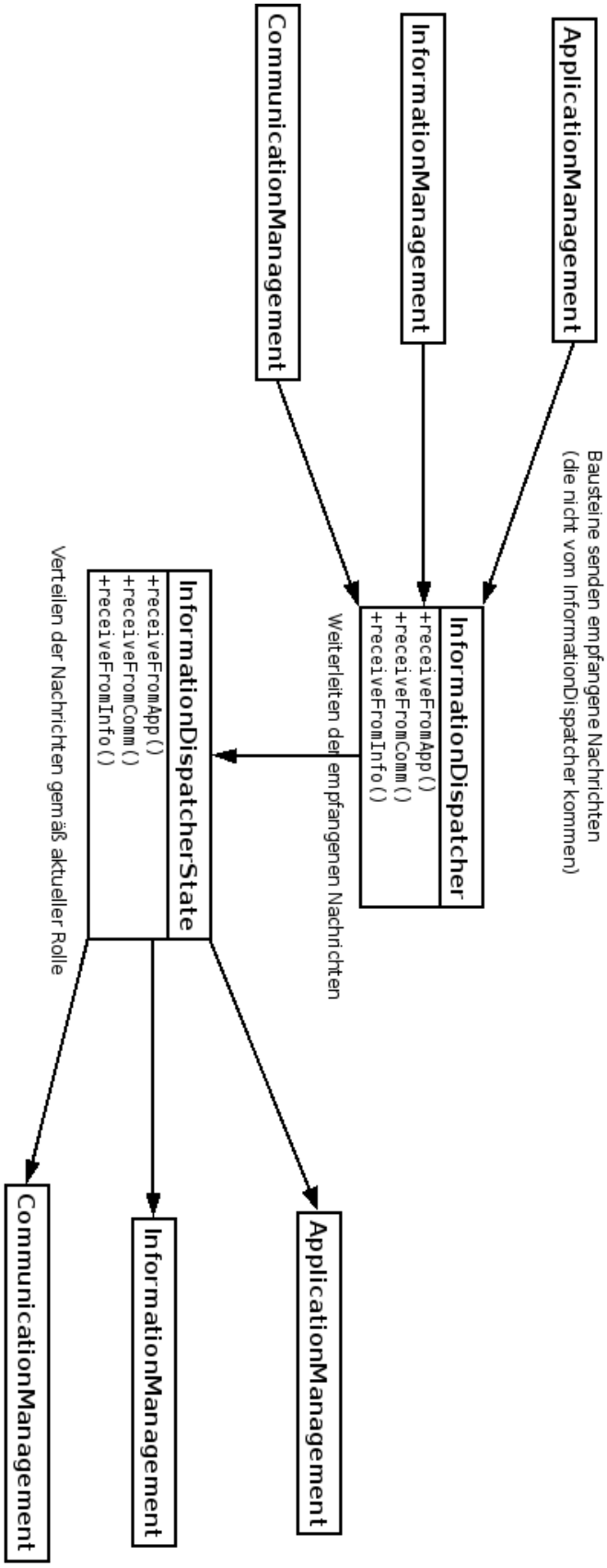

Abbildung 5.2: Nachrichtenverarbeitung durch den InformationDispatcher

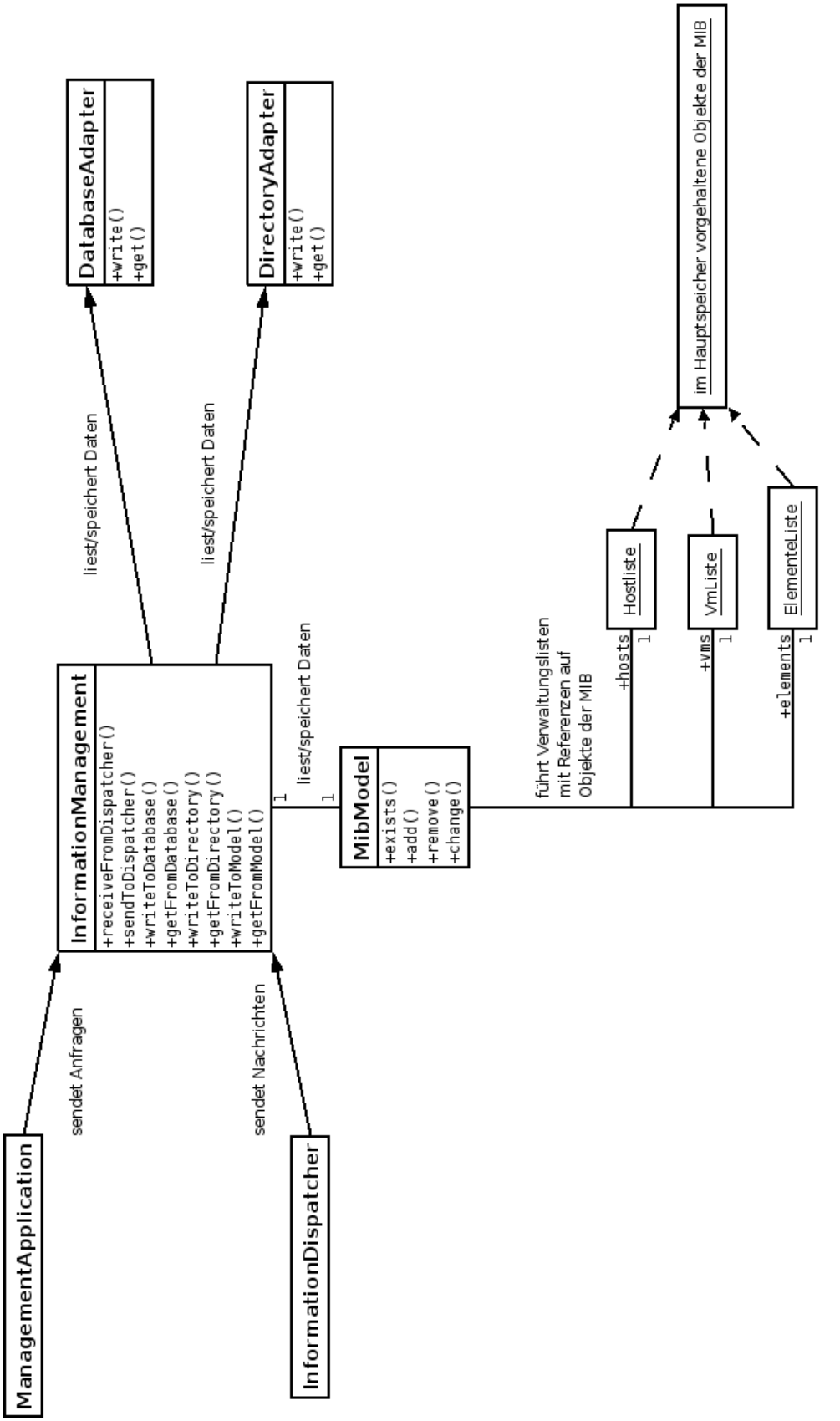

Abbildung 5.3: Die Informationsverwaltung

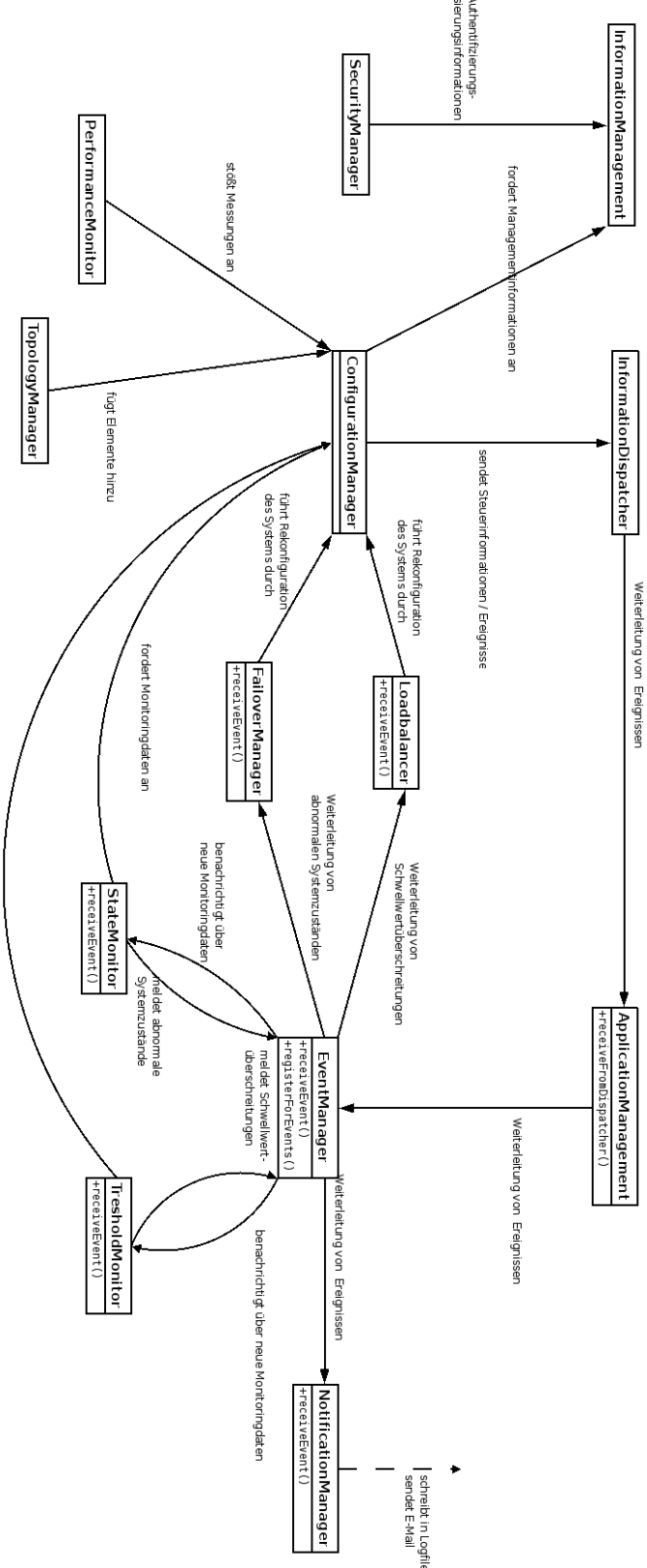

Abbildung 5.4: Die Managementapplikationen

richten aus den zugehörigen Quellen entgegen. Die darüber empfangenen Nachrichten werden entsprechend ihres Empfängers weitergeleitet. Zur Unterscheidung welche Empfänger lokal erreichbar sind und über welchen Adapter, führt das CommunicationManagement eine Verwaltungsliste. Darin sind die Empfänger-IDs sowie die zugehörigen Adapter gespeichert. Updates der Liste erfolgen über *AddMessages* und *RemoveMessages* (siehe Abschnitt 4.3.4).

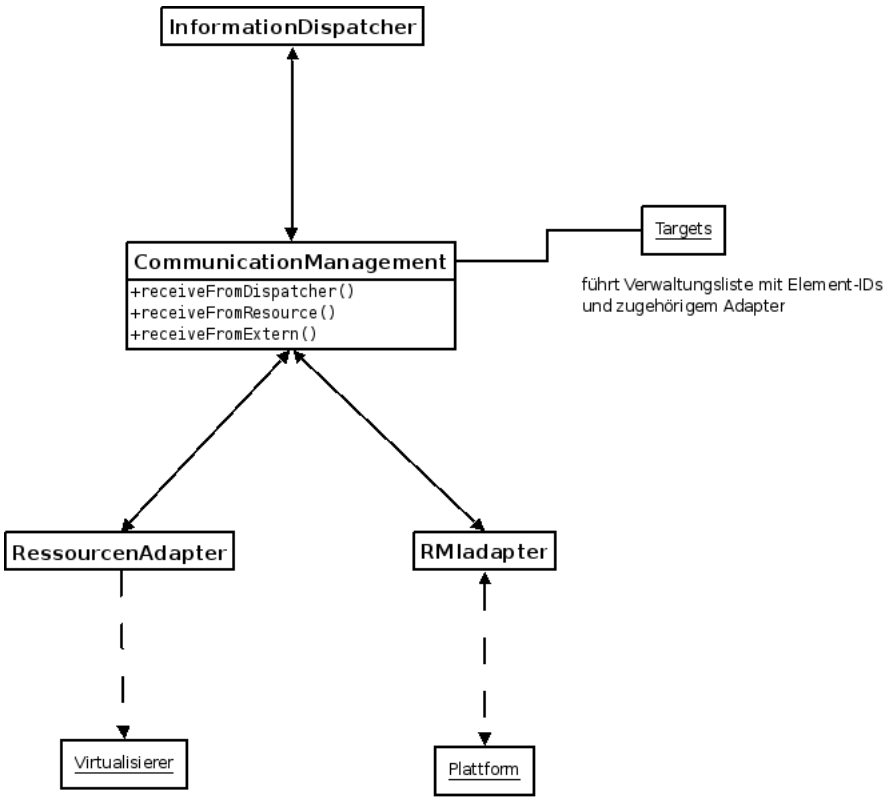

Abbildung 5.5: Der Kommunikationsbaustein

5.2.6 Der Oberflächenbaustein

Abbildung 5.6 stellt den Datenfluss eines Zugangsadapters, in diesem Falle des Soapadapters, dar. Der Soapadapter nimmt Anfragen über das Soapprotokoll entgegen und setzt diese in Anfragen an die Managementapplikationen um. Zur Kommunikation mit diesen dient das *PlatformInterface*. Dieses bildet die Schnittstelle zu den verfügbaren Managementapplikationen.

Der SecurityManager wird in diesem Modell nicht in den Datenfluss eingebunden (siehe 5.1). Die Authentifizierung der Benutzer würde der AccessManager in Zusammenspiel mit dem SecurityManager vornehmen. Jeden Zugriff auf Informationen würde das PlatformInterface vom SecurityManager validieren lassen.

Über die Methode *getSysteminfo()* werden aktuelle Systeminformationen angefordert (z.B. Anzahl der Hosts und VMs, Gesamtzahl der verfügbaren Elemente).

Die Methode *getHostlist()* fordert eine Liste der im System verfügbaren Hosts an.

addDummyHost fügt einen Testadapter dem System hinzu, welcher Hosts und VMs zu

Testzwecken simuliert.

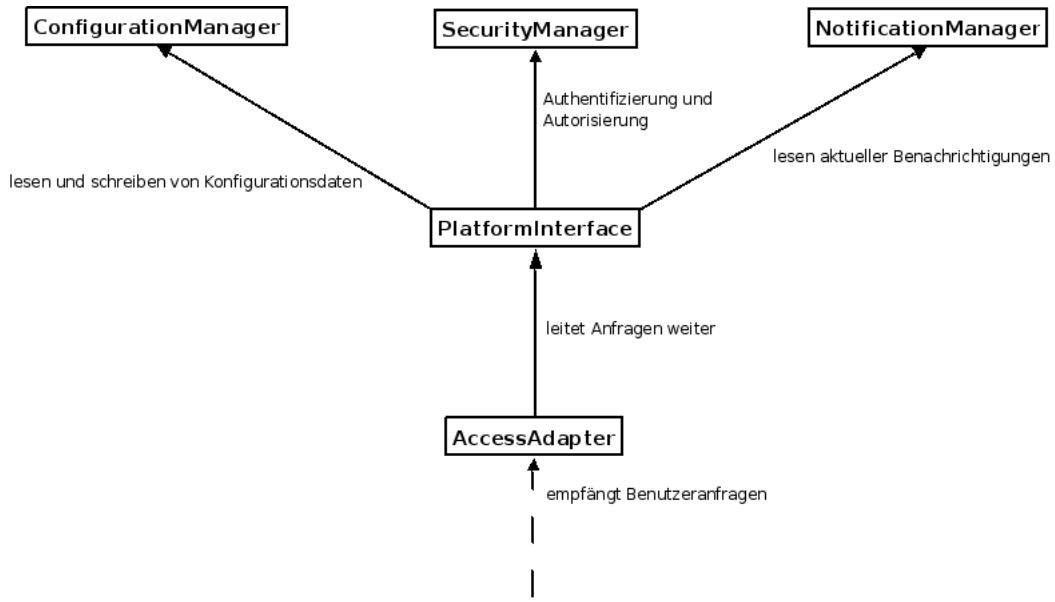

Abbildung 5.6: Der Oberflächenbaustein

5.3 Entwurf des Webinterfaces

Dieser Abschnitt geht kurz auf den Entwurf des Webinterfaces ein. Zuerst wird auf die Anbindung an die Plattform eingegangen, anschließend auf die Benutzerschnittstelle.

5.3.1 Anbindung an die Plattform

Die Anbindung des Webinterfaces erfolgt über den bereitgestellten Webservice mittels dem Soapprotokoll. Die Kommunikationsendpunkte bilden die Soapadapter, mit Serverfunktionalität auf Seite der Plattform, mit Clientfunktionalität auf Seite des Webinterfaces. Die Methoden sind auf beiden Seiten gleich, jedoch werden die Methoden des Servers vom Client aufgerufen und die des Client von der Benutzerschnittstelle.

5.3.2 Benutzerschnittstelle

Die Benutzerschnittstelle bietet verschiedene Sichten auf das System, welche unterschiedliche Informationen und Funktionen zu Verfügung stellen.

Host-Liste Die Host-Liste stellt die verfügbaren Hostsysteme in einer Liste dar. Relevante Informationen, z.B. CPU-Auslastung und Anzahl der darauf laufenden VMs, werden in der Liste direkt dargestellt. Weitere Informationen zu den einzelnen Host können aufgerufen werden.

Host-Details Alls weiteren Informationen zu einem Hostsystem sind hier ersichtlich. Außerdem können für den Host relevante Funktionen ausgeführt werden, z.B. Herunterfahren des Hostsystems.

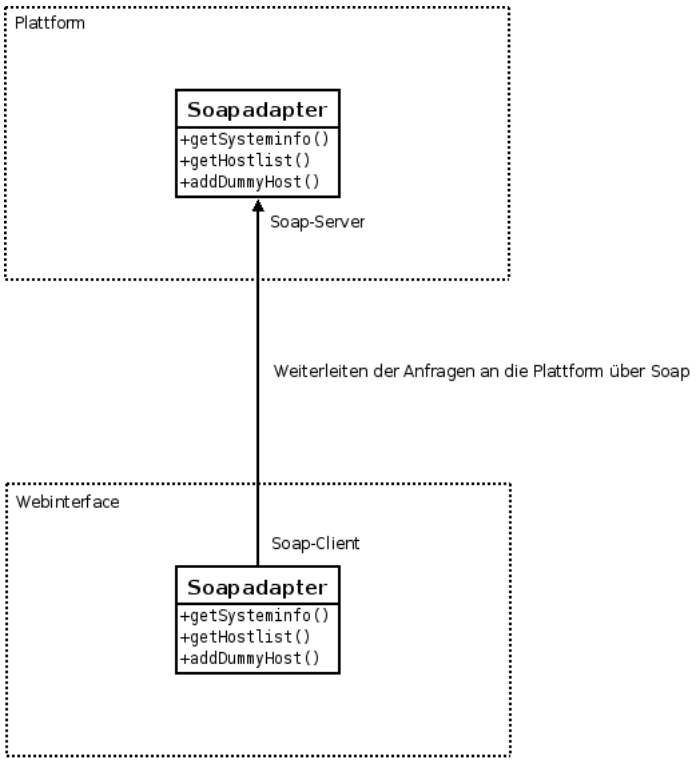

Abbildung 5.7: Anbindung des Webinterfaces an die Plattform

VM-Liste Stellt alle verfügbaren VMs oder nur die eines Hosts dar. Weitere Informationen zur VM können hierüber abgerufen werden.

VM-Details Stellt alle weiteren Informationen zu einer VM dar. Für diese VM relevante Funktionen können ausgeführt werden, z.B. starten und stoppen der VM.

Nachrichten Die Nachrichtensicht stellt aktuelle Informationen dar, z.B. Bestätigungen nach dem Ausführen einer Funktion.

Systeminformationen Allgemeine Informationen über das System können hier eingesehen werden, z.B. Anzahl der verfügbaren Hosts und VMs, Anzahl der Elemente insgesamt.

6 Implementierung

Dieses Kapitel erläutert die Implementierung des Prototypen. Der erste Abschnitt erläutert zuerst die getroffenen Einschränkungen, da nicht alle Funktionalitäten umgestzt wurden. Danach folgen kurze Darstellungen der Managementplattform sowie des Webinterfaces.

Informationen zur Installation des Prototypen finden sich in Anhang A.1.

6.1 Einschränkungen

Die folgenden Einschränkungen wurden vorgenommen, da der Schwerpunkt dieser Arbeit mehr auf der theoretischen Ausarbeitung besteht, und der Prototyp vorrangig Demonstrationszwecken dient.

6.1.1 Verzeichnisdienstanbindung

Auf die Anbindung eines Verzeichnisdienstes wird bei der Implementierung verzichtet, da das Entwurfsmodell die Authentifizierung und Autorisierung der Benutzer erst zu einem späteren Zeitpunkt vorsieht.

6.1.2 Datenbankanbindung

Die Anbindung einer Datenbank wird nicht vorgenommen, alle Managementinformationen werden nur im Hauptspeicher vorgehalten, dies ist jedoch für den Prototyp zu Testzwecken ausreichend.

6.1.3 Kommunikation der Plattformen untereinander

Die Kommunikation der Plattformen untereinander wird bei der Implementierung nicht berücksichtigt.

6.1.4 Kommunikation mit den Ressourcen

Die Implementierung läßt die Kommunikation mit den Ressourcen außer Acht. Zwei Adapter, einer für Libvirt, einer für VMware ESX, sind im Rahmen eines Fortgeschrittenenpraktikums [HM08] entwickelt worden.

6.2 Implementierung der Plattform

Die Implementierung der Plattform wird in der Sprache Java in Version 6 vorgenommen. Java ist weit verbreitet, plattformunabhängig und bietet Unterstützung für Webservices mittels Soap [FB07, S. 1043 ff]. Im folgenden wird, als Ergänzung zu den Modellen im Entwurf, auf die Besonderheiten der Implementierung in Java eingegangen. UML-Klassendiagramme

werden nur bei größeren Abweichungen zu den Entwurfsmodellen angegeben, ansonsten sind diese im Anhang unter A.4 zu finden.

6.2.1 Packages

Die verwendeten Klassen sind in die folgenden Packages unterteilt:

access enthält alle Komponenten des Oberflächenausteins. Dazu kommt eine eigene MIB für Objekte die nicht in der MIB für VL abgedeckt sind.

application enthält alle Komponenten des Bausteins Managementapplikationen.

communication enthält alle Komponenten des Kommunikationsbausteins.

core enthält Komponenten des Anwendungskerns. Dazu gehören u.a. Eine Klasse zum starten der Anwendung (Platform), der InformationDispatcher und die dazugehörigen Zustände, sowie ein ID-Generator zu Generierung eindeutiger IDs.

information enthält alle Komponenten der Informationsverwaltung.

message enthält die Nachrichtenklassen aus dem Kommunikationsmodell.

mib enthält die Klassen aus der MIB.

6.2.2 Verwendete Datenstrukturen

In Kapitel 5 sind einige Objekte mit Listen verknüpft. Im folgenden werden diesen Java-Datenstrukturen zugeordnet.

Warteschlangen zur Nachrichtenverarbeitung

Die Weiterleitung von Nachrichten erfolgt nach dem store-and-forward Prinzip, d.h. sie werden von Komponente zu Komponente übergeben, zwischengespeichert und wieder weitergeleitet. Die Nachrichtenverarbeitung erfolgt dabei nach dem FIFO-Verfahren. Dazu sind Java-Datenstrukturen nötig die dieses ermöglichen. Java stellt dazu verschiedene *Queues* bereit, wobei die Subklasse *LinkedList* das gewünschte unterstützt. Das folgende Beispiel zeigt dies kurz auf.

Zeile 1 initialisiert die Warteschlange. In Zeile 2 wird eine Nachricht in die Warteschlange am Ende eingefügt. In Zeile 3 wird die älteste Nachricht entnommen.

```
1 LinkedList<Message> messages = new LinkedList<Message>();
2 messages.add(new Message());
3 Message m = messages.remove();
```

Einfache Objektlisten

Für einfache Listen, d.h. nach Objekten muss nicht gesucht werden, eignet sich die Klasse *ArrayList*, eine *Subklasse* von Collection. Der Einsatz dieser Liste ist z.B. in der Informationsverwaltung sinnvoll, da diese mehrere Verwaltungslisten vorhält, u.a. eine die nur Hostsysteme enthält. Diese dient hauptsächlich zur Zählung der verfügbaren Hosts sowie der

Generierung von Hostlisten aufgrund von Benutzeranfragen. Folgendes Beispiel erläutert dessen Funktionsweise.

Zeile 1 initialisiert die Liste. In Zeile 2 wird ein neues System der Liste hinzugefügt und in Zeile 3 wird deren Größe ermittelt.

```
1 ArrayList<mib.System> hosts = new ArrayList<mib.System>();
2 hosts.add(new System());
3 hosts.size();
```

Suchlisten für Objekte

Suchlisten werden benötigt um nach Objekten anhand deren eindeutiger ID zu suchen. Dies wird u.a. dazu eingesetzt um festzustellen welcher Adapter innerhalb des Kommunikationsbausteins für welche Ressourcen zuständig ist. Java stellt für diesen Zweck die Klasse *Hashtable* zu Verfügung. Das folgende Beispiel verdeutlicht dessen Funktionsweise.

Zeile 1 initialisiert die Hashtabelle. Das in Zeile 2 generierte System wird in Zeile 3 anhand dessen ID in die Tabelle eingefügt. In Zeile 4 wird es anhand der ID in der Tabelle gesucht und zurückgegeben.

```
1 Hashtable elements = new Hashtable();
2 mib.System s = new mib.System();
3 elements.put(s.id, s);
4 s = elements.get(s.id);
```

6.2.3 Bereitstellung des Webservice

Die Bereitstellung des Webservices erfolgt über die bei Java 6 mitgelieferten Soap-Bibliotheken [FB07, S. 1043 ff]. Das folgende Beispiel zeigt die Einrichtung des Webservices mit einer Methode auf.

In den Zeilen 1 und 2 wird der Webservice und dessen Typ definiert. Im Konstruktor (ab Zeile 6 wird ein Soap-Endpunkt erstellt und an den Port 8160 am lokalen Interface gebunden. In Zeile 11 wird eine Webservicemethode definiert welche die Methode ab Zeile 12 bereitstellt.

```
1  @WebService
2  @SOAPBinding(style=SOAPBinding.Style.RPC)
3  public class SoapAdapter extends AccessAdapter{
4      private PlatformInterface pi;
5
6      public SoapAdapter(AccessManager am){
7          this.pi = am.getPlatformInterface();
8          Endpoint ep = Endpoint.publish(
9                  "http://localhost:8160/utilities", this);
10     }
11
12     @WebMethod(operationName="getSysteminfo")
13     public PlatformData getSysteminfo(){
14         return this.pi.getSysteminfo();
```

```
15      }
16  }
```

6.3 Implementierung des Webinterfaces

Das Webinterface wurde auf Basis des PHP-Frameworks Prado [pra08] realisiert. Dieses Framework ermöglicht den schnellen Aufbau einer Webanwendung nach dem MVC-Prinzip. Voraussetzungen für dessen Einsatz sind ein Webserver mit PHP-Unterstützung sowie PHP selbst. Für den Prototyp wurde ein Apache 2.2.9-7 zusammen mit PHP 5.2.6-3 benutzt. Die Bibliotheken zur Nutzung eines Webservices mittels Soap sind in dieser PHP-Version enthalten (siehe [Tra05, S. 241 ff]).

Anmerkungen zu Installation und Betrieb des Frameworks sind in Anhang A.2 zu finden.

Zusätzliche Informationen, u.a. zu diversen Problemen zwischen Java, PHP und WSDL (Web Services Description Language) sind in Anhang A.3 zu finden.

6.3.1 Anbindung an die Plattform

Die Anbindung an die Plattform nimm die Klasse *soap* vor. Der nachfolgende Code-Ausschnitt zeigt das Einrichten der Verbindung sowie einen Methodenaufruf per Soap:

```
1  class soap extends TModule{
2    private $client;
3    public $wsdlUrl;
4
5    public function soap(){
6      $this->wsdlUrl = "http://localhost/vlgui/wsdl.xml";
7      $classmap = array(
8        'host' => 'Host',
9        'platformdata' => 'PlatformData',
10       'managedElement' => 'ManagedElement',
11       'statisticalData' => 'StatisticalData',
12       'hostPerf' => 'HostPerf'
13       );
14     $this->client = new SoapClient($this->wsdlUrl,
15         array('classmap' => $classmap));
16   }
17
18   public function getSysteminfo(){
19     return $this->client->getSysteminfo();
20   }
21 }
```

In Zeile 6 wird die URL mit der WSDL-Datei angegeben, in dieser sind u.a. die Objekte definiert, die vom Server übertragen werden sollen. Ab Zeile 7 werden diese mit existierenden PHP-Klassen verbunden. Die Verbindung kommt schließlich in Zeile 14 zustande. In Zeile 19 wird ein entfernter Aufruf getätigt und die Systeminformationen abgefragt.

6.3.2 Benutzerschnittstelle

Die Implementierung erfolgt, gemäß dem MVC-Prinzip, in verschiedenen Dateien. Durch Verwendung der asynchronen Übertragungstechnik *Ajax* [Gar05] sind für Controller und View nur zwei Dateien vonnöten, da nur Teile der Seiten ausgetauscht werden, nicht wie bei einer herkömmlichen Webanwendung die komplette HTML-Seite. Die Datei */protected/pages/Home.page* enthält als View das HTML-Gerüst der in Abschnitt 5.3.2 beschriebenen Sichten. In */protected/pages/Home.php* sind die dazugehörigen Controller-Aktionen enthalten. Das Model befindet sich im Verzeichnis */protected/modules/vlmib*, hier befinden sich die Klassen entsprechend der MIB.

Die Benutzerinteraktionen folgen dem in Abbildung 6.1 dargestellten Verlauf. Der Benutzer fordert durch Aktivieren eines Hyperlinks Daten vom Server an. Dieser ermittelt die Quelle des Ereignisses und ruft die dazugehörige Funktion auf. Diese wiederum ruft eine Funktion über Soap auf der Plattform auf. Die zurückgegebenen Daten dieser Funktion bereitet der Webserver auf und aktualisiert den relevanten Teil auf der Webseite, worauf der Benutzer die Änderungen sieht.

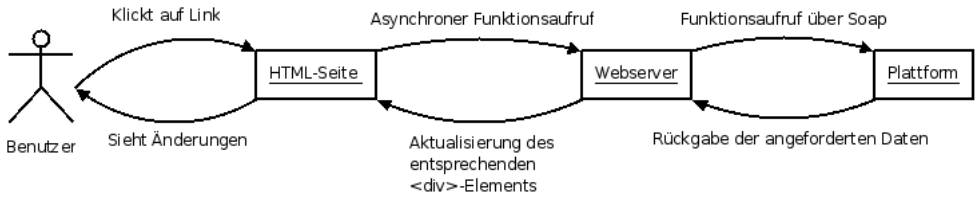

Abbildung 6.1: Benutzerinteraktion mit dem Webinterface

Das folgende Listing zeigt einen Ausschnitt des HTML-Gerüsts. Für die Systeminformationen der Plattform wird innerhalb des *div*-Elements mit der Id *status-content* ein Platzhalter gesetzt.

```
1 <div id="status" style="display: none"
2         class="statusPage dataWindow">
3   <h4>Status</h4>
4   <div id="status-content">
5     <com:TPlaceHolder ID="statusPage" />
6   </div>
7 </div>
```

Der Platzhalter wird durch Benutzerinteraktion mit den Daten der Plattform ersetzt. Dazu wird in die Webseite ein entsprechender Link eingesetzt:

```
1 <li><com:TActiveLinkButton ID="button_status" Text="Status"
2         OnClick="button_clicked" /></li>
```

Ein Klick auf den eingefügten Link aktiviert die folgende Funktionskette im Controller, die den Platzhalter durch die gewünschten Informationen ersetzt. Die Funktion *button_clicked* ermittelt die Quelle des „klicks„, anschließend wird die zugehörige Funktion *showStatus* aufgerufen. In Zeile 12 werden die Daten von der Plattform abgerufen, danach formatiert und schließlich in Zeile 16 auf der Webseite angezeigt.

```
1   function button_clicked($sender, $param){
```

```
2       switch($sender->ID){
3         case "button_status":
4           $this->showStatus();
5           $this->CallbackClient->slidedown("status");
6           break;
7       }
8     }
9
10    function showStatus(){
11      $content = "WSDL: ".$this->soap->getWsdlUrl()."<br/>";
12      $stat = $this->soap->getSysteminfo();
13      $content = $content."# Elemente: ".$stat->elementCount."<br/>";
14      $content = $content."# Hosts: ".$stat->hostCount."<br/>";
15      $content = $content."# VMs: ".$stat->vmCount."<br/>";
16      $this->CallbackClient->update("status-content", $content);
17    }
```

7 Zusammenfassung und Ausblick

Dieses Kapitel gibt einen Überblick über die Arbeit und geht auf die erreichten Ziele sowie mögliche Folgearbeiten ein.

7.1 Überblick über die Arbeit

Ausgehend von der aktuellen Situation beim Management von Virtualisierungslösungen am Lehrstuhl wurden die aktuellen Defizite beim Management dargestellt und ein Anforderungskatalog an eine Managementplattform aufgestellt. Der anschließende Vergleich mit einigen am Markt erhältlichen Tools unterstrich die Notwendigkeit dieser Arbeit, eine Managementplattform zu entwerfen, da kein Tool die Anforderungen hinreichend erfüllte. Die nachfolgende Analyse und Modellierung zeigte die Umsetzung der Anforderungen anhand etablierter Mangementmodelle auf und beschrieb diese. Die Modellierung wurde zu einem Entwurf weiterentwickelt und anhand von UML-Diagrammen dargestellt. Schließlich wurde die Implementierung, die rudimentär erfolgte, kurz beschrieben und die Schlüsselelemente näher erläutert.

7.2 Erreichte Ziele

Die in dieser Arbeit modellierte Managementplattform ist hinsichtlich der Anbindung von Virtualisierungslösungen, Backendkomponenten und Zugriffsmöglichkeiten sehr flexibel gestaltet. Der Einsatz diverser Adaptoren erleichtert die Anbindung der verschiedenen Technologien, damit einhergehend auch die Anbindung neuer, aktuell nicht vorgesehener Komponenten.

Der Entwurf der Plattform enthält nahezu einen identischen Funktionsumfang wie die Modellierung. Zugleich gibt er einen Überblick über die Funktionsabläufe in den Implementierungen von Plattform und Webinterface.

Die Implementierung dagegen deckt nur einen kleinen, aber grundlegenden, Teil der Plattform ab. Sie demonstriert die Umsetzbarkeit der Managementplattform in Code und legt gleichzeitig eine Basis für Weiterentwicklungen.

Tabelle 7.1 zeigt die erreichten Funktionalitäten in den Abschnitten Modellierung, Entwurf und Implementierung auf.

7.3 Erweiterungen / Folgearbeiten

Die folgenden Themen sind für Anschlußarbeiten geeignet:

Vervollständigung der Implementierung Im Rahmen dieser Arbeit ist die Implementierung der modellierten Managementplattform prototypisch erfolgt, d.h. einige wenige Funktionalitäten wurden zu Demonstrationszwecken umgesetzt. Der Großteil der Imple-

Tabelle 7.1: Erreichte Funktionalitäten

	M	E	I
Funktionale Anforderungen			
Hinzufügen/Entfernen von Ressourcen	✓	✓	✓
Konfigurieren der Ressourcen	✓	✓	×
Monitoring der Ressourcen	✓	✓	✓
Dienstgüteüberwachung der Ressourcen	✓	✓	×
Zustandsüberwachung der Ressourcen	✓	✓	×
VM-Migration	✓	✓	×
VM-Steuerung	✓	✓	×
VM-Vorlagen	✓	✓	×
Klonen von VMs	✓	✓	×
Benutzer-/Gruppenverwaltung	✓	✓	×
Benachrichtigungen	✓	✓	×
Autodiscovery	✓	✓	×
Nichtfunktionale Anforderungen			
Einheitliche MIB	✓	✓	✓
Zugriffsberechtigung auf Ressourcen	✓	(✓)	×
Abstrakte Anbindung an Datenbanken	✓	✓	×
Abstrakte Anbindung an Verzeichnisdienste	✓	✓	×
Schnittstelle für externe Anwendungen	✓	✓	✓
Plattformunabhängige Bedienbarkeit	✓	✓	✓
Skalierbarkeit in größeren Umgebungen	✓	✓	×
Führen einer Datenhistorie	✓	✓	×

Abkürzungen

M	Modellierung
E	Entwurf
I	Implementierung
(✓)	(eingeschränkt) unterstützt
×	nicht unterstützt

mentierung, z.B. die Datenbankanbindung, steht jedoch noch aus und kann in diversen zukünftigen Arbeiten (z.B. Praktika) erfolgen.

Authentifizierung und Autorisierung Die Authentifizierung und Autorisierung der Benutzer ist bislang nur modelliert, in Entwurf und Implementierung sind diese Punkte nicht oder nur rudimentär realisiert. Eine Erweiterung dieser Arbeit wäre die Integration der Authentifizierung samt zugehöriger Verzeichnisdienstanbindung, sowie die Autorisierung anhand der modellierten ACLs.

Erweiterung des Berechtigungsmodells Das Berechtigungskonzept sieht für die Komponenten System, Plattform, Gruppe einige spezifische ACLs vor. Als Erweiterung könnten weitere Komponenten, z.B. die verschiedenen Devices, auch in das Konzept eingearbeitet werden.

Loadbalancing-Policies Die Aktivität des Loadbalancers beschränkt sich im bisherigen Modell auf das Wiederherstellen eines stabilen Systemzustandes durch Migration von VMs. Als Erweiterung wären Policies denkbar, welche feingranular festlegen, welche Maßnahmen durchgeführt werden dürfen. Z.B. können in einer Überlastsituation zusätzliche Hostsysteme gestartet werden, oder Dienste mit niedriger Priorität pausiert werden. Auch auf Komponentenebene wären geeignete Maßnahmen, z.B. hinzufügen/entfernen von virtuellen CPUs, eine Überlegung Wert.

Nagios-Adapter Die Integration des Managementsystems in die Monitoringlösung Nagios [nag08] erfordert ein zusätzliches Nagios-Plugin. Die Erstellung eines Plugins wird in [Bar05, Abschnitt C.3.2, S. 431 ff] beschrieben.

Erweiterung der Komponenten Das aktuelle Modell unterstützt nur eine eingeschränkte Menge an Komponenten, aus denen ein System bestehen kann. Xen unterstützt z.B. die eingeschränkte Weitergabe von PCI-Komponenten an eine VM (siehe [RM06, S. 229 ff]). Eine Erweiterung des Modells um weitere Komponenten wäre denkbar.

7.4 Zusammenfassung

Die in dieser Arbeit modellierte Managementplattform bildet die Grundlage für ein flexibles und einheitliches Management von Virtualisierunglösungen. Die Defizite aktueller Lösungen wurden bei der Modellierung berücksichtigt, und das Modell somit robuster gestaltet. Durch die konsequente Anbindung von Adaptern ist die Austauschbarkeit vieler Komponenten gewährleistet, woraus sich nahezu unbegrenzte Erweiterungsmöglichkeiten ergeben. Die hierarchische Struktur der Managementplattformen sowie deren flexibles Rollenmodell bilden gleichzeitig die Basis für eine Skalierbarkeit auch in größeren Umgebungen.

Die Umsetzbarkeit der modellierten Managementplattform ist durch die begleitende Implementierung des Prototypen gezeigt.

A Anhang

A.1 Installation und Betrieb des Prototypen

Dieser Abschnitt beschreibt die Installation sowie den Betrieb des Prototypen.

A.1.1 Voraussetzungen

Der Prototyp benötigt Java in Version 6. Bei Debian ist hier das Paket *sun-java6-jdk* zu installieren.

A.1.2 Auschecken aus Repository

Der Ort des Repositories sowie die Zugangsdaten sind beim Lehrstuhl zu beantragen. Das Vorgehen wird anhand eines Beispiels exemplarisch gezeigt.
Repository: */users/stud/bittner/svn/vm-mgmt* auf *pcheger12.nm.ifi.lmu.de*
Typ des Repositories: Subversion
Checkout:

```
svn checkout svn+ssh://username@pcheger12.nm.ifi.lmu.de/
    users/stud/bittner/svn/vm-mgmt}
```

A.1.3 Compilieren

Der Code wird in das Verzeichnis */users/stud/bittner/bin* compiliert:

```
javac -d /users/stud/bittner/bin /users/stud/svn/vm-mgmt/
    Code/src/Managementplattform2/core/Platform.java
```

A.1.4 Starten

```
java /users/stud/bittner/bin/core.Platform <platform-typ> <log-level>
```

platform-typ gibt die Rolle der Plattform an:

master Plattform startet in Rolle Master (default)

proxy Plattform startet in Rolle Proxy

cproxy Plattform startet in Rolle Caching-Proxy

log-level gibt die Granularität (java.util.logging.LogLevel) des Loggers (java.util.logging.Logger) an:

severe Stellt das Loglevel auf SEVERE.

warning Stellt das Loglevel auf WARNING.

info Stellt das Loglevel auf INFO (default).

config Stellt das Loglevel auf CONFIG.

fine Stellt das Loglevel auf FINE.

finer Stellt das Loglevel auf FINER.

finest Stellt das Loglevel auf FINEST.

A.2 Installation und Betrieb des Webinterfaces

Dieser Abschnitt beschreibt die Installation sowie den Betrieb des Webinterfaces.

A.2.1 Voraussetzungen

Das webinterface benötigt PHP5, Die PHP-GD-Library und einen PHP-aktivierten Webserver. Im Beispiel ist das ein Apache2.

Unter Debian kann man ide benötigten Pakete folgendermaßen installieren:

```
apt-get install apache2 php5 php5-gd
```

A.2.2 Setzen der Berechtigungen für den Webserver

Folgende Befehle sind in */users/stud/bittner/bin /users/stud/svn/vm-mgmt/Code/src/WebUI* auszuführen:

```
chgrp www-data assets -R
chgrp www-data images -R
chgrp www-data protected/runtime -R

chmod 775 assets -R
chmod 775 images -R
chmod 775 protected/runtime -R
```

A.2.3 Erstellen einer Webserverkonfiguration

Die Datei vlgui.conf ist im Verzeichnis */etc/apache2/conf.d* zuerstellen:

```
Alias /vlgui /users/stud/bittner/bin /users/stud/svn/vm-mgmt/Code/src/WebUI

<Directory /users/stud/bittner/bin /users/stud/svn/vm-mgmt/Code/src/WebUI>
        Options Indexes FollowSymLinks
        DirectoryIndex index.php

        <IfModule mod_php5.c>
                AddType application/x-httpd-php .php
```

```
                php_flag magic_quotes_gpc Off
                php_flag track_vars On
                php_flag register_globals Off
                php_value include_path .
        </IfModule>
</Directory>
```

Danach ist der Webserver mit */etc/init.d/apache2 restart* neuzustarten.

Erreichbarkeit des Webinterfaces

Das Webinterface ist auf dem Rechner selbst mittels *http://localhost/vlgui* oder von außerhalb unter *http://rechnername oder -ip/vlgui* zu erreichen.

A.3 Soap mit PHP und Java

PHP ab Version 5 (siehe [Tra05, S.241 ff]) sowie Java ab Version 6 (siehe [FB07, S.1043 ff]) bringen Webservice-Erweiterungen mit.

Folgende Probleme traten bei der Arbeit damit auf:

WSDL-Datei wird nicht aktualisiert Bei Änderungen am Server, z.B. hinzufügen einer neuen Webservice-Methode, wurde die WSDL-Datei beim Client nicht aktualisiert. Grund dafür ist der standardmäßig aktivierte Cache. Die folgenden Soap-Optionen sind in der php.ini zu finden:

```
soap.wsdl_cache_enabled=1
```

aktiviert den Cache. Eine Deaktivierung funktionierte nicht (Soap hat nicht mehr geantwortet)

```
soap.wsdl_cache_dir="/tmp"
```

gibt das Verzeichnis mit der Cache-Datei an. Ein löschen der Datei funktionierte teilweise.

```
soap.wsdl_cache_ttl=86400
```

gibt die Dauer der Gültigkeit des Caches in Sekunden an. Eine Änderung auf 5 löste das Problem.

PHP kann erweiterte Elementdefinitionen nicht laden Bei der Verwendung zusammengesetzter Typen, z.B. Array, erschien die Fehlermeldung

```
Fatal error: Uncaught SoapFault exception: [WSDL] SOAP-ERROR: Parsing WSDL:
Couldn't find <definitions> in 'http://localhost/[...]
```

Das Problem sind hierbei die ausgelagerten Definitionen, standardmäßig liegt die WSDL-Datei in

```
http://localhost:8160/utilities?wsdl
```

zusätzliche Definitionen werden jedoch eingebunden, z.B. aus

```
http://localhost:8160/utilities?xsd=1
```

Dieser Import macht PHP aus unbekannten Gründen Probleme. Ein manuelles zusammenführen der Dateien half jedoch.
Folgende Ausschnitte der Dateien müssen dabei überarbeitet werden:

aus http://localhost:8160/utilities?wsdl:

```
<types>
<xsd:schema>
<xsd:import namespace="http://jaxb.dev.java.net/array"
  schemaLocation="http://localhost:8160/utilities?xsd=1"></xsd:import>
</xsd:schema>
</types>
```

aus http://localhost:8160/utilities?xsd=1:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0"
  targetNamespace="http://jaxb.dev.java.net/array">
<xs:complexType name="stringArray" final="#all">
<xs:sequence>
<xs:element name="item" type="xs:string" minOccurs="0" maxOccurs="unbounded"
  nillable="true"></xs:element>
</xs:sequence>
</xs:complexType>
</xs:schema>
```

in zusammengeführte Datei:

```
<types>
<xsd:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0"
  targetNamespace="http://jaxb.dev.java.net/array">
<xs:complexType name="stringArray" final="#all">
<xs:sequence>
<xs:element name="item" type="xs:string" minOccurs="0"
 maxOccurs="unbounded" nillable="true"></xs:element>
</xs:sequence>
</xs:complexType>
</xsd:schema>
</types>
```

Eine automatische Zusammenführung bietet das Rubyskript *wsdl-merge.rb* welches im Hauptverzeichnisses des Webinterfaces zu finden ist. Einzige Voraussetzung ist die Sprache Ruby [rub08].

Objekte zwischen Java und PHP per Soap übergeben Leider können Objekte nicht standardmäßig per Soap übergeben werden. Folgende Schritte sind zur Objektübergabe notwendig:

1. Objekt in Java definieren: Zuerst muss das Objekt serverseitig (hier: Java) definiert werden:

   ```
   public class Host {
       public String name;
       public String usage;

       public Host(){
       }
   }
   ```

   Wichtig ist hierbei die öffentliche Sichtbarkeit der Klassenvariablen. Ansonsten kann auf diese nicht zugegriffen werden.

2. Anpassen der WSDL-Definition:

   Die WSDL-Datei muss die folgende Objekt-Definition enthalten:

   ```
   <xs:complexType name="host">
     <sequence>
       <element name="name" type="xsd:string"/>
       <element name="usage" type="xsd:string"/>
     </sequence>
   </xs:complexType>
   ```

3. Objekt in PHP definieren: In PHP muss das gleiche Objekt definiert werden:

   ```
   class Host{
     public $name;
     public $usage;
   }

    \item Objekt in PHP mappen:

   \end{enumerate}
   ```

4. Objekte in PHP mappen:

   Danach müssen die Objekte in PHP gemapped werden:

   ```
       $this->wsdl_url = "http://localhost/vlgui/wsdl.xml";
       $classmap = array('host' => 'Host');
       $this->client = new SoapClient($this->wsdl_url,
         array('classmap' => $classmap));
   ```

In diesem Beispiel wird die Soap-Klasse host auf die PHP-Klasse Host gemapped. Die Soap-Klasse host wiederum ist ursprünglich in Java definiert und wird per WSDL-Definition weitergegeben.

5. Objekte auslesen in PHP:

   In diesem Beispiel wird in Java ein Array aus Hosts (Host[]) übergeben. In PHP befinden sich die Hostobjekte in einem Array, welches der Parameter *item* des Objekts *stdClass* enthält. Wie folgt können die Hosts ausgelesen werden:

```
$hosts = $client->getHostList();
$hostlist = $hosts->item;
var_dump($hostlist);
foreach($hostlist as $h){
  echo $h->name." : ".$h->usage."<br/>";
}
```

A.4 UML-Diagramme der Implementierung

Die folgenden UML-Diagramme wurden aus der Java-Implementierung des Prototypen erzeugt (Abbildungen A.1, A.2, A.3, A.4, A.5, A.6).

A.5 Xen 3.2 unter Debian Etch (4.0) compilieren und installieren

1. Benötigte Pakete installieren

   apt-get install gcc make binutils libzzip-dev python-dev libncurses-dev libssl-dev xorg-dev bridge-utils iproute mercurial gettext gawk

2. Xen und benötigten Kernel downloaden und entpacken

   wget http://xenbits.xensource.com/oss-xen/release/3.2.0/xen-3.2.0.tar.gz

   tar -xvzf xen-3.2.0.tar.gz

   hg clone http://xenbits.xensource.com/linux-2.6.18-xen.hg

3. Compilieren

   cd xen-3.2.0 make world

   Achtung: Das Verzeichnis linux-2.6.18-xen muss normalerweise innerhalb des Verzeichnisses xen-3.2.0 liegen. Jedoch löscht make world dieses mit den ersten Befehlen. Man kann jedoch kurz nach Beginn des Compiliervorgangs einen Symlink auf das linux-2.6.18-xen Verzeichnis erstellen, dieser bleibt dann für dieses Compiliervorgang erhalten.

   cd xen-3.2.0

   ln -s ../linux-2.6.18-xen .

4. Installieren

   make install

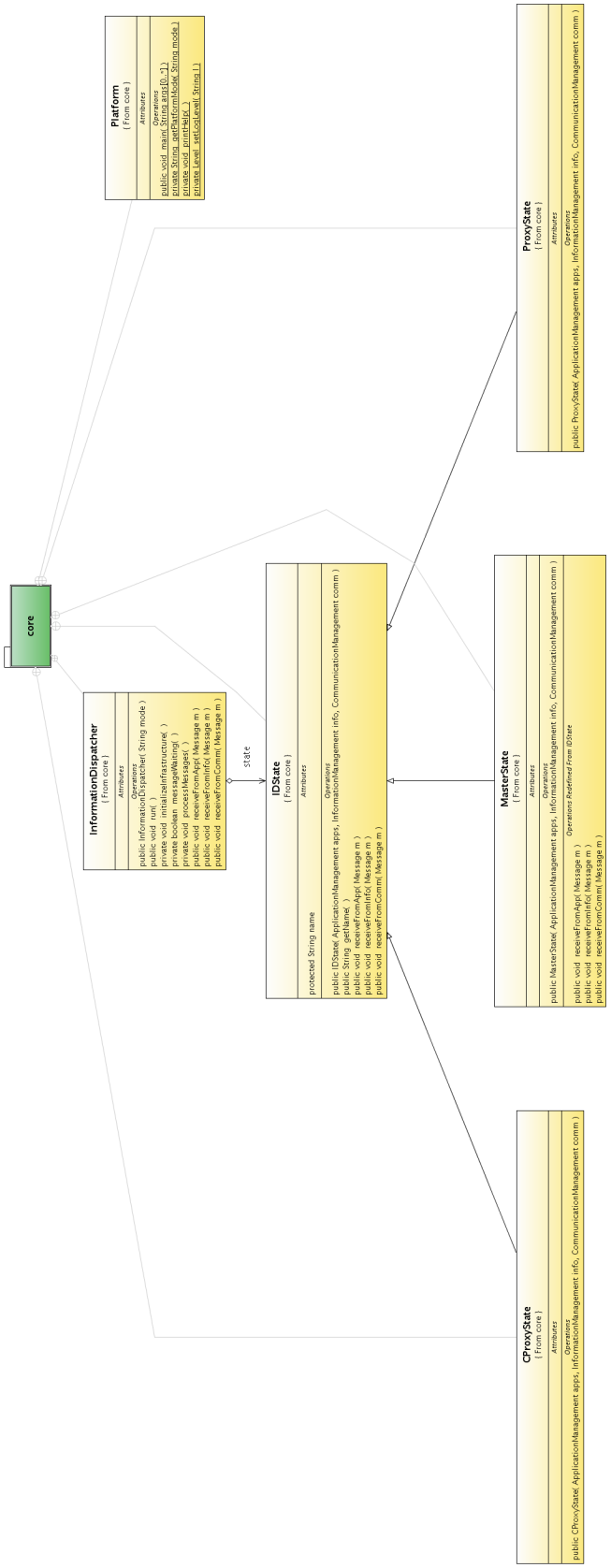

Abbildung A.1: Die Basisklassen der Plattform

5. initrd für den Xen-Kernel erstellen

   mkinitramfs -o /boot/initrd.img-2.6.18.8-xen 2.6.18.8-xen

6. Grub Bootloader für Xen und mit Fehlerbehandlung konfigurieren. Beispiel für eine funktionierende Bootloaderkonfiguration (/boot/grub/menu.lst) mit aktiviertem Fallback auf einen funktionierenden Kernel falls der Xen-Kernel nicht gebootet werden kann:

```
default          saved
timeout          5
fallback  1

title            Xen 3.2 / XenLinux 2.6.18
root             (hd0,0)
kernel           /boot/xen-3.2.0.gz
module          /boot/vmlinuz-2.6.18.8-xen root=/dev/hda1 ro
module          /boot/initrd.img-2.6.18.8-xen
savedefault fallback

title            Debian GNU/Linux, kernel 2.6.18-4-686
root             (hd0,0)
kernel           /boot/vmlinuz-2.6.18-4-686 root=/dev/hda1 ro
initrd          /boot/initrd.img-2.6.18-4-686
savedefault
```

   Für den ersten Startvorgang muss noch der default-Wert per Hand gesetzt werden:

   grub-set-default 0

7. Zugriff auf die Xen-API übers Netz aktivieren

   In die /etc/xen/xend-config.sxp folgenden Eintrag hinzufügen:

   (xen-api-server ((9363)))

8. Migration erlauben

   In die /etc/xen/xend-config.sxp folgenden Eintrag hinzufügen:

   (xend-relocation-hosts-allow ")

9. Xen automatisch starten

   Um Xen nach einem Systemstart automatisch zu starten kann man in die Datei /etc/rc.local folgenden Eintrag vornehmen:

   /etc/init.d/xend start

A.6 Tests

Im Rahmen dieser Arbeit wurde u.a. die Online-Migration mit Xen getestet.

A.6.1 Online-Migration mit Xen

Verschoben wurde vm2 (192.168.2.32) von 192.168.2.25 auf 192.168.2.24 mittels

date;xm migrate –live vm2 192.168.2.24;date

Die Zeitmessung ergab ein Differenz von 13 Sekunden zwischen Beginn und Ende der Migration. Die VM2 enthielt ein minimales Debian etch (4.0) ohne zusätzliche Dienste. Während der Migration erhöhte sich die Latenz der Netzverbindung zur migrierenden VM.

Der Ping auf 192.168.2.32 ergab folgendes Bild:

```
64 bytes from 192.168.2.32: icmp_seq=1 ttl=64 time=2.89 ms
64 bytes from 192.168.2.32: icmp_seq=2 ttl=64 time=1.36 ms
64 bytes from 192.168.2.32: icmp_seq=3 ttl=64 time=0.299 ms
64 bytes from 192.168.2.32: icmp_seq=4 ttl=64 time=1.40 ms
64 bytes from 192.168.2.32: icmp_seq=5 ttl=64 time=0.499 ms
64 bytes from 192.168.2.32: icmp_seq=6 ttl=64 time=91.5 ms
64 bytes from 192.168.2.32: icmp_seq=7 ttl=64 time=91.1 ms
64 bytes from 192.168.2.32: icmp_seq=8 ttl=64 time=92.8 ms
64 bytes from 192.168.2.32: icmp_seq=9 ttl=64 time=92.5 ms
64 bytes from 192.168.2.32: icmp_seq=10 ttl=64 time=94.2 ms
64 bytes from 192.168.2.32: icmp_seq=11 ttl=64 time=93.6 ms
64 bytes from 192.168.2.32: icmp_seq=12 ttl=64 time=95.5 ms
64 bytes from 192.168.2.32: icmp_seq=13 ttl=64 time=94.9 ms
64 bytes from 192.168.2.32: icmp_seq=14 ttl=64 time=96.8 ms
64 bytes from 192.168.2.32: icmp_seq=15 ttl=64 time=96.3 ms
64 bytes from 192.168.2.32: icmp_seq=16 ttl=64 time=97.9 ms
64 bytes from 192.168.2.32: icmp_seq=18 ttl=64 time=1.52 ms
64 bytes from 192.168.2.32: icmp_seq=19 ttl=64 time=0.458 ms
```

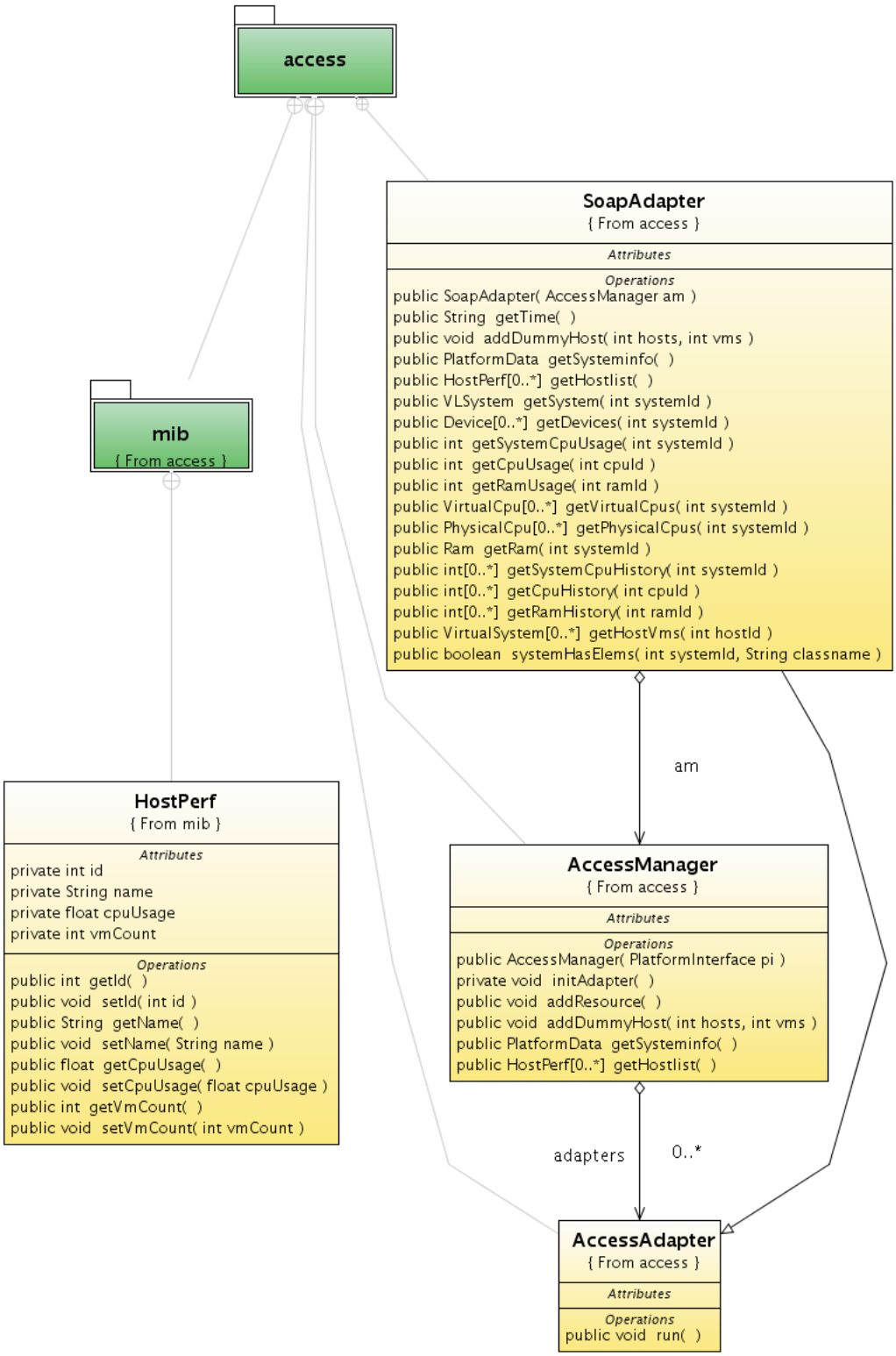

Abbildung A.2: Das Package access

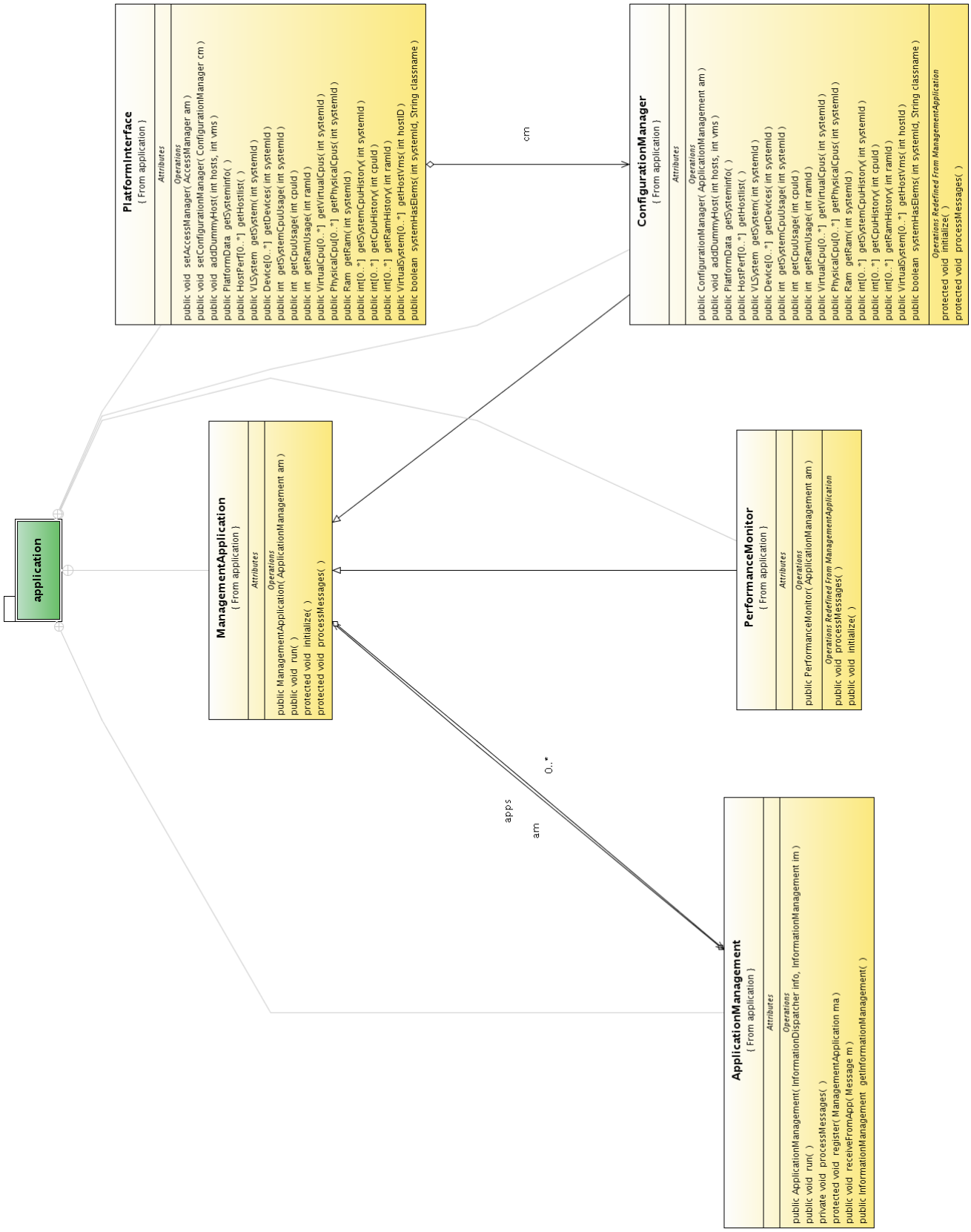

Abbildung A.3: Das Package application

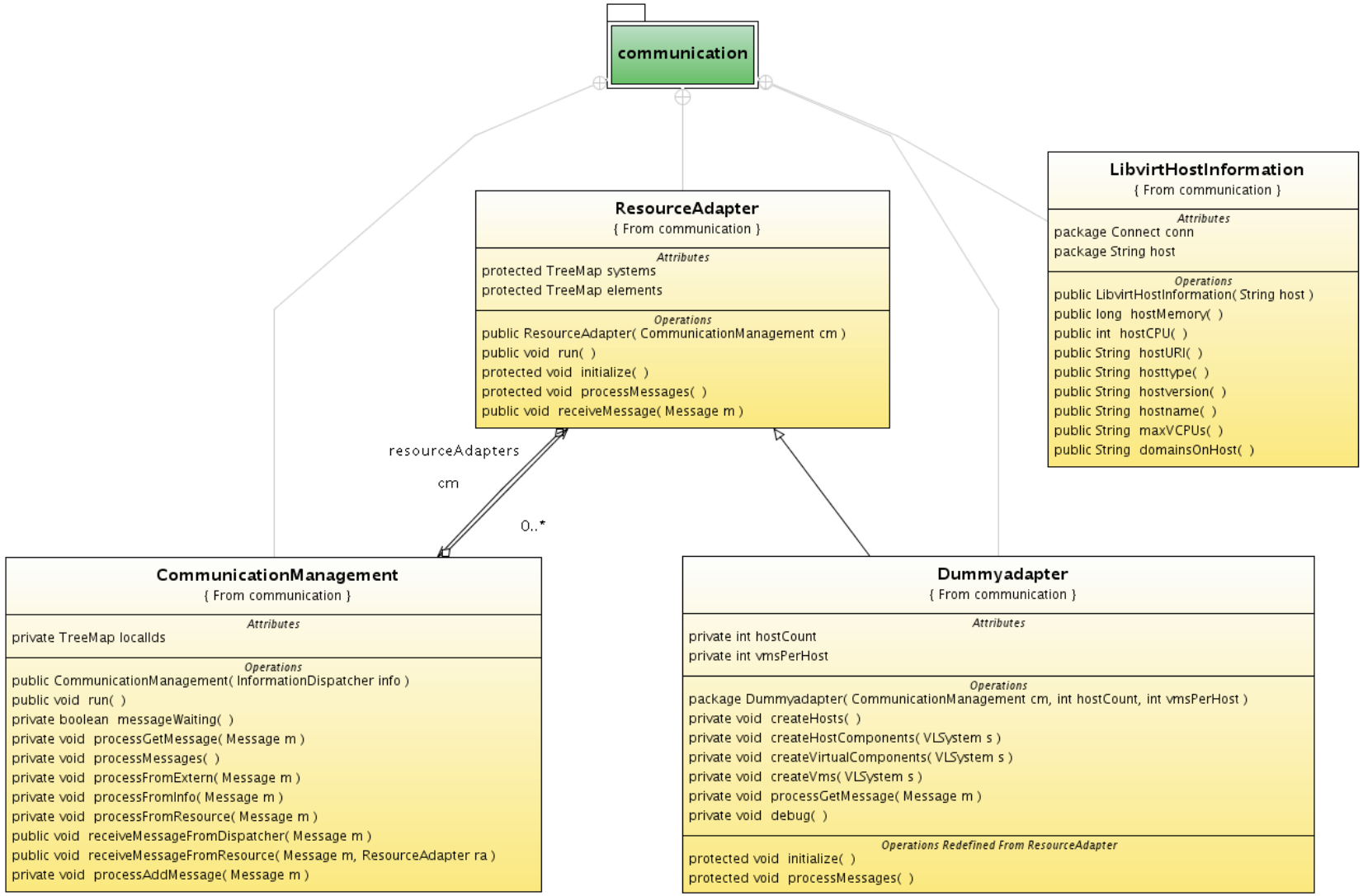

Abbildung A.4: Das Package communication

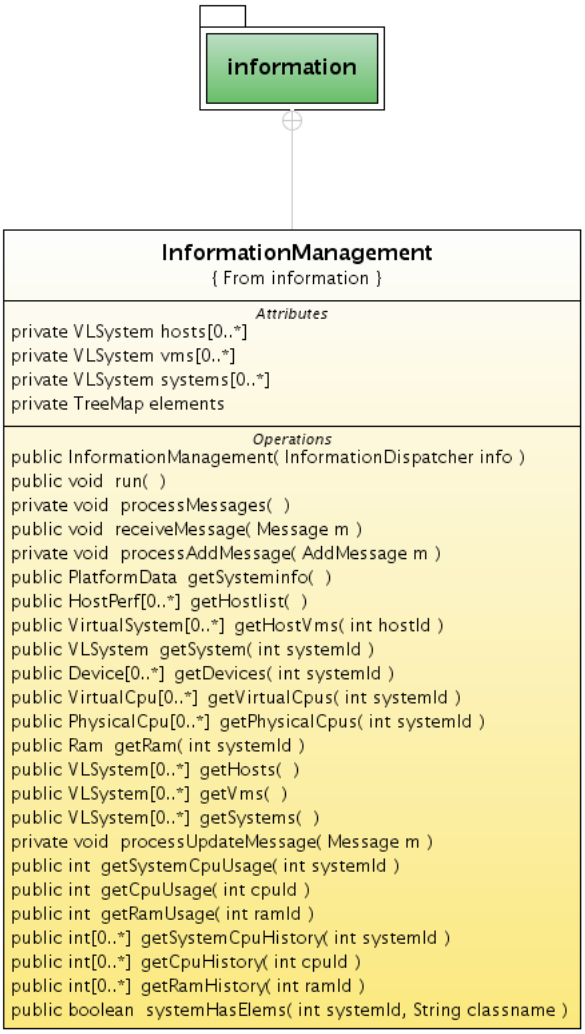

Abbildung A.5: Das Package information

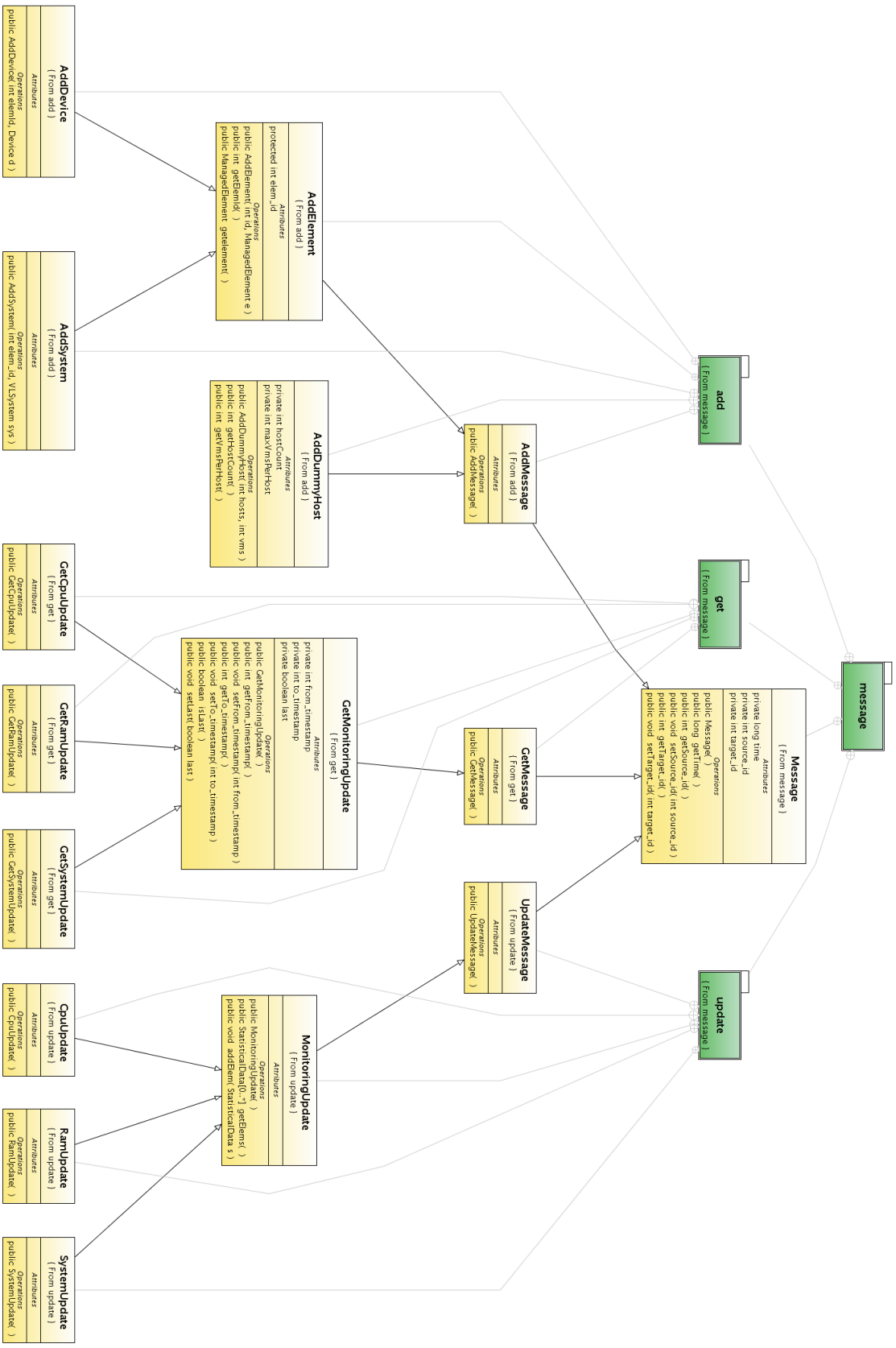

Abbildung A.6: Das Package message

Abbildungsverzeichnis

Literaturverzeichnis

[Bar05] Barth, Wolfgang: *Nagios System- und Netzwerk-Monitoring.* Open Source Press, 2005. 472 S.

[cim08] *Common Information Model (CIM)*, 2008.

[con08] *ConVirt*, 2008.

[eno08] *Enomalism*, 2008.

[fas08] *Entwurfsmuster Fassade*, 2008.

[FB07] Frischalowski, Dirk und Ulrike Böttcher: *Java 6 Programmierhandbuch.* entwickler.press, 2007. 1136 S.

[Gar05] Garrett, Jesse James: *Ajax: A New Approach to Web Applications*, 2005.

[HAN99] Hegering, H.-G., S. Abeck und B. Neumair: *Integriertes Management vernetzter Systeme — Konzepte, Architekturen und deren betrieblicher Einsatz.* dpunkt-Verlag, ISBN 3-932588-16-9, 1999. 607 S.

[HM08] Hauser, Michael und Marcel Michelmann: *Entwicklung von Adaptoren zum Management von Hostvirtualisierungslösungen*, 2008.

[ibm08] *IBM Virtualization Manager*, 2008.

[kvm08] *KVM*, 2008.

[lib08] *Libvirt*, 2008.

[Lin06] Lindinger, Tobias: *Virtualisierung einer Praktikumsinfrastruktur zur Ausbildung im Bereich vernetzter Systeme*, 2006.

[nag08] *Nagios Website*, 2008.

[nis08] *NIS*, 2008.

[ope08] *OpenLDAP*, 2008.

[pra08] *PHP-Framework Prado*, 2008.

[RM06] Radonic, Andrej und Frank Meyer: *XEN 3.* Francis Verlag, 2006. 439 S.

[rub08] *Ruby*, 2008.

[sid08] *TM Forum Information Framework (SID)*, 2008.

[Tra05] Trachtenberg, Adam: *Umsteigen auf PHP5.* O'Reilly, 2005. 330 S.

[vir08] *VMware Virtual Center*, 2008.