INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Diplomarbeit

Klassifizierung und Modellierung von Dienstmanagement-Informationen - ein Design-Pattern basierter Ansatz

Cyril Bitterich

Aufgabensteller: Prof. Dr. Heinz-Gerd Hegering

Betreuer: Martin Sailer
Michael Schiffers

Abgabetermin: 19. Dezember 2005

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Diplomarbeit

Klassifizierung und Modellierung von Dienstmanagement-Informationen - ein Design-Pattern basierter Ansatz

Cyril Bitterich

Aufgabensteller: Prof. Dr. Heinz-Gerd Hegering

Betreuer: Martin Sailer
Michael Schiffers

Abgabetermin: 19. Dezember 2005

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

München, den 19. Dezember 2005

. . . . . .

(Unterschrift des Kandidaten)

Zusammenfassung

Als die ersten Computer in Firmen Verwendung fanden, waren dies große, schwere und langsame Maschinen, die von sogenannten Operatoren bedient wurden. Deren einzige Aufgabe war es, die autonom arbeitenden Großrechner mit Informationen zu füttern, die Ergebnisse an die Auftraggeber zurückzugeben und die großen Rechner am Laufen zu halten. Mittlerweile hat sich das Verhältnis gewandelt und es sind nur wenige Administratoren, die oft hunderte von Rechnern zu verwalten haben. Zudem sind diese Rechner mittlerweile untereinander vernetzt und dadurch auch nicht mehr vollkommen unabhängig voneinander. Durch diese Möglichkeiten haben sich auch die Anforderungen geändert. War es ganz am Anfang ausreichend, sich um jede einzelne Maschine zu kümmern, so kam mit der Vernetzung die Erkenntnis auf, dass häufig gleich ganze Netze zu betrachten waren, wann immer eine Änderung an den Systemen erfolgen sollte.

Diese Entwicklung hat sich fortgesetzt und die Betrachtungsweise hat sich mit der wachsenden Benutzerfreundlichkeit der Rechenmaschinen von der rein technischen Betrachtung zu einer anwendungsorientierten Betrachtungsweise hin gewandelt. Heutzutage ist für den Kunden die technische Lösung einer Anwendung in der Regel ohne Bedeutung. Ihn interessiert, dass die Dienstleistung, die er sich vom Computer erwartet, erbracht wird. Das “Wie” ist Aufgabe der Anbieter.

Um diesen Anforderungen gerecht zu werden, wurde der Begriff des Dienstmanagements eingeführt. Diese Art des Managements verwendet eigene Informationen, ja zum Teil auch eine eigene Sprache, die in Einklang mit den Informationen und der Ausdrucksweise der technischen Realisierung gebracht werden müssen. Im Rahmen dieser Arbeit wird ein Objektkatalog erstellt, der es ermöglicht, Dienste als ebensolche unabhängig von der darunterliegenden Infrastruktur zu verwalten. Gleichzeitig wird jedoch Wert auf eine einfache Abbildbarkeit der Attribute und Methoden dieses Modelles auf die Attribute und Methoden der darunterliegenden Komponenten gelegt. Ausgangsbasis werden dabei die Managementarchitekturen aus ITIL und OSI sein, die sich von ihrem Prinzip her bewährt haben. Um auch bei der Modellierung auf Bewährtes zurückgreifen zu können, werden hierfür Design Pattern Anwendung finden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Das Management der IUK-Systeme ändert sich bereits seit seinen Anfängen und trägt damit der immer weiter zunehmenden Komplexität der IUK-Landschaft Rechnung. Zu Beginn wurde jedes Gerät (Device) einzeln für sich betrachtet und dementsprechend auch verwaltet und konfiguriert (Device-Management).
Mit der zunehmenden Vernetzung und dem Einsatz von Serverfarmen an Stelle von einzelnen großen "Hosts", die nur von einem "Operator" bedient wurden, stellte sich in den 90'er Jahren heraus, dass eine neue Form der Verwaltung benötigt wurde. Um den Abhängigkeiten der Einzelsysteme und auch der Tatsache, dass oft eine ganze Reihe von Systemen bearbeitet werden musste, entgegenzukommen, wurde das Netzwerkmanagement entwickelt. Bei diesem wurde von den einzelnen Geräten abstrahiert und das Augenmerk auf Funktionalitäten gelegt, die die Geräte jeweils auszuführen hatten.

Mit der weiteren Entwicklung nahm dann jedoch auch die Interaktion der einzelnen Netze zu. Zudem ist die physische Struktur spätestens seit der Entwicklung der VPNs (über öffentliche Leitungen geführte Netze, die logisch als eigenständige Netze auftreten und so zum Beispiel Außenstellen von Firmen über das Internet verbinden) nicht mehr mit der logisch zu verwaltenden gleichzusetzen. Auch hat sich die Sichtweise des "Benutzers" mehr und mehr geändert. War früher die IT noch ein Bereich im Haus, der man ehrfürchtig seine Aufgaben übergab, weil sich sonst niemand mit den komplizierten "Rechenmaschinen" auskannte, so wird heute mehr und mehr der "Benutzer" als "Kunde" im Rahmen eines Businessprozesses angesehen. Auf der anderen Seite wurde die für die IT zuständige Abteilung zu einem "Dienstleister", welche ihre Kosten nach den Maßstäben der Betriebswirtschaft mit einem monetären Kostenvorteil für die Firma zu begründen hat.
Durch das sogenannte Outsourcing verschwinden auch die Grenzen des "internen" zum "externen" Kunden immer mehr. Diese Änderungen der Struktur schlagen auch auf das Management der IT-Systeme durch. Deren Verwaltung kümmert sich um die Schnittstellen, welche dem Kunden angeboten werden. In den meisten Fällen ist dies schlicht und ergreifend ein Dienst (Service), der für ihn erbracht wird. Ob dies nun die Berechnung einer komplexen Formel (CPU-Time), das Anbieten von Speicherplatz in einem NAS oder einfach nur das Drucken einer Webseite ist; für den Benutzer ist es einfach ein Dienst, den er - ohne sich Gedanken über Ort, Netzwerk oder gar physikalisch benutzte Maschine zu machen - nutzen möchte.

1.1 Aufgabenstellung

Ursprünglich wurden Managementsysteme auf der Basis der zu befriedigenden Bedürfnisse entwickelt. Dafür wurden die verfügbaren Informationen zusammengefasst und "bottom-up" Managementsysteme erstellt. Mit dieser Vorgehensweise entstand eine Vielzahl von Systemen, welche sehr genau auf die zum Zeitpunkt der Erstellung benötigten jeweiligen Anforderungen zugeschnitten waren.
Solange nur eine geringe Anzahl von Komponenten zu verwalten war, war dieser Ansatz vollkommen ausreichend. In den letzten Jahren hat jedoch die Zahl der zu verwaltenden Systeme zugenommen. Zusätzlich müssen die Abhängigkeiten von Systemen aus verschiedenen Organisationen beachtet werden. Dabei zeigte sich, dass die Integration der verschiedenen Managementsysteme oft nicht möglich war, da die Anforderungen auf welchen sie basierten, zu unterschiedlich waren.

Nachdem die Vernetzung und Komplexität der Systeme zunahm, mussten neue Ansätze geschaffen werden, welche alle möglichen Komponenten zueinander in Beziehung stellen konnten. Dazu wurde die "top-down" Sichtweise des Managements der Firmen in den Vordergrund gestellt. Gleichzeitig entwickelte sich die Betrachtung der zu verwaltenden Objekte weg von den einzelnen Komponenten hin zu den erbrachten Diensten.
Aufgrund der abstrahierten Betrachtung der "top-down" Ansätze ist es möglich, mit diesen beliebige Dienste zu modellieren. Bei dem Versuch, die Ansätze anzuwenden, zeigte sich jedoch, dass die praktische Anwendung noch nicht möglich war. Es fehlten schlicht und ergreifend die Dienstinformationen, welche die tatsächliche Ausprägung der Dienste ausmachen.

Aufgabe der vorliegenden Arbeit ist es, einen Ansatz zu schaffen, welcher zwar allgemein genug ist, um alle Dienste darstellen zu können, gleichzeitig aber auch speziell genug, um die Ausprägungen der Dienste ebenfalls modellieren zu können.
Hierfür werden verschiedene "top-down" Ansätze analysiert, um daraus grundlegende Informationen zu bestimmen, welche für die detaillierte Darstellung eines Dienstes notwendig sind. Mit Hilfe dieser Dienstinformationen wird dann "bottom-up" ein einer Management Information Base (MIB) entsprechender Objektkatalog erstellt werden. Dieser ist so zu erstellen, dass er für beliebige Dienste verwendbar ist und Dienste verschiedener Organisationen auf einer gemeinsamen Basis verwaltbar sind, ohne dass ein gemeinsames Managementsystem verwendet werden muss. Die Modellierung sei dabei durch Design Pattern zu unterstützen.
Abschließend ist der entstandene Objektkatalog noch an Beispielen aus der Praxis zu überprüfen.

1.2 Vorgehensweise in dieser Arbeit

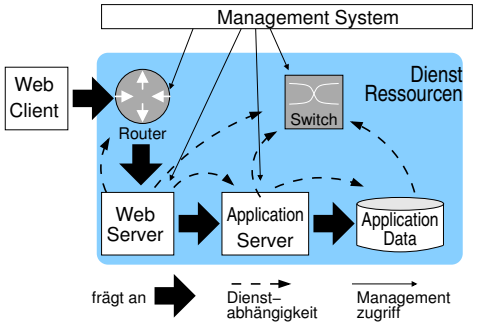

Abbildung 1.1: Beispiel für Dienste und dazugehörige Abhängigkeiten

Das Beispiel in Abbildung 1.1 stellt den typsichen Aufbau eines Dienstes mit seinen Abhängigkeiten dar.Der Benutzer sieht mit seinem WebClient einen WebServer, welcher ihm bestimmte Informationen darstellt. Diese Informationen beruhen aber auf Daten, welche aus einer Datenbank stammen und durch einen Applikationsserver aufbereitet wurden. Dadurch ergibt sich eine Abhängigkeit des Dienstes, welchen der Webserver anbietet, vom Applikationsserver. Die Abhängigkeiten gehen jedoch noch weiter, da die Zusammenarbeit der drei Server nur dann funktionieren kann, wenn sie über einen Switch verbunden werden. Und die Erreichbarkeit des Webservers ist in dem Moment nicht mehr gegeben, wenn der Router, der die Verbindung herstellt ausfällt.
Ausgehend von diesem Szenario, illustriert an einem Beispiel aus der Praxis, werden zunächst die funktionalen und organisatorischen Anforderungen bestimmt, die an eine Management Information Base (MIB) für ein Service-Managementsystem gestellt werden.

Parallel dazu erfolgt eine Informationsanalyse auf Basis verschiedener Fragestellungen des Szenarios, den Büchern zum Management nach der IT-Infrastructure Library (ITIL), sowie durch eine Aufsplittung der Begriffe hinter dem Acronym FCAPS des OSI-Modelles der International Organization for Standardization (ISO). Diese wird abgeschlossen durch die Bestimmung der zentralen, für das Service-Management wichtigen Attribute.

Zudem werden unter Berücksichtigung des Basisszenarios verschiedene Modellierungsansätze (unter anderem Internet Management (SNMP), CIM, SID) und -verfahren betrachtet, welche derzeit als Standard existieren oder als zukünftige Standards in Frage kämen. Auch wird untersucht, ob andere wissenschaftliche Ansätze das Problem bereits ganz oder teilweise beheben.
Aus einem Vergleich des “State of the Art” mit den betrachteten Ansätzen folgt eine Analyse der Defizite und Stärken der jeweiligen Ansätze.

Aufbauend auf dem Vorhergehenden wird dann eine Kategorisierung der einzelnen Attribute, sowie der Versuch einer Abbildung der hier verwendeten Attribute zu den in bereits bestehenden Ansätzen verwendeten gemacht. Auch erfolgt eine letzte Auslese bezüglich der Verwendbarkeit in einem allgemeinen Schema.

Anschließend wird dann unter Verwendung von Design-Pattern ein Objektkatalog für das Ursprungsszenario aufgebaut.

In einem letzten Schritt wird der vorgestellte Objektkatalog an dem Beispiel und dem aus dem LRZ übernommenen Szenario übernommen.

1.3 Gliederung der Arbeit

Die im vorherigen Kapitel vorgestellte Vorgehensweise bei der Entwicklung der Arbeit wird in folgende Gliederung überführt.

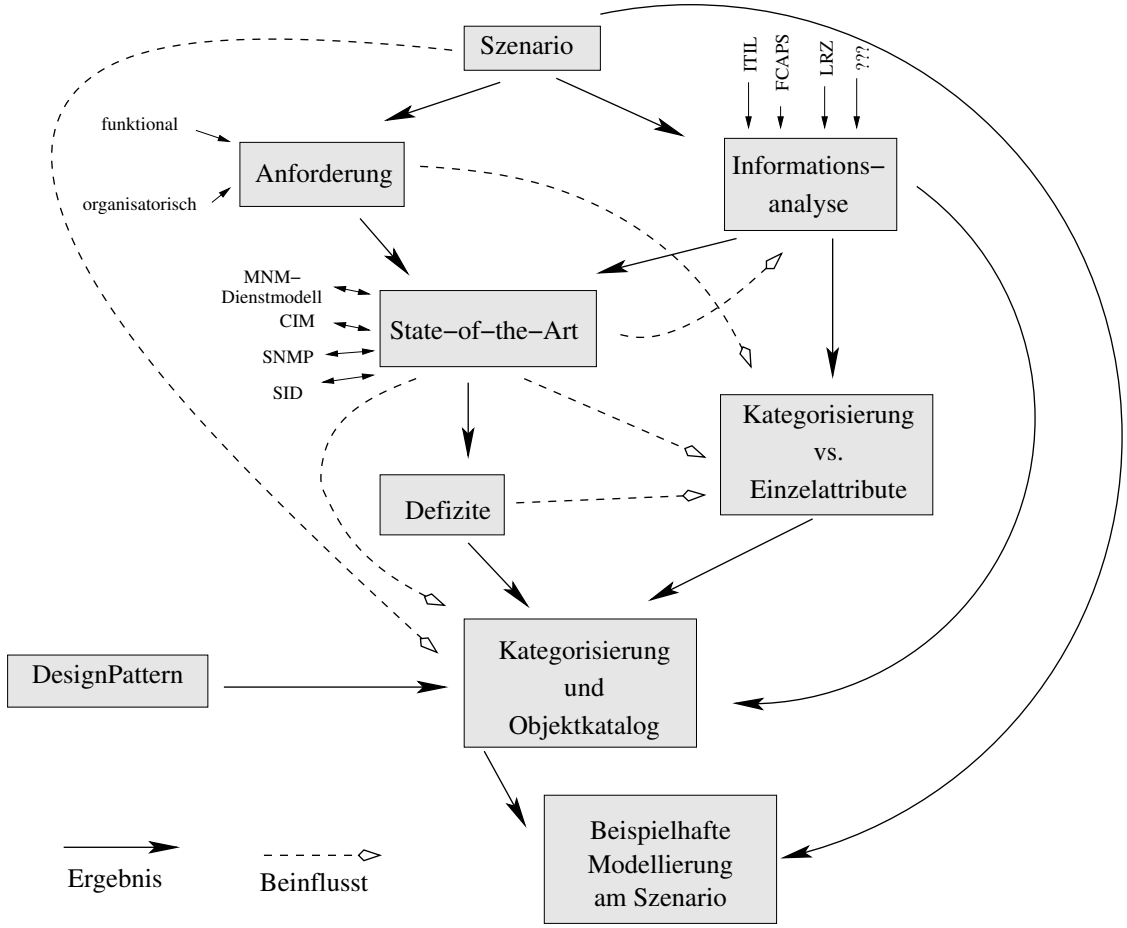

Abbildung 1.2: Strukturelles Vorgehensmodell

Zu Beginn der Arbeit wird in Kapitel 2.1 ein für den aktuellen Stand typisches Szenario vorgestellt, bei dem das Service-Management noch nicht eingeführt ist. Auch auf die dabei auftretenden Probleme wird kurz eingegangen werden und es wird ein kleiner Katalog an Problemen aufgestellt, deren Lösung man sich dort durch das Service-Management erhofft. Des weiteren wird eine Reihe von Anforderungen aufgestellt, die für ein allgemeines Servicemanagement von Bedeutung sind und an welchen sich die existierenden Ansätze messen lassen müssen.

Diese Ansätze werden in Kapitel 3 vorgestellt. Der Schwerpunkt der Betrachtungen wird dabei auf der Anwendbarkeit für das Dienstmanagement nach den allgemeinen Anforderungen liegen. Das Ergebnis dieser Betrachtungen wird am Ende noch einmal zusammengefasst.

Darauf aufbauend wird in Kapitel 4 anhand zweier bestehender Standards ein Fragenkatalog erstellt, der die Anforderungen an ein Dienstmanagementsystem durch konkrete Fragestellungen beleuchten soll.

Anhand dieser Fragestellungen wird in Kapitel 5 eine Attributssammlung und ein erster Klassendiagramm für den Objektkatalog erstellt.

Dieses wird dann in Kapitel 6 unter Zuhilfenahme von "Design Pattern" vereinfacht. Als Ergebnis wird ein allgemeingültiger Ansatz mit Hinweisen auf die Umsetzung in Projekten entstehen.

In Kapitel 7 wird der Objektkatalog auf die vorgestellten Szenarion angewendet. Anschliesend wird die Lösung einer Managementaufgabe mit Hilfe des Objektkataloges vorgestellt.

Abschließend wird in Kapitel 8 noch einmal eine kurze Reflektion über den entstandenen Objektkatalog stehen, sowie Hinweise auf mögliche Verbesserungen gegeben.

1.4 Begriffsbestimmungen

In dieser Arbeit gibt es ein paar Begriffe, welche nicht klar definiert sind oder im diesem Rahmen in einer besonderen Art und Weise verwendet werden. Um zu verdeutlichen, wie die einzelnen Begriffe innerhalb dieser Arbeit verwendet werden, erfolgt zunächst einmal eine Begriffsbestimmung.

"Dienst" (engl. Service) Kaum ein Begriff unterliegt so vielen verschiedenen Bedeutungen wie der des Dienstes. Im Folgenden soll unter einem "Service" ein System verstanden werden, das über eine Schnittstelle anderen "Services" (oder direkt dem Endanwender) die Lösung von Aufgabenstellungen ermöglicht. Hierbei ist es unerheblich, ob der "Service" eine ausgefeilte Darstellung einer Webseite mit Datenbankanbindung und "on the fly"-Berechnung eines Modells ist oder lediglich die Durchleitung der Datenpakete, die eben diese Webseite zum Rechner des Benutzers bringen.

Komponente Ein weiterer Begriff, welcher ein größeres Bedeutungsspektrum haben kann, ist der der Komponente. Im Folgenden sei dieser Begriff in seiner allgemeinsten Form als Bestandteil eines Ganzen verstanden. Dabei ist die Stofflichkeit der "Komponente" nicht von Belang. So ist zum Beispiel ein Lenkrad ebenso eine Komponente eines funktionierenden Autos, wie ein Vokal in der Sprache Teil eines Wortes ist. Auf die IT abgebildet ist sowohl ein Rechner als auch die Darstellung einer Graphik im Web. Seine Begrenzung findet diese weite Auslegung darin, dass eine "Komponente" immer einen technischen Aspekt hat.

Organisation Mit Organisation sei im Folgenden jede Menge von Menschen gemeint, welche nach außen hin eine Einheit bilden. Dabei kann eine Organisation wiederum weitere Organisationen umfassen. So ist sowohl die IT-Abteilung innerhalb einer Firma eine Organisation, welche ihre Dienste anderen Abteilungen anbietet, als auch die Firma selbst. Auch die Zusammenarbeit zweier IT-Abteilungen verschiedener Firmen sei als Zusammenarbeit zweier Organisationen anzusehen. Anders ausgedrückt, definiert der Begriff eine Einheit von (juristischen und natürlichen) Personen.

Incident - Störung Ein weiterer Begriff mit mehrfacher Bedeutung ist der englische Begriff **incident**. Tatsächlich umfasst er den weiten Bereich eines eigenständigen Ereignisses bis hin zu einem Vorfall, der einer Krise vorangeht. Im Nachfolgenden wird der Begriff der **Störung** verwendet werden, da innerhalb des IT-Managements meistens ein Vorfall erst dann von Interesse ist, wenn er sich störend auf den Ablauf der Infrastruktur auswirkt.

Kapitel 2

Szenario

Das Leibniz RechenZentrum (LRZ) ist für den Betrieb der Infrastruktur des Münchener WissenschaftsNetzes (MWN) zuständig, welches die Vernetzung der wissenschaftlichen Organisationen in München und Umgebung ermöglicht. Neben der Verwaltung des Netzwerkes mit über 60 Standorten und mehr als 50.000 angeschlossenen Geräten, bietet das LRZ jedoch noch weitere Dienste für die angeschlossenen Institutionen auf einer höheren Ebene an. Mit einem verteilten Dateisystem für über 30.000 Benutzer, sowie einem Storage Area Network (SAN) mit 100 Terrabyte Online-Plattenspeicher zählt es zu den größeren Rechenzentren in Deutschland.
Für die Verbindung der Welt zu den Universitäten betreibt es neben den Mailservern für alle Studenten und Mitarbeiter der Ludwig Maximilians Universität München und der Technischen Universität München auch die Nameserver und Directoryserver für das MWN. Als weiteres Angebot werden auch - bis auf wenige Ausnahmen - die Webserver der Institute der angeschlossenen Universitäten und universitätsnahen Einrichtungen durch das LRZ betrieben.

Dadurch stellt das LRZ im Bereich des Webhostings ein Rechenzentrum mittlerer Größe dar, was Probleme in dieser Organisation auch in anderen Organisationen dieser Größenordnung wahrscheinlich macht. Es ist davon auszugehen, dass Lösungen, die in diesem Umfeld gefunden werden, auf viele Organisationen weltweit angewendet werden können.

2.1 Aufbau der Beispielumgebung am LRZ

Das **LRZ** betreibt für die öffentliche Darstellung der Teilnehmer des MWN einen Pool von 20 Webservern, welcher eine möglichst unterbrechungsfreie Außendarstellung der Institute mit ca. 300 virtuellen Webservern ermöglichen soll. Hierzu werden jeweils zwei Rechner so eingerichtet, dass sie auf unterschiedlichen Netzpfaden erreichbar sind, aber die selben Daten ausliefern. Zunächst treffen beim LRZ eingehende Pakete auf einen von zwei Level2/Level4 Switches (Router) der Firma ServerIron (später eventuell das Produkt Big IP der Firma f5). Davon ist normalerweise nur einer aktiv, während der andere im HotStandBy-Modus auf einen Ausfall des Partners wartet, um dann übernehmen zu können. Diese Maschinen agieren als Load-Balancer und sorgen für eine Verteilung der Anfragen auf die Webserver. Direkt an diese ist ein Notserver angeschlossen, der für den Fall eines größeren Ausfalles so konfiguriert ist, dass er eine Benachrichtigungsseite darstellt, welche auf die Störung hinweist. An diese Eingangsswitches sind redundant zwei weitere Level2 Switches angeschlossen,

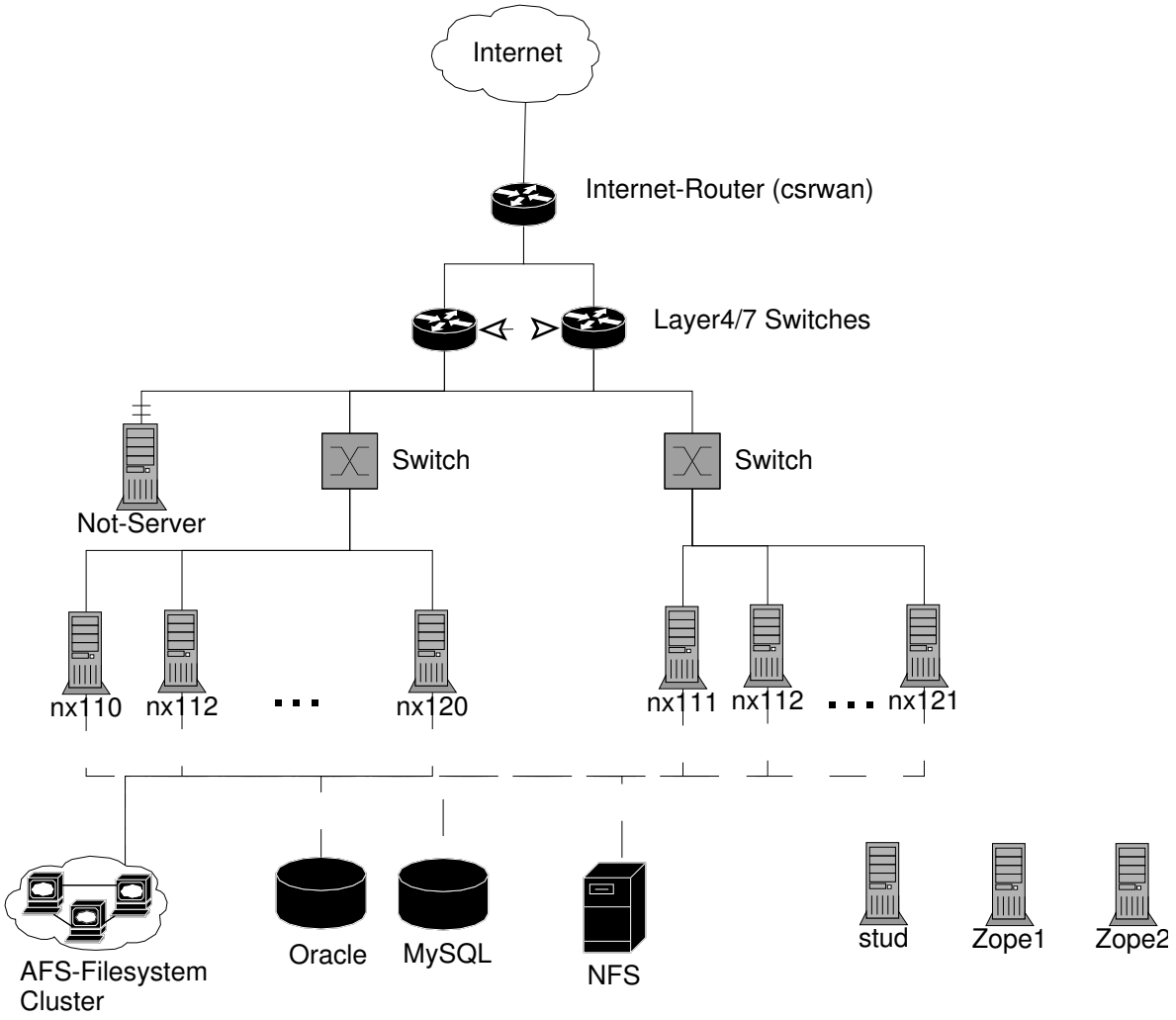

Abbildung 2.1: Infrastruktur zur Auslieferung von Webseiten am LRZ

welche die eigentliche Anbindung der Webserver realisieren. An jedem der Switches sind jeweils 10 Webserver angeschlossen. Diese sind mit Ausnahme der darauf laufenden Webserver gleich ausgestattet und konfiguriert. Jeder dieser Server hat zudem per Andrew File System (AFS) Zugriff auf die Fileserver (nähere Informationen hierzu unter [LRZ-AFS]), einen Oracle- und einen MySQL-Server. Des weiteren können über einen Network File Service (NFS)-Server ein gemeinsames temporäres Verzeichnis (welches als $POOLS_TMP deklariert ist) und die Logdateien für eine Auswertung angesprochen werden. Je zwei dieser Server, welche an unterschiedlichen Switches angeschlossen sind, bilden ein Pärchen, auf welchem die gleichen "Dämonen" laufen. Des weiteren existieren ein Server für die studentischen Webseiten und zwei Zope-Server.

2.1.1 Konfiguration der Webdienste

Nicht alle Webseiten haben die gleichen Ansprüche. Einige davon beruhen auf PHP, andere wiederum benötigen Skripte und wieder andere sind mit einem ungepatchten ("vanilla") Apache zufrieden. Um

die Sicherheit zu erhöhen, ist es unerwünscht, dass allen Webseiten alle Möglichkeiten zur Verfügung stehen. Es werden deshalb vier verschiedene Apache-Webserver gebaut, die sich in ihrer Zusammensetzung unterscheiden. Im Sprachgebrauch der für die Webserver zuständigen Arbeitsgruppe werden sie als "Dämonen" bezeichnet.

lrz

Der "Dämon" mit der Bezeichnung "lrz" ist auf die Anforderungen der Webseiten des LRZ augelegt. Um Sicherheitsprobleme durch eingeschleusten Code oder nachträglich eingefügte Module zu minimieren, ist in dieser Variante der Apache statisch kompiliert.

virt

Ein anderer "Dämon" trägt den Namen "virt" und ist darauf ausgelegt, die virtuellen Webserver der Kunden darzustellen. Hier ist ebenfalls wieder alles statisch kompiliert, aber die Modulauswahl (unter anderem php, suexec und fastcgi) ist etwas anders ausgelegt als beim "lrz-Dämon".

spez

Wieder andere Dienste benötigen eine ganz spezielle Umgebung. Zu diesen gehören der Webmaildienst und das Angebot, mit einer Webschnittstelle namens "phpmyadmin" auf einen Datenbankserver zuzugreifen. Da ihre Anforderungen es verbieten, mit statisch einkompilierten Modulen zu arbeiten, sind diese bei dem Modul "spez" per "Dynamic Shared Objects" (DSO) eingebunden.

ars

Eine andere Software, welche besondere Anforderungen an die Ablauf-Umgebung stellt, ist das Action Request System (AR System) der Firma Remedy. Diese für das Fehlermanagement und die Dokumentation von Hard- und Software verwendete Programmsuite benötigt zum Beispiel zwingend das Modul mod_perl. Dies führte zu der Entscheidung, diesem System ebenfalls einen eigenen "Dämon" zu bauen, welcher die Bezeichnung "ars" trägt.

Um auf unvoherhergesehene Ereignisse schneller reagieren zu können, ist die Hardware und die darauf installierte Software inklusive aller Dämonen für alle Server identisch ausgelegt, was bedeutet, dass auf allen Rechnern auch alle "Dämonen" aufgespielt sind. Grundsätzlich ist aber auf jedem Serverpärchen immer nur einer aus der oben vorgestellten Gruppe im Betrieb. Diese Pärchen sind so ausgelegt, dass je ein Rechner über einen anderen (Level2-) Switch an das Internet angebunden ist. Die einzige Ausnahme bildet der "ars" Dämon, der momentan auf dem selben Serverpaar aktiv ist, wie der "spez" Dämon. Da beide eine geringe Belastung für den Server bedeuten resultieren daraus keine Nachteile.

Das Kompilieren der "Dämonen" erfolgt über make-Files. Dadurch wird der Arbeitsaufwand verringert, der beim Neukompilieren der "Dämonen" anfällt. Dies ist zum Beispiel immer dann notwendig wenn eine neue - fehlerbereinigte - Version eingespielt werden soll. Die Verwendung der make-Files führt auch dazu, dass der Ablauf der Konfiguration für das Kompilieren immer der gleiche ist.
Die Abhängigkeiten der einzelnen "Dämonen" werden in einzelnen Textdateien an verschiedenen Orten abgelegt. Auch die vorgenommenen Änderungen an den "Dämonen" sowie zwischen den Software-Versionen werden als Textdatei gespeichert.

2.1.2 Verwaltung der Komponenten am LRZ

An diesem Teilbereich des LRZ arbeiten bereits mindestens fünf voneinander unabhängige Arbeitsgruppen. (Die im Folgenden gewählten Bezeichnungen der Arbeitsgruppen stimmen nicht mit den tatsächlichen Bezeichnungen innerhalb des LRZ überein!)

1. **Netz**
   Die Netzgruppe hat die Aufgabe, für die Anschaffung, die Inbetriebnahme und auch die Aufrechterhaltung des Betriebes von allen Netzwerkkomponenten und den damit verbundenen Diensten zu sorgen. In dem vorgestellten Szenario (Abbildung 2.1) ist sie für die Router und Switches, sowie den darauf laufenden IP-Dienst zuständig.
2. **Hardware**
   Die Hardwaregruppe ist für die Beschaffung, Inbetriebnahme und Wartung der Rechner verantwortlich, auf welchen weitere Dienste aufgesetzt werden. Diese Gruppe übergibt die Rechner mit einem lauffähigen Betriebssystem (im Falle der Webserver ist dies Solaris der Firma Sun Microsystems, Inc.) an die Arbeitsgruppen, welche die höheren Dienste (AFS, Web, etc.) betreuen. Danach bleibt sie natürlich auch weiterhin zuständig für alle Konfigurationsänderungen und die Behebung von Störungen.
3. **AFS**
   Ein weiterer Dienst, der innerhalb des LRZ eine weite Verbreitung hat, ist die Bereitstellung von Festplattenkapazität für die Benutzer mit Hilfe des verteilten Dateisystems AFS, welches derzeit in der Version der Firma IBM verwendet wird. Da dieser Dienst die Plattenkapazitäten für alle Mitarbeiter des LRZ sowie 30.000 externe Benutzer zur Verfügung stellt, wird auch die Aufgabe der Betreuung dieses Dienstes durch eine eigene Arbeitsgruppe erfüllt.
4. **Datenbanken**
   Die beiden genannten Datenbankmanagementsysteme MySQL (der Firma MySQL AB) und Oracle (der Firma Oracle) werden ebenfalls von einer eigenständigen Arbeitsgruppe in Betrieb genommen, konfiguriert und auf dem laufenden Stand gehalten.
5. **Web**
   Die Außendarstellung des LRZ (wie auch vieler anderer Firmen) erfolgt heutzutage in einem immer größeren Maße auf der Webseite. Hierüber werden neben allgemeinen Informationen zum LRZ selber auch verschiedenste Informationsangebote an die "Kunden" des LRZ publiziert. Zusätzlich werden über die Infrastruktur noch Webdienste für die Mitglieder des MWN angeboten. Die Aufgabe der Web-Arbeitsgruppe ist es nun, diese Dienste aufrecht zu erhalten und an neue Entwicklungen anzupassen. Hierfür warten sie eine Reihe von Apache-Dämonen und auch einen Zope-Server.

2.1.3 Überwachung der Dienste

Die hier vorgestellten Dienste müssen rund um die Uhr funktionieren. Um sicherzustellen, dass ein Ausfall oder Unregelmäßigkeiten beim Betrieb bemerkt werden, erfolgt eine Überwachung der Dienste über drei verschiedene Wege - zwei Softwareprodukte und auch manuell. werden die Dienste auf verschiedene Weise überwacht. Die Überwachung der hier dargestellten Komponenten erfolgt über zwei verschiedene Produkte. Zusätzlich wird auch noch manuell vorgegangen.

a) Direkte Kontrolle der Layer4/7 Switches Die beiden Layer4/7-Switches überwachen sich - unter anderem um eine möglichst geringe Latenzzeit für die Übernahme bei Ausfällen zu erreichen - gegenseitig. Um längerfristige Ausfälle bemerken zu können, gibt es in regelmäßigen Abständen zusätzlich noch eine Aktivitätskontrolle durch das zentrale Monitoringsystem (b)). Bei der nächsten Generation von Fail-over Geräten, welche an dieser Position eingesetzt werden sollen, gibt es Überlegungen, die in diesen Geräten eingebauten Überwachungsmodule zur Überwachung fremder Dienste (wie z.B. der Webserver) einzusetzen.

b) Zentrale Überwachung durch OVO Die allgemeine Überwachung der Komponenten erfolgt mittels der Software Open View Operations (OVO) der Hewlett-Packard GmbH ([HP OVO]). Diese ist in der Lage, die Überwachung über zwei verschiedene Strategien zu bewerkstelligen.

Zum einen gibt es die Möglichkeit, bestimmte Merkmale eines Dienstes zentral von einer Managementmaschine aus zu überwachen (beispielweise die grundsätzliche Erreichbarkeit eines Rechners per "ping"). Auf diese Art und Weise wird z.B. die generelle Erreichbarkeit des NFS-Servers oder auch der Zope-Server überprüft.

Zum anderen kann ein Agent, welcher auf der zu überwachenden Maschine läuft, für eine feingranularere Überwachung eingesetzt werden.
So läuft auf den Webservern jeweils einer dieser Agenten, der die Verfügbarkeit der Cron-Dämonen und des jeweils aktiv geschalteten Web-Dämons überwacht. Im Falle eines Fehlers startet der Agent den Dämon selbständig wieder neu. Des weiteren werden von diesen Agenten auch andere Dienste, wie die Erreichbarkeit der benötigten Teile des AFS-Clusters und auch die Datenbankserver, überprüft.

Nachkontrolle durch Logfileanalyse Zu guter Letzt wird über eine Analyse der Logfiles eine Kontrolle der Auslastung der einzelnen Maschinen, sowie eventuell auftauchender Fehler, vorgenommen.

2.2 Problemstellung am LRZ

Derzeit gibt es im LRZ zwar ein funktionierendes Systemmanagement - jedoch *kein durchgehendes Dienstmanagementkonzept*.

2.2.1 Systemmanagement als vorwiegende Methode

Das am LRZ vorherrschende Systemmanagement zeichnet sich besonders dadurch aus, dass im Falle von Änderungen immer nur *einzelne voneinander unabhängige Pakete* betrachtet bzw. erstellt werden. Diese haben keinerlei Bezug zueinander und ähnliche Pakete müssen daher oft auf gleichem Wege voneinander unabhängig erzeugt, in Betrieb gebracht und gewartet werden. Dies führt gerade bei Änderungen, welche aufgrund von Sicherheitslücken vorgenommen werden müssen, zu Verzögerungen. Die *Datenhaltung und Datenverwaltung* erfolgt dabei *dezentral*, was dazu führt, dass besondere Auswirkungen, welche durch Veränderungen an den angebotenen Diensten zutage treten könnten, immer auf zusätzlichen Kanälen kommuniziert werden müssen. Auch erfolgt die Dokumentation in unterschiedlichen Formaten, Dateien und Ordnern, welche im Falle des Ausfalles einiger Mitglieder der jeweiligen Arbeitsgruppe durch die Vertreter erst wieder neu zusammengesucht und verinnerlicht werden müssen. Gerade bei Änderungen, welche andere Dienste beeinflussen könnten oder bei der

Bestimmung des Bedarfs an einem Dienst, ist es bisher nicht möglich, diese Informationen direkt einzusehen. Sie müssen jeweils erst von den betreffenden Spezialisten angefordert werden.

So sind auch die *Abhängigkeiten in anderen Dokumenten* untergebracht, als die Konfigurationsinformationen, oder - im Extremfall - mindestens für eine Weile nur in den Köpfen derjenigen Mitarbeiter abgelegt, welche den Dienst aktuell bearbeiten. Somit erfolgt die Abbildung von Abhängigkeiten unter den verschiedenen Diensten immer über die beteiligten Arbeitsgruppen und kann nicht automatisch eingesehen werden. Letzteres wäre jedoch bei der automatischen Erkennung und Behebung von Störungen vorteilhaft. So bieten aktuelle Dienst- und Netzwerkmanagementsysteme heutzutage die Möglichkeit an, Abhängigkeiten zwischen einzelnen betrachteten Diensten in die Konfiguration mit einzubeziehen. Die hierin verewigten Informationen sind jedoch im Regelfall nicht Teil der Dokumentation und müssen so parallel zur eigentlichen Dokumentation gepflegt werden.
Es bleibt dem einzelnen Mitarbeiter überlassen, ob er in seiner Dokumentation nur den aktuellen Stand vermerkt oder ob er eine Historie über alle Änderungen - in einem eigenen Format oder durch ein Revisionsverwaltungssystem unterstützt - aufbewahrt. So ist es oft nachträglich nicht mehr möglich, den Ablauf von Änderungen nachzuvollziehen, wenn sich diese dem Gedächtnis der verantwortlichen Person entzogen hat.

Im Falle von *Anpassungen* bestehender Pakete werden diese nicht einfach nur verändert, sondern es entstehen *neue Pakete*, die keinerlei Informationen über die Pakete besitzen, welche sie ersetzen. Dies führt auf der einen Seite zwar dazu, dass Ballast abgebaut werden kann, andererseits ist die Erzeugung eines neuen Paketes immer aufwändiger als die Anpassung eines existierenden, da alle Merkmale des neuen Paketes überprüft werden müssen, statt nur die Merkmale, welche direkt durch das zu ändernde Paket beeinflusst werden. Die Planung des Lebenszyklus eines Paketes erfolgt dabei nur in den Köpfen der beteiligten Mitarbeiter und wird nicht dokumentiert. Dadurch ist aber auch keine definierte Außerdienststellung von Paketen möglich.

Der Vorteil der bisherigen Struktur im LRZ mit dem Systemmanagement ist, dass bei der Einführung neuer Produkte oder Dienste diese nicht erst umständlich an die Struktur des bestehenden Managementsystems angebunden werden müssen. Der Nachteil ist allerdings, dass die Abhängigkeiten oft erst dann richtig eingeschätzt werden können, wenn das neue System verstanden und in einer ähnlichen Umgebung getestet worden ist. Wünschenswert wäre sicher eine Managementumgebung, die nicht nur eine einfachen Einbindung der neuen Komponente ermöglicht, sondern auch auf Basis bereits vorhandener Abhängigkeitsinformationen diese auf die neue Komponente anzuwenden hilft.

Es wird zwar in einzelnen Bereichen versucht, das Management auf Dienstmanagement umzustellen, dies erfolgt aber zumeist jeweils nur im Hinblick auf den betroffenen Bereich, nicht auf ein die gesamte Organisationsstruktur umfassendes Konzept. Dieses würde auch viele Bereiche überfordern, da sie dabei Informationen verwenden müssten, welche nicht ihren eigenen Arbeitsbereich betreffen.
Bis dato gibt es noch keine fertigen Tools, die das Dienstmanagement so umfassen, wie es in einer Organisation wie dem LRZ vonnöten wäre. Außerdem wird im LRZ noch ein anderer Ansatz verfolgt, indem Managementsysteme auf wissenschaftlicher Basis erforscht und entwickelt werden. Dabei besteht jedoch das Problem, dass Versuche nur dann durchgeführt werden können, wenn sie den regulären Betrieb nicht einschranken. Aufgrund der wissenschaftlichen Betätigung wäre es möglich, dass einzelne Komponenten eines Servicemanagementsystemes im Rahmen eines offenen Standards selber entwickelt und geschrieben werden. Dies allerdings nur dann, wenn das zentrale System definierte und standardisierte Schnittstellen zur Verfügung stellt, an die das neue Produkt als eigenständiges Modul andocken kann.

2.2.2 Darstellung an einem Beispiel beim Fehlermanagement

Um die Schwierigkeiten zu verdeutlichen, welche derzeit bei dem Fehlermanagement am LRZ auftreten, nun ein kleines Beispiel. Was passiert, wenn einer der Server für den AFS-Dienst ausgefallen ist und ein Mitarbeiter der für die Webserver zuständigen Arbeitgruppe bemerkt, dass er auf das AFS-System nicht mehr zugreifen kann?

Nachdem der Mitarbeiter die Störung festgestellt hat, geht er zu seinem Nachbarn und erkundigt sich dort, ob das Problem dort ebenfalls vorliegt. Dies wird ihm im Regelfall von seinem Kollegen bestätigt werden, da der zugehörige Server ja nicht in Betrieb ist. Als nächstes wird er bei der Hotline nachfragen, ob diese Störung dort bekannt ist und weitere Informationen verfügbar sind. Gehen wir nun davon aus, dass dies nicht der Fall ist. Er wird sich daraufhin beim zuständigen Betreuer der AFS Arbeitsgruppe erkundigen, ob dort das Problem bereits bekannt ist (was ja aufgrund der Verwendung verschiedenster Managementtools der Fall sein könnte) und ob das Problem aufgrund dessen eventuell schon behoben ist. Es hat sich gezeigt, dass bei Arbeitsgruppen, deren Dienste von einer starken Abhängigkeit zu einzelnen anderen Diensten geprägt sind, diese das Problem häufig melden, bevor es an anderer Stelle aufgefallen ist und das Problem demzufolge auch noch nicht behoben werden konnte. Wenn der Mitarbeiter der AFS-Gruppe die Störung dann nachvollziehen konnte, wird er diese beseitigen und den Personenkreis, der die betreffende Störung gemeldet hat, von der Behebung dieser unterrichten.

Im Gespräch mit Herrn Hahn im LRZ ([Hahn 05]) wurden noch einige weitere Punkte erörtert, welche für ein vereinfachtes Management des Webdienstes wünschenswert wären.

Verschiedene Überwachungsmodule Der Aufbau der dezeitigen Überwachung erfolgt durch verschiedene Monitoringsysteme, welche nicht miteinander kommunizieren. Die Einbindung dieser Systeme in ein gemeinsames Managementsystem würde es ermöglichen, die existierenden Produkte weiterzuverwenden und Spezialsoftware einzusetzen, die spezielle Aufgaben erfüllt.

Verschiedene Plattformen Das LRZ verwendet nicht nur Produkte einer einzelnen Firma. So werden sowohl Computer mit der Intel-, als auch mit der MIPS-Architektur der Firma Sun eingesetzt. Als Betriebssysteme werden Linux und Solaris genutzt.

Einfacher Aufbau der Dämonen mit kleineren Unterschieden Die vorgestellten Dämonen decken bereits die Mehrheit aller Anwendungsfälle ab. Es gibt jedoch Spezialfälle, in den denen ein neuer Dämon gebraucht würde, der nur marginal von einem bereits existierenden verschieden ist. Bisher muss jeder neue Dämon von Grund auf neu erstellt werden. Wünschenswert wäre eine Unterstützung bei der Erstellung von Dämonen, welche sich nur marginal unterscheiden.

Einfache Zuweisung der Verantwortlichkeit bei Problemen Bei Problemen ist die Frage der Verantwortung bisher erst dann bestimmbar, wenn das Problem bereits gelöst und die fehlerhafte Komponente bestimmt worden ist. Bisher müssen alle beteiligten Arbeitsgruppen zusammenarbeiten, um zu bestimmen, welche Komponenten nicht funktionieren.

Vereinfachte Verwaltung und Aufbau der Dämonenkonfiguration Nicht nur der Aufbau von ähnlichen Dämonen liese sich vereinfachen, sondern auch der Aufbau der allgemeinen Dämonen, da diese bei Sicherheitsupdates immer neu compiliert werden müssen. Ebenso verhält es sich mit der Verwaltung der zugehörigen Konfigurationen.

2.3 Managementanforderungen am LRZ

Durch die höhere Auslastung der Dienste gerät das bisher verwendete System Management an eine Grenze, an der die Wartung und Fortentwicklung des Angebotes nicht mehr möglich ist. In Folge dessen nehmen Routineaufgaben einen immer größeren Zeitrahmen in Anspruch und damit Zeit, die für neue Projekte notwendig wäre. Auch wird die Einführung neuer Projekte immer schwieriger, da die Auswirkungen auf die bestehende Umgebung nur durch Einzelbetrachtungen evaluiert werden können.

Wünschenswert wäre ein erhöhter Automatisierungsgrad bei der Verwaltung der alltäglichen Aufgaben. Dies wird zwar durch Eigenentwicklungen der einzelnen Sachbereiche bereits teilweise erreicht, jedoch sind diese Entwicklungen jeweils auf den gerade vorliegenden Fall spezialisiert und müssen für die nächste Komponente wieder überarbeitet oder gar vollkommen neu konzipiert werden. Es wäre erfahrungsgemäß oft einfacher, wenn es eine Möglichkeit gäbe, welche verschiedene Ausprägungen einer Konfiguration zusammenfasst und eine einfache Modifizierungsgrundlage ermöglicht.

Auch und gerade zwischen den verschiedenen Managementtools existiert keine definierte Schnittstelle, so dass man für den Transfer und Abgleich der Informationen auf das Wissen der jeweiligen Mitarbeiter angewiesen ist. Dies ist gerade im Fehlermanagement (wie an obigem Beispiel ja bereits kurz angeführt) eine zeitraubende und auch fehlerträchtige Angelegenheit. Gerade im Falle der Abwesenheit eines Mitarbeiters können Änderungen, welche dieser vorgenommen hat, vor allem deren Auswirkungen auf die verschiedenen Komponenten des Systems, nicht mehr nachvollzogen und erfasst werden. Dagegen helfen auch regelmäßige und zeitnahe Vermerke in einer Änderungsdatenbank nur bedingt, da sich das Wissen des Mitarbeiters nur in einem zeitlich und örtlich sehr begrenzten Umfang wiederfinden lässt. Gerade Abhängigkeiten von anderen Diensten oder anderer Dienste zu der Komponente, welche ihr Verhalten absichtlich (z.B. durch eine Konfigurationsänderung) oder unabsichtlich (z.B. durch einen Fehler) verändert hat, sind nur sehr schwer in den bisher verwendeten Tools zu modellieren. Zumeist erfolgt dies - unter Zuhilfenahme einfacher Textdateien - immer noch im Gedächtnis des für die Komponente zuständigen Mitarbeiters.

Des weiteren soll die Lokalisierung von Fehlern über Organisationsgrenzen hinweg vereinfacht werden. Hier ist bisher noch eine manuelle Abfrage verschiedener Verantwortlicher notwendig, welche ihre Arbeit unterbrechen müssen, um nachzusehen, ob der Fehler in ihrem Aufgabenbereich aufgetreten ist, oder etwa in einem anderen Bereich, von dem sie wiederum abhängig sind. Umgekehrt wäre es wünschenswert, gezielt Warnungen an einzelne Organisationen weiterzugeben, wenn Wartungen oder Änderungen notwendig sind.

Zu guter Letzt ist auch eine gezielte Überwachung verschiedener Parameter, aber auch der aufgetretenen Problemfälle für die Entscheidung über benötigte Kapazitäten eine Option, welche für den Betrieb ihre Bedeutung hat.

2.4 Anforderungen an das Informationsmodell

Die Managementanforderungen am LRZ, die im Gespräch mit der Web-Gruppe erörtert wurden, sind spezialisiert auf die Situation vor Ort angepasst. Um eine Lösung zu finden, die nicht nur in dem spezifischen Fall und innerhalb dieser Organisation funktioniert, werden nun im Folgenden eine Reihe von Anforderungen beschrieben, die allgemein von Interesse sind. Diese Liste hat keinen Anspruch auf

Vollständigkeit, lehnt sich aber an [Kell 98] und [Lang 01] an, wo weitere Anforderungen ausführlich behandelt werden.

Herstellerunabhängigkeit Da üblicherweise Produkte mehrerer Firmen eingesetzt werden und manche Dienste auch nicht mehr von der eigenen Organisation besessen werden, ist es notwendig, dass die eingesetzten Managementprodukte es erlauben, mit Produkten unterschiedlichster Herkunft zu kommunizieren. Es ist nicht mehr akzeptabel, wenn nur die Produkte eines bestimmten Herstellers unterstützt werden.

Offenheit Um Herstellerunabhängigkeit gewährleisten zu können, ist es notwendig, dass die Spezifikationen für die Managementansätze für alle einsehbar sind. Idealerweise ist dies bereits im Entwicklungsstadium der Fall, so dass verschiedene Organisationen, welche sich mit den gleichen Diensten beschäftigen, zusammen zu einer gemeinsamen Spezifikation kommen können, ohne dass die Lösung nur auf die Produktlinie eines Herstellers fixiert ist.

Reifegrad Eines der wichtigsten Entscheidungskriterien für jede Managementstruktur ist ihr Reifegrad. Eine Managementstruktur, bei der die Standardisierung schon weit fortgeschritten oder besser gar bereits abgeschlossen ist, bietet Sicherheit bei der Implementierung in neue Systeme. Aber auch der Abdeckungsgrad der verschiedenen Diensteigenschaften ist ein nicht zu unterschätzender Faktor, da eigenständige Erweiterungen auch zu Inkompatibilitäten mit den Produkten anderer Hersteller führen können.

Verbreitung Ein Indikator für den Reifegrad ist der Verbreitungsgrad der Managementstruktur. Je höher die Verbreitung der Managementstruktur, desto wahrscheinlicher ist die "Zukunftssicherheit" bei der Investition. Im Bezug auf die MIB gibt dieser Indikator an, inwieweit die Verwendung der Managementstruktur bei der Entwicklung der Attribute für das Dienstmanagement in Betracht gezogen werden sollte.

Einfachheit für Anwender Die Dienste, welche mit Hilfe der MIB betreut und konfiguriert werden, sollen auch von normalen Administratoren ohne Spezialkenntnisse überwacht werden. Dafür ist es bereits früh notwendig, das System so zu gestalten, dass es für den normalen Anwender in seinen Grundzügen schnell verstanden werden kann. Dies sollte sowohl bei dem verwendeten Managementsystem der Fall sein, wie auch bei der zugrundeliegenden Struktur der Informationsgewinnung. Gerade gut durchdachte Lösungen zeichnen sich meist durch eine verblüffend einfach anmutende Struktur aus.

Flexibilität bzgl. Erweiterungen Um einem System Zukunftssicherheit mitzugeben ist es heute nicht mehr ausreichend, eine solide Basis zu schaffen, auf der alles aufbaut, sondern es ist notwendig, eine geordnete Evolution des Systems zuzulassen. Nicht nur für die Vorbereitung auf zukünftige Entwicklungen ist es notwendig, dass flexibel auf Erweiterungen eingegangen werden kann, sondern auch immer dann, wenn andere (alte) Systeme in das System mit einbezogen werden sollen. Somit kann man auch Systeme in die eigene Verwaltung mit aufnehmen, über die man keine Verfügungsgewalt hat.

Objektorientiertheit Nach Möglichkeit sollte die MIB einfach zu modellieren und implementieren sein. Dafür bietet sich die Objektorientierung wegen ihrer Kerneigenschaften der Datenabstraktion, Kapselung, Polymorphie und Vererbung besonders an. Gerade bei der Entwicklung der grundlegenden Klassen für eine MIB - um die es in dieser Arbeit geht - ist dies besonders hilfreich.

Allgemeiner Dienstbegriff Dienstmanagement kann nicht mehr nur auf einen einzelnen Dienst oder eine kleine Gruppe von Diensten hin betrachtet werden. Heutzutage erfolgt die Bereitstellung von Diensten oft erst durch die Zusammenarbeit verschiedenster Dienste. Unter diesen können sehr niedriglevelige Dienste, wie grundlegende Paketvermittlungstätigkeiten, aber auch relativ hochlevelige (Datenbankserver, Web, Mail) sein. Die Architektur muss in der Lage sein, beliebige Dienste beschreiben zu können, ob es sich nun um Verbindungs-, Darstellungs- oder Speicherdienste handelt.

Abstraktion Aufgrund der großen Anzahl von Diensten ist es heutzutage gar nicht mehr möglich, sich jeweils auf die einzelnen Dienste zu konzentrieren, die zusammengenommen dann einen weiteren Dienst darstellen; ebenso wie heutzutage der normale Autofahrer kaum darüber nachdenkt, in welcher Reihenfolge welches Teil in einem Automotor wie bewegt werden muss, damit das Auto funktioniert. Diese Aufgabe wird im Fehlerfall dem Techniker, im Regelfall aber einem automatisierten System, überlassen. Ein Managementsystem soll zum Beispiel ganze Parametersets verändern, wenn ein bestimmter Paramter verändert wurde, oder eine Anzahl von Diensten als einen einzigen maskieren.

Standardisierte Schnittstellen Ebenso wichtig ist es, zumindest an den Berührungspunkten mit anderen Systemen, einheitliche Schnittstellen zu entwickeln. Besser noch ist eine Beschränkung auf wenige verschiedene Schnittstellen innerhalb des Systemes. Dies betrifft zum einen die Anzahl der Stellen, an welchen Informationen abgeholt werden, zum anderen aber auch das Aussehen dieser Übergabepunkte. So ist es für eine Integration in andere Systeme hinderlich, wenn sich diese die benötigten Informationen aus verschiedensten Quellen (Mailinglisten, Usenetpostings, Webseiten, SNMP-Agenten, spezielle Auswertungsprogramme, etc.) besorgen müssen. Zum anderen ist es für den Kunden unwichtig, woher der Anbieter eines Dienstes die Metainformationen bekommt. Für ihn ist es hingegen besonders wichtig, dass diese Informationen den Zustand und die Qualität des Dienstes hinreichend beschreiben. Des weiteren ist es wichtig, dass die gewünschten Informationen an den Schnittstellen einheitlich und verständlich vorliegen.

Abhängigkeiten Wenn man das Bild mit dem einfachen Aufbau eines Dienstes mit seinen Abhängigkeiten (Abbildung 1.1) noch einmal betrachtet, so ist auffällig, dass bereits bei der relativ simplen Auslieferung einer einfachen, mit Hilfe einer Datenbank generierten Webseite verschiedenste Dienste zusammenarbeiten müssen. Dabei ist die Erfüllung der Aufgabe einer Komponente von der Erfüllung ihrer Aufgaben durch eine andere Komponente abhängig. Und genau die Darstellungsmöglichkeit dieser Abhängigkeiten fehlt bisher in vielen Managementsystemen.

Lebenszyklus Im Allgemeinen wird beim Dienstmanagement nur die Phase betrachtet, in der dieser Dienst in Betrieb ist (bzw. sein sollte). Der Grund hierfür liegt zum einen darin, dass die Inbetriebnahme von Diensten, Veränderungen an diesen, sowie die Außerbetriebnahme bisher nicht als Aufgabe des Dienstmanagements angesehen werden, da sie relativ selten vorkommen. Somit werden die daraus resultierenden Aufgaben und Betrachtungen bisher immer unabhängig von Menschen geplant und durchgeführt. Mit der immer kurzlebigeren Laufzeit von Verträgen zu Diensten wird das Verhältnis von Bereit- und Außerdienststellung eines Dienstes zur Laufzeit des Dienstes jedoch immer höher. Daher muss diese Aufgabe immer mehr von automatisierten Systemen übernommen werden und somit in die Dienstmanagementarchitekturen eingearbeitet werden.

Vorhandene MIBs / Objektkataloge Eine der besonderen Schwierigkeiten bei der Entwicklung einer Dienstmanagementarchitektur liegt in der Erstellung eines Objektkataloges. Dieser sollte immer so allgemeingültig sein, dass alle zu verwaltenden Dienste darin modelliert werden können. Er sollte aber nicht so allgemein sein, dass bei der Implementierung des Dienstmanagementsystems jeder zu einem vollkommen eigenständigen, zu anderen Implementierungen inkompatiblen Lösung, kommt. Somit ist es bei der Entwicklung dieser Arbeit von Interesse, in wie weit bereits Objektkataloge durch andere Arbeiten zur Verfügung gestellt wurden und in wie weit sie sich mit dem Dienstmanagement vereinbaren lassen.

Abbildung auf Programmiersprachen Eine MIB ist um so wertvoller, je einfacher sie und die auf ihr basierenden Objektkataloge sich in einer gängigen Programmiersprache abbilden lassen. Dies ist auch Voraussetzung für die Verwendbarkeit über Organisationsgrenzen hinweg. Da hierdurch auch die Verwendung von Konvertierungstools, die bei der Übersetzung oft Fehler machen, nicht mehr notwenidig ist, ist eine Erhöhung der Verständlichkeit der vermittelten Daten zu erwarten.

Dienstattribute Viele Objektkataloge bieten zwar grundsätzliche Klassen an, die dann beliebig erweitert werden können, aber keine Attribute, die die Dienste allgemein beschreiben. Dies führt zu unterschiedlichen Schnittstellen, da die Attributsnamen ineinander überführt werden müssen. Ein einfaches Beispiel hierfür sind die Benennungen von Attributen in verschiedenen Sprachen. So heißt ein Objekt im Deutschen “Objekt”, während die englische Schreibweise “object” wäre. Eine Beschränkung auf die englische Schreibweise würde dieses Problem zwar lösen, jedoch gibt es auch den entgegengesetzten Fall, bei dem gleichnamige Attribute unterschiedliche Bedeutungen haben. So ist der “owner” einer Komponente vielleicht in dem einen Fall der Verantwortliche für diese, in einem anderen Fall handelt es sich um den aktuellen “Besitzer” der Komponente, also denjenigen, unter dessen physischer Kontrolle diese steht.

Methoden Ebenso wie bei den Attributen ist es auch bei Methoden sinnvoll, eine Auswahl an Methoden zu haben, die allgemeingültig verwendet werden können und deren Wirkung definiert ist. Die Begründung ist hier ähnlich wie bei den Attributen.

Historie Um das Management der Dienste überprüfen zu können, ist es notwendig, Änderungen nachvollziehen zu können, wofür ein sogenannter “audit trail” notwendig ist. Aber auch bei der Behebung von aktuellen und der Entdeckung von zukünftigen Fehlern kann ein Blick in die Historie viele notwendige Daten liefern. Aus diesen Gründen betrachtet jedes gute Dienstmanagementsystem nicht nur die gerade aktuellen Daten, sondern ermöglicht immer auch den Blick auf die Historie.

Kapitel 3

Verwandte Arbeiten

Die Idee, den Dienst in den Fokus der Betrachtung zu stellen, ist bereits seit mehreren Jahren Gegenstand verschiedener Arbeiten. Die meisten Arbeiten versuchen dabei nicht, alle Bereiche des Dienstmanagements abzudecken, sondern beschränken sich auf ausgewählte Teilprobleme.
Im folgenden Kapitel werden nun Arbeiten vorgestellt, die sich mit dem Dienstmanagement im allgemeinen befassen. Die Blickwinkel sind dabei unterschiedlich gewählt:
Das **MNM-Dienstmodell** ist ein akademischer Ansatz, die Dienste und ihre Beziehungen zu anderen Diensten sowie den Akteuren zu beschreiben.
ITIL kommt aus der Ebene des sogenannten "Upper Management" und versucht eine Verbindung der betriebswirtschaftlichen mit der technischen Sicht der Unternehmensführung herzustellen.
OSI wurde dagegen von Technikern entwickelt und versucht einen allgemeinen Ansatz für die technische Sicht zu schaffen. Anhand des hier vorgestellten Kommunikationsmodelles kann man die verschiedenen Granularitäten oder auch Stufen bei der Verwaltung von Komponenten erkennen. Somit ist OSI der einzige Ansatz, der darstellt, dass alle Abstraktionsebenen ihre Bedeutung haben und innerhalb des selben Ansatzes betrachtet werden müssen. Dabei betrachtet OSI jedoch immer nur die allgemeine Sichtweise über einen Dienst, nicht jedoch dessen Attribute.

CIM auf der anderen Seite bemüht sich, alle möglichen Arten von Komponenten und deren Attribute darzustellen. Dafür wird ein sehr pragmatischer Ansatz gewählt, der verfügbare Attribute mit Hilfe des objektorientierten Paradigmas gruppiert. Entwickelt wird CIM dabei entsprechend den Anforderungen der Mitgliedsfirmen des DMTF-Konsortiums.
SID wurde entwickelt, da auf den Sektor der Telekommunikationsfirmen die bei CIM entwicklenten Klassen nicht mehr 1:1 abbildbar sind. SID verwendet die Sammlung aus CIM als Ausgangsbasis und verallgemeinert die Klassen. Um dabei eine höhere Wiederverwertbarkeit zu erreichen, wurde auf Design Pattern zurückgegriffen.

Aus der täglichen Arbeit ist die **Internet Managementarchitektur** entstanden. Mit dem Beginn des Netzwerkmanagements wurden Verwaltungsmechanismen notwendig, die viele verschiedene Komponenten über das Netzwerk verwaltbar machten. OSI war noch in der Entwicklung und vielen Firmen zu aufwändig. Also wurde ein Ansatz entwickelt, der einen gezielten Zugriff auf die verschiedensten Einstellungen und Werte von Komponenten erlaubte. Das Kommunikationsmodell des Internet Management stellt jedoch auch seine größte Schwachstelle dar und zeigt, dass auch die Sicherheit bei der Implementierung eines Systems von vornherein bedacht werden muss. Obwohl ursprünglich nur als Interimslösung gedacht, ist das Internet Management bis heute die vorherrschende Methode bei der

Verwaltung von vernetzten Komponenten.
Ähnlich erging es den Mitarbeitern von Apple, als sie die Vernetzung von Macintosh-Systemen von AppleTalk auf TCP/IP umstellten. Die Aufgabe bestand hier darin, die in AppleTalk gewohnte Möglichkeit der direkten Ansprache von Diensten aufgrund der Beschreibung der Komponenten unter TCP/IP abzubilden. Das Ergebnis ist als **DNS-Based Service Discovery (DNS-SD)** bekannt gemacht worden. Im Gegensatz zu den vorherigen Ansätzen beschreibt DNS/SD jedoch nicht das Dienstmanagement, sondern nur ein Verfahren der Suche nach Diensten mit bestimmten Eigenschaften, also nur einen sehr begrenzten Teilaspekt.

3.1 Das MNM Service Modell

Das Munich Network Management Team (MNM-Team), eine interuniversitäre Arbeitsgruppe der Institute für Informatik an der Ludwig-Maximilians-Universität München und der Technischen Universität München und des Leibniz Rechenzentrum München, haben in [GHHK 01a] einen ersten Schritt zu einem generischen Dienstmodell geschaffen.

Bei der Entwicklung wurde besonderer Wert darauf gelegt, dass das resultierende Modell eine generische, abstrakte Definition eines Dienstes anbietet, dabei aber auch den Aspekt der an diesem Dienst beteiligten Organisationen nicht vernachlässigt. Um eine klare Darstellung zu ermöglichen, wurden der Dienst und seine Implementierung getrennt betrachtet und generische Bausteine erschaffen. Der ganze Ansatz wurde dabei aus der Management-Sicht mit der Überlegung erstellt, dass das Management des Dienstes bereits einen integralen Bestandteil desselben darstellt.

Als Charakteristik eines jeden Dienstes wurde dabei herausgearbeitet, dass es immer zwei verschiedene Rollen gibt, die einen Dienst betrachten. Auf der einen Seite den Anbieter (provider), auf der anderen Seite den Abnehmer (customer) eines Dienstes. Die Zusammenarbeit dieser beiden ist nach dem MNM-Dienstmodell die Grundlage für das Zustandekommen eines Dienstes und somit auch die Grundlage für die Entwicklung eines Dienstmodelles. Zusätzlich wurde noch ein Teil des Dienstes identifiziert, welcher unabhängig von den beiden Rollen (side independent) ist und in der abstrakten Sichtweise die Schnittstelle zwischen den beiden Rollen darstellt.
Die Interaktion zwischen den beiden Parteien wird dabei auf Basis des Lebenszyklus des Dienstes modelliert, so dass jeder Dienst die Phasen des Designs, der Verhandlung, der Beschaffung, des Gebrauchs (mit Änderungen) und der Deinstallierung durchmacht. Da die Interaktionen jedoch abhängig von der Betrachtung auf veschiedenen Wegen stattfinden, bietet das MNM-Dienstmodell zwei verschiedene Sichten an:

- eine mit Betrachtung der abstrakten Definition des Dienstes, wie in Abbildung 3.1 dargestellt. (Service View)
- eine mit Betrachtung der Implementierung des Dienstes, wie in Abbildung 3.2 dargestellt. (Implementation View)

Beim Service View steht die Funktionalität des Dienstes im Vordergrund, welche durch die Interaktion von provider und customer geprägt ist. Die Details der Implementierung werden dabei ausgeblendet.

Der Implementation View beschränkt sich dagagen auf die Seite des Dienstanbieters und hat seinen Fokus auf der Identifikation der beteiligten Objekte und deren Assoziationen zueinander. Dienste bestehen dabei aus “services”, die direkt angeboten werden, sowie sogenannten “subservices”, die durch “sub-service clients” eingebunden werden.

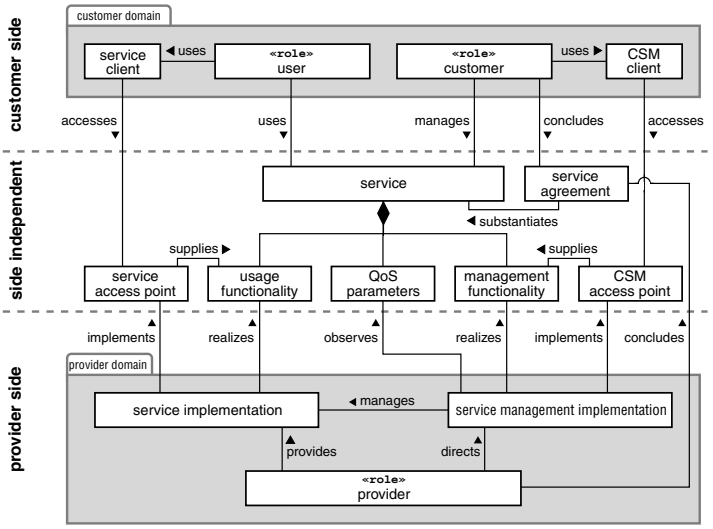

Abbildung 3.1: MNM-Servicemodell - Service View

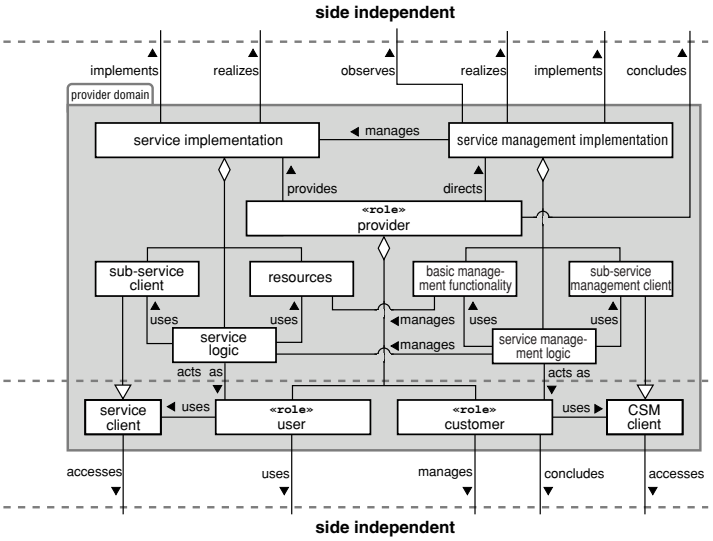

Abbildung 3.2: MNM-Servicemodell - Implementation View

Durch die als “sub-service” bezeichneten Dienste wird dabei der rekursive Aufbau von Diensten aus Diensten ermöglicht. Der Dienstanbieter ist dann der “provider” des “sub-service”, während der Dienstabnehmer der “provider” des “service” ist.

Dieser Ansatz erfüllt die Forderung, ein generisches Framework für das Dienstmanagement zu schaffen. Durch die Abstraktion des Dienstes von seiner Implementierung schafft er auch Unabhängigkeit von der Betrachtungsweise einzelner Hersteller und bietet eine einheitliche Schnittstelle an, die dabei offen genug ist, auch Erweiterungen zuzulassen. Durch seinen universitären Ursprung ist er auch allen offen zugänglich.
Er leidet jedoch auch an den Problemen, die allen generischen Ansätzen zu eigen sind: Es gibt keine Abbildung des Modelles auf einen real existierenden Objektkatalog, so dass jede Implementierung

einen eigenen - mit hoher Wahrscheinlichkeit zu anderen inkompatiblen - Objektkatalog entwickeln muss. Auch die Abhängigkeit zwischen Management und Implementierung desselben ist nicht direkt zu verarbeiten.

3.2 ITIL

Entstanden in den späten 1980'er Jahren ist **ITIL** eine Entwicklung der **CCTA** (Vorgänger des **OGC** (Office of Government Commerce)), um die Vorgehensweisen innerhalb der britischen Regierung zu standardisieren. Zu diesem Zweck wurden bei verschiedenen Organisationen und Firmen Informationen über deren Vorgehensweise beim Service Management gesammelt. Diese wurden daraufhin entsprechend den Anforderungen an die IT der CCTA und der Regierung des United Kingdom gefiltert und analysiert. Aus den gewonnenen Daten wurde dann eine Sammlung von "best practices" erarbeitet, welche sich später auch für viele andere Organisationen als brauchbar erwiesen.

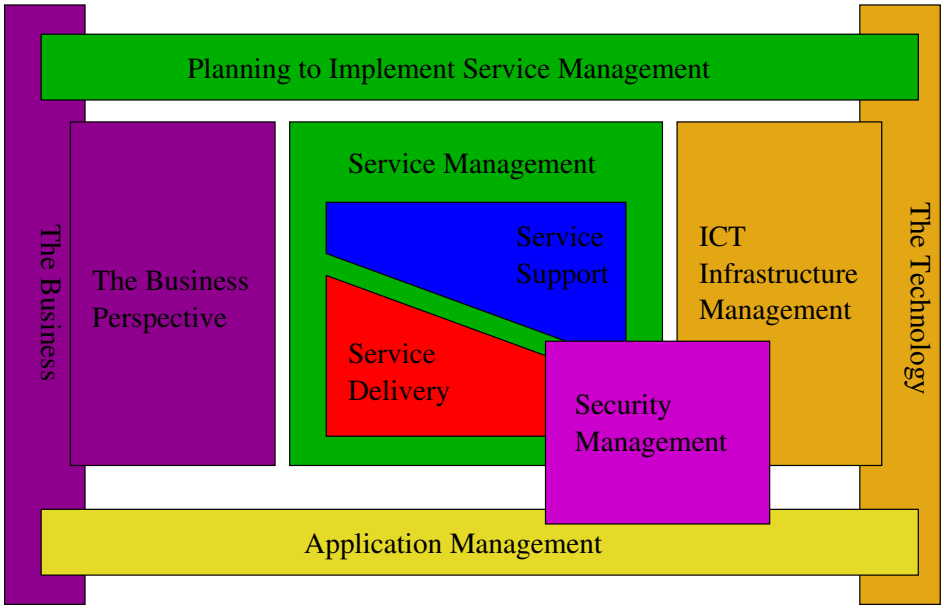

Abbildung 3.3: ITIL Publication Framework

Nach eigener Aussage ist ITIL seit Mitte der 1990'er Jahre der de facto Standard für das Service Management. In Deutschland hat er sich aber erst in den letzten Jahren immer mehr durchgesetzt. Durch das Fehlen fester Vorgaben für die Umsetzung im laufenden Betrieb soll eine einfache Anpassung bestehender Strukturen möglich sein. Dies ist aus verwaltungstechnischer Sicht auch richtig. Auch gibt es Abbildungen von ITIL auf CobiT, welches sich als Standard für den Auditing Bereich zu etablieren versucht.

Allerdings ist gerade bei der technischen Umsetzung der best practices in laufende Anwendungen zur Service Verwaltung besonders diese Unbestimmtheit ein großes Manko. Jede Implementierung verwendet ein eigenes Design, wodurch eine Interoperabilität zwischen verschiedenen Implementierungen nicht mehr gegeben ist. Das zentrale Element zur Konfigurationsverwaltung ist die sogenannte Configuration Management Database (**CMDB**). Diese ist als zentrale Datenbank angelegt, welche In-

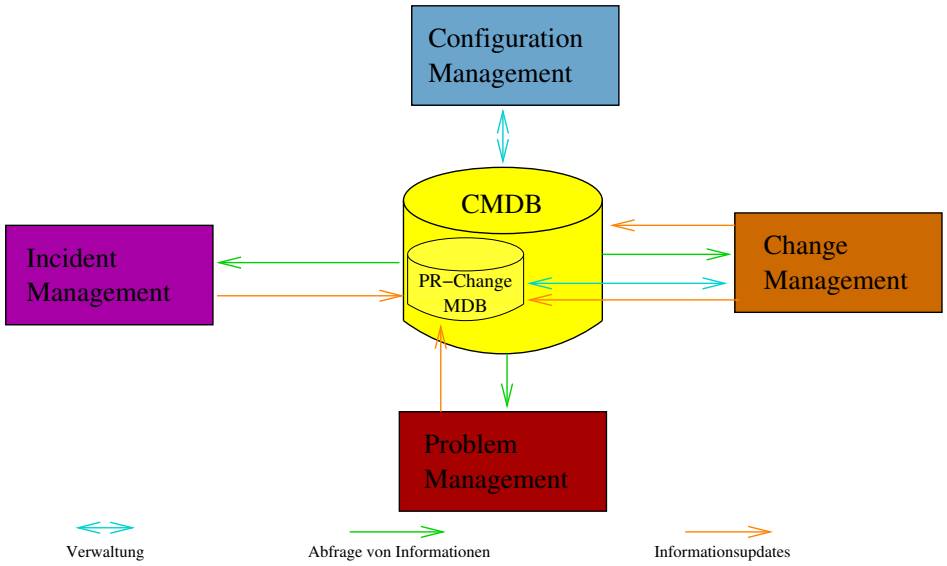

Abbildung 3.4: Zentrale Datenhaltung bei ITIL

- an 80% fit to operational requirements
- a meeting of *ALL* mandatory requirements
- little (if any) product customisation
- ITIL compliance
- a sound data structure and handling
- business-driven not technology-driven
- administration and maintenance costs within budget

Tabelle 3.1: Anforderungen von ITIL an Serviceverwaltungstools ([ITIL 00], S. 246)

formationen über alle zu verwaltenden Objekte enthält, die ihr jeweils von den vorhandenen Systemen bei Änderungen oder Indienststellungen zugetragen werden. Jedoch ist diese nur relativ abstrakt (eben als “best practice”) und nicht im Hinblick auf eine Implementierung mit Software beschrieben.
Kapitel 10 (Service Management Softwaretools) von [ITIL 00] beschäftigt sich ausdrücklich mit diesem Thema. Begonnen wird das Kapitel mit der Frage: “The first question you should ask yourself on this topic is ’Do I really need software tools?’ “ um dann noch weiter festzustellen: “However, good people, good process descriptions, and good procedures and working instructions are the basis for successfull Service Management”. Hier wird deutlich, dass ein Softwaresystem bestimmte Grundanforderungen an die Umgebung nicht ersetzen kann, aber auch, dass ITIL ein Softwaresystem nicht als zentrales Element, sondern nur als Unterstützung der Verwaltung der Elemente ansieht.

Das Hauptaugenmerk liegt folglich definitiv mehr auf den angewandten Bussines-Prozessen als auf irgendwelcher Software. So ist die Anforderungsliste an ein softwaregestütztes System auch relativ kurz und grundsätzlicher Natur (siehe Tabelle 3.1). Interessanterweise ist eine dieser Anforderungen die “ITIL compliance”.

ITIL teilt die Aufgabenstellung in mehrere Bereiche auf, die in verschiedenen Büchern ([ITIL 99], [ITIL 00], [ITIL 01], [ITIL 02a], [ITIL 02b], [ITIL 02c] und [ITIL 03]) dargestellt sind. Einen Über-

blick über die Bereiche bietet Abbildung 3.3 aus [ITIL 02a]. Das Hauptaugenmerk ist hierbei auf die Prozesse und die einzelnen Schritte zur Erledigung einer Aufgabe gelegt worden. Diese wurden jeweils aus sogenannten “best practices” gewonnen, wobei sich leider - wie bereits in [Jäge 05] beschrieben -, immer wieder Inkonsistenzen eingeschlichen haben.

Modelle zum Management der Dienste werden durch ITIL nicht vorgegeben. Die Aufteilung in die verschiedenen Managementbereiche kommt einem Modell noch am nächsten. Dies macht die Verwendung als Grundlage für eine MIB nur bedingt empfehlenswert, da aus den Prozessen erst die Kenntnis gewonnen werden muss, welche Informationen zur Erfüllung der Prozesse benötigt werden. Erst mit diesen Informationen kann man bestimmen, welche Daten verwaltet werden müssen.

ITIL stellt jedoch einige Anforderungen an sich und an Produkte, die in Einklang mit den Ideen von ITIL stehen.

Funktionale Abhängigkeiten Während bei vielen anderen Ansätzen die Abhängigkeiten zwischen verschiedenen Diensten ausgeklammert werden, stellt die Modellierung dieser Abhängigkeiten für ITIL eine der zentralen Informationsquellen dar. Sei es bei der Verwaltung der Infrastruktur, im Fehlermanagement, bei Änderungen oder beim Ressourcenmanagement.

Verständlichkeit Eine durchgängige, leicht verständliche Strukturierung der Datenmodellierung erleichtert die Implementierung und Erweiterung der Daten. Bei der Pflege der Daten kann nur durch eine intuitiv erfassbare Datenstruktur sichergestellt werden, dass die Informationen regelmäßig aktualisiert werden und korrekt sind.

Standardisierte Schnittstellen Auch ITIL sieht, dass es zumeist nicht möglich ist, alle Dienste mit Hilfe eines einzigen Managementsystems zu verwalten, sondern dass Informationen von anderen Systemen mit eingebunden werden müssen. Dies ist aber nur dann möglich, wenn standardisierte Schnittstellen vorhanden sind.

Offene Standards Auch die IT Infrastructure Library fordert, offene internationale Standards zu verwenden. Damit meint sie zum einen sicher sich selbst, denkt aber auch an die Verbreitung der zu verwendenden Modelle. Nur offene und weit verbreitete Standards gewährleisten die Aussicht auf eine lange Lebensdauer.

Zentrale Datenbank Des weiteren fordert ITIL die Verwendung einer zentralen Datenbank, in welcher alle Daten vorgehalten werden. Diese Anforderung stammt aus einer Zeit, in der verteilte Datenbanken bedeuteten, dass die Informationen in einzelnen Zellen zusammengelagert waren. Diese Zellen waren jeweils informationstechnisch voneinander abgeschottet und Daten der einen Zelle konnten nicht von der anderen aus eingesehen werden. Heutzutage hat sich das Paradigma gewandelt und die Vorteile einer dezentralen Datenbank mit *zentralem Zugriff* heben die Nachteile auf, während sie weiterhin dem Benutzer das Gefühl geben, alle Informationen an einem Ort zu haben. Gerade bei weit verteilten Systemen ist hierdurch die abgerufene Information aktueller, während gleichwohl im Falle von Problemen auf dem Übertragungsweg die Informationen an dem Ort verfügbar sind, für den sie bestimmt sind, ohne sie doppelt halten zu müssen.

3.3 Open Systems Interconnection

Einer der ausführlichsten Ansätze - wenn nicht der ausführlichste Ansatz überhaupt - stammt von der ISO und nennt sich Open Systems Interconnection (OSI). Zumeist nur eingeschränkt auf das auch als OSI-Referenzmodell bezeichnete Schichtenmodell, umfasst es tatsächlich mehrere eigenständige Modelle, welche zusammengenommen erst eine vollständige Abbildung des Managements ermöglichen. Aufgrund seines relativ hohen Alters (es wurde in den 1970'er und 1980'er Jahren entwickelt) spricht es nicht ausdrücklich von Service Management, ist aber doch allgemein genug gehalten, um auch dessen Anforderungen bereits vorweg zu nehmen.

Es wird in ein ein Informations-, Organisations-, Kommunikations- und Funktionsmodell aufgeteilt.

3.3.1 Informationsmodell

Das von der ISO vorgeschlagene Informationsmodell ist durchgehend objektorientiert. Durch die Verwendung abstrakter Oberklassen und der Vererbung wird eine hohe Erweiterbarkeit der Daten sichergestellt. Aufgrund der Verschattung der tatsächlichen Datenquelle erfolgt eine Abstraktion nach außen, so dass verschiedene Datenquellen dennoch zu einer einzigen zu verwaltenden Sicht führen. Eine Verwaltung von bereits bekannten ähnlichen Systemen wird besonders dann unterstützt, wenn das Managementsystem das Prinzip der Allomorphie anzuwenden in der Lage ist, also sowohl die Informationen der Klasse als auch die der Instanz der Klasse verwenden kann. Hierdurch wird es ermöglicht, eine dem menschlichen Abstraktionsvermögen inhärente Anpassung auch mit Computern zu realisieren. Ähnliche Komponenten werden erkannt und analog zu bekannten verwaltet, ohne dass zunächst auf Sonderfunktionen der neuen Komponente eingegangen wird. So ist es zum Beispiel jedem Menschen, der das Schlüssel-Schloß-Prinzip erläutert bekommen hat, möglich, beliebige Schlösser mit beliebigen Schlüsseln zu erkennen und zu verwenden.

Modelliert werden die gemanagten Objekte (MOs) "template-basiert" in einer eigenen Meta-Sprache, welche unter [ISO 10165-4] spezifiziert ist. Laut dieser beinhaltet ein MO die folgenden Teile:

1. Attribute
2. Einen Satz an Aktionen
3. Einen Satz an Operationen
4. Einen Satz an Meldungen (notifications)
5. Verhaltensregeln darunter
    - Semantik der Attribute
    - Semantik von Operationen
    - Semantik der Meldungen
    - Beziehungen zu anderen MOs
    - Seiteneffekte
    - Informelle Beschreibung des Verhaltens in normaler Sprache
6. bedingte Pakete

Als kleine Besonderheit stehen die Typen der Attribute nicht von vornherein fest, sondern passen sich dem jeweils zu modellierenden Objekt an. Hierdurch wird eine größtmögliche Flexibilität gewährleistet.

3.3.2 Organisationsmodell

Innerhalb von OSI wird Management als verteilte Anwendung verstanden, bei der Komponenten eine Manager- oder eine Agenten-Rolle einnehmen. Die Verteilung der Rollen ist jedoch nicht starr vorgegeben, sondern jedes System kann jeweils beide Rollen einnehmen.

3.3.3 Kommunikationsmodell

Beim Kommunikationsmodell werden 3 verschiedene Ebenen unterschieden, die verschieden tief in die zu verwaltende Komponente eingreifen. Hier findet auch das bekannte OSI-Schichtenmodell seine Anwendung. Allen drei Ebenen ist gemeinsam, dass sie jeweils auf die gleiche MIB zugreifen können, dabei aber verschieden detaillierte Informationen verwenden.

Protocol Management Dieses stellt die feinste Granularität dar. Hier geht es um die Verwaltung der spezifischen Eigenschaften einzelner Protokolle, wie zum Beispiel “window sizes”, “timer” etc.

Layer Management Hier geht es um die Verwaltung der Eigenschaften innerhalb eines bestimmten Layers des OSI-Schichtenmodelles (siehe Abbildung 3.5). Dazu gehören z.B. Loop-Tests oder

Layer	Englische Bezeichnung	Deutsche Bezeichnung
7	Application Layer	Anwendungs– schicht
6	Presentation Layer	Darstellungs– schicht
5	Session Layer	Sitzungs– schicht
4	Transport Layer	Transport– schicht
3	Network Layer	Vermittlungs– Netzwerkschicht
2	Data Link Layer	Sicherungs– schicht
1	Physical Layer	Bitübertragungs– schicht

Abbildung 3.5: Schichten nach dem OSI-Referenzmodell

Routinginformationen.

Dieser Bereich wurde von der ISO zwar benannt, aber nicht weiter bearbeitet. Auch die IEEE hat einige Standards veröffentlicht, die hier einzuordnen sind.

Systems Management Dieser Bereich beschäftigt sich mit der Verwaltung ganzer Systeme. Der Austausch der Daten erfolgt über das **CMIP** via **CMIS**. CMIP war eigentlich als Ersatz für das weit verbreitete **SNMP** vorgesehen und liefert eine wesentlich höhere Sicherheit sowie eine bessere Benachrichtigung bei ungewöhnlichen Netzwerkereignissen. Auch kann es beispielsweise verwendet werden, um bestimmte Aufgaben auszuführen, was unter SNMP nicht möglich ist. Die

Verwaltung des MOs erfolgt über eine Einordnung dieser in eine Baumstruktur, wodurch eine Navigation mit Hilfe relativ einfacher Sprachen zur Graphennavigation ermöglicht wird.

Obwohl es in einigen Punkten SNMP übertrifft, hat sich CMIP gegen das althergebrachte, unsicherere und einfacherere SNMP nie durchsetzen können.

Das im Rahmen dieser Arbeit zu verwendende Service Management stellt hier somit lediglich eine logische Fortführung dar.

3.3.4 Funktionsmodell

Eine weitere Möglichkeit stellt die Aufgliederung in die verschiedenen Funktionsbereiche dar (siehe Abbildung 3.6, entnommen aus [HAN 99]) . Die Darstellung dieses Modelles lässt die rein technische

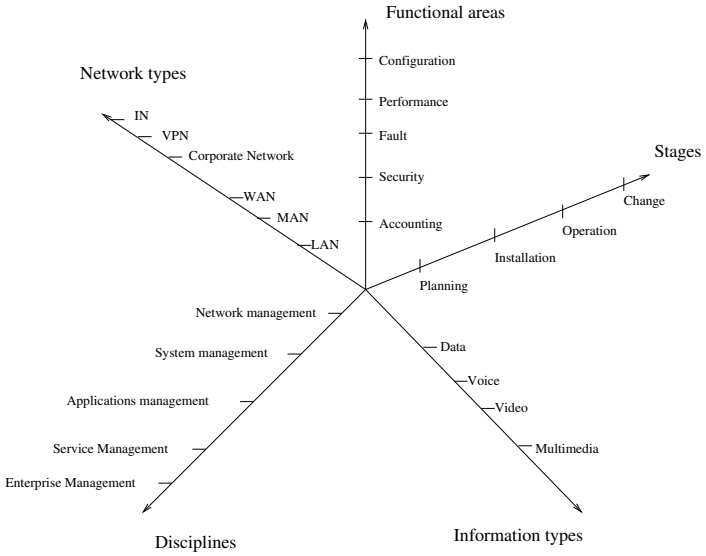

Abbildung 3.6: Dimensionen des Managements nach ISO

Spezifikation hinter sich und ermöglicht die Einordnung eines Managamentsansatzes innerhalb des Gesamtmanagements. Dabei werden verschiedene Aspekte des Managements aus der Sicht verschiedener Funktionsbereiche (zum Beispiel des Fehlermanagements) betrachtet. Durch die Betrachtung eines Aspektes aus den Blickwinkeln verschiedener Funktionen wird ein vollständigeres Bild der Einzelteile erzeugt.
Dieses Bild kann wiederum als Basis für ein generisches Modell der Funktionen, Dienste und assoziierten Objekte verwendet werden, welches dann in einer Bibliothek oder einem Submodell zum Tragen kommt.

3.4 Shared Information/Data Model (SID)

Das Shared Information/Data Modell (**SID**) [SID] ist eine Entwicklung des TeleManagement Forums [TMF HP] und Teil des Next Generation Operations Support Systems (NGOSS)-Frameworks[NGOSS]. Aufgabe von SID soll die Verbindung zu dem Bereich “System Analysis and

Design" sein. Aufbauend auf DEN-ng (welches wiederum auf DEN, OSS/J und MetaSolv basiert) wurde von der Arbeitsgruppe versucht, einen Standard zu schaffen, welcher die Verwaltung aller Assets im Bereich, der für die Telekommunikation notwendigen IT ermöglicht. Auch werden in diesem Ansatz die Grundkonzepte von CIM wieder verwendet bzw. beibehalten. Die Schwierigkeiten mit CIM waren jedoch nach Auffassung der Autoren zu groß, als dass dies als Grundmodell Verwendung hätte finden können. Der Ansatz von SID ist objektorientiert.
Die Methodik des TeleManagement Forums ist ein top-down Ansatz mit verschiedenen Abstufungen (Levels). Die beiden obersten Level sind bereits relativ weit entwickelt, jedoch ist der Bereich, welcher die Attribute zur Verwaltung enthält, noch immer in der "to be discussed" - Phase, so dass eine tatsächliche Anwendung des Modells in größerem Maßstab oder auch vorwärtskompatibel nicht möglich ist. Es gibt lediglich eine Reihe von Klassendiagrammen, welche jedoch keinerlei Methoden oder Attribute enthalten. Von weiten Bereichen des ServiceManagements wie dem Change Management sind nur einige Schlüsselkonzepte ausgearbeitet. Einer der treffendsten Sätze zu SID in [SID-4SO03] wird auf den Seiten 7 und 8 immer wieder wiederholt: "Additional work is needed in future phases of the SID."

Ein weiterer Punkt, welcher der dieser Arbeit zugrunde liegenden Aufgabenstellung widerspricht, ist die Unterteilung von Services in zwei verschiedene Sichten, welche auch unterschiedlich modelliert werden: Auf der einen Seite gibt es dem Benutzer zugewandte ("CustomerFacing") Services, auf der anderen Seite "ResourceFacing" Services, welche sich mit der technischen Seite beschäftigen sollen (Abbildung 3.7).

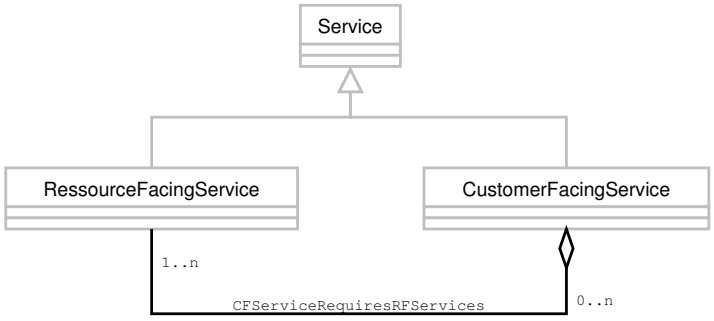

Abbildung 3.7: SID - Customer- und RessourceFacing Service

Einerseits erleichtert diese Trennung die Modellierung, da aufbauend auf den verschiedenen Sichtweisen nur diejenigen Informationen eingebracht werden müssen, welche von der jeweils angesprochenen Zielgruppe benötigt werden. Dies gilt um so mehr, als manche Dienste über verschiedene "Backends" zum Kunden gebracht werden können (z.B. ist es dem Kunden bei normalen HTML-Webseiten meist relativ gleichgültig, ob diese von einem Apache- oder einem IIS- Webserver verwaltet werden).
Andererseits sorgt gerade diese Trennung dafür, dass über weitere Konstrukte eine gegenseitige Abbildung der verschiedenen Sichten erzeugt werden muss. Hierdurch wird die sichere Gewinnung von Informationen, welche in dieser Sicht nicht als relevant erachtet werden, erschwert.

Eines der größeren Probleme von IUK-Systemen ist die Verwendung verschiedener "Sprachen" zur Verwaltung der Devices. Die Sprachen sind dabei nicht nur zwischen verschiedenen Herstellern unterschiedlich, sondern zum Teil auch zwischen verschiedenen major versions der Software der selben

Produktreihe. Um dieses Problem zu beheben und eine einheitliche Schnittstelle zu schaffen, setzt SID auf Design Pattern wie z.B. das Composite Pattern oder das Adapter Pattern.

3.5 Common Information Model (CIM)

CIM [CIM STD] wird von der Distributed Management Task Force des IETF entwickelt, welche aus der Desktop Management Task Force entstanden ist. Deren Aufgabe war es, ein Modell zur Beschreibung eines Rechnersystemes zu schaffen. Dementsprechend ist der Ansatz zur Modellierung hier bottom-up und sehr umfassend (von der Netzwerkdose über das Hostsystem bis zum Passwort der User). Der Anspruch von CIM ist, die Nachfolge von SNMP zu übernehmen. Folglich existiert hier eine Menge von Attributen innerhalb der Klassendiagramme. Die Entwicklung und vor allem die Namensgebung von Methoden für das Zusammenspiel der einzelnen Klassen bleibt jedoch jeweils demjenigen überlassen, der das Modell implementieren möchte, so dass CIM auch keine eindeutigen Interfaces für eine organisationsübergreifende Verwaltung liefern kann. Das Modell beschränkt sich in weiten Teilen auf das Device- und Netzwerkmanagement. Das Thema Service-Management wird nur von einer Arbeitsgruppe bearbeitet, und der Umfang umfasst nur diejenigen Attribute (mit den dazugehörigen Klassen), welche direkt aus den verschiedenen Devices abgefragt werden können.

Um den Aufwand für die Wartung und Entwicklung von CIM in einem verwaltbaren Rahmen zu halten, besteht CIM aus einem “core model” welches die grundlegendsten Klassen umfassen soll, und weiteren “workgroup models”, welche jeweils über die Klassen des “core model” miteinander kommunizieren. Eines der Hauptprobleme von CIM ist, dass die Entwicklung bis dato noch nicht abgeschlossen ist und die Änderungen zwischen den einzelnen Releases oft schwerwiegend sind. Zwar sind Änderungen am “core model” selten und Attribute dürfen nicht verändert, sondern nur hinzugefügt oder als “deprecated” markiert und später entfernt werden. Diese Änderungen sind an den von den Arbeitsgruppen verwalteten Teilen oft derartig massiv, dass sich Teile des Designs innerhalb eines Jahres vollkommen verändern können.

CIM bietet zwar eine ganze Reihe von Attributen an, ist jedoch eher in bestimmten Teilen des Device-Managements (wie zum Beispiel bei der SNIA) zu gebrauchen. Die stetige Veränderung und die nur sehr stiefmütterliche Behandlung von Service-Management bei CIM sprechen jedoch gegen eine direkte Verwendung von CIM im Service anagement.

3.6 Internet Managementarchitektur

Die Internet Managementarchitektur - oft auch nur nach ihrem Managementprotokoll **SNMP** SNMP-Protokoll genannt - hat derzeit eine marktbeherrschende Stellung in der PC basierten IT-Welt. Dies hat sie unter anderem der Tatsache zu verdanken, dass die Management-Konzepte wesentlich einfacher aufgebaut und deshalb schnell und relativ günstig auch in billige Produkte einzubauen sind. Ein weiterer Grund liegt in dem einfacheren Standardisierungsverfahren über sogenannte **request-for-comments (RFCs)**, welches vom Internet Architecture Board (IAB) durch die Internet Research Task Force (IRTF) und die Internet Engineering Task Force (IETF) kontrolliert wird.

Arbeitsgruppen erarbeiten hierfür Vorschläge für Standards, welche dann nach einem Rezensionsprozess entweder zu einem “proposed standard”, “draft standard” oder “internet standard” werden.

Dabei erhalten sie auch eine eindeutige (aufsteigende) RFC-Nummer, welche das Dokument während seiner gesamten Lebenszeit behält. Lediglich die Dokumente, welche es zum “internet standard” - auch einfach nur “Standard” genannt - schaffen, erhalten zusätzlich noch eine STD-Nummer. Soll ein neuer RFC einen alten ersetzen, so wird der alte deswegen nicht aus dem System gelöscht, sondern erhält entweder ein “update” oder das neue Dokument vermerkt den alten RFC als “obsolete”. Das ursprüngliche Dokument erhält jedoch keinen Vermerk darauf und bleibt in Gänze unangetastet. Dieser Prozess geht zurück bis in die Urzeiten des Internet als 1969 das ARPANET aufgebaut wurde. Es ist innerhalb des eigenen Systems aktuell in der 3. Revision in [Brad 96] als Best Current Practice dokumentiert.

Organisationsmodell

Das Organisationsmodell bei der Internet Managementarchitektur ist asymmetrisch angelegt. Definiert sind dabei ein Manager und ein Agent, welche jeweils über das Managementprotokoll SNMP miteinander kommunizieren. Um auch Komponenten einbinden zu können, welche kein SNMP sprechen können, wurde in das Konzept um sogenannte Proxy-Agenten erweitert, die diese Ressourcen überwachen und steuern können. Seit SNMPv2 wird auch die Kommunikation zwischen 2 Agenten ermöglicht, so dass auch Hierarchiebildungen ermöglicht werden.

Informationsmodell

Das Informationsmodell ist wesentlich weniger mächtig als zum Beispiel das von OSI. Managementobjekte werden zwar definiert, es liegt jedoch kein objektorientierter Ansatz zugrunde. Es wird stattdessen ein Ansatz verfolgt, der datentyporientiert ist. Diese Managementobjekte können entweder Variablen oder Tabellen darstellen und werden in einer Baumstruktur (Internet-Registrierungsbaum) angeordnet. In diesem Baum werden die Managementobjekte nach bestimmten Vorgaben registriert. Die Namensgebung und der Zugriff auf die einzelnen Objekte erfolgt entsprechend dieser Baumstruktur. Dadurch erfolgt auch eine Gruppierung inhaltlich zusammengehöriger Objekte nach bestimmten Kriterien. Tabellen und Gruppen bilden dabei die Knoten und Variablen die Blätter des Baumes. Abbildung 3.8 (welche in [HAN 99] auf Seite 161 zu finden ist) zeigt einen Ausschnitt dieses Baumes mit der umgangssprachlichen und der numerischen Bezeichnung der Knoten. Es wird hier nur eine begrenzte Anzahl an Knoten dargestellt. Blätter befinden sich weiter aussen am Baum.

Die Definition der Managementobjekte erfolgt über eine Templatesprache, die auf ASN.1 basiert. Die Informationen, welche dabei ein einzelner SNMP-Agent zur Verfügung stellt, werden als Management Information Base (MIB) definiert. Aufgrund der streng hierarchischen Struktur kann dabei nur einer Form der Gruppierung Rechnung getragen werden. Eine Wiederverwendbarkeit von Gruppen ist nicht möglich. So ist ein Komponente aus dem Hause HP immer unter dem Präfix .1.3.6.1.4.1.11 zu finden, eine aus dem Hause IBM unter .1.3.6.1.4.1.2. Dass beide Komponenten dabei einen Drucker darstellen, ist aus dem jeweils verwendeten Pfad zur eigentlichen Node, welche das Objekt kennzeichnet, nicht zu erkennen. Somit ist es auch unmöglich, gleichartige Produkte aufgrund der Nomenklatur des Internetinformationsmodelles zu kennzeichnen. Dadurch ergibt sich eine unübersehbare Anzahl an Objekten, deren jeweilige Funktion nicht eingeschätzt werden kann.

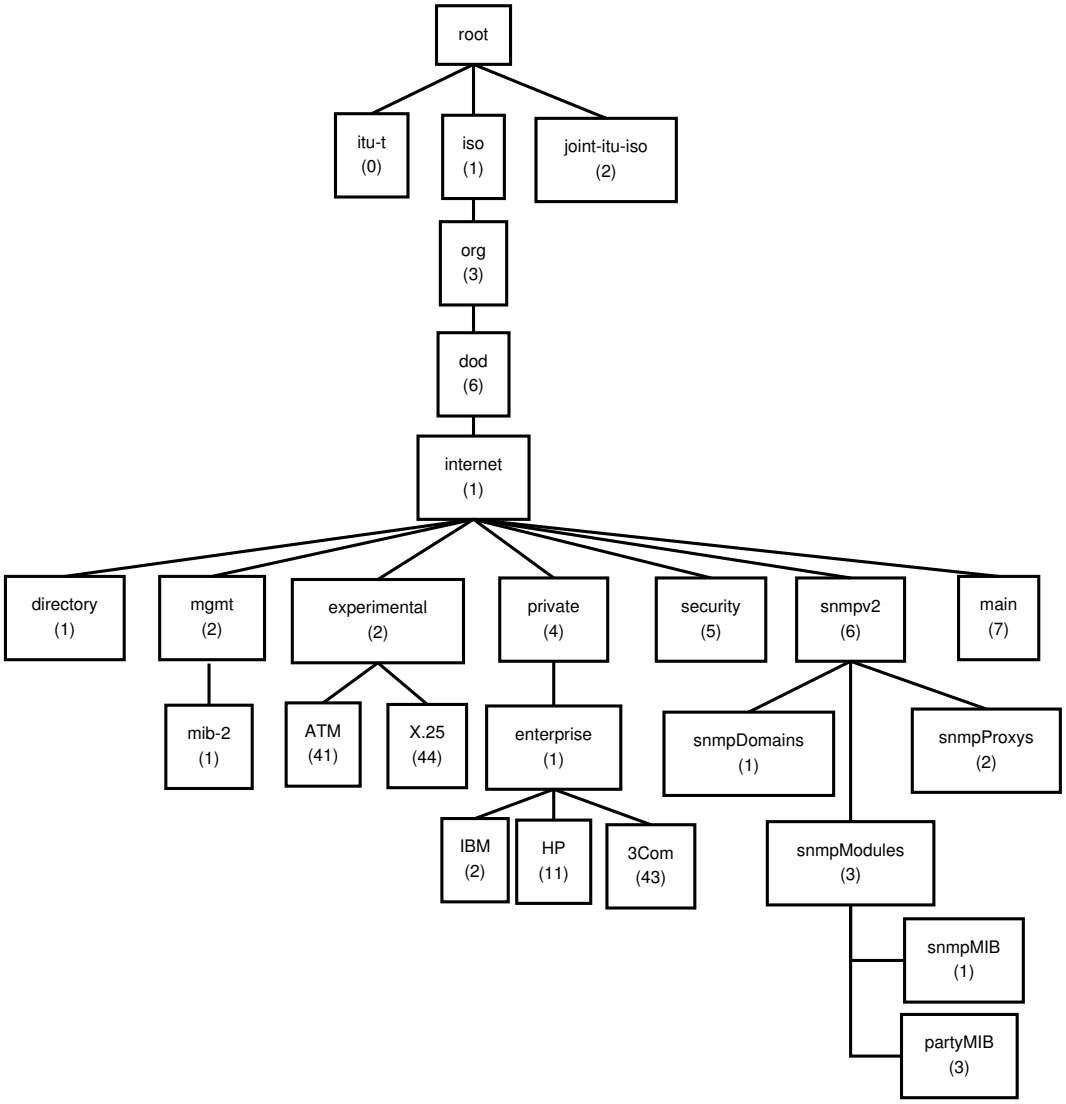

Abbildung 3.8: Ausschnitt aus dem Internet Registrierungsbaum

Kommunikationsmodell

Das Kommunikationsmodell des Internet Managements basiert auf SNMP. Dieses setzt auf dem verbindungslosen User Datagramm Protocol (UDP) auf und hat keine weiteren Sicherheitsvorkehrungen zur Überprüfung der Übertragung. Es sind verschiedene Protokollinteraktionen zum Lesen und Setzen der MIB-Variablen definiert. Einen Sicherheitsmechanismus zur Überprüfung der Autorisierung von Anfragen gab es in der ersten Version von SNMP (auch SNMP-v1 genannt und in [CFSD 90]) definiert) de facto nicht. Die verwendeten Community Strings entsprechen einem frei verfügbaren und weit sichtbaren Banner als Identifikationsmerkmal. Erst in Version 2 ([CMRW 93], [GaMc 93], [MaVi 96], [CMRW 96] obsoleted by [Pres 02]) gab es erste Ansätze, diesem durch ein zusätzliches Passwort Einhalt zu gebieten. Ein ordentliches Rechtemanagement wurde erst mit SNMPv3 ([HPW 02]) im Dezember 2002 eingeführt. Bis heute wird jedoch hauptsächlich das unverschlüsselte auf dem “community concept” basierende SNMPv1 eingesetzt.

3.6.1 MIB-II: Management Information Base of network Management of TCP/IP-based internets

Diese Erweiterung der ursprünglichen MIB hat weitere, auf das Internet-Management hin zugeschnittene Gruppen in den Managementbereich mit aufgenommen. Diese bieten die grundlegenden Parameter an, welche für das Management der meisten Systeme, die über das Internet erreichbar sein sollen, vonnöten sind. Die aktuellen Gruppen mit den dazugehörigen Nummern, welche ihre Position im Baum darstellen, sind wie folgt definiert:

System (1) Diese Gruppe enthält allgemeine Informationen zu dem System, wie eine allgemeine Beschreibung, einen eindeutigen Identifikator, eine Variable für den Aufenthaltsort einer Komponente, wie lange das System schon aktiv ist, Kontaktinformationen zu verantwortlichen Personen und eine Integer Nummer, welche den Protokolllayer (nach OSI) angibt, der von der Komponente unterstützt wird.

Interfaces (2) Beinhaltet alle Schnittstellen, welche das System nach außen hin unterstützt, die dabei verwendeten Protokolle, sowie verschiedenste Zähler für Statistiken und so weiter. Eine ausführliche Beschreibung hierzu ist in [McKa 00] niedergelegt.

Protocol Groups (4,5,6,7,8,11) Für alle bekannten Protokolle des Internet (ip, icmp, tcp, udp, egp, snmp) sind hier Zähler angelegt, die statistische Daten zu jedem möglichen eingehenden und ausgehenden Verkehr enthalten. Andere Tabellen enthalten Informationen über Routing, Verbindungen, benachbarte Knoten und so weiter.

Transmission (10) In dieser Gruppe werden verschiedenste Managementinformationen abgelegt, die die Übertragungsprotokolle wie X.25, Ethernet, FDDI, ... betreffen. Dabei gibt es jeweils eine eigene MIB für jedes Protokoll, welches bis zu 100 verschiedene Variablen beinhaltet.

Allein aus dieser Aufzählung ist bereits zu erkennen, dass dieser Ansatz nicht für das Dienstmanagement sondern für das Devicemanagement auf Netzwerkebene gedacht ist. Hier lässt sich auch keinerlei Aggregation vorhandener Daten finden. Diese Aufgabe muss jeweils von den abfragenden Managern übernommen werden.

3.6.2 Application Management MIB

Nachdem es immer deutlicher wurde, dass es nicht ausreichend ist, nur die Netzwerkebene zu managen, wurden verschiedene RFCs herausgebracht, welche sich mit dem Management von Applikationen beschäftigen. Dazu gehören zum Beispiel eine “Mail Monitoring MIB” (RFC 2249), eine “Network Services monitoring MIB” (RFC 2248), “Relational Database Management System (RDBMS) Management Information Base” (RFC 1697) und so weiter.

Aufbauend auf den Informationen, die bei der Entwicklung dieser MIBs gewonnen wurden, haben Kalbfleisch, Krupczak, Preshun und Saperia mit [KKPS 99] eine Application Management MIB entwickelt, die eine Grundlage für das Management von Anwendungen darstellt. Innerhalb des Internet Registrierungsbaumes lautet der Präfix “1.3.6.1.2.1.62” (oder in der textuellen Form “iso.org.dod.internet.mgmt.mib-2.applicationMib”).

Die hierin angegebene Struktur eignet sich zur Darstellung der Verbindungen zwischen der Applikation, welche betrachtet wird und dem Betriebssystem bzw. der Hardware, auf welcher diese läuft.

Abstrahiert wird hier nur sehr schwach, so dass zum Beispiel ein Dateizugriff auf das Dateisystem und der Zugriff auf Netzwerkressourcen gemeinsam über einen generischen I/O-Channel geführt werden; auch als Stream bezeichnet. Dienste oder gar Abhängigkeiten zwischen Diensten können damit nicht modelliert werden.

3.6.3 Managed Objects for WWW-Services

Ein anderer RFC ([HKS 99]) mit dem Namen Definitions of Managed Objects for WWW services von Hazewinkel, Kalbfleisch und Schönwälder (Präfix "1.3.6.1.2.1.65") geht hier noch einen Schritt weiter und bietet eine MIB für verschiedene Webservices an.

Darunter verstehen die Autoren im Prinzip jegliche Anwendung, welche "Dokumente" über das Internet versendet. Das von ihnen deklarierte abstrakte "document transfer protocol" hat den Zweck, die Definitionen der MIB unabhängig von konkreten Protokollen, wie HTTP und FTP werden zu lassen. Ziel ist, eine standardisierte Sammlung von (gemanagten) Objekten zu erzeugen, die zu besserer Performance und besserem Fehlermanagement beitragen. Die Sichtweise ist dabei auf den Dienst gerichtet, den die verwalteten Dienste anbieten und nicht auf die Prozesse, welche dahinterstecken.

Dabei konzentriert sich der MIB jedoch auf allgemeine Informationen sowie Statistiken zu den Protokollen und zu den versendeten Dokumenten. Anforderungen, wie die Unterstützung eines Lebenszyklus werden hier ebenso wenig Genüge getan, wie der Erfassung von Konfigurationsdaten.

3.6.4 Zusammenfassung zur Internet Managementarchitektur

Zusammenfassend lässt sich sagen, dass das Internet Management Modell zwar viele gute Ansätze zeigt und im Netzwerkmanagement auch seine Bedeutung hat. Für eine Verwendung im Dienstmanagement fehlen ihm jedoch einige wichtige Aspekte. Wobei das Hauptproblem darin liegt, dass dieses Modell sich frei entwickelt und keine vorgegebene Struktur hat, die erfüllt werden muss. Genau dies führt aber auf der anderen Seite dazu, dass sich dieser Ansatz sehr weit verbreitet und schnell an aktuelle Produkte mit den jeweils angebotenen Attributen angepasst werden kann.
Da der Ansatz nicht objektorientiert ist, fehlt auch die Möglichkeit, Komponenten wiederzuverwenden, was zu einer hohen Zahl von Objekten führt, welche sich zum Teil nur durch Kleinigkeiten unterscheiden. Auch die Referenzierung anderer Komponenten ist aus dem Modell heraus nicht möglich und muss durch externe Datenbanken erfolgen.

3.7 DNS-Based Service Discovery (DNS/SD)

3.7.1 A DNS RR for specifying the location of services (DNS SRV) - RFC 2782

[GVE 00] Bei diesem Verfahren geht es um die Ermittlung von Diensten innerhalb einer Domain im Sinne der RFC 1034 [PoCr 71]. Da eine Organisationseinheit normalerweise getrennte Domains nur dann verwendet, wenn auch die Organisationseinheiten eine starke Trennung voneinander aufweisen, kann Domain in der praktischen Anwendung auch mit Organisationseinheit gleichgestellt werden. Bei dem hier vorgestellten Verfahren wird die Fähigkeit des Domain Name Systemes (DNS) verwendet,

welches nicht nur eine Abbildung von Namen auf Nummern, sondern auch eine Abbildung von Namen auf beliebigen Text vornehmen kann. Es verwendet einen bestimmten ResourceRecord (SRV) der Klasse IN, um eine Verbindung zwischen einem beliebigen Dienst und einem Rechner mit dem Port, über welchen dieser angesprochen werden kann, herzustellen. Die Service Types werden momentan von zwei verschiedenen Organisationen verwaltet. Die IANA verwaltet die Records, denen ein fester Port zugewiesen wurde. Dagegen verwaltet eine Arbeitsgruppe unter www.dns-sd.org diejenigen Dienste, welche unter einem beliebigen Port laufen sollen. Die IANA ist im Moment dabei, das Verfahren so umzustellen, dass für eine Registrierung nicht mehr unbedingt ein fester Port verwendet werden muss. Sobald dies abgeschlossen ist, wird sie die Aufgabe von dns-sd.org übernehmen.

3.7.2 Besondere Eigenschaften

Vorteile:

- Verwendung eines bereits erprobten Verfahrens zur Verwaltung der Dienste
- Trennung von Dienst und Rechner und auch Port
- Verfahren zum Load-balancing ohne Loadbalancer bereits integriert
- Internationalisierung der Dienstenamen unter Einbeziehung der IANA
- Implementierung unter Open-Source verfügbar

Nachteile:

- Keinerlei Abhängigkeiten
- Mögliche Verbreitung interner Informationen nach außen unerwünscht
- All or Nothing Prinzip in der Ansicht
- Lediglich Auskunftsdienst, also keine Verfahren zur Konfiguration der Dienste vorhanden

3.7.3 Praktische Anwendung → Apple Bonjour (Rendevous)

Bonjour, welches früher von Apple als Rendevous bezeichnet wurde, ist eine Implementierung des DNS-Based-Service Discovery 3.7 und entstand im Zuge der Umstellung von AppleTalk auf IP. Seit Mac OS X ist sie in das Betriebssystem integriert. Eine Windows-Referenzimplementierung existiert ebenfalls. Da man auf ein möglichst einfaches Handling Wert legt, ist normales DNS mit expliziten DNS-Servern nicht das Mittel der Wahl. Stattdessen wird mDNS - eine DNS Variante, die zwar nur lokal, dafür aber über ad hoc DNS Server per Multicast arbeitet - verwendet. Aufgrund schlechter Erfahrung mit dem Informationspush-Verfahren unter AppleTalk wird über ein Informationspullverfahren und mit Multicast gearbeitet. Aus den Bestrebungen von Apple ist die IETF Zeroconf Working Group entstanden. Es bleibt aber immer noch der Mangel, dass nur Dienste gefunden werden können und keine Dienstverwaltung und keine Abhängigkeitsmodellierung existieren. Apple verwendet Rendevous zudem für ihr XGrid-Projekt, um die freien Ressourcen zu announcen.

3.8 Zusammenfassung der Betrachtung

In dem vorhergehenden Kapitel wurden verschiedene Ansätze für das Service Management vorgestellt und daraufhin untersucht, ob sie die Anforderungen, wie sie in Kapitel 2.4 vorgestellt wurden, erfüllen. Für einen leichteren Überblick werden in Tabelle 3.2 die Möglichkeiten und Defizite noch einmal zusammengefasst.

Dabei ist zu sehen, dass die Vereinigungmenge aller Ansätze alle Anforderungen zuminest teilweise erfüllen würde, jedoch kein einzelner Ansatz soweit ausgearbeitet wurde, dass man diesen sofort für das Dienstmanagement verwenden könnte.

Besonders schwer wiegt dabei, wenn Abhängigkeiten nicht modelliert werden können. Ohne Abhängigkeiten wären Dienste nur isolierte Inseln im Raum.
Weiterhin hat sich das objektorientierte Paradigma als Standard bei der Entwicklung größerer Projekte durchgesetzt, so dass eine aktuelle Modellierung von Dienstmanagementinformationen dieses ebenfalls verwenden sollte.

Als Basis für ein eigenes Modell sollte man allerdings nur Modelle nehmen, welche bereits eine längere Entwicklungsphase hinter sich gebracht und daher einen hohen Reifegrad erreicht haben.

Diese Punkte erfüllen sowohl ITIL, als auch OSI aus einer allgemeinen, übergreifenden Sicht, als auch das Internet Management, welches seine Paradigmen aus einer Sammlung von jeweils benötigten Teilaspekten bezieht. Aufgrund der Fülle von einzelnen Attributen, welche dabei beim Internet Management anfallen, ist es zeitlich gesehen nicht ratsam, dieses als Ausgangspunkt für eine eigene Sammlung von Attributen für das Dienstmanagement zu verwenden.

Die Idee der Zusammenführung von technischer und betriebswirtschaftlicher Sicht wird nur von ITIL verfolgt, aber heutzutage immer wichtiger. Zum Beispiel werden Service Level Agreements nicht mehr auf technischer Basis mit “harten Anforderungen” wie einer Antwortzeit von “30ms” erstellt, sondern mit “weichen Anforderungen” wie einer angemessenen Reaktionszeit, die 30ms oder auch 90 ms sein kann. Andererseits bietet OSI eine solide Basis für Netzwerkmanagement und zeigt die Anforderungen an das IT-Management in allen seinen Facetten auf. Es ist umfangreicher als das Internet Management und sollte dieses ursprünglich auch ablösen. Da ITIL und OSI die wichtigsten Anforderungen erfüllen, beide angesehene Standards darstellen und dennoch von verschiedenen Gesichtspunkten des Managements ausgehen, empfehlen sie sich als Ausgangsbasis für die Erstellung eines Objektkataloges, wie er in dieser Arbeit erstellt werden soll.

existierende Modelle / Anforderungen	MNM-Service Modell	ITIL	OSI	SID	CIM	Internet Management	DNS-SD
Herstellerunabhängigkeit	✔✔	✔✔	✔✔	✘	✔	✔✔	✔✔
Offenheit	✔✔	✔[1]	✔[1]	✘✘	✘	✔ ✔	✔✔
Reifegrad	✔	✔ ✔	✔✔	✔	✔	✔✔	0
Verbreitung	✘✘	0	✘✘	✘✘	0	✔✔	✔
Einfachheit für Anwender	0	✘	✘✘	0	0	0	✔
Flexibilität bzgl. Erweiterungen	✔	✔	✔✔	✔	✔	✔	✔✔
Objektorientiertheit	✔	✘✘	✔✔	✔✔	✔✔	✘✘	✘✘
Allgemeiner Dienstbegriff	✔	✔	0	✔	✘	0	✘
Abstraktion	✔	✔	✘✘	✔	✘✘	✘✘	✘
Standardisierte Schnittstellen	✔✔	0	✘	✔	✔	✘ ✘	✔
Abhängigkeiten	✔	✔	0	✔	✔✔	✘✘	✘✘
Lebenszyklus	✔	✔✔	✘	✘✘	✘✘	✘✘	✘✘
Vorhandene MIBs / Objektkataloge	✘✘	✘	0	✔	0[2]	✔	✘✘
Abbildung auf Programmiersprachen	0	✘ ✘	0	✘	✘	0	✘
Dienstattribute	✘	0	✔	✔	✔	✔	✘✘
Methoden	✘	✘	✘	0	✘✘	0	0
Historie	✘✘	✔✔	✘✘	✘✘	✘✘	✘	✘✘

[1] kostenpflichtiger Zugang über Mitgliedschaften
[2] in Spezialbereichen (Bluefin)

Legende:

Erfüllt die Anforderung
- ✔✔ umfassend
- ✔ weitgehend
- 0 teilweise
- ✘ eingeschränkt
- ✘✘ unzureichend

Tabelle 3.2: Möglicheiten und Defizite existierender Ansätze

Kapitel 4

Ableitung von aussagekräftigen Dienstmerkmalen

Wie man bereits aus dem vorhergehenden Kapitel ersehen konnte, gibt es bereits eine ganze Reihe von Ansätzen zum Dienstmanagement. Einige davon erfüllen die Anforderungen auch bereits in vielen Teilen. Bisher stellt aber noch kein Ansatz sowohl einen generellen Überblick her als auch wichtige Attribute für das Dienstmanagement dar. So hat CIM zwar viele Attribute, eine Auswahl wichtiger Attribute muss jedoch vom Anwender selber gefunden werden. ITIL wiederum stellt zwar den Überblick her, bietet aber keine direkt anwendbaren Attribute. Somit muss jeder, der ein System basierend auf ITIL herstellen will, sich diese selber erarbeiten. Dies führt in der Folge dazu, dass eine große Vielfalt von verschiedenen Attributen und Methoden entsteht, welche nicht ineinander überführt werden können. Dieser Vorgang und das daraus folgende Ergebnis wurde in der Bibel mit dem "Turmbau zu Babel" umschrieben. Die Sprachen von damals sind den Implementierungen von heute gleichzusetzen. Um zumindest an den Schnittstellen in der selben Sprache zu sprechen, ist die Entwicklung eines Objektkatalogs wichtig, welcher die wichtigsten Attribute festhält.

Sowohl ITIL als auch OSI stellen eine Reihe von Aufgaben vor, welche für erfolgreiches Dienstmanagement gelöst werden müssen. Dabei sind die von OSI etwas allgemeiner gehalten als die von ITIL. ITIL stellt mit ihren "activities" bereits eine Liste an Aufgaben zusammen, welche einfach nur noch Schritt für Schritt abgearbeitet werden muss. In diesem Kapitel wird nun mit Hilfe dieser in ITIL und OSI dargestellten Aufgaben ein Katalog von Managementaufgaben erstellt werden. Anschließend gilt es, Attribute zu finden, welche für die Erfüllung der Aufgaben dieses Kataloges notwendig sind.

4.1 Methodik zur Ableitung

Mit der Erstellung des Aufgabenkataloges wird die Verwendung einer Methodik ermöglicht, welcher der bei ITIL uns OSI verwendeten entgegengesetzt ist. Als Ergebnis wird ein Objektkatalog entstehen, welcher zwar die Ideen und Aufgaben von der beiden Ausgangsansätze beinhaltet aber nicht deren hohe Abstraktion. Diese beiden grundsätzlichen Methodiken sind als "top-down" (Erstellung von Feinerem aus dem Groben) und "bottom-up" (Erstellung der Generalisierung aus den Einzelteilen) bekannt.

4.1.1 Top-down vs. bottom up

Der Teil der Ansätze, welcher eine praktischere Ausrichtung verfolgt, hat in der Entwicklung eine Vorgehensweise verfolgt, welche als **bottom-up** Ansatz bezeichnet wird. Hierbei wird ausgehend von existierenden Anforderungen aus dem jeweils eigenen Bereich und den dort verwendeten Attributen (z.B. im Protokoll SNMP vorhandenen MIBs) ein Objekt-Katalog erstellt, der an diese Anforderungen angepasst ist. Die Frage lautet hier "Welche Informationen liefern mir die Komponenten?" bzw. "Welches Interface ist am einfachsten zu implementieren?". Diese Vorgehensweise hat den Vorteil, dass der Objektkatalog zunächst schlank ausfällt, einfach zu designen ist und der Fokus auf der eigenen Architektur bleiben kann. Der Ansatz hat jedoch auch einige Nachteile. So wird der Objektkatalog

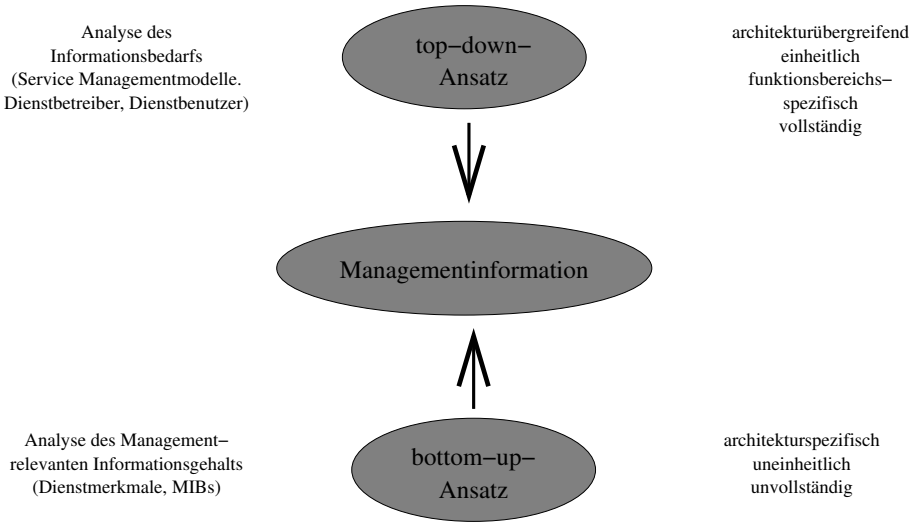

Abbildung 4.1: Ansätze zur Erstellung eines Objektkataloges

nach mehrfacher Überarbeitung und dem Einbau weiterer Komponenten umfangreich und dadurch unübersichtlich. Das führt dazu, dass er immer wieder vollkommen neu überarbeitet werden muss. Besonders problematisch wird dieser Ansatz jedoch dann, wenn organisationsübergreifend gearbeitet werden soll und verschiedene Systeme zusammenarbeiten sollen, welche auf dem bottom-up Ansatz basieren. Hier wirkt sich dann die spezielle Anpassung an die eigenen Anforderungen störend aus, da der Aufbau zwischen den Systemen zu weit divergiert.

Um den Problem des bottom-up Ansatzes zu entgehen, gibt es den **top-down** Ansatz. Bei diesem werden zunächst die Anforderungen und der Informationsbedarf verschiedener Dienstbetreiber und -nutzer analysiert und aus diesen zunächst ein sehr abstraktes Modell entwickelt. Das Ergebnis ist dann ähnlich dem des MNM-Dienstmodelles (Kapitel 3.1).
Ein weiterer Vorteil dieser Vorgehensweise ist, dass das Resultat nicht nur architekturübergreifend einheitlich ist, sondern auch gerade durch seine Vollständigkeit einen direkten organisationsübergreifenden Austausch von Informationen erlaubt. Ein weiterer Vorteil ist, dass der Informationsaustausch nicht über komplexe, abstahierende und damit fehlerträchtige Übersetzungen ablaufen muss. Dieser Ansatz hat jedoch, wenn er auf einer zu abstrakten Ebene bleibt, ebenfalls seine Nachteile. Die Aufgabe ist nämlich noch nicht erfüllt, wenn ein abstraktes Modell besteht, welches das Dienstmanagement "im Prinzip" beschreibt. Die Erstellung eines Objektkataloges, welcher die tatsächliche Anwendung des abstrakten Modelles erst ermöglicht, erfordert ebenso viel Arbeit. Er ist für die Anwendbarkeit

des Systemes sogar noch wichtiger als das Modell.

Eine weitergehende Diskussion zum Thema “bottom-up” oder “top-down” wurde unter anderem auch in [Neum 93] geführt, welchem auch die Grundlage zu Graphik 4.1 entnommen wurde.

4.1.2 Vorgehensweise bei der Ableitung

Da es in dieser Arbeit zum einen um einen Ansatz geht, der das Dienstmanagement innerhalb des LRZ zu modellieren in der Lage ist, darüber hinaus aber auch allgemein Anwendung finden soll, wird im Folgenden hauptsächlich auf dem top-down Ansatz aufgebaut. Die beiden Ansätze ITIL (Kapitel 3.2) und OSI (Kapitel 3.3) werden dabei eine Ausgangsbasis bilden, aus der die notwendigen Attribute bestimmt werden sollen, um Dienstmanagement durchzuführen.
ITIL ist deswegen eine gute Basis, weil es erstmals einen Ansatz verfolgt hat, der nicht von der Sichtweise der Technik gesteuert ist, sondern aus der Sicht des Managements aufgebaut wurde. Der Nachteil von ITIL ist allerdings, dass es kaum Hinweise auf Attribute gibt, welche zu verwalten sind, sondern den Fokus auf die Prozesse legt, welche für das Dienstmanagement notwendig sind.
OSI ist eine gute Basis, weil es bereits alles mitbringt, was für das Management notwendig wäre. Allerdings ist die Ausrichtung noch auf das Systemmanagement fokussiert und es hat sich trotz seines Alters bis heute nicht durchsetzen können. Dies ist zum Teil auch der Tatsache geschuldet, dass der Objektkatalog sehr komplex wird und nur in einer kleinen Zahl von Programmiersprachen überhaupt abgebildet werden kann. Der für diese Arbeit relevante Teil ist dabei die Dimension des funktionalen Managements.

Bei der Betrachtung der beiden vorgenannten Ansätze wird ein Katalog von Fragen aufgebaut werden, welche für die Lösung der Aufgaben im jeweiligen Managementbereich beantwortet werden müssen. Teilweise werden auch direkt Attribute angesprochen, auf die in den Ansätzen besonderer Wert gelegt wird. Um später auf die Fragen und Attribute verweisen zu können, werden sie nach folgendem Schema markiert:

ITIL-i	Frage zu einer Aufgabe in ITIL
OSI-i	Frage zu einer Aufgabe aus OSI
ITIL-A	Attribut zu ITIL
OSI-A	Attribut zu OSI

Dabei steht “**i**” für eine römische Ziffer und “**A**” für einen beliebigen Buchstaben.

Aus den gesammelten Fragen wird eine weitere Liste von Fragen erstellt werden, die eine Zusammenfassung der bereits erarbeiteten darstellen. Dabei werden zu den Fragen jeweils Attribute vorgeschlagen, welche diese zu beantworten helfen.

Die auf diese Weise gewonnenen Attribute stellen die Grundlage für einen ersten Objektkatalog für das Dienstmanagement dar. Dieser wird dann mit Hilfe von Design Pattern vereinfacht und verfeinert, so dass am Schluss ein einfaches Klassendiagramm zu Tage tritt, welches eine Darstellung des Ausgangszenarios erlaubt.

- functional requirements support, and the level of integration with, for example, Service Delivery processes and tools
- data structure, data handling and integration, including the capability to support the required functionality
- integration of multi-vendor infrastructure components, and the need to absorb new components in the future - these will place particular demands on the data-handling and modelling capabilities of the tool
- conformity to international open standards
- flexibility in implementation, usage and data sharing
- usability: the overall ease of use permitted by the User interface
- service levels: performance and availability
- distributed clients with centralised shared database (e.g. client-server)
- backup-up, control and security provisions
- the quality of information provided by the supplier, and its validation by contact with other Users.

Tabelle 4.1: Wichtigste Anforderungen laut ITIL ([ITIL 00], Seite 246)

4.2 Managementaufgaben nach ITIL

ITIL gibt, wie bereits bemerkt wurde, keine Modelle vor, wie die Datenhaltung beim Servicemanagement zu erfolgen hat. Vielmehr geht es davon aus, dass die Daten entsprechend der Vorgehensweise der verschiedenen Arbeitsgruppen zur Verfügung zu stehen haben. So gibt es lediglich ein paar Anforderungen an ein softwaregestütztes Service-Managementsystem vor, wenn dieses in Übereinstimmung mit den Vorgehensweisen von ITIL verwendet werden soll (Tabelle 4.1).
Diese Aufgaben sind jedoch so allgemein gehalten, dass sie zwar einige Eigenschaften eines Tools und des zugrunde liegenden Datenmodelles beschreiben, aber für die Erschaffung eines Objektkataloges nicht zu gebrauchen sind.

Wesentlich geeigneter sind dafür die verschiedenen Managementaufgaben, welche in ITIL definiert werden.

4.2.1 Untergliederung der Managementaufgaben bei ITIL

Wie in Anhang A nachzulesen, unterteilt ITIL die Aufgaben in verschiedene Bereiche. Im Folgenden werde ich mich allerdings zunächst auf die Bereiche “Incident Management and Problem Management”, “Configuration Management” und “Change Management” beschränken.
Einen groben Überblick über das Zusammenspiel dieser Bereiche stellt Abbildung 4.2 dar.

Das *“Service Level Management”* (SLA Management) eines der wichtigsten Bereiche bei der inter-organisatorischen Verwaltung. Wichtig vor allem, da es hier um empfindliche Strafen für den Dienstanbieter bei Verstößen gegen die Vereinbarungen gehen kann. Das zu beachtende Spektrum würde hierdurch zudem wesentliche erweitert und den Rahmen sprengen. Das *“Capacity Management”*, welches enge Beziehungen zum SLA Management aufweist, wird folglich auch nicht behandelt. Den zu erwartenden Umfang kann man bereits dadurch erahnen, dass ihm in ITIL eine eigene *“Capacity Management Database”* zugeordnet wird.

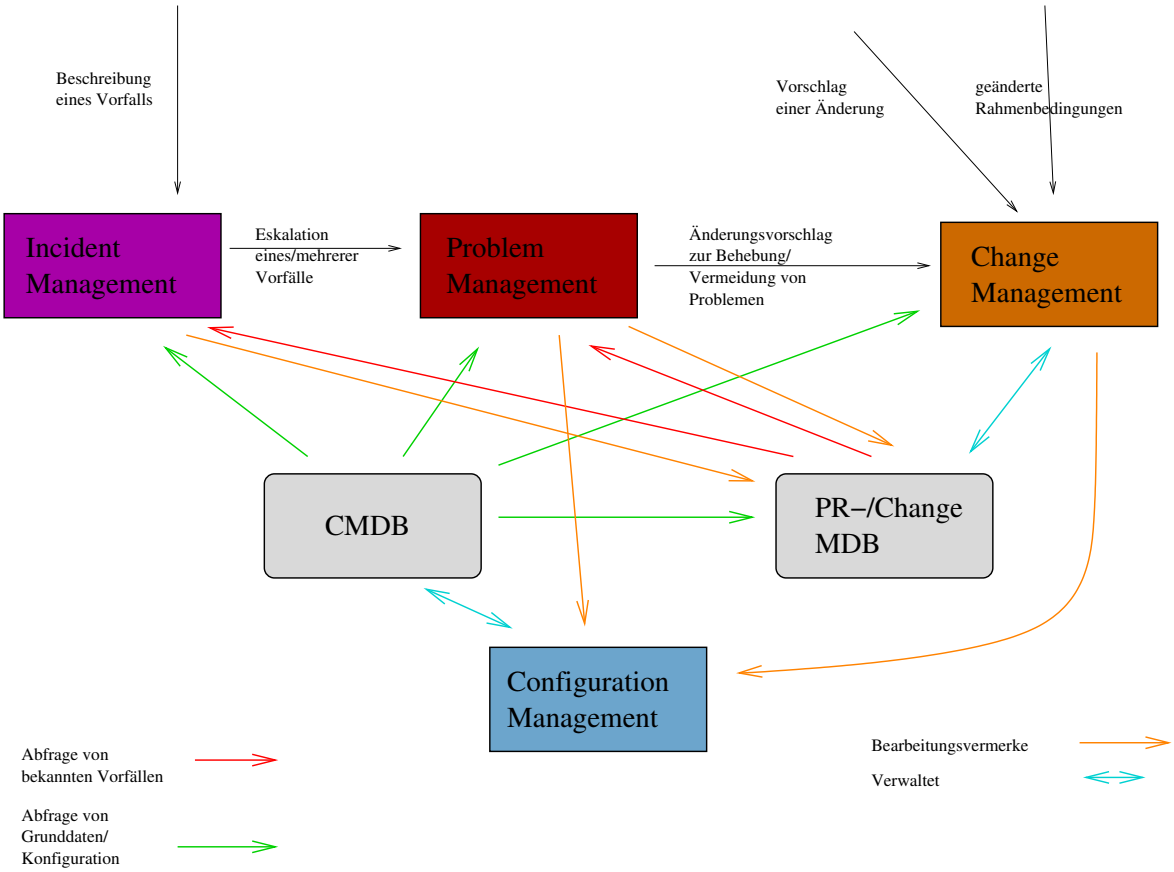

Abbildung 4.2: Zusammenspiel der Bereiche Incident/Problem/Change/Configuration Management bei ITIL

Auch das “IT Service Continuity Management” werde ich nicht explizit bearbeiten. Diese Komponente ist zwar eine der zentralen Elemente eines jeden Auditings, doch gibt es hierfür seit “Sarbanes Oxley” einige Modelle (wie zum Beispiel Cobit der Information Systems Audit and Control Association (ISACA)), um den organisatorischen Teil erschöpfend zu bearbeiten. Die wichtigsten technischen Details jedoch sollten bereits durch eine ordentlich geführte CMDB beantwortet werden können. Natürlich haben Änderungen durch das *“Change Management”* auch hier ihre Auswirkungen.
Die Stelle, um die sich die IT am Seltensten kümmert, ist das *“Financial Management”*. Aufgrund der orthogonalen Interessen wäre für die hier benötigten Daten jedoch ein anderer Ansatz vonnöten.
Das *“Availabiltity Management”* möchte ich hier als Teilstruktur des “Problem Managements” bearbeiten, da die hierfür benötigten Teile aus eben diesem entstammen.
Die Aufgaben des *“Service Desk”* sind kommunikativer Struktur. Dieser muss als Schnittstelle zwischen “Kunden” und IT Zugriff auf die Daten der verschiedenen Bereiche haben, hat aber keine gesonderten Anforderungen.
Das *“Release Management”* befasst sich mit der organisatorischen Verwaltung der Inbetriebnahme von Komponenten, welches ich hier aber vereinfacht als Teil des Change Managements behandeln werde.

Störungs- und Problemmanagement

Diese beiden Bereiche, die in der ausführlichen Beschreibung getrennt behandelt werden, befassen sich mit der Lösung im Betrieb auftretender Probleme. Das Incident Management ist hierbei die sozusagen “erste Stufe” und behandelt die Erfassung und Behandlung von im laufenden Betrieb gemeldeten Störungen im Nachhinein. Hierzu gehören aber auch Meldungen wie “Toner low”. Das Hauptaugenmerk liegt hier auf der Wiederherstellung der Funktionalität des Dienstes für den Anwender. Auf der anderen Seite befasst sich das Problemmanagement mit der Entfernung von Störungsursachen und mit der proaktiven Behandlung solcher.
Während der Auslöser für das Störumgsmanagement Meldungen an die Administration sind, werden Probleme über Statistiken oder von mit der Störung beauftragten Personen bestimmt. Auslöser kann eine erhöhte Anzahl von Incidents zu einem bestimmten CI sein. Aufgrund dieser kann es dann zu einem Change (siehe Change Management, Seite 59) kommen. Zu einem proaktiven Fehlermanagement im Rahmen des Availability Managements gehört natürlich auch, dass bereits im Vorfeld bestimmt werden kann, welche Komponenten oder gar Businessprozesse vom Ausfall einer Komponente betroffen sind.

4.2.2 Störungsmanagement

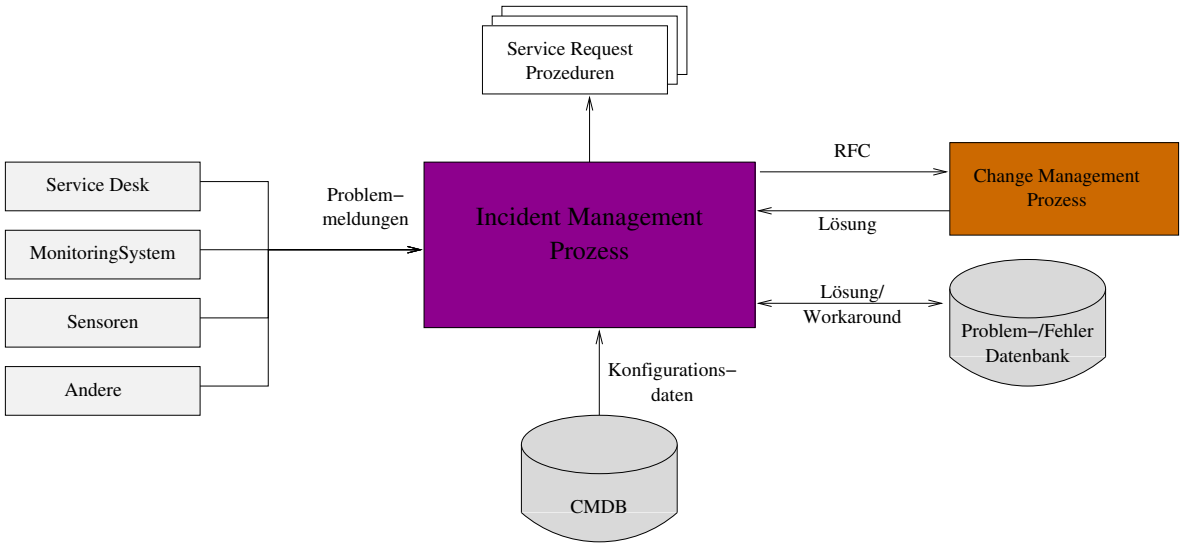

Abbildung 4.3: Ein-/Ausgaben des Incident Management Prozesses

4.2.2.1 Incident detection and recording

Beim Störungsmangement gibt ITIL sogenannte “acitvities” (Aktivitäten) vor, die eine Systematik zur Behandlung von Störungen darstellen. Hierfür wird jeweils angegeben, welche Daten die Aktivität zur Verfügung gestellt bekommt, was getan wird und welches Ergebnis am Ende erwartet wird. Um die Querverbindungen zu den entsprechenden Aktivitäten bei ITIL in [ITIL 00] zu erleichtern, werden in den folgenden Kapiteln die Namen aus ITIL übernommen.

- unique reference number
- Incident classification
- date/time recorded
- name/id of the person and/or group recording the Incident
- name/department/phone/location of User calling
- call-back method (telephone, mail etc.)
- description of symptoms
- category (often a main category and a subcategory)
- impact/urgency/priority
- Incident status (active, waiting, closed etc.)
- related Configuration Item
- support group/person to which the Incident is allocated
- related Problem/Known Error
- resolution date and time
- closure category
- closure date and time

Tabelle 4.2: Basisdatensatz für das Incident Management ([ITIL 00], Seite 92)

Vorgehensweise

ITIL-i *Templates für die Störungsaufnahme*
Nachdem die Störung erkannt wurde, wird sie zumeist über den "Service Desk" aufgenommen. Dazu hat es sich als vorteilhaft erwiesen, ein Template zu verwenden. Dadurch soll sichergestellt werden, dass auch die Daten mit aufgenommen werden, welche nach Ansicht des Eintragenden nicht wichtig sind, sich aber im Verlauf der Untersuchungen als wichtig erweisen können. Manche sind vielleicht auch erst in einem späteren Zeitraum interessant, zum Beispiel bei einer Untersuchung der Effektivität bzw. Auslastung der mit dem Incident Management betrauten Abteilung. Eine Übersicht über derartige Attribute, welche sich aus dem "best practice"-System von ITIL ergeben haben, ist in Tabelle 4.2 zusammengestellt.

ITIL-ii *Serviceverantwortlicher*
Im Falle einer schwerwiegenden Störung ist es sinnvoll, bereits zu einem möglichst frühen Zeitpunkt den für den Service Verantwortlichen zu informieren.

ITIL-iii *Service Level Agreements*
Um die Auswirkung der Störung auf die beteiligten Businessprozesse beurteilen zu können, ist nicht nur die rein technische Seite von Bedeutung. So kann der Ausfall eines ganzen Netzsegmentes weniger Bedeutung haben als der Ausfall eines einzelnen Dienstes. Dies ist immer dann der Fall, wenn die Bedeutung durch ein Service Level Agreement (SLA) vorgegeben wird. Somit müssen auch die betroffenen SLAs mit dem Vorfall verknüpft werden.

Ergebnis
Durch den hier vorgestellten ersten Schritt in der Verarbeitung einer Störung durch die Erstellung eines Basisdatensatzes soll sichergestellt werden, dass alle Details zu der Störung

möglichst vollständig und aktuell sind. Hierdurch wird es auch leichter, Diskrepanzen zu den Daten in der CMDB zu erkennen und darauf aufbauend gleich auf Unstimmigkeiten hingewiesen zu werden. Zu guter Letzt wird es hierdurch auch möglich gemacht, die Kunden zu informieren, wenn die Störung behoben wurde.

4.2.2.2 Classification and initial support

Eine weitere Stufe besteht in der Ermittlung der Störungsursache. Auch die Priorität des Vorganges wird hier festgelegt.

Eingaben

Hierfür stehen idealerweise neben der bereits im vorhergehenden Schritt erfassten Störungsmeldung auch alle Daten aus der CMDB zur Verfügung. Eine Datenbank mit bestehenden Fehlern/Problemen sowie Lösungsmöglichkeiten vereinfacht die Aufgabe ebenfalls.

Vorgehensweise

ITIL-iv *Störungsursache*

Zunächst einmal ist es wichtig, festzustellen, welches die eigentliche Ursache für die Störung ist. Ein Großteil der Störungsmeldungen hat einfache Lösungen wie das Ändern eines Passwortes oder das Auswechseln des Toners. Manche Betriebssysteme verlangen auch einen regelmäßigen Neustart für einen reibungslosen Betrieb. Derartige Lösungen sind oft direkt bekannt und benötigen keine weitere Untersuchung. Andere Störungen treten nicht so häufig auf oder sind schwieriger zu beheben. In diesem Fall bringt eine Suche in einer Datenbank mit Lösungen zu bekannten Fehlern/Problemen oft das gewünschte Ergebnis.

Die Klassifizierung einer Störung ist ein größerer Prozess, da hier verschiedene Elemente zusammengeführt und ausgewertet werden müssen.

ITIL-v *Störungsquelle*

So ist es hier wichtig zu wissen, welcher Dienst/welche Dienste von der Störung wirklich betroffen ist/sind. Bei einer Störungsmeldung durch den Kunden werden oft nur die Symptome beschrieben, welche er bemerkt. Diese müssen jedoch nicht unbedingt direkt auf die Ursache hinweisen. Um im vorgestellten Szenario zu bleiben, ist die Nichterreichbarkeit einer Webseite nicht unbedingt dem Ausfall des Webservers geschuldet. Eine mögliche Ursache könnte auch ein ausgefallener Switch sein.

ITIL-vi *Kunden zu einem SLA*

Wie bereits vorher beschrieben, ist in der business-orientierten Welt einer der wichtigsten Parameter beim Umgang mit dem Kunden und seinen Problemen sein SLA. Wenn also SLAs bestehen, die diese Dienste betreffen, so sollten diese in die Überlegungen für die Klassifizierung der Störung mit einbezogen werden. Eine Übersicht über die Beziehungen von Dienst, Kunde und SLA im Falle einer Störung stellt Abbildung 4.4 dar.

ITIL-vii *Störungsumfang*

Entsprechend der Ursache für die Störung ist es natürlich möglich, dass weitere Dienste betroffen sind. Hierdurch erweitert sich dementsprechend auch der Horizont der beteilig-

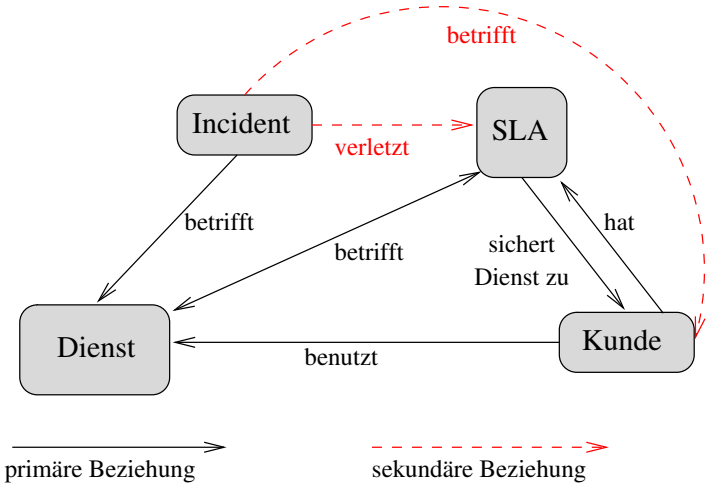

Abbildung 4.4: Beziehungen von Dienst/Kunde/SLA beim Incident-Management

ten Kunden und SLAs. Zudem kann hierdurch oft erst der wirkliche Umfang (**ITIL-A**) einer Störung und die Komplexität der Störung bestimmt werden: Andererseits ist es auch möglich, dass die wirkliche Ursache für einen Fehler erst durch den Vergleich der gestörten Dienste gefunden werden kann. Kann ein Kunde zum Beispiel auf mehrere Webseiten nicht zugreifen, die auf unterschiedlichen Servern liegen, dann ist die Störung eines Servers weniger wahrscheinlich als der Ausfall einer anderen Komponente.

ITIL-viii *Von der Störung betroffene Kunden*
Die Reichweite einer Störung beeinflusst die Auswirkungen auf den Betrieb. Dazu gehören neben den zugehörigen SLAs auch die betroffenen User (**ITIL-B**). Mit Hilfe der dazugehörigen Kontaktdaten der User (**ITIL-C**) ist es möglich, mit diesen in Kontakt zu treten und auf das Problem hinzuweisen oder weitergehende Informationen (**ITIL-D**) zu einer Störung zu erhalten. Mit Hilfe einer Prognose ist es dann oft wesentlich einfacher, weiteren Problemen aus dem Weg zu gehen.

ITIL-ix *Dringlichkeit der Störung*
Aufgrund der Auswirkungen einer Störung bestimmt sich auch die Dringlichkeit der Lösung. Teilweise wird die Dringlichkeit aber auch durch weitere Daten eingeschränkt. So wird der Ausfall eines Dienstes während eines angekündigten Wartungszeitraums nicht dieselbe Dringlichkeit haben, wie während des Hochbetriebes.

ITIL-x *Voraussichtlicher Aufwand*
Manche Störungen sind zwar nicht besonders dringend, es wird aber auch nicht damit gerechnet, dass die Beseitigung besonders viel Zeit in Anspruch nehmen wird. So ist die Änderung eines Passwortes für einen Kunden in bestimmten Konstellationen nicht besonders dringend, aber es würde auch nur ein paar Minuten in Anspruch nehmen und der Kunde ist zufrieden. Dies ist oft ein Faktor, der in der Klassifizierung eines Fehlers eine Rolle spielt.

ITIL-xi *Spezialist für den Dienst*
Je nach Auswirkung auf den Betrieb und die Dringlichkeit kann es angebracht sein, dass der Bearbeiter direkt mit den Spezialisten zu einem Dienst (**ITIL-E**) in Verbindung tritt.

Dieser kann das Problem evtl. unter Umgehung der Problemdatenbank ohne Verzug lösen oder eine Prognose der zu erwartenden Störungsdauer abgeben.

ITIL-xii *Alternativ-Dienste*
Für die Kundenzufriedenheit ist wichtig, dass "seine" Störung möglichst schnell behoben werden kann. Hier hilft oft auch eine alternative Lösung oder ein schneller "Fix", der die Ursache zwar nicht behebt, es dem Kunden aber ermöglicht, weiterzuarbeiten.

Ergebnis
Durch diesen Vorgang ist es in den meisten Fällen bereits möglich, eine genaue Beschreibung des Problems sowie die Lösung zu dem Problem zu erhalten. Bei grundlegenden Problemen kann dies zu einem RFC (Request for Change) führen, der dann im Change Management weiterbearbeitet wird. Ansonsten kann die Störung zumeist entweder direkt behoben oder durch eine provisorische Lösung zunächst abgemildert werden. Durch die genauere Beschäftigung mit der Störung durch geschultes Personal werden die Dokumentationsdetails zu der Störung weiter vervollständigt und die Dringlichkeit unabhängig von den Vorstellungen des Kunden überprüft. Wenn das zu Grunde liegende Problem nicht behoben werden kann, ist es möglich, die Störung dann auch zum second- oder gar third-line Support zu eskalieren.

4.2.2.3 Investigation and diagnosis

Wenn die Störung im vorherigen Schritt noch nicht behoben werden kann, weil es zum Beispiel keine Standardlösung oder standardisierte provisorische Lösung gibt, ist es die Aufgabe einer dedizierten Arbeitsgruppe, sich des Problemes anzunehmen, um dem Kunden zumindest die Nutzung des Dienstes unter bestmöglichen Bedingungen zu ermöglichen. Darunter ist zum Beispiel die Möglichkeit zu verstehen, bei einem ausgefallenen Drucker die Dokumente an einem anderen Standort auszudrucken.

Eingaben
Hierfür erhält diese Arbeitsgruppe die im vorherigen Schritt aktualisierten Daten zur Störung und die Details zur Konfiguration aus der CMDB.

Vorgehensweise

ITIL-xiii *Bestimmung der Ursache*
Die spezialisierte Arbeitsgruppe hat nun die Aufgabe, sich alle Aspekte der Störung anzusehen und auf der Basis eigenständig die tatsächliche Ursache herauszufinden und eventuell auch mögliche Lösungen zur Behebung der Ursache auszuarbeiten.

Des weiteren gehört zu diesem Schritt auch die regelmäßige Information des Kunden über den Stand der Fehlersuche.

Ergebnis
Dieser Schritt stellt die eigentliche Behandlung der Störung dar. Am Ende sollte ein Vorschlag für eine Lösung des Problemes stehen, welche in die Fehlerdatenbank aufgenommen werden kann. Des weiteren werden die Daten zur Störung noch einmal präzisiert.

4.2.2.4 Resolution and recovery

Der nächste Schritt bei der Störungsbehandlung beschreibt die eigentliche Lösung des Problems durch einen Workaround oder alternativ den Verlauf eines RFC.

Eingaben
Zur Beginn dieses Arbeitsschrittes sind Details zur Störung ebenso auf dem neuesten Stand wie das Ergebnis etwaiger gestarteter RFCs oder (provisorischer) Lösungen.

Vorgehensweise
Erst an dieser Stelle wird der eigentliche Fehler behoben bzw. ein RFC gestartet, damit der Fehler behoben werden kann.

Ergebnis
Am Ende sollte ein RFC stehen, damit der Fehler in Zukunft abgestellt ist. Die Störung sollte nun auch tatsächlich behoben und die Daten zur Störung auf dem neuesten Stand sein.

Abschließend beschreibt ITIL noch zwei weitere Aktivitäten, welche im Rahmen des Incident Managements durchgeführt werden sollten. Diese sind als “incident closure” und “incident ownership, monitoring and communication” benannt. Beide sind jedoch nur noch auf der Managementseite interessant, bieten sie doch keine weiteren Elemente mehr, welche bei der Entwicklung einer MIB behilflich sind.

4.2.3 Problemmanagement

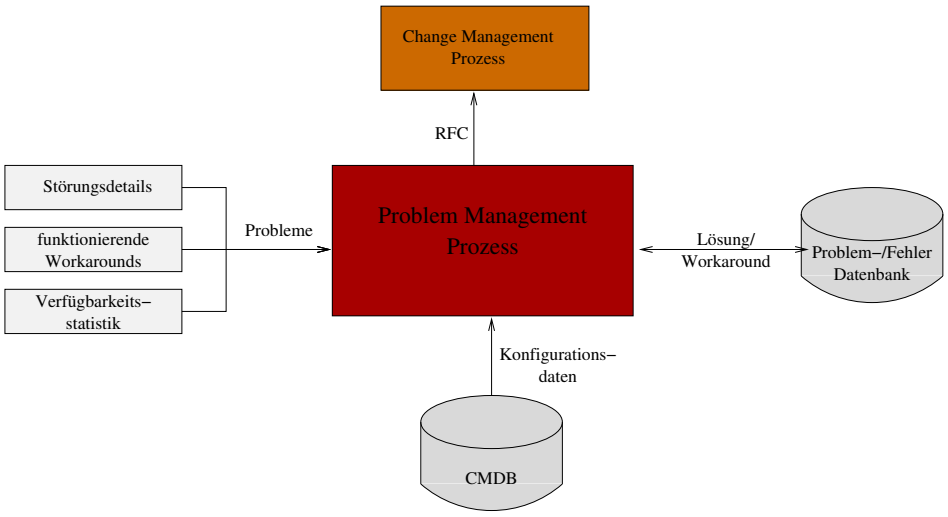

Abbildung 4.5: Ein-/Ausgaben des Problemmanagement Processes

Das Problemmanagement stellt die “zweite Stufe” zu Behandlung von Problemen dar und hat seine Hauptaufgabe, wie bereits erwähnt, im Anschluss an das Incident Management. Zusätzlich ist es aber auch Aufgabe des Problemmanagements die Problem-/Fehlerdatenbank zu pflegen, so dass möglichst viele Störungen und Probleme keine aufwändigen Suchen nach den Ursachen hervorrufen, sondern schnell und reproduzierbar beseitigt werden können.

Die Identifizierung eines Problems kann auf verschiedene Arten vor sich gehen. Und zwar durch:

- Analyse der Störungen, wenn sie auftreten (reaktives Problemmanagement).
- Analyse der Störungen über längere Zeiträume hinweg (proaktives Problemmanagement).
- Analyse der IT-Infrastruktur.
- Bereitstellung einer Wissensdatenbank.
- Entwickler/Verkäufer, wenn neue Produkte vorgestellt werden.

4.2.3.1 Problem control

Das Problem control gehört zum reaktiven Problemmanagement und hat die Aufgabe, die Ursache eines Fehlers zu identifizieren. Auf der anderen Seite hat es aber auch Empfehlungen für mögliche "work-arounds" zu liefern, die vom Incident Management verwendet werden können. Durch entsprechende Dokumentation der Fehlerursache wird des weiteren eine Möglichkeit aufgezeigt, die Wiederholung des Fehlers zu vermeiden.

Eingaben
Die bearbeitete Störung wird vom Incident Management übergeben. Als weitere Informationsquelle steht die CMDB zur Verfügung.

Vorgehensweise
Um die Aufgabe in geordneten Bahnen koordiniert ablaufen zu lassen, ist die Aufgabe dreigeteilt:

1. Problemidentifizierung und -aufzeichnung
   Diese findet statt, wenn
   - das Incident Management in der Phase des "initial support and classification" (siehe Kapitel 4.2.2.2) keine Übereinstimmung zu bekannten Problemen und Fehlern finden kann.
   - die Analyse der Störung nahelegt, dass die selbe Störung immer wieder auftritt.
   - die Analyse der Symptome der Störung zeigt, dass eine gleichartige Störung bisher noch nicht aufgetreten ist.
   - die Analyse der Infrastruktur ein Problem aufweist, welches eventuell zukünftige Störungen verursachen könnte.
   - eine bedeutende Störung auftritt, für die eine strukturelle Lösung gefunden werden muss.

   ITIL-xiv *Aufzeichnungen zum Problem*
   Hierzu ist der Bearbeiter auf "problem records" angewiesen. ITIL empfiehlt, die selben Templates zu gebrauchen, die auch für das Incident Management verwendet werden (siehe Tabelle 4.2). Bestimmte Elemente, die im Problemmanagement nicht mehr benötigt werden, sind dabei zu ignorieren. Dazu gehört zum Beispiel die Person, die die Störung gemeldet hat. Aufgrund der hohen Bedeutung dieser Daten wird empfohlen, sie in die CMDB aufzunehmen.

2. Problemklassifizierung

   ITIL-xv *Gruppierung des Problems*
   Nachdem das Problem identifiziert wurde, wird es zunächst in verwandte Gruppen oder Domains kategorisiert (z.B. Hardware, Software, Support Software, ...). Für die Priorisierung der Arbeit und der Ressourcen ist eine dem im Incident Management (siehe Seite 44) analoge Klassifizierung notwendig. Der zu erwartende Impakt wird aufgrund der anderen Zielvorgaben etwas anders gewichtet.

   ITIL-xvi *Komponentenabhängigkeiten*
   Für die Betrachtung der Auswirkungen ist es hilfreich, wenn aus der CMDB die Beziehungen der einzelnen Komponenten zueinander und vor allem zu der problembehafteten Komponente bestimmt werden können. Ist dies nicht der Fall, so müssen diese Daten jedes Mal wieder aufs Neue fehlerträchtig aus anderen Quellen gewonnen werden.

   ITIL-xvii *Kodierungsschema für die Auswirkung eines Problems*
   Für die organisationsweite Einteilung der Probleme ist es sinnvoll, ein Kodierungsschema zu entwickeln, mit dessen Hilfe die Auswirkung eines Problems schnell bestimmt werden kann.

   ITIL-xviii *Zeitpunkt der Bestimmung der Auswirkungen*
   Besonders zu beachten ist, dass die Analyse der Auswirkungen, auf Grund welcher die Ressourcen zur Lösung eines Problemes zugewiesen werden, nicht ein für alle mal feststehen, sondern sich je nach Rahmenbedingungen ändern können, also immer eine subjektive Komponente enthalten. So ist es möglich, dass ursprünglich nur eine Störungsmeldung eingegangen ist, aufgrund der Überlastung des Teams diese jedoch über einen weiteren Zeitraum nicht bearbeitet wurde. Mittlerweile sind jedoch 200 weitere Störungsmeldungen eingegangen, so dass festgestellt werden kann, dass die Störung einen größeren Kundenkreis betrifft.

3. Problemuntersuchung und -diagnose
   Diese erfolgt analog zu 4.2.2.3 im Incident Management, jedoch ist die Zielsetzung hier eine andere.

   Beim Incident Management ist die primäre Aufgabe die schnelle Wiederherstellung der Verfügbarkeit des Dienstes für den Kunden. Wenn dies nur mit Hilfe einer Notlösung möglich ist, so erfüllt dies dennoch die Aufgabe des Incident Managements.

   Beim Problemmanagement ist die Hauptaufgabe die Suche nach der zugrundeliegenden Ursache des Fehlers. Zusätzlich ist es jedoch auch die Aufgabe des Problemmanagements, gefundene Workarounds zur gemeldeten Störung in die Problem-/Fehler DB einzutragen. Dies soll bei einem erneuten Auftreten einer Störung die Bekämpfung der Störung beschleunigen.

   ITIL-xix *Prozedurale Fehler ebenfalls in die Datenbank*
   Aufgrund der Tatsache, dass Probleme oft nicht durch eine fehlerhafte Komponente, sondern durch prozedurale Fehler entstehen, sollten auch diese in der Datenbank vermerkt werden. Da derartige Fehler jedoch nicht an eine einzelne Komponente gebunden sind, empfiehlt ITIL hier besondere “Configuration Items” in die Datenbank

einzutragen, welche die fehlerhafte Prozedur und den etwaig vorhandenen zugehörigen RFC enthalten. Also soll die Datenbank in der Lage sein, auch nicht-technische Elemente aufzunehmen.

ITIL-xx *Dokumentation zur gesamten Infrastruktur*
Die Verfahren, welche während des Incident und Problem control Prozesses verwendet werden, verlangen die Verfügbarkeit der Dokumentation aller Produkte der IT-Infrastruktur. Dazu gehören alle beteiligten Applikationen (**ITIL-F**), Systemsoftware (**ITIL-G**), Utilities (**ITIL-H**), Netzwerkhard und -software (**ITIL-I**), sowie Konfigurations- und Netzwerkdiagramme (**ITIL-J**) zur Übersicht.

ITIL-xxi *Dokumentation vorgenommener Änderungen*
Ein weiterer wichtiger Punkt für die Problemidentifizierung ist die Dokumentation vorgenommener Änderungen. Oft treten Probleme erst dann in Erscheinung, wenn Änderungen vorgenommen wurden. Dies kann durch die Nebenwirkungen der Änderung selbst passieren, oder dadurch, dass das Problem bisher durch die geänderte Komponente ausgeglichen wurde.

Ergebnis
Nach Beendigung dieses Prozesses sollte bekannt sein, wodurch das Problem ausgelöst wurde, und nach Möglichkeit auch eine Lösungsmöglichkeit vorgeschlagen werden. Des weiteren ist es hier die Aufgabe des Problemmanagements, in der Problem- und Fehlerdatenbank einen aktuellen Workaround anzubieten. Im Falle von prozeduralen Fehlern bei einer Änderung im System sollten diese ebenso vermerkt werden wie physikalische Fehler in einer bestimmten Komponente.

Wenn sich herausgestellt hat, welche Komponente das Problem verursacht, so wird aus einem Problem ein “Known Error” und der Prozess des “error control systems” tritt in Kraft.

4.2.3.2 Error control

Die Aufgabe des “Error control” ist die Eliminierung eines “Known Error” mit Hilfe des Change Managements, sofern dies realisierbar ist und die Kosten zu vertreten sind. Des weiteren ist es für die Überwachung von Fehlern zuständig. Zusätzlich muss es sowohl ein Standbein im Test- als auch eines im Produktivbetrieb haben, da Fehler, welche im Testbetrieb auftauchen, evtl. auch im Produktivbetrieb zu korrigieren sind.

Eingabe
Zu Beginn dieses Prozesses ist die Ursache des Problems bestimmt und als “Known Error” dokumentiert worden. Ebenso geht aus der Dokumentation hervor, welches ein möglicher Workaround für dieses Problem darstellt.

Vorgang
Ebenso wie beim “problem control” beschrieben ist auch der “error-control” Prozess in verschiedene Abschnitte unterteilbar.

1. Identifikation und Meldung des Fehlers
   Da der Bereich des “error control” sowohl aus dem Produktivsystem als auch aus dem Entwicklungssystem aufgerufen werden kann, ist die Dokumentation ein wenig unterschiedlich zu gestalten.

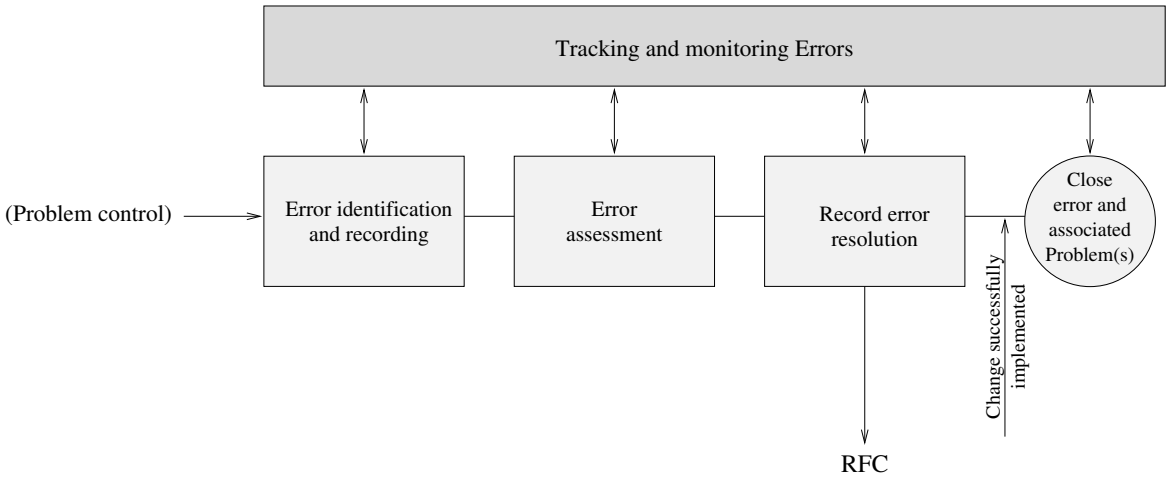

Abbildung 4.6: Phasen des Error Control Prozesses (analog zu [ITIL 00], Seite 106)

Im Falle des Aufrufes aus dem Produktivsystem ist die Grundlage der bereits als ITIL-xiv vermerkte Datensatz zur Beschreibung des Systemes. Tatsächlich wird hier meist nichts anderes getan, als das Problem zu einem "Known Error" umzudefinieren.
Im Falle des Aufrufes aus dem Entwicklungssystem heraus ist die Meldung erst einmal für das Produktivsystem nicht direkt relevant. Da aber Fehler oft auch nach der Produktivstellung wieder auftauchen können, ist es wichtig, die gesammelten Fehler zu den bestreffenden Produkten auch in die Datenbank für das Produktivsystem einzupflegen.

ITIL-xxii *Erzeugen eines "audit trails"*
Des weiteren ist dieser Bereich nicht nur für die direkte Verwendung beim Störungs- und Problemmanagement von Bedeutung, sondern muss auch "audit trails" für spätere Überprüfungen zulassen.

2. Fehlerabschätzung

   ITIL-xxiii *Kosten der Fehlerbehebung*
   Als nächstes ist es wichtig abzuschätzen, wie (kosten-/zeit-) aufwändig die Behebung des Fehlers voraussichtlich werden wird. Falls notwendig, ist es nun an der Zeit, einen RFC entsprechend den Vorgaben des Change Managements zu vervollständigen.

   Die eigentliche Behebung des Fehlers sowie das Testen der gewählten Lösung ist Aufgabe des Change Managements (siehe Kapitel 4.2.5)

3. Dokumentation der Fehlerbehebung
   Es ist natürlich wichtig, den Fehler möglichst genau zu beschreiben, wofür gemäß ITIL besonders die betroffenen Komponenten (**ITIL-K**), die Symptome, die Lösung oder die Umgehung des Problemes zu allen Fehlern in der Fehlerdatenbank zu speichern sind.

4. Abschließen des Fehlers
   Dazu gehört auch ein ordentlicher Abschluß der Fehlerbehandlung. Auch wenn dieser nur in einem Telephonanruf beim Kunden besteht, in welchem mitgeteilt wird, dass die Störung behoben ist. Bei schwerwiegenderen Fehlern ist es eventuell notwendig, eine genauere Untersuchung vorzunehmen.

5. Überwachung des Problem- und Fehlerbehebungsprozesses

ITIL-xxiv *Verbindung der Problembeschreibung zu vorgeschlagenen RFCs*
Da die tatsächliche Änderung im System nur durch das Changemanagement erfolgen kann, der Fehler aber vom Problemmanagement bearbeitet wird, ist es notwendig, dass in regelmäßigen Abständen eine Statusmeldung zum Fortschritt (**ITIL-L**) der Änderung erfolgt. Natürlich sollte auch die Auswirkung einer Störung beobachtet werden, da sich diese - wie bereits beschrieben - im Laufe der Zeit ändern kann.

ITIL-xxv *Anzahl erlaubter offener Probleme in dem SLA*
Des Weiteren gibt es SLA's , welche eine Maximalanzahl von ungelösten Problemen vorgeben. Auch dies muss bei der Behandlung der Fehler beachtet werden, um im Falle von kritischen Werten die Priorität der Bearbeitung der Fehler anpassen zu können.

Ergebnis
Das wünschenswerte Ziel bei diesem Prozess ist die Beseitigung aller bekannten Fehler durch Change Management Prozesse, welche hier gestartet werden. Dabei ist jedoch zu beachten, dass es manchmal nicht wünschenswert ist, einen Fehler zu beheben, da dies zu teuer oder technisch unmöglich ist. Auch könnten dadurch weitere Fehler erzeugt werden, die einen höheren Stellenwert haben.

Zu beachten ist auch, dass Hardwarefehler zumeist bereits im Incident Management (Kapitel 4.2.2) behoben werden können, im Falle einer Änderung an der Infrastruktur (zum Beispiel weil die Ersatzkomponente andere Eigenschaften hat als die ursprüngliche) über das Change Management erfolgen müssen.

4.2.3.3 Proactive problem management

Bisher wurden nur die Fehlerbehebungen betrachtet, die aktiv werden, nachdem eine Störung aufgetreten ist. Um die Verfügbarkeit der Dienste zu maximieren, ist es jedoch vorteilhaft, wenn Probleme bereits erkannt werden, bevor Störungen auftreten.
Das Wirkungsfeld des proaktiven Problemmanagements beginnt bei der Verteilung von Informationen an Benutzer, wie sie Probleme (zum Beispiel durch Entwickeln von sicheren und merkbaren Passwörtern) umgehen können und geht bis zur Indienststellung weiterer Dienste, um Dienste, die der Last nicht mehr gewachsen sind, zu entlasten. Um dies zu erreichen gibt es verschiedene Ansätze:

1. Trendanalyse
   Die Berichte aus dem Incident- und Problemmanagement bieten Informationen für vorausschauende Maßnahmen, um die Dienstgüte zu verbessern. Die Aufgabe ist hier. "schwache" Komponenten zu identifizieren und die Gründe für die "Schwäche" zu beheben. Dazu sind geeignete Berichte turnusmäßig zu erstellen.

   ITIL-xxvi *Ergebnisse von Änderungen*
   Dazu gehören zum Beispiel Trends, welche sich erst nach Änderungen zur Lösung bestimmter Problemtypen entwickeln.

ITIL-xxvii *Typen von Störungen*
Dazu gehören zum Beispiel auch beginnende Fehler eines bestimmten Types, welche sich durch bestimmte Störungen und Probleme ankündigen.

ITIL-xxviii *Sich wiederholende Störungen*
Andererseits ist es möglich, dass Fehler sich als nicht reproduzierbar erweisen, sondern nur sporadisch auftreten und dadurch nicht richtig behoben werden (können). Manchmal ist der Versuch der Reproduktion von anderen Faktoren abhängig, welche bei der normalen Problemsuche nicht beachtet worden sind. Ein beinahe legendäres Beispiel dieser Art wäre der sich wiederholende Serverausfall am späten Nachmittag, der von der Putzfrau verursacht wird, welche die Steckdose für den Staubsauger benötigt.
Zu guter Letzt kann es aber auch einfach nur notwendig sein, die Kunden besser zu schulen und/oder die Dokumentation zu verbessern.

ITIL-xxix *Problemkategorisierung*
Wenn Probleme bestimmten Kategorien zugeordnet werden, wird die Analyse von möglichen Problemen transparent. Hierdurch können - zusammen mit ein wenig kreativer Analyse - unter Umständen neue Zusammenhänge bei Problemen zutage gefördert werden.

2. Gezielte präventive Aktionen
Wenn bei der Trendanalyse Hinweise auf bestimmte Probleme vorgefunden werden, so müssen diese Probleme natürlich behandelt werden. Jedoch ist es oft so, dass die Ressourcen für die Behandlung von akuten Störungen und Fehlern knapp bemessen sind. Um besser einschätzen zu können, ob eine proaktive Behandlung einer möglicherweise zukünftig auftretenden Störung dem Aufwand angemessen ist, hat es sich als sinnvoll erwiesen, einen sogenannten **pain factor** einzuführen. Bei diesem Konzept wird jeder Störungskategorie ein **Schmerzwert** zugewiesen, der in die Berechnung des Ausmaßes mit einbezogen wird. Dieser wird durch verschiedene Faktoren beeinflusst, wie zum Beispiel:

ITIL-xxx
dem Umfang der Störungen

ITIL-xxxi
der Anzahl der betroffenen Kunden

ITIL-xxxii
die Dauer und die Kosten für die Behebung der Störung

ITIL-xxxiii
die Kosten der Störung für den Betrieb.

4.2.4 Konfigurationsmanagement

Konfigurationsmanagement ist eines der zentralen Bereiche bei der Verwaltung der IT-Landschaft. Hier ist die Inventarisierung aller Assets in der jeweiligen Organisation ebenso angesiedelt, wie die verschiedenen Konfigurationen. Da dieser Bereich ebenso die Grundlagen für das Incident- und Problemmanagement, wie für das Change- und das Release Management bieten muss, ist es besonders wichtig, dass die hier verwalteten Informationen aktuell und korrekt sind. Die Einbettung des Konfigurationsmanagement in das IT-Management ist am besten in Tabelle 4.2 zu erkennen.

• Release contents, including component CIs and their version numbers
• component CIs and their version numbers in the test and live environments
• CIs affected by a scheduled (authorised) Change
• all Requests for Change (RFCs) relating to one particular CI
• CIs purchased from a particular supplier within a specific period
• CI history
• equipment and software at a given location, for example to assist an audit
• CIs that are scheduled to be upgraded, replaced or decommissioned
• Changes and Problem records associated with a CI
• all CIs affected by a problem

Tabelle 4.3: Beispiele für den Nutzen einer Konfigurationsdatenbank ([ITIL 00], Seite 124)

ITIL nennt eine Reihe von Aufgaben des Konfigurationsmanagements, darunter (nach [ITIL 00], Seite 121):

ITIL-xxxiv *Identifizierung und Bestimmung des Eigentümers*
Auswahl und Identifizierung der Eigenschaften aller "Configuration Items" (CI) inklusive ihrer "Eigentümer", Beziehungen untereinander und der dazugehörigen Dokumentation.

ITIL-xxxv *Nur autorisierte Komponenten erlaubt*
Sicherstellen, dass nur autorisierte Komponenten zum Einsatz kommen, inklusive der Erkennung nicht autorisierter Elemente. Dementsprechend muss auch ein Änderung an einer Komponente durch das Change Management gehen und in der CMDB vermerkt werden.

ITIL-xxxvi *Geschichte der Stati*
Führen einer Buchhaltung über die verschiedenen Stati (zum Beispiel "aktiv", "außer Betrieb", "in Wartung", ...), welche eine Komponente im Laufe ihres "Lebens" inne hat. Natürlich auch die Möglichkeit der Abfrage des aktuellen Status.

ITIL-xxxvii *Regelmässige Überprüfung der CMDB*
Kontrolle der oben angegebenen Punkte. So soll unter anderem sichergestellt werden können, dass alle vermerkten Komponenten auch noch vorhanden und die Informationen darüber auch noch auf dem aktuellsten Stand sind.

ITIL gibt eine Liste von Informationen vor, bei denen die Ablage in einer Datenbank (CMDB) im Bereich des Konfigurationsmanagement nützlich ist (Tabelle 4.3).

4.2.4.1 Configuration Management Planing

Das Configuration Management Planing ist aus der Sicht von ITIL die Grundlage des Konfigurationsmanagement und umschreibt noch einmal dessen Basisaufgaben. Interessant ist im Rahmen dieser Arbeit aber nur ein Teil, wie zum Beispiel die Existenz einer Namenskonvention der verwalteten Komponenten (**ITIL-M**).

ITIL-xxxviii *Schnittstellen zu anderen Service Management Systemen*
Es ist für ein funktionierendes Service Management System vonnöten, Schnittstellen zu anderen Service Management Systemen vorzuhalten, um einen reibungslosen Austausch von Informationen zu gewährleisten.

ITIL-xxxix *Rollen und Verantwortungsbereiche*
Innerhalb einer größeren Organisation ist es wichtig, dass die Rollen und Verantwortungsbereiche für die einzelnen Dienste innerhalb der Organisation definiert und einsehbar sind.

4.2.4.2 Configuration identification

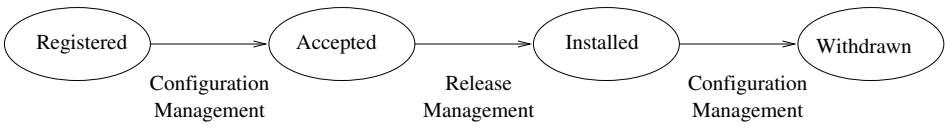

Abbildung 4.7: Beispiel für den Release Life-cycle einer Anwendung (aus [ITIL 00], Seite 141)

ITIL-xl *Aufteilung in kleinere Komponenten*
Für die eigentliche Verwaltung der Komponenten ist es sinnvoll, wenn die Gesamtinfrastruktur in kleinere Einheiten zerlegt wird, die dann als einzelne Komponenten verwaltet werden können.

ITIL-xli *Eltern-Kind Beziehungen*
Dafür sollten die Beziehungen zwischen verschiedenen Komponenten in Form einer Eltern-Kind-Relation ausgedrückt werden können, wofür sich natürlich ein objektorientiertes Modell empfiehlt.

ITIL-xlii *Einbinden von Organisationen*
Zudem sollten andere Organisationen als eine eigene zu verwaltende Komponente angesehen werden können. Auch wenn die Einflussmöglichkeiten hier natürlich wesentlich eingeschränkt sind, kann dadurch die Aus- und Einwirkung auf anderer Organisationen modelliert werden.

ITIL-xliii *Klassifizierung der Komponenten*
Um die Übersicht über die verschiedenen Komponenten zu behalten, hat es sich als sinnvoll erwiesen, sie in Gruppen einzugliedern. Typisch hierfür sind zum Beispiel “Software Projekt”, “Business System”, “Betriebssystem”, “Server”, “Mainfraime”, “Workstation”, “Laptop”, “Router”, “Hubs”, etc.

ITIL-xliv *Einbinden des Lebenszyklus*
Die Verwaltung soll den gesamten Lebenszyklus der Komponente im Blick behalten (als Beispiel für einen Lebenszyklus siehe Bild 4.7) und nicht nur einen begrenzten Teil, zum Beispiel die mit “Installed” markierte Produktivphase.

ITIL-xlv *Beziehungen*
Es ist nicht ausreichend, nur die ’Vererbungshierarchie’ der Komponenten darstellen zu können, vielmehr müssen alle Beziehungen einer Komponente zu den anderen Komponenten abgebildet werden können. Dazu gehören maßgeblich:

- wie in ITIL-xli genannt, die Teil-Beziehungen (**ITIL-N**)
- die physischen Verbindungen (**ITIL-O**) zu anderen Komponenten
- ob eine Komponente eine andere für ihren Betrieb verwendet (**ITIL-P**)
- Verweise auf RFCs (**ITIL-Q**), Inicident records (**ITIL-R**), Problem records (**ITIL-S**) und bekannte Fehler (**ITIL-T**).

ITIL-xlvi *Partiell vorhandene Informationen*
Gerade bei derart vielen Informationen ist es nicht immer sofort möglich, alle Details zu einer Komponente aufzuführen. Auch wird es gerade am Anfang des Lebenszyklus einer Komponente einige Informationen, wie zum Beispiel Störungsreports, noch gar nicht geben.

ITIL-xlvii *Standardkomponenten*
Andererseits gibt es natürlich eine Menge Informationen, die gerade bei größeren Installationen höchst redundant in verschiedenen Komponenten vorgefunden werden. Um zu verhindern, dass sich hier Fehler einschleichen, ist anzuraten, "Standardkomponenten" zu erstellen, bei welchen bereits alle komponentenübergreifend gleichen Attribute belegt sind und als Template für die Einstellung aller gleichartigen Komponenten fungieren.

ITIL-xlviii *Komponenten-/Dienstpakete*
Eine andere Art von Beziehungen sind die, welche einen vollständigen Systemzustand beschreiben können. So ist anzudenken, den Zustand einer Menge von Diensten inklusive dazugehöriger Dokumentation als eine einzige Komponente anzusehen. Beispielhaft wäre deren Nutzen anzuführen bei Darstellung

- eines stabilen Ausgangszustandes für ein eventuelles Rollback bei der Vorbereitung und Durchführung von Changes.
- eines Softwarestandes, der für Verteilung auf entfernte Standorte vorgesehen ist.
- eines angestrebten zukünftigen Softwarestandes.
- der Anforderungen für ein Update der Hardware.
- der Anforderungen für ein Update der Software.

ITIL-xlix *Konventionen*
Wichtig für eine durchgehende Verständlichkeit der benutzten Bezeichnungen ist die Verwendung von eindeutigen Namenskonventionen. Dabei sollten die Namen möglichst kurz und prägnant sein. Vorzuschlagen wäre die Einhaltung international angesehener Konventionen.

ITIL-l *Versionsnummern*
Anzuraten ist in diesem Zusammenhang, mit Versionnummern, statt mit sprechenden Namen wie "live", "next", "old" zu arbeiten. Gerade bei größeren Organisationen hat es sich gezeigt, dass oft mehrere Versionen gleichzeitig "live" - also aktiv - sind. Ein einfaches Beispiel hierfür wäre zum Beispiel eine Organisation mit Nebenstellen, bei der die "roll-outs" zunächst in der Zentrale durchgeführt werden und dann nach und nach in den Außenstellen.

ITIL-li *Eindeutige Identifikatoren*
Um Komponenten jederzeit eindeutig zuordnen zu können, sollte an jeder eine eindeutige Identifikationsnummer angebracht sein. Bei einer physikalischen Komponente in Form einer nicht ablösbaren Markierung, bei nicht physischen zum Beispiel in Form eines nicht änderbaren Attributes.

4.2.4.3 Control of CIs

Die Aufgabe dieses Bereiches liegt darin, dafür zu sorgen, dass nur autorisierte und auch identifizierbare Komponenten in der Organisation eintreffen und in der CMDB vermerkt werden. Dies hat natürlich auch Auswirkungen auf das Lizenzmanagement.

ITIL-lii *Möglichst frühzeitiger Beginn der Dokumentation*
Komponenten müssen bereits dann registriert werden, wenn ihre (An-)Schaffung in Auftrag gegeben wird. Um den aktuellen Stand der Lieferung verfolgen zu können, sollte auch der Lieferstatus (**ITIL-U**) vermerkt werden.

ITIL-liii *Benennung von Verantwortlichen*
Des weiteren ist es wichtig, feststellen zu können, wem die Komponente im Augenblick zugeordnet ist. Wer also die Verantwortung über diese Komponente übernimmt.

ITIL-liv *Aktueller Abschnitt im Lebenszyklus*
Nachdem eine Komponente sich innerhalb der Organisation befindet (im Falle einer gekauften Komponente), aber auch wenn sie sich in der Entwicklung befindet ist es wichtig, über den aktuellen Stand ihres Lebenszyklus (development, test, live, archive, ...) Buch zu führen.

ITIL-lv *Letzte Änderung im Lebenszyklus*
Auch die letzte Änderung des Abschnittes innerhalb des Lebenszyklus, in welchem sich besagte Komponente befindet, ist wichtig.

ITIL-lvi *Jeweils aktuelle Dokumentation zur Komponente*
Da sich mit der Entwicklung einer Komponente auch die zugehörige Dokumentation ändert, ist eine Referenz auf die zur aktuell betrachteten Version zugehörige Dokumentation unabdingbar.

ITIL-lvii *Lizenzen*
Da sich heutzutage das Besitz- und Nutzungsrecht als eines der wichtigsten Elemente im täglichen Geschäft etabliert hat, ist es unabdingbar, die zu einer Komponente gehörigen Lizenzbedingungen und Lizenzen möglichst schnell verfügbar zu haben.

ITIL-lviii *Lizenzüberwachung*
Die Lizenzkontrolle soll möglichst automatisiert durch das Managementsystem übernommen werden. Eine Überprüfung per Hand ist zu aufwändig und fehleranfällig.

ITIL-lix *Updaten der Dokumentation bei Störungen, Problemen, Änderungen, ...*
Auch das Aktualisieren der bei ITIL-xlv angesprochenen Verweise auf Störungsmeldungen, Problemberichten, Fehlermeldungen, Änderungsanfragen, Änderungsdokumentationen und Dokumentationen zur Freigabe einer neuen Version gehören zu den Aufgaben des Configuration Control Prozesses.

ITIL-lx *Archivierung jeglicher Dokumentation*
Nach der produktiven Phase ist die Dokumentation nicht plötzlich unnütz geworden, sondern muss für zukünftige Einsichtnahmen in einem Archiv abgelegt werden. Sie wird zum Beispiel bei der Suche nach Fehlern eines Nachfolgeproduktes oder als Teil des jährlichen Audits benötigt.

ITIL-lxi *Integritätsüberwachung der Änderungen*
Dabei ist besonderes Augenmerk darauf zu legen, dass die Integrität der CMDB gewährleistet bleibt, wenn Daten eingefügt oder verändert werden.

ITIL-lxii *Integrität der CMDB*
Die Integrität sollte dabei nicht nur bei der Änderung der Daten überprüft werden sondern auch durch einen regelmäßigen Check auf die Existenz und Korrektheit der notwendigen Daten. Dies kann durch eine händische Suche oder automatische Checks auf die “Anwesenheit” und/oder “Ansprechbarkeit” von Komponenten geschehen.

ITIL-lxiii *Disaster Recovery*
Das Ziel ist es, mit Hilfe dieser Daten im Rahmen eines Disaster Recovery wieder eine produktive Umgebung herstellen zu können.

ITIL-lxiv *Zugangskontrolle*
Um sicherzustellen, dass System-Einstellungen nicht unbefugt geändert werden, soll mit Hilfe von Zugangskontrollsystemen verhindert werden, dass Unbefugte Änderungen an der CMDB, Hardware, Software und/oder Dokumentation vornehmen können.

ITIL-lxv *Update nach Kontrollen*
Zu guter Letzt gehört zu der Kontrolle auch die Aktualisierung der CMDB bei der Entdeckung nicht oder falsch dokumentierter Hardware. Dies sollte um ITIL-lxiv Genüge zu leisten, je nach Aufgabe und Berechtigung des "Entdeckers" auf verschiedenen Wegen geschehen:

- über eine Mitteilung an den Service Desk
- über ein Update der Dokumentation zur betreffenden Komponente
- durch die Markierung des Datensatzes als "falsch" aufgezeichnet
- durch die Meldung einer Störung, um eine Untersuchung einzuleiten
- durch die Einreichung eines RFC für die Korrektur der CMDB

Im Falle einer unregistrierten Komponente ist es wichtig, deren Ursprung und den Weg zu bestimmen, über welchen sie in die Organisation gekommen ist.

4.2.4.4 Configuration status accounting

Um einen Überblick über die verwaltete Umgebung zu behalten, sollte regelmäßig ein Statusreport über alle Komponenten erzeugt werden, der ihre aktuelle Version und ihre Historie enthält.

ITIL-lxvi *Eindeutige Identifizierung*
Um zu wissen, über welche Komponente gerade gesprochen wird, ist es angebracht, einen eindeutigen Identifikator, wie in ITIL-li bereits verwendet, zu benutzen. In Verbindung mit der Versionsnummer der aktuellen Konfiguration lässt sich der betreffende Datensatz leicht auffinden.

ITIL-lxvii *Aktueller Status*
Genau so wichtig wie die Identität der Komponente ist für das Dienstmanagement auch deren aktueller Status.

ITIL-lxviii *Verantwortlicher zur letzten Änderung*
Um Probleme nachvollziehen zu können, ist es von Vorteil, wenn die Person mit aufgeführt wird, welche die letzten Änderungen durchgeführt hat.

ITIL-lxix *Audit trail*
Auch andere Daten, die für einen "Audit Trail" benötigt werden, sind mit zur Verfügung zu stellen.

ITIL-lxx *Offene Probleme und RFCs*
Noch offene Probleme und RFCs, die zu einer Komponente vermerkt sind, verdienen besondere Aufmerksamkeit.

4.2.4.5 Configuration verification and audit

Vor größeren Änderungen sollte immer eine Überprüfung der Umgebung erfolgen.

ITIL-lxxi *Kontrolle auf unregistrierte Komponenten*
Dazu gehört auf jeden Fall ein Check dahingehend, ob nur die vermerkten Komponenten real existieren. Sollten andere Komponenten entdeckt werden, so sind diese, wie in ITIL-lxv beschrieben, in das Managementsystem einzubringen.

ITIL-lxxii *Absegnung durch Change Manager*
Auch sollte überprüft werden, dass alle Änderungen vom Change Management ordentlich abgesegnet wurden und nicht etwa kleinere Workarounds undokumentiert und ungetestet ihren Weg in die endgültige Konfiguration gefunden haben.

Die Zeitpunkte, bei denen der Audit vorzunehmen ist, sind:

- kurz nach Indienststellung eines neuen Konfigurationsmanagement Systems
- vor und nach größeren Änderungen an der IT Infrastruktur
- vor einem Release oder einer Installation
- nach einem Disaster Recovery und nach dem 'return to normal'
- in zufälligen Intervallen
- nach Entdeckung einer nicht autorisierten Komponente
- innerhalb eines regelmäßigen Überprüfungsintervalles

4.2.4.6 CMDB back-ups and housekeeping

ITIL-lxxiii *Backups der CMDB*
Um dem Ziel von ITIL-lxiii nachzukommen, sollten regelmäßige Kopien der CMDB an einen entfernten Ort ausgelagert werden ("off-site storage"), um auch dann noch darauf zurückgreifen zu können, wenn die Hauptdaten nicht mehr erreichbar sind.

In Anhang B befindet sich ein Vorschlag von ITIL für Attribute, welche in einer CMDB vorhanden sein sollten.

4.2.5 Änderungsmanagement

"Stillstand ist der Tod." - Dieses Sprichwort gilt auch in der IT. Die IT-Landschaft einer Organisation verändert sich ständig - muss sich ständig verändern. Dass dies in geordneten Bahnen abläuft, ist die Aufgabe des Change-Managements. Das Change Management laut ITIL beschäftigt sich nicht mit sogenannten "service request(s)" (darunter werden einfache Änderungen, wie ein neues Passwort, oder das Ersetzen eines defekten Arbeitsplatzcomputers verstanden), sondern um den "process of moving from one defined state to another" ([ITIL 00], Seite 168) (Vorgang des Überganges von einem definierten Zustand in einen anderen). Es handelt sich bei den durch das Change Management verwalteten Vorgängen also immer auch um Projekte mit größerer Auswirkung. ITIL bietet in [ITIL 00] eine Reihe von Punkten an, welche durch ein Formular abgefragt werden sollten (siehe Anhang C).

ITIL-lxxiv *Änderungsdokumentation*
Da es sich um ein bleibendes Dokument handelt, welches evtl. auch über Organisationsgrenzen hinweg bearbeitet wird, ist eine eindeutige Zuordnung der zu ändernden Komponenten wichtig.

ITIL-lxxv *Arbeitsgruppe für Änderung*
Für Anfragen, die einen größeren Personenkreis betreffen, ist es wichtig, die an der Bearbeitung beteiligten Personen zu nennen.

ITIL-lxxvi *Versionierung der Komponenten*
Um Missverständnissen aufgrund falscher Datenlage entgegenzutreten, ist die Version der zu ändernden Komponente ein wichtiges Hilfsmittel.

ITIL-lxxvii *Zeitpunkt für Änderung*
Des weiteren ist für die Einschätzung einer Änderung auch der (vorgesehene) Zeitpunkt der Änderung von Bedeutung, um betroffene Benutzer vorzuwarnen oder auch, um damit zusammenhängende Änderungen rechtzeitig vorzunehmen.

ITIL-lxxviii *Priorität der Änderung*
Gerade in großen Organisationen mit (üblicherweise) begrenzten Ressourcen, wird im Konfliktfalle anhand von Prioritäten entschieden.

ITIL-lxxix *Betroffene Komponenten*
Da selten eine Änderung nur eine einzelne Komponente betrifft, ist in Betracht zu ziehen, welche anderen Komponenten durch die Änderung betroffen sein werden.

ITIL gibt ein Anforderungsprofil für eine Software zur Verwaltung des Change Managements vor (siehe Anhang D). Um ein solches Tool ordentlich einbinden zu können, sollten Schnittstellen vorhanden sein, die die Beantwortung einiger hier aufgestellter Fragen erlauben.

ITIL-lxxx *Dienstabhängigkeiten*
Dazu gehören primär - wie es sich nicht nur durch ITIL, sondern durch das gesamte moderne Dienstmanagement zieht - die Abhängigkeiten der Dienste und ihrer Komponenten untereinander.
Besonders wünschenswert ist, dass die Anzeige der betroffenen Komponenten automatisch durch das System erfolgen kann und nicht händisch vorgenommen werden muss.

ITIL-lxxxi *Historie der Probleme bei Änderungen*
Oft haben Änderungen Probleme nicht nur als Ursache sondern zur Folge. Deswegen ist es ratsam, eine Historie der Probleme in der Datenbank vorzuhalten.

ITIL-lxxxii *Historie der Änderungen*
Die Änderungshistorie ist eine zwingende Vorraussetzung bzw. ein zwingender Bestandteil für ein Changemanagement-Tool. Aus dieser wird zum Beispiel der audit-trail zu den Änderungen gewonnen.

ITIL-lxxxiii *Komponentenbetreuer*
Für Änderungen an Komponenten muss es einen Verantwortlichen geben. Deswegen sollte diese Person auch immer bei den jeweiligen Änderungsdokumenten direkt vermerkt werden.

ITIL-lxxxiv *Zeitpunkt der Aktivierung*
Der Zeitpunkt der Änderung einer Komponente und die Produktivsetzung (Aktivierung) der Änderung sind in der Regel verschieden.

ITIL-lxxxv *Zukünftige Systemparamter*
Was bringt das Datum einer Änderung, wenn das System keinen Vermerk darüber hat, welche zukünftigen Parameter gelten sollen?

ITIL-lxxxvi *Funktionierende Version*
Im Falle von Fehlern bei Änderungen der Konfiguration des Dienstes sollten die Änderungen möglichst gefahrlos rückgängig gemacht werden können, ohne neue Fehler zu produzieren. Um dies möglichst einfach erreichen zu können, ist vor Beginn der Änderungen eine funktionierende Version der Einstellungen zu sichern, um im Falle von Problemen auf diese zurückgreifen zu können.

ITIL-lxxxvii *Workflow*
Da bei Änderungen oft verschiedene Dienste bzw. Komponenten betroffen sind, ist der Vorgang von der ersten Version eines RFC über die Absegnung des finalen RFC bis hin zur tatsächlichen Ausführung ein weiter Weg. Um dabei eine gleichbleibende, korrekte Vorgehensweise sicherzustellen, bietet es sich an, den ganzen Prozess in einen Workflow einzubinden, der dann idealerweise auch alte Revisionen zur Verfügung stellen kann.

ITIL-lxxxviii *Änderungsreihenfolge*
Bei vernetzten Diensten ist oft eine bestimmte Reihenfolge bei der Änderung einzuhalten. So kann manche Änderung erst dann durchgeführt werden, wenn die Änderung bei einer anderen Komponente bereits durchgeführt worden ist oder eine Änderung muss erst teilweise durchgeführt werden, bevor die nächste durchgeführt werden kann. Und erst wenn diese wiederum abgeschlossen ist, kann die erste vollständig abgeschlossen werden. Als Beispiel sei hier der Umzug eines zentralen Managementservers zu sehen, der besondere Berechtigungen auf den zu managenden Diensten hat. Zunächst einmal muss den gemanagten Diensten die neue Adresse des Managementservers bekannt gemacht werden, dann erst kann der Server umziehen und wenn dieser Umzug abgeschlossen ist, muss auf den gemanagten Diensten die alte Adresse des Managementservers wieder entfernt werden.

4.3 Managementaufgaben nach OSI

Der Aufteilung der verschiedenen Sichtweisen unter ITIL entspricht im OSI Management Framework dem Zweig der Aufteilung nach funktionalen Bereichen (siehe Grafik 3.6 auf Seite 27). Die Zusammenarbeit dieser wird in Abbildung 4.8 dargestellt.

Die fünf Bereiche werden oft zusammengefasst als **FCAPS** bezeichnet. Im einzelnen stehen sie jeweils für einen funktionalen Bereich des Managements und seine Anforderungen.

4.3.1 Fault Management

Probleme in diesem Bereich sind für den User die auffälligsten. Die Aufgabe dieses Bereiches ist die Erkennung, Lokalisierung und Behebung von anormalem Systemverhalten. Damit entspricht es den Bereichen des Incident- und Problemmanagement aus ITIL. Das Hauptaugenmerk liegt jedoch auf der Behebung von Problemen, welche bereits eingetreten sind. Das proaktive Fehlermanagement wird hier nicht gesondert beachtet. Allerdings gehört zu diesem Ansatz explizit auch das selbständige Erkennen von Fehlern und damit das Überwachen der einzelnen Komponenten auf Anormalitäten.

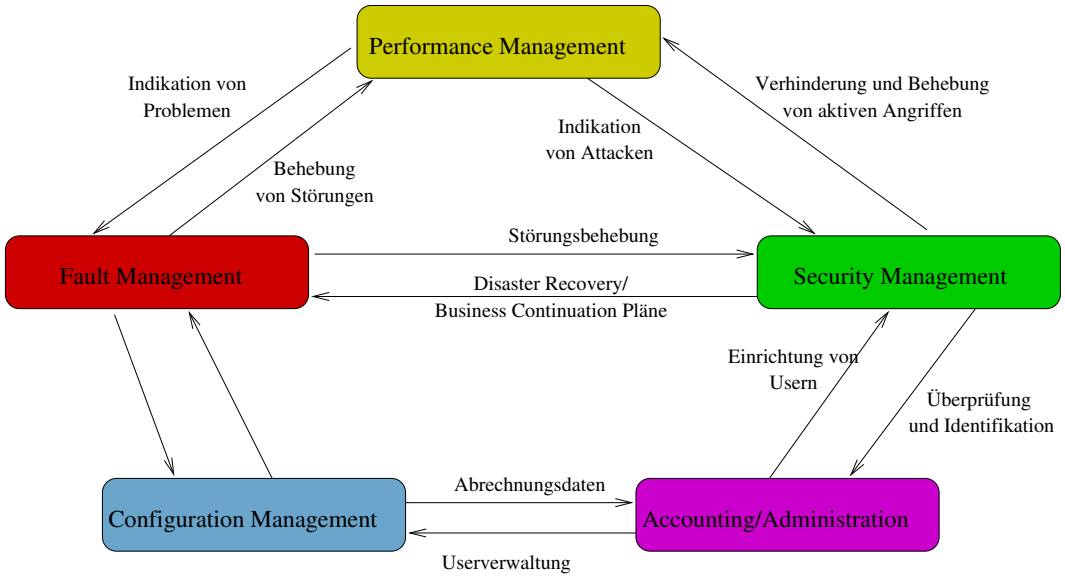

Abbildung 4.8: Betrachtete Zusammenarbeit der funktionalen Bereiche bei OSI

[HAN 99], Kapitel 3.2.2 gibt eine Übersicht über die verschiedenen Aufgaben, welche im Bereich des **Fault Management** bearbeitet werden sollten. Um diese Aufgaben möglichst effizient bearbeiten zu können, sollte eine MIB bei der Beantwortung der nachfolgenden Punkte behilflich sein:

OSI-i *Identifikation der Komponente*
Um ein Problem einer Komponente zuordnen zu können, ist es notwendig, den Dienst zu identifizieren. Diese Identifikation sollte nach Möglichkeit so geschehen, dass sie eindeutig ist.

OSI-ii *Status des Dienstes*
Zunächst einmal ist es für die Behandlung eines Fehlers wichtig zu wissen, in welchem Status (“aktiv”, “in Wartung”, “defekt”, ...) sich der betreffende Dienst bzw. seine Komponente(n) befindet bzw. befinden. Anhand dessen kann bestimmt werden, wie dringend eine Behandlung des Problemes ist, aber auch ob die Störung der Komponente, die zu einem Dienstausfall geführt hat, tasächlich als Störung anzusehen ist.

OSI-iii *Erreichbarkeit des Dienstes*
Danach ist es wichtig, zu wissen, ob ein Dienst überhaupt erreichbar ist. Sollte der Dienst selbst erreichbar sein, nicht jedoch ein benötigter (Sub-)Dienst, so funktioniert natürlich der Dienst selber nicht richtig.

OSI-iv *Alarme als Meldungen von Störungen an des Managementsystem*
Um Fehler zu finden oder überhaupt erst auf Fehler aufmerksam gemacht zu werden, gibt es Systeme, welche im Falle von abnormem Systemverhalten Alarmmeldungen abgeben. Es ist notwendig, dass diese im System verarbeitet werden können. Es werden auch Meldungen von Benutzern eines Systemes der Einfachheit halber als Alarme bezeichnet.

OSI-v *Quelle des Alarms*
Es ist nicht nur wichtig, zu wissen, dass überhaupt eine Alarmmeldung aufgetreten ist, sondern auch, wo der Alarm ursprünglich herkam und welche Komponente den Alarm ausgelöst hat.

OSI-vi *Bestätigung des Alarms*
Um das Auflaufen einer übermäßig hohen Anzahl von Alarmen, die lediglich zu einer Verringerung der Übersichtlichkeit führen, zu verhindern, gibt es in jedem System die Möglichkeit zu bestätigen, dass ein Alarm zur Kenntnis genommen wurde.

OSI-vii *Grund des Alarms*
Auch ist es wichtig zu wissen, welches der eigentliche Grund für den Alarm ist.

OSI-viii *Fehlerhafte Komponente*
Hat sich herausgestellt, wodurch der Alarm ausgelöst wurde, sollte man als nächstes herausfinden, welche Komponente die Ursache des Problems beinhaltet bzw. welche Komponenten nicht richtig zusammenarbeiten und somit das Problem verursachen.

OSI-ix *Verantwortlicher für die Komponente*
Für die weitere Bearbeitung des Problems ist neben der Kenntnis der Komponente auch die für die Komponente verantwortliche Person wichtig.

OSI-x *Fehlerdatenbank*
Um bei Alarmen und auch allgemeinen Problemen nicht jedesmal wieder aufs Neue dieselben (Fehl-) Schlüsse zu ziehen, bietet es sich an, eine **Fehlerdatenbank** aufzusetzen. Diese ermöglicht es, aufgrund der vorhandenen Symptome auf bereits bekannte Ursachen und deren Behebung zugreifen zu können. Um diese Datenbank möglichst effektiv zu gestalten, sollte sie auch die folgenden Punkte beinhalten:

OSI-xi *Behebung des Alarms*
Oft treten Alarmmeldungen nicht zum ersten Mal auf, sondern wiederholen sich auf die eine oder andere Weise. Daher ist es immer auch im Interesse einer schnellen Behebung eines Problemes zu wissen, wie die Ursache für den Alarm früher behoben wurde. Dies kann unter Umständen auch Hinweise auf die Komponente geben, welche ursächlich für das Auslösen des Alarms zeichnet.

OSI-xii *Zeitweilige Ersatzmöglichkeiten für den gestörten Dienst*
Für den Benutzer eines Dienstes ist es oft nicht interessant, wodurch ein Problem ausgelöst wurde. Er möchte vielmehr möglichst schnell den Dienst wieder verwenden können. Zum Teil ist er auch bereit, für eine möglichst schnelle Wiederherstellung der Verfügbarkeit gewisse Einschränkungen in Kauf zu nehmen. Zumeist existiert für Probleme in der Infrastruktur eine Umgehungsmöglichkeit, welche das Problem zwar nicht behebt, es dem Nutzer jedoch ermöglicht, weiterzuarbeiten.

OSI-xiii *Problemlösungen*
Solch eine Ersatzlösung behebt jedoch nicht die Ursache des Problems. Um einen reibungslosen Betrieb entsprechend der ursprünglich vereinbarten Bedingungen sicherzustellen, muss eine endgültige Lösung zur Behebung des Problemes und zur Wiederherstellung des alten Zustandes gefunden werden.

OSI-xiv *Ideale Problemlösung*
Sollte sich herausstellen, dass es mehrere Möglichkeiten zur Problemlösung gibt, so ist die effektivste zu ermitteln.

Jedoch können die Komponenten oft auch selber einiges zum Fehlermanagement beitragen:

OSI-xv *(eindeutiges) Identifikationsmerkmal der Komponente*
Zunächst bietet sich die Identifikation einer Komponente an. Durch den Vergleich der erwarteten Identität einer Komponente mit der tatsächlichen können nicht nur Verwechslungen vermieden, sondern auch falsch aufgestellte Komponenten entdeckt werden.

OSI-xvi *Schnittstellen*
Eine Auskunft über verfügbare Schnittstellen bietet die Möglichkeit, herauszufinden, ob eine Komponente an der vorgesehenen Stelle ihre Aufgaben erfüllt und wie sie durch andere Komponenten angesprochen werden kann.

OSI-xvii *Soll-Parameter der Schnittstelle*
Natürlich ist es für jede Schnittstelle notwendig, zu definieren, innerhalb welcher Parameter diese arbeiten soll. Die hier eingestellten Werte sollten im Normalfall mit den in der Managementsoftware hinterlegten übereinstimmen.

OSI-xviii *Überprüfung der Schnittstelle*
Im Falle einer Störung ist zunächst einmal zu überprüfen, ob die Schnittstellen innerhalb der vorgegebenen Parameter arbeiten.

OSI-xix *Liste der Parameter*
Auch eine Sammlung aller anderen Parameter, welche diesen Dienst beschreiben, sind bei der Suche nach der Ursache für ein Problem von Nutzen.

OSI-xx *Soll-Werte der Parameter*
Wie bei den Schnittstellen sind die aktuellen Sollwerte der Parameter von Bedeutung.

OSI-xxi *Überprüfung der Parameter*
Parallel zu den Schnittstellen, ist es auch wünschenswert, direkt überprüfen zu können, ob die Parameterwerte eingehalten werden.

OSI-xxii *Setzen der Soll-Werte*
Zu guter Letzt ist es für die Behebung von Fehlern, die durch falsche Parameterwerte entstanden sind, notwendig, diese neu setzen zu können.

OSI-xxiii *Ändern des Status*
Manchmal muss man den Status eines Dienstes insgesamt ändern können, sei es um Puffer neu zu initialisieren oder einem Dienst andere Eigenschaften zuzuordnen. Eine Funktionalität zum Starten eines Resets, dem Neueinlesen der Konfiguration (zu einem bestimmten Zeitpunkt/sofort) ist daher notwendig.

OSI-xxiv *Setzten der Schwellwerte für Alarme*
Die Schwellwerte für das Auslösen eines Alarms sollten in den meisten Fällen variabel sein, um an die jeweilige Umgebung angepasst werden zu können.

OSI-xxv *Automatische Alarmmeldungen*
Um möglichst schnell auf Probleme hingewiesen zu werden, ist eine selbsttätige Mitteilung bestimmter Meldungen an Trouble Ticket Systeme und/oder den Help Desk eine effektive Möglichkeit. Durch die Automation des Verfahrens und die Meldung von Anormalitäten durch die betroffenen Dienste selber ist oft eine Meldung von (potentiellen) Problemen möglich, bevor die Benutzer dieses Dienstes davon etwas mitbekommen.

Andere Informationen wiederum können nicht von den Komponenten selber geliefert werden, sondern müssen aus der Managementumgebung heraus bestimmt werden.

OSI-xxvi *Weg zum benötigten Dienst*
Dazu gehört zum Beispiel der "Weg", um von einem abhängigen Dienst zum Anbieter des gestörten Dienstes zu kommen.

OSI-xxvii *Benötigte Komponenten*
Die Komponenten, welche für den Betrieb des Dienstes benötigt werden, müssen ebenfalls bekannt sein.

OSI-xxviii *Fehlerarchiv*
Um die Ursache für ein Problem finden zu können, ist die Historie zur Erkennung von Fehlern/-Abnormalitäten wichtig.

OSI-xxix *Zeitpunkt des Fehlers*
Für die Ursachenforschung ist der genaue Zeitpunkt, zu dem der Fehler aufgetreten ist wichtig. Anhand dessen können zum Beispiel regelmässige wiederkehrende Fehler erkannt werden.

4.3.2 Konfigurationsmanagement

Der Begriff einer Konfiguration kann - wie [Heil 97] bereits für das Netz- und Systemmanagement beschreibt - verschiedene Bedeutungen annehmen, wobei alle für das **Konfigurationsmanagement** von Bedeutung sind.

4.3.2.1 Beschreibendes Konfigurationsmanagement

Wie der Name schon sagt, geht es hier um die Beschreibung der zu managenden Umgebung. Es hilft bei der Inventarisierung und Lokalisierung von Diensten ebenso, wie bei der Bestimmung der direkt daran angeschlossenen Komponenten. Um diese Aufgabe erfüllen zu können, müssen einige Parameter bekannt sein:

OSI-xxx *Geographische Lokalität*
Die geographische Lokalität der Komponente ist einer dieser Parameter. Gerade bei Anfragen zur Sicherheit oder zu den Besitzverhältnissen ist dies wichtigstes Datum. Besonders wenn man sich vor Augen hält, dass fast alle Komponenten der IT-Branche eine Umgehung der Sicherheitseinrichtungen erlauben, wenn man physikalischen Zugriff auf das Gerät hat, welches den physikalischen Container für den Dienst darstellt.

OSI-xxxi *Physikalische Lokalität*
Aufgrund der hohen Vernetzung ist es mittlerweile für die reine Funktionalität aus Anwendersicht nicht mehr wichtig, wo der Dienst physikalisch vorhanden ist. Im Falle von Fehlern kann jedoch gerade diese Information wieder von Bedeutung sein, vor allem, wenn eine Komponente auf einer niedrigen Schicht der OSI-Layer ausgefallen ist.

OSI-xxxii *Logische Position*
Wenn ein Dienst auf einem höheren OSI-Level ausgefallen ist, so ist weniger die Information der physikalischen oder der geographischen Lokalität von Interesse; es ist wichtiger zu wissen,

an welcher logischen Position innerhalb der Dienstematrix der problembehaftete Dienst vorhanden ist. Auch für Abrechnungszwecke kann es sinnvoll sein, zu wissen, welcher Organisation der jeweilige Dienst untersteht.

OSI-xxxiii *Teilkomponenten*
Häufig sind Dienste nicht 1:1 auf die Komponenten oder Programme abbildbar, auf denen sie ausgeführt werden, sondern können ihre Funktionalität erst durch das Zusammenspiel weiterer Subdienste ausführen. Für einen Überblick über den Dienst an sich ist es nun gerade interessant zu wissen, durch das Zusammenspiel welcher Komponenten der neue Dienst entsteht.

OSI-xxxiv *Abhängigkeiten*
Dazu gehört die Information, von welchen anderen Diensten der betrachtete Dienst abhängig ist. Während zum Beispiel ein moderner Webserver oft eine PHP- oder Java-Umgebung beinhaltet, benötigen die Webserver zur Auslieferung und Datenhaltung der anzuzeigenden Webseiten die Mithilfe von eigenständig verwalteten und agierenden Datenbank Management Systemen (DBMS) und Fileserverserverdiensten.

OSI-xxxv *Verbindungen*
Auf einer niedrigeren Schicht ist es interessant, mit welchen anderen Diensten der betroffene Dienst verbunden ist. Sowohl direkt - über eine logische Verbindung - als auch indirekt - über die physikalischen Komponenten, auf denen der Dienst aufgesetzt ist.

4.3.2.2 Prozessorientiertes Konfigurationsmanagement

Hier geht es hauptsächlich um die Beschreibung des Überganges von einem definierten (Ausgangs-) Zustand in einen definierten neuen (Ziel-)Zustand. Dieser erfolgt zumeist über das Setzen und Verändern von Parametern, welche das Verhalten des Prozesses bestimmen. Damit wird es zum ausführenden Organ dessen, was ITIL als “Change Management” bezeichnet.

OSI-xxxvi *Dienststatus*
Um die Auswirkungen der Änderungen auf das Gesamtsystem einschätzen zu können, ist auch der Status einer Komponente von Bedeutung. Eine inaktive Komponente oder eine Komponente im Entwicklungssystem kann in der Regel ohne Auswirkungen auf das Produktivsystem verändert werden.

OSI-xxxvii *Parameter*
Benötigt werden hierfür die *Parameter*, welche das Verhalten des Dienstes bestimmen, sowie die Information darüber, welche anderen Dienste davon betroffen werden und ebenfalls verändert werden müssen.

OSI-xxxviii *Aktivierungszeitpunkt von Parameteränderungen*
Auch ist es wichtig zu wissen, ob die Parameter sofort, nach einem bestimmten Befehl oder erst nach einem Neustart des Gesamtsystems aktiv werden.

OSI-xxxix *Pseudoparameter*
Oft ist es nötig, mit Hilfe eines übergeordneten “Pseudoparameters” eine ganze Reihe von einzelnen Parametern zu verändern, so dass nicht jeder Parameter vom Verwalter einzeln angefasst werden muss.

OSI-xl *Validierung neuer Parameter*
Um Flüchtigkeitsfehler oder Fehler, welche der Komplexität der Einbindung einer Komponente bzw. der Komponente selbst geschuldet sind, zu vermeiden, ist es wünschenswert, dass eine Kontrolle des neu eingegebenen Parametersatzes möglichst automatisiert erfolgen kann.

OSI-xli *Aktuelle Konfiguration*
Ist eine Änderung der Konfiguration durchgeführt worden bzw. zur Vorbereitung auf eine Änderung der Konfiguration, ist die Auskunft über die aktuelle Konfiguration immer von Bedeutung.

OSI-xlii *Konfigurationsarchiv*
Dabei ist es wünschenswert, dass eine Dokumentation nicht nur über den aktuellen Stand existiert, sondern dass auch eine Historie erfolgter Änderungen verfügbar ist.

OSI-xliii *Lizenzen*
Ebenso ist es Aufgabe des Konfigurationsmanagement über bestimmte (lizenz-)rechtlich relevante Komponenten des Systems Buch zu führen. Dazu gehören neben der Anzahl der Komponenten (**OSI-A**) auch die Art der Lizenzen (**OSI-B**), welche für den Betrieb der Komponenten notwendig sind. Des weiteren ist der Zeitpunkt des Ablaufs der Lizenzen (**OSI-C**) und der jeweilige Besitzer der Lizenzen (**OSI-D**) wichtig.

OSI-xliv *Garantie*
Für die Abschätzung der Kosten und der Wirtschaftlichkeit einer Reparatur einer Komponente ist es wichtig, über die zugehörigen Garantien informiert zu sein. Neben der Art der Garantie (**OSI-E**) ist auch der Ablauf der Garantiefrist (**OSI-F**) und die Einschränkung der Garantie (**OSI-G**) wichtig.

4.3.3 Accounting Management, User Administration

Dieser Bereich umfasst zwei Bereiche, die aufgrund der technischen Sicht des Management Frameworks zusammengefasst sind. Es geht hier um die Sicht dem “Kunden” gegenüber.

4.3.3.1 Organisatorisch

Beim organisatorischen Teil geht es um die Verwaltung der Benutzer eines Dienstes, angefangen bei der Identifizierung über die Authentifizierung bis hin zur Autorisierung.

Hierbei handelt es sich weitgehend um Attribute zur Darstellung eines Benutzers, wie er zum Beispiel auch bei dem OSI-fernen CIM bereits sehr ausführlich ausgearbeitet wurde. Da für das Dienstmanagement lediglich die Eigenschaft des Benutzers von Bedeutung ist und nicht seine einzelnen Attribute, wird die Modellierung des Benutzers in einem Managementsystem hier nicht weiter ausgearbeitet.

Einen anderen großen Teil beim organisatorischen Accounting Management wird von der Verwaltung der Berechtigungen (Autorisierungen) eingenommen. Dies wird im Rahmen dieser Arbeit ebenfalls nicht bearbeitet.

4.3.3.2 Fiskal

Der Punkt, den das obere Management natürlich immer am meisten interessiert, ist die Frage des Geldes. Dieser Bereich des IT-Management befasst sich mit genau diesem Punkt, bzw. mit der Beschaffung der Daten zur Beantwortung der damit zusamenhängenden Fragen. So stellt sich die Frage der Abrechnung für angebotene und geleistete Dienste. Hierfür gibt es grundsätzlich zwei Abrechnungmodelle. Zum einen die sogenannte “flat rate”, bei der ein bestimmter Betrag für die Nutzung des Dienstes erhoben wird, der unabhängig von der tatsächlichen Nutzung des Dienstes ist. Zum anderen gibt es die nutzungsabhängige Abrechnung, welche dafür Daten über die tatsächlichen Nutzung des Dienstes eines Benutzers benötigt.

OSI-xlv *Dienstnutzer*
Hierfür ist zunächst einmal wichtig, eine Zuordnung der Nutzung des Dienstes zu einem Nutzer vorzunehmen. Der Nutzer kann hierbei auf eine Person oder eine Organisation zurückzuführen sein.

OSI-xlvi *Parameter für die Abrechnung*
Weiter unterscheiden sich die Dienste dahingehend, dass sie unterschiedliche Parameter anbieten, welche für eine Abrechnung herangezogen werden können. Als typische Beispiele wären hier Zeitpunkt, Zeitdauer, Anzahl versendeter Einheiten (Pakete), Anzahl von Anfragen oder auch der vereinbarte QoS anzuführen.

OSI-xlvii *Einheiten der jeweiligen Zähler*
Um die Abrechnung auch über verschiedene Dienste hinweg vornehmen zu können ist die Einheit der Abrechnung von Interesse. So rechnen manche den Netzwerktraffic in Byte ab, während andere die Einheiten in Megabyte oder in Paketen zählen. Somit ist die Einheit (Bit, Baud, Euro, etc.) immer ebenso wichtig, wie die Anzahl der Einheiten.

OSI-xlviii *Quotas*
Um eine übermäßige Nutzung eines Dienstes zu verhindern, wurde bereits sehr früh das System der **Quotas** eingeführt. Diese können je nach Ausführung eine unterschiedliche Bedeutung haben. Die einen stellen eine harte Grenze an Einheiten dar, welche von einem Kunden in Anspruch genommen werden können. Andere wiederum stellen eine weiche Grenze für Leistungen dar, die mit dem vereinbarten Grundentgelt abgerechnet sind, während weitere zu einem anderen Tarif nachgekauft werden müssen.

OSI-xlix *Nutzungsstatistiken*
Um die Planung der kostendeckenden Verrechnung eines Dienstes zu ermöglichen, ist zumeist neben vagen Prognosen ein Blick in die Vergangenheit nötig. Diesen Blick wiederum bieten Statistiken über die Nutzung der Dienste an.

OSI-l *Abrechnungsmodalitäten*
Zu guter Letzt sind selten die Abrechnungsmodalitäten für alle Dienste und alle Kunden gleich. Um nun die jeweiligen Kosten für einen Kunden zu bestimmen, ist auch ein Blick auf die vereinbarten Modalitäten notwendig.

Es ist *nicht* Aufgabe der technischen Seite, vorzugeben, welche Parameter tatsächlich zur Abrechnung verwendet werden. Die Aushandlung der Verträge wie auch die Durchsetzung der Zahlungen ist nicht Aufgabe der IT.

- subscriber details (demographic data, contract ID, credit information, subscriber history)
- contract information services covered
- contract validity
- authorised users
- quotas
- service level agreements
- billing and payment details
- tariff information
- usage information
- administration system parameters

Tabelle 4.4: Teil der Attribute zum Accounting bei Telcos ([Var 94])

OSI-li *Beispiel von Attributen bei Telcos*
Einige Attribute welche für die Abrechnung bei Telcos benötigt werden und bereitgestellt werden müssen, sind in Tabelle 4.4 angeführt.

4.3.4 Performancemanagement

Dieser Bereich ist eigentlich nur eine Verfeinerung bzw. Erweiterung des Fault Managements. Während letzteres sich nur damit beschäftigt, dass die Dienste überhaupt erreichbar sind, ist es die Aufgabe des **Performancemanagement**, sicherzustellen, dass diese innerhalb bestimmter Parameter angeboten werden und entspricht damit dem Begriff des "Service Level Management" unter ITIL.

Je nach Dienst oder auch Kunde gibt es verschiedene Vorstellungen, welche Parameter zur korrekten, gewünschten Funktionsweise führen.
So kann es für den einen Kunden ohne weiteres innerhalb der akzeptablen Rahmenbedingungen liegen, wenn die Antwort auf eine Anfrage innerhalb eines halben Tages bei ihm eintrifft (z.B. Auskunft über den Besitzer eines Fingerabdruckes), während er die Antwort einer anderen Anfrage innerhalb von Millisekunden haben möchte (z.B. Auskunft über den Halter eines Fahrzeuges). Um ein Beispiel für die unterschiedlichen Anforderungen verschiedener Kunden an das selbe System zu geben, denke man an ein Flugbuchungssystem. Hier benötigt der Ticketverkauf innerhalb kürzester Zeit einen möglichst aktuellen Stand der Belegung, während dies für eine spätere statistische Auswertung der Belegung der verschiedenen Flüge nicht mehr darauf ankommt, dass die Daten sofort verfügbar sind.

Auch gibt es verschiedene Vorstellungen von den technischen Parametern, die erfüllt werden müssen. So kann es für einen Kunden wichtig sein, dass seine Nachricht möglichst schnell und in der richtigen Reihenfolge durchgeleitet wird, dafür aber einzelne wenige Pakete verloren gehen dürfen (Multimedia), während für einen anderen Kunden die Reihenfolge nicht so wichtig ist, dafür aber die Integrität der Daten.

Problematisch ist in diesem Zusammenhang, dass normalerweise mehrere Systeme zusammenspielen müssen und das schwächste System die Gesamtperformance bestimmt.

OSI-lii *Parameter für die Performancebetrachtung*
Es gilt zunächst einmal zu bestimmen, welcher Parameter betrachtet werden soll. Das ziellose Betrachten aller Parameter führt meist dazu, dass die gewonnenen Daten nicht mehr sinnvoll verarbeitet werden können.

OSI-liii *Komponenten für die Performancebetrachtung*
Nachdem die gewünschten Parameter bestimmt sind, ist in einem nächsten Schritt zu bestimmen, welche Komponenten den Dienst ausmachen und welche Daten er vorhält, um die Werte des Parameters zu bestimmen.

OSI-liv *Performancevergleiche*
So kann man über das Performancemanagement zwei verschiedene Arten von Leistungsdaten erhalten. Auf der einen Seite sind dies vergleichende Daten zwischen verschiedenen Komponenten (wenn gewünscht jeweils in unterschiedlichen Situationen).

OSI-lv *Perfomancespitzen*
Auf der anderen Seite ist es möglich, einfach die Spitzenwerte bei der Performance zu ermitteln.

Darüber hinaus bietet das Performancemanagement eine Möglichkeit für das proaktive Fehlermanagement. So gibt es einige Merkmale, die mögliche Fehler vorhersagen können:

OSI-lvi *Historie der Performance*
So lässt eine nachlassende verfügbare Leistung zum Beispiel auf potentielle Probleme schließen, denen zumeist durch die Auswertung von Statistiken bereits im Vorfeld begegnet werden kann.

OSI-lvii *Performanceeinbusen*
Aber nicht nur graduelle Verschlechterungen der gemessenen Werte sind ein Zeichen für Probleme. Auch plötzliche Leistungseinbusen, gleichgültig ob durch eine fehlerhafte Komponent oder besondere Nachfrage bedingt, sind beachtenswert.

OSI-lviii *Zeitpunkte der Leistungseinbrüche*
Ein einzelner Leistungsabfall kann durch verschiedenste Umstände hervorgerufen werden. Durch die Betrachtung der Zeitpunkte, zu denen diese auftreten, lassen sich jedoch evtl. Muster erkennen, die die eigentlichen Ursache finden helfen.

OSI-lix *Koinzidenzen bei Leistungseinbrüchen*
Nicht nur die zeitlichen Zusammenhänge sind wichtig. Gerade im Dienstmanagement, bei dem viele Dienste erst durch die Zusammenarbeit verschiedenster Komponenten zustande kommen, ist es oft noch wichtiger, herauszufinden, ob bestimmte Korrelationen zwischen den Komponenten bezüglich der Leistungseinbrüche zu bemerken sind.

Auf Basis der bereits angesprochenen Daten kann auch bestimmt werden, ob weitere Kunden zu deren Vorstellungen des Service Levels vom System bedient werden können, ohne die Vereinbarungen mit anderen Kunden zu verletzen. In diesem Sinne deckt es auch den Bereich ab, welchen ITIL als “Availability Management” bezeichnet.

4.3.5 Security Management

Das "S" ist zwar der letzte Buchstabe in "FCAPS", deswegen jedoch bei weitem nicht der unwichtigste. Es geht hier nicht darum, dass die Sicherheit innerhalb des Management durchgesetzt wird (dies ist Aufgabe der User Administration), sondern um die Sicherung des laufenden Betriebes vor Attacken.

Der Begriff der Sicherheit hat viele Facetten, weswegen auch die Ansätze zur Gewährleistung derselbigen sich stark unterscheiden. Im großen und ganzen gibt es jedoch nur zwei Hauptrichtungen: Die *Bedrohungen*, denen es zu begegnen gilt, und die *Tasks*, welche dafür sorgen, dass die Bedrohungen vermieden werden.

Bedrohungen Die verschiedenen Bedrohungen sind gegen die die Sicherheit eines Computersystems gerichtet. Dazu gehören:

Passive Attacken: Dies sind Angriffe, die keine Veränderungen an den betroffenen Diensten vornehmen. Ein Beispiel aus der realen Welt wäre hierbei das Abhören einer Email, welche das letzte Angebot zu einem Großauftrag enthält. Die Sendung wird hierbei nicht verändert oder gestört, aber das Wissen kann verwendet werden, um ein entsprechend knapp günstigeres Angebot abzugeben. Außer dem Abhören von Informationen zählen hierzu auch das Erzeugen von Benutzerprofilen, oder sogenanntes "social engineering".

Aktive Attacken: Darunter versteht man Angriffe, die Veränderungen an den betroffenen Diensten vornehmen. Dazu zählen zum Beispiel das Verändern von Nachrichten, die unzulässige Wiederholung einer Nachricht, das Erschleichen einer Benutzerkennung oder von Rechten, Manipulation von Ressourcen durch Überlastung (DOS) oder auch Rekonfiguration bzw. Reprogrammierung von Diensten.

Gestörte Dienste: Unternehmenskritische Dienste müssen immer erreichbar sein und ihre Aufgabe erfüllen können. Zur Sicherheit eines Unternehmens gehört aber auch, dass im Falle eines Unfalles oder gar einer Katastrophe die Erreichbarkeit und Produktivität möglichst schnell wiederhergestellt werden kann. Um Notfallpläne nicht erst im Falle eines Falles unter Stress und limitierten Ressourcen ausarbeiten zu müssen, sollte es sogenannte Business Continuation Pläne oder auch Disaster Recovery Pläne geben, die in einem solchen Notfall, der ein Problem (siehe Kapitel 4.3.1) in sehr großem Maßstab betrifft, verwendet werden können.

Fehlerhaftes Verhalten von Diensten: Zu guter Letzt muss sich der Bereich der Sicherheit auch mit dem Qualitätsmanagement beschäftigen, um Schaden, der durch das fehlerhafte Verhalten von Diensten verursacht wird, möglichst bereits im Vorfeld vermeiden zu können.

Tasks Zu den Aufgaben, welche bearbeitet werden müssen, um ein System überhaupt erst sicher zu machen, gehören unter anderen (aus [HAN 99] entnommen und ohne Anspruch auf Vollständigkeit):

- Das Erstellen einer Gefahrenanalyse
- Die Definition und das Durchsetzen von Sicherheitspolicies
- Das Überprüfen der Identität (Authentisierung über Unterschriften oder (elektronische) Signaturen, Beglaubigung oder Zertifizierung)

- Das Durchführen und Durchsetzen von Zugriffskontrollen
- Das Garantieren von Vertraulichkeit (z.B. über Verschlüsselung)
- Das Gewährleisten der Integrität der Daten
- Die Überwachung von Systemen, um Gefahren für die Sicherheit zu verhindern.
- Die Berichterstattung über den Sicherheitsstatus und versuchte und erfolgreiche Verletzungen der Sicherheit.

Während die Bedrohungen meist durch Erfahrung bekannt geworden sind, sind die Tasks zu einem Teil entstanden, um neuen Bedrohungen entgegenzutreten, zum anderen kommen sie aber auch aus der IT fremden Bereichen und finden in dieser lediglich ein neues Anwendungsfeld. Am meisten kann über diese unter dem Begriff des **Auditing** in Erfahrung gebracht werden, welcher aus dem Businessbereich stammt. Ursprünglich für die Bereiche wie der Zugangskontrolle am Tor des Betriebes, der Warenwirtschaft oder dem Finanzcontrolling entwickelt, sind diese in den alten Bereichen bereits sehr ausgefeilt. Mit der zunehmenden Bedeutung der IT in den Unternehmen und der Virtualisierung des Betriebsgeländes im Internet hat die Abbildung dieser alten Konzepte in die “neue”, “virtuelle” Welt einen immer größeren Stellenwert bekommen.

Das Thema “Sicherheit” muss besonders bei Planung und Entwicklung neuer Systeme eingehend bearbeitet werden. Im laufenden Betrieb sind die meisten Aufgaben immer noch stark von menschlicher Kontrolle geprägt. In automatisierten Systemen wird dies hauptsächlich durch Kontrollmechanismen unterstützt. Die Hauptaufgabe liegt hier in der Sicherung von Übertragungen, der Überprüfung der Einhaltung von Parametern und Begrenzungen, sowie vor allem der Meldung von Abweichungen.

Dies führt dazu, dass hier keine eigenständigen Parameter zu erwarten sind.

OSI-lx *Überprüfen der Parameter*
Stattdessen werden hier Operationen benötigt, die die Überprüfungen von Parametern erlauben.

4.4 Ergebnis der Aufgabenanalyse

Durch die Analyse der verschiedenen Managementaufgaben ist nun ein Katalog von Einzelaufgaben der in ITIL und OSI beschriebenen Aufgaben entstanden. Aus diesem kann man nun die jeweils benötigten Attribute relativ einfach extrahieren. Das Ergebnis eines einfachen Mappings ist in Anhang E für die Attribute aus ITIL und in Anhang F für die Attribute aus OSI zu ersehen. Die dort vorgestellten Attribute wurden jedoch nicht weiter überarbeitet. Die beiden Ansätze wurden getrennt voneinander betrachtet, so dass diese Attribute sich nicht für einen allgemeinen Objektkatalog eignen, der die Anforderungen beider Ansätze erfüllt.

Die so gewonnenen Attribute erfüllen aber die selben Anforderungen, wie der jeweils zugrunde liegende Managementansatz. Dadurch wurde zum Beispiel das große Manko von ITIL in Bezug auf die Anforderungen behoben, da nun Dienstattribute und Methoden vorhanden sind. Einzig das Problem der fehlenden Abstraktion bei OSI, ist durch die alleinige Bestimmung von Attributen leider nicht zu beheben. Dies kann nur in der Darstellung der Attribute in einem Klassendiagramm erfolgen.

Die noch anstehende Kombination der beiden Aufgabenkataloge von ITIL und OSI, sowie die Erstellung eines Klassendiagrammes, welches die Abstraktion von Diensten auch unter OSI ermöglicht, wird im nächsten Kapitel erfolgen.

Kapitel 5

Vorstellung von Dienstmanagementattributen

In dem vorhergehenden Kapitel wurden die Managementaufgaben, welche sich aus den Verfahren zum Dienstmanagement bei ITIL und bei bei OSI ergeben haben, getrennt voneinander betrachtet. Das Resultat waren zwei Attributssammlungen, die jedoch das zu Beginn von Kapitel 4 beschriebene Problem der Nichtüberführbarkeit aufwiesen. Eine ganze Reihe von Attributen haben zwar die gleiche Funktion, werden jedoch aufgrund ihrer Herkunft unterschiedlich bezeichnet. Um dieses Problem zu beheben werden in diesem Kapitel die Aufgaben aus ITIL und OSI zusammengebracht und unter neue Fragestellungen gruppiert. Zusätzlich werden die Attribute bestimmt werden, welche für die Erfüllung der Aufgaben aus ITIL und OSI benötigt werden. Aus den gruppierten Aufgabenstellungen resultieren immer noch eine größere Menge von Fragestellungen. Daher erfolgt die Darstellung unterteilt in verschiedene Bereiche des Dienstmanagements.

Daran anschliesend werden aus den Attributen Klassen erarbeitet, welche die Darstellung der Objekte beim Dienstmanagement ermöglichen. Um den Objektkatalog fertigzustellen, werden diese Klassen dann in einem Klassendiagramm in den Zusammenhang gestellt.

5.1 Bestimmung der Attribute aus gruppierten Managementaufgaben

Viele Aufgaben bei ITIL und OSI ähneln sich. Diese werden nun zusammengestellt und die entsprechenden Attribute bestimmt. Damit diese lange Liste von Aufgaben -/Attributspaaren übersichtlich bleibt, erfolgt die Darstellung nochmals gruppiert und nach Managementaufgaben sortiert.

Die Darstellung erfolgt blockweise und wird jeweils mit einer allgemeinen Frage überschrieben, welche in diesem Bereich des Dienstmanagementes gestellt werden könnte. Dann erfolgt eine Auflistung von Attributen, deren Bedeutung jeweils kurz beschrieben wird. Am Ende folgen die Managementaufgaben, die unter der jeweiligen Frage zusammengefasst wurden. Um die Darstellung kompakt zu halten, werden die Managementaufgaben nur durch ihre Kürzel (Erläuterung siehe Kapitel 4.1.2) referenziert.

5.1.1 Attribute für das Fehlermanagement

5.1.1.1 Wo tritt der Fehler auf? (logisch, zeitlich, örtlich)

Attribut	Beschreibung
elementID	eindeutige Bezeichnung der Störungsmeldung
elementID	eindeutige Bezeichnung des Dienstes
name	Name der Komponente
location	Ort, an dem sich die Komponente befindet
userLocation	Ort, an dem sich der Benutzer der Komponente befindet
serviceLocation	Ort, an dem der Service physikalisch erbracht wird
problemTime	Zeitpunkt, zu dem das Problem gemeldet wurde
incidentType	Art der Störung
incidentReporter	(Rolle) Wer meldet den Vorfall?
owner	(Rolle) Bei wem steht die Komponente?
manager	(Rolle) Wer verwaltet die Komponente?
reachable	Ist es möglich, die Komponente zu erreichen?
Managementaufgaben: ITIL-v, ITIL-xv, ITIL-xxxiv, OSI-i, OSI-xv	

5.1.1.2 Welche Art von Fehler liegt vor?

Attribut	Beschreibung
elementType	Art der Komponente
incidentType	Art der Störung
alarmDescription	Eine Beschreibung der Alarmmeldung
Managementaufgaben: ITIL-xix, ITIL-xv	

5.1.1.3 Welcher Dienst ist wirklich ausgefallen? Und welcher nur aufgrund eines ausgefallenen Dienstes.

Attribut	Beschreibung
incidentType	Art der Störung
impairedService	Der ausgefallene Dienst
elementID	eindeutige Bezeichnung der Komponente/des Dienstes
location	Ort, an dem sich die Komponente befindet
alarm	(Klasse) Alarmmeldungen
alarmSource	Quellkomponente des Alarms
alarmDescription	Eine Beschreibung der Alarmmeldung
alarmParameter	Welcher Parameter der Komponente wurde als fehlerhaft gemeldet
status	Status der Komponente
alarmAck	Methode zur Bestätigung des Alarms
reachable	Ist es möglich, die Komponente zu erreichen?
Managementaufgaben: ITIL-iv, ITIL-v, ITIL-li, ITIL-lxvii, OSI-ii, OSI-iii, OSI-iv, OSI-v, OSI-vi, OSI-viii, OSI-xxv, OSI-xxxvi	

5.1.1.4 Welcher Art ist der ausgefallene Dienst?

Attribut	Beschreibung
documentation	Weiterführende Dokumentation zum Dienst
serviceType	Art des Dienstes
elementType	Art der Komponente
Managementaufgaben: ITIL-xx(ITIL-F,ITIL-G,ITIL-H, ITIL-I), ITIL-xliii	

5.1.1.5 Welche Dienste hängen von dem defekten Dienst ab, bzw. von welchen Diensten hängt der nicht funktionierende Dienst ab und funktionieren diese?

Attribut	Beschreibung
manager	(Rolle) Wer verwaltet die Komponente?
user	Benutzer des Dienstes
dependentElement	Komponenten, die von dieser Komponente abhängen
childElement	Komponenten, die diese Komponente zum Funktionieren benötigt
configurationHistory	alle vorherigen Konfigurationen
servicePriority	Wichtigkeit des Dienstes
connection	Verbindungen zu anderen Komponenten
serviceRoute	Dienste, die benötigt werden, um den konkreten Dienst zu erreichen
status	Status der Komponente
Managementaufgaben: ITIL-ii, ITIL-vii, ITIL-ix, ITIL-xvi, ITIL-xxx, ITIL-xl, ITIL-xli, ITIL-xlv, OSI-xxvi, OSI-xxxiii, OSI-xxxiv	

5.1.1.6 Wer ist von dem Fehler betroffen? (Kunden, Mitarbeiter)

Attribut	Beschreibung
user	Benutzer des Dienstes
sla	(Klasse) Service Level Agreements, die diesen Dienst betreffen
dependentElement	Komponenten, die von dieser Komponente abhängen
Managementaufgaben: ITIL-ii, ITIL-vi, ITIL-vii, ITIL-viii, ITIL-xxxi	

5.1.1.7 Wer ist für den fehlerhaften Dienst verantwortlich?

Attribut	Beschreibung
manager	(Rolle) Wer verwaltet die Komponente?
Managementaufgaben: ITIL-xi, ITIL-xvii, ITIL-xxxiv, ITIL-xlii, ITIL-liii, OSI-ix	

5.1.1.8 Wie sieht die aktuelle Konfiguration des Dienstes aus?

Attribut	Beschreibung
configuration	Eine Konfiguration
configurationHistory	alle vorherigen Konfigurationen
Managementaufgaben: ITIL-xxxvi	

5.1.1.9 Hat es in der letzten Zeit Änderungen an dem Dienst gegeben? Wenn ja, welche?

Attribut	Beschreibung
changeRecord	Dokumentation des Änderungsprozesses
changeHistory	alle vorhergehenden Änderungen, mit Änderungszeitpunkten
deploymentState	In welchem Teil des Lebenszyklus befindet sich der Dienst?
deploymentStateHistory	Historie über den bisherigen Lebenszyklus
documentation	Weiterführende Dokumentation zum Dienst
lastChangeDate	Datum der letzten Änderung
Managementaufgaben: ITIL-xliv, ITIL-liv, ITIL-lv, ITIL-lvi	

5.1.1.10 Wie stark war der Dienst ausgelastet, bevor er nicht mehr ansprechbar war?

Attribut	Beschreibung
serviceStatistics	Sammlung von statistischen Daten zu dem Dienst
serviceConfiguration	Die Konfiguration des Dienstes
performanceHistory	Geschichte der verschiedenen Perfomancezustände
Managementaufgaben: OSI-lv, OSI-lvi	

5.1.1.11 Wie hoch sind die Kosten des Ausfalles?

Attribut	Beschreibung
sla	(Klasse) Service Level Agreements, die diesen Dienst betreffen
slaFine	was ist bei Vertragsverletzung zu zahlen?
incidentTimetoSolve	Wie lange wird es dauern/hat es gedauert, die Störung zu beseitigen
incidentCosttoSolve	Was wird es kosten/hat es gekostet, die Störung zu beseitigen
serviceValue	Was ist der geschätzte Wert dieses Dienstes
servicePriority	Wichtigkeit des Dienstes
dependentService	Abhängiger Dienst
incidentTimestampBegin	Wann hat die Störung begonnen?
incidentTimestampEnd	Wann wurde die Störung behoben?
user	Benutzer des Dienstes
warranty	(Klasse)Garantieinformationen zu diesem Dienst
serviceUnderWarranty	Garantievereinbarungen zu diesem Dienst
incidentPriority	Wichtigkeit der Störungsbehebung
coveredByWarranty	Wird die Störungsbeseitigung durch die Garantie abgedeckt?
warrantyExpirationDate	Ablaufdatum der Garantie
warrantyType	Art der Garantie
Managementaufgaben: ITIL-vi, ITIL-x, ITIL-xvii, ITIL-xviii, ITIL-xxiii, ITIL-xxx, ITIL-xxxii, ITIL-xxxiii, OSI-xliv	

5.1.1.12 Wie hoch ist die geforderte Qualität des nicht mehr ansprechbaren Dienstes?

Attribut	Beschreibung
sla	(Klasse) Service Level Agreements, die diesen Dienst betreffen
slaResponseTime	Vereinbarte Antwortzeit
slaAvailability	Vereinbarte Verfügbarkeit
servicePriority	Wichtigkeit des Dienstes
Managementaufgaben: ITIL-iii, ITIL-vi, ITIL-ix, ITIL-xxv	

5.1.1.13 Wie kann man den Ausfall des Dienstes evtl. umgehen?

Attribut	Beschreibung
alternativeService	ein anderer Dienst, der einen Teil der Aufgaben übernehmen kann
equivalentService	gleichartiger Dienst
incidentRecord	Dokumentation zur Störung
incidentRecordHistory	Historie der Störungsmeldungen
changeRecord	Dokumentation zum Änderungsprozess
changeRecordHistory	Archiv der Änderungsdokumentationen
changeHistory	Archiv der Änderungen
Managementaufgaben: ITIL-xii, ITIL-lix, ITIL-lxiii	

5.1.1.14 Gibt es schon bekannte Probleme dieser Art?

Attribut	Beschreibung
incidentRecord	Dokumentation zur Störung
elementID	eindeutige Bezeichnung der Komponente/des Dienstes
incidentType	Art der Störung
knowledgeBase	Befragung einer Wissensdatenbank
Managementaufgaben: ITIL-xiv, ITIL-xx, ITIL-xxvii, ITIL-xxviii, ITIL-xxix, ITIL-lix, ITIL-lxx, OSI-x, OSI-xi, OSI-xii, OSI-xiii, OSI-xiv	

5.1.2 Attribute für die Vorbeugung von Ausfällen

5.1.2.1 Treten Anomalien auf?

Attribut	Beschreibung
performanceHistory	Geschichte der verschiedenen Perfomancezustände
incidentRecordHistory	Historie der Störungsmeldungen
alarmHistory	Historie der Alarmmeldungen
Managementaufgaben: ITIL-xxvi, ITIL-lix, OSI-iv, OSI-xxviii, OSI-lvi, OSI-lvii	

5.1.2.2 Von welchen Diensten hängt der Dienst ab?

Attribut	Beschreibung
dependentElement	Komponenten, die von dieser Komponente abhängen
childElement	Komponenten, die diese Komponente zum Funktionieren benötigt
configurationHistory	alle vorherigen Konfigurationen
connection	Verbindungen zu anderen Komponenten
elementID	eindeutige Bezeichnung der Komponente/des Dienstes
routeTo	Verbindung von einer Komponente zu einer anderen.
Managementaufgaben: ITIL-vii, ITIL-xl, ITIL-xli	

5.1.2.3 Wie hoch ist die minimale/maximale Auslastung des Dienstes?

Attribut	Beschreibung
performancePeak	maximale Auslastung im Betrachtungszeitraum
performanceDrop	minimale Auslastung im Betrachtungszeitraum
performance	aktuelle Performance
performanceHistory	Geschichte der verschiedenen Perfomancezustände
Managementaufgaben: OSI-lv	

5.1.2.4 Was ist der Bottleneck bei diesem Dienst?

Attribut	Beschreibung
performanceHistory	Geschichte der verschiedenen Perfomancezustände
Managementaufgaben: ITIL-iv, OSI-lix	

5.1.2.5 Wo sind evtl. vorhandene Schwachstellen?

Attribut	Beschreibung
alarmHistory	Historie der Alarmmeldungen
alarmConfiguration	Konfiguration, die den Alarm bestimmt
incidentType	Art der Störung
problemRecordHistory	Historie der Problemdokumentationen
inParameter	Sind die aktuellen Paramter innerhalb des vorgegeben Rahmens?
performanceHistory	Geschichte der verschiedenen Perfomancezustände
Managementaufgaben: OSI-iv, ITIL-iv, ITIL-xxvii, ITIL-xxviii, OSI-xxi, OSI-xxviii, OSI-lvi	

5.1.3 Attribute für das Änderungsmanagement

5.1.3.1 Wann passiert(e) die Änderung?

Attribut	Beschreibung
changeRecord	Dokumentation des Änderungsprozesses
changeDate	Datum der Änderung
changeActivationBy	Wodurch wird eine Änderung aktiv? (Sofort, bei Neustart,...)
Managementaufgaben: ITIL-xxi, ITIL-lxxxiv, ITIL-lxxvii, OSI-xxxviii	

5.1.3.2 Was passiert bei der Änderung genau? Welche Konfigurationen müssen angepasst werden?

Attribut	Beschreibung
serviceConfiguration	Die Konfiguration des Dienstes
configurationHistory	Historie der Konfigurationen
configurationOld	Die alte (aktuelle) Konfiguration
configurationNew	Die neue Konfiguration
dependentElement	Komponenten, die von dieser Komponente abhängen
rfc	Dokument, welches den Änderungsprozess beschreibt
Managementaufgaben: ITIL-xxiv, ITIL-xxvi, ITIL-lxxxv	

5.1.3.3 Welche Dienste sind betroffen?

Attribut	Beschreibung
dependentElement	Komponenten, die von dieser Komponente abhängen
elementID	eindeutige Bezeichnung der Komponente/des Dienstes
childElement	Komponenten, die diese Komponente zum Funktionieren benötigt
configurationHistory	alle vorherigen Konfigurationen
affectedService	Dienste, die angepasst werden werden müssen
Managementaufgaben:	ITIL-lxxiv, ITIL-lxxix, ITIL-lxxx

5.1.3.4 Wer muss gegebenenfalls informiert werden?

Attribut	Beschreibung
owner	(Rolle) Bei wem steht die Komponente?
manager	(Rolle) Wer verwaltet die Komponente?
user	Benutzer des Dienstes
slaContact	Kontaktperson aus dem SLA
dependentElement	Komponenten, die von dieser Komponente abhängen
deploymentState	In welchem Teil des Lebenszyklus befindet sich der Dienst?
Managementaufgaben:	ITIL-vi, ITIL-xxxiv, ITIL-xliv, ITIL-liii, ITIL-lxviii, ITIL-lxxii, ITIL-lxxv

5.1.3.5 In welcher Reihenfolge müssen die Änderungen durchgeführt werden, um eine möglichst hohe Systemverfügbarkeit zu garantieren.

Attribut	Beschreibung
dependentElement	Komponenten, die von dieser Komponente abhängen
determineChangeOrder	in welcher Reihenfolge müssen die einzelnen Komponenten aktiviert werden?
Managementaufgaben:	ITIL-lxxxviii

5.1.3.6 Welche Auswirkungen werden die Änderungen haben?

Attribut	Beschreibung
servicePriority	Wichtigkeit des Dienstes
dependentElement	Komponenten, die von dieser Komponente abhängen
elementParameter	Parameter, welche die Funktionsweise der Komponente bestimmen
Managementaufgaben:	ITIL-lxxix, ITIL-lxxviii, OSI-xxiii

5.1.3.7 Was ist die aktuelle Konfiguration?

Attribut	Beschreibung
configuration	Eine Konfiguration
elementParameter	Parameter, welche die Funktionsweise der Komponente bestimmen
Managementaufgaben:	ITIL-lxxiv, ITIL-lxxvi, ITIL-lxxxvi, OSI-xxxvii, OSI-xli

5.1.3.8 Welches wird die zukünftige Konfiguration sein?

Attribut	Beschreibung
configuration	Eine Konfiguration
configurationVersion	Versionsnummer der Konfiguration
Managementaufgaben:	ITIL-lxxxv, OSI-xxxvii

5.1.4 Attribute für das Konfigurationsmanagement

5.1.4.1 Wie wird die Komponente eindeutig bezeichnet?

Attribut	Beschreibung
name	Name der Komponente
elementID	eindeutige Bezeichnung der Komponente/des Dienstes
elementVersion	Versionsnummer der Komponente
location	Ort, an dem sich die Komponente befindet
Managementaufgaben: ITIL-xxxiv, ITIL-xxxv, ITIL-li, ITIL-lxvi	

5.1.4.2 Wo befindet sich der Dienst?

Attribut	Beschreibung
location	Ort, an dem sich die Komponente befindet
dependsOn	wovon hängt der Dienst ab?
Managementaufgaben: OSI-xxx, OSI-xxxi, OSI-xxxii	

5.1.4.3 Wie kann man die Dateneingabe vereinfachen?

Attribut	Beschreibung
serviceTemplate	Ist dieser Dienst nur ein Template oder aktiv?
childElement	Komponenten, die diese Komponente zum Funktionieren benötigt
configurationHistory	alle vorherigen Konfigurationen
Managementaufgaben: ITIL-xlvi, ITIL-xlvii, ITIL-xlviii, ITIL-xlix	

5.1.4.4 Welche Verbindungen hat die Komponente zu anderen Komponenten?

Attribut	Beschreibung
connection	Verbindungen zu anderen Komponenten
interface	Abstrakte Beschreibung des Interfaces
serviceConfiguration	Die Konfiguration des Dienstes
Managementaufgaben: OSI-xvi, OSI-xxxv	

5.1.4.5 Welche Attribute müssen gesetzt werden, um ein bestimmtes Ergebnis zu erreichen?

Attribut	Beschreibung
serviceConfiguration	Die Konfiguration des Dienstes
nextServiceParameter	zukünftiger Parameterwert
documentation	Weiterführende Dokumentation zum Dienst
knowledgeBase	Befragung einer Wissensdatenbank
Managementaufgaben: ITIL-lxi, OSI-xvii, OSI-xix, OSI-xxii, OSI-xxxvii, OSI-xxxix	

5.1.4.6 Welche Dienste müssen lauffähig vorhanden sein, damit der Dienst überhaupt lauffähig ist? Welche Optionen müssen aufgrund dieser Dienste gesetzt werden?

Attribut	Beschreibung
childElement	Komponenten, die diese Komponente zum Funktionieren benötigt
configurationHistory	alle vorherigen Konfigurationen
configuration	Eine Konfiguration
validateConfiguration	Validierung der Konfiguration
Managementaufgaben: OSI-xxvii, OSI-xl	

5.1.5 Attribute für das Accounting

5.1.5.1 Was kostet der Dienst (wem, wann)?

Attribut	Beschreibung
sla	(Klasse) Service Level Agreements, die diesen Dienst betreffen
slaOwner	Vertragspartner des SLA
owner	(Rolle) Bei wem steht die Komponente?
licenceType	Art der Lizenz
licenceCost	Kosten der Lizenz
licenceOwner	Eigentümer der Lizenz
licenceCustomer	Kunde der Lizenz
user	Benutzer des Dienstes
Managementaufgaben: ITIL-vi, ITIL-liii, ITIL-lvii, OSI-xliii, OSI-xlv	

5.1.5.2 Wie hoch sind die angefallenen Kosten?

Attribut	Beschreibung
slaConfiguration	Parameterset, der in der SLA bestimmt wurde
usageCost	Laufende Kosten für den Betrieb
licenceCost	Kosten der Lizenz
coveredElement	Komponenten, die von der Lizenz abgedeckt werden
coveredElementNo	Anzahl der Komponenten, die von der Lizenz abgedeckt werden
licenceOwner	Eigentümer der Lizenz
licenceExpires	Ablaufdatum der Lizenz
Managementaufgaben: ITIL-lviii, oqlicences	

5.1.5.3 Wonach wird abgerechnet?

Attribut	Beschreibung
accountingConfiguration	Parameterset, der zur Abrechnung verwendet wird
accountingParameterUnits	Einheiten zu den Parametern
licenceDescription	Beschreibung der Lizenz
quota	maximale Einheiten zur Nutzung
sla	(Klasse) Service Level Agreements, die diesen Dienst betreffen
Managementaufgaben: OSI-xlvi, OSI-xlvii, OSI-xlviii, OSI-l	

5.1.5.4 Sind die vereinbarten SLA's eingehalten worden?

Attribut	Beschreibung
sla	(Klasse) Service Level Agreements, die diesen Dienst betreffen
slaDowntime	Dauer der erlaubten Nichtverfügbarkeit
slaConfiguration	Parameterset, der in der SLA bestimmt wurde
Managementaufgaben: ITIL-iii, ITIL-vi, ITIL-xxv, OSI-xlvi	

5.1.6 Attribute für Performancebetrachtungen / Statistiken

5.1.6.1 Was wird betrachtet?

Attribut	Beschreibung
serviceConfiguration	Die Konfiguration des Dienstes
performanceParameter	Parameter, die zur Performancemessung verwendet werden
Managementaufgaben: OSI-lii, OSI-liii	

5.1.6.2 Wie gut war die Reaktionszeit des Dienstes auf Anfragen?

Attribut	Beschreibung
performanceHistory	Geschichte der verschiedenen Perfomancezustände
performancePeak	maximale Auslastung im Betrachtungszeitraum
Managementaufgaben: OSI-liv, OSI-lv	

5.1.6.3 Wie gut war die Reaktionszeit anderer Dienste auf Anfragen dieses Dienstes?

Attribut	Beschreibung
performanceMeasuringPoint	Service, von dem aus die Performancewerte erhoben wurden
Managementaufgaben: OSI-liv, OSI-lv	

5.1.6.4 Wann ist die höchste Auslastung? Und wie hoch ist die Auslastung in diesem Zeitraum?

Attribut	Beschreibung
performanceHistory	Geschichte der verschiedenen Perfomancezustände
Managementaufgaben: OSI-xviii, OSI-xxi, OSI-xlix, OSI-lv, OSI-lvi	

5.1.6.5 Wie hoch ist die Verfügbarkeit?

Attribut	Beschreibung
performanceHistory	Geschichte der verschiedenen Perfomancezustände
incidentRecord	Dokumentation zur Störung
incidentTimestampBegin	Wann hat die Störung begonnen?
incidentResolutionTime	Zeit, die für die Behebung der Störung benötigt wurde
changeRecord	Dokumentation zum Änderungsprozess
Managementaufgaben: ITIL-xxv, ITIL-lxxxi, OSI-lvi	

5.1.7 Attribute für Sicherheitanforderungen

5.1.7.1 Wer hat Zugriff auf diesen Dienst?

Attribut	Beschreibung
sla	(Klasse) Service Level Agreements, die diesen Dienst betreffen
user	Benutzer des Dienstes
accessRestrictions	Zugriffsbeschränkungen
owner	(Rolle) Bei wem steht die Komponente?
manager	(Rolle) Wer verwaltet die Komponente?
Managementaufgaben:	ITIL-vi, ITIL-xxxix, ITIL-xlii, ITIL-liii, ITIL-lxiv

5.1.7.2 Wer hat Zugriff auf die Daten, die diesen Dienst betreffen?

Attribut	Beschreibung
owner	(Rolle) Bei wem steht die Komponente?
manager	(Rolle) Wer verwaltet die Komponente?
Managementaufgaben:	ITIL-xxxiv, ITIL-xxxix

5.1.7.3 Wer darf welche Einstellungen, die diesen Dienst betreffen, verändern?

Attribut	Beschreibung
owner	(Rolle) Bei wem steht die Komponente?
manager	(Rolle) Wer verwaltet die Komponente?
accessRestrictions	Zugriffsbeschränkungen
changeManager	(Rolle) Wer hat die Verantwortung für die Änderung?
Managementaufgaben:	ITIL-xxxiv, ITIL-xxxix, ITIL-lxiv, ITIL-lxviii, ITIL-lxxii, ITIL-lxxxiii

5.1.7.4 Wie werden Änderungen dokumentiert?

Attribut	Beschreibung
changeRecord	Dokumentation zum Änderungsprozess
version	In welcher Version befindet sich die aktuelle Komponente?
accessRestrictions	Zugriffsbeschränkungen
Managementaufgaben:	ITIL-xlix, ITIL-l, ITIL-lii, ITIL-lxiv

5.1.7.5 Welche Dokumentation existiert über abnormes Verhalten des Dienstes?

Attribut	Beschreibung
alarmHistory	Historie der Alarmmeldungen
incidentRecord	Dokumentation zur Störung
problemRecord	Problemdokumentation
validateConfiguration	Validierung der Konfiguration
Managementaufgaben:	ITIL-xxxvii, ITIL-lix, ITIL-lxv, ITIL-lxix

5.1.7.6 Welche vorherigen Zustände der Dokumentation gibt es?

Attribut	Beschreibung
statusHistory	Historie der Stati der Komponente
documentation	Weiterführende Dokumentation zum Dienst
changeRecord	Dokumentation zum Änderungsprozess
Managementaufgaben: ITIL-lxix, ITIL-xxxvi, ITIL-lvi, ITIL-lx, ITIL-lxxiii, ITIL-lxxxii, ITIL-lxxxvii, OSI-x, OSI-xiii, OSI-xxviii, OSI-xxix, OSI-xlii	

5.2 Zusammenfassung der Attribute in Klassen

Nachdem aus den Managementaufgaben die Attribute extrahiert worden sind, müssen noch die Zusammenhänge dargestellt werden. Gerade bei den Attributen, welche eigentlich Beziehungen (und damit Zusammenhänge der Klassen untereinander) darstellen, ist dies unbedingt notwendig.

Abbildung 5.1 stellt eine Gruppierung zusammengehöriger Attribute in Klassen dar. Diese Darstellung beschränkt sich auf die einzelnen Klassen, die Assoziationen, welche zwischen diesen Klassen bestehen, werden dort nicht dargestellt. Es fällt auf, dass einige dieser Klassen, wie zum Beispiel *RFC*, keine Attribute enthalten. Das ist nicht auf ein Versehen oder Zeitnot zurückzuführen. Vielmehr ist der Inhalt dieser Klasse - im Gegensatz zu ihrer Existenz - für das Dienstmanagement nicht interessant. Die Attribute dieser Klasse sind zum Beispiel mit Hilfe des in Anhang C dargestellten Vorschlages für ein Datenblatt für Requests for Change (RFC) aus ITIL zu erhalten. Ebenso verhält es sich mit der Klasse *Location*, für welche Attribute (*Name*, *PhysicalPos*, *Address*) im Core-Modell von CIM [DMTF 03] (bei Version CIM Version 2.7) beschrieben sind.

Die Klassen alleine sind jedoch für das Dienstmanagement nicht ausreichend. Um den Kontext der Dienste verstehen zu können, sind die Beziehungen der einzelnen Objekte untereinander unverzichtbar. Die Darstellung dieser Assoziationen und Abhängigkeiten wird mit Hilfe des in Abbildung 5.2 dargestellten Klassendiagramm vorgenommen. Die Beziehungen entstanden dabei aus den Attributen, welche durch die Zusammenstellung zu Klassen wurden; zum Teil wurden sie jedoch bei der Attributssammlung bereits als Abhängigkeiten beschrieben.

Ein großer Teil der Logik wurde hierbei in die engültige Modellierung der zu verwaltenden Dienste und in die Implementierung gesteckt. Zum Beispiel kann in der Realität nicht jeder Dienst aus beliebigen anderen Diensten bestehen, was mit diesem Modell ohne weiteres möglich wäre.

Ein anderer Nachteil, der bei der vorliegenden Modellierung auftreten kann, ist bei CIM zu sehen. Aufgrund der offenen Struktur gibt es bei CIM eine sehr hohe Anzahl an Klassen, die miteinander assoziiert sind. Dabei ziehen sich die Assoziationen quer durch die Modellierung. Es ist nicht mehr möglich, einzelne Bereiche unabhängig von anderen zu betrachten. Im Gegenteil muss man immer Klassen im Blick behalten und versuchen im großen Kontext zu arbeiten.

Schwierig ist in dem vorgestellten Klassendiagramm auch, ein externes System für die Verwaltung der Dokumentation von Störungen, Problemen, Änderungen oder sonstiger Dokumentation einzubinden. Dieses müsste an mehreren Stellen in die Modellierung eingefügt und angepasst werden. Es existiert keine definiert Schnittstelle über welche die Dokumentation abgewickelt werden kann.

Für die Lösung der aufgezeigten Probleme werden nachfolgend Design Pattern verwendet werden.

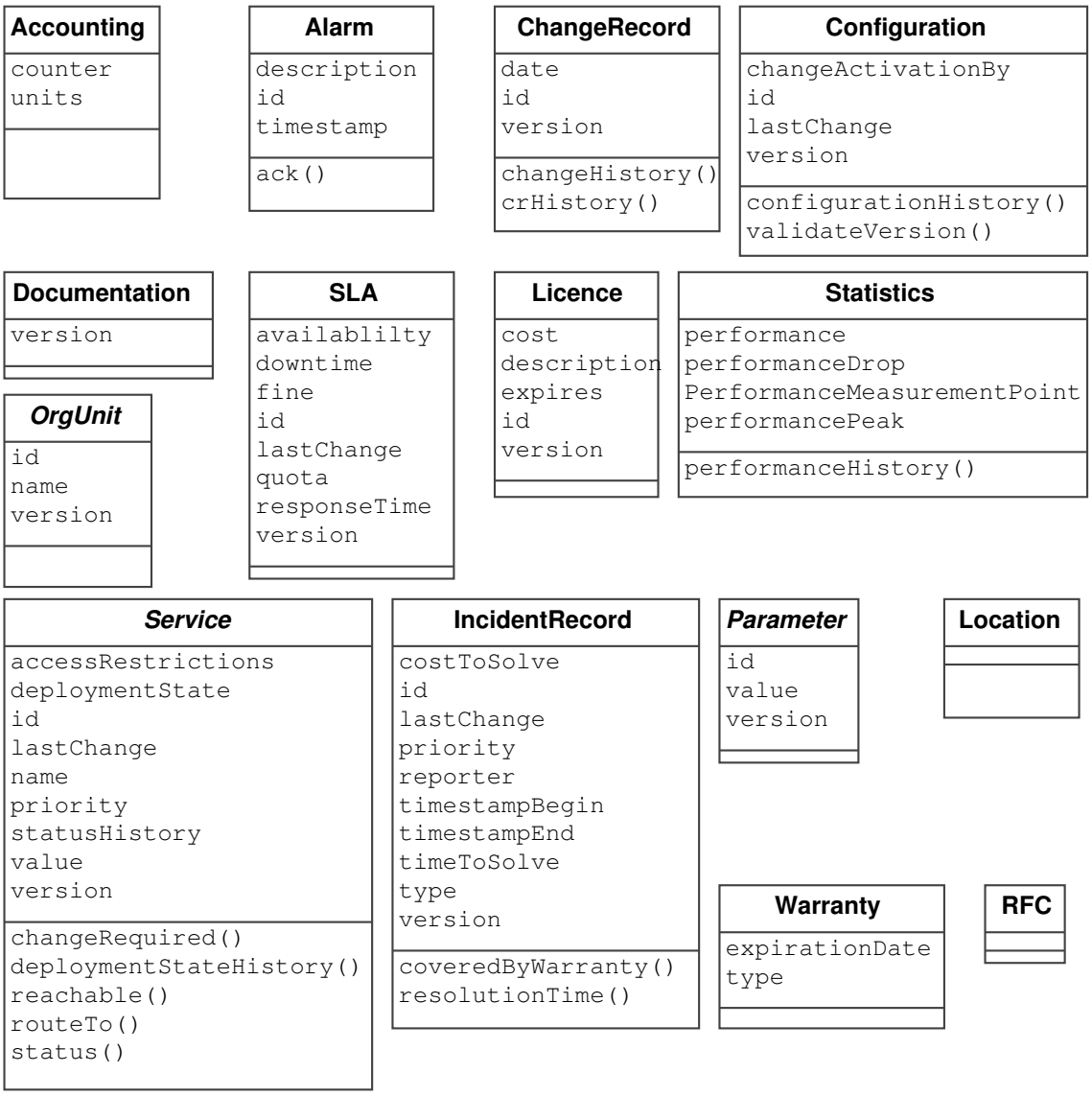

Abbildung 5.1: Übersicht der einzelnen Klassen

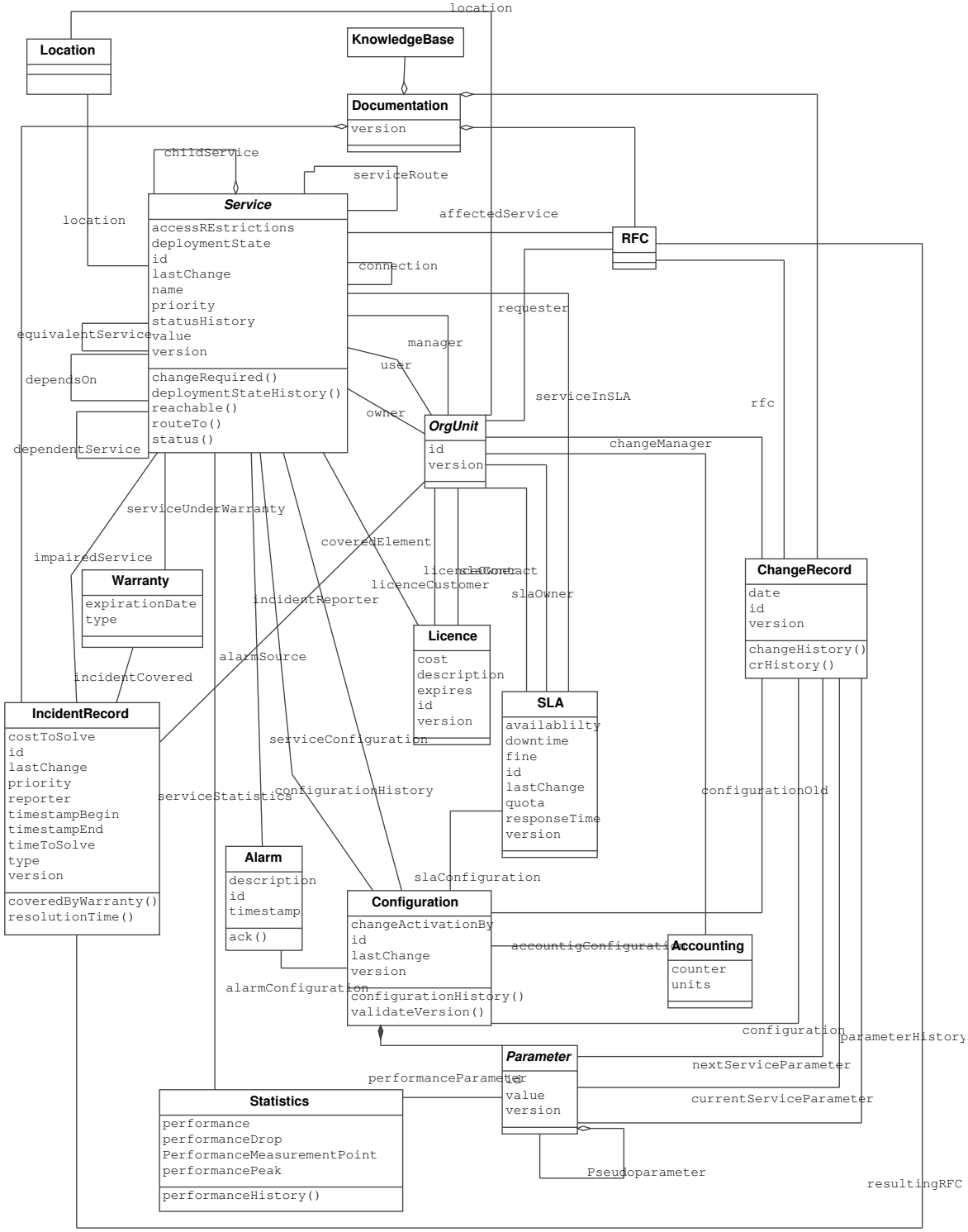

Abbildung 5.2: Klassendiagramm der Service-Informationen

Kapitel 6

Vereinfachung des Objektkataloges

Wie bereits in vorherigen Kapitel vermerkt, ist das in Abbildung 5.2 dargestellte Klassendiagramm und der daraus resultierende Objektkatalog nicht optimal. Mit Hilfe von Design Pattern wird der Objektkatalog nun verfeinert werden, so dass die in Kapitel 2.4 aufgeführten Anforderungen erfüllt werden können. Besonders werden die Design Pattern behilflich sein, Herstellerunabhängigkeit und Modularität herzustellen. Sie helfen zusätzlich auch bei der Verfeinerung der darstellbaren Klassen, so dass bestimmte unmögliche Kombinationen bereits durch das Design ausgeschlossen werden können.

6.1 Werkzeug: Design Pattern

Jedem Programmierer ist schon einmal aufgefallen, daß er Code, den er gerade mühsam produziert, in ähnlicher Form schon einmal geschrieben hat. Also wird er sich mit der Zeit eine Reihe von Codekonstrukten aufbauen, die er mit leichten Anpassungen immer wieder verwenden kann. Diese Konstrukte wird er dann ähnlich den Bibliotheken, welche sich wiederholende Methoden beinhalten, immer häufiger in seine Produkte einbauen.

6.1.1 Entstehung von Design Pattern

Das Problem der Modularisierung gibt es auch für den Bereich des objektorientierten Softwaredesigns. Die Module nennen sich “Design Pattern”. Diese entstanden aus der Überlegung heraus, dass erfahrene Programmierer es geschafft haben, die größten Fettnäpfchen beim Design von objektorientierten Anwendungen zu umgehen und wiederverwendbaren Code mit wiederverwertbarem Design zu schaffen. Dagegen sind weniger erfahrene Programmierer oft von den Möglichkeiten erschlagen und verirren sich in Designmethoden, die mit Objektorientierung wenig zu tun haben. Ganz davon zu schweigen, dass der Code meistens eben nicht wiederverwendbar war.
Erfahrene Programmierer verwenden beim Design ihrer Systeme immer wieder bestimmte Muster, die sich als brauchbar erwiesen haben und lassen diejenigen weg, welche sich nicht bewährt haben; diese brauchbaren Muster nennt man “Design Pattern”.

In [GHJV 95] haben Gamm, Helm, Johnson und Vlissides einige der Pattern zusammengefasst, welche sich in ihrer täglichen Arbeit als hilfreich erwiesen haben. Dem Aufruf, weitere Pattern zu finden

und zu begründen, sind weitere Designer wie zum Beispiel Martin Fowler in [Fowl 02] gefolgt, wodurch eine ganze Sammlung von Design Pattern entstanden ist. Als zentrale Anlaufstelle hat sich eine von der Hillside Group betriebene Webseite ([HillGr]) etabliert. Es gibt sogar Versuche, das Prinzip auf andere Bereiche, wie das Design von Häusern und Städten, abzubilden.

6.1.2 Verwendete Design Pattern

Es gibt mittlerweile eine große Anzahl an verschiedenen Pattern, die alle in dem einen oder anderen Bereich der Softwareentwicklung ihre Bedeutung haben. Um den Umfang dieser Arbeit jedoch nicht ausufern zu lassen ([GHJV 95], beschreibt einen Teil der verfügbaren Pattern am Beispiel eines Programms in einem ganzen Buch), wird hier nur auf eine sehr begrenzte Anzahl an Pattern eingegangen. Dazu gehört das Composite Pattern, welches die saubere Darstellung von rekursiv darstellbaren Objekten ermöglicht. War bisher bei der Modellierung von Diensten in Diensten immer ein großer Aufwand notwendig, um primitive und zusammengesetzte Dienste zu unterscheiden, ist dies mit Hilfe dieses Pattern leicht zu bewerkstelligen.
Das Facade Pattern ermöglicht es, logisch voneinander unabhängige Bereiche, die im Modell sehr stark miteinander verflochten sind, voneinander zu trennen. Dies ist gerade im Hinblick auf die Flexibilität bezüglich Erweiterungen und standardisierte Schnittstellen notwendig.
Das Bridge Pattern wiederum schafft eine Trennung von abstrakter Darstellung des Dienstes und seiner tatsächlichen Implementierung. Um herstellerunabhängig arbeiten zu können, ist dies eine der Hauptvoraussetzungen. Ob die Webseiten nun von einem Apache-Webserver in Version 1.x oder in der Version 2.x oder gar durch einen Internet Information Server (IIS) der Firma Microsoft dargestellt werden, ist für den Anwender nicht relevant. In der Implementierung der Attribute und Methoden unterscheiden sich diese jedoch zum Teil massiv.

6.1.2.1 Composite Pattern

Zunächst einmal wird das sogenannte **Composite Pattern** betrachtet werden. Wenn man sich das Szenario in Kapitel 2.1 betrachtet, so wird auffallen, dass die Verantwortlichen für einen Dienst nicht nur einzelne Personen sind, sondern zum Teil auch Organisationseinheiten. Diese wiederum können auch wieder Teil einer Organisation sein und so weiter. So ist der Betreuer des Webservers (eine Person) Mitglied der Arbeitsgruppe, welche wir “Web” genannt haben, diese wiederum ist Teil des LRZ, welches wiederum Teil des MWN ist. Diese Verhältnisse könnte man relativ einfach mit dem Klassendiagramm in Abbildung 6.1 darstellen. Diese Darstellung zeigt aber auch die Schwachstellen des Modelles auf. Personen, welche in Organisationen eingegliedert werden, haben eine andere Beziehung zu ihrer Organisation, als Organisationen, die Teil einer anderen Organisation sind. Diese einfache Darstellung verwendet - natürlich etwas elaborierter - zum Beispiel auch CIM bei der Darstellung der Organisationen und Benutzer.
Wendet man nun aber das Composite Pattern an, so ergibt sich ein anderes Bild (Abbildung 6.2). Wie hier zu sehen ist, kann jede Person Teil einer Organisation sein, ebenso wie jede Organsition wieder Teil der Organisation sein kann; und das ohne mehrere Beziehungen abfragen zu müssen. Wichtig ist dabei, dass beim Composite Pattern die Oberklasse immer eine abstrakte Klasse ist, welche sowohl die primitive Klasse als auch die kombinierte Klasse darstellt.

Das Composition Pattern findet seine Anwendung auch, wenn es darum geht, die Forderung nach Abstraktion zu erfüllen, wenn verschiedenste Dienste zusammenarbeiten. Um dies zu veranschauli-

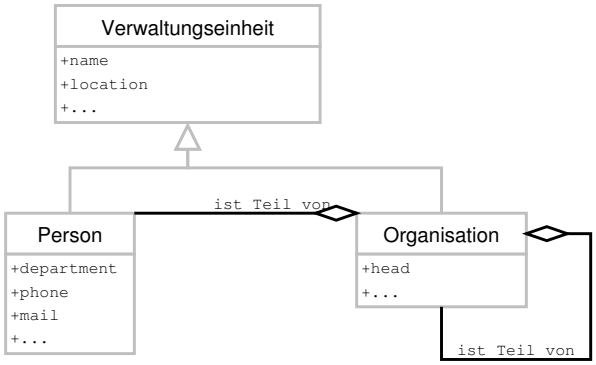

Abbildung 6.1: Naive Modellierung von Personen in Organisationen

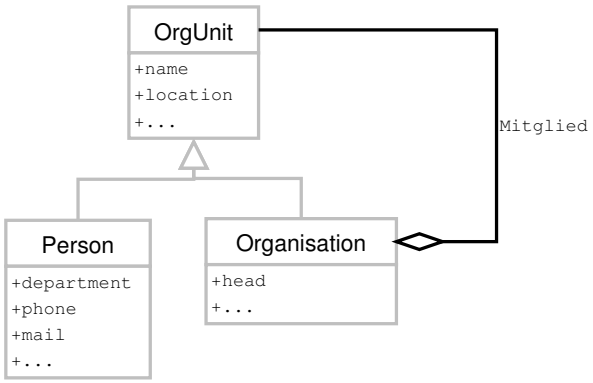

Abbildung 6.2: Darstellung der Personen in Organisationen mit dem Composite Pattern

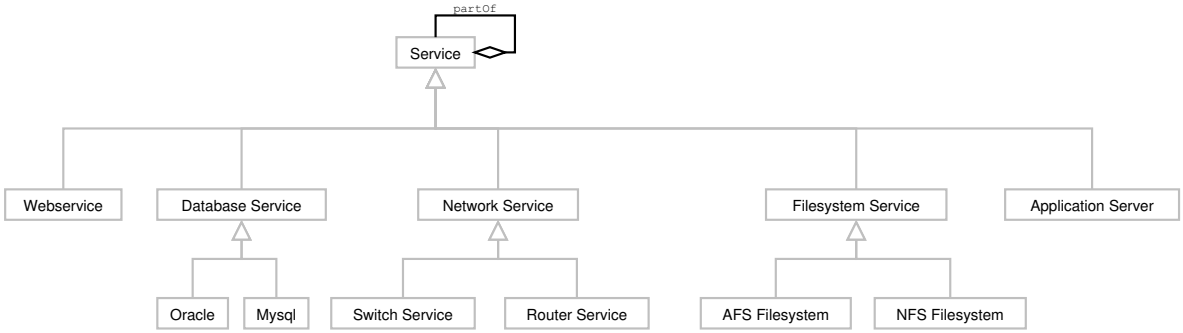

Abbildung 6.3: Naiver Ansatz der Modellierung von Diensten

chen, wird mit Abbildung 6.3 ein sehr naiver Ansatz zur Modellierung der verschiedenen Dienste aus dem Anfangsszenario gezeigt. Zu diesem Modell muss gesagt werden, dass es noch falsche Dienstkombinationen zulässt, wie zum Beispiel einen Switch-Service, der aus Datenbank- und Webservern zusammengebaut ist. Sollen solche Kombinationen verhindert werden, sind weitere Vererbungshierarchien notwendig. Es muss die Aggregation aufgelöst und durch eine Vielzahl anderer Aggregationen ersetzt werden. Wesentlich einfacher und erstaunlicherweise auch korrekter wird die Modellierung, wenn die Dienste in der Form dargestellt werden, wie dies Abbildung 6.4 zeigt. An dieser Darstellung ist auffällig, dass die einzelnen Dienste nicht mehr aufgeführt werden. Dies hat seine Ursache in der

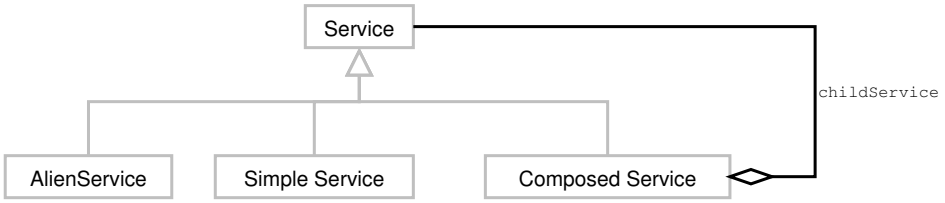

Abbildung 6.4: Darstellung der Rekursion von Diensten mit dem Composite Pattern

Vielzahl der Dienste, die in einer Reihe nebeneinander stehen müssten.
Deswegen hier eine Beschreibung, die zeigt, welche Dienste an welche Stelle gesetzt werden müssen:

Simple Service "Simple Service" ist jeder normale Dienst, der nicht weiter zusammengesetzt ist. Dazu gehören: "Switch Service", "Router Service", "NFS-Service", "AFS-Service", "Oracle", "Mysql" und jeder einfache Webservice, der nicht aus mehreren Komponenten besteht.

Alien Service Die Bezeichnung "Alien Service" ist für all diejenigen Dienste gedacht, über welche die managende Organisation keine direkte Kontrolle haben. Dies kann zum Beispiel ein eingekaufter Datenbankserver sein, oder im Falle des LRZ aus Sicht der "Web"-Arbeitsgruppe der AFS-Dienst.

Composed Service Die eigentliche Bedeutung erlangt das Pattern aber erst durch den mit "Composed Service" bezeichneten Service. Hierbei handelt es sich um jegliche Art von Dienst, der wiederum aus Diensten bestehen, die wiederum aus Diensten bestehen, und so weiter. Als Beispiel hierfür wäre ein Webservice zu nennen, der aus einem einfachen Webservice, einem Mysql- und einem NFS-Dienst besteht. Aus Sicht der für die AFS-Server zuständigen Arbeitsgruppe kann hierunter aber auch der AFS-Cluster verstanden werden, der aus einzelnen AFS-Servern besteht.

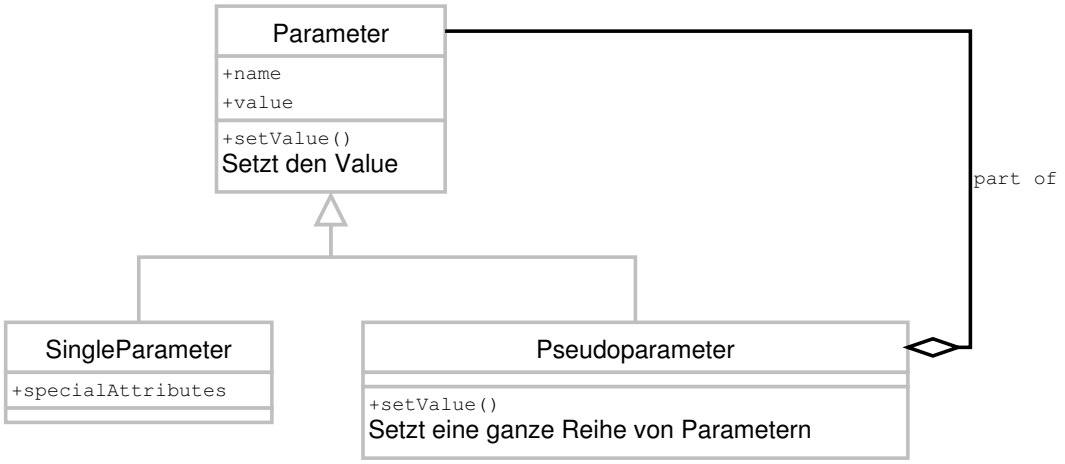

Abbildung 6.5: Modellierung eines Pseudoparameters

Zu guter Letzt noch eine Darstellung der Verwendung dieses Patterns bei den Parametern. So verlangt OSI in OSI-xxxix nach einem Pseudoparamter, welcher in einem Arbeitsschritt gleich eine ganze Menge von einzelnen Parametern zu verändern in der Lage ist. Mit Hilfe des Composite Pattern modelliert stellt sich dies dann, wie in Abbildung 6.5 aufgezeigt, dar.

Das Composite Pattern eignet sich also immer dann, wenn mit Bestandteilen von Komponenten gearbeitet wird, welche wiederum selber rekursiv zusammengesetzt werden können. Und gerade davon gibt es im Dienstmanagement eine ganze Menge. Um dies noch einmal an einem Beispiel zu demonstrieren: Cluster bestehen aus Rechnern, diese können aber wieder zu Clustern zusammengesetzt sein, die innerhalb des großen Clusterverbundes als eine Einheit auftreten.

6.1.2.2 Facade Pattern

Ein anderes Pattern, der bei der Modellierung des Dienstmanagements von unschätzbarem Wert ist, ist das **Facade Pattern**. Ohne es groß zu bemerken, ist es bereits in den vorhergehenden Beispielen zum Composite Pattern verwendet worden. Die Aufgabe des Facade Pattern ist es, voneinander unabhängige Subsysteme zu schaffen, die getrennt behandelt werden können. Dazu werden alle Verbindungen zu und von diesem System durch eine gemeinsame Schnittstelle geleitet, welche die Verbindung des Subsystems zu allen anderen Systemen herstellt. Diese Schnittstellenklasse verschattet also die Komplexität und die dahinter liegenden Klassen.

Verwendet wurde es zum Beispiel bei der Darstellung des AFS-Dienstes, welcher bisher als ein einziger Dienst angesehen wurde. Tatsächlich handelt es sich aus Sicht der “AFS”-Arbeitsgruppe jedoch um eine ganze Reihe von Servern, die unterschiedliche Aufgaben wahrnehmen können und untereinander ihre eigenen Beziehungen zum Austausch von Nachrichten pflegen. Nach “außen” - also zu den Clients hin - treten sie jedoch als ein System mit einer gemeinsamen Schnittstelle auf. Die eigentliche Komplexität wird also im Normalfall gegenüber den Clients verschattet. Ein anderes Beispiel sind die als “Alien Services” bezeichneten Dienste. Auf welcher Basis diese Dienste arbeiten, und wovon sie abhängen, muss den Kunden normalerweise nicht interessieren. Was er zu sehen bekommt, ist ein Dienst; egal wie komplex dieser Dienst tatsächlich aufgebaut ist.

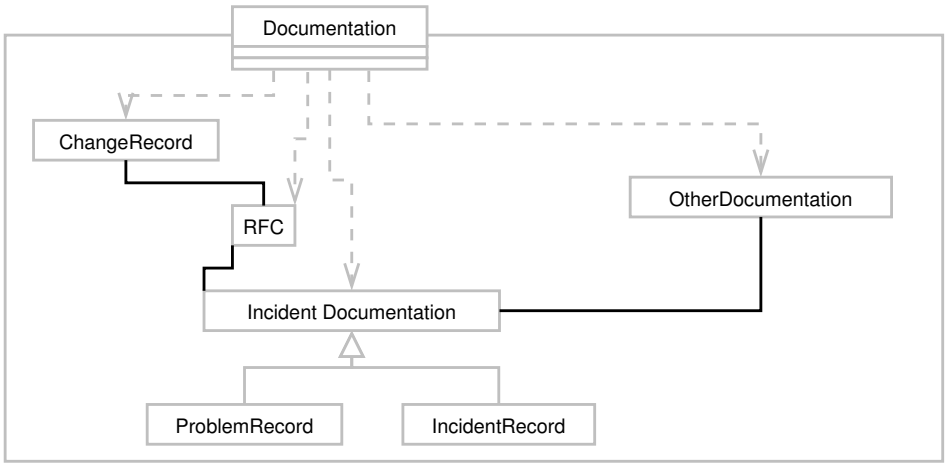

Abbildung 6.6: Verschattung der Dokumentationen mit dem Facade Pattern

Ein anderer Punkt, an dem die Verwendung des Facade Pattern als sinnvoll erachtet werden kann, ist die Dokumentation. Da Dokumentation zu allen Vorgängen und Komponenten innerhalb des gemanagten Bereiches vorhanden sein soll, haben die Klassen, welche die tatsächliche Dokumentation vornehmen, Assoziationen zu einer großen Anzahl von Klassen; und dies dazu mehrfach pro Klasse. Durch die Verwendung des Facade-Patterns, wie es in Abbildung 6.6 vorgeschlagen wird, lassen

sich diese Mehrfachassoziationen elegant auflösen. In diesem Fall jedoch mit dem Nachteil, dass eine zusätzliche Nachricht versendet werden muss, welche Art von Dokumentation gewünscht wird.
Wenn allerdings, wie zum Beispiel am LRZ, bereits ein System zur Dokumentenverwaltung in Betrieb ist, ermöglicht dies erst, dieses in das Design mit aufzunehmen, was den vermeintlichen Nachteil bei weitem aufwiegt.

6.1.2.3 Bridge Pattern

Eine andere Zielrichtung verfolgt das **Bridge Pattern**. Dieses hat seine Aufgabe weniger in der Modellierung, als vielmehr später in der Implementierung. Abbildung 6.7 stellt das Pattern anhand eines abstrakten Dienstes dar. Die beiden Oberklassen, welche die Verbindung zwischen Abstraktion und Implementierung realisieren, sind dabei abstrakte Klassen. Vererbungshierarchien darunter sind also ausdrücklich erwünscht.

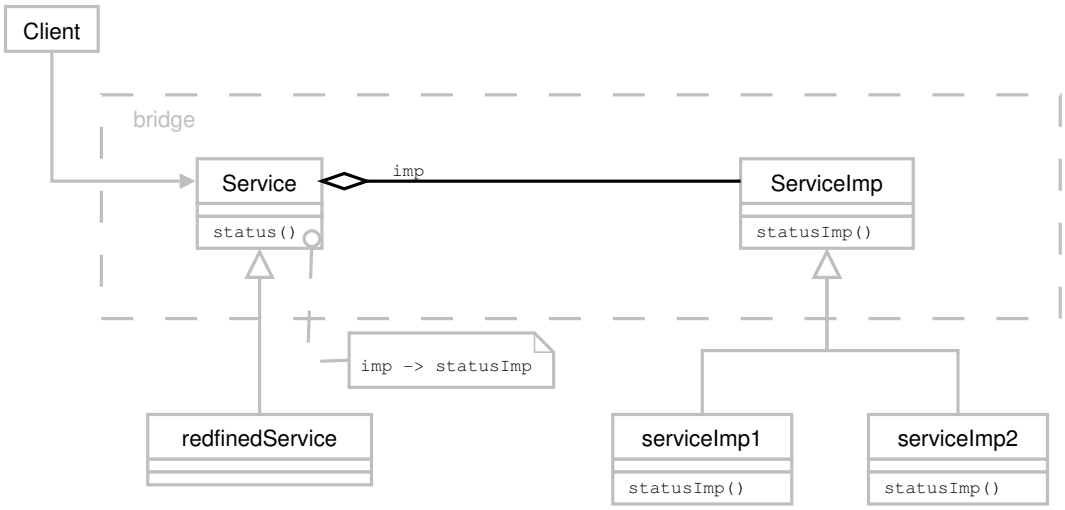

Abbildung 6.7: Darstellung des Bridge Pattern mit abstraktem Dienst

Die Aufgabe des Bridge Pattern ist vornehmlich die Trennung von Abstraktion und Implementierung. Somit ist es unter anderem möglich, die Implemetierung einer Klasse auch noch während der Laufzeit zu ändern. Ein typischer Fall ist die Einbindung eines Treibers, und zwar nicht nur bei Hardwaretreibern, sondern auch bei der Anbindung von Datenbanksystemen zum Beispiel via Open DataBase Connectivity (ODBC). Dabei stellt ODBC eine standardisierte Schnittstelle bereit, die von Programmen angesprochen werden kann. Die Klassen dieser Schnittstelle sorgen für die Konvertierung der Anweisungen in die entsprechenden Befehle für die jeweiligen Datenbanken.

Ähnliche Szenarien treten auch beim Service Management immer wieder auf. Ein typisches Beispiel wurde auch schon im LRZ-Szenario angesprochen. Durch die geplante Umstellung der Layer4/7-Switches ändert sich auch ein Teil der Schnittstellendetails für die Verwaltung dieser Komponenten. Normalerweise würde dies bedingen, dass die dazugehörigen Klassen umgeschrieben und zum Zeitpunkt der Umstellung der Switches das gesammte Managementsystem auf einen neuen Stand gebracht werden müsseb.
Durch die Trennung von Abstraktion und Implementierung lässt sich dies jedoch vermeiden und der Umstellungszeitpunkt der Software wird unabhängig von dem der Hardware - natürlich mit dem Cons-

traint, dass die neue Software vor der neuen Hardware in Einsatz kommen muss. Abbildung 6.8 stellt

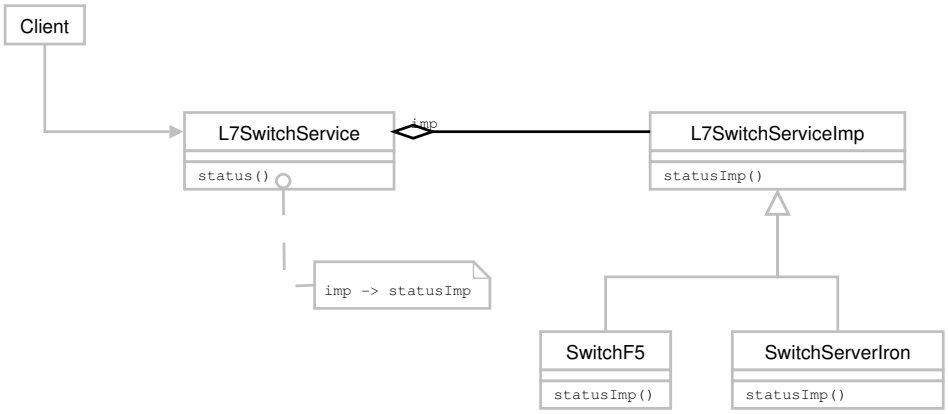

Abbildung 6.8: Bridge Pattern bei der Umstellung der Layer4/7 Switches

dies noch einmal graphisch dar. Hier wurde jedoch das für die Dienste verwendete Composite Pattern bewusst nicht mit eingebunden, um die Graphik nicht unnötig zu verkomplizieren.

6.2 Design des Objektkataloges mit Hilfe der Design Pattern

Unter Anwendung dieser drei vorgenannten Design Pattern lässt sich das ursprüngliche Klassendiagramm 5.2 zur Dienstmodellierung bereits wesentlich übersichtlicher und näher an den ursprünglichen Anforderungen designen.

Beim Vergleich dieser beiden Klassendiagramme ist die Wirkungsweise des Facade Pattern besonders gut zu sehen. Durch die Verwendung wirkt das Klassendiagramm wesentlich aufgeräumter und vor allem ist es möglich, den Bereich der Dokumentation über eine eigene Schnittstelle anzusprechen, was einen modularen Aufbau erst ermöglicht.

Die undeutliche Darstellung der Komposition von Diensten durch eine Beziehung von Diensten zu Diensten konnte ebenfalls aufgehoben werden und durch eine eindeutigere Darstellung ersetzt werden. Somit ist es jetzt möglich, die Beziehung der Teile zum Ganzen deutlich zu sehen. Zusätzlich ist es einfacher zu bestimmen, welche Dienste selbständig sind und welche aus mehreren Diensten zusammengesetzt wurden.

Ein anderer Vorteil, der sich durch die Verwendung von Design Pattern ergibt, ist die klarere und eindeutigere Modellierung der Umgebung. Ist die Darstellung der Organisationen in der aktuellen Version von CIM mit 11 miteinander zusammenarbeitender Klassen noch aufwändig zu lösen, so benötigt der hier verwendete Ansatz nur drei Klassen.

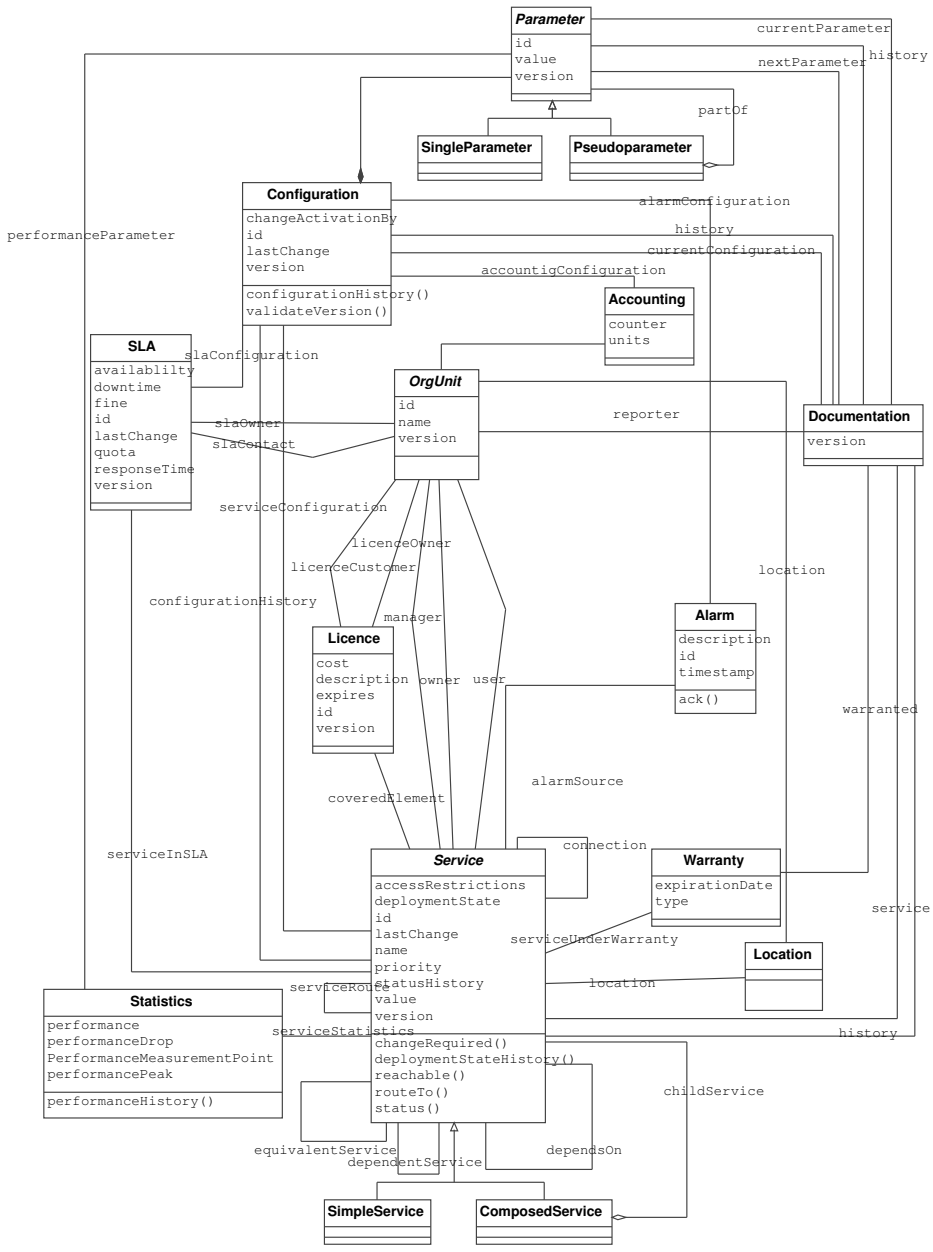

Abbildung 6.9: Diagramm der Klassen unter Verwendung von Design Pattern

Kapitel 7

Anwendung des Objektkataloges

Dass der Objektkatalog auch tatsächlich anwendbar ist, soll im nun Folgenden nachgewiesen werden. Zunächst einmal wird das ursprüngliche Beispiel aus Abbildung 1.1 mit Hilfe des vorgestellten Objektkataloges modelliert, da die grundsätzliche Anwendbarkeit der Klassen und die Modellierbarkeit der Abhängigkeiten hier schön dargestellt werden können.
Um die Anwendbarkeit des Objektkataloges auch an einem realitätsnäheren Beispiel zu verdeutlichen, bei dem die Darstellung nicht so einfach und von vornherein abstrakt ist, wird anschließend das Szenario aus Kapitel 2.1 mit Hilfe des Objektkataloges abgebildet werden.
Abschließend wird die praktische Anwendbarkeit an dem "Problemfall" aus Kapitel 2.2.2 dargestellt werden. Das vorgestellte Problem wurde jedoch ein wenig erweitert, um die Möglichkeiten und Vorteile des mit Hilfe des Objektkatalogs implementierten Dienstmanagements besser darstellen zu können.

7.1 Abbildung des Beispieles

Um mit einer einfachen Modellierung zu beginnen, wird das Beispiel aus Abbildung 1.1 verwendet. Es eignet sich deswegen für eine erste Darstellung der Verwendbarkeit des Objektkataloges, da es zwar einerseits relativ einfach aufgebaut ist, andererseits aber doch die wesentlichen Anforderungen an das Dienstmanagement zeigt. Die einzelnen Objekte haben zwar keine Attribute, aber die Abhängigkeiten sind dennoch deutlich zu sehen. Das Ergebnis ist in Abbildung 7.1 zu sehen.

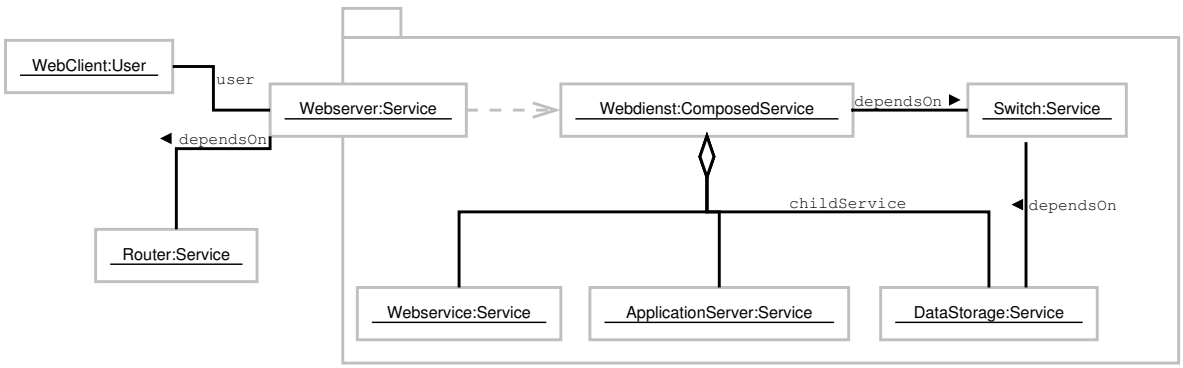

Abbildung 7.1: Objektdiagramm zum Ausgangsbeispiel

Die tatsächliche Sichtweise das WebClients, also des Benutzers des Dienstes wird ebenfalls dargestellt. Er sieht nur den Dienst, welchen der “WebServer” erbringt. Die dahinterliegende Komplexität, dass der Webserver in Wirklichkeit seine Daten aus einer anderen Quelle bekommt, wird vor ihm verborgen.
Dank des Bridge Pattern ist auch es auch nicht nötig, ihm diese Details darzustellen. Sie können vollkommen unabhängig zu einem späteren Zeitpunkt eingebunden werden.

7.2 Abbildung des LRZ-Szenarios

Um ein komplexeres Beispiel darzustellen, wird nun die Modellierung des Szenarios am LRZ (Kapitel 2.1) vorgeführt. Zunächst gilt es, die Dienste des LRZ zu modellieren. Abbildung 7.2 stellt hierfür die einzelnen Dienste vor, welche bis dato nur als “simpleService” benannt wurden. Dafür sind die

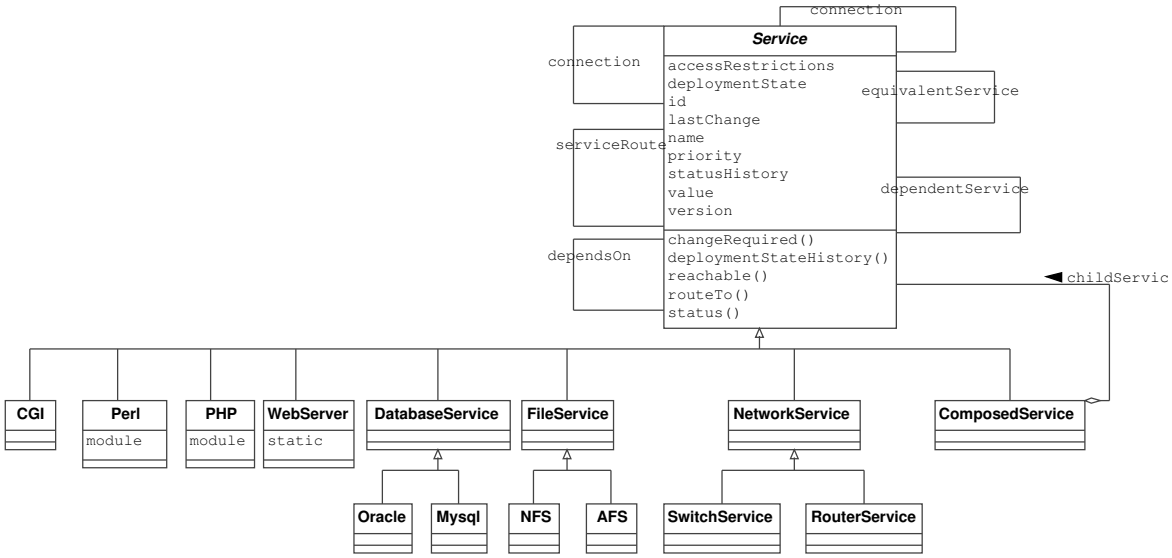

Abbildung 7.2: Dienste im Szenario

einzelnen Komponenten der Infrastruktur, welche in Abbildung 2.1 dargestellt wurden, jeweils als primitive Dienste angesehen worden. Zusätzlich wurden die bei der Beschreibung der Webserver vorgestellten Bestandteile der verschiedenen “Dämonen” hier eingefügt.

Um die Darstellung auf das bekannte Szenario zurückführen zu können, hier eine Anleitung für die Zusammensetzung der verschiedenen Dämonen aus dem Kapitel 2.1.1 mit Hilfe der primitiven Dienste und den Werten ihrer Attribute:

lrz Der LRZ-Dämon besteht aus den Apache Dämon, in der statischen Fassung.
spez Der spezielle Dämon ist nicht statisch und beinhaltet die Dienste PHP als Modul und CGI.
virt Der Dämon für die virtuellen Webserver ist statisch kompiliert und beinhaltet die Dienste PHP als Modul und CGI.
ars Der Dämon für den Ars-Dämon ist ebenfalls nicht statisch ausgelegt und beinhaltet den Dienst Perl

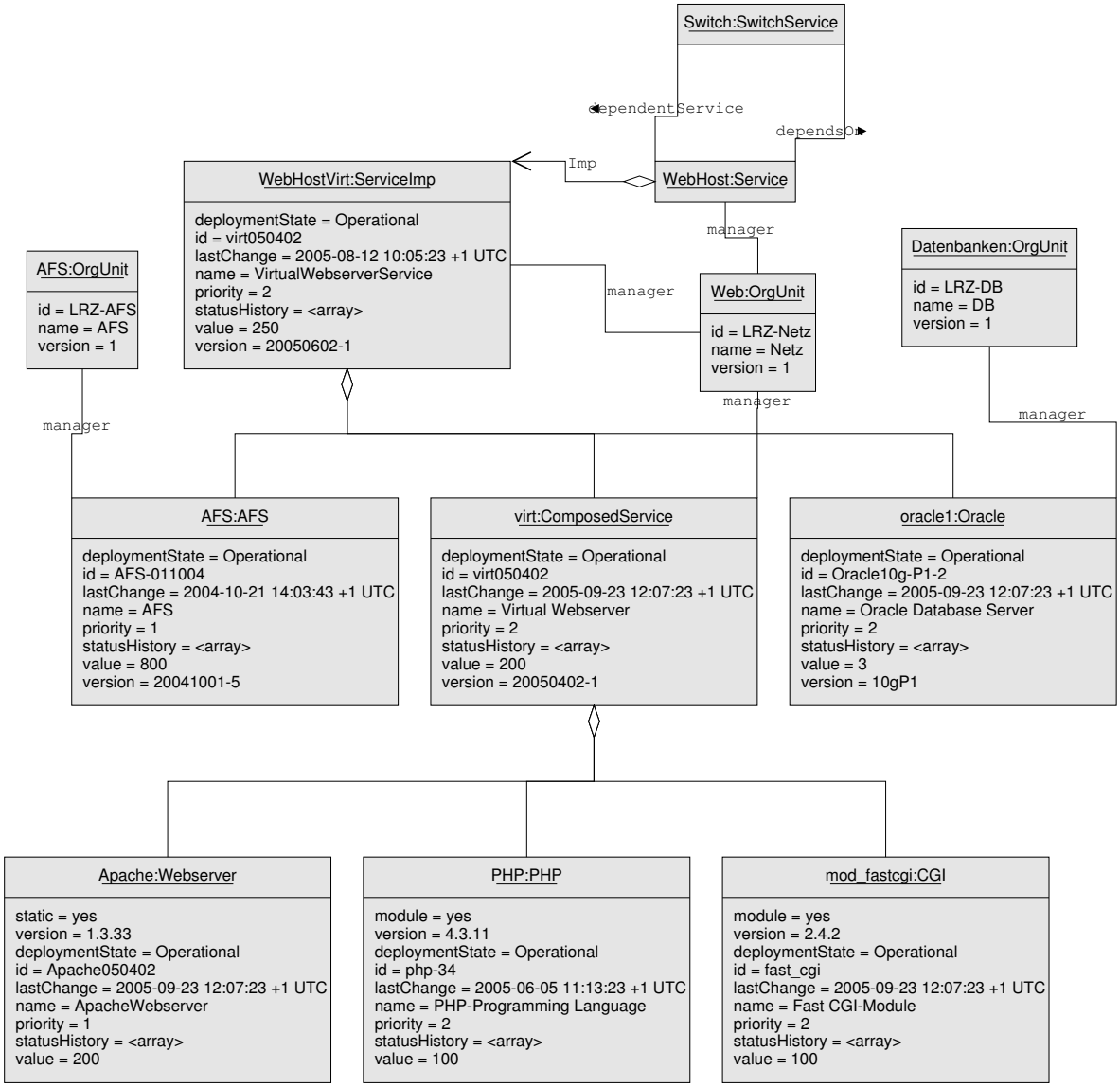

Abbildung 7.3: Objektdiagramm zum Webhostingdienst virt

In Abbildung 7.3 wird eine exemplarische Darstellung für den virtuellen Webserver des LRZ vorgestellt. Die verwendeten Attribute entsprechen keiner reellen Konfiguration. Um das Bild nicht allzu ausladend werden zu lassen, sind dabei die Teile weggelassen worden, welche in der Szenariobeschreibung nicht ausdrücklich erwähnt werden sind, wie das Accounting, der Ort und auch die spezifischen Konfigurationen.

Für die Erstellung des Diagramms wurde zunächst der Dienst, welcher als *WebHostVirt* bezeichnet werden soll, daraufhin untersucht, ob er überhaupt ein eigenständiger Dienst ist oder ob er nicht vielmehr nur eine Variation anderer Webhostingangebote darstellt. Nachdem bereits beschrieben worden ist, dass die Webgruppe des LRZ auch anders zusammengesetzte Webserver (*lrz, spez und ars*) anbietet, ist dies nicht der Fall. Somit ist dieser Dienst ganz klar ein Fall für das Bridge Pattern, dessen abstrakte Seite als “WebHostService” bezeichnet wurde und von der Funktionstätigkeit eines bestimmten Switches *switch* abhängig ist.

In einem nächsten Schritt wurde dann geschaut, ob *WebHostVirt* ein primitiver Dienst ist, oder ob er aus weiteren Diensten zusammengesetzt ist. Im vorliegenden Beispiel besteht der Dienst aus den weiteren Diensten *AFS* (einem AFSCluster), *oracle1* (einem Oracle Server) und dem Dämonen *virt*. Die beiden erstgenannten Dienste erweisen sich als primitive Dienste und sind somit nicht weiter zu beachten. Der zusammengesetzte Dienst wird dann entsprechend der Vorschrift des Composite Pattern modelliert.

Anders verhält es sich mit dem "Dämon" *virt*. Dieser ist aus weiteren Komponenten aufgebaut. Im vorliegenden Beispiel besteht er aus den Komponenten *Apache*, *PHP* und *mod_fastcgi*, welche selber wiederum primitive Dienste der Klassen *Webserver*, *PHP* und *CGI* sind. Auch hier werden die einzelnen primitiven Dienste mit Hilfe des Bridge Pattern zu *virt* verbunden.

Anschließend gilt es noch die Beziehungen der verschiedenen Klassen zu anderen Klassen zu untersuchen. Im vorgestellten Beispiel beschränkt sich dies auf die jeweiligen *manager*, also die für die Funktionsfähigkeit zuständigen Organisationseinheiten.

7.3 Examplarische Fehlerbearbeitung anhand des neuen Modells

Um die Vorteile des neuen Objektmodells darstellen zu können, wird nun ein ähnlicher Problemfall dargestellt werden, wie in 2.2.2. Als Ausgangspunkt sei die Modellierung des virtuellen Webservers, wie in Abbildung 7.3 gezeigt, gegeben. Der Fehlerfall ist an den in Kapitel 2.2.2 vorgestellten angelehnt.
Der Unterschied ist hier jedoch, dass nicht bekannt ist, dass das AFS-System die Störung aufweist. Das Symptom sei hier, dass die Darstellung von Webseiten nicht funktioniert.

Die Bestimmung der Ursache des Problems erfolgt auf eine andere Art und Weise als bisher und kann vom Arbeitsplatz eines Mitarbeiters des Helpdesk ausgeführt werden.
Mit Hilfe des Managementsystems wird zunächst einmal der Dienst, der das Webhosting der virtuellen Server *WebHostVirt* darstellt, mit Hilfe der Methode "reachable" auf Erreichbarkeit überprüft. Da *WebHostVirt* nicht in Ordnung ist, verläuft dieser Test nicht erfolgreich. Da die Funktionalität dieses Dienstes aber davon abhängt, dass der Switch-Service des Switches *Switch* erfolgreich erbracht wird, wird dieser als nächstes auf Funktionsfähigkeit überprüft. In unserem Fall liegt das Problem aber nicht an dem Switch, so dass der Test erfolgreich ist.
Als nächstes werden die Dienste überprüft, auf welchen der Webserver basiert.
Dafür wird der Datenbank-Server *oracle1* auf Funktionsfähigkeit überprüft. Dieser Check verläuft erfolgreich, ebenso bei dem Webserver *virt*.
Bei dem Versuch, den AFS-Dienst *AFS* zu erreichen, schlägt der Test jedoch fehl. Somit ist das Problem beim AFS zu suchen. Eine genauere Analyse ist anhand des hier dargestellten Szenarios nicht möglich, da im vorliegenden Beispiel der Dienst *AFS* aus keinen weiteren Komponenten besteht.

Da die Arbeitsgruppe "AFS" als Verwalter dieses Dienstes vermerkt ist, kann diese nun benachrichtigt werden, um das Problem zu beheben.

Wie zu sehen ist, kann mit Hilfe des neuen Modells die manuelle Suche nach der Ursache des Problems entfallen und von dem Managementsystem automatisch übernommen werden. Dabei können alle Abhängigkeiten auch von Personen überprüft werden, welche Laien bezüglich dieser Dienste sind. Mitarbeiter, deren Produkte reibungslos funktionieren, müssen nun nicht mehr zur Störungssuche herangezogen werden, da die fehlerhaften Komponenten direkt bestimmt werden können.

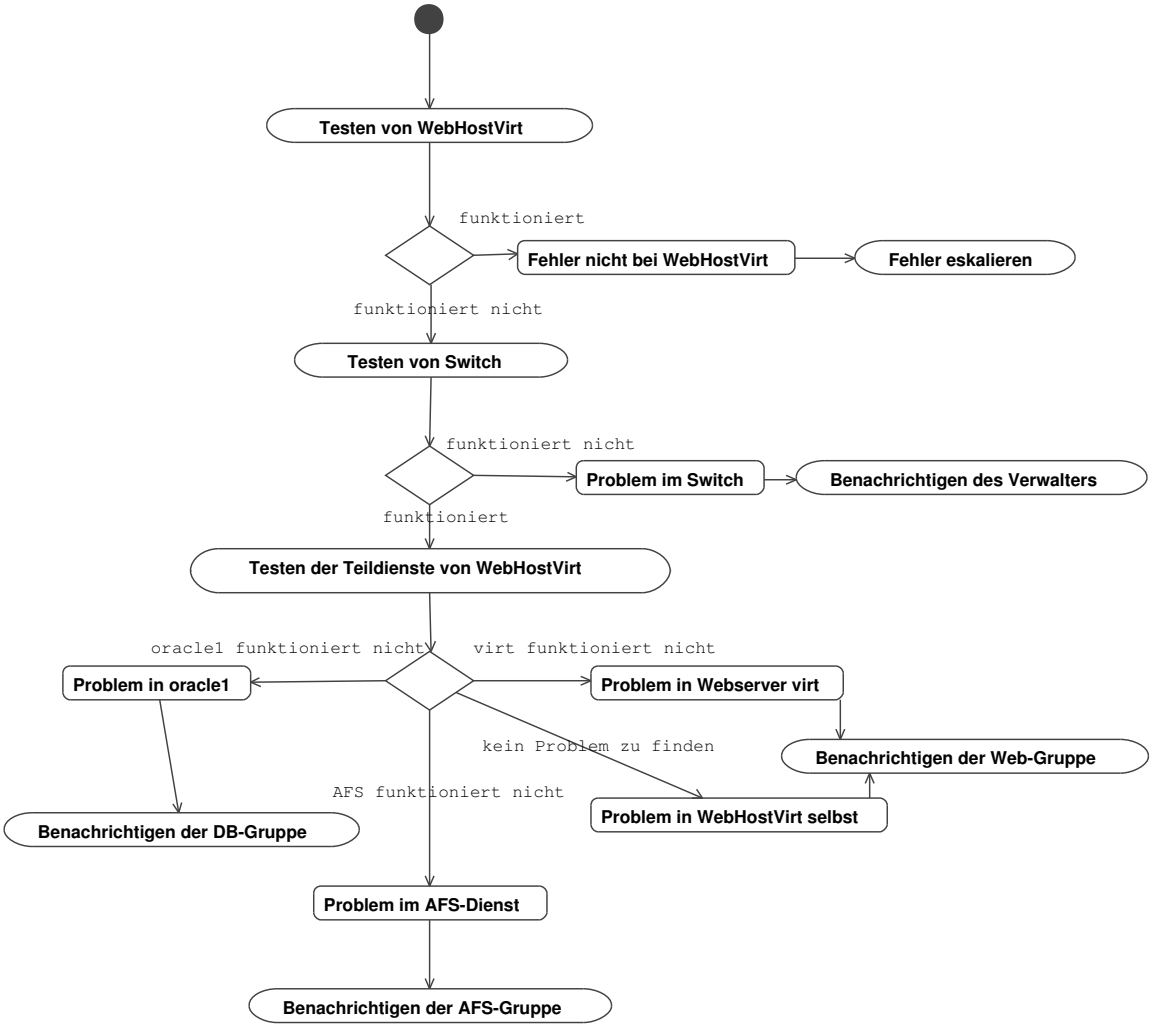

Abbildung 7.4: Ablaufdiagramm bei der Fehlerbearbeitung

Kapitel 8

Bewertung und Ausblick

In der vorliegenden Arbeit wurde zunächst auf der Basis von ITIL und OSI ein Objektkatalog entwickelt. Dies war notwendig, da beide Managementansätze lediglich eine Beschreibung der zu verwaltenden Objekte ohne Attribute anbieten. Deren Beschreibungen zum Servicemanagement, müssen erst mit großem Aufwand auf die jeweiligen realen Bedürfnisse angepasst werden. Der im Rahmen dieser Arbeit entwickelte Objektkatalog leistet genau dies.

8.1 Bewertung des entstandenen Objektkataloges

Mit der vorliegenden Arbeit wurde ein Objektkatalog geschaffen, welcher einen der großen Nachteile von ITIL und OSI behebt. Mit seiner Hilfe ist nun eine Sammlung von Attributen verfügbar, welche die Verwendung dieser beiden Ansätze mit wenig Aufwand ermöglicht. Der Vorteil ist hierbei, dass die Attribute aus den Managementaufgaben von ITIL und OSI selber, das heißt "top-down", entstanden sind und nicht auf Basis von aktuell anstehenden Problemen mit bestimmten, jeweils vorgegeben Komponenten, das heißt "bottom-up". Aufgrund dieser Vorgehensweise wird gewährleistet, dass alle Bereiche, welche von ITIL oder OSI angesprochen werden, auch tatsächlich bedacht worden sind.
Dadurch wird eine Verringerung der Abstraktion erreicht, welche die Anwendung anderer top-down Ansätze erschwert. Gerade bei ITIL, welches sich als neuer Standard bei der Verwaltung von Informationstechnologien zu etablieren scheint, kann hierdurch eine bessere Akzeptanz erreicht werden.

Bei Anwendung der hier vorgestellten Attribute wird zudem die Zusammenarbeit von Managementsystemen verschiedener Organisationen erleichtert, da die Nomenklatur gleich ist. Da es aber nicht ausreicht, nur in der gleichen Sprache zu sprechen, wurde im Objektkatalog auch großer Wert auf die Erstellung von Schnittstellen gelegt. Dabei wurden Design Pattern eingesetzt, um die Implementierung unabhängig vom Interface der Schnittstellen zu halten, was die Herstellerunabhängigkeit von verwalteten Diensten, aber auch von Überwachungsmodulen sicherstellt. Da Schnittstellen nicht nur zu anderen Systemen geschaffen worden sind, sondern auch innerhalb des Objektkataloges vorhanden sind, wird ein modularer Aufbau des daraus resultierenden Managementsystems unterstützt.

Bei Betrachtung des Objektkataloges (Abbildung 6.9) fällt auf, dass die Anzahl der dargestellten Attribute wesentlich geringer ist als bei CIM oder auch der Variablen bei den MIBs. CIM ist durch seine vielen Klassen unübersichtlich. Die MIBs des Internet Management sind dagegen nur Blätter am Rande einer Baumstruktur, deren Inhalte nicht in die Hauptstruktur mit eingebunden werden können.

Andererseits ist es jederzeit möglich, die Klassen aus CIM in den hier vorgestellten Objektkatalog zu überführen oder die MIBs in die Konfiguration eines Dienstes einzuhängen.

Der entwickelte Objektkatalog ist kompakt und seine Klassen und ihre Beziehungen zueinander können problemlos erkannt werden. Er ist andererseits ausführlich genug, dass die praktische Anwendung der jeweiligen Klassen ebenfalls schnell erkennbar und möglich ist.

Die Plattformunabhängigkeit wurde bei diesem Modell gewahrt, so dass es keine Auswirkung hat, ob nun Dienste der Firma Sun oder der Firma Microsoft unterstützt werden. Auch Dienste, welche von ganzen Clustern von Rechnern angeboten werden, können mit diesem Ansatz ohne weiteres modelliert und verwaltet werden.

Die einfache Anpassung bestehender Dienste an aktuelle Anforderungen wurde durch die Unterstützung einer modularen Dienstsicht ermöglicht. Wenn ein Modul eines Dienstes sich ändert, muss nun nicht mehr der ganze Dienst neu in die Verwaltung aufgenommen werden, sondern nur der Teil, welcher sich geändert hat. Ebenso verhält es sich bei der Konfiguration. Es können auch mehrere Varianten derselben Konfiguration/desselben Dienstes verwaltet werden.

Bei der Suche nach der Ursache von Störungen kann der hier vorgestellte Objektkatalog seine Stärken besonders gut ausspielen. Ihm ist nicht nur bekannt, welche Dienste existieren, sondern auch, von welchen anderen Diensten sie abhängen und aus welchen Diensten der fragliche Dienst besteht. Dadurch wird es einfach möglich, über automatisierte Systeme die Ursache einer Störung zu bestimmen und die für die gestörte Komponente Verantwortlichen zu informieren.

8.2 Ausblick

Der Fokus dieser Arbeit lag auf der Darstellung der Dienste. Dadurch wurden andere Teile des Managements ausgeklammert, wie zum Beispiel die des Service Level Managements oder auch des Accounting.
Die vorgestellten Attribute reichen aber aus, um zumindest ein grundlegendes SLA- oder Abrechnungsmanagement durchführen zu können. Die Anbindung eines entsprechenden Spezialsystems mit erweiterter Funktionalität wäre daher kein Problem. Die Darstellung der durch solch ein System zur Verfügung gestellten Attribute jedoch würde den Rahmen der vorliegenden Arbeit sprengen.

Im Allgemeinen liegt eine Schwierigkeit bei der organisationsübergreifenden Zusammenarbeit in der unterschiedlichen Bezeichnung der Komponenten. Dieses Problem tritt um so mehr in Erscheinung, je internationaler die Zusammenarbeit ausfällt und je unabhängiger die kooperierenden Organisationen voneinander sind. Ein typisches Beispiel hierfür sind die sogenannten Webservices. Um dieses Problem zu lösen, ist die Ontologiesprache “OWL-S” [OWL-S] geschaffen worden. Interessant wäre in diesem Zusammenhang eine Abbildungsvorschrift des hier vorgestellten Objektkataloges in die von OWL-S verwendete Syntax.

Als einzige der hier vorgestellten bekannten Ansätze liefern SID und CIM Attribute, welche die zu verwaltenden Objekte genauer beschreiben. Die Aufgabe, die dort vorgestellten Klassen mit ihren Attributen auf die in dieser Arbeit vorgestellten abzubilden, muss ais Zeit- und Kapazitätsgründen auf später verschoben werden werden, da zum Beispiel CIM eine große Menge verschiedenster Klassen beinhaltet. Die Abbildung einer Klasse ist relativ einfach, sobald die Bedeutung der Klassen in CIM verstanden wurde. Als Beispiel sei hier auf die im Rahmen der Erläuterung der Design Pattern erfolgte

Abbildung des in CIM 11 Klassen umfassenden Komplexes zu Personen und Organisationen mit Hilfe des Composite Pattern verwiesen.

Um die Anpassung der Schnittstellen für die verschiedenen Überwachungsmodule einfach zu gestalten, wäre eine allgemeine Sprache und ein allgemeines Modell zur Beschreibung der auszuführenden Methoden, welche die Funktionsbereitschaft prüfen, wünschenswert. Derartiges existiert in einem allgemeinen Ansatz bisher jedoch noch nicht.

Abschließend möchte ich anmerken, dass mir dieses Thema im Laufe der sukzessiven Vertiefung immer mehr Spaß gemacht hat. Allerdings habe ich die Vielfalt der Aspekte, welche auf das Design eines Objektkataloges Einfluss haben, anfänglich stark unterschätzt und konnte aus Zeitmangel leider viele interessante und wichtige Aspekte der Integration nicht ausreichend detailliert ausarbeiten.
Ob der hier entwickelte Objektkatalog in der Praxis die gestellten Anforderungen wirklich wie behauptet voll und ganz erfüllen wird, müsste mit einem Prototyp noch bewiesen werden.

Ich möchte mich an dieser Stelle bei Herrn Hahn im LRZ für den Einblick in die praktische Arbeit seiner Arbeitsgruppe noch einmal herzlich bedanken. Mein besonderer Dank gilt meinen Betreuern Martin Sailer und Michael Schiffers für die tatkräftige Unterstützung in Form von bereitwillig gegebenen Informationen, zur Verfügung gestellten Ressourcen und konstruktiver Kritik beim Erstellen dieser Arbeit.

Ich hoffe, dass das hier vorgestellte Konzept ganz oder teilweise in einem integrierten Ansatz umgesetzt werden kann. Es würde mich auch freuen, wenn ich in meinem jetzt beginnenden beruflichen Leben weiter in diesem Sachgebiet arbeiten könnte.

Anhang A

ITIL: The principal aspects of interfaces

(ITIL Infrastructure Management [ITIL 02a], p.31 f.)

- **Service Level Management (SLM)** is the process concerned with managing the quality and level of ICT service delivered to the business. The main areas of interest to D and P are:
  - the Service Catalogue
  - SLA and SLRs
  - service Plans, reviews and reports
  - the Continuous Service Improvement Programme (CSIP)
  - supplier contract and support OLA information, such as details of Service Level targets

  The D and P function will need to work closely with SLM to ensure that the infrastructure provides and continues to provide the quality of service documented within agreed SLAs and SLRs. The CSIP should be used to continually improve the levels of service delivered.
- **IT Service Continuity Management** is the Process for ensuring that the ICT service resilience and recovery is appropriate to the business impact and disruption caused by the loss of the service. It is one of the most valuable Service Management Processes and provides:
  - the IT Service Continuity Plan
  - Business Impact Analysis (BIA) information
  - risk analysis and Management information
- **Financial Management for IT Services** is the process of management and control of all financial resources within the ICT function. It principally supplies D and P with:
  - the financial plan
  - costs, budgets and charges
  - the cost of functions and processes
  - the cost of failures
- **Availability Management** is concerned with ensuring that the availability targets agreed within SLAs are met or exceeded. The main value to D and P is the:
  - Availability Plan
  - Availability measures and guidelines for new services
  - Availability reviews and reports
  - identification of weak or failing components within the ICT infrastructure
  - Component Failure Impact Analysis (CFIA) information
- **Capacity Management** produces plans for the capacity and performance of all services and

ICT components to support the existing and predicated service levels. In order to do this Capacity Management will provide:
  - the Capacity Plan
  - the Capacity Management Database (CDB)
  - infrastructure usage and trends
  - capacity reviews and reports
  - facilities for modelling and prototyping of new services and existing services and systems
  - the current workloads and schedules
- **Service Desk** provides a single point of contact for all customers and users on all aspects of the usage of ICT function. It is principally of value in supplying D and P with:
  - a valuable set of management information
  - a customer satisfaction feedback
- **Incident Management and Problem Management** provide integrated processes for the management of all incidents, problems and known errors associated with the ICT infrastructure and services. The processes supply:
  - valuable management information on incidents, problems and known errors
  - areas of proactive improvement
- **Configuration Management** provides a single process for the control of all ICT assets and configuration information, including:
  - the Configuration Management Database (CMDB)
  - Configuration Management plans
  - status information on all components of the ICT infrastructure
  - topology and schematic connectivity and relationship information
  - service and component impact information
- **Change Management** provides a single process for the control and management of all changes and provides:
  - the Forward Schedule of Change
  - details of all changes, change status and change plans
- **Release Management** is a single process for the control and management of all releases and provides:
  - Release plans, policies and contents
  - the Definitive Hardware Store (DHS)
  - the Definitive Software Library (DSL)
  - Release implementation plans and schedules

Anhang B

Attributsvorschläge CI nach ITIL

ITIL schlägt in [ITIL 00] auf S.164 tatsächlich einen Subset an Attributen vor, welche für das Konfigurationsmanagement gebraucht werden:

The following attributes are examples that could be used in the CMDB. Note that hardware CI types will have different attributes from software CI types.

Attribute	Description
CI Name	The unique name by which this type of CI is known.
Copy or Serial Number	The number that uniquely identifies the particular instances of this CI - for example, for software the copy number, for hardware the serial number.
Category	Classification of CI (e.g. hardware, software, documentation etc.).
Type	Description of CI type, amplifying 'category' information (e.g. hardware configuration, software package, hardware device or program module).
Model Number (hardware)	Model of CI (corresponding, for example, to supplier's model number e.g. Dell model xxx, PC/aa model yyy).
Warranty expiry date	Date when suppliers warranty expires for the CI.
Version Number	The version number of the CI.
Location	The Location of the CI, e.g. the library or media where the software CIs reside, the site/room where a service is located.
Owner responsible	The name and/or designation of the owner responsible for the CI.
Responsibility Date	Date the above owner became responsible for the CI.
Source/Supplier	The source of the CI, e.g. developed in-house, bought in from company xxxxx etc.
Licence	Licence number or reference to licence agreement.
Supply date	Date when CI was supplied to the organisation.
Accepted Date	Date when the CI was accepted by the organisation as satisfactorily tested.
Status (current)	The current status of the CI; e.g. under 'test', 'live','archived'.
Status (scheduled)	The next scheduled status of the CI (with the date or indication of the event that will trigger the status change).
Parent CI(s) relationships	The unique CI identifier(s) - name/copy/number/model/number/ of the 'parent(s)' of this CI.

Child CI(s) relationships	The unique CI identifier(s) of all 'children' of this CI.
Relationships	The relationships of the CI with all CIs other than 'parent' or 'child' (e.g. this CI 'uses' another CI, this CI 'is connected to another CI, this CI is 'resident on' another CI, this CI 'can access' another CI).
RFC Numbers	The identification numbers of all RFCs affecting this CI.
Change Numbers	The identification number of all Change records affecting this CI.
Problem Numbers	The identification number of all Problem records affecting this CI.
Incident Numbers	The identification number of all Incident records affecting this CI.
Comment	A comment field to be used for textual narrative; for example, to provide a description of how this version of the CI is different from previous version.

For RFCs, Change records, package Release records, etc., the names, copy numbers, model numbers and version numbers of CIs affected by the Change, and how they are affected, should be recorded in the CMDB. A reversion path, and the consequences of reversion, should also be recorded.

Anhang C

ITIL - Items to include in an RFC form

C.1 Hauptformular

Auf Seite 174 in [ITIL 00] listet ITIL folgende Punkte auf, welche durch ein Formular für ein "Request for change" (RFC), abgefragt werden sollten:

- RFC number (plus cross reference to Problem report number, where necessary)
- description and identity of item(s) to be changed (including CI identification(s) if Configuration Management System is in use)
- reason for Change
- effect of not implementing Change
- version of item to be changed
- name, location and telephone number of person proposing the change
- date that the Change was proposed.
- Change priority
- impact and resource assessment (which may be on separate forms where convenient)
- CAB[1] recommendations where appropriate (which may be held separately, with impact and resource assessments, where convenient)
- authorisation signature (could be electronic)
- authorisation date and time
- scheduled implementation (Release identification and/or date or time)
- location of Release/implementation plan
- details of Change builder/implementer
- back-out plan
- actual implementation date and time
- review date
- review results (including cross-reference to new RFC where necessary)
- risk assessment and management
- impact on business continuity and contingency plans
- status of RFC - i.e. 'logged','assessed', 'rejected', 'accepted', 'sleeping'

[1] Change Advisory Board

C.2 Impact and resource assessment

[ITIL 00] spricht auf Seite 185 des weiteren ein paar Punkte an, welche beim “impact and resource assessment” beachtet werden sollten:

- the impact the Change will make upon the Customer’s business operation
- the effect upon the infrastructure and Customer service, as defined in the SLA, and upon the capacity and performance, reliability and resilience, contingency plans, and security
- the impact on other services that run on the same infrastructure (or on software development projects)
- the impact on non-IT infrastructures within the organisation - for example, security, office services, transport, business - Costumer Help Desks
- the effect of not implementing the Change
- the IT, business and other resources required to implement the Change, covering the likely costs, the number and availability of people required, the elapsed time, and any new infrastructure elements required
- the current FSC[2] and PSA[3]
- additional ongoing resources required if the Change is implemented

[2] Forward Schedule of Change
[3] Projected Service Availability

Anhang D

ITIL - Change Management Tool - Anforderungen

In [ITIL 00], S. 199 finden sich eine Reihe von Anforderungen an ein Software Tool zur Verwaltung des Change Managements.

- RFCs and PRs stored upon the same database, in an easily accessible format
- the ability to identify the relationship between RFCs[1], PRs [2] and CIs[3]
- the capability to link RFCs to projects
- the means to identify easily the other CIs that will be impacted whenever a Change to any specific CI is proposed
- automatic production of requests for impact and resources assessment to the 'owners' of the impacted CIs
- the ability for all authorised personnel to submit RFCs from their own terminal or location
- the ability to 'progress' requests through the appropriate stages of authorisation and implementation and to maintain clear records of this progress
- the ability to allow Change Management staff, Change builders, testers, etc. to add text to Change records
- clear definition of back-out procedures should a change cause problems
- automatic warnings of any RFCs that exceed pre-defined time periods during any stage
- automatic prompting to carry out reviews of implemented Changes
- automatic generation of management and trend information relating to Changes
- the ability to build Changes
- automatic production of FSCs [4]
- a process/workflow feature

[1] Request for Change
[2] Problem Record
[3] Configuration Item
[4] FSC = Forward Schedule of Change

Anhang E

Attribute zu den Fragen aus ITIL

In diesem Anhang wird eine Reihe von Attributen vorgestellt, welche sich alleine aus den Anforderungen von ITIL heraus bestimmen lassen. Die hier vorgestellten Attribute sind jedoch lediglich in durch eine einfache Gruppierung der Managementaufgaben aus Kapitel 4.2 entstanden und nicht überarbeitet worden.

E.1 Hauptdatensatz

Attributsname	Managementaufgaben	Besonderheiten
name	Anhang B	Eindeutiger Name
id	ITIL-xxxiv, ITIL-li, ITIL-lxvi, Anhang B	Eindeutige ID
modelNumber	Anhang B	Modellnummer des Herstellers
sla	ITIL-iii, ITIL-vi, ITIL-xxv	
owner	ITIL-xxxiv, ITIL-liii, Anhang B	
lastOwnerChange	Anhang B	
status	ITIL-xxxvi, ITIL-lxvii, Anhang B	
expert	ITIL-xi (ITIL-E)	
relation	ITIL-xvi	
itemVersion	Tabelle 4.3	Version der Komponente
version	Tabelle 4.3, ITIL-l, Anhang B	Version des Datensatzes
supplier	Tabelle 4.3, Anhang B	
purchaseDate	Tabelle 4.3	
location	Tabelle 4.3, Anhang B	
category	Anhang B	Grobeinteilung der Dienstart
type	ITIL-xliii, Anhang B	
maturity	ITIL-xliv, ITIL-lii (ITIL-U), ITIL-liv	Stand im Lebenszyklus
maturityChange	ITIL-lv	
deliveryStatus	ITIL-lii(ITIL-U)	

Attributsname	Managementaufgaben	Besonderheiten
supplyDate	Anhang B	
AcceptedDate	Anhang B	
warrantyExpires	B	Ablauf der Garantie
connections	ITIL-xlv	physische Verbindungen
requiredBy	ITIL-xlv, Anhang B	benötigen andere diese Komponente?
childItems	ITIL-xl, ITIL-xli, ITIL-xlv(ITIL-N), ITIL-xlviii, Anhang B	
requirements	ITIL-xlviii, Anhang B	Nutzungsanforderungen
managable	ITIL-xlii	Bestimmte Komponenten sind nicht managebar
valid?	ITIL-lxii	Korrektheit der Daten
accessRights	ITIL-lxiv	
checkRegistred	ITIL-xxxv, ITIL-lxxi	Methode für den Check auf nicht registrierte Komponenten
licences	ITIL-lvii , ITIL-lviii, Anhang B	Lizenzen
rfcs	Tabelle 4.3, ITIL-xlv(ITIL-Q), Anhang B	
changes	ITIL-xxi, ITIL-xxvi, Tabelle 4.3, ITIL-lix, Anhang B	vorgenommene Änderungen (Tabelle E.4)
documentation	ITIL-xx, ITIL-lvi	Dokumentation zu der Komponente
incidents	ITIL-lxix, Tabelle 4.3, ITIL-xlv(ITIL-R), ITIL-lix, Anhang B	Dokumentation von Störungen (Tabelle E.2)
problems	ITIL-lxix, Tabelle 4.3, ITIL-xlv(ITIL-S), ITIL-lix, Anhang B	Dokumentation von Problemen (siehe E.2)

Tabelle E.1: Attribute zu den Managementaufgaben aus ITIL (Core)

E.2 Incident Record

Attributsname	Managementaufgaben	Besonderheiten
referenceNo	Tabelle 4.2	Eindeutige Referenznummer
classification	Tabelle 4.2	Klassifizierung der Störung
timestamp	Tabelle 4.2, ITIL-xxxii	Zeitpunkt der Aufnahme der Störung
recordby	Tabelle 4.2	Name der aufnehmenden Person/Gruppe
affectedUser	Tabelle 4.2, ITIL-xxxi	Meldende/betroffene User mit ihren Daten (Tabelle E.3)
callbackmethod	Tabelle 4.2	gewünschte Möglichkeit der Kontaktaufnahme

Attributsname	Managementaufgaben	Besonderheiten
symptomdescription	Tabelle 4.2	Beschreibung der Symptome
category	Tabelle 4.2, ITIL-xv, ITIL-xxvii, ITIL-xxix	Haupt- und Unterkategorie
impact	Tabelle 4.2, ITIL-viii, ITIL-xvii	
impactDeterminationTime	ITIL-xviii	Zeitpunkt, zu dem die Auswirkungen bestimmt wurden
urgency	Tabelle 4.2, ITIL-ix	
priority	Tabelle 4.2	
status	Tabelle 4.2	
affecteditem	Tabelle 4.2, ITIL-vii, ITIL-xxx, Tabelle 4.3	betroffene Komponenten
source	ITIL-v	wirklich fehlerhafte Komponente
allocatedTo	Tabelle 4.2	Person/Gruppe, der die Bearbeitung zugeordnet ist
estimatedCost	ITIL-xxiii	
relatedError	Tabelle 4.2	siehe relatedProblem
resolutionTimestamp	Tabelle 4.2	
incidentCost	ITIL-xxxii	
closureCategory	Tabelle 4.2	
closureTimestamp	Tabelle 4.2, ITIL-xxxii, ITIL-lxx	
recurring	ITIL-xxviii	Tritt das Problem wiederholt auf?

Tabelle E.2: Attribute zu den Managementaufgaben aus ITIL (IncidentRecord)

E.3 Benutzer

Attributsname	Managementaufgaben	Besonderheiten
name	Tabelle 4.2	
department	Tabelle 4.2	
location	Tabelle 4.2	
phone	Tabelle 4.2	
mail		

Tabelle E.3: Attribute zu den Managementaufgaben aus ITIL (User)

E.4 Change Record

Attributsname	Managementaufgaben	Besonderheiten
RFC		Dazugehöriger RFC
changeDate	ITIL-lxxvii	Zeitpunkt, zu dem die Änderung aktiv geschaltet wird
affectedItems	Tabelle 4.3, ITIL-lxxiv, ITIL-lxxix	
changeOpen?	Tabelle 4.3	
authorizedBy	ITIL-lxviii, ITIL-lxxii	
scheduledStatus	Anhang B	

Tabelle E.4: Attribute zu den Managementaufgaben aus ITIL (ChangeRecord)

E.5 RFC-Formular

Attributsname	Managementaufgaben	Besonderheiten
ID		Eindeutige Referenznummer
items		Liste zu ändernder Komponenten
itemVersion		Version der zu ändernden Komponente
proposingUser		Benutzer, der die Änderung vorschlägt (Tabelle E.3)
proposalDate		Vorschlagsdatum
changeTime		
status		

Tabelle E.5: Attribute zu den Managementaufgaben aus ITIL (RFC Formular)

Anhang F

Attribute zu den Fragen aus OSI

Die hier vorgestellten Attribute lassen sich aus den Anforderungen von OSI ableiten. Die Attribute sind in einer einfachen Form vorgestellt und nicht weiter überarbeitet worden.

F.1 Hauptdatensatz

Attributsname	Managementaufgaben	Besonderheiten
id	OSI-i, OSI-xv	Eindeutige ID
owner	OSI-ix	"Besitzer" der Komponente
version	OSI-xlii	Versionsnummer der Komponente
status	OSI-ii, OSI-xxxvi	"Aktiv", "In Wartung", "Historisch"
setStatus	OSI-xxiii	Methode
location	OSI-xxx, OSI-xxxi	Ort, an dem die Komponente aufzufinden ist.
locatedOn	OSI-xxxii	Ist Teilkomponente von Komponente x. Im Falle von logischen Diensten auch die physikalische Komponente.
reachable	OSI-iii	Überprüft die Erreichbarkeit des Dienstes vom Management aus.
unreachableSub	OSI-viii	Ist eine Methode, die ausgibt, welche Subkomponente nicht erreicht werden konnte.
routeToClient	OSI-xxvi	Methode
parametercheck	OSI-lx	Methode, überprüft aktive und Soll-Parameter.
components	OSI-xxxiii	
dependencies	OSI-xxvii, OSI-xxxiv	
configuration	OSI-xli	Methode, gibt aktive Konfiguration zurück
superparamter	OSI-xxxix	Methode, holt ihre Werte aus spezieller Datenbank

Attributsname	Managementaufgaben	Besonderheiten
alarm	OSI-iv	Liste der Alarmmeldungen (Tabelle F.2)
errors	OSI-x, OSI-xxviii	Liste von Fehlern (Tabelle F.3)
interfaces	OSI-xvi	Liste von Interfaces (Tabelle F.4)
parameter	OSI-xix, OSI-xxxvii	Liste von Parametern (Tabelle F.5)
licence	OSI-xliii	Liste von Lizenzen (Tabelle F.6)
warranty	OSI-xliv	Garantie (Tabelle F.7)
sla		Liste von SLAs, die den Dienst betreffen (Tabelle F.8)
user	OSI-xlv	Liste der (berechtigten) Nutzer F.9
statistics	OSI-xlix	Liste mit historischen/verarbeiteten Parameterwerten

Tabelle F.1: Attribute zu den Managementaufgaben aus OSI (Überblick)

F.2 Alarme

Attributsname	Managementaufgaben	Besonderheiten
source	OSI-v	
acknowledged?	OSI-vi	
reason	OSI-vii	
fix	OSI-xi	
setParameter	OSI-xxiv	Methode
automated	OSI-xxv	

Tabelle F.2: Attribute zu den Managementaufgaben aus OSI (Alarm)

F.3 Fehler

Attributsname	Managementaufgaben	Besonderheiten
description		Klartextbeschreibung des Problems
timestamp	OSI-xxix	
quickfix	OSI-xii	
fix	OSI-xiii	
bestFix	OSI-xiv	

Tabelle F.3: Attribute zu den Managementaufgaben aus OSI (Fehler)

F.4 Interfaces

Attributsname	Managementaufgaben	Besonderheiten
sollParameter	OSI-xvii	
inSoll	OSI-xviii	Methode
connectsTo	OSI-xxxv	

Tabelle F.4: Attribute zu den Managementaufgaben aus OSI (Interfaces)

F.5 Parameter

Attributsname	Managementaufgaben	Besonderheiten
sollParameter	OSI-xx	
inSoll	OSI-xxi	Methode
setSoll	OSI-xxii	Methode
activeAt	OSI-xxxviii	
valid	OSI-xl	Methode
accountable	OSI-xlvi	
accountingUnitCount	OSI-xlvi	
accountingUnit	OSI-xlvii	Nur bei Abrechnungen benötigt

Tabelle F.5: Attribute zu den Managementaufgaben aus OSI (Parameter)

F.6 Lizenzen

Attributsname	Managementaufgaben	Besonderheiten
type	OSI-xliii(OSI-B)	
number	OSI-xliii(OSI-A)	
owner	OSI-xliii(OSI-D)	
expiry	OSI-xliii(OSI-C)	

Tabelle F.6: Attribute zu den Managementaufgaben aus OSI (Licence)

F.7 Garantie

Attributsname	Managementaufgaben	Besonderheiten
kind	OSI-xliv(OSI-E)	
expiry	OSI-xliv(OSI-F)	
limitations	OSI-xliv(OSI-G)	

Tabelle F.7: Attribute zu den Managementaufgaben aus OSI (Warranty)

F.8 SLA

Attributsname	Managementaufgaben	Besonderheiten
user	OSI-xlv	Liste
quota	OSI-xlviii	
parameter	OSI-xlvi	Parameter, die abgerechnet werden.
modi	OSI-l	
contractID	OSI-li	
services	OSI-li	
validity	OSI-li	

Tabelle F.8: Attribute zu den Managementaufgaben aus OSI (SLA)

F.9 Benutzer

Attributsname	Managementaufgaben	Besonderheiten
Name	OSI-li	
sla	OSI-li	Service Level Agreements (Tabelle F.8)
location	OSI-li	

Tabelle F.9: Attribute zu den Managementaufgaben aus OSI (User)

Glossar

AFS Andrew File System

AppleTalk Netzwerkprotokoll von Apple

ARPA (U.S.) Advanced Research Projects Agency

CCTA Central Computer and Telecommunications Agency, seit 2000 OGC

CI Configuration Item

CIM Common Information Model

CMDB Configuration Management Database

CMIP Common Management Information Protocol

CMIS Common Management Information Services

CobiT Control Objectives for Information and related Technology

DBMS DatenBankManagementSystem

DNS Domain Name System

DOS Denial of Service

DSO Dynamic Shared Object

IAB Internet Architecture Board

IANA Internet Assigned Numbers Authority

IEEE Institute of Electrical and Electronics Engineers, Inc.

IETF Internet Engineering Task Force

IN IN - Klasse des DNS-Dienstes.

IRTF Internet Research Task Force

ISO International Organization for Standardization

ITIL IT-Infrastructure Library

IUK-System Informations- und Kommunikationssystem

LRZ Leibniz RechenZentrum

mDNS Multicast DNS

MIB Management Information Base

MNM-Team Munich Network Management Team

MO Managed Object

MWN Münchener WissenschaftsNetz

NAS Network Attached Storage

NFS Network File Service

NGOSS Next Generation Operations Support Systems

ODBC Open DataBase Connectivity

OGC Office of Government Commerce (eine Unterabteilung von HM Treasury der Englischen Regierung)

OSI Open Systems Interconnection (ISO)

QoS Quality of Service

RFC Request for Change (ITIL)

RFC Request for Comments (Internet)

SID Shared Information/Data

SLA Service Level Agreement

SNIA Storage Network Industry Association

SNMP Simple Network Management Protocol

SRV SRV - Ressource Record beim DNS-Based ServiceDiscovery

VPN Virtual Private Networks

Literaturverzeichnis

[Brad 96] BRADNER, S.: *RFC 2026: The Internet Standards Process – Revision 3*. RFC, IETF, Oktober 1996, ftp://ftp.isi.edu/in-notes/rfc2026.txt .

[CFSD 90] CASE, J., M. FEDOR, M. SCHOFFSTALL und J. DAVIN: *A Simple Network Management Protocol (SNMP)*. STD, IETF, 1990, ftp://ftp.rfc-editor.org/in-notes/std/std15.txt .

[CIM STD] DMTF SYSTEM AND DEVICES WORKING GROUP: *Common Information Model (CIM) Standards*. Technischer Bericht, DMTF, 2005, http://www.dmtf.org/standards/cim/ .

[CMRW 93] CASE, J., K. MCCLOGHRIE, M. ROSE und S. WALDBUSSER: *RFC 1441: Introduction to version 2 of the Internet-standard Network Management Framework*. RFC, IETF, April 1993, ftp://ftp.isi.edu/in-notes/rfc1441.txt .

[CMRW 96] CASE, J., K. MCCLOGHRIE, M. ROSE und S. WALDBUSSER: *RFC 1907: Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)*. RFC, IETF, Januar 1996, ftp://ftp.isi.edu/in-notes/rfc1907.txt .

[DMTF 03] DMTF: *CIM Core Specification v 2.7*. Distributed Management Task Force, April 2003, http://www.dmtf.org/standards/documents/CIM/CIM_Schema27/CIM_Core27-Final.pdf .

[Fowl 02] FOWLER, MARTIN: *Patterns of Enterprise Application Architecture*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[GaMc 93] GALVIN, J. und K. MCCLOGHRIE: *RFC 1445: Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2)*. RFC, IETF, April 1993, ftp://ftp.isi.edu/in-notes/rfc1445.txt .

[GHHK 01a] GARSCHHAMMER, M., R. HAUCK, H.-G. HEGERING, B. KEMPTER, I. RADISIC, H. RÖLLE, H. SCHMIDT, M. LANGER und M. NERB: *Towards generi Service Management Concepts - A Service Model Based Approach*. In: *Integrated Network Management VII (IM 2001)*, Seiten 719–732.

[GHJV 95] GAMMA, E., R. HELM, R. JOHNSON und J. VLISSIDES: *Design Patterns — Elements of Reusable Object–Oriented Software*. Addison-Wesley Professional Computing Series. Addison-Wesley, 1995. ISBN 0-201-63361-2.

[GVE 00] GULBRANDSEN, A., P. VIXIE und L. ESIBOV: *RFC 2782: A DNS RR for specifying the location of services (DNS SRV)*. RFC, IETF, Februar 2000, ftp://ftp.isi.edu/in-notes/rfc2782.txt .

[Hahn 05] SAILER, BITTERICH: *Gesprächsprotokoll mit Herrn Hahn, LRZ*, 9. Mai 2005.

[HAN 99] HEGERING, H.-G., S. ABECK und B. NEUMAIR: *Integrated Management of Networked Systems – Concepts, Architectures and their Operational Application*. Morgan Kaufmann Publishers, ISBN 1-55860-571-1, 1999.

[Heil 97] HEILER, K.: *Eine Methodik zur Modellierung von Konfigurationsvorgängen für Szenarien im Netz- und Systemmanagement*. Doktorarbeit, Technische Universität München, 1997.

[HillGr] THE HILLSIDE GROUP: *Patterns Library*, http://hillside.net/patterns/ .

[HKS 99] HAZEWINKEL, H., C. KALBFLEISCH und J. SCHOENWAELDER: *RFC 2594: Definitions of Managed Objects for WWW Services*. RFC, IETF, Mai 1999, ftp://ftp.isi.edu/in-notes/rfc2594.txt .

[HP OVO] *HP - Operations for UNIX*, http://www.managementsoftware.hp.com/products/ovoux/ .

[HPW 02] HARRINGTON, D., R. PRESUHN und B. WIJNEN: *An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks*. STD, IETF, 2002, ftp://ftp.rfc-editor.org/in-notes/std/std62.txt .

[IM 01] PAVLOU, G., N. ANEROUSIS und A. LIOTTA (Herausgeber): *Integrated Network Management VII (IM 2001)*, Seattle, Washington, USA, Mai 2001. IEEE Publishing.

[ISO 10165-4] *Information Technology – Open Systems Interconnection – Structure of Management Information – Part 4: Guidelines for the Definition of Managed Objects*. IS 10165-4, International Organization for Standardization and International Electrotechnical Committee, 1992.

[ITIL 00] OFFICE OF GOVERNMENT COMMERCE (OGC) (Herausgeber): *Service Support*. IT Infrastructure Library (ITIL). The Stationary Office, Norwich, UK, 2000.

[ITIL 01] OFFICE OF GOVERNMENT COMMERCE (OGC) (Herausgeber): *Service Delivery*. IT Infrastructure Library (ITIL). The Stationary Office, Norwich, UK, 2001.

[ITIL 02a] OFFICE OF GOVERNMENT COMMERCE (OGC) (Herausgeber): *ICT Infrastructure Management*. IT Infrastructure Library (ITIL). The Stationary Office, Norwich, UK, 2002.

[ITIL 02b] OFFICE OF GOVERNMENT COMMERCE (OGC) (Herausgeber): *Application Management*. IT Infrastructure Library (ITIL). The Stationary Office, Norwich, UK, 2002.

[ITIL 02c] OFFICE OF GOVERNMENT COMMERCE (OGC) (Herausgeber): *Planning to Implement Service Management*. IT Infrastructure Library (ITIL). The Stationary Office, Norwich, UK, 2002.

[ITIL 03] OFFICE OF GOVERNMENT COMMERCE (OGC) (Herausgeber): *The Business Perspective*. IT Infrastructure Library (ITIL). The Stationary Office, Norwich, UK, 2003. (not yet published).

[ITIL 99] OFFICE OF GOVERNMENT COMMERCE (OGC) (Herausgeber): *Security Management*. IT Infrastructure Library (ITIL). The Stationary Office, Norwich, UK, 1999.

[Jäge 05] JÄGER, J.: *Entwicklung eines Toolkonzeptes für die Unterstützung der ITIL Service Support Prozesse*. Diplomarbeit, Technische Universität München, Januar 2005, http://wwwmnmteam.informatik.uni-muenchen.de/php-bin/pub/show_pub.php?key=jaeg05 .

[Kell 98] KELLER, ALEXANDER: *CORBA-basiertes Enterprise Management: Interoperabilität und Managementinstrumentierung verteilter kooperativer Managementsysteme in heterogener Umgebung*. Doktorarbeit, Technische Universität München, Dezember 1998.

[KKPS 99] KALBFLEISCH, C., C. KRUPCZAK, R. PRESUHN und J. SAPERIA: *RFC 2564: Application Management MIB*. RFC, IETF, Mai 1999, ftp://ftp.isi.edu/in-notes/rfc2564.txt .

[Lang 01] LANGER, MICHAEL: *Konzeption und Anwendung einer Customer Service Management Architektur*. Doktorarbeit, Technische Universität München, März 2001.

[LRZ-AFS] LRZ: *Webseite des LRZ zur Datenhaltung via AFS*. Webseite, http://www.lrz-muenchen.de/services/datenhaltung/afs/ .

[MaVi 96] MANNING, B. und P. VIXIE: *RFC 2010: Operational Criteria for Root Name Servers*. RFC, IETF, Oktober 1996, ftp://ftp.isi.edu/in-notes/rfc2010.txt .

[McKa 00] MCCLOGHRIE, K. und F. KASTENHOLZ: *RFC 2863: The Interfaces Group MIB*. RFC, IETF, Juni 2000, ftp://ftp.isi.edu/in-notes/rfc2863.txt .

[Neum 93] NEUMAIR, BERNHARD: *Objektorientierte Modellierung von Kommunikationsressourcen für ein integriertes Performance Management*. Doktorarbeit, Technische Universität München, 1993.

[NGOSS] TELEMANAGEMENTFORUM: *NGOSS Overview*, http://www.tmforum.org/browse.asp?catID=1912 .

[OWL-S] *Semantic Webservices mit DAML - Webpage*, http://www.daml.org/services/owl-s/ .

[PoCr 71] POSTEL, J.B. und S.D. CROCKER: *RFC 104: Link 191*. RFC, IETF, Februar 1971, ftp://ftp.isi.edu/in-notes/rfc104.txt .

[Pres 02] PRESUHN, R.: *RFC 3418: Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)*. RFC, IETF, 2002, ftp://ftp.rfc-editor.org/in-notes/rfc3418.txt .

[SID] TELEMANAGEMENTFORUM: *SID Overview Page*, http://www.tmforum.org/browse.asp?catID=2008 .

[SID-4SO03] TELEMANAGEMENT FORUM: *Shared Information/Data (SID) Model, Addendum 4SO - Service Overview Business Entity Definitions*, July 2003. Members Evaluation Version 2.0.

[Slo 94] SLOMAN, M. (Herausgeber): *Network and Distributed Systems Management*. Addison Wesley, 1994.

[TMF HP] FORUM, TELEMANAGEMENT: *Homepage des TeleManagement Forum*, http://www.tmforum.org/ .

[Var 94] VARLEY, B.: *User Administration and Accounting*, Seiten 381–402. In: *Var 94* [Slo 94], 1994.

Index