UNIVERSITÄT LEIPZIG
FAKULTÄT FÜR MATHEMATIK UND INFORMATIK

Diplomarbeit

Nutzung der Daten von operativen Informationsmanagement-Werkzeugen für 3LGM²-Modelle am Beispiel von Nagios

vorgelegt von

THOMAS TROMMER
geb. am 16.09.1982

bearbeitet am
Institut für Medizinische Informatik, Statistik und Epidemiologie
der Medizinischen Fakultät
bei Prof. Dr. Alfred Winter
Betreuer: Alexander Strübing
Leipzig, Januar 2007

Danksagung

Für die Überlassung des Themas möchte ich Herrn Prof. Dr. Alfred Winter danken, der trotz vieler anderer Aufgaben immer einen Termin für mich gefunden hat. Meinem Betreuer Herrn Alexander Strübing danke ich für die anregenden Diskussionen und Hilfen sowie Herrn Dr. Thomas Wendt für sein Engagement.

Meiner Verlobten Eva bin ich dafür dankbar, dass Sie mir in allen Phasen der Diplomarbeit beistand, egal wie weit Sie gerade weg war.

Besonderer Dank gilt meiner Mutter sowie meinen Großeltern, die mich in jeder Hinsicht vor und während meines Studiums unterstützt haben.

Herr Dr. Michael Raus vom Evangelischen Krankenhaus Hamm und Herr René Meinhold vom IMISE haben mit realen Nagioskonfigurationen entscheidend zur Verbesserung dieser Arbeit beigetragen.

Zusammenfassung

3LGM²-Modelle sind eine wichtige Informationsquelle für das strategische Informationsmanagement. Der Nutzen eines 3LGM²-Modells wird dabei um so größer, je genauer es ein Informationssystem abbildet. Das exakte Modellieren eines Informationssystems ist allerdings sehr aufwendig, weswegen zur Minderung des Arbeitsaufwandes teilweise nur grob modelliert wird.

Verschiedene Informationsmanagement-Werkzeuge enthalten Daten über Hard- und Softwarebestände eines Informationssytems, die genauer sein können als die des 3LGM²-Modells. Einige Klassen dieser Informationsmanagement-Werkzeuge wie das Computer Aided Facility Management, das Management verteilter Informationssysteme und IT Infrastructure Library sowie einzelne Softwareprodukte, die Vertreter dieser Klassen sind, werden hinsichtlich ihrer Eignung als Datenquelle für 3LGM²-Modelle untersucht. Orientierend am 3LGM²-Metamodell und bestehender 3LGM²-Modelle werden Anforderungen für eine Datenintegration aufgestellt, um zusammen mit den theoretischen Grundlagen der Datenintegration für den Entwurf einer Referenzarchitektur zu dienen.

Mit dem Netzwerk Monitoring System Nagios wird anschließend diese Referenzarchitektur an einem Produkt realisiert. Dafür werden die nötigen Grundlagen zu Nagios geschaffen und reale Datensätze analysiert. Die dabei auftretenden Probleme und Lösungen werden beschrieben. Das Ergebnis ist eine Menge von Modulen, die auf Nagioskonfigurationen angewendet werden können, um Teile von 3LGM²-Modellen abzuleiten.

Inhaltsübersicht

1 Einleitung 1

2 Aufbau 5

I Grundlagen 7

3 Das 3LGM²-Metamodell 9

4 Datenintegration 15

5 Ansatzpunkte zur Datengewinnung 25

II Rahmenbedingungen für die Datenintegration 37

6 Datengewinnung aus Softwareprodukten 39

7 Hard- und Softwarebestände in 3LGM²-Modellen 45

8 Anforderungen an eine Datenintegration 51

9 Referenzarchitektur für die Datenintegration mit dem 3LGM²-Baukasten 55

III Datenintegration mit Nagios 59

10 Nagios Grundlagen 61

11 Beispiel einer Nagioskonfiguration 79

12 Umsetzung der Referenzarchitektur für Nagios und 3LGM²-Baukasten 85

13 Nutzen der 3LGM²-Datenintegration 109

Z Abschluss 113

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Gegenstand und Motivation . . . 1
1.1.1 Gegenstand . . . 1
1.1.2 Problematik . . . 2
1.1.3 Motivation . . . 2
1.2 Problemstellung . . . 3
1.3 Zielstellung . . . 3
1.4 Fragestellung . . . 3

2 Aufbau **5**

I Grundlagen **7**

3 Das 3LGM²-Metamodell **9**
3.1 Die Fachliche Ebene . . . 9
3.2 Die Logische Werkzeugebene . . . 10
3.3 Die Physische Werkzeugebene . . . 11
3.4 Interebenenbeziehungen . . . 12
3.5 Modellieren mit dem 3LGM²-Baukasten . . . 13
3.6 Zusammenfassung . . . 13

4 Datenintegration **15**
4.1 Integrationsziele . . . 15
4.2 Physische Integration . . . 16
4.3 Definition der Datenintegration . . . 16
4.4 Abgrenzung gegenüber Importfunktionalitäten . . . 17
4.5 Analogie zu Datenbanksystemen . . . 17
4.6 Der Datenintegrationsprozess . . . 18
4.6.1 Integration der Datenstrukturen . . . 19
4.6.2 Integration der Datenbestände . . . 22
4.7 Automatisierung der Datenintegration . . . 24
4.8 Zusammenfassung . . . 24

5 Ansatzpunkte zur Datengewinnung **25**
5.1 Computer Aided Facility Management . . . 25
5.1.1 Facility Management . . . 26
5.1.2 CAFM- und Buchhaltungssysteme . . . 28
5.1.3 Zusammenfassung CAFM . . . 28
5.2 Management verteilter Informationssysteme . . . 28
5.2.1 Managementfunktionen . . . 29
5.2.2 Managementobjekte . . . 29

5.2.3 Standards für Informationsmodelle . . . 30
5.2.4 Managementarchitekturen . . . 30
5.3 IT Infrastructure Library . . . 33
5.4 Software Asset Management . . . 35
5.5 Zusammenfassung . . . 36

II Rahmenbedingungen für die Datenintegration 37

6 Datengewinnung aus Softwareprodukten 39
6.1 DX-Union Asset Assistant . . . 39
6.2 IBM Tivoli NetView . . . 41
6.2.1 Tivoli Data Warehouse . . . 42
6.2.2 Tivoli Enterprise Console . . . 42
6.3 HP OpenView . . . 42
6.4 OpenNMS . . . 42
6.5 Topologieerkennung . . . 43
6.6 Zusammenfassung . . . 44

7 Hard- und Softwarebestände in 3LGM²-Modellen 45
7.1 Für die Abbildung von Hard- und Softwarebeständen relevante 3LGM²-Metamodellklassen . . . 45
7.2 Verwendung der HSKs in 3LGM²-Modellen . . . 47
7.3 Zusammenfassung . . . 49

8 Anforderungen an eine Datenintegration 51
8.1 Funktionelle Anforderungen . . . 51
8.2 Nicht funktionelle Anforderungen . . . 52
8.3 Zusammenfassung . . . 54

9 Referenzarchitektur für die Datenintegration mit dem 3LGM²-Baukasten 55
9.1 Lokale und globale Datenstrukturen . . . 55
9.2 Lokale und globale Datenbestände . . . 56
9.3 Bindung der Integrationskomponente . . . 57
9.4 Alternativen zur Referenzarchitektur . . . 57
9.5 Zusammenfassung . . . 58

III Datenintegration mit Nagios 59

10 Nagios Grundlagen 61
10.1 Motivation . . . 61
10.2 Funktionsumfang . . . 62
10.3 Funktionsweise . . . 62
10.4 Konfiguration . . . 62
10.5 Inhalt der Konfiguration . . . 63
10.5.1 Terminologie für Nagios . . . 63
10.5.2 Nagiosobjekte und deren Informationsgehalt . . . 65
10.5.3 Vergleich mit den HSKs . . . 70
10.5.4 Vererbungsbeziehungen durch Nagiostemplates . . . 71

10.5.5 Reguläre Ausdrücke in Nagios . . . 71
10.5.6 Interne und externe Relationen . . . 73
10.5.7 Interpretation interner und externer Relationen . . . 74
10.6 Zusammenfassung . . . 78

11 Beispiel einer Nagioskonfiguration 79
11.1 Analysewerkzeug . . . 79
11.2 Umfang und Erstellung . . . 79
11.3 Konfigurationsinhalt . . . 81
11.4 Zusammenfassung . . . 83

12 Umsetzung der Referenzarchitektur für Nagios und 3LGM²-Baukasten 85
12.1 Vorüberlegungen . . . 86
12.1.1 Ziel . . . 86
12.1.2 Informationsbedarf und Datenmodell . . . 86
12.1.3 Modellierung der lokalen Datenstruktur des 3LGM²-Baukastens . . . 87
12.1.4 Modellierung der lokalen Datenstruktur von Nagios . . . 87
12.1.5 Aufbau der globalen Datenstruktur . . . 87
12.1.6 Erstellen des globalen Datenbestandes . . . 87
12.1.7 Physische Integration . . . 91
12.2 Module zum Finden von Zuordnungen . . . 91
12.2.1 Modul Rechnernetz . . . 91
12.2.2 Modul Physischer Datenverarbeitungsbaustein . . . 92
12.2.3 Modul Hostgroup . . . 93
12.2.4 Modul Servicegroup . . . 94
12.2.5 Modul Bausteintyp . . . 95
12.2.6 Modul Contactgroup/Timeperiod . . . 95
12.2.7 Modul ist_verbunden_mit-Relation . . . 95
12.2.8 Modul Anwendungsprogramm . . . 96
12.2.9 Modul Datenbanksystem . . . 97
12.2.10 Modul Bausteinschnittstelle . . . 98
12.2.11 Modul Servicetemplate . . . 98
12.2.12 Modul Hosttemplate . . . 99
12.2.13 Modul Multiservice . . . 99
12.2.14 Modul 3LGM²-Modellelementfelder . . . 99
12.3 Ausblenden von Nagiosobjekten . . . 99
12.4 Filtern von Services . . . 100
12.5 Vermeiden von identischen 3LGM²-Modellelementen . . . 100
12.6 Persistenz des globalen Datenbestandes und der Nagiosschlüssel . . . 100
12.7 Synchronisation . . . 103
12.8 Implementierung . . . 105
12.9 Umsetzung der Anforderungen . . . 106
12.9.1 Funktionelle Anforderungen . . . 106
12.9.2 Nicht funktionelle Anforderungen . . . 106
12.10 Erweiterung der Umsetzung . . . 107
12.11 Zusammenfassung . . . 107

13 Nutzen der 3LGM²-Datenintegration 109
13.1 Nutzen von Nagioskonfigurationen für 3LGM²-Modelle . . . 109

13.1.1 Feinere Granularität . . . 109
13.1.2 Bessere Aktualität . . . 110
13.1.3 Verminderter Arbeitsaufwand . . . 110
13.1.4 Reflexion in Nagios . . . 110
13.2 Nutzen von 3LGM²-Modellen für Nagios . . . 110
13.2.1 Konfigurationsunterstützung . . . 110
13.2.2 Reflexion in 3LGM²-Modelle . . . 111
13.3 „Best Practice“ für Nagios . . . 111
13.3.1 Objekte . . . 111
13.3.2 Hierarchien . . . 112
13.4 Zusammenfassung . . . 112

Z Abschluss **113**
Z.1 Diskussion . . . 113
Z.2 Zielerfüllung . . . 113
Z.3 Ausblick . . . 116

1 Einleitung

1.1 Gegenstand und Motivation

1.1.1 Gegenstand

Im Jahr 2005 wurden in Deutschland mehr als 16 Millionen Menschen in 2139 Krankenhäusern stationär behandelt [STAT. BUNDESAMT 06]. Dabei wurden für die Behandlung jedes Patienten Informationen benötigt, erstellt und verarbeitet. Die Informationslogistik, d.h. die richtige Information zur richtigen Zeit am richtigen Ort zu haben ist für Krankenhäuser in diesem Zusammenhang besonders wichtig, da auf Informationen ärztliche Entscheidungen basieren, die für den Patienten lebenswichtig sein können. Weiterhin werden Informationen für die Dokumentation und Abrechnung von Leistungen benötigt. Alle an der Erfüllung dieser informationsverarbeitenden Aufgaben beteiligten Personen und Geräte werden zusammengefasst als ein Krankenhausinformationssystem (KIS) bezeichnet. (vgl. [WINTER et al. 03] S. 544) Dieses heterogene Subsystem eines Krankenhauses muss ständig neue Anforderungen erfüllen, die nicht nur durch veränderte Informationsbedürfnisse der Nutzer, sondern auch durch gesetzliche Änderungen entstehen. Daraus folgend ist für eine funktionierende Informationslogistik ein adäquates Informationsmanagement Voraussetzung.

Am Institut für Medizinische Informatik, Statistik und Epidemiologie der Universität Leipzig wurde daher das 3LGM²-Metamodell entwickelt, um Informationssysteme beschreiben zu können und dadurch über alle notwendigen Informationen für ein erfolgreiches Informationsmanagement zu verfügen. Ein 3LGM²-Modell beschreibt dabei ein Informationssystem auf 3 Ebenen: der Fachlichen, der Logischen und der Physischen Werkzeugebene. Auf dem 3LGM²-Metamodell basiert ein Softwarewerkzeug, der 3LGM²-Baukasten, welcher es ermöglicht, auch komplexe KIS in Modellen abzubilden. Der 3LGM²-Baukasten wird von mehreren Institutionen genutzt und im Moment vollständig überarbeitet, um entstandene Anwenderwünsche zu berücksichtigen.

Ein KIS ist ein soziotechnisches Subsystem, dessen rechnerbasierter Anteil nicht unerheblich ist. Große Krankenhäuser besitzen oft mehrere tausend PCs und eine Vielzahl von Softwareprodukten, die darauf installiert ist. Für dieses rechnerbasierte Subsystem des KIS gibt es verschiedene Informationsmanagement-Werkzeuge (IM-Werkzeuge), die meist dem operativen Informationsmanagement dienen und auch in anderen Branchen zum Einsatz kommen. Sie können Daten über Hard- und Softwarebestände eines Informationsystems enthalten. Zur kontinuierlichen Überwachung von Rechnern können Netzwerkmanagementsysteme eingesetzt werden und mit Hilfe von Inventory Management kann eine Katalogisierung von PCs erfolgen. Manche IM-Werkzeuge sind Teil eines Computer Aided Facility Management Systems (CAFM).

1.1.2 Problematik

Wenn Änderungen des KIS in das dazugehörige 3LGM²-Modell übernommen werden oder ein 3LGM²-Modell neu erstellt wird, werden Daten erzeugt, die womöglich bereits in operativen IM-Werkzeugen vorhanden sind. Dieses Vorgehen kann eine Redundanz zwischen den Daten der operativen IM-Werkzeuge und dem 3LGM²-Modell erzeugen. In der Folge haben Änderungen im KIS sowohl in den operativen IM-Werkzeugen als auch im 3LGM²-Modell Auswirkungen.

Neben dem Problem der Redundanz besteht auch das Problem einer groben Granularität von 3LGM²-Modellen. Handelt es sich beispielsweise um tausend PCs eines Krankenhausinformationssystems, die in das 3LGM²-Modell aufgenommen werden sollen, dann neigen die Modellierer dazu, nicht alles zu beschreiben bzw. zu verallgemeinern. Das resultierende 3LGM²-Modell enthält dann weniger Informationen und die daraus gezogenen Schlussfolgerungen für das Informationsmanagement werden ungenauer.

1.1.3 Motivation

Wenn es gelingt, IM-Werkzeuge als Informationsquellen für den 3LGM²-Baukasten zu erschließen und Datenintegration herzustellen, dann könnte dem Benutzer durch die zusätzlichen Informationen wesentlich bei der Erstellung vollständiger 3LGM²-Modelle geholfen werden. Auch durch eine unterstützte Erstellung initialer Modelle wäre eine mögliche Akzeptanzschwelle für den 3LGM²-Baukasten überwunden.

3LGM²-Modelle könnten Informationssysteme durch die Daten aus operativen IM-Werkzeugen feingranularer darstellen. Dadurch wäre es Nutzern der operativen IM-Werkzeuge möglich, zusätzliche Informationen über die in den operativen IM-Werkzeugen bearbeiteten Sachverhalte mit Hilfe des 3LGM²-Baukastens auch aus 3LGM²-Modellen zu entnehmen und das ihnen zur Verfügung stehende Informationsangebot zu vergrößern.

Es zeichnen sich manche operative IM-Werkzeuge dadurch aus, dass sie Änderungen im rechnerbasierten Informationssystem automatisch erkennen können, was zur kontinuierlichen Aktualisierung von 3LGM²-Modellen genutzt werden könnte.

Ferner ist es denkbar, dass die in 3LGM²-Modellen enthaltenen Informationen operativen IM-Werkzeugen zur Verfügung gestellt werden.

1.2 Problemstellung

Problem P1 Der 3LGM²-Baukasten ist nicht mit anderen Softwareprodukten gekoppelt, mit deren Daten Benutzer beim Erstellen von 3LGM²-Modellen unterstützt werden könnten.

1.3 Zielstellung

Ziele zu Problem P1

Ziel Z1.1 Überblick über Softwareprodukte, die für 3LGM²-Modelle relevante Hard- oder Softwarebestände eines Unternehmens enthalten

Ziel Z1.2 Konsistenzbedingungen für 3LGM²-Modelle, die externe Daten enthalten

Ziel Z1.3 Strategien, wie externe Daten in ein 3LGM²-Modell integriert werden

Ziel Z1.4 Implementierung einer Schnittstelle zwischen einem Softwareprodukt und dem 3LGM²-Baukasten

1.4 Fragestellung

Fragen zu Ziel Z1.1

Frage F1.1.1 Welche Softwareprodukte gibt es, die als Quellen für Informationen dienen können?

Frage F1.1.2 Welche potenziell nutzbaren Schnittstellen bieten diese Softwareprodukte?

Frage F1.1.3 Welche für 3LGM²-Modelle relevanten Daten und Schnittstellen bietet der „DX-Union Asset Assistant“?

Frage F1.1.4 Welche für 3LGM²-Modelle relevanten Daten und Schnittstellen bietet der „IBM Tivoli NetView“?

Frage F1.1.5 Welche Daten können/können nicht integriert werden?

Frage F1.1.6 Welche Daten sollten/sollten nicht integriert werden?

Frage F1.1.7 Ist es denkbar, den 3LGM²-Baukasten als Datenquelle für andere Softwareprodukte zu nutzen?

Fragen zu Ziel Z1.2

Frage F1.2.1 Welche zusätzlichen Konsistenzbedingungen gelten für 3LGM²-Modelle mit externen Daten?

Frage F1.2.2 Wie kann man die Konsistenz des 3LGM²-Modells mit externen Daten sicherstellen?

Frage F1.2.3 Müssen externe Daten im 3LGM²-Modell besonders gekennzeichnet werden?

Fragen zu Ziel Z1.3

Frage F1.3.1 Welche Anforderungen gibt es bei der Integration von Daten aus externen Quellen?

Frage F1.3.2 Müssen bei unterschiedlichen Quellen unterschiedliche Anforderungen an die Integration gestellt werden?

Frage F1.3.3 Inwieweit können externe Informationen über Krankenhausinformationssysteme automatisch in 3LGM²-Modelle abgebildet werden?

Frage F1.3.4 Wie erfolgt dies einmalig?

Frage F1.3.5 Wie erfolgt dies kontinuierlich?

Fragen zu Ziel Z1.4

Frage F1.4.1 Welches Softwareprodukt bietet sich an, mit dem 3LGM²-Baukasten gekoppelt zu werden und wie?

Frage F1.4.2 Welcher Grad der Kopplung wird realisiert und warum?

Frage F1.4.3 Entspricht die Realisierung den Erwartungen?

2 Aufbau

Diese Arbeit ist in 3 Teile gegliedert. Zu Beginn werden die existierenden Grundlagen erläutert, die für die folgenden Kapitel wesentlich sind.

In einem zweiten Teil werden Rahmenbedingungen für die Datenintegration diskutiert mit dem Ziel, einen Vorschlag zu schaffen, der für zukünftige Datenintegrationen mit ähnlicher Problematik anwendbar ist.

Im dritten Teil wird basierend auf den vorangegangenen Erkenntnissen die Integration von Daten aus dem Netzwerk Monitoring System Nagios mit dem 3LGM²-Baukasten vorgestellt. Dafür wird der allgemeine Lösungsansatz konkretisiert und und prototypisch implementiert.

Teilübergreifend werden die Themen Datenintegration, 3LGM²-Metamodell und verschiedene Aspekte von IM-Werkzeugen bearbeitet.

Den Abschluss dieser Arbeit bildet die Diskussion der Ergebnisse.

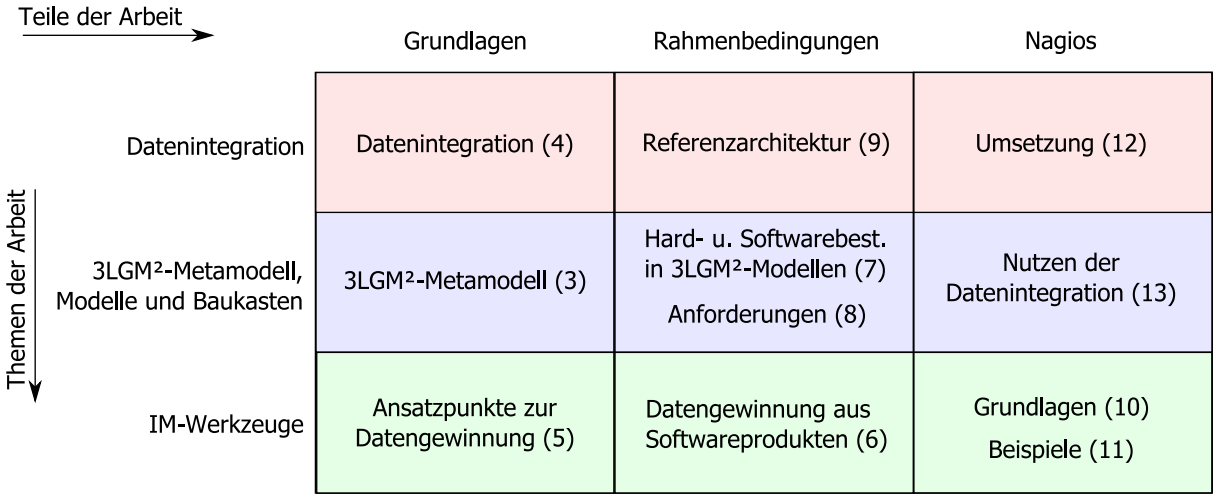

Abbildung 2.1: Die Kapitel der Arbeit dargestellt im Zusammenhang mit den Themen und Teilen.

Typographische Konventionen

Die Definitionen, Festlegungen und Annahmen, die grau hinterlegt sind, sind besonders wichtig für das Verständnis der darauf folgenden Teile der Arbeit und deshalb hervorgehoben.

Teil I

Grundlagen

3 Das 3LGM²-Metamodell

Inhaltsangabe

3.1 Die Fachliche Ebene 9
3.2 Die Logische Werkzeugebene 10
3.3 Die Physische Werkzeugebene 11
3.4 Interebenenbeziehungen 12
3.5 Modellieren mit dem 3LGM²-Baukasten 13
3.6 Zusammenfassung 13

3LGM² ist die Abkürzung für Three Layer Graph Based Meta-Model. Das 3LGM²-Metamodell dient der Beschreibung, Bewertung und Planung von Informationssystemen im Gesundheitswesen. Es kann auch für andere Arten von Informationssystemen (IS) verwendet werden, ist aber für medizinisch orientierte Informationssysteme optimiert. Wesentlich ist zum einen die Teilung in 3 Ebenen, bestehend aus Fachlicher Ebene, Logischer Werkzeugebene und Physischer Werkzeugebene und zum anderen die Definition als Meta-Modell, die darauf hinweist, dass das 3LGM²-Metamodell für das Beschreiben von Modellen konzipiert ist und nicht als Modell selbst.

In einem 3LGM²-Modell gibt es Beziehungen zwischen Elementen auf den einzelnen Ebenen und zwischen den Elementen der verschiedenen Ebenen. Es lässt sich sowohl der rechnerbasierte Teil als auch der nicht rechnerbasierte Teil des Informationssystems beschreiben. Dies ist notwendig, da ein Informationssystem ein soziotechisches System mit menschlichen und maschinellen Handlungsträgern ist. Neben der Einteilung in Ebenen wird noch in Sichten unterteilt. Dabei gibt es die dynamische Sicht und die statische Sicht. Das 3LGM²-Metamodell unterstützt insbesondere die statische Sicht und gibt nur an wenigen Stellen die Möglichkeit, dynamische Aspekte eines Informationssystems zu modellieren. Das 3LGM²-Metamodell ist mit Hilfe der Unified Modeling Language (UML) beschrieben. Weitere Informationen zum 3LGM²-Metamodell mit seinen in den nächsten Abschnitten vorgestellten Ebenen finden sich in [WINTER et al. 03] und [WENDT et al. 04].

3.1 Die Fachliche Ebene

Auf der Fachlichen Ebene werden *Aufgaben* des Unternehmens dargestellt. Diese Aufgaben sind unabhängig von den zu ihrer Erfüllung benötigten Ressourcen. Sie benötigen und erzeugen Informationen, die entsprechenden *Objekttypen* zugeordnet werden. Handlungsträgern des Informationssystems können gewisse *Rollen* zugeordnet werden, die ebenso zur Erledigung

einer *Aufgabe* beitragen. Weiterhin werden die *Aufgaben* eines Unternehmens in *Organisationseinheiten* erledigt. Das UML-Diagramm der Fachlichen Ebene ist in Abbildung 3.1 zu finden.

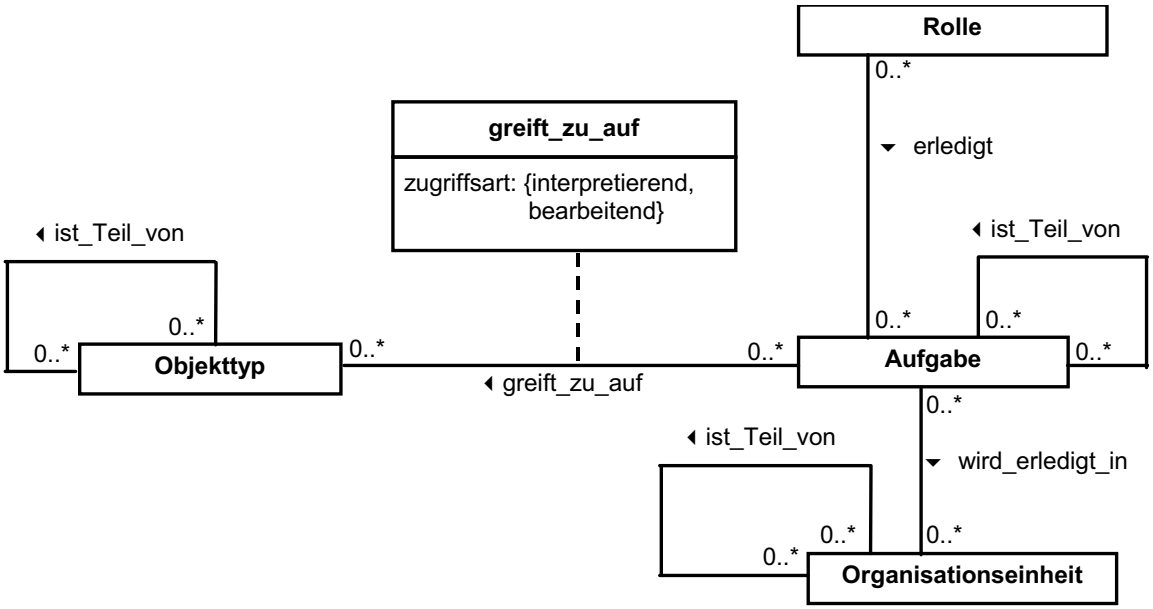

Abbildung 3.1: Das UML-Diagramm der Fachlichen Ebene des 3LGM²-Metamodells (aus [WINTER et al. 06])

3.2 Die Logische Werkzeugebene

Die umfangreichste Ebene des 3LGM²-Metamodells ist die Logische Werkzeugebene. Die hier beschriebenen *Anwendungsbausteine* unterstützen die Unternehmensaufgaben und können *rechnerbasiert* oder *papierbasiert* sein. An dieser Stelle wird das 3LGM²-Metamodell der Anforderung gerecht, dass Informationssysteme soziotechnisch sind. Ein *Anwendungsbaustein* kann demnach durch ein *Softwareprodukt* mit *Anwendungsprogrammen* repräsentiert werden, muss es aber nicht. Die Repräsentation durch einen *Papierbasierten Anwendungsbaustein*, der durch einen *Organisationsplan* ergänzt wird, reicht aus. Die Unterstützung einer *Aufgabe* in der *Fachlichen Ebene* durch einen *Anwendungsbaustein* wird realisiert durch eine *Anwendungsbaustein-Konfiguration*. Dabei entsteht eine Interebenenbeziehung, die noch näher erläutert wird.

Ebenfalls auf der logischen Werkzeugebene befinden sich die Beschreibungselemente für die Speicherung und Kommunikation von Daten. Daten können in *Datenbanksystemen* oder *Dokumentensammlungen* verwahrt werden, wobei erstere noch durch ein *Datenbankverwaltungssystem* erweitert werden können. Zu diesen beiden Speichervarianten werden zusätzlich noch die zu speichernden Datentypen in Form von *Datensatztypen* und *Dokumententypen* definiert. Diese beiden Typen und der *Nachrichtentyp* sind *Repräsentationsformen*, die den oben erwähnten *Objekttyp* der Fachlichen Ebene repräsentieren.

Der *Nachrichtentyp* und der *Dokumentenentyp* werden zur Beschreibung der Kommunikation

auf der Logischen Werkzeugebene benutzt. Das 3LGM²-Metamodell kann Kommunikation darstellen, die auf Ereignissen basiert und beschreibt diese mit einem *Ereignis-Nachrichtentyp* oder einem *Ereignis-Dokumententyp*. Dieser vereinigt mit dem *Ereignistyp* das auslösende Ereignis, sowie einen *Nachrichtentyp* oder *Dokumententyp*. Daneben gibt es noch *Kommunikationsschnittstellen* der *Anwendungsbausteine*. Zu den *Kommunikationsschnittstellen* zählen die *Bausteinschnittstelle*, für die Kommunikation zwischen *Anwendungsbausteinen* und die *Benutzerschnittstelle*, die, wie der Name bereits sagt, Kommunikation mit dem Benutzer ermöglicht. Durch die Verbindung von *Ereignis-Nachrichtentyp* oder *Ereignis-Dokumententyp* und *Bausteinschnittstelle* kann die Fähigkeit zur Kommunikation beschrieben werden. Diese Fähigkeit kann noch mit einem *Kommunikationsstandard* konkretisiert und durch eine *Kommunikationsbeziehung* realisiert werden.

Das UML-Diagramm der Logischen Werkzeugebene ist in Abbildung 3.2 zu finden.

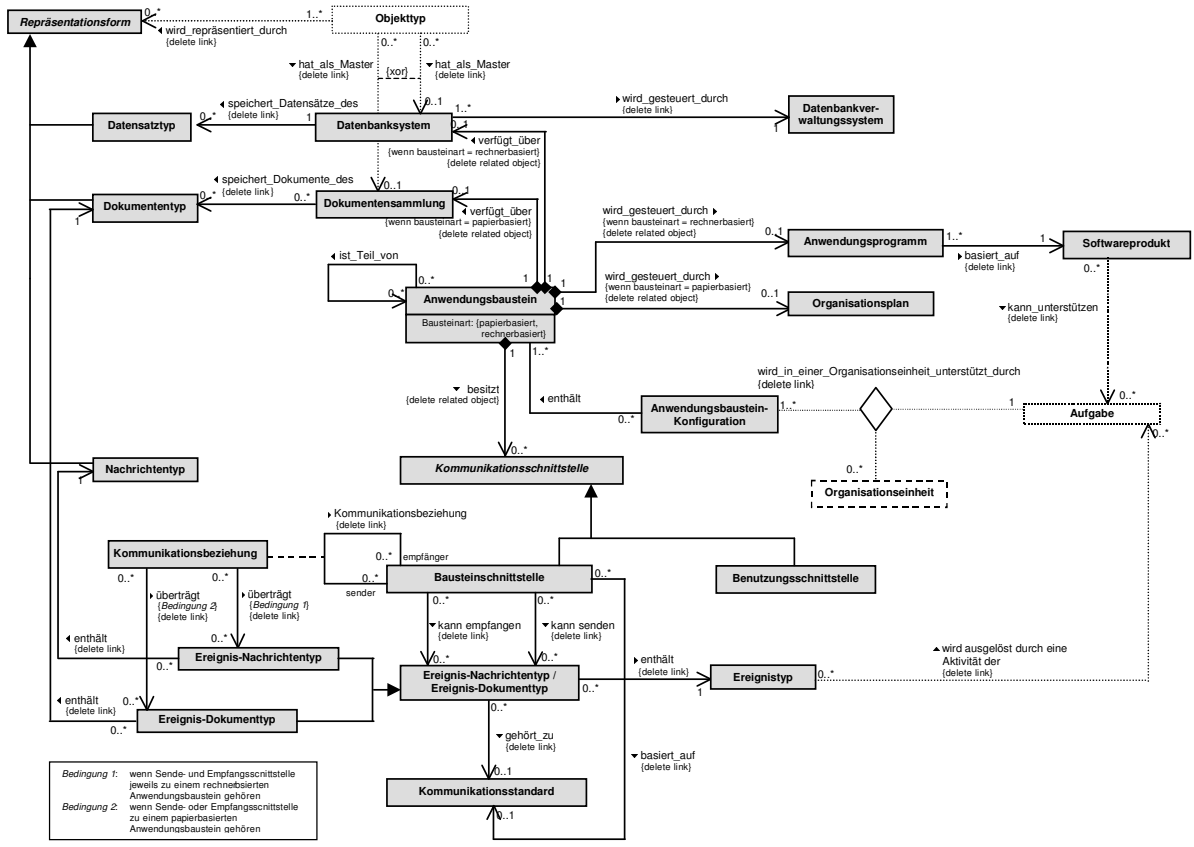

Abbildung 3.2: Das UML-Diagramm der Logischen Werkzeugebene des 3LGM²-Metamodells (aus [WINTER et al. 06])

3.3 Die Physische Werkzeugebene

Die Physische Werkzeugebene stellt die physischen Bestandteile eines Informationssystems dar. Zentrales Element der Physischen Werkzeugebene ist daher der *Physische Datenverarbeitungsbaustein*, wobei unter *Physischem Datenverarbeitungsbaustein* durchaus auch die Systeme

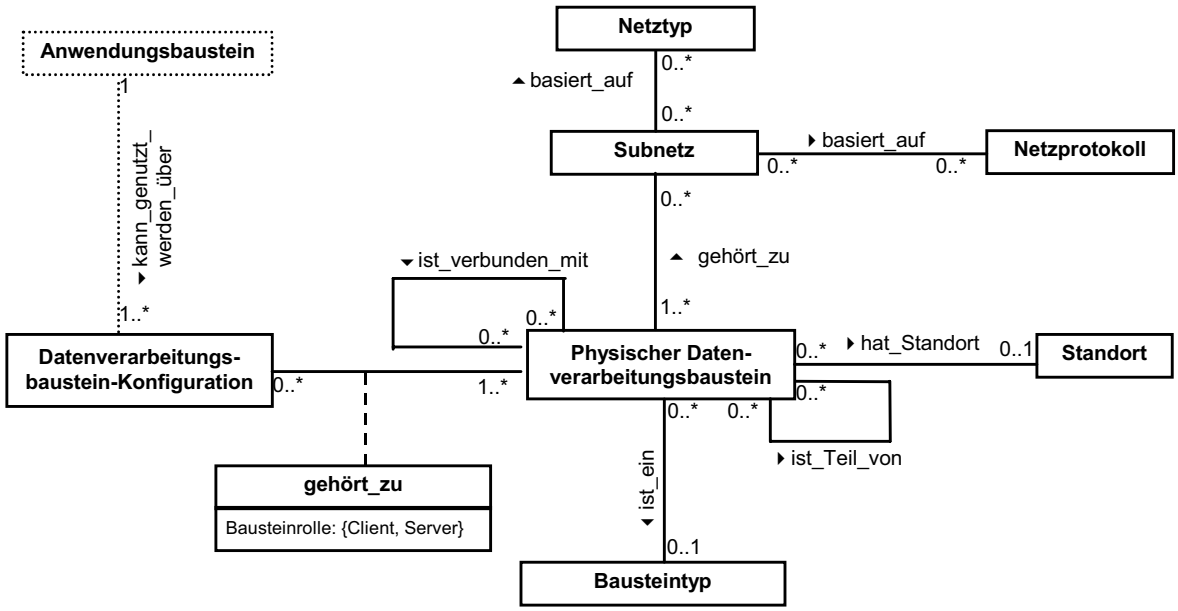

Abbildung 3.3: Das UML-Diagramm der Physischen Werkzeugebene des 3LGM²-Metamodells (aus [WINTER et al. 06])

menschlicher Handlungsträger und konventionelle Werkzeuge verstanden werden können. Ihm kann ein *Bausteintyp* zugeordnet werden sowie ein oder mehrere *Subnetze*, die wiederum durch *Netzprotokolle* und *Netztypen* näher bestimmt werden können. Falls ein *Standort* bekannt ist, kann dieser ebenfalls einem *Datenverarbeitungsbaustein* zugeordnet werden. Die Verbindung zwischen *Anwendungsbausteinen* aus der Logischen Werkzeugebene und den *Physischen Datenverarbeitungsbausteinen* der Physischen Werkzeugebene stellt die *Datenverarbeitungsbaustein-Konfiguration* her, die genau einem *Anwendungsbaustein* und mindestens einem *Physischen Datenverarbeitungsbaustein* zugeordnet ist. Dabei nimmt der *Physische Datenverarbeitungsbaustein* eine Client- oder Serverrolle ein

Das UML-Diagramm der Physischen Werkzeugebene ist in Abbildung 3.3 zu finden.

3.4 Interebenenbeziehungen

Von besonderem Interesse sind die Beziehungen zwischen den einzelnen Ebenen. Verschiedene Fragestellungen, zum Beispiel bezüglich der Redundanz von *Anwendungsbaustein-Konfigurationen*, lassen sich durch Interebenenbeziehungen beantworten. Sowohl zwischen der Fachlichen Ebene und der Logischen Werkzeugebene als auch zwischen der Logischen Werkzeugebene und Physischen Werkzeugebene gibt es verschiedene Interebenebenbeziehungen.

3.5 Modellieren mit dem 3LGM²-Baukasten

Um auch große Informationssysteme beschreiben und Veränderungen in bereits erstellten 3LGM²-Modellen vornehmen zu können, wurde eine Software entwickelt, die es ermöglicht, 3LGM²-Modelle zu erstellen, zu ändern und für Analysen zu nutzen. Diese Software ist der 3LGM²-Baukasten, der am Institut für Medizinische Informatik, Statistik und Epidemiologie (IMISE) entwickelt wurde. Im 3LGM²-Baukasten sind ebenfalls die drei Ebenen sowie eine Baumansicht vorhanden und es werden Analysefunktionen angeboten, die spezielle Fragestellungen anhand eines 3LGM²-Modells beantworten. Der 3LGM²-Baukasten ist damit ein Werkzeug des taktischen und strategischen Informationsmanagements.

Im Moment wird der 3LGM²-Baukasten vollständig überarbeitet, um zukünftigen Nutzeranforderungen besser gerecht werden zu können. Daher wird im Folgenden teilweise von neuem und altem 3LGM²-Baukasten gesprochen. Die Funktionalität des alten 3LGM²-Baukastens soll im neuen 3LGM²-Baukasten erhalten bleiben und erweitert werden. Beide Varianten des 3LGM²-Baukastens basieren auf der Programmiersprache Java.

3.6 Zusammenfassung

Das 3LGM²-Metamodell spielt eine zentrale Rolle in der gesamten Arbeit. Dieses Kapitel hat daher die Grundlagen des 3LGM²-Metamodells und den 3LGM²-Baukasten als Werkzeug zur Umsetzung des 3LGM²-Metamodells kurz vorgestellt. Zudem wurde der Nutzen des 3LGM²-Metamodells dargestellt.

4 Datenintegration

Inhaltsangabe

4.1 Integrationsziele 15
4.2 Physische Integration 16
4.3 Definition der Datenintegration 16
4.4 Abgrenzung gegenüber Importfunktionalitäten 17
4.5 Analogie zu Datenbanksystemen 17
4.6 Der Datenintegrationsprozess 18
4.6.1 Integration der Datenstrukturen 19
4.6.2 Integration der Datenbestände 22
4.7 Automatisierung der Datenintegration 24
4.8 Zusammenfassung 24

Neben den Grundlagen des 3LGM²-Metamodells werden die Grundlagen der Datenintegration benötigt, um die Ziele dieser Arbeit zu erreichen. Dabei sollen verschiedene Varianten aufgezeigt werden, von denen später für eine Realisierung eine Variante ausgewählt wird.

4.1 Integrationsziele

Zunächst eine Definition für Integration:

Definition Integration is a union of parts making a whole, which as opposed to its parts, displays a new quality. ([HAUX et al. 04] S.127)

Ziel der Integration ist danach eine Erhöhung der Qualität durch das Verbinden von Einheiten, die vorher unverbunden waren. Dies ist zu allgemein, um für das Problem dieser Arbeit angewendet zu werden, stellt aber die Eigenschaft dar, die allen Integrationskategorien gemein ist. Zu den verschiedenen Kategorien zählen: (vgl. [WENDT 06] S.12)

- physische Integration
- Datenintegration
- funktionale Integration
- semantische Integration

- Kontextintegration
- Präsentationsintegration
- Zugriffsintegration

In den nächsten Kapiteln sollen die *physische Integration* und die *Datenintegration* erläutert werden. Die *Datenintegration* ist dabei der Kern dieser Arbeit und die *physische Integration* Voraussetzung für die *Datenintegration*. Nur mit der *Datenintegration* werden die Daten von 3LGM²-Modellen erweitert, wodurch die integrierten Daten auch für Analysen genutzt werden können und das Ziel dieser Arbeit erreicht wird.

4.2 Physische Integration

Physische Integration fordert eine funktionsfähige Datenverbindung zwischen den zu integrierenden Komponenten, so dass diese in eine Kommunikationsbeziehung treten können. Beispielsweise kann durch Netzwerke physische Integration zwischen Rechnern hergestellt werden.

Für den 3LGM²-Baukasten bedeutet dies, er muss Zugriff auf die Daten oder Schnittstellen der zu koppelnden Software haben. Dieser Zugriff muss entweder über Netzwerkverbindungen oder durch Installation auf ein und dem selben Rechner erfolgen. (vgl. [WENDT 06] S.16)

4.3 Definition der Datenintegration

Die folgende Definition beschreibt die Datenintegration speziell für Krankenhausinformationssysteme.

Definition Data integration is guaranteed in a hospital information system when data that have been recorded are available wherever they are needed, without having to be reentered. Thus, each data item needs to be recorded, changed, deleted, or otherwise edited just once - even if it is used in several application components. ([HAUX et al. 04] S.127)

In dieser Definition wird gefordert, dass alle Daten, die in das IS aufgenommen sind, wiederverwendet werden können. Das geht weiter als das Ziel dieser Arbeit, da hier nicht ganze Informationssysteme betrachtet werden. Die folgende Definition soll daher zusätzlich angeführt werden.

Definition Data integration is the problem of combining data residing at different sources, and providing the user with a unified view of these data. ([LENZERINI 02] S.233)

Die zweite Definition aus [LENZERINI 02] entspricht der Auffassung vieler Veröffentlichungen über Datenintegration. Leider greift diese Definition für das Ziel dieser Arbeit zu kurz, da die reine Existenz einer gemeinsamen Sicht auf die Daten dabei für Datenintegration steht. Die Verwendung dieser Daten durch Anwendungen, wie sie in [HAUX et al. 04] vorgeschlagen wurde, soll in dieser Arbeit auch Teil der Datenintegration sein.

4.4 Abgrenzung gegenüber Importfunktionalitäten

Eine Importfunktionalität ist das einmalige Kopieren fremder Daten in den eigenen Datenbestand. Dabei ist jeder Importvorgang unabhängig von vorangegangenen Importen. Es gibt keine dauerhafte Beziehung zwischen Datenquelle und Datensenke. Demgegenüber soll die angestrebte Datenintegration als dauerhafter Prozess verstanden werden, der eine Beziehung zwischen den Daten herstellt.

Während bei einem Datenimport der Nutzen durch die zusätzlich verfügbaren Daten nur für eine Anwendung besteht, ermöglicht die Datenintegration einen Nutzen für mehrere Anwendungen. Zwar wird in dieser Arbeit der Nutzen für 3LGM²-Modelle priorisiert, die Entscheidung für eine Datenintegration soll aber einen zukünftigen Nutzen auch für andere Anwendungen ermöglichen.

4.5 Analogie zu Datenbanksystemen

Das Thema der Datenintegration ist von besonderem Interesse im Bereich der Datenbanken. Das ist auf eine Entwicklung zurückzuführen, die nicht mehr auf zentralisierte Datenbankmanagementsysteme setzt, sondern auf Informationssysteme mit autonomen Datenquellen.
(vgl. [NAUMANN 03] S.45)

Auch wenn sich die hier angestrebte Datenintegration nicht auf Datenbanken bezieht, so lassen sich dennoch die Datenbankerkenntnisse zu großen Teilen auf den 3LGM²-Baukasten projizieren. Dieser Vergleich ist lohnenswert, da sich so auch die Probleme und Lösungsansätze von Datenbanken für den 3LGM²-Baukasten nutzen lassen. Um Missverständnissen vorzubeugen, soll folgende Terminologie verwendet werden:

Festlegung Anstelle des Begriffs Schema oder Datenbankschema wird hier allgemeiner *Datenstruktur* verwendet.

Festlegung Als *Datenbestand* sollen die von einer Anwendung erzeugten bzw. verwendeten Daten verstanden werden.

Definition Das *Datenmodell* legt die Modellierungskonstrukte fest, mittels derer man ein computerisiertes Informationsabbild der realen Welt (bzw. des relevanten Ausschnitts) generieren kann. ([KEMPER et al. 04] S.21)

4.6 Der Datenintegrationsprozess

Festlegung Der Prozess, der vom Vorhandensein verschiedener Datenquellen zum integrierten Datenbestand führt, soll hier als *Datenintegrationsprozess* verstanden werden.

Der Datenintegrationsprozess teilt sich in 2 Teilphasen auf, die *Integration der Datenstrukturen* und die *Integration der Datenbestände*. (vgl. [JUNG 06] S.157) Dabei stellt der erste Teil die Grundlagen zur Verfügung, dass die Datenbestände integriert werden können und im 2. Teil werden die Daten integriert. Der 2. Teilprozess kann unter Umständen iterieren, wenn in den Datenquellen weiterhin Datenänderungen stattfinden und eine ständige Datenintegration benötigt wird. In Abbildung 4.1 sind die verschiedenen Phasen in Aktivitäten unterteilt. Die Integration der Datenstrukturen unterteilt sich in zwei Varianten, die in Kapitel 4.6.1.1 besprochen werden. Ebenso teilt sich auch die Integration der Datenbestände in 2 Varianten die in Kapitel 4.6.2.2 besprochen werden. In der Literatur ist die Integration der Datenstrukturen gegenüber der Integration der Datenbestände deutlich stärker vertreten. Insbesondere Arbeiten zur Datenbankschemaintegration sind zu finden.

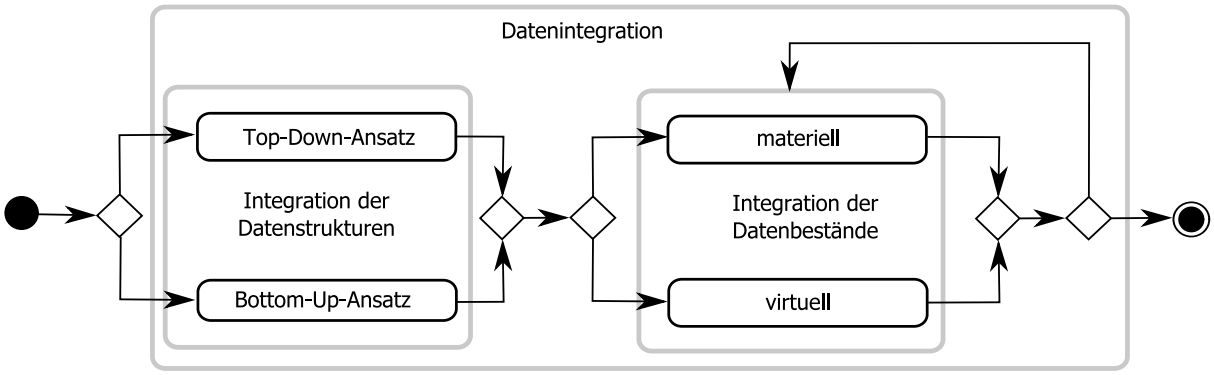

Abbildung 4.1: Die Phasen der Datenintegration, dargestellt in Anlehnung an ein UML-Aktivitätsdiagramm. Graue Aktivitäten fassen dabei die schwarzen Aktivitäten zusammen. Die Datenintegration besteht aus der Struktur- und der Datenbestandsintegration, die sich wiederum in 2 Varianten aufteilen.

4.6.1 Integration der Datenstrukturen

Ziel der Datenstrukturintegration ist es, eine einheitliche Beschreibung für die Strukturen der verschiedenen Quellen zu finden. In diesem Zusammenhang wird auch der Begriff „Mapping“ verwendet.

Definition Unter *Mapping* ist die Zuordnung eines Attributs aus einem lokalen Schema zu einem Attribut aus dem globalen Schema zu verstehen. ([JUNG 06] S.158)

Nach [LENZERINI 02] S.235 ist das „Mapping“ eine der wichtigsten Aufgaben der Datenintegration.

4.6.1.1 Lokale und globale Datenstruktur

Sowohl in [JUNG 06] S.156 als auch in [LENZERINI 02] S.233 werden mehrere lokale und eine globale Datenstruktur definiert.

Die *lokalen Datenstrukturen* sind dabei die Datenstrukturen der zu integrierenden Datenbestände. Im hier bearbeiteten Fall sind dies die Datenstruktur des 3LGM²-Baukastens und die Datenstruktur der zu integrierenden Anwendung. Liegen keine Informationen über die Struktur der Daten vor, so muss diese Datenstruktur rekonstruiert werden.

Die *globale Datenstruktur* ist die Datenstruktur, in die die lokalen Datenstrukturen gemappt werden sollen. Sie muss im Allgemeinen neu erstellt werden, wozu man einen „Bottom-up“ oder „Top-down“ Ansatz wählen kann.

Der *Bottom-Up-Ansatz* erzeugt eine globale Datenstruktur, die aus den lokalen Datenstrukturen abgeleitet ist. Dies erleichtert die anschließende Integration der Datenbestände, wie in Abbildung 4.2 dargestellt. Dieser Ansatz wird auch als *declarative* (vgl. [CALVANESE et al. 98] S.3) oder *Global-As-View* (vgl. [LENZERINI 02] S.233) bezeichnet. Dabei hat der *Bottom-Up-Ansatz* den Vorteil, dass die lokalen Datenstrukturen in der globalen Datenstruktur gut erkennbar bleiben und den Nachteil, dass das globale Schema recht umfangreich sein kann.

Der *Top-Down-Ansatz* orientiert sich am Informationsbedarf, der durch Anwendungen und Nutzer entsteht, wie in Abbildung 4.3 dargestellt. Teilweise findet man den Top-Down-Ansatz auch als *procedural* (vgl. [CALVANESE et al. 98] S.3) oder *Local-As-View* (vgl. [LENZERINI 02] S.233). Dabei ist von Vorteil, dass die globale Datenstruktur nicht unbedingt geändert werden muss, wenn sich Änderungen in den lokalen Datenstrukturen ergeben. Als zusätzlicher Schritt muss allerdings nach der Erstellung einer ersten Version der globalen Datenstruktur diese gegen die lokalen Datenstrukturen gemappt werden und bei Unzulänglichkeit gegenüber den lokalen Datenstrukturen verbessert werden.

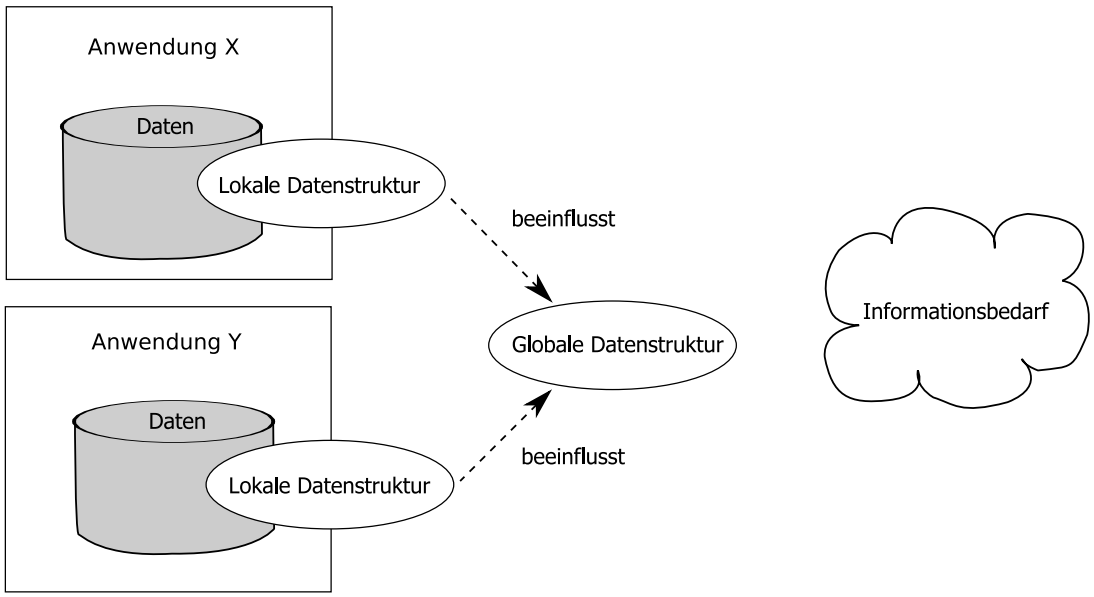

Abbildung 4.2: Die Abhängigkeiten der globalen Datenstruktur bei der Datenintegration durch einen Bottom-Up-Ansatz

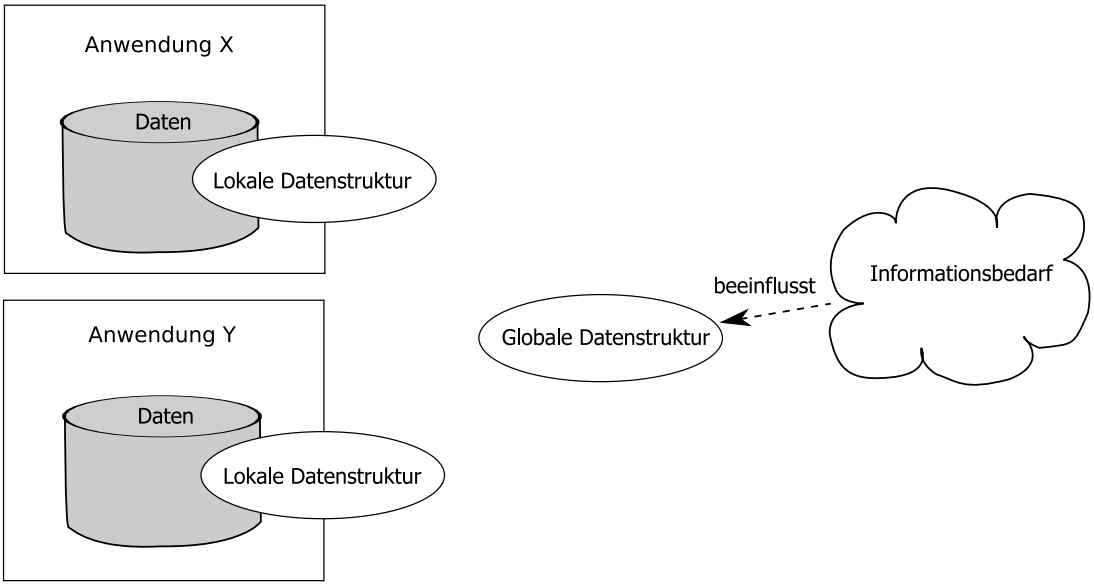

Abbildung 4.3: Die Abhängigkeiten der globalen Datenstruktur bei der Datenintegration durch einen Top-Down-Ansatz

4.6.1.2 Überlappung von Datenstrukturen und Datenbeständen

Bei der Datenstrukturintegration wird versucht, aus zwei oder mehr Datenstrukturen eine gemeinsame Datenstruktur zu erstellen. Dabei können die verschiedenen Datenbestände die gleichen Objekte der realen Welt beschreiben und sich in ihren Datenstrukturen überlappen. In [NAUMANN 03] wird diese Überlappung genauer wie folgt beschrieben.

> Autonome Datenquellen überlappen sich extensional, also in den Objekten, die sie repräsentieren, und intensional, also in den Daten (Attributen) über die einzelnen Objekte. ([NAUMANN 03] S.56)

Kommt es zu keiner extensionalen Überlappung, so können die Datenstrukturen leicht integriert werden, da es zu keinen Konflikten kommt.

Bei sich überlappenden Datenbeständen entstehen einerseits Probleme beim Mapping der Datenstrukturen und andererseits beim Integrieren der Datenbestände, da ein Objekt der realen Welt gleichzeitig durch verschiedene lokale Datenbestände unterschiedlich beschrieben werden kann.

Geht man von einem *Buttom-Up-Ansatz* für die Strukturintegration aus, so lassen sich die Probleme den im Folgenden aufgelisteten Problemfeldern zuordnen. Beim *Top-Down-Ansatz* umgeht man diese Probleme zunächst, da sich die globale Datenstruktur nur auf den Informationsbedarf stützt. Bei dem anschließenden Mapping der lokalen Datenstrukturen auf die globale Datenstruktur treten dann allerdings wieder die selben Problemfelder auf, was zu den angesprochenen Nachbesserungen führt.

4.6.1.3 Problemfelder der Datenstrukturintegration

Die folgenden Abschnitte stützen sich auf die in [BATINI et al. 86] S.334 aufgeführten Problemfelder.

Different Perspectives Datenstrukturen werden immer mit einem Ziel erstellt. Sie sollen gewisse Anforderungen erfüllen, während andere Aspekte vernachlässigt werden. Die Anforderung an das 3LGM²-Metamodell ist es, (Krankenhaus-) Informationssysteme strukturiert unter funktionellen Gesichtspunkten zu beschreiben. Ein Netzwerkmanagementsystem beschreibt mit einer Netzwerktopologie ebenfalls einen Teil eines Informationssystems, allerdings aus einem eher technischen Blickwinkel.

Equivalence among Constructs of the Model Beim Betrachten von Daten lassen sich oft Objekte erkennen, die durch Attribute näher spezifiziert werden. Beim 3LGM²-Baukasten sind die Objekte die Ausprägungen der 3LGM²-Modellklassen und die Attribute beispielsweise die Beschreibung oder der Name eines Objekts. *Anwendungsprogramme* sind in 3LGM²-Modellen Objekte der 3LGM²-Metamodellklasse *Anwendungsprogramm*. In anderen Datenenquellen können die Anwendungsprogramme eines Rechners als Attribute eines Objekts „Rechner X“ betrachtet werden. In diesem Fall würden identische Sachverhalte einerseits als Objekt und andererseits als Attribut beschrieben werden. Es läge keine Gleichwertigkeit von Modellkonstrukten vor.

Incompatible Design Specifications Nichtkompatible Datenstrukturen sind Probleme, die sich nur schwer lösen lassen, da ihr Ursprung eine Abstraktionsebene höher liegt, im Verständnis des Sachverhalts. Im 3LGM²-Metamodell hat jedes *Softwareprodukt* mindestens ein *Anwendungsprogramm*. Würden zu integrierende Daten Softwareprodukten keine Anwendungsprogramme zuordnen, wäre dies nicht kompatibel.

4.6.2 Integration der Datenbestände

4.6.2.1 Extensionale und intensionale Überlappung

Nach der Erstellung einer globalen Datenstruktur schließt sich der Vorgang der Integration der Datenbestände an.

Handelt es sich um extensional und intensional nicht überlappende Daten, so ist die Integration der Datenbestände ebenso wie die Integration der Datenstrukturen vergleichsweise einfach. Weitaus schwieriger wird es, wenn sich Daten überlappen. Überlappungen müssen bei der Datenintegration erkannt und richtig behandelt werden. Oftmals wird der Informationsgewinn der integrierten Daten aber gerade aus dieser Überlappung gezogen.

Ziel ist es, zu erkennen, welche Daten der lokalen Datenbestände ein und das selbe Objekt der realen Welt beschreiben. Idealerweise kann das mit Hilfe eines Schlüsselwertes erfolgen, der in allen lokalen Datenbeständen existiert. Leider ist dies nicht immer der Fall. Kann man nicht ausschließen, dass in den verschiedenen lokalen Datenbeständen teilweise die selben Objekte beschrieben werden, so muss man notfalls manuell die Daten den Objekten zuordnen, was das Einführen von entsprechenden Schlüsseln beinhaltet, um die Objekte dauerhaft einander zuordnen zu können.

In diesem Schritt findet auch das Data Cleaning statt.

Definition *Data cleaning*, also called data cleansing or scrubbing, deals with detecting and removing errors and inconsistencies from data in order to improve the quality of data. ([RAHM et al. 00])

4.6.2.2 Materielle und virtuelle Datenintegration

In Bezug auf den Verbleib von integrierten Daten gibt es zwei Möglichkeiten, die bei [JUNG 06] beschrieben werden. Es wird zwischen *materieller Datenintegration* und *virtueller Datenintegration* unterschieden. Bei der *virtuellen Datenintegration* wird kein Datenbestand mit den integrierten Daten angelegt. Sie werden immer bei Bedarf aus den Ursprungsdaten erstellt. Bei der *materiellen Datenintegration* existiert ein integrierter Datenbestand. *Materielle Datenintegration* wird auch *fetch in advance* und *virtuelle Datenintegration fetch on demand* genannt. (vgl. [JUNG 06] S.176) Dabei gibt diese Einteilung keinen Aufschluss darüber, ob der globale Datenbestand persistent oder transient ist, also gespeichert wird oder flüchtig ist.

Abbildung 4.4 zeigt die Realisierungsmöglichkeiten des globalen Datenbestandes.

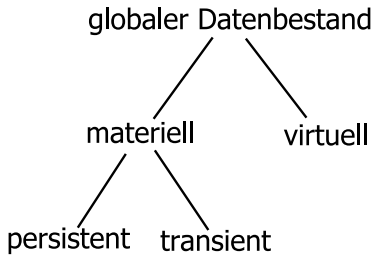

Abbildung 4.4: Die verschiedenen Möglichkeiten einen globalen Datenbestand zu realisieren

4.6.2.3 Zeitliche Verzögerung der Integration von Datenbeständen

Gründe für zeitliche Verzögerung Eine zeitliche Verzögerung der Integration von Datenbeständen kann nur dann auftreten, wenn die Daten redundant gehalten werden und die lokalen Datenbestände weiter genutzt werden. Die Redundanz entsteht dabei zwischen den lokalen Datenbeständen der Anwendungen und dem globalen Datenbestand. Beim Übernehmen von Änderungen aus den lokalen Datenbeständen in den globalen Datenbestand kann es dann zu Verzögerungen kommen. Die Verzögerung kann ungewollt sein und durch unzureichende Leistungsfähigkeit der zugrunde liegenden Informationstechnik hervorgerufen werden. Das ist ein Zeichen für mangelnde physische Integration. Oder sie kann bewusst herbeigeführt sein, um eben diese Informationstechnik nicht zu überlasten oder zu blockieren.

Eine Überlastung ist insbesondere dann möglich, wenn zum Zeitpunkt der Notwendigkeit eines erneuten Abgleichs zwischen den Datenbeständen nicht klar ist, welche Datenobjekte übertragen werden müssten, um wieder Datenintegration herzustellen. Die Folge ist, dass die gesamten Datenbestände erneut übertragen und integriert werden müssen.

Synchrone und asynchrone Datenintegration Die Definition von Datenintegration beschreibt einen ständig vorhandenen Zustand (vgl. [HAUX et al. 04] S.127). Um diesen Zustand unter Beibehaltung der lokalen Datenbestände zu gewährleisten, muss die Datenintegration ein ständig laufender Prozess sein.

Eine etwas differenziertere Sichtweise (vgl. [JUNG 06] S.199) geht davon aus, dass eine Datenintegration einer Synchronisierungskontrolle unterliegt, die entweder *lokal-zu-global* oder *global-zu-lokal* synchronisiert. Die Synchronisierungskontrollen unterscheiden sich danach durch den Ort, an dem Änderungen vorgenommen werden, ob im lokalen oder im globalen Datenbestand. Davon ausgehend kann der zeitliche Aspekt zusätzlich in *synchron* und *asynchron* unterteilt werden, wobei die *synchrone Datenintegration* der Definition von [HAUX et al. 04] entspricht. Die *asynchrone Datenintegration* wird weiter unterteilt nach der Wahl des Zeitpunktes für eine erneute Datenintegration:

- *zeitorientiert*: wenn Zeitintervalle oder Zeitpunkte feststehen
- *benutzerinitiiert*: durch den Benutzer ausgelöst
- *anwendungsinitiiert*: durch die Anwendung ausgelöst
- *opportunistisch*: zum frühest möglichen Zeitpunkt

4.7 Automatisierung der Datenintegration

Der Grad der Automatisierung einer Datenintegration setzt sich zusammen aus der Automatisierung der Integration der Datenstrukturen und der Automatisierung der Integration der Datenbestände. Insbesondere wenn die Integration der Datenbestände mehrfach durchgeführt werden muss ist eine weitestgehende Automatisierung dieses Teils der Datenintegration wichtig. Automatisierung von Datenintegration kann man grob wie folgt unterteilen.

Gelingt die Datenintegration ohne manuelle Hilfe, so ist eine *automatische Datenintegration* erreicht.

Ist nur ein Teil der Datenintegration automatisch und der Rest manuell, so handelt es sich um eine *semiautomatische Datenintegration*.

Um eine *manuelle Datenintegration* handelt es sich, wenn kein Teil der Datenintegration semiautomatisch oder automatisch durchgeführt wird.

Einwurf In dieser Arbeit soll die Automatisierung der Datenintegration an der Automatisierung der Integration der Datenbestände gemessen werden. Im Bereich der Datenintegration mit Datenbanken ist auch die weitestgehende Automatisierung der Integration der Datenstrukturen (Datenbankschemata) ein wichtiges Thema.

4.8 Zusammenfassung

In diesem Kapitel wurde eine Übersicht über die Möglichkeiten der Datenintegration gegeben. Dabei wurden Abgrenzungen und Überschneidungen mit anderen Themen aufgezeigt und der Datenintegrationsprozess mit seinen Varianten vorgestellt. Im zweiten Teil der Arbeit wird die Datenintegration dann im Kontext des 3LGM²-Baukastens betrachtet.

5 Ansatzpunkte zur Datengewinnung

Inhaltsangabe

5.1 Computer Aided Facility Management 25
5.1.1 Facility Management 26
5.1.2 CAFM- und Buchhaltungssysteme 28
5.1.3 Zusammenfassung CAFM 28
5.2 Management verteilter Informationssysteme 28
5.2.1 Managementfunktionen 29
5.2.2 Managementobjekte 29
5.2.3 Standards für Informationsmodelle 30
5.2.4 Managementarchitekturen 30
5.3 IT Infrastructure Library 33
5.4 Software Asset Management 35
5.5 Zusammenfassung 36

Daten über Hard- und Softwarebestände, die für 3LGM²-Modelle genutzt werden können, finden sich in verschiedenen Softwareprodukte, deren Anzahl viel zu groß ist, als dass sie hier vollständig aufgezählt werden könnten. Die Softwareprodukte werden aber entsprechend ihres Einsatzzweckes in verschiedene Kategorien eingeteilt. Um im Weiteren konkrete Softwareprodukte richtig einordnen zu können, werden die einzelnen Kategorien in diesem Kapitel erläutert. Aspekte, die für die angestrebte Datenintegration von besonderem Interesse sind, wie der Inhalt der Daten, werden ausführlicher vorgestellt.

Im Bereich der betriebswirtschaftlich ausgerichteten Softwareprodukte werden *Computer-Aided-Facility-Management-Systeme* vorgestellt.

Im Bereich der operativen Aufgaben des Informations-Technik-Managements werden Systeme für das *Management verteilter Informationssysteme* vorgestellt.

Und im Bereich des IT Service Managements wird auf *IT Infrastructure Library* unterstützende Softwareprodukte eingegangen.

5.1 Computer Aided Facility Management

Als Computer-Aided-Facility-Managemen-Systeme (CAFM-Systeme) werden an die speziellen Bedürfnisse eines Unternehmens bzw. einer Branche angepasste Komplettlösungen zur Unterstützung der Prozesse des Facility Managements bezeichnet. (vgl. [GEFMA 400 02] S.1) Ein CAFM-System besteht dabei aus einem multifunktionalen Anwendungssystem oder einer

Sammlung monofunktionaler Anwendungssysteme. Für den Fall der Zusammensetzung aus monofunktionalen Anwendungssystemen müssen zwischen diesen Einzelsystemen Schnittstellen existieren, die wiederum für die Kopplung mit dem 3LGM²-Baukasten in Frage kommen. Multifunktionale CAFM-Systeme sind nicht darauf angewiesen, Schnittstellen bereit zu stellen. Nutzen multifunktionale CAFM-Systeme eine Datenbank, so kann versucht werden, diese als Schnittstelle zu nutzen.

Vor der Betrachtung von Schnittstellen soll zunächst ein Einblick in das Facility Management Klarheit darüber schaffen, welche Teilbereiche des FM es gibt und inwieweit diese für die Datenintegration relevant sind.

5.1.1 Facility Management

Definition Facility Management (FM) ist eine Managementdisziplin, die durch ergebnisorientierte Handhabung von Facilities und Services im Rahmen geplanter, gesteuerter und beherrschter Facility Prozesse eine Befriedigung der Grundbedürfnisse von Menschen am Arbeitsplatz, Unterstützung der Unternehmens-Kernprozesse und Erhöhung der Kapitalrentabilität bewirkt. Hierzu dient die permanente Analyse und Optimierung der kostenrelevanten Vorgänge rund um bauliche und technische Anlagen, Einrichtungen und im Unternehmen erbrachte (Dienst-) Leistungen, die nicht zum Kerngeschäft gehören. ([GEFMA 100-1 02] S.3)

Zwar sollten CAFM-Systeme auch versuchen, das FM mit all seinen Prozessen zu unterstützen, aber die große Zahl der Prozesse macht dies unmöglich. Aus diesem Grund unterstützen CAFM-Systeme oft nur einen Teilbereich des FMs. Allerdings besteht ein Konsens darüber, dass beispielsweise Hausmeistertätigkeiten oder Reinigung von Räumen allein noch kein FM sind.

Bestandteile des Facility Managements

Teile des FMs, die sich mit spezifischen Facility Prozessen auseinander setzen, sind beispielsweise: Liegenschaftsmanagement, Flächenmanagement, Inventarisierung, Raumvergabe, Kostenmanagement, Accessmanagement, Reinigungsmanagement, Umzugsmanagement, Personalmanagement, Vertragsmanagement, Dokumentenmanagement, Rohrleitungs- und Kabelmanagement, Anlagendatenerfassung, Brandschutz, Instandhaltungsmanagement, Störungsmanagement, Informations- und Kommunikationstechnik Management, Energiemanagement, Arbeitssicherheit und Umweltschutz. (vgl. [PERSON 01] S.4)

Die obige Auflistung verdeutlicht, dass FM per Definition Überschneidungen mit vielen anderen Bereichen hat, die sich sonst als eigenständig sehen. So lassen sich Netzwerkmanagementsysteme in den FM-Bereich Informations- und Kommunikationstechnik Management einordnen. Mit der Inventarisierung enthält das FM einen Teilbereich, der relevant für die angestrebte Datenintegration scheint.

Die angestrebte Datengewinnung setzt voraus, dass die für 3LGM²-Modelle relevanten Elemente im Inventar beschrieben sind. Dem wirkt entgegen, dass die Granularität der Inventarisierung nicht zu klein sein sollte, da sonst die Pflege der Daten sehr aufwändig wird. (vgl. [PERSON 01] S.6) Das hat zur Folge, dass in der Inventarisierung nicht zwangsläufig für den 3LGM²-Baukasten relevante Daten vorhanden sind. Allerdings ist die Granularität der Inventarisierung nicht nur davon abhängig, wie viel Arbeitszeit und Aufwand investiert werden soll, sondern auch von gesetzlichen Bestimmungen. Wird die Inventarisierung zur Erstellung von Bilanzen herangezogen, so muss sie den gesetzlichen Vorgaben entsprechen. Um welche Vorgaben es sich dabei handelt, soll im nächsten Abschnitt geklärt werden.

Inventar

Der Begriff des Inventars ist im Handelsgesetzbuch (HGB) § 240 „Inventar“ zu finden

> (1) Jeder Kaufmann hat zu Beginn seines Handelsgewerbes [...] Vermögensgegenstände genau zu verzeichnen [..].
>
> (2) Er hat demnächst für den Schluß eines jeden Geschäftsjahrs ein solches Inventar aufzustellen. [...]
>
> (3) [...] Jedoch ist in der Regel alle drei Jahre eine körperliche Bestandsaufnahme durchzuführen.
>
> (4) Gleichartige Vermögensgegenstände [...] können jeweils zu einer Gruppe zusammengefaßt [...] werden.

Diese Regelung ist für die Datengewinnung des 3LGM²-Baukastens in so fern günstig, dass jedes bilanzierende Unternehmen vollständig inventarisiert werden muss und damit auch sein Informationssystem. Leider wird die Granularität der Inventarisierung nicht vorgegeben, so dass durch Abstraktion und Gruppenbildung die interessanten Informationen über das Informationssystem verloren gehen können.

In diesem Zusammenhang ist auch bedeutsam, dass es neben dem Begriff des Inventars noch das Anlagevermögen gibt, welches in einer Bilanz aufgeführt werden muss. § 247 des HGB zum „Inhalt der Bilanz“:

> (1) In der Bilanz sind das Anlage- und das Umlaufvermögen [...] gesondert auszuweisen und hinreichend aufzugliedern.
>
> (2) Beim Anlagevermögen sind nur die Gegenstände auszuweisen, die bestimmt sind, dauernd dem Geschäftsbetrieb zu dienen.
>
> [...]

Damit ist das Inventar eine Obermenge des Anlagevermögens. Wie in der Abbildung 5.1 deutlich wird, beinhaltet Inventar eigentlich etwas mehr als alle Gegenstände, die sich in einem Gebäude befinden. Mit Inventar wie es im Zusammenhang mit CAFM-Systemen gebraucht wird, ist teilweise Anlagevermögen gemeint. Je nach Autor wird auch von Anlagen gesprochen und nicht von Inventar wie beispielsweise in [HEERTEN 96] S.8.

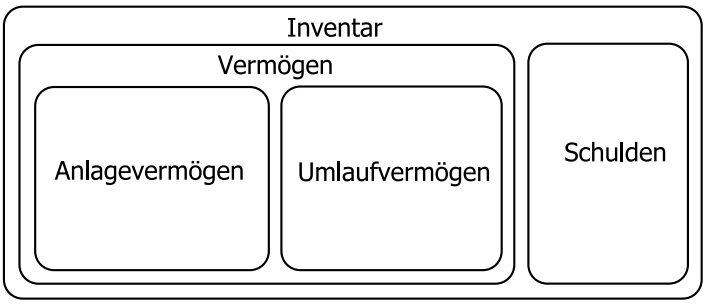

Abbildung 5.1: Zusammensetzung des Inventars: Grundstücke und Gebäude, Maschinen, Fuhrpark und Geschäftsausstattung gehören zum Anlagevermögen. Verbrauchsmaterialien, Bankguthaben, Bargeld und anderes gehören zum Umlaufvermögen. Anlagevermögen und Umlaufvermögen zusammen bilden das Vermögen.

5.1.2 CAFM- und Buchhaltungssysteme

Das Anlagevermögen wird wiederum in der Anlagenbuchhaltung benötigt, die nicht Teil des FM ist. Um Redundanz zu vermeiden, sollte die Anlagenbuchhaltung natürlich nicht ihren eigenen Anlagendatenbestand halten. Vielmehr sollte sie die Inventardaten des CAFM nutzen. Allerdings ist die EDV-Unterstützung der Buchhaltung oftmals höher als die des FM, so dass es durchaus sein kann, dass das CAFM die vorhandenen Daten der Buchhaltung nutzt indem es diese dupliziert.

Es wird auch diskutiert, inwieweit Standardanwendungen wie für das Rechnungswesen CAFM Systeme ersetzen können. Die Firma SAP bietet beispielsweise für ihre Unternehmenssoftware R/3 ein Modul RE (Immobilienmanagement) an.

Welche Form der Verwaltung von Inventar/Anlagen nun die bessere ist, soll nicht Gegenstand dieser Arbeit sein. Der obige Abschnitt soll aber zeigen, dass es „gewachsene“ Lösungen gibt und der Begriff Inventar keine Garantie für eine gewisse Granularität ist.

5.1.3 Zusammenfassung CAFM

Es ist nicht möglich, mit Sicherheit zu sagen, dass CAFM-Systeme Daten enthalten, die sinnvoll zu integrieren sind. Leider gilt gleiches auch für die kurz angesprochenen Buchhaltungssysteme und ihre Anlagendaten. Im Einzelfall jedoch können derartige Softwaresysteme relevante Daten enthalten.

5.2 Management verteilter Informationssysteme

Eine weitere Klasse von Softwareprodukten, die Informationen über ein Informationssystem enthalten, sind die Managementsysteme, die sich nur mit den IT-Netzwerken und allen angeschlossenen Geräten befassen. Management vernetzter Systeme umfasst in seiner allgemeinsten Definition alle Maßnahmen, die einen effektiven und effizienten, an den Zielen des Unter-

nehmens ausgerichteten Betrieb der Systeme und ihrer Ressourcen sicherstellen. Es dient dazu, die Dienste und Anwendungen des vernetzten Systems in gewünschter Güte bereitzustellen und ihre Verfügbarkeit zu gewährleisten. (vgl. [HEGERING et al. 99] S.2)

5.2.1 Managementfunktionen

Dabei untergliedert sich das Management verteilter Systeme in verschiedene Managementfunktionen. Die Wesentlichen davon sind:

- Konfigurationsmanagement
- Fehlermanagement
- Leistungsmanagement
- Abrechnungsmanagement, Benutzerverwaltung
- Sicherheitsmanagement

Später wird das Monitoring von Netzwerken, welches im Wesentlichen zum Fehlermanagement gehört, noch eine Rolle spielen. Das Monitoring ist damit eine Teildisziplin des Managements verteilter Informationssysteme und überwacht das Informationssystem.

Die für den 3LGM²-Baukasten relevanten Informationen sind bei dieser Einteilung Teil des Konfigurationsmanagements.

Bei [TERPLAN 95] S.57 wird die zusätzliche Funktionalität des Bestandsmanagements eingeführt. Das Bestandsmanagement schließt seinerseits die Funktionen Bestandsführung, Archivierung, Backup, Änderungsdienst und Bestellwesen ein. Aber unabhängig davon, ob von Bestandsmanagement oder von Konfigurationsmanagement gesprochen wird, ist ein Datenbestand über die Bestandteile des Informationssystems und deren Zusammenhang nötig, um die restlichen oben genannten Funktionalitäten unterstützen zu können.

Leider sind aber auch diese Managementsysteme nicht daran gebunden, die Informationssysteme vollständig zu beschreiben. Teile von Informationssystemen, die nicht überwacht werden müssen oder deren Überwachung nur einen geringen Nutzen hat, könnten zur Vereinfachung aus der Konfiguration entfernt werden. Weiterhin tauchen auch diejenigen Teile des Informationssystems nicht auf, die keine Verbindung mit dem Netzwerk haben.

5.2.2 Managementobjekte

Die Informationseinheit, mit der beim Management von verteilten Systemen umgegangen wird, sind die Managementobjekte (MO), die Charakteristika einer Ressource darstellen. Ressourcen von verteilten Systemen sind dabei Übertragungsmedien und Komponenten sowie Endsysteme. (vgl. [HEGERING et al. 99] S.101) Leider ist der Inhalt eines Managementobjekts sehr von der verwendeten Managementarchitektur abhängig. Während beim OSI-Management ein MO durchaus eine reales Element des Informationssystems sein kann, wie beispielsweise ein Netzwerkdrucker, so sind MOs im Internet-Management (SNMP) die Parameter dieser Elemente, z. B. `Anzahl_gedruckter_Seiten`. Dies ist wichtig für die nächsten Abschnitte, in denen einige Managementarchitekturen hinsichtlich ihrer Beschreibung solcher Managementobjekte und damit letztlich der Informationssystemelemente untersucht werden.

Zwei weitere wichtige Begriffe sind „Manager“ und „Agent“. Der Manager ist dabei das Softwaresystem, das die Aufgaben des Datensammelns, Datenverteilens und der Datenanalyse hat. Der Agent hingegen ist ein Teil des zu managenden Elements und stellt dem Manager eine Schnittstelle zum Element zur Verfügung.

5.2.3 Standards für Informationsmodelle

Das Informationsmodell einer Managementarchitektur ist sozusagen das Metamodell für die Repräsentation des zu managenden Informationssystems. Oder mit anderen Worten, das Informationsmodell gibt vor, wie das IS beschrieben wird. Es gibt mehrere Informationsmodelle, von denen hier einige zusammen mit ihren Managementarchitekturen vorgestellt werden sollen.

5.2.4 Managementarchitekturen

5.2.4.1 Open Systems Interconnection (OSI) Managementarchitektur

(vgl. [HEGERING et al. 99] S.113ff)

Die OSI-Managementarchitektur wird gern aufgrund ihres Referenzcharakters zum Vergleich mit anderen Architekturen herangezogen. Sie gilt als sehr komplex und findet ihre praktische Anwendung insbesondere im Telekommunikationsbereich.

Das im ISO-Standard (International Organization for Standardization) definierte Informationsmodell der OSI-Architektur wird auch als Structure of Management Information (SMI) bezeichnet und ist vollständig objektorientiert. Dabei sind die Managementobjekte Instanzen von Managementobjektklassen (MOC), die in der Management Information Base (MIB) instanziiert wurden. Zu jeder Managementobjektklasse gibt es noch eine Managed Object Boundary, die Attribute, Operationen, Meldungen und Beschreibungen von Verhalten umfasst. Die Managed Object Boundary stellt damit eine Abstraktion dar. Auch Polimorphie ist erlaubt, was zu einer komplexen Vererbungshierarchie führen kann. Die Managementobjektklassen, werden mit Hilfe von Templates beschrieben, die wiederum mit der Nachrichtenbeschreibungssprache Abstract Syntax Notation One (ASN.1) erstellt wurden. Ohne jetzt weiter auf die Erstellung einer Managementobjektklasse einzugehen, könnten die entstehenden Managementobjekte selbst, ihre Attribute oder Methoden für ein 3LGM²-Modell nützlich sein.

Weiterhin wird für die OSI-Managementarchitektur die Kommunikation zwischen dem Managementsoftwaresystem (dem sogenannten Manager) und den gemanagten Informationssystemelementen (vertreten durch einen Agenten, der das Protokoll versteht) beschrieben. Dabei ist wesentlich, dass der Manager Kenntnis darüber hat, welche Informationen er bei einem bestimmten Agenten abfragen kann. Die Management Information Base enthält genau diese Informationen und hilft damit dem Manager, nur relevante Abfragen an den Agenten zu senden und dem Agenten, die Anfragen des Managers richtig zu interpretieren.

Management Information Base und deren Informationsgehalt Die MIB für das OSI-Management muss die Beschreibungen aller Managementobjekte beinhalten, die vom Managementsys-

tem genutzt werden sollen. Dabei definiert die MIB die Managementobjekte, die im konkreten System vorhanden sind. Diese wiederum beschreiben, welche Attribute und Methoden von den Informationssystemelementen zur Verfügung stehen. Durch eine Baumstruktur der MIB wird erreicht, dass jedes MO durch seinen Pfad eindeutig referenziert wird. Die Beziehungen in diesem Management Information Tree (MIT) sind Enthaltenseinsbeziehungen, weshalb diese Hierarchie auch als *Enthaltenseinsbaum* bezeichnet wird. Dieser hat keinen Zusammenhang zum weiter unten erläuterten Vererbungsbaum oder dem Registrierungsbaum und wird nur durch die Namensgebung bei der Instanziierung eines MOs gebildet. Durch ihn wird beispielsweise ausgedrückt, dass ein Festplatten-MO Teil eines PC-MOs ist. Eine solche MIB kann mit Sicherheit viele Informationen über ein IS bieten, ist aber evtl. sehr feingranular.

Der *Registrierungsbaum* ist ein Teilbaum des OID-Tree (Object Identifier), der durch Normen ISO/IEC 9834 und DIN 66334 standardisiert ist und dezentral verwaltet wird. Im OID-Tree werden allen darin aufgeführten Objekten weltweit eindeutige IDs in Form von Nummernfolgen zugeordnet. Hierin sind auch die Templates für die Definition von MOCs enthalten. Ein ähnliches Verfahren wird beim Internet Management angewendet (siehe Kapitel 5.2.4.3). Templates können in diesem Zusammenhang Managed Object Class, Package, Parameter, Attribute usw. sein. Eine MOC wird aus mehreren dieser Templates, aber mindestens aus einem MOC-Template, zusammengesetzt.

Interessant für den 3LGM²-Baukasten sind weniger die Blätter dieses Baums, die Templates, sondern die inneren Knoten, die Verallgemeinerungen der Blätter und damit der Templates darstellen. Ein Beispiel für einen MOC-Template-Pfad:

`iso(1).std(0).iso8802(8802).csma(3).hubmgt(18).objectclass(0).hubobjectClass(X)`

Dieses MOC-Template beschreibt eine Hub-MOC. Leider sind die inneren Knoten (iso8802, csma, hubmgt, objectclass) zwar Verallgemeinerungen, aber sehr technisch orientiert und somit als Kriterium für das Zusammenfassen schlecht geeignet.

Unabhängig davon gibt es noch die *Vererbungshierarchie* der MOCs, die sich aus den MOC-Definitionen ablesen lässt. Inwieweit die Vererbungshierarchie der Klassen für den 3LGM²-Baukasten sinnvoll ist, ist schwer zu sagen, da die Definitionen der Klassen herstellerspezifisch sind.

Die drei vorgestellten Bäume des OSI-Managements, *Registrierungsbaum*, *Vererbungsbaum* und *Enthaltenseinsbaum*, enthalten daher mehr oder weniger ausgeprägte Hierarchien, aus denen man Informationen über die Verallgemeinerung von MOs ableiten kann.

Ferner haben MOs auch die Möglichkeit, auf andere MOs über Attribute zu verweisen, was eine zusätzliche Information über ein IS darstellt.

5.2.4.2 Common Object Request Broker Architecture

(vgl. [HEGERING et al. 99] S.177ff)

Die Common Object Request Broker Architecture (CORBA) ist keine ausgezeichnete Managementarchitektur, sondern ist für die Entwicklung unterschiedlichster verteilter Anwendungen gedacht, weswegen hier auch nicht näher auf die Funktionsweise von CORBA eingegangen werden soll. Diese Architektur ist aber interessant, da viele Managementsysteme verteilte Sys-

teme sind und den CORBA Ansatz nutzen. Die Verwendung von CORBA als Managementarchitektur würde somit zu einer Vereinheitlichung führen.

Aufgrund des allgemeinen Ansatzes von CORBA werden die Managementobjekte mit der Objektbeschreibungssprache Interface Description Language (IDL) definiert. Dabei wird zwischen Schnittstellen und Objekten unterschieden, wobei die Schnittstellen nur die Signaturen von Operationen enthalten dürfen und keine Operationen selbst. Es gibt *Vererbungsbeziehungen* zwischen Schnittstellen untereinander und zwischen Objekten und Schnittstellen. Eine managementspezifischere Beschreibung für Managementobjekte gibt es im Moment nicht.

Auch hier ergeben sich Möglichkeiten, die Vererbungsbäume zur Verallgemeinerung und zum Zusammenfassen für ein 3LGM²-Modell zu nutzen. Aber auch hier lässt sich nur an konkreten Klassen prüfen, ob dies sinnvoll ist oder nicht.

5.2.4.3 Simple Network Management Protocol (SNMP)

(vgl. [HEGERING et al. 99] S.143ff)

Das Simple Network Management Protocol (SNMP) wurde von der Internet Engineering Task Force (IETF) entwickelt und wird daher manchmal als Internet-Management bezeichnet. Im Gegensatz zu den bisher betrachteten objektorientierten Ansätzen, ist das Management mittels SNMP deutlich einfacher aufgebaut. Dies hat sicherlich mit dazu beigetragen, dass diese Variante heute sehr verbreitet ist und schon von sehr einfachen und günstigen Geräten unterstützt wird.

Die Managementsoftware kommuniziert dabei mit Agenten, die Teil der überwachten Komponenten sind. Die Agenten sind in der Lage, komponentenspezifische Informationen auszulesen, oder zu ändern und im Fehlerfall Fehlernachrichten zu versenden.

Management Information Base Ebenso wie beim OSI-Management gibt es auch beim Internet Management eine Management Information Base (MIB), in der Manager und Agent Identifikationen von MOs nachschlagen können. Im Gegensatz zum OSI-Management steht, dass die MOs keine Objekte mit Attributen sind, sondern die Attribute selbst. Die Sicht des Internet-Managements auf die verwalteten Elemente ist daher deutlich einfacher. Dennoch ist die MIB nicht flach, sondern in einer baumartigen Hierarchie angeordnet, in der nur die Blätter MOs darstellen. Es gibt dabei keinen *Vererbungsbaum* und keinen *Enthaltenseinsbaum*, aber wie auch beim OSI-Management die inneren Knoten des *Referenzbaums*, die Verallgemeinerungen der Blätter sind. Das Verfahren zur Identifizierung eines MOs ist ähnlich dem Verfahren des OSI-Managements, allerdings befinden sich die SNMP-MOs in anderen Teilbäumen des OID-Trees als die OSI-MOs. Ein Beispiel:
`iso(1).identified-organization(3).dod(6).internet(1).private(4).enterprise(1).cisco(9).temporary variables(3).AppleTalk(3).atInput(1)`
Dieser Pfad beschreibt ein MO, welches die Anzahl der eingegangenen AppleTalk-Pakete eines Routers enthält. Leider zeigt sich aber auch hier, dass die inneren Knoten dieses Baumes nicht sehr viel verwendbare Information für den 3LGM²-Baukasten enthalten.

5.3 IT Infrastructure Library

IT Infrastructure Library (ITIL) ist eine Sammlung von „Best Practice"-Leitfäden für das IT Servicemanagement. ITIL wurde vom Office of Government Commerce (OGC) in Zusammenarbeit mit dem IT Service Management Forum (itSMF) erstellt und setzt sich aus mehreren Publikationen zusammen, die sich mit serviceorientiertem Betrieb von IT beschäftigen. Dabei ist ITIL allgemein anerkannt und stellt einen De-Facto-Standard für das IT-Management dar. (vgl. [ELSÄSSER et al. 05] S.6) Im Moment besteht die ITIL Bibliothek unter anderem aus folgenden Büchern:

- Introduction to ITIL
- Service Support
- Service Delivery
- Planning to Implement Service Management
- ICT Infrastructure Management
- Application Management
- Security Management
- Software Asset Management
- Business Perspective
- ITIL Small-Scale Implementation

Service Support und Service Delivery sind dabei die beiden grundlegenden Bücher von ITIL, wobei sich Service Support eher mit operationalen und Service Delivery mit den strategischen Prozessen auseinandersetzt. Fünf ITIL-Basisprozesse werden für den Service Support beschrieben.

- Incident Management
- Problem Management
- Configuration Management
- Change Management
- Release Management

Hinzu kommt der Service Desk, der die Schnittstelle zwischen IT-Organisation und Anwender bildet, der aber eine Funktion und kein Prozess ist. All diese Basisprozesse benötigen Informationen über den Aufbau und die Zusammensetzung der IT-Infrastruktur, um ihre Managementaufgaben erfüllen zu können. Besonders deutlich wird dies am Change und Configuration Management.

Change Management überwacht den Lebenszyklus einer Änderung und beinhaltet das Initiieren, Einschätzen, Rechtfertigen, Planen und Prüfen eines Änderungsvorschlages.

Configuration Item und Configuration Management Database

Configuration Management hat zur Aufgabe Configuration Items (CI), die für IT Services benötigt werden, aktuell zu halten, einschließlich deren Beziehungen und diese CIs zur Verfügung

zu stellen, wann immer sie benötigt werden. Die CIs befinden sich dazu in einer Configuration Management Database (CMDB). Einen Vorschlag für die Attribute eines CIs finden sich im Buch für ITIL Service Support. (vgl. [SAGER 05] S.52) Die Attribute dieses Vorschlags sind in der Tabelle 5.1 aufgelistet.

Tabelle 5.1: CI-Attribute für eine CMDB (vgl. [SAGER 05] S.52)

Attribut	Beschreibung
CI Name	eindeutiger Name des CIs (ITIL schreibt fälschlicherweise „des CI-Typs“)
Copy or Serial Number	eindeutige CI-ID, beispielsweise bei Software die Copy Number oder bei Hardware die Seriennummer
Type	Beschreibung des CI-Typs zur Detaillierung der Category, beispielsweise Hardware-Konfiguration, Software-Package, Hardware-Device oder Programmmodul
Model Number (Hardware)	Modellnummer des CIs, beispielsweise übereinstimmend mit der Modellnummer des Herstellers
Warranty Expiry Date	Ablaufdatum der Herstellergarantie
Version Number	Versionsnummer des CIs
Location	Ortsinformation des CIs, beispielsweise das Medium oder die Bibliothek, in der ein Software-CI gespeichert ist, die Adresse oder der Raum, in dem ein Service bereitgestellt wird
Owner Responsible	Name und/oder die Kontaktdaten des CI-Verantwortlichen
Responsibiliy Date	Datum, an dem die Verantwortung übertragen wurde
Source/supplier	Herkunft eines CIs, beispielsweise „intern bereitgestellt“, „extern eingekauft“
Licence	Lizenznummer oder Referenz auf das Lizenzabkommen
Supply Date	Datum, an dem das CI in der Organisation bereitgestellt wurde
Accepted Date	Datum, an dem das CI als erfolgreich getestet in der Organisation aufgenommen wurde
Status (current)	derzeitiger Status des CIs, beispielsweise „test“, „live“, „archived“
Status (scheduled)	nächster, geplanter CI-Status (mit Datum oder Angabe des Ereignisses, welches den Statuswechsel anzeigt)
Parent CI(s) relationships	eindeutige CI-IDs der „parent(s)“ dieses CIs
Child CI(s) relationships	eindeutige CI-IDs aller „children“ des CIs

Relationships	alle anderen Beziehungen über „parents“ und „children“ hinaus (beispielsweise CI „nutzt“ anderes CI, CI „ist verbunden“ mit anderem CI, CI „ist enthalten auf“ anderem CI, CI „kann zugreifen auf“ anderes CI)
RFC Numbers	Schlüssel aller RFCs, die sich auf das CI beziehen
Change Numbers	Schlüssel aller Changes, die sich auf das CI beziehen
Problem Numbers	Schlüssel aller Probleme, die sich auf das CI beziehen
Incident Numbers	Schlüssel aller Incidents, die sich auf das CI beziehen
Comment	ein Kommentarfeld für freie Einträge, beispielsweise darüber, wie sich die CI-Version von einer Vorgängerversion unterscheidet

Leider stellt diese Tabelle wie alles in ITIL nur eine Empfehlung dar. Das bedeutet, dass man keine Erwartungen an eine Datenbank stellen kann, die als CMDB bezeichnet wird. Es gibt auch Bestrebungen, Datenbanken von CAFM Systemen (siehe Kapitel 5.1) durch geringe Erweiterungen zu CMDBs umzuwandeln. Inwieweit derartige Datenbestände dann die ITIL-Prozesse unterstützen können, bleibt fraglich. Eine CMDB, die den Vorgaben entspricht oder diese sogar noch erweitert, kann für ein 3LGM²-Modell relevante Daten liefern. Beispielsweise können über bestimmte *Typen* CIs gefunden werden, die Server repräsentieren und mit Hilfe der *Relationships* deren topologische Einbindung nachvollzogen werden.

5.4 Software Asset Management

Software Asset Management (SAM) bezeichnet eine Reihe von Geschäftsprozessen, die nötig sind, um den Softwarebestand eines Unternehmens in allen Lebenszyklusphasen zu verwalten, zu kontrollieren und zu schützen.(vgl. [MICROSOFT 06]) Die Notwendigkeit von SAM wird meist damit begründet, dass Unternehmen sehr wenig über ihre Software Assets, wie Software- und Lizenzbestände in diesem Zusammenhang genannt werden, im Verhältnis zu anderen Wertgegenständen wissen. Das kann dazu führen, dass zu wenige oder zu viele Lizenzen in einem Unternehmen existieren, was im ersten Fall rechtlich bedenklich und im zweiten unwirtschaftlich ist. SAM kann als eigenständige Disziplin oder im Zusammenhang mit ITIL betrachtet werden.

Am 22. Mai 2006 wurde der ISO/IEC Standard für das Software Asset Management (SAM) verabschiedet. (vgl. [ISO 06]) Im ersten Teil von ISO/IEC 19770 werden die Prozesse des Risk Managements, der Kostenkontrolle und des Competitive Advantage definiert. Erst im Teil zwei soll eine Anwendung beschrieben werden, die den Software-Inventarisierungsprozess vereinfacht. Der Teil zwei ist zum Zeitpunkt dieser Arbeit noch nicht veröffentlicht.

Im Zusammenhang mit ITIL sollten Software Assets mit in die CMDB aufgenommen werden und über diese für die Verwendung in 3LGM²-Modellen zur Verfügung stehen. Wird SAM außerhalb von ITIL verwendet, ist allerdings unklar, inwieweit die Daten für den 3LGM²-Bau-

kasten erreichbar sind. Allerdings hat SAM den Vorteil, dass die dafür benötigten Daten über Softwarebestände zwangsläufig vollständig sein sollten.

5.5 Zusammenfassung

Die Betrachtung der Kategorien von Softwareprodukten hat gezeigt, dass es durchaus schon auf dieser Ebene Ansatzpunkte für die Gewinnung strukturierter Daten gibt. Leider sind einige dieser Ansatzpunkte für die Nutzung in 3LGM²-Modellen ungeeignet oder zu unverbindlich, so dass die Betrachtung von Einzelprodukten nicht vermeidbar ist.

Teil II

Rahmenbedingungen für die Datenintegration

6 Datengewinnung aus Softwareprodukten

Inhaltsangabe

6.1 DX-Union Asset Assistant 39
6.2 IBM Tivoli NetView 41
6.2.1 Tivoli Data Warehouse 42
6.2.2 Tivoli Enterprise Console 42
6.3 HP OpenView 42
6.4 OpenNMS 42
6.5 Topologieerkennung 43
6.6 Zusammenfassung 44

In diesem Kapitel sollen konkrete Softwareprodukte hinsichtlich ihrer Qualitäten als Datenquelle untersucht werden, da die Betrachtung auf Ebene der Kategorien von Softwareprodukten nicht zufriedenstellend war. Dazu muss zunächst geklärt werden, welche Informationen diese enthalten und beispielhaft welche Wege es gibt, diese Daten zu nutzen. Dabei wird DX-Union Asset Assistant ein Vertreter der Klasse der *Software Asset Management Systeme* sein und IBM Tivoli NetView, HP OpenView und OpenNMS sind Vertreter für das *Management verteilter Informationssysteme*. Es soll sich im Folgenden nur um einen Überblick handeln, eine detailliertere Darstellung wie sie noch für Nagios vorgenommen wird, kann hier nicht erfolgen.

6.1 DX-Union Asset Assistant

Das Produkt *DX-Union Olympia* der Firma Materna ist ein Softwarepaket für ein ganzheitliches Workplace Management und umfasst verschiedene Kernfunktionen. Die Inventarisierung ist eine dieser Kernfunktionen und wird durch die Software DX-Union Asset Assistant realisiert. Der Asset Assistent inventarisiert PCs, Drucker, Server und andere Netzwerkkomponenten. Dabei werden auch Veränderungen an diesen Geräten protokolliert. Es können ca. 7500 Werte abgefragt werden, die aber nicht jedes Gerät zur Verfügung stellt. Einige Beispiele, für Werte die gesammelt werden können, sind:

Motherboard, BIOS, Bussystem, Prozessor, Speicher, Schnittstellen, Slots, Diskette, Festplatte, CD-ROM/DVD, Logische Laufwerke, Soundkarte, Grafikkarte, Netzwerkadapter, Controller, Plug&Play Hardware, Tastatur, Maus, Monitor, Drucker, Freigaben, Betriebssystem, Patches, installierte Software, Treiber, Dienste, Prozesse

Die dafür genutzten Schnittstellen sind zum einen die Microsoft-spezifische Windows Management Instrumentation Schnittstelle (WMI) und für alle anderen Geräte SNMP.

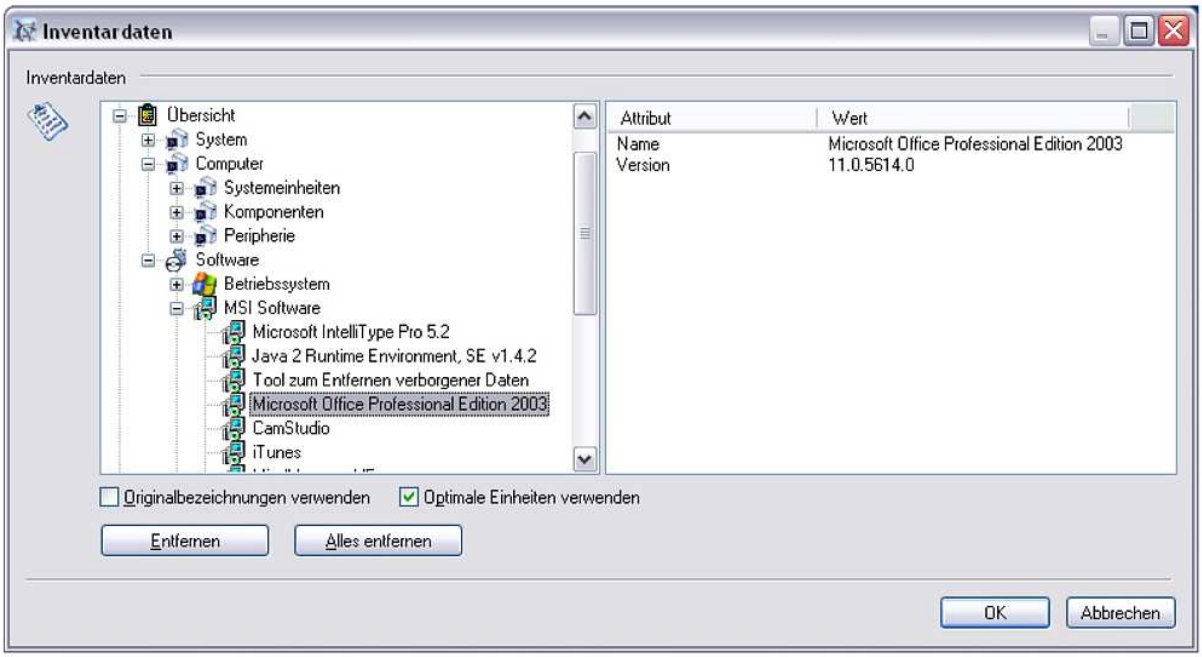

Abbildung 6.1: Ein Screenshot des Asset Assistants

Die gesammelten Daten werden dann in einer CMDB[1] abgelegt, auf die mittels Lightweight Directory Access Protocol (LDAP) Version 2 zugegriffen werden kann. Es können über LDAP auch Änderungen in der CMDB vorgenommen werden. Eine Auflistung der Attribute dieser Datenbank war aufgrund der großen Anzahl nicht zu erhalten. Sie wird sich aber an den Möglichkeiten des WMI und SNMP Standards orientieren.

Eine weitere Verwendung findet der Asset Assistent als integriertes Fremdprodukt in der Software „BMC Remedy Action Request System“. Die Daten werden dann in der dazugehörigen Datenbank abgelegt. Auf diese Datenbank kann dann zusätzlich zu den genannten Methoden mittels Structured Query Language (SQL) zugegriffen werden.

Es existiert noch eine Kommandozeilenschnittstelle[2], die eine eigene Syntax besitzt und ebenfalls genutzt werden kann. Die CLI-Schnittstelle hat beim schreibenden Zugriff den Vorteil, dass Semantiküberprüfungen durchgeführt werden.

In den Daten der Datenbanken lassen sich aber leider keine Hinweise darauf finden, welche Beziehungen zwischen den einzelnen inventarisierten Komponenten bestehen. In einer überarbeiteten Version wird der Asset Assistent zwar in der Lage sein, Netzwerke nach inventarisierbaren Komponenten abzusuchen, allerdings ohne die Netzwerktopologie selbst zu erfassen.

In Abbildung 6.1 ist die Programmoberfläche des DX-Union Asset Assistants zu sehen.

[1] Es gibt unterschiedliche Ansichten, ob diese Datenbank eine CMDB ist oder nicht.
[2] in diesem Zusammenhang als Command Line Interface (CLI) bezeichnet

6.2 IBM Tivoli NetView

Unter der Bezeichnung „Tivoli“ bietet IBM eine Reihe von Produkten an, die zum Steuern und Überwachen von IT-Infrastruktur dient. Tivoli war ehemals eine eigenständige Firma, die von IBM 1996 aufgekauft wurde.

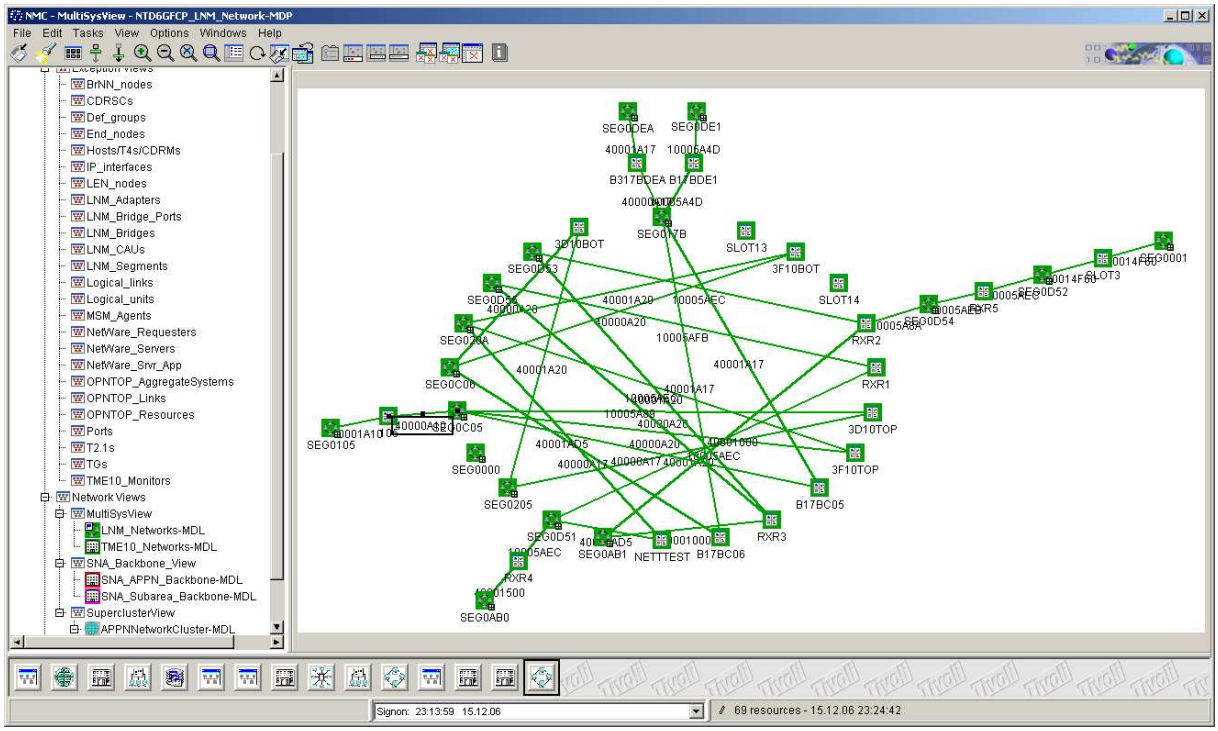

Abbildung 6.2: Die NetView Management Console ist eine grafische Benutzerschnittstelle zum NetView System

„Tivoli NetView“ ist die Bezeichnung des hier betrachteten Produkts, welches zur proaktiven Verwaltung von Netzwerkressourcen genutzt wird. „Proaktiv“ soll in diesem Zusammenhang das vorausschauende Erkennen von Netzwerkproblemen verdeutlichen. NetView ist eine verteilte Lösung. Das bedeutet, es kann mehrere lokal zuständige NetView Anwendungen geben, die untereinander kommunizieren. Eine weitere Fähigkeit von NetView ist das Darstellen von Netzwerktopologien, siehe Abbildung 6.2. Die einzelnen Tivoli-Produkte basieren auf einer CORBA Architektur, über die sie untereinander kommunizieren.

Das Tivoli Framework arbeitet in sogenannten Tivoli Management Regions (TMR), die Tivoli-Verwaltungsbereiche darstellen. Dabei kann ein Netzwerk zu einer TMR gehören (single TMR) oder in mehrere aufgeteilt sein (multi TMR). Eine TMR hat als zentralen Bestandteil den TMR-Server (auch Tivoli-Server), der die vollständige Tivoli-Software enthalten sollte oder diese zumindest referenziert. Der Tivoli-Server enthält zusätzlich die komplette objektorientierte Objektdatenbank. Zusätzlich ist die Datenbank über alle Managed Nodes, d.h. überwachten Komponenten, verteilt.

Der „Tivoli Integration Pack for NetView“ ermöglicht es zudem, Daten der Tivoli Anwendungen Inventory und Netview zu integrieren. Inventory ist ein Produkt, das Daten über die

Komponenten des Netzwerks sammeln soll, ähnlich wie der Asset Assistant aus Kapitel 6.1. (vgl. [COOK et al. 99] S.43)

6.2.1 Tivoli Data Warehouse

Das Tivoli Data Warehouse wird nicht als eigenes Produkt vermarktet oder verkauft, sondern ist in den anderen Tivoli Produkten enthalten. Ziel des Data Warehouse ist es, insbesondere historische Daten der restlichen Tivoli Produkte zu sammeln, um darauf Analysen durchführen zu können, die wiederum Geschäftsentscheidungen beeinflussen. Dazu werden die Daten der einzelnen Tivoli Anwendungen in einer DB2 Datenbank als Data Warehouse zusammengefasst. (vgl. [MANOEL et al. 04] S.167)

6.2.2 Tivoli Enterprise Console

Die Tivoli Enterprise Console vereint betriebswirtschaftliche Daten mit Daten des Netzwerkmanagements. Dazu wird unter anderem das Produkt Tivoli NetView in die Enterprise Console integriert.

Damit sind sowohl die Tivoli Enterprise Console sowie das Tivoli Data Warehouse als auch die verteilte Datenbank der TMR eine Möglichkeit für den 3LGM²-Baukasten, auf die vom Tivoli System erhobenen Daten zuzugreifen.

6.3 HP OpenView

Unter der Bezeichnung „HP OpenView“ fasst die Firma HP eine Menge von Werkzeugen zusammen, um Geschäftsprozesse zu überwachen, Leistung von Anwendungen zu steigern, Kontrolle über IT-Infrastrukturen zu erreichen und Unternehmensentscheidungen zu unterstützen. (vgl. [HP 06] S.4)

Ein interessantes Produkt aus dieser Plattform ist der HP OpenView Network Node Manager (NNM), der ebenfalls mit Hilfe von SNMP Netzwerke überwachen kann. Da er wie die IBM Tivoli Produkte für den Einsatz in größeren Netzwerken gedacht ist, kann er Netzwerke selbstständig inventarisieren und die Topologie abbilden. Diese Daten sowie die Zustände der Netzwerkkomponenten werden in einer Datenbank abgelegt, die zugänglich ist.

6.4 OpenNMS

Das Projekt openNMS bezeichnet sich als erste Netzwerk Management Plattform für größere Unternehmen, die unter der GPL veröffentlicht ist. OpenNMS nutzt SNMP für den Informationsaustausch mit den überwachten Elementen. Die Elemente des Netzwerks werden nicht bestimmt, sondern „entdeckt“, indem openNMS einen bestimmten IP-Adressbereich ständig nach neuen Netzwerkkomponenten (Nodes) durchsucht. Die Topologie wird aus den Informationen abgeleitet, die beispielsweise in Switches oder Routern vorhanden sind und über SNMP ausgelesen werden können. OpenNMS erkennt auch selbständig Netzwerkdienste und

überwacht diese wie in Abbildung 6.3 dargestellt. Alle Daten werden in einer Datenbank abgelegt.

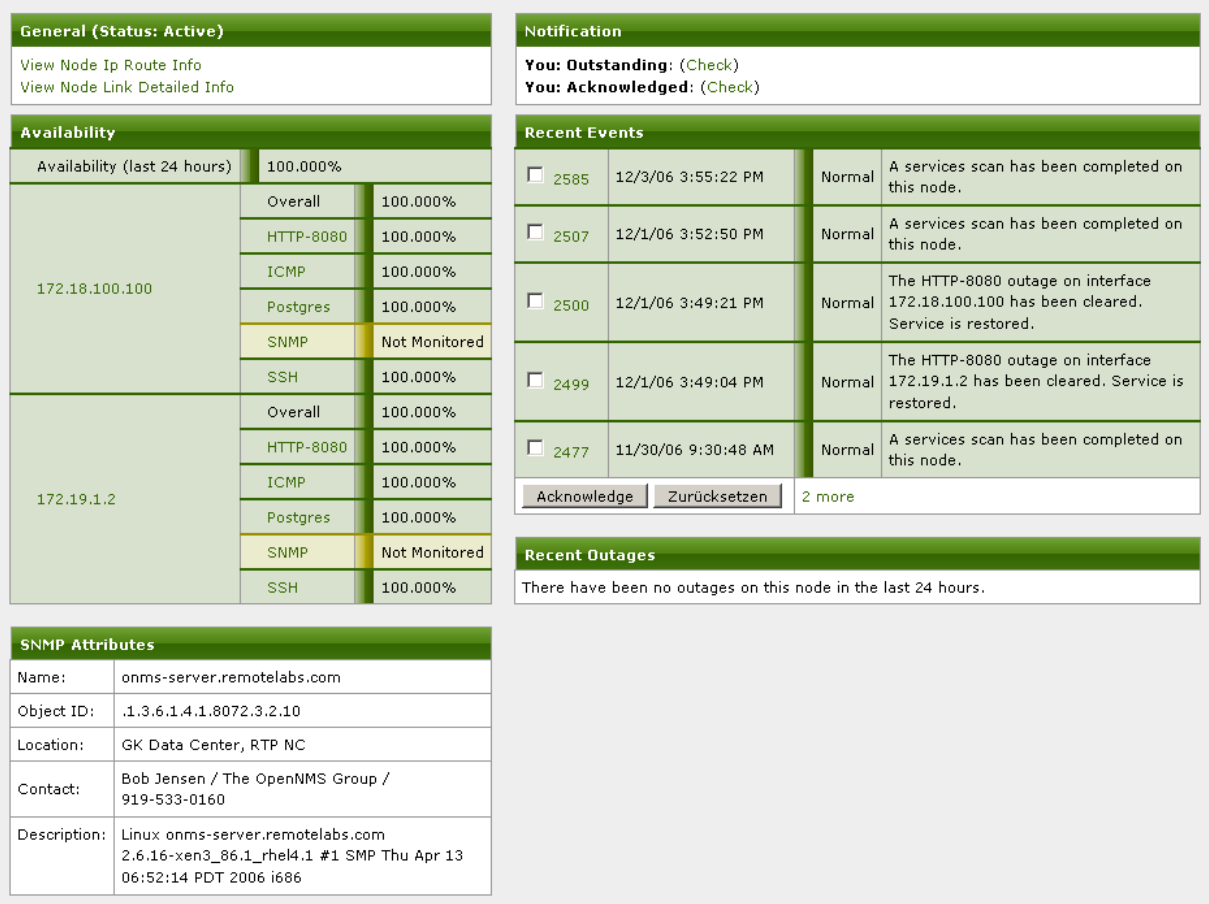

Abbildung 6.3: Ein Ausschnitt der openNMS-Oberfläche. Angezeigt werden Verfügbarkeitsdaten eines Servers.

6.5 Topologieerkennung

Zweifelsohne ist es in großen Netzwerken nicht mehr möglich, ohne Hilfsmittel die einzelnen Netzwerkkomponenten aufzulisten oder die Topologie des Netzwerkes zu kennen, weswegen einige Werkzeuge anbieten, die Topologie und die Komponenten automatisch zu erkennen. Dabei werden oftmals herstellerspezifische Verfahren genutzt, um an Informationen über das Netzwerk zu kommen. Unter Umständen kann es dabei zu den Problemen kommen, dass Netzwerkkomponenten nicht richtig erkannt werden, oder die Topologie fehlerhaft ist. Eine herstellerunabhängige Variante, die Topologie eines Netzwerkes zu erkennen, ist das Link Layer Discovery Protocol (LLDP), das im Standard IEEE 802.1AB beschrieben wird. Da der Standard erst 2005 verabschiedet wurde, wird er von vielen Netzwerkkomponenten leider noch nicht unterstützt.

6.6 Zusammenfassung

Bei der Betrachtung einzelner Softwareprodukte wird deutlich klarer, welche Daten über Hard- und Softwarebestände von diesen erwartet werden können. Es wird auch deutlich, dass sich die 3 Vertreter des *Managements verteilter Informationssysteme* hinsichtlich der Daten, die sie enthalten, nicht allzu sehr voneinander unterscheiden. Leider war für keines dieser Produkte eine Aussage darüber zu finden, wie Elemente eines Informationssystems in den Datenbanken repräsentiert werden. Später soll das hier nicht aufgeführte Softwareprodukt „Nagios“ vollständig betrachtet werden.

7 Hard- und Softwarebestände in 3LGM²-Modellen

Inhaltsangabe

7.1 Für die Abbildung von Hard- und Softwarebeständen relevante 3LGM²-Metamodellklassen 45

7.2 Verwendung der HSKs in 3LGM²-Modellen 47

7.3 Zusammenfassung 49

In den letzten Kapiteln wurde deutlich, welche Daten über Hard- und Softwarebestände durch eine Datenintegration verfügbar werden. Noch nicht betrachtet wurde, welche Daten überhaupt gebraucht und verwendet werden können. Dieses Kapitel hat zum Ziel, die Verwendbarkeit der Daten in 3LGM²-Modellen zu ergründen.

7.1 Für die Abbildung von Hard- und Softwarebeständen relevante 3LGM²-Metamodellklassen

Festlegung Die für die Abbildung von Hard- und Softwarebeständen relevanten 3LGM²-Metamodellklassen sollen im Weiteren abkürzend als HSK bezeichnet werden.

In der Einleitung wurde deutlich, dass insbesondere die Integration von Daten über Hard- und Softwarebestände eines Unternehmens angestrebt wird. Nicht alle Klassen des 3LGM²-Metamodells beschreiben solche Bestände, weswegen einige Klassen im Weiteren von untergeordnetem Interesse sind. Unter HSKs sollen hier diejenigen 3LGM²-Metamodellklassen verstanden werden, für die es vorstellbar ist, dass Instanzen dieser 3LGM²-Metamodellklassen aus den integrierten Daten erstellt werden. Ob und wie weit die folgenden Vorstellungen umsetzbar sind, wird sich immer erst in Verbindung mit einer ganz konkreten Datenquelle zeigen. Dies wird Inhalt der anschließenden Kapitel sein.

HSKs für die Fachliche Ebene

Auf der Fachlichen Ebene finden sich keine 3LGM²-Metamodellklassen, deren Instanzen sich sinnvoll aus Hard- und Softwaredaten erzeugen ließen. Zwar kann man, wenn man weiß, dass

es eine Software für die Patientenaufnahme gibt, vermuten, dass es ebenso einene *Aufgabe* „Patientenaufnahme“ geben muss, aber dieser Schluss ist zum einen nicht zwingend (es gibt die Software aufgrund der Unternehmensaufgabe und nicht die Aufgabe aufgrund der Software) und zum anderen ergibt sich dadurch kein Informationsmehrwert für das 3LGM²-Modell. Hinsichtlich der *Organisationseinheiten* lassen sich mehrere Szenarien vorstellen, bei denen Informationen für das 3LGM²-Modell gewonnen werden können. Der einfachste Fall wäre, wenn zusätzlich zu den speziellen Daten über Hard- und Softwarebestände noch administrative Daten wie Kostenstellen vorhanden wären. Die Kostenstellen könnten Hinweise darauf geben, welche *Organisationseinheiten* existieren. Auch eine Netzwerktopologie könnte Hinweise auf *Organisationseinheiten* geben, da sicherlich anzunehmen ist, dass Organisationseinheiten intern stärker vernetzt sind als untereinander und so Cluster bilden. Diese Option der Datengewinnung soll hier aber nur kurz erwähnt werden, da sie weit führende graphentheoretische Konzepte benötigt und ohnehin noch zu zeigen wäre, dass Cluster wirklich *Organisationseinheiten* wiederspiegeln. Für *Objekttypen* oder *Rollen* wird man in den Hard- und Softwaredaten kaum Anhaltspunkte finden können.

HSKs für die Logische Werkzeugebene

Auf der Logischen Werkzeugebene lassen sich direkt abbildbare HSKs finden. So lassen sich in Informationen über Softwarebestände *Anwendungsprogramme* und *Softwareprodukte* identifizieren. Ist zusätzlich zur Bezeichnung eines *Softwareprodukts* noch der genutzte Rechner (z. B. durch die IP Nummer) bekannt, so können ein *Anwendungsprogramm* und ein *Rechnerbasierter Anwendungsbaustein* mit einer *Datenverarbeitungsbaustein-Konfiguration* erstellt werden. Weiterhin könnten *Datenbankssysteme* und *Datenbankverwaltungssysteme* in den Beständen aufgeführt sein. Eventuell können noch Informationen über unterstützte *Kommunikationsstandards* und *Bausteinschnittstellen* von Anwendungsprogrammen gefunden werden.

HSKs für die Physische Werkzeugebene

Die Physische Werkzeugebene stellt, wie oben beschrieben, den physischen Teil des Informationssystems dar und das beinhaltet sowohl die datenverarbeitenden Geräte als auch die datenverarbeitenden Personen. Hard- und Softwarebestände beziehen sich auf den rechnerbasierten Teil des Informationssystems. Zentrale Klasse dieser Ebene ist der *Physische Datenverarbeitungsbaustein*, der in Form von Rechnern, Servern und Netzwerkkomponenten in Hardwarebestandsdaten auftauchen sollte. Sollten in diesen Hardwarebeständen auch Informationen über den jeweiligen Hardwaretyp enthalten sein, so kann dies genutzt werden, um *Bausteintypen* abzuleiten.

Subnetze sind recht einfach zu finden, wenn sie in den Daten als solche gekennzeichnet sind. Sind sie nicht gekennzeichnet, so ergeben sich Möglichkeiten, über die Topologie *Subnetze* zu identifizieren. Diese Variante wurde bereits für das Finden von *Organisationseinheiten* erwähnt. Für die Klassen *Netzprotokoll*, *Netztyp* und *Standort* lassen sich ohne nähere Betrachtung der Datenquellen noch keine Aussagen treffen. Instanzen dieser 3LGM²-Metamodellklassen können in den Datenbeständen zu finden sein, wenn das IS entsprechend detailliert beschrieben ist.

Das Verbindungsglied zwischen *Anwendungsbaustein* und ein oder mehreren *Physischen Datenverarbeitungsbausteinen* ist die *Datenverarbeitungsbaustein-Konfiguration*. Sie wird im 3LGM²-Baukasten automatisch erstellt und sollte auch erstellt werden, wenn Datenbestände diese Zuordnung zulassen. Allerdings muss bei dieser Zuordnung die *Bausteinrolle* festgelegt werden, die entweder einer *Serverrolle* oder einer *Clientrolle* entspricht. Die Zuordnung von Rollen ist nicht im 3LGM²-Baukasten implementiert.

Zusammenstellung der HSKs

Auf der Fachlichen Ebene

- *Organisationseinheit*

Auf der Logischen Werkzeugebene

- *Anwendungsprogramm*
- *Softwareprodukt*
- *Rechnerbasierter Anwendungsbaustein*
- *Datenbanksystem*
- *Datenbankverwaltungssystem*
- *Kommunikationsstandard*
- *Bausteinschnittstelle*

Auf der Physischen Werkzeugebene

- *Physischer Datenverarbeitungsbaustein*
- *Datenverarbeitungsbaustein-Konfiguration*
- *Bausteintyp*
- *Subnetz*
- *Netzprotokoll*
- *Netztyp*
- *Standort*

7.2 Verwendung der HSKs in 3LGM²-Modellen

Während im letzten Kapitel das theoretisch Mögliche des 3LGM²-Metamodells für die Integration von Daten umrissen wurde, soll es in diesem Kapitel nun um die praktische Verwendung der HSKs in 3LGM²-Modellen gehen. Dazu soll ein vorhandenes 3LGM²-Modell hinsichtlich der Verwendung dieser Klassen untersucht werden. Betrachtet wurde ein 3LGM²-Modell des Universitätsklinikums Leipzig. Ziel dieses Abschnitts ist es, festzustellen, wie Nutzer das 3LGM²-Metamodell beim Modellieren interpretieren, um anschließend die Integration von Daten ähnlich zu gestalten. Dies ist wichtig, um die Nutzerakzeptanz einer Kopplung zu gewährleisten.

Verwendung auf der Fachlichen Ebene

Im UKL-Modell werden als *Organisationseinheiten* die einzelnen Kliniken gewählt, die meist als untergeordnete *Organisationseinheiten* die einzelnen Stationen enthalten. Es werden fast alle *Organisationseinheiten* zusätzlich dem „Universitätsklinikum Leipzig AöR“ untergeordnet. Im Teilmodell MCC (Medical Control Center, ein Produkt der Firma Meierhofer AG) werden einige Stationen nicht dem „Universitätsklinikum Leipzig AöR“ untergeordnet.Das UKL-Modell weist mit dem „Arztzimmer“ nur eine räumlich definierte *Organisationseinheit* auf.

Verwendung auf der Logischen Werkzeugebene

Softwareprodukte werden im 3LGM²-Modell des UKL mit ihrem Namen ohne Versionsnummer oder Versionsbesonderheiten aufgeführt (z. B. MS Office anstelle von Microsoft Office 2000 Professional). Diese Vereinfachung der Bezeichnung ermöglicht das Vernachlässigen von Versionswechseln, bedeutet aber auch einen Informationsverlust.

Die *Anwendungsprogramme* sind das Bindeglied zwischen den *Softwareprodukten* und den *rechnerbasierten Anwendungsbausteinen*, wobei sie sich namentlich eher an den *rechnerbasierten Anwendungsbausteinen* orientieren. Das Erstellen von *Anwendungsprogrammen* aus zu integrierenden Daten sollte sich daher, wie auch im 3LGM²-Baukasten realisiert, an die Erstellung eines *rechnerbasierten Anwendungsbausteins* binden und nicht an die eines *Softwareproduktes*.

Im UKL-Modell wurden die *Datenbanksysteme* entsprechend ihrem Verwendungszweck bezeichnet und die *Datenbankverwaltungssysteme* entsprechend den verschiedenen Datenbankherstellern.

Bei den *Kommunikationsstandards* im UKL-Modell verhält es sich wie bei den *Softwareprodukten*, sie werden ohne Versionsnummer beschrieben. Bezeichnungen von *Bausteinschnittstellen* im UKL-Modell sind so gewählt, dass daraus die Funktion der Schnittstelle ersichtlich wird (z. B. KomServ-Red-Empfang-SS).

Verwendung auf der Physischen Werkzeugebene

Der *Physische Datenverarbeitungsbaustein* ist eine sehr allgemein gehaltene Klasse des 3LGM²-Metamodells, weswegen er im 3LGM²-Baukasten zusätzlich in Cluster, RAID, Spiegelsatz, Standby und *Physischen DV-Baustein* unterteilt wird. Diese Unterteilung hat aber keinen Einfluss auf die Eigenschaften des *Physischen DV-Bausteins*, sondern ist nur eine funktionelle Einordnung. Im UKL-Modell werden diese Untergruppen auch nicht genutzt. Die aufgeführten *Physischen DV-Bausteine* tragen als Bezeichnung ihre Verwendung (z.B. SAP QTS, SAP DB) und teilweise, wenn es sich um Rechner handelt, ihren Rechnertyp als Beschreibung (z.B. SunFire V480).

Bausteintypen sind erwartungsgemäß die Oberbegriffe für Gruppen von *DV-Bausteinen*, sowohl für menschliche als auch rechnerbasierte (z.B. Person, Server, PC, Fax). Als *Subnetz* sind im UKL-Modell nur „MRZ“ und „Zentrales Server-Subnetz“ angelegt, woraus sich keine Verwendungsrückschlüsse ziehen lassen.

Für *Netzprotokoll* und *Netztyp* gibt es ebenfalls keine Verwendung im UKL-Modell. *Standorte*

sind im 3LGM²-Modell exakte Ortsbeschreibungen, die durch Straße, Nummer, Raum beschrieben werden. Sie grenzen sich damit deutlich von den Organisationseinheiten ab.

7.3 Zusammenfassung

In Kapitel 7 wurde deutlich, an welchen Stellen Daten über Hard- und Softwarebestände vom 3LGM²-Metamodell beschrieben werden und wie Informationssysteme von Modellierern genutzt werden. Darauf aufbauend können in den nächsten Kapiteln die Betrachtungen für das 3LGM²-Metamodell auf diese HSKs beschränkt werden.

8 Anforderungen an eine Datenintegration

Inhaltsangabe

8.1 Funktionelle Anforderungen 51

8.2 Nicht funktionelle Anforderungen 52

8.3 Zusammenfassung 54

In der Softwaretechnik ist es üblich, Anforderungen zu spezifizieren. Da eine Datenintegration ohne die Implementierung von Software praktisch nicht möglich ist, soll an dieser Stelle kurz auf Anforderungen eingegangen werden, die bereits auf der Ebene der Rahmenbedingungen für eine Datenintegration erkennbar sind. Dazu gehören sowohl funktionelle als auch nicht funktionelle Anforderungen.

8.1 Funktionelle Anforderungen

Konsistenzbedingungen für HSKs Das 3LGM²-Metamodell ist als UML-Diagramm beschrieben und hat dadurch die Möglichkeit, Kardinalitäten anzugeben und bei mehreren möglichen Kardinalitäten, Multiplizitäten. (vgl. [OESTEREICH 05] S.273) Diese werden unter anderem dazu genutzt, bei der Erstellung eines 3LGM²-Modellelements die Erstellung oder Benennung eines anderen unbedingt benötigten Elements zu erzwingen. Solche Anforderungen müssen natürlich auch bei der Integration aus Datenquellen beachtet werden. Einige der im letzten Kapitel identifizierten 3LGM²-Metamodellklassen enthalten solche Kardinalitätsregeln.

- Jedes *Anwendungsprogramm* braucht ein ihm zugeordnetes *Softwareprodukt*. Diese Metamodellbedingung ist allerdings nicht im 3LGM²-Baukasten als solche implementiert. Um aber möglichst vollständige 3LGM²-Modelle zu erhalten, sollte sie erfüllt werden.
- Jedes *Anwendungsprogramm* braucht einen *rechnerbasierten Anwendungsbaustein*. Diese Forderung wird im 3LGM²-Baukasten umgesetzt und muss daher auch bei der Datenintegration berücksichtigt werden.
- Jedes *Datenbanksystem* braucht ein zugehöriges *Datenbankverwaltungssystem*. Diese Metamodellbedingung ist nicht im 3LGM²-Baukasten umgesetzt. Um aber möglichst vollständige 3LGM²-Modelle zu erhalten sollte, auch sie erfüllt werden.
- Ebenso braucht jedes *Datenbanksystem* einen *rechnerbasierten Anwendungsbaustein*, was wiederum durch den 3LGM²-Baukasten erzwungen wird.
- Auch jede *Bausteinschnittstelle* benötigt einen *rechnerbasierten Anwendungsbaustein*. Dies ist im 3LGM²-Baukasten umgesetzt.

- Eine Forderung auf der Physischen Ebene ist, dass jedes *Subnetz* zu mindestens einem *Physischen Datenverarbeitungbaustein* gehört. Diese Forderung wird nicht im 3LGM²-Baukasten umgesetzt.

Einbindung vorhandener 3LGM²-Modellelemente Da in 3LGM²-Modellen bereits 3LGM²-Modellelemente erstellt sein können, ist es wichtig, diese so weit möglich auch für die integrierten Daten zu verwenden.

Transportierbarkeit 3LGM²-Modelle müssen zwischen verschiedenen 3LGM²-Baukasteninstallationen austauschbar sein. Dabei sollte es keine Teile eines 3LGM²-Modells geben, die nicht „transportierbar" sind. Diese Anforderung ermöglicht beispielsweise das Versenden eines Modells per E-Mail.

Unabhängigkeit des Datenintegrationsprozesses vom 3LGM²-Baukasten Aufgrund der eben beschriebenen Transportierbarkeit muss es auch möglich sein, 3LGM²-Modelle in jeder 3LGM²-Baukasteninstallation synchronisieren zu können. (siehe Kapitel 4.6.2.3) Das bedeutet, dass keine Abhängigkeit zwischen 3LGM²-Baukasteninstallation und 3LGM²-Modellen mit integrierten Daten bestehen darf.

8.2 Nicht funktionelle Anforderungen

Aufwand Ist der *Aufwand*, der zur Erreichung einer Datenintegration benötigt wird, zu groß, so wird darunter die Nutzerakzeptanz leiden und die Datenintegration nicht durchgeführt.

Modellierstile Aus den Beobachtungen in Kapitel 7.2 folgt, dass es nicht üblich ist, aus zu integrierenden Daten *rechnerbasierte Anwendungsbausteine* und *Datenverarbeitungsbaustein-Konfigurationen* zu erstellen, die jeweils auf den Informationen zu nur einem PC gründen. Die *rechnerbasierten Anwendungsbausteine* sind im Wesentlichen Einheiten, die umfangreicher sind. Sie stellen ganze Subsysteme des Informationssystems dar (z. B. Bildbetrachtungssystem).

Datenqualität Um Datenqualität zu garantieren, muss die Qualität von Daten messbar sein. Dafür können verschiedene Aspekte herangezogen werden, die in den Hard- und Softwaredaten vorhanden und für den Modellierer relevant sein müssen. Ein Beispiel für einen Qualitätsaspekt wäre die Aktualität von Daten.

Ist es in der Integrationskomponente möglich, qualitativ ungenügende Daten zu entdecken, so sollten diese dem Modellierer zum „Aussortieren" vorgeschlagen werden. Darüber hinaus muss es für den Benutzer immer möglich sein, Daten, die er für qualitativ ungenügend befindet, auszusortieren.

Datenmenge Die Beherrschung der Datenmenge ist eine der zentralen Aufgaben, um die Benutzerakzeptanz zu begünstigen. Die Anzahl der hinzukommenden 3LGM²-Modellelemente

kann 3LGM²-Modelle unübersichtlich werden lassen. Eine Möglichkeit, die in das 3LGM²-Modell eingehende Menge von neu integrierten 3LGM²-Modellelementen zu beschränken, soll dadurch gegeben werden, dass die Datenintegration zunächst in der Integrationskomponente in einem temporären globalen Datenbestand stattfindet. Der Nutzer soll in der Integrationskomponente die Möglichkeit haben, die Menge der entstehenden 3LGM²-Modellelemente zu kontrollieren. Dabei muss beachtet werden, dass sich im 3LGM²-Modell bereits 3LGM²-Modellelemente befinden können, die mit zu integrierenden 3LGM²-Modellelemente identisch sind.

Visualisierung Bei der Visualisierung ist vieles möglich, wenn man den 3LGM²-Baukasten in seiner Funktionalität erweitern. Für große Hierarchien lassen sich beispielsweise hyperbolische Ebenen nutzen, wie in [LAMPING et al. 94] beschrieben. Aber auch im alten 3LGM²-Baukasten gibt es Funktionen, die die Übersichtlichkeit bei großer Elementanzahl bewahren sollen. So ist es möglich, Elemente, die sich in einer hierarchischen Beziehung befinden, grafisch auf das übergeordnete Element zu reduzieren. Diese Funktion wird als Auf- bzw. Zuklappen bezeichnet und ist in Abbildung 8.1 dargestellt. Das Auf- und Zusammenklappen funktioniert auch über mehrere Ebenen hinweg.

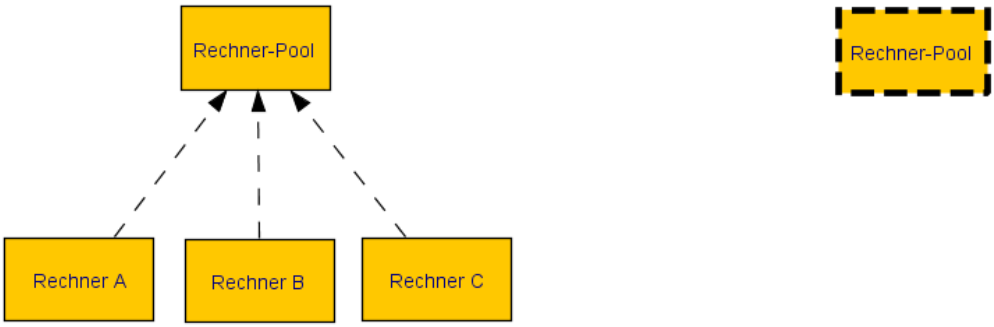

Abbildung 8.1: Links sind physische DV-Bausteine in einer Teil-von-Beziehung aufgeklappt dargestellt. Rechts sind die selben Bausteine zugeklappt dargestellt.

Automatisierung Ideal wäre eine vollständige Automatisierung der Integration der Datenbestände. Diese steht aber der Forderung entgegen, die integrierte Datenmenge nicht all zu groß werden zu lassen, da ein automatischer Prozess in der Regel nicht entscheiden kann, welche Elemente unwichtig sind und nicht in das 3LGM²-Modell aufgenommen werden müssen. Weiterhin kann vermutlich nicht festgestellt werden, ob Elemente, die vom Benutzer manuell erstellt wurden, mit Elementen die integriert werden sollen, identisch sind. Eine gewisse Interaktion mit dem Benutzer wird daher oft erfolgen müssen. Sie sollte aber so stark wie möglich reduziert werden.

Im Zusammenhang mit dem 3LGM²-Baukasten ist es wahrscheinlich, dass Interaktion bei folgenden Punkten nötig wird.

- Auswahl der Daten, die in ein 3LGM²-Modell integriert werden
- Vereinigung eines manuell erstellten und eines integrierten 3LGM²-Modellelements

8.3 Zusammenfassung

Die hier formulierten Anforderungen sind nicht abschließend und können durchaus um weitere Punkte ergänzt werden. Allerdings stellen sie die Eckpunkte dafür dar, dass eine Datenintegration von Hard- und Softwaredaten mit dem 3LGM²-Baukasten konsistent und für einen Modellierer anwendbar ist. Dabei leiten sich einige Anforderungen von Beobachtungen in früheren Kapiteln dieser Arbeit ab.

9 Referenzarchitektur für die Datenintegration mit dem 3LGM²-Baukasten

Inhaltsangabe

9.1 Lokale und globale Datenstrukturen 55
9.2 Lokale und globale Datenbestände 56
9.3 Bindung der Integrationskomponente 57
9.4 Alternativen zur Referenzarchitektur 57
9.5 Zusammenfassung 58

Im Folgenden soll eine Referenzarchitektur für die Datenintegration des 3LGM²-Baukastens mit einem Hard- und/oder Softwarebestände enthaltenden Softwareprodukt vorgeschlagen werden. Das Wesen einer Referenzarchitektur orientiert sich dabei an dem Wesen von Referenzmodellen. (vgl. [WINTER et al. 99] S.176) Somit sollen von der Referenzarchitektur durch Modifikationen, Einschränkungen oder Ergänzungen konkrete Architekturen abgeleitet werden. Die Anwendung, die den 3LGM²-Modellen als Datenquelle dient, soll im Folgenden als *Fremdanwendung* bezeichnet werden.

9.1 Lokale und globale Datenstrukturen

Die lokalen Datenstrukturen sind zum einen die Datenstruktur des 3LGM²-Baukastens, die durch das 3LGM²-Metamodell beschrieben wird, und zum anderen die Datenstruktur des Hard- und Softwarebestandes, die für die Referenzarchitektur unbekannt ist. Ohne Kenntnis über die Fremdanwendung der Hard- und Softwarebestände ist auch unbekannt, welchen Informationsbedarf diese Fremdanwendung gegenüber 3LGM²-Modellen hat. Daher ist die Datenintegration in der Referenzarchitektur eher einseitig motiviert. Die neue Qualität, von der in den Definitionen der Datenintegration die Rede ist, soll sich durch den in der Einleitung motivierten Informationsgewinn zunächst nur auf Seiten des 3LGM²-Baukastens äußern.

Aufgrund der in Kapitel 4.6.1 beschriebenen Eigenschaften ist für die Integration der Datenstrukturen ein *Top-Down-Ansatz* zu empfehlen, da dabei die globale Datenstruktur überschaubar bleibt. Allein der Informationsbedarf bestimmt bei diesem Ansatz die globale Struktur. (vgl. [JUNG 06] S.157) Da der Informationsbedarf, wie in Abbildung 9.1 dargestellt, stark vom 3LGM²-Baukasten beeinflusst wird, sind es insbesondere die HSKs aus Kapitel 7, die die globale Datenstruktur prägen sollten.

Ein sinnvolles Datenmodell für die globale Struktur ist daher das Datenmodell des 3LGM²-Metamodells, die Unified Modeling Language (UML).

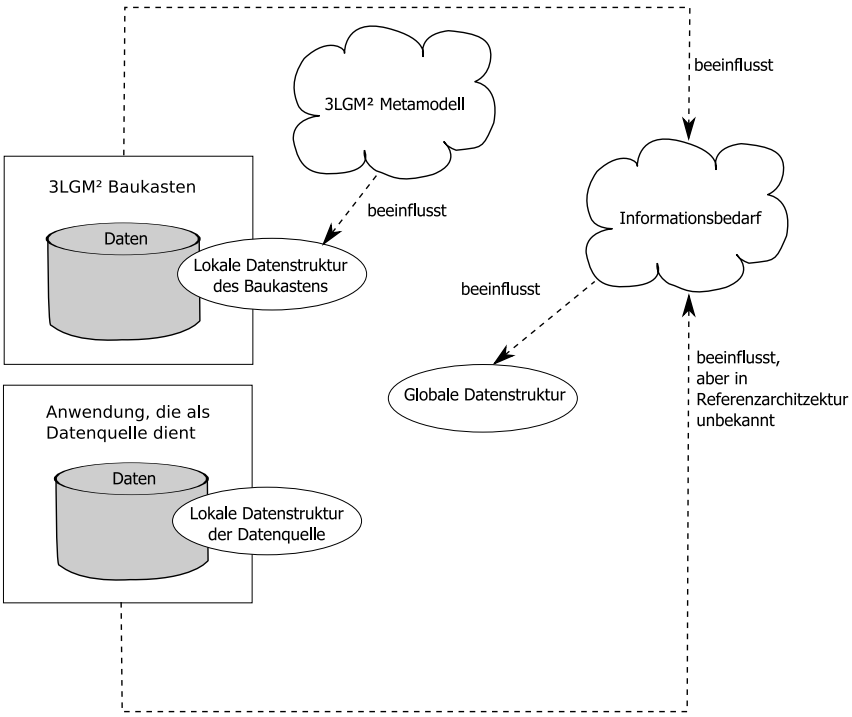

Abbildung 9.1: Die Abhängigkeiten der globalen Datenstruktur bei der Datenintegration des 3LGM²-Baukastens

9.2 Lokale und globale Datenbestände

Nach der globalen Datenstruktur, die im letzten Abschnitt beschrieben wurde, stellt sich die Frage, ob eine materielle Datenintegration erfolgen soll, bei der ein vollständiger integrierter Datenbestand aufgebaut wird oder ob nur eine virtuelle Datenintegration erfolgt, bei der „Anfragen“ aus den lokalen Datenbeständen heraus beantwortet werden. Wird ein Datenbestand aufgebaut, so kann dieser persistent oder transient sein. (siehe Kapitel 4.6.2.2)

Dabei müssen zwei Anforderungen beachtet werden: die Transportierbarkeit von 3LGM²-Modellen und die Unabhängigkeit des Datenintegrationsprozesses von einer 3LGM²-Baukasteninstallation. Beide Anforderungen sind eher unüblich für klassische Datenintegration. Sie führen dazu, dass die beteiligten Anwendungen jederzeit Zugriff auf den globalen Datenbestand haben müssen, auch wenn nicht alle lokalen Datenquellen verfügbar sind, was eine virtuelle Datenintegration ausschließt. Auch eine transiente materielle Datenintegration wird diesen Anforderungen nicht gerecht, da sie ebenfalls Zugriff auf alle beteiligten lokalen Datenquellen benötigt, um sich aufzubauen. Damit bleibt als einzige Realisierung ein materieller, persistenter globaler Datenbestand. (vgl. Abbildung 4.4) Um die ständige Verfügbarkeit des globalen Datenbestandes für ein 3LGM²-Modell zu garantieren, muss der globale Datenbestand mittransportiert werden. Dies gilt auch für die Fremdanwendung, wenn sie denn den globalen Datenbestand nutzen will.

Ob eine Anwendung zukünftig mit ihrem lokalen Datenbestand arbeitet oder nur noch mit dem globalen Datenbestand, kann ohne konkreten Anwendungsfall nicht entschieden werden. Jedoch muss die Integrationskomponente in der Lage sein, die verschiedenen Datenbestände

wieder zu synchronisieren.

Die zweite Bedingung, die Unabhängigkeit von der 3LGM²-Baukasteninstallation, wird im Abschnitt 9.3 erläutert.

Absorption des globalen Datenbestandes Der globale Datenbestand enthält einen Informationsmehrwert, der in den lokalen Datenbeständen nicht vorhanden ist und normalerweise auch nicht dargestellt werden kann. Dieser Mehrwert entsteht beispielsweise durch das Verbinden von Objekten, die vorher unverbunden waren. Es kann aber auch möglich sein, dass dieser Informationsmehrwert doch in lokalen Datenbeständen darstellbar ist. Dann ist es auch möglich, auf den globalen Datenbestand zu verzichten und statt dessen den Informationsmehrwert in den lokalen Datenbestand aufzunehmen. Aus dem lokalen Datenbestand und dem Informationsmehrwert muss es aber möglich sein, den globalen Datenbestand zu rekonstruieren. Gilt dies für alle lokalen Datenbestände, dann ist der globale Datenbestand nicht mehr persistent sondern transient.

Dieser Sonderfall wird im Teil 3 dieser Arbeit auftreten. Dort wird der Informationsmehrwert durch einen Nagiosschlüssel im lokalen Datenbestand des 3LGM²-Baukastens realisiert.

9.3 Bindung der Integrationskomponente

Mit Bindung ist hier die programmtechnische Zugehörigkeit gemeint, die Einflüsse auf die Aktivierung der Integrationskomponente hat. Die Datenintegration ist zunächst sehr stark 3LGM²-baukastenorientiert und sollte daher durch eine im 3LGM²-Baukasten integrierte Komponente durchgeführt werden. Eine schematische Darstellung der Bindung in der Referenzarchitektur ist in Abbildung 9.2 zu finden. Da der 3LGM²-Baukasten kein Werkzeug ist, welches in ständiger Benutzung ist, sondern nur bei Bedarf gestartet wird, kann die Datenintegration nicht sofort bei Bedarf erfolgen, sondern erfolgt in unbestimmten Zeitintervallen, was einer *asynchronen* und *benutzerinitiierten* Datenintegration entspricht. (nach [JUNG 06] S.199) (siehe Kapitel 4.6.2.3)

Da die Unabhängigkeit der Datenintegration von der 3LGM²-Baukasteninstallation gefordert wird und die Integrationskomponente an den 3LGM²-Baukasten gebunden sein soll, dürfen keine zur Datenintegration notwendigen Daten in der Integrationskomponente verbleiben. Daher kann der in der Integrationskomponente aufgebaute globale Datenbestand nach beendeter Datenintegration gelöscht werden und ist damit transient. Alle wichtigen Informationen müssen in den verteilten globalen Datenbeständen vorhanden sein.

9.4 Alternativen zur Referenzarchitektur

Manche Daten lassen sich dem Modellierer schlecht durch die hier vorgeschlagene Architektur einer Datenintegration zur Verfügung stellen. Beispielsweise veralten Verfügbarkeitsdaten von Netzwerkkomponenten sehr schnell, weshalb sie durch eine benutzerinitiierte Datenintegration nicht sinnvoll integriert werden können. Eine Funktion des 3LGM²-Baukastens, die es

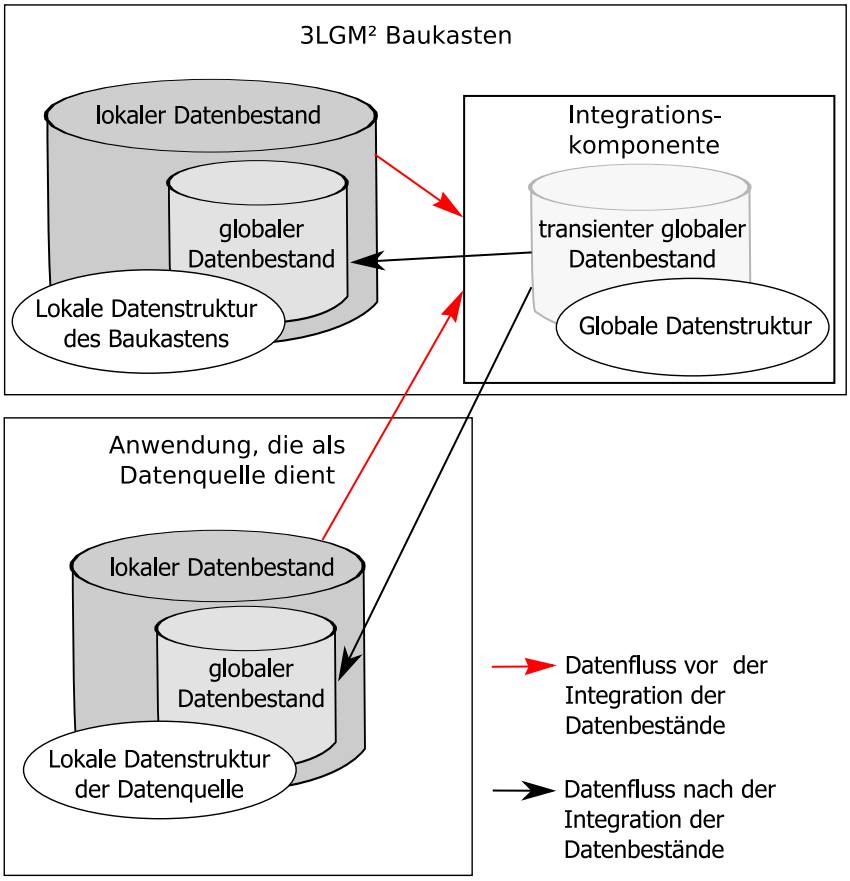

Abbildung 9.2: Referenzarchitekturvorschlag für die Datenintegration

ermöglicht, auf die Verfügbarkeitsdaten zuzugreifen, wäre aber denkbar.

In solchen Fällen stellt die *funktionelle Integration* teilweise eine Alternative dar und sollte berücksichtigt werden. Diese Dualität wird wie folgt beschrieben:

> Wenn zwischen verschiedenen Anwendungssystemen funktionelle Integration hinsichtlich einer bestimmten Funktionalität gegeben ist, dann müssen die mit dieser Funktionalität verarbeiteten Informationen, genauer: die die Informationen repräsentierenden Daten, nicht zwischen den betreffenden Anwendungssystemen abgeglichen werden. Der Zugriff erfolgt nur über die gemeinsam genutzten Funktionen, wodurch eine einmalige Speicherung genügt. ([WENDT 06] S.16)

9.5 Zusammenfassung

Für die Referenzarchitektur konnten einige Varianten der Datenintegration ausgeschlossen und die Bindung der Integrationskomponente sowie alternative Integrationsmöglichkeiten vorgeschlagen werden. Dabei handelt es sich um die Lösung, die unter Berücksichtigung der Erkenntnisse der letzten Kapitel immer möglich ist. Das Abweichen von dieser Referenzarchitektur zugunsten einer besseren Lösung im Einzelfall bleibt davon unberührt.

Teil III

Datenintegration mit Nagios

10 Nagios Grundlagen

Inhaltsangabe

10.1 Motivation 61
10.2 Funktionsumfang 62
10.3 Funktionsweise 62
10.4 Konfiguration 62
10.5 Inhalt der Konfiguration 63
10.5.1 Terminologie für Nagios 63
10.5.2 Nagiosobjekte und deren Informationsgehalt 65
10.5.3 Vergleich mit den HSKs 70
10.5.4 Vererbungsbeziehungen durch Nagiostemplates 71
10.5.5 Reguläre Ausdrücke in Nagios 71
10.5.6 Interne und externe Relationen 73
10.5.7 Interpretation interner und externer Relationen 74
10.6 Zusammenfassung 78

Nach der Betrachtung der Datenintegration im Allgemeinen soll in den nächsten Kapiteln die Datenintegration des Softwareprodukts Nagios mit dem Baukasten beschrieben werden. Daher wird in diesem Kapitel zunächst die Funktionsweise und Konfiguration von Nagios vorgestellt, um darauf aufbauend die Referenzarchitektur umzusetzten.

Nagios ist ein Werkzeug für das System- und Netzwerkmonitoring und bedient damit eine Teildisziplin des Managements verteilter Systeme. Nagios steht unter der GNU General Public License (GPL) Version 2 und ist daher vielfältig nutzbar. Die Entwicklung von Nagios beginnt 1999 unter dem Namen NetSaint und geht auf Ethan Galstad zurück, der heute mit seiner Firma Open-Source Dienstleistungen anbietet. Damit ist die Entwicklung von Nagios deutlich jünger als beispielsweise die von IBM *NetView*. Die aktuelle Version von Nagios ist zum Zeitpunkt dieser Arbeit Version 2.5. Nagios ist für Linux Betriebssysteme entwickelt.

10.1 Motivation

Die Vergleichbarkeit von Nagios mit kommerziellen Anwendungen bezüglich der Anzahl an Installationen ist nicht gegeben, da Nagios keine nachvollziehbaren Kundenbeziehungen hat. Dennoch soll kurz das Nutzerprofil skizziert werden, um zu verdeutlichen, dass es sinnvoll ist, Nagios mit dem 3LGM²-Baukasten zu koppeln.

Von über 1000 bekannten Nutzern von Nagios stammen 18% aus dem deutschen Sprachraum und von allen Installationen entfallen 4,4% auf im Gesundheitswesen tätige Einrichtungen. (vgl. [GALSTAD 06]) Beispielsweise wird in der Tiroler Landeskrankenanstalten GmbH (TILAK) sowohl der 3LGM²-Baukasten als auch Nagios für das Management des Informationssystems eingesetzt.

Zudem beschränkt sich Nagios nicht nur auf das Überwachen reiner Netzwerkdienste, sondern kann auch spezielle Sachverhalte (z. B. besondere Dienste eines selbstentwickelten Kommunikationsservers) testen. Auch die Ausrichtung auf Tests von Datenbanken kommt dem Informationsbedarf des 3LGM²-Baukastens entgegen.

Nicht zuletzt hat Nagios den Vorteil, dass die Informationen über Netzwerke gut strukturiert und als Konfigurationsdateien auch gut erreichbar sind.

10.2 Funktionsumfang

Nagios überwacht Netzwerkdienste wie SMTP, POP, HTTP und Ressourcen von Rechnern wie Festplattenplatz, Prozessorbelastung usw. Grundsätzlich kann aber alles überwacht werden, was durch Rechner messbar ist, wie beispielsweise die Raumtemperatur oder die Luftfeuchtigkeit. Es wird bei Problemen im Netzwerk durch Kenntnis der Topologie zwischen den Fehlerklassen „DOWN“ und „UNREACHABLE“ unterschieden. In Abhängigkeit vom Fehler werden auch unterschiedliche Kontaktgruppen über unterschiedliche Kommunikationswege (SMS, E-Mail,...) informiert. Unbearbeitete Fehler können Eskalationsstufen auslösen, wodurch weitere Fehlermeldungen verschickt werden. Nagios kann sowohl redundant als auch verteilt aufgebaut werden, um beispielsweise die Ausfallsicherheit zu erhöhen. Die Benutzerschnittstelle von Nagios ist eine über Webbrowser bedienbare HTML-Schnittstelle.

10.3 Funktionsweise

Den Kern der Anwendung bildet der Nagios-Dämon, der als ständiger Dienst im Hintergrund agiert. Von diesem Dämon aus werden die verschiedenen Tests für das Netzwerk angestoßen. Ausgeführt werden aber alle Tests von Plugins, die als Ergebnis eines Tests OK, WARNING, CRITICAL oder UNKNOWN zurückgeben. Eine schematische Darstellung des Aufbaus von Nagios (nicht redundant und nicht verteilt) findet sich in Abbildung 10.1.

10.4 Konfiguration

Nagios wird durch Konfigurationsdateien beeinflusst. In diesen Dateien werden zum einen Parameter gesetzt, die den Nagios-Dämon und seine Plugins beeinflussen und zum anderen wird in ihnen beschrieben, was Nagios überwachen soll. Die Aufteilung der Konfiguration in einzelne Dateien ist nicht zwingend, fördert aber die Übersichtlichkeit. Für das weitere Vorgehen wird angenommen, dass die Parameter für den Nagios-Dämon und seine Plugins in den Dateien `nagios.cfg` und `resource.cfg` beinhaltet sind.

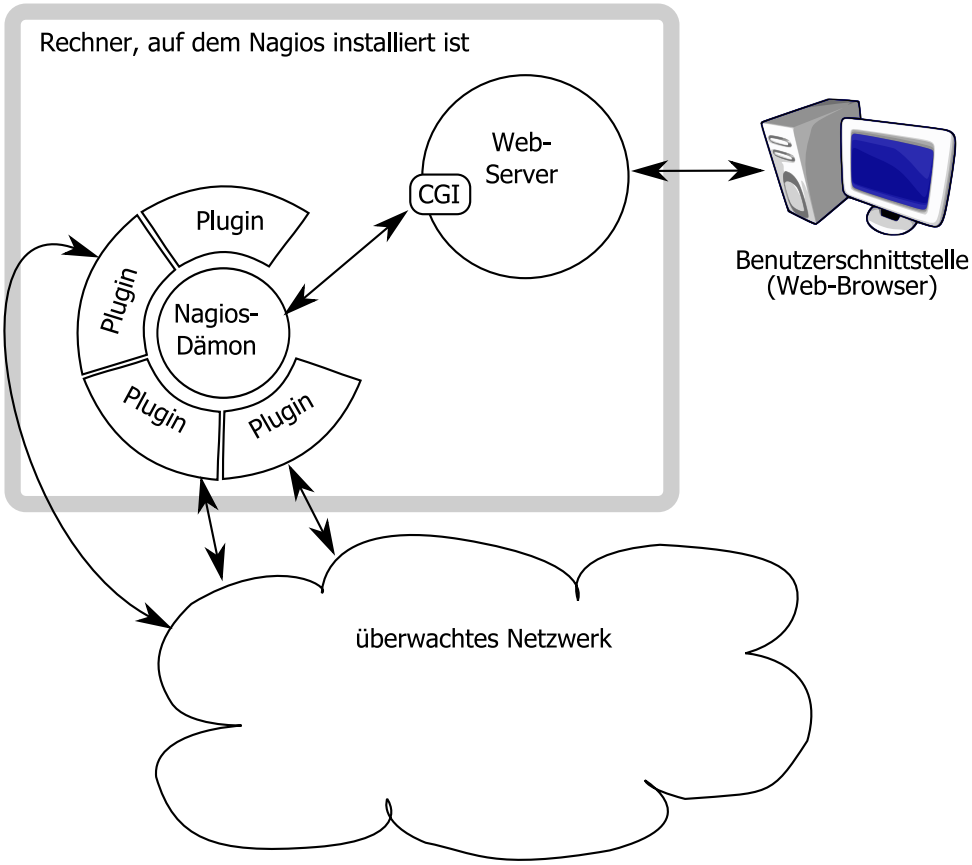

Abbildung 10.1: Aufbau einer Nagios Installation; der Dämon führt Tests mit Hilfe der Plugins durch und die Benutzerschnittstelle ist über einen Webserver zugänglich. Common Gateway Interface (CGI) ist die Schnittstelle, mit der die HTML-Seiten erzeugt werden. Der Webserver und die CGI-Benutzerschnittstelle sind optional.

Im Weiteren sollen mit Konfigurationsdateien nur jene Konfigurationsdateien gemeint sein, deren Inhalt das zu überwachende Netzwerk beschreibt.

10.5 Inhalt der Konfiguration

10.5.1 Terminologie für Nagios

Nagios arbeitet in seinen Konfigurationsdateien mit Nagiosdefinitionen, deren Realisierungen Nagiosobjekte sein können. Um im Folgenden Klarheit zu schaffen, werden nun die Begriffe *Nagiosobjekttyp*, *Nagiosdefinition*, *Nagiosobjekt* und *Nagiostemplate* definiert.

Festlegung *Nagiosobjekttypen* sind:

- Service
- Service Group
- Host
- Host Group
- Contact
- Contact Group
- Command
- Time Period
- Service Escalation
- Service Dependency
- Host Escalation
- Host Dependency
- Extended Host Information
- Extended Service Information

Die nagiosobjekttypen sind Klassen, zu denen Nagiosobjekte zugeordnet werden und bestimmen, welche Attribute in einer Nagiosdefinition zwingend und optional vorkommen.

Festlegung Eine *Nagiosdefinition* beschreibt für einen Nagiosobjekttyp ein konkretes Nagiosobjekt und/oder ein Nagiostemplate.

Syntax einer Nagiosdefinition:

```
define OBJEKTTYP
               {
               ATTRIBUT WERT
               ATTRIBUT WERT
               ...
               }
```

Festlegung Ein *Nagiosobjekt* ist eine Realisierung einer Nagiosdefinition, die nicht mit dem Attribut und Attributwert `register 0` gekennzeichnet ist.

Das `register`-Attribut wird später noch erläutert.

Wird im Folgenden eine Nagiosobjekttypbezeichnung alleinstehend verwendet, so ist ein Nagiosobjekt selbigen Typs gemeint. z. B. Host (=Nagiosobjekt vom Nagiosobjekttyp Host)

Festlegung Ein *Nagiostemplate* ist eine Nagiosdefinition, die von einer anderen Nagiosdefinition über das `use`-Attribut genutzt wird.

Das `use`-Attribut wird später noch erläutert.

Die Definition von Nagiostemplate schließt damit alle Nagiosdefinitionen aus, die zwar als Nagiostemplate gekennzeichnet sind, aber nicht als solches verwendet werden.

Mit Nagiosdefinitionen wird das von Nagios zu überwachende Netzwerk beschrieben. In dieser Beschreibung enthalten sind unter anderem die Topologie, die überwachten Geräte und die überwachten Dienste.

10.5.2 Nagiosobjekte und deren Informationsgehalt

Im Folgenden soll beschrieben werden, welche Informationen Nagiosobjekte enthalten. Dabei ist schon im Voraus zu erkennen, dass nicht alle Nagiosobjekte wertvolle Informationen für ein 3LGM²-Modell darstellen. Daher wird auf die nähere Betrachtung folgender Objekte im Weiteren verzichtet:

Nagiosobjekte des Nagiosobjekttyps Contact Contacts-Nagiosobjekte sind Kontaktinformationen der Administratoren, die im Fehlerfall informiert werden und für ein 3LGM²-Modell nicht von besonderem Interesse.

Nagiosobjekte des Nagiosobjekttyps Host/Service Escalation Nagiosobjekte dieses Nagiosobjekttyps bestimmen das Benachrichtigungsverhalten im Fehlerfall.

Nagiosobjekte des Nagiosobjekttyps Host/Service Dependency Nagiosobjekte dieser Nagiosobjekttypen beschreiben Regeln, um Benachrichtigungen über Fehler zu unterdrücken in Abhängigkeit von Zuständen anderer Hosts bzw. Services.

Nagiosobjekte des Nagiosobjekttyps Command Nagiosobjekte dieses Nagiosobjekttyps beschreiben auszuführende Überwachungsbefehle.

Nagiosobjekte des Nagiosobjekttyps Extended Host/Service Information Nagiosobjekte dieser Nagiosobjekttypen enthalten Informationen, die ihre Darstellung in der Benutzerschnittstelle beeinflussen (Grafiken).

10.5.2.1 Nagiosobjekttyp Host

In Nagiosdefinitionen von Hosts finden sich Informationen zu den überwachten Netzwerkkomponenten. Das können Rechner, Hubs, Printserver und andere Komponenten eines Netzwerkes sein. Entscheidend ist die Adressierbarkeit, die durch das IP-Protokoll mit einer IP-Adresse vorgegeben ist. Damit sind auch virtuelle Rechner, die eine eigene IP-Adresse tragen,

Hosts. Die in Kapitel 10.5.2.3 erläuterten Services müssen Hosts angeben, an die sie gebunden sind. Optional kann für einen Host ein eigener Test definiert werden, der genutzt wird, wenn alle zugehörigen Services negativ getestet wurden.

Nun folgt ein Beispiel einer Nagiosdefinition für einen Host, die hinsichtlich nutzbarer Informationen untersucht werden soll.

```
define host{
        use                     generic-host
        host_name               novell1
        alias                   Novell Server #1
        address                 192.168.1.2
        check_command           check-host-alive
        max_check_attempts      10
        check_period            24x7
        notification_interval   120
        notification_period     24x7
        notification_options    d,u,r
        contact_groups          novell-admins
        }
```

Im Beispiel wird ein Rechner definiert, dessen genauere Beschreibung verrät, dass es sich um einen Novell Server handeln muss. Das Beispiel benutzt das Host-Template `generic-host`. Ein Host-Template ist dabei identisch zu einer Host-Definition aufgebaut mit dem zusätzlichen Attribut *name*, das den Templatenamen angibt. Alle dieses Template benutzenden Definitionen erben dann die Eigenschaften dieses Hosts, solange diese nicht mit den eigenen Attributen überschrieben werden. Weitere Informationen zu Templates befinden sich in Kapitel 10.5.4.

Die Pflichtattribute einer Host-Definition sind: `host_name`, `alias`, `address`, `max_check_attempts`, `check_period`, `notification_interval`, `notification_period`, `notification_options`, `contact_groups`. Weiterhin ist das Attribut `check_command` aufgeführt, welches aber nicht zwingend ist.

Inhalt der Attribute `host_name`, `alias` und `address` sind verständliche Attribute, die teilweise recht gut ihre Entsprechung im 3LGM²-Baukasten finden können. Zum `host_name` ist zu bemerken, dass dieser nicht der Rechnername sein muss. Der `host_name` wird genutzt, um innerhalb von Nagios dieses Objekt zu referenzieren. Zwei Host-Objekte dürfen daher auch nicht den selben `host_name` tragen. `alias` dient zur ausführlicheren Beschreibung des Geräts und `address` entspricht der IP-Adresse oder alternativ einem Fully Qualified Domain Name (FQDN).

`max_check_attempts`, `notification_interval`, `notification_period`, `notification_options` und `check_command` sind eher technisch orientiert und sollen nicht näher erläutert werden.

`check_period` und `contact_groups` enthalten Daten, die in einem 3LGM²-Modell verwendet werden können, da sie Verantwortlichkeiten und Verfügbarkeit eines Hosts beschreiben.

`check_period` gibt an, wann ein Host geprüft wird. Dafür wird ein anderes Nagiosobjekt vom Typ Timeperiod referenziert (hier mit dem Namen 24x7).
`contact_groups` bestimmt eine Gruppe von Personen, die in gewisser Weise Verantwortung für die Funktion des Geräts tragen.

optionale Attribute Hier sollen noch einige optionale Attribute aufgeführt werden, die ebenfalls für die Erstellung von 3LGM²-Modellen genutzt werden können. Das ist zunächst das Attribut `parents`. Dieses Attribut definiert die Abhängigkeit eines Hosts von einem oder mehreren anderen Hosts und damit natürlich auch die Abhängigkeit von den `parents` des `parents`. Über dieses einfache Konstrukt wird die Netzwerktopologie beschrieben. Auch 3LGM²-Modelle können die Topologie eines Netzwerkes darstellen. Nagios nutzt die Topologie, um entscheiden zu können, ob ein Gerät als DOWN oder als UNREACHABLE im Fehlerfall eingestuft wird. Die `parents`-Relation wird auch genutzt, um eine „Statusmap“ des Netzwerkes zu erzeugen (vgl. Abbildung 10.5).

Auch nutzbar ist das Attribut `hostgroups`, welches das Host-Objekt entsprechend vielen (vorher definierten) Gruppen zuordnet. Diese Gruppenbildung kann im 3LGM²-Modell zur Zusammenfassung mehrerer Elemente genutzt werden, beispielsweise in Form einer *ist_teil_von*-Beziehung.

10.5.2.2 Nagiosobjekttyp Hostgroup

In einer Hostgroup werden mehrere Geräte die Hosts sind, in einer Gruppe zusammengefasst. Das eben in der Host-Definition gesehene Attribut `hostgroups` kann zusätzlich oder alternativ verwendet werden. Vorausgesetzt, die in der Host-Definition verwendete Gruppe ist definiert.

Eine solche Hostgroup-Definition hat die folgende Struktur:

```
define hostgroup{
        hostgroup_name  db-server
        alias           database server
        members         db1, db2, db3
        }
```

Inhalt der Attribute Alle verwendeten Attribute sind auch obligatorische Attribute.
`hostgroup_name` ist der Name der Gruppe, auf den in Nagios immer wieder verwiesen wird. `alias` dient wie in der Host Definition zur näheren Beschreibung des Objekts. `members` listet Geräte auf, die zur Gruppe gehören und definiert sind. Zur Gruppe gehören außerdem die Geräte, bei denen in der Host-Definition im Attribut `hostgroups` diese entsprechende Gruppe eingetragen ist.

10.5.2.3 Nagiosobjekttyp Service

Als Services sind Dienste definiert, die geprüft werden können. Dabei sei noch einmal erwähnt, dass Nagios alle Überprüfungen durch Plugins durchführt. Die Standardausstattung von Plug-

ins unterstützt im Wesentlichen allgemeine Netzwerkdienste wie SMTP, POP und HTTP. Daraus lassen sich für den 3LGM²-Baukasten nur wenige Informationen gewinnen, da insbesondere beim Modellieren von Krankenhausinformationssystemen solche Dienste eine untergeordnete Rolle spielen. Allerdings gibt es sehr viele weitere Plugins, die insbesondere Datenbanken prüfen, die wiederum im 3LGM²-Baukasten modelliert werden können.

Eine Service-Definition hat beispielhaft folgende Struktur. Alle angegebenen Attribute sind obligatorisch.

```
define service{
        host_name               linux-server
        service_description     check-disk-sda1
        check_command           check-disk!/dev/sda1
        max_check_attempts      5
        normal_check_interval   5
        retry_check_interval    3
        check_period            24x7
        notification_interval   30
        notification_period     24x7
        notification_options    w,c,r
        contact_groups          linux-admins
        }
```

Inhalt der Attribute Viele der hier angegebenen Attribute sind wieder technischer Natur. Dazu zählen: `check_command`, `max_check_attempts`, `normal_check_interval`, `retry_check_interval`, `notification_interval`, `notification_period`, `notification_options`.

Das Attribut `host_name` gibt die Referenz auf den Host an, auf dem der Service läuft. Beschrieben wird der Dienst in `service_description`, was Rückschlüsse auf das verwendete Softwareprodukt liefern könnte. Da es kein Namensattribut gibt, werden `service_description` und `host_name` gemeinsam zur Referenzierung eines Services herangezogen. Ähnlich wie bei den Hosts können auch für Services Zeiten definiert werden, zu denen die Überprüfung stattfindet. Informationen dazu finden sich in `check_period`. Ebenfalls als Attribut vorhanden und in den 3LGM²-Baukasten integrierbar sind die für diesen Dienst verantwortlichen Personengruppen, die in `contact_groups` definiert sind. Eine Möglichkeit einen Service einer Gruppe zuzuordnen, besteht im Auflisten der Gruppe im Feld `servicegroups` oder alternativ in der Definition der entsprechenden Service-Gruppe wie im nächsten Abschnitt beschrieben.

Es gibt noch eine Vielzahl von optionalen Attributen, die aber alle technisch orientiert sind und daher hier nicht behandelt werden.

10.5.2.4 Nagiosobjekttyp Servicegroup

Die Servicegroups dienen ähnlich wie die Hostgroups der Zusammenfassung von Diensten und stellen damit eine Verallgemeinerung ihrer enthaltenen Dienste dar.

Die Struktur einer Servicegroup-Definition hat folgendes Aussehen:

```
define servicegroup{
        servicegroup_name       switch-tests
        alias                   several tests for switches
        members                 switch1, switch2
        }
```

Dabei sind alle aufgeführten Attribute obligatorisch und es gibt keine weiteren optionalen Attribute.

Inhalt der Attribute Angegeben werden muss in `servicegroup_name` der Name der Gruppe, in `alias` die möglichst aussagekräftige Beschreibung und in `members` die Services, die zu dieser Gruppe gehören. Services können alternativ oder zusätzlich auch eine Gruppe in ihrer Service-Definition angeben.

10.5.2.5 Nagiosobjekttyp Timeperiods

Mit Timeperiods werden Zeitphasen festgelegt, die in allen anderen Objekten, sofern eine Zeitspanne anzugeben ist, verwendet werden können. Die Verfügbarkeit von Services und Hosts wird ebenfalls mit Timeperiods bestimmt. Zeitspannen festzulegen, kann sehr sinnvoll sein, da zum Beispiel eine Zeitspanne „Arbeitszeit_Funktionsabteilung“ dadurch nur einmal definiert und mehrfach verwendet werden kann. Gleichzeitig muss diese Zeitspanne im Bedarfsfall auch nur einmal geändert werden.

Ein Beispiel:

```
define timeperiod{
        timeperiod_name         nonworkhours
        alias                   Non-Work Hours
        sunday                  00:00-24:00
        monday                  00:00-09:00,17:00-24:00
        tuesday                 00:00-09:00,17:00-24:00
        wednesday               00:00-09:00,17:00-24:00
        thursday                00:00-09:00,17:00-24:00
        friday                  00:00-09:00,17:00-24:00
        saturday                00:00-24:00
        }
```

Inhalt der Attribute Das Prinzip ist recht einsichtig, hat aber offensichtlich auch Einschränkungen. Zeitperioden wiederholen sich wöchentlich. Obligatorisch sind bei dieser Definition nur

`timeperiod_name` und `alias`, die wie zuvor in jeder Definition den in Nagios verwendeten Namen und eine Beschreibung enthalten. Die Attribute „Wochentage“ können auch weggelassen werden, wenn an ganzen Tagen die Zeitperiode nicht zutrifft.

10.5.2.6 Nagiosobjekttyp Contactgroup

Wie schon die Host- und Servicegroups, so sind auch die Contactgroups eine Zusammenfassung von Contacts. Die Kontaktdaten der einzelnen Administratoren sollen für den 3LGM²-Baukasten aber nicht interessieren. Wie informatiosreich die Contactgroups sind, ist leider sehr von der Beschreibungsqualität im Feld `alias` abhängig.

Zunächst aber eine Beispiel-Definition einer Contactgroup:

```
define contactgroup{
        contactgroup_name       novell-admins
        alias                   Novell Administrators
        members                 jdoe,rtobert,tzach
        }
```

Inhalt der Attribute Alle hier aufgeführten Attribute sind dabei obligatorisch. Der Name im Attribut `contactgroup_name` stellt wieder die nagiosinternen Referenzen her und die Beschreibung im Attribut `alias` soll weitere Informationen zur Gruppe liefern. Die Sammlung der Mitglieder einer Gruppe im Attribut `members` ist potenziell nicht vollständig, da auch im Nagiosobjekt Contact eine Gruppenzugehörigkeit definiert werden könnte.

10.5.3 Vergleich mit den HSKs

Auf Basis der Informationen, die in den Konfigurationsdateien stecken, lassen sich einige HSKs ausschließen, da Nagios keinerlei Informationen zu ihnen enthält.

Auf der Fachlichen Ebene

- *Organisationseinheit*

Auf der Logischen Werkzeugebene

- *Kommunikationsstandard*

Auf der Physischen Werkzeugebene

- *Netzprotokoll*
- *Netztyp*
- *Standort*

10.5.4 Vererbungsbeziehungen durch Nagiostemplates

Am Beispiel der Definition von Hosts wurde bereits auf Nagiostemplates hingewiesen. Nagiostemplates werden genutzt, um Vererbungsbeziehungen[1] auszudrücken. Alle Objektdefinitionen unterstützen die Nutzung von Nagiostemplates. Dabei erbt eine Nagiosdefinition von dem Nagiostemplate, welches sie benutzt und zeigt diese Nutzung mit dem Attribut `use` an. Wenn von einer Nagiosdefinition geerbt wird, dann muss diese anzeigen, dass es sich bei ihr um ein Nagiostemplate handelt. Dies geschieht über das Attribut `name`, das unabhängig von anderen Attributen wie `host_name` oder `servicegroup_name` ist. Weiterhin muss ein Nagiostemplate anzeigen, ob es sich um ein *abstraktes*[2] oder *reales* Nagiostemplate handelt. Dies geschieht mittels des Attributs `register`, das den Wert 0 (*abstrakt*) oder 1 (*real*) enthalten kann. Alle drei Attribute (`use`, `name`, `register`) können optional in allen Nagiosdefinitionen verwendet werden.

Das folgende Beispiel zeigt eine Vererbungsbeziehung. Dabei erbt der Host `bighost2` von dem Host-Template `hosttemplate1`, welches durch den Host `bighost1` definiert wird. Überwacht werden daraufhin zwei Hosts: `bighost1` und `bighost2`. Würde in der Definition von `bighost1` das Attribut `register` mit dem Wert `0` stehen, dann wäre bighost1 *abstrakt* und Nagios würde nur `bighost2` überwachen.

```
define host{
        host_name               bighost1
        check_command           check-host-alive
        notification_options    d,u,r
        max_check_attempts      5
        name                    hosttemplate1
        }

define host{
        host_name               bighost2
        max_check_attempts      3
        use                     hosttemplate1
        }
```

Auch Nagiostemplates können wiederum Nagiostemplates benutzen, wodurch eine komplexe Vererbungshierarchie entstehen kann. Die Vererbung stellt eine Vererbung zwischen Definitionen dar und nicht zwischen Objekten. Im laufenden Nagiossystem wird an keiner Stelle ersichtlich, ob ein Nagiosobjekt durch eine Definition erstellt wurde, die geerbt hat.

10.5.5 Reguläre Ausdrücke in Nagios

Auch Nagios steht dem selben Problem wie der 3LGM²-Baukasten gegenüber: das Beschreiben von komplexen Informationssystemen ist sehr arbeits- und zeitaufwändig. Um die Arbeit beim

[1] Ohne „Vererbung“ diskutieren zu wollen, wird die Nutzung von Nagiostemplates in der Nagiosdokumentation als „Inheritance“ bezeichnet.

[2] „Abstrakt“ soll bedeuten, dass kein Nagiosobjekt aus dieser Nagiosdefinition entsteht.

Konfigurieren von Nagios an den Stellen zu minimieren, an denen es sich um analoge Sachverhalte handelt, ist es möglich reguläre Ausdrücke in Nagiosmultidefinitionen zu verwenden. Mit einer Nagiosmultidefinition kann man so eine Vielzahl gleichartiger Nagiosdefinitionen erstellen.

Hier ergibt sich eine Lücke in der Terminologie, die jetzt geschlossen werden soll, um im Weiteren Missverständnissen vorzubeugen.

Festlegung Werden in einer *Nagiosdefinition* reguläre Ausdrücke verwendet, so wird diese zu einer *Nagiosmultidefinition* und ist keine Nagiosdefinition mehr. Eine *Nagiosmultidefinition* besteht aus einer endlichen Anzahl von Nagiosdefinitionen.

Zum besseren Verständnis ein Beispiel:

```
define service{
               host_name               *
               service_description     SOMESERVICE
               other service directives ...
               }
```

Der Asterisk (*) als Wert des Attributs `host_name` zeigt an, dass alle Host-Objekte diesen Dienst (`SOMESERVICE`) unterstützen.

Nagios stellt 3 verschieden „starke“ Möglichkeiten zur Wahl:

***-Operator und Aufzählung** Standardmäßig können Aufzählungen (z. B.: rechner1, rechner2, rechner3) und der *-Operator (alle passenden Objekte) bei Nagiosmultidefinitionen für Servicedefinitionen[3] verwendet werden. Bei Servicedefinitionen kann auch an Stelle des Attributs `host_name` das Attribut `hostgroup_name` verwendet werden, um damit Servicedefinitionen für alle Hosts, der durch das Attribut `hostgroup_name` bestimmten Hostgroups zu erstellen.

*** und ?-Operator** Durch Aktivieren der Option `use_regexp_matching`[4] können der *-Operator (beliebig viele Zeichen) und der ?-Operator (genau ein beliebiges Zeichen) in Objektnamen verwendet werden.

reguläre Ausdrücke Durch Aktivieren der Option `use_true_regexp_matching`[5] können Objektnamen mit regulären Ausdrücken nach dem POSIX-Standard[6] verwendet werden.

Durch reguläre Ausdrücke ist es nie möglich, unendlich viele Objekte zu erstellen. Sie können nur genutzt werden, um gleich mehrere Objekte zu referenzieren. Definitionen, die reguläre Ausdrücke enthalten, können somit immer aufgelöst werden, indem man alle ableitbaren Definitionen auflistet.

[3] teilweise auch möglich für Service Escalations, Service Dependencies, Host Escalations, Host Dependencies, Hostgroups

[4] in der allgemeinen Konfigurationsdatei nagios.cfg

[5] ebenfalls in der allgemeinen Konfigurationsdatei nagios.cfg

[6] wie diese in Linux umgesetzt werden, kann nach Eingabe des Linuxbefehls `man 7 regex` nachgelesen werden

Ebenso wie die Vererbung mittels Nagiostemplates ist die Nutzung von regulären Ausdrücken im laufenden Nagios nicht mehr nachzuvollziehen.

Ein UML-Klassendiagramm soll noch einmal den Zusammenhang der verwendeten Begriffe in Abbildung 10.2 verdeutlichen.

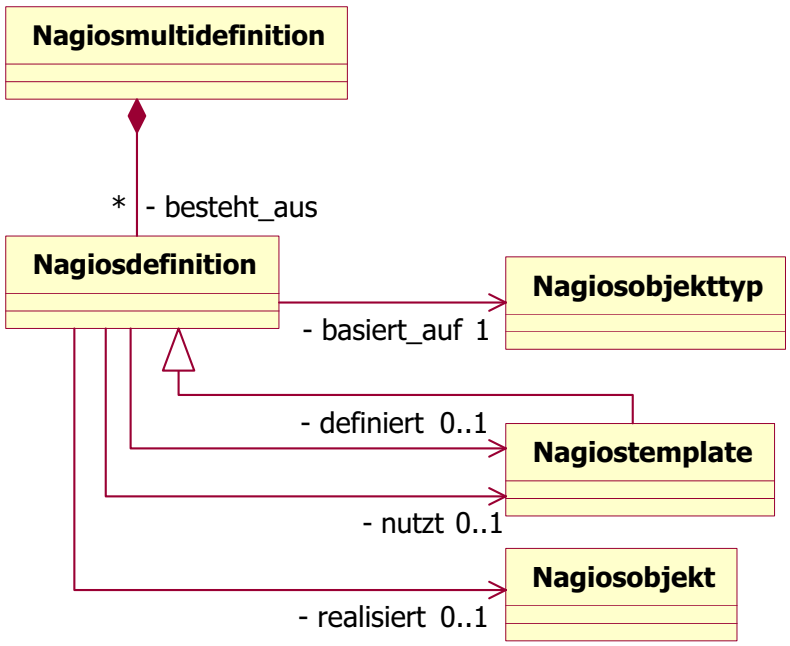

Abbildung 10.2: Die eingeführte Terminologie als Klassendiagramm dargestellt.

10.5.6 Interne und externe Relationen

Festlegung Alle Relationen zwischen den Nagiosobjekten Host, Service, Hostgroup, Servicegroup, Timeperiods und Contactgroup, inklusive der durch das `parents`-Attribut entstehenden Topologie, sollen im Folgenden als *interne Relationen* bezeichnet werden.

Das „intern“ soll verdeutlichen, dass diese Relationen im laufenden Nagios ersichtlich werden.

Annahme Die Verwendung von Nagiostemplates in Nagiosdefinitionen, die durch die Verwendung des Attributs `use` entsteht, wird als Ausdruck einer Gemeinsamkeit gesehen. Zwischen den Nagiosdefinitionen entsteht eine *Vererbungs-Relation*.

Die Verwendung von Nagiosmultidefinitionen für die Erstellung von Nagiosdefinitionen wird ebenfalls als Ausdruck einer Gemeinsamkeit gesehen. Zwischen den Nagiosmultidefinitionen und den abgeleiteten Nagiosdefinitionen entsteht eine *Inklusions-Relation*.

Festlegung Alle Vererbungs-Relationen und Inklusions-Relationen zwischen Nagiosdefinitionen, die durch das `use`-Attribut und/oder durch die Verwendung von Nagiosmultidefinitionen entstehen, sollen im Folgenden als *externe Relationen* bezeichnet werden.

Dieser Sachverhalt ist zusätzlich in Abbildung 10.3 dargestellt.

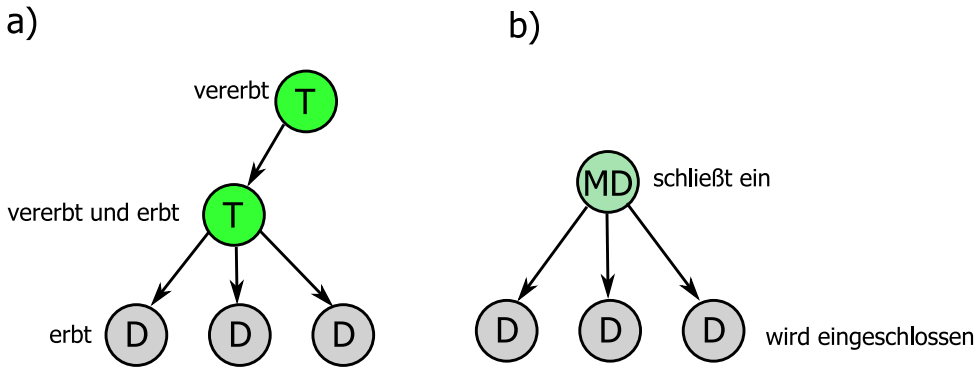

Abbildung 10.3: Die Vererbungs-Relation ist in Teilbild a) und die Inklusions-Relation ist in Teilbild b) dargestellt. D = Definition, T = Nagiostemplate, MD = Nagiosmultidefinition

Das „extern“ soll verdeutlichen, dass diese Relationen im laufenden Nagios nicht nachvollzogen werden können.

Interne Relationen haben direkten Einfluss auf das Nagiossystem. Werden diese Relationen nicht sinngemäß verwendet, dann funktioniert Nagios auch nicht wie erwünscht. Im Gegensatz dazu können nicht sinngemäß verwendete externe Relationen durchaus korrekte Objekte erzeugen und damit für den Nutzer unbemerkt bleiben. Daher sind die internen Relationen für Rückschlüsse auf das zugrunde liegende Netzwerk den externen Relationen vorzuziehen. In Abbildung 10.4 ist eine schematische Darstellung interner und externer Relationen zu sehen.

10.5.7 Interpretation interner und externer Relationen

Annahme Die Relationen zwischen den Nagiosobjekten Host, Service, Hostgroup, Servicegroup, Timeperiods und Contactgroup, inklusive der durch das `parents`-Attribut entstehenden Topologie, werden entsprechend der von Nagios angedachten Verwendung benutzt.

Hier wird angenommen, dass Nagiosdefinitionen sinngemäß verwendet werden, ohne „sinngemäß“ näher zu spezifizieren. Dies ist möglich, da die Verwendung der einfacheren Relationen bereits bei der Erläuterung der Nagiosobjekte deutlich geworden sein sollte.

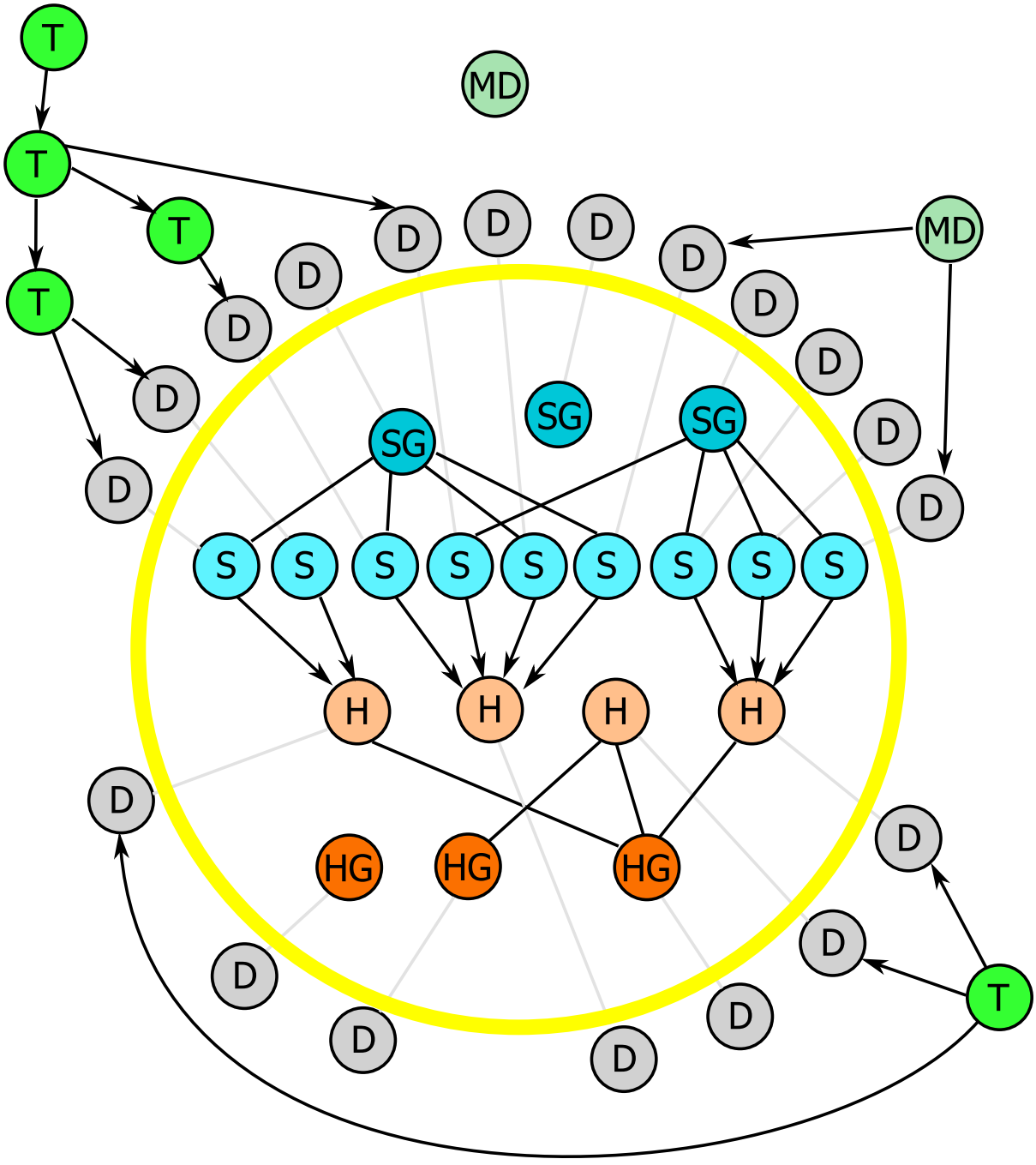

Abbildung 10.4: Schematische Darstellung einer Nagioskonfiguration mit internen Relationen (im Kreis) und externen Relationen (außerhalb des Kreises). S = Service, H = Host, HG = Hostgroup, SG = Servicegroup, D = Definition, T = Nagios-template, MD = Nagiosmultidefinition

Dazu zählen:

Service →[7] Host
Host ↔ Hostgroup
Host → Timeperiod
Host → Contactgroup
Service ↔ Servicegroup
Service → Timeperiod
Service → Contactgroup

Hier sollen nun weitere, bisher nicht ausreichend beschriebene Relationen auf ihre Semantik[8] hin untersucht werden.

Dabei besteht besonderes Interesse an der Identifizierung von hierarchischen[9] Beziehungen, da diese der Ansatzpunkt für die Beherrschung von großen Mengen von Elementen sind. Hierarchien können Verallgemeinerungen aufzeigen, die es zu nutzen gilt.

10.5.7.1 Semantik interner Relationen

Von den semantisch bereits untersuchten Relationen sind Host ↔ Hostgroup und Service ↔ Servicegroup einstufige Hierarchien. Dabei ist eine Hostgroup bzw. eine Servicegroup jeweils der Vaterknoten der Hosts bzw. Services.

`parents`-Relation(Host → Host) Bei einer `parents`-Relation ist der durch `parents` referenzierte Host dem referenzierenden Host übergeordnet. Für Nagios bedeutet das: Die Überprüfbarkeit des referenzierenden Hosts durch Nagios ist abhängig von der Überprüfbarkeit des referenzierten Hosts. Ein Beispiel ist in Abbildung 10.5 zu sehen. Der Wurzelknoten der `parents`-Hierarchie ist der Rechner, auf dem Nagios läuft.

10.5.7.2 Semantik externer Relationen

Vererbungs-Relation Die aus der Vererbungsrelation abgeleitete Vererbungshierarchie stellt alle Nagiosdefinitionen, die Nagiostemplates nutzen oder als Nagiostemplate genutzt werden, zueinander in Beziehung. Nagiostemplates sind Vorgaben für Nagiosdefinitionen und stellen in dieser Hierarchie einen Vaterknoten gegenüber den von ihnen erbenden Nagiosdefinitionen dar. Dass alle von einem Nagiostemplate übernommenen Attribute und Attributwerte neu überschrieben oder auf „null“ gesetzt werden können, ist eine nicht sinngemäße Verwendung.

Inklusions-Relation Die Inklusions-Relation kann zur Gruppierung genutzt werden, da alle beteiligten Nagiosdefinitionen untereinander bis auf Referenzen identisch sind. Die Inklusions-Relation ist immer einstufig.

[7] verkürzte UML-Schreibweise für eine Assoziation; → unidirektional bzw. ↔ bidirektional
[8] Semantik ist die Bedeutung syntaktisch korrekter Konstrukte. (vgl. [APPELRATH et al. 00] S.194)
[9] System von Elementen, die einander über- bzw. untergeordnet sind

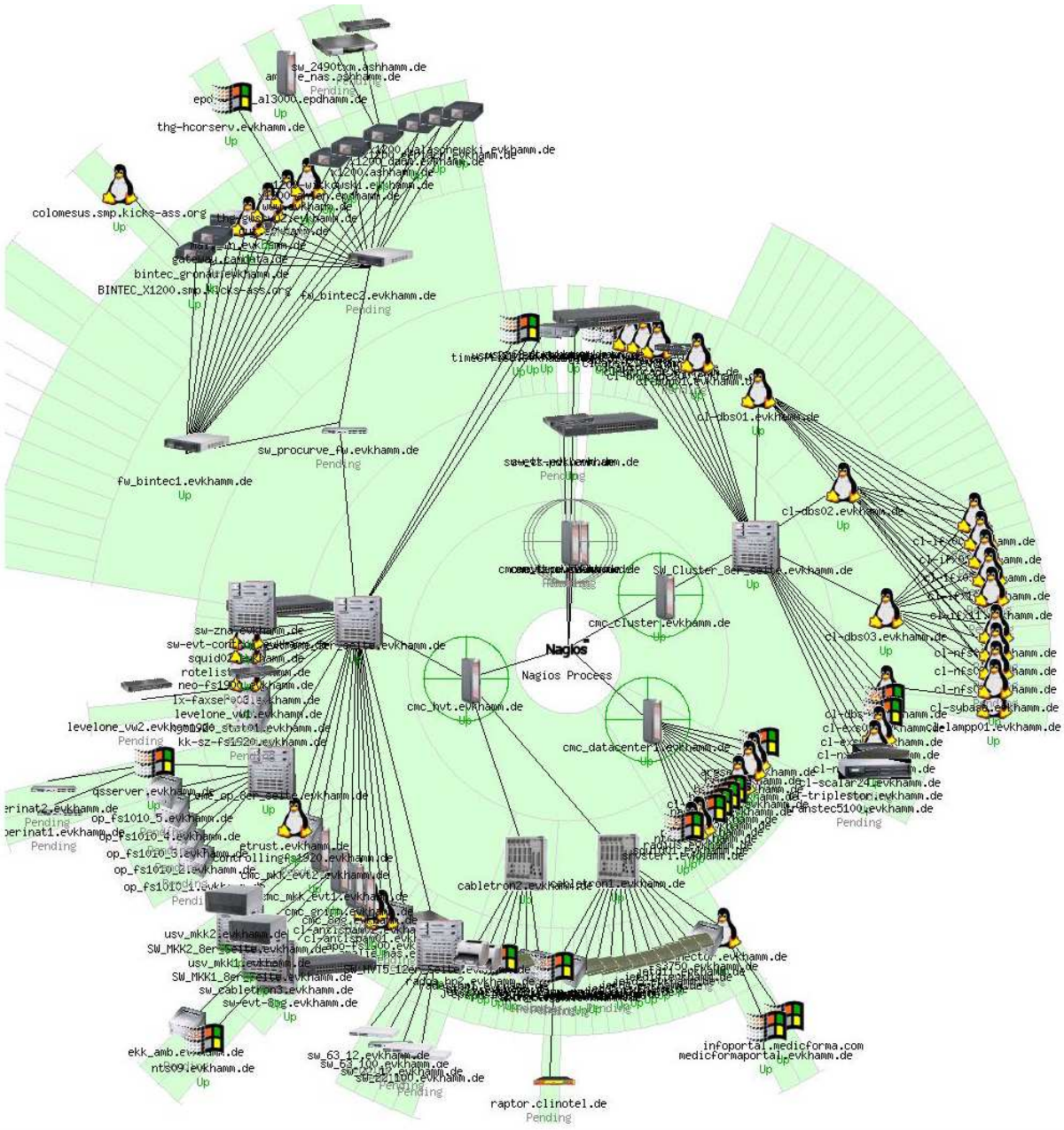

Abbildung 10.5: Beispiel einer „Statusmap“ aus Nagios. Jede Verbindung ist eine `parents`-Relation. Je weiter „innen“ ein Knoten ausfällt („DOWN“) um so mehr äußere Knoten sind nicht mehr überprüfbar („UNKNOWN“).
(Quelle: Ev. Krankenhaus Fördergesellschaft Hamm)

10.6 Zusammenfassung

Dieses Kapitel erläuterte die Grundlagen von Nagios, so dass nun eine Vorstellung darüber existiert, welche Daten in Nagioskonfigurationen zu finden sind und welche nicht. Auch die Beziehungen der einzelnen Nagiosobjekte untereinander wurden erwähnt und Besonderheiten, wie reguläre Ausdrücke vorgestellt. Darauf aufbauend soll die Referenzarchitektur in Kapitel 12 umgesetzt werden.

11 Beispiel einer Nagioskonfiguration

Inhaltsangabe

11.1 Analysewerkzeug 79
11.2 Umfang und Erstellung 79
11.3 Konfigurationsinhalt 81
11.4 Zusammenfassung 83

Nagios rühmt sich einer großen Variabilität, die es ermöglicht, neben Rechnerparametern auch diverse andere Parameter zu überwachen. Allerdings werden nicht alle Möglichkeiten von Nagios für jeden Anwendungsfall benötigt. Um einen besseren Einblick in die praktische Anwendung von Nagiosobjekten zu gewinnen, wird in diesem Kapitel eine konkrete Nagioskonfiguration betrachtet. Es handelt sich dabei um Daten aus dem Evangelischen Krankenhaus in Hamm, einem Krankenhaus mit ca. 450 Betten. Möglich war dies nur durch die freundliche Unterstützung von Herrn Dr. Dipl. Inf. Michael Raus, dem Leiter der dortigen IT-Abteilung. Weiterhin standen die Daten des IMISE zur Verfügung, die deutlich weniger Nagiosobjekte enthielten, aber nicht weniger aufschlussreich waren.

11.1 Analysewerkzeug

Für die Analyse von Nagioskonfigurationen dient ein zu dieser Arbeit entwickelter Prototyp der Integrationskomponente. Zur Veranschaulichung von Nagioskonfigurationen wurden diese eingelesen und als Graph dargestellt. Der Graph enthält die in Kapitel 10.5.2 erörterten Nagiosobjekte[1] als Knoten und jeder Verweis eines solchen Nagiosobjekts auf ein anderes ist als Kante dargestellt. Die „Statusmap“ ist daher ein Teilgraph dieses Graphen. Der Prototyp verfügt über Filter, um die Knoten auf das Wesentliche reduzieren zu können. In Abbildung 11.1 ist beispielhaft der Graph der Nagioskonfiguration des Evangelischen Krankenhauses in Hamm dargestellt. Dort sind der Host „cl-dbs03.evkhamm.de“ und seine Services sowie Hostgroups, Hosttemplates und übergeordnete bzw. untergeordnete Hosts hervorgehoben.

11.2 Umfang und Erstellung

Die Daten des Krankenhauses Hamm enthalten über 600 Nagiosobjekte[2] mit entsprechend vielen Verweisen untereinander. Dabei werden keine Arbeitsplatzrechner abgebildet, sondern im

[1] Host, Hostgroup, Service, Servicegroup, Timeperiod und Contactgroup
[2] nur Host, Hostgroup, Service, Servicegroup, Timeperiod und Contactgroup

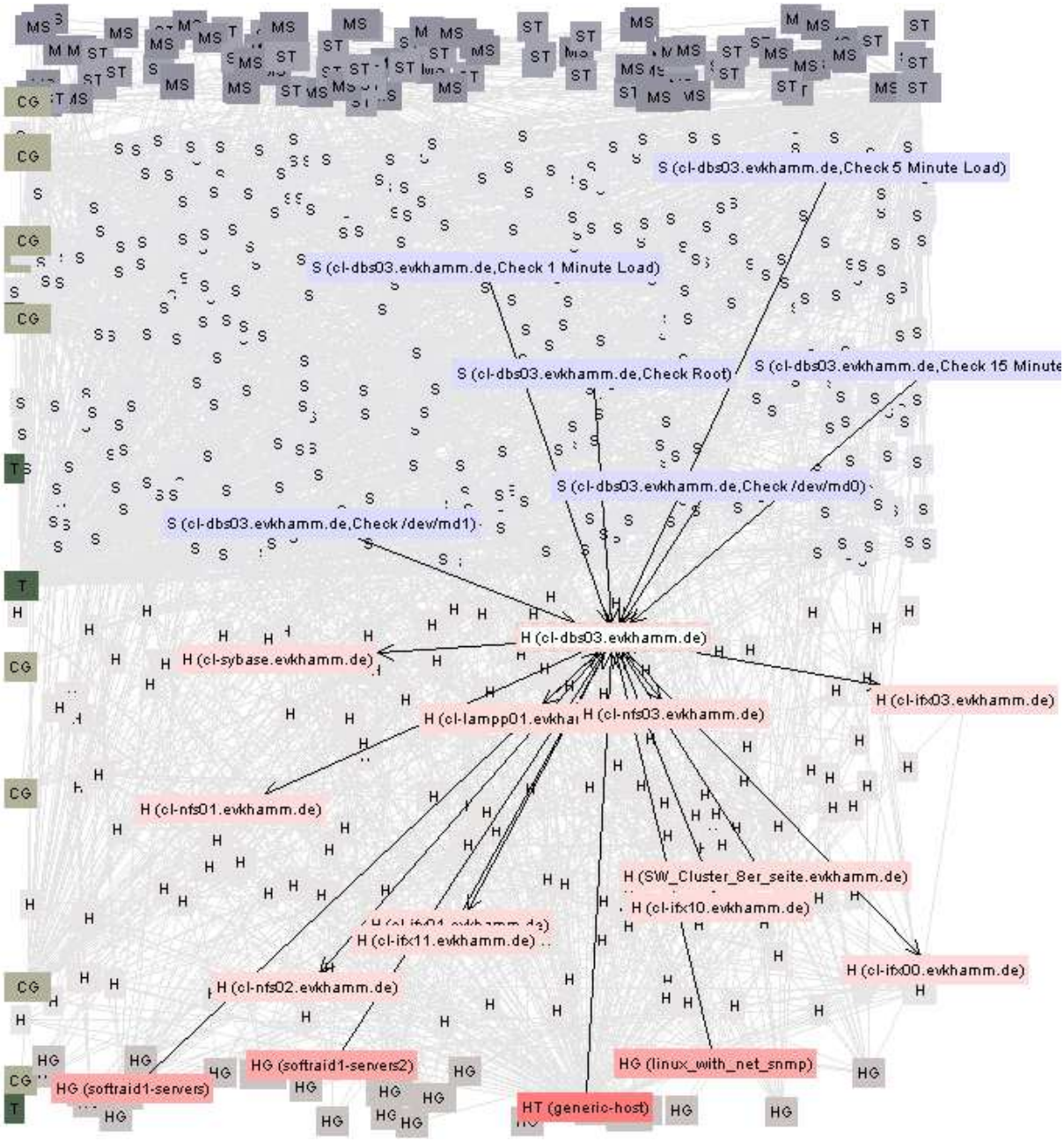

Abbildung 11.1: Graphische Darstellung der Konfigurationsdaten aus Hamm durch den Prototyp.

Wesentlichen Server und Switches. Da sich solch eine große Anzahl von Nagiosobjekten nicht mehr sinnvoll von Hand in Konfigurationsdateien verwalten lässt, wurde die Software NagMIN[3] genutzt. NagMIN ermöglicht es, Informationen über Nagiosobjekte in einer Datenbank zu halten und diese mit Hilfe eines Webbrowsers relativ komfortabel zu verwalten. Die Konfigurationsdateien werden bei Änderungen aus den Daten der Datenbank erzeugt. NagMIN ist dabei ein Plugin für die Software WebMIN[4], welche noch weitere Dienste für Administratoren zur Verfügung stellt. NagMIN nutzt die Möglichkeiten von Nmap[5], mit dem es möglich ist, die Topologie des Netzwerkes weitgehend automatisch abzubilden. Die Statusmap der Nagioskonfiguration des Krankenhauses in Hamm ist in Abbildung 10.5 dargestellt. Das Netzwerk des IMISE wird manuell in den Konfigurationsdateien beschrieben, wobei dabei die Topologie des Netzwerkes nicht mit einfließt.

11.3 Konfigurationsinhalt

Um einen ersten Eindruck über die Ausdrucksfähigkeit eines Modells zu erhalten, kann man eine kleine Statistik erheben. Der später noch vorgestellte Prototyp zu dieser Arbeit erstellt eine solche Statistik.

Die Ausgabe des Prototypen für die Konfigurationsdaten aus Hamm:

```
kleine Statistik
Anzahl der Objekte vom Typ Host: 143
Anzahl der Objekte vom Typ Hostgroup: 26
Anzahl der Objekte vom Typ Service: 354
Anzahl der Objekte vom Typ Multi-Service: 56
Anzahl der Objekte vom Typ Contactgroup: 10
Anzahl der Objekte vom Typ Timeperiod: 4
Anzahl der Objekte vom Typ Host-Template: 1
Anzahl der Objekte vom Typ Service-Template: 45
Services pro Host 2,48
Parents pro Host 1,18
Hostgroups/Templates pro Host 2,06
Servicegroups/Templates/Multi-Services pro Service 1,89
Gruppierungsqualität 1,94
Objekte:639
```

Die Ausgabe des Prototypen für die Konfigurationsdaten aus dem IMISE:

```
kleine Statistik
Anzahl der Objekte vom Typ Host: 22
Anzahl der Objekte vom Typ Hostgroup: 8
Anzahl der Objekte vom Typ Service: 71
```

[3] http://nagmin.sourceforge.net/
[4] http://www.webmin.com/
[5] ein Network Mapper zum (automatischen) Durchsuchen von Netzwerken, http://insecure.org/nmap/

```
Anzahl der Objekte vom Typ Contactgroup: 3
Anzahl der Objekte vom Typ Timeperiod: 4
Anzahl der Objekte vom Typ Host-Template: 1
Anzahl der Objekte vom Typ Service-Template: 1
Services pro Host 3,23
Parents pro Host 0,00
Hostgroups/Templates pro Host 2,05
Servicegroups/Templates/Multi-Services pro Service 1,00
Gruppierungsqualität 1,25
Objekte:110
```

Interessant für die Datenintegration ist dabei besonders die Gruppierungsqualität, die gewichtet aus Hostgroups/Templates pro Host und Servicegroups/Templates/Multi-Services pro Service zusammengesetzt ist. Sie ist ein Ausdruck für die Anzahl der Gruppen, denen ein Nagiosobjekt zugeordnet ist. Dabei muss beachtet werden, dass es oftmals die Standard-Templates „generic-service“ und „generic-host“ gibt, zu denen alle Services bzw. alle Hosts gehören, die aber keine für die Datenintegration verwertbare Semantik haben. Eine Konfiguration, die nur diese beiden Standard-Templates verwendet, hat schon eine Gruppierungsqualität von 1.

Im Folgenden wird die Verwendung der Nagiosobjekte in den Daten des Evangelischen Krankenhauses in Hamm kurz beschrieben.

Host Hosts sind in den Nagiosdaten zumeist Server, Switches, Printserver und USVs. Es gibt nur wenige Ausnahmen, wie z. B. den Host „cmc_hvt.evkhamm.de“, bei dem es sich um eine Computer-Multi-Control-Komponente (CMC) zur Raumüberwachung handelt. Die Beschreibung der Hosts entspricht leider meist den Hostnamen und ist zugleich der Fully Qualified Domain Name, über den die Hosts angesprochen werden können. Allerdings gibt es noch einen NagMIN-eigenen Kommentar, der von Nagios nicht genutzt wird, aber weitere Informationen enthält.

Mit den `parents`-Relationen wird die Topologie des Netzwerkes nachgebildet, inklusive der an manchen Stellen vorhandenen Redundanz. Die meisten Hosts gehören zu einer oder mehreren Hostgroups. Welche Beziehungen ein Host in diesen Daten haben kann, ist in Abbildung 11.1 zu sehen. Leider ist die Topologiebeschreibung nicht ganz im Sinne der Nagios Dokumentation [6], da Hosts von den oben erwähnten Computer-Multi-Control-Komponenten abhängig sind, obwohl sie auch bei einer Fehlermeldung dieser Komponenten (z. B. Serverschranktür ist geöffnet) noch funktionieren würden. Diese Vorgehensweise, Abhängigkeiten, die keinen topologischen Ursprung haben, über die Topologie zu beschreiben, macht es bei der Datenintegration sehr schwer, „echte“ `parents`-Relationen von „falschen“ zu unterscheiden. Leider wird auch in [BARTH 05] die alternative Verwendung von `parents`-Relationen beschrieben, aber es wird auch eine Lösungsmöglichkeit mit Hilfe von Service und Service-Dependencies vorgeschlagen, die die Topologie nicht beeinträchtigt.

[6] http://nagios.sourceforge.net/docs/2_0/networkreachability.html

Hostgroups Die Hostgroups beschreiben Typen von Netzwerkkomponenten, z. B. „cabletron-switches“, „windows-server“ und „jetdirect-boxes“, wobei ein Host durchaus gleichzeitig zu „cabletron-switches“ und „switches“ gehören kann. Jeder Host gehört nach Abzug von „generic-host“ zu weiteren 1,05 anderen Hostgroups/Templates.

Services Die Dienste, die geprüft werden, umfassen Netzwerkdienste wie SMTP, POP und HTTP aber auch Hardware Parameter wie freien Speicherplatz, CPU-Belastung, Lüfterfunktionen und Stromversorgung sowie Raumeigenschaften wie offene Türen, Temperaturen und Luftfeuchtigkeit. Auch die Verfügbarkeit von Anwendungen, die für ein 3LGM²-Modell wichtig sind, werden geprüft, z. B. RoteListe, Medicforma Infoportal und THG-Server[7].

Servicegroups Leider sind keine Servicegroups im Modell verwendet worden.

Contactgroups und Timeperiods Es existieren verschiedene Contactgroups wie „tape-Admins“ und „server-Admins“, aber auch die Gruppe „kdti-admins“, die für einen externen Dienstleister gedacht ist.

Templates Es gibt in der Konfiguration die beiden Standard-Templates „generic-service“ und „generic-host“, denen alle Hosts, aber nicht alle Services untergeordnet sind. Services sind teilweise auch dem Template „NM-SMTP“ oder „NM-SNMP“ untergeordnet, die jeweils Vorgaben für die Verwendung eines bestimmten Protokolls enthalten. Es gibt für verschiedene Protokolle eine Reihe weiterer Service-Templates, die von keinem Service genutzt werden.

11.4 Zusammenfassung

Die Betrachtung der Nagioskonfigurationen hat gezeigt, dass diese durchaus sehr unterschiedlich sein können. Zum einen unterschiedlich hinsichtlich der genutzten Möglichkeiten von Nagioskonfigurationen und zum anderen hinsichtlich der Konformität mit den Konzepten von Nagioskonfigurationen. Diese Erkenntnis erschwert natürlich die folgende Umsetzung der Referenzarchitektur, da es nur wenige verlässliche Regeln für Nagioskonfigurationen gibt.

[7] Thorax-, Herz und Gefäßchirurgie Telematikverbindung zum Universitätsklinikum Münster

12 Umsetzung der Referenzarchitektur für Nagios und 3LGM²-Baukasten

Inhaltsangabe

12.1 Vorüberlegungen . . . 86
12.1.1 Ziel . . . 86
12.1.2 Informationsbedarf und Datenmodell . . . 86
12.1.3 Modellierung der lokalen Datenstruktur des 3LGM²-Baukastens . . . 87
12.1.4 Modellierung der lokalen Datenstruktur von Nagios . . . 87
12.1.5 Aufbau der globalen Datenstruktur . . . 87
12.1.6 Erstellen des globalen Datenbestandes . . . 87
12.1.7 Physische Integration . . . 91
12.2 Module zum Finden von Zuordnungen . . . 91
12.2.1 Modul Rechnernetz . . . 91
12.2.2 Modul Physischer Datenverarbeitungsbaustein . . . 92
12.2.3 Modul Hostgroup . . . 93
12.2.4 Modul Servicegroup . . . 94
12.2.5 Modul Bausteintyp . . . 95
12.2.6 Modul Contactgroup/Timeperiod . . . 95
12.2.7 Modul ist_verbunden_mit-Relation . . . 95
12.2.8 Modul Anwendungsprogramm . . . 96
12.2.9 Modul Datenbanksystem . . . 97
12.2.10 Modul Bausteinschnittstelle . . . 98
12.2.11 Modul Servicetemplate . . . 98
12.2.12 Modul Hosttemplate . . . 99
12.2.13 Modul Multiservice . . . 99
12.2.14 Modul 3LGM²-Modellelementfelder . . . 99
12.3 Ausblenden von Nagiosobjekten . . . 99
12.4 Filtern von Services . . . 100
12.5 Vermeiden von identischen 3LGM²-Modellelementen . . . 100
12.6 Persistenz des globalen Datenbestandes und der Nagiosschlüssel . . . 100
12.7 Synchronisation . . . 103
12.8 Implementierung . . . 105
12.9 Umsetzung der Anforderungen . . . 106
12.9.1 Funktionelle Anforderungen . . . 106

12.9.2 Nicht funktionelle Anforderungen 106
12.10 Erweiterung der Umsetzung 107
12.11 Zusammenfassung 107

Im vorangegangenen Kapitel wurde theoretisch der Datenintegrationsprozess beschrieben. Orientierend an diesen Grundlagen und der vorgeschlagenen Referenzarchitektur aus Kapitel 9 sowie den Erkenntnissen aus realen Nagioskonfigurationen, soll in diesem Kapitel die Datenintegration von Nagios durchgeführt werden.

12.1 Vorüberlegungen

12.1.1 Ziel

3LGM²-Modelle sollen Nagiosdaten nutzen. Das heißt, es sollen 3LGM²-Modellelemente in Abhängigkeit von Nagiosdaten erstellt, gelöscht und geändert werden. Um dies zu erreichen, werden Nagiosobjekte und 3LGM²-Modellelemente einander zugeordnet. Die Zuordnungen verdeutlichen, dass die beteiligten Elemente bzw. Objekte ein und das selbe Objekt der realen Welt beschreiben. Dabei sind nur die Objekte der realen Welt interessant, die von Nagios besser und aktueller beschrieben werden als von einem bestimmten 3LGM²-Modell.

Zunächst soll festgelegt werden, was durch die Begriffe *Zuordnung/zuordnen* erreicht werden soll.

Festlegung Ist ein 3LGM²-Modellelement einem Nagiosobjekt *zugeordnet*, so muss das 3LGM²-Modellelement gelöscht oder die *Zuordnung* aufgehoben werden, wenn das *zugeordnete* Nagiosobjekt gelöscht wurde.

12.1.2 Informationsbedarf und Datenmodell

Der Informationsbedarf orientiert sich besonders an den im Kapitel 7 beschriebenen HSKs, die im Kapitel 10.5.3 durch die in Nagios verfügbaren Daten eingeschränkt wurden. Die globale Datenstruktur muss demnach diese restlichen ableitbaren 3LGM²-Metamodellelemente beschreiben können. Zum Informationsbedarf gehört auch die Darstellung der Zuordnung zwischen Nagiosobjekten und 3LGM²-Modellelementen. Diese wird zunächst als eine n:n-Beziehung zwischen Nagiosobjekten und 3LGM²-Modellelementen gesehen, da diese Art der Beziehung keine späteren Einschränkungen durch die globalen Datenstruktur mit sich bringt.

Als Datenmodell wird, wie in der Referenzarchitektur vorgeschlagen, UML verwendet. UML ist zudem eine gute Grundlage für die Umsetzung der Integrationskomponente mit der Programmiersprache Java.

12.1.3 Modellierung der lokalen Datenstruktur des 3LGM²-Baukastens

Die lokale Datenstruktur des 3LGM²-Baukastens ist zwischen altem und neuem 3LGM²-Baukasten unterschiedlich. Beiden gemein ist aber, dass sie auf Grundlage des 3LGM²-Metamodells entstanden sind. Im Folgenden wird das 3LGM²-Metamodell als lokale Datenstruktur des 3LGM²-Baukastens verwendet, da sich dieses einfach in die echten Datenstrukturen des alten und neuen 3LGM²-Baukastens übersetzen lässt. In Abbildung 12.1 ist ein UML-Klassendiagramm der lokalen Datenstruktur des 3LGM²-Baukastens dargestellt, die allerdings schon auf die HSKs reduziert ist.

12.1.4 Modellierung der lokalen Datenstruktur von Nagios

Aus der Diskussion der in Nagios vorkommenden Nagiosobjekttypen in Kapitel 10.5 lässt sich ein Teil der lokalen Datenstruktur von Nagios ablesen. Es handelt sich nur um einen Teil, da lediglich die für eine Datenintegration nützlichen Nagiosobjekte und Attribute diskutiert wurden.

In Abbildung 12.2 wurde ein UML-Klassendiagramm der Nagiosobjekttypen erstellt.

12.1.5 Aufbau der globalen Datenstruktur

Die globale Datenstruktur orientiert sich am Informationsbedarf des 3LGM²-Baukastens. Dabei werden alle Informationen benötigt, die genutzt werden können, um von Nagiosobjekten auf 3LGM²-Modellelemente zu schließen. Das beinhaltet eine vollständige Darstellung der relevanten Nagiosobjekte und 3LGM²-Modellelemente. Ein UML-Diagramm befindet sich in Abbildung 12.3. Dabei konnte die Darstellung von Nagiosobjekten vereinfacht werden, was sich im Prototypen bewährt hat. In der globalen Datenstruktur schlägt sich auch nieder, dass 3LGM²-Modellelemente ihre zugeordneten Nagiosobjekte kennen.

Durch den *Top-Down-Ansatz* und damit die Stützung auf den Informationsbedarf konnten für die Datenintegration unwichtige Klassen des 3LGM²-Metamodells und von Nagios weggelassen werden.

12.1.6 Erstellen des globalen Datenbestandes

Da die globale Datenstruktur aus Teilen der lokalen Datenstrukturen von 3LGM²-Baukasten und Nagios zusammengesetzt wurde, beschränkt sich das Erstellen des globalen Datenbestandes auf das Kopieren der für die Datenintegration nützlichen und in die globale Datenstruktur passenden Datenobjekte. Dabei müssen auf Seiten des 3LGM²-Baukastens die 3LGM²-Modellelemente ausgewählt werden, die integriert werden sollen. Das sind die 3LGM²-Modellelemente, die Instanzen der HSKs sind. Sie müssen an die globale Datenstruktur angepasst werden, ohne dabei ihre Werte zu verändern. Für Nagios müssen zunächst die Konfigurationsdateien interpretiert werden, um alle Nagiosobjekte zu kennen. Die für die Datenintegration wichtigen Nagiosobjekte werden dann ebenfalls in den globalen Datenbestand übernommen.

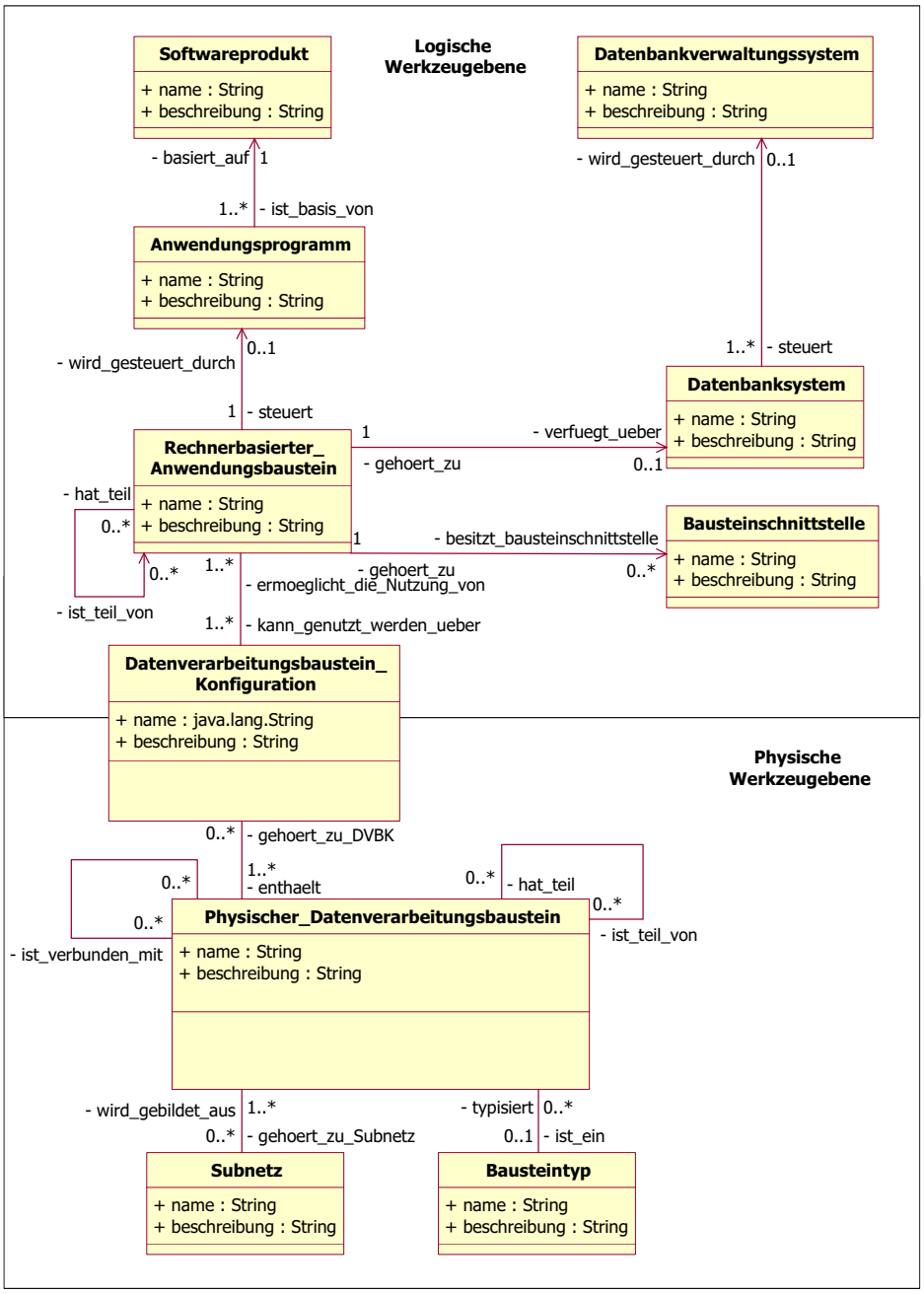

Abbildung 12.1: Die lokale Datenstruktur des 3LGM²-Baukastens, reduziert auf die HSKs.

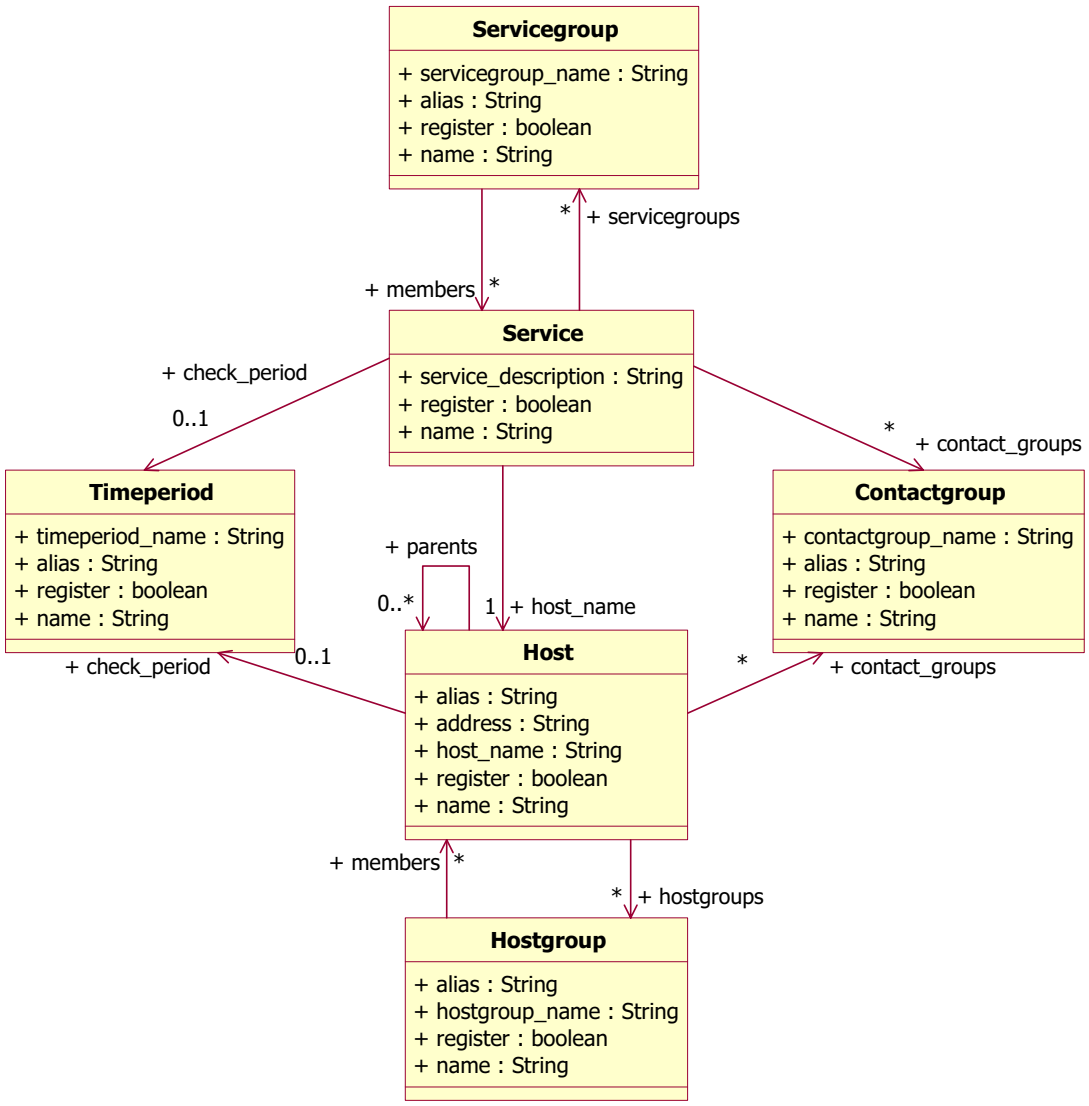

Abbildung 12.2: Das UML-Klassendiagramm der lokalen Datenstruktur von Nagios, abgeleitet aus den Nagiosobjekten

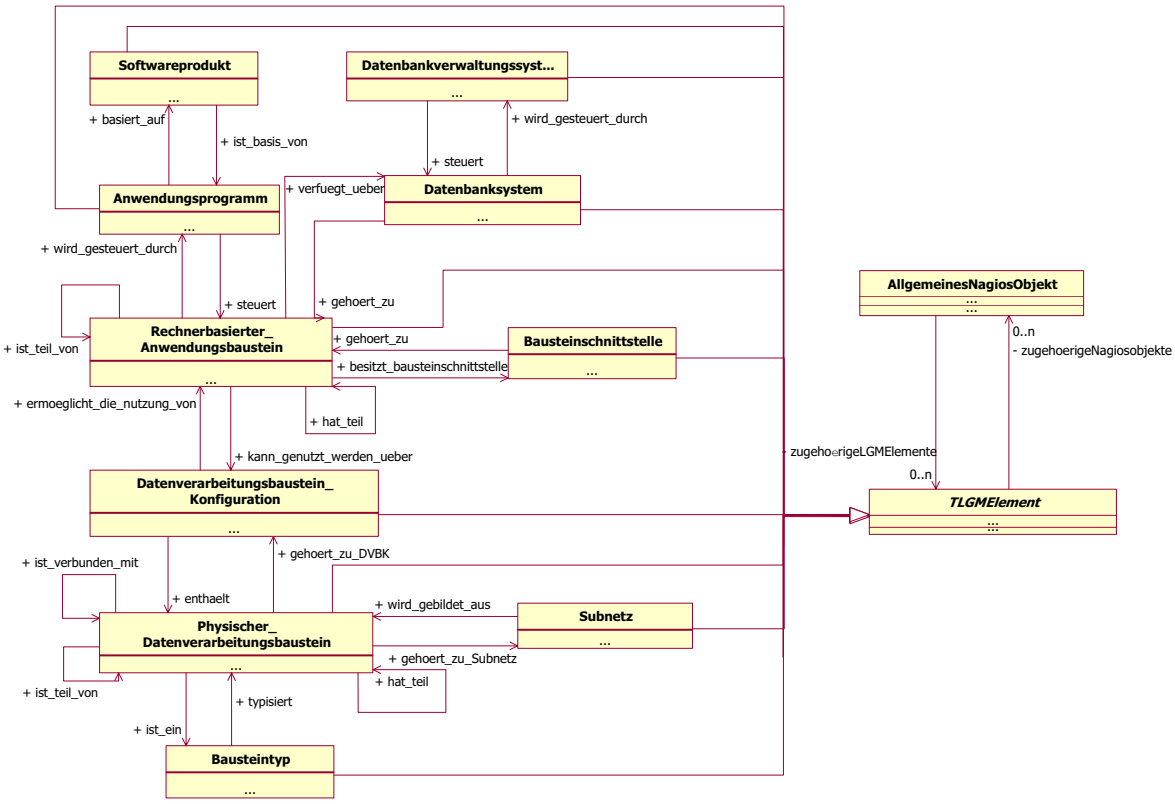

Abbildung 12.3: Die globale Datenstruktur für die Datenintegration. Sie ist aus dem zum Testen erstellten Prototypen entnommen.

12.1.7 Physische Integration

Eine Nagioskonfiguration ist eine Sammlung von Dateien, die sich im Allgemeinen auf dem Rechner befindet, auf dem auch der Nagiosdämon läuft. Diese Dateien sind relevant für die Sicherheit eines Informationssystems und daher normalerweise nur für Administratoren zugänglich. Eine Möglichkeit für die physische Integration wäre, diese Dateien Modellieren als Freigabe des entsprechenden Verzeichnisses zur Verfügung zu stellen. Für die hier beschriebene Umsetzung würde sogar ein lesender Zugriff ausreichend sein.

Das Anfordern der Konfigurationsdateien bei Bedarf von den Administratoren, z. B. mit Hilfe von E-Mails, ist auch möglich.

Eine weitere Variante ist das zur Verfügung stellen über eine eigens entwickelte Server-Komponente, die auf dem Rechner der Konfigurationsdateien installiert ist und direkt mit der Integrationskomponente kommuniziert. Diese letzte Lösung hätte den Vorteil, dass von der Server-Komponente sicherheitsrelevante Daten zurückgehalten werden könnten[1].

12.2 Module zum Finden von Zuordnungen

Es gibt verschiedene Möglichkeiten, zwischen Nagioskonfigurationen und 3LGM²-Modellen Zuordnungen zu finden und damit Abhängigkeiten zu beschreiben. Einige basieren auf der Semantik von Nagioskonfigurationen, andere basieren auf den Konventionen, die in den meisten Nagioskonfigurationen eingehalten werden. Leider kann ein realer Datensatz von Konventionen und auch von Vorgaben seitens Nagios abweichen. Um aber dennoch ein Maximum an Abhängigkeiten automatisch erkennen zu können, werden hier Module beschrieben, die für konkrete Nagiosdaten anwendbar sind oder nicht. Für jede Datenintegrationssituation lassen sich so Pakete von Modulen bilden, die angewendet werden können. Jedes Modul hat „Voraussetzungen der Nagioskonfiguration“ die von der Nagioskonfiguration erfüllt sein müssen, damit das Modul angewendet werden kann und „Vorausgegangene Module“, die vor der Anwendung eines Moduls bereits angewendet worden sein müssen. Zusätzlich wird, wenn möglich, der „Grad der Automatisierung“ angegeben.

Teilweise werden Struktogramme nach Isaac Nassi und Ben Shneiderman zur Verdeutlichung der Algorithmen verwendet.

12.2.1 Modul Rechnernetz

Voraussetzungen der Nagioskonfiguration Es wird ein Rechnernetz beschrieben.

Vorausgegangene Module keine

Grad der Automatisierung manuell

Inhalt Dieses Modul ermöglicht es dem Benutzer, Nagiosobjekttypen den Klassen des 3LGM²-Metamodells zuzuordnen. Eine tabellarische Aufstellung der Zuordnungen findet sich in Tabelle 12.1. Diese Zuordnungen sind aus semantischen Vorgaben für Nagioskonfigurationen abgeleitet worden.

[1] Die Konsequenzen für die Aussagekraft des 3LGM²-Modells müssen dabei berücksichtigt werden.

Nagiosobjekttyp	zugeordnete 3LGM²-Metamodellklasse
Host	Physischer Datenverarbeitungsbaustein, Subnetz, Bausteintyp
Hostgroup	Physischer Datenverarbeitungsbaustein, Subnetz, Bausteintyp
Service	Softwareprodukt, Anwendungsprogramm, Datenbankverwaltungssystem, Datenbanksystem, Bausteinschnittstelle, Rechnerbasierter Anwendungsbaustein
Servicegroup	Softwareprodukt, Anwendungsprogramm, Datenbankverwaltungssystem, Datenbanksystem, Bausteinschnittstelle, Rechnerbasierter Anwendungsbaustein
Contactgroup	
Timeperiod	

Tabelle 12.1: Zuordnung von Nagiosobjekttypen zu 3LGM²-Metamodellklassen

Bemerkung Postitiv an der Zuordnung ist, dass lediglich 2 Objekttypen keine Zuordnung besitzen (*Contactgroup* und *Timeperiod*). Negativ ist, dass die restlichen Nagiosobjekttypen leider eine Vielzahl von Zuordnungsmöglichkeiten haben, die ohne weitere Informationen über die Nagiosobjekte nicht einschränkbar sind. An dieser Stelle müsste der Benutzer für jedes Nagiosobjekt entscheiden, welcher Klasse des 3LGM²-Metamodells er dieses zuordnen möchte. Die *Datenverarbeitungsbaustein-Konfiguration* konnte keinem Nagiosobjekttyp zugeordnet werden. Sie nimmt eine Sonderstellung ein, die sie auch schon im 3LGM²-Baukasten besitzt. Dort wird sie automatisch angelegt, sobald ein *Rechnerbasierter Anwendungsbaustein* mit einem *Physischen Datenverarbeitungsbaustein* verbunden wird. (vgl. [STRÜBING 05] S.50) Dieser Konvention soll auch hier gefolgt werden.

Das Modul Rechnernetz erzeugt noch keine 3LGM²-Modellelemente, sondern legt lediglich die Grundlage für die Erstellung von Zuordnungen.

12.2.2 Modul Physischer Datenverarbeitungsbaustein

Voraussetzungen der Nagioskonfiguration Es handelt sich bei allen Hosts um Server, Switches oder sonstige Geräte, die zum Betrieb des Netzwerkes wichtig sind und nicht um CMCs oder andere zur reinen Überwachung gedachten Geräte.

Vorausgegangene Module Modul Rechnernetz

Grad der Automatisierung automatisch

Inhalt Dieses Modul erstellt für alle Hosts *Physische Datenverarbeitungsbausteine* und ordnet diese dem jeweiligen Host zu. Wenn es zu einem Host schon ein 3LGM²-Modellelement gibt, so wird kein neues angelegt.

Bemerkung Dass Hosts mit *Physischen Datenverarbeitungsbausteinen* identisch sind, ist leicht nachzuvollziehen, wenn man die Verwendung von Hostobjekten in Nagios betrachtet.

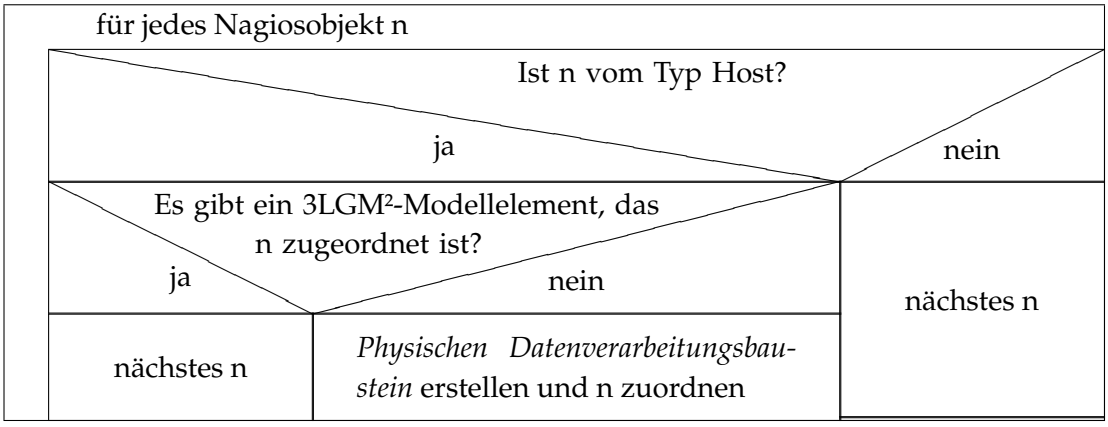

Abbildung 12.4: Das Nassi-Shneiderman Diagramm des Moduls Physischer Datenverarbeitungsbaustein

Dieses Modul widerspricht allerdings der Beobachtung, dass die *Physischen Datenverarbeitungsbausteine* bisher in 3LGM²-Modellen keine Einzelrechner darstellten. Andererseits hat die Kopplung mit Nagios das Ziel, 3LGM²-Modelle aussagekräftiger zu machen, was durch eine feinere Granularität des 3LGM²-Modells erreicht wird. Da die Verbesserung der Aussagekraft im Vordergrund steht, können die *Physischen Datenverarbeitungsbausteine* auch Einzelrechner sein. Der verwendete Algorithmus ist als Struktogramm in Abbildung 12.4 dargestellt.

12.2.3 Modul Hostgroup

Voraussetzungen der Nagioskonfiguration Hostgroups fassen mehrere Hosts zusammen, z. B. funktionell, räumlich, organisatorisch oder anderweitig.

Vorausgegangene Module Modul Rechnernetz

Grad der Automatisierung manuell

Inhalt Sind *Bausteintypen*, *Physische Datenverarbeitungsbausteine* und *Subnetze* Nagiosobjekten zugeordnet und diese Nagiosobjekte ihrerseits über eine Host ↔ Hostgroup-Relation verbunden, so können die in Tabelle 12.2 dargestellten 3LGM²-Relationen mit Hilfe des Benutzers abgeleitet werden.

Host	Hostgroup	Relation
Physischer DV-Baustein	*Physischer DV-Baustein*	ist_teil_von
Physischer DV-Baustein	*Subnetz*	gehoert_zu_Subnetz
Physischer DV-Baustein	*Bausteintyp*	ist_ein

Tabelle 12.2: Ableitungen der Relation Host ↔ Hostgroup

12.2.4 Modul Servicegroup

Voraussetzungen der Nagioskonfiguration Servicegroups fassen mehrere Services zusammen, z. B. funktionell, herstellerspezifisch, organisatorisch oder anderweitig.

Vorausgegangene Module Modul Rechnernetz

Grad der Automatisierung manuell

Inhalt Sind *Softwareprodukte*, *Rechnerbasierte Anwendungsbausteine* und *Datenbankverwaltungssysteme* Nagiosobjekten zugeordnet und diese Nagiosobjekte ihrerseits über eine Service ↔ Servicegroup-Relation verbunden, so können die in Tabelle 12.3 dargestellten 3LGM²-Relationen mit Hilfe des Benutzers abgeleitet werden.

Service	Servicegroup	Relation
Rechnerbasierter Anwendungsb.	*Rechnerbasierter Anwendungsb.*	ist_teil_von
Anwendungsprogramm	*Softwareprodukt*	basiert_auf
Datenbanksystem	*Datenbankverwaltungssystem*	wird_gesteuert_durch

Tabelle 12.3: Ableitungen der Relation Service ↔ Servicegroup

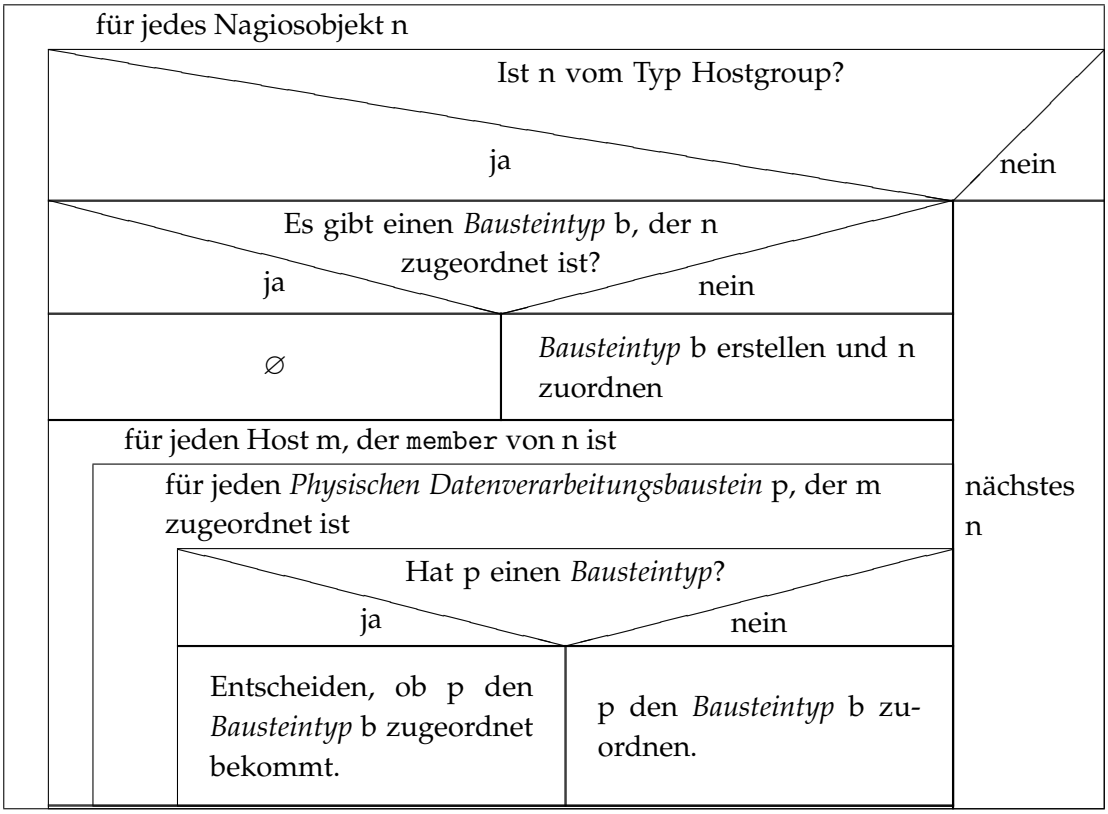

Abbildung 12.5: Das Nassi-Shneiderman Diagramm des Moduls Physischer Bausteintyp

12.2.5 Modul Bausteintyp

Voraussetzungen der Nagioskonfiguration Hostgroups typisieren Hosts. Ein typisches Beispiel ist eine Hostgroup namens „switches“.

Vorausgegangene Module Modul Rechnernetz, Modul Physischer Datenverarbeitungsbaustein, Modul Hostgroup

Grad der Automatisierung semiautomatisch

Inhalt Dieses Modul erstellt für jede Hostgoup einen *Bausteintyp* und ordnet diesen Bausteintyp der jeweiligen Hostgroup zu. Anschließend wird die Relation *ist_ein/typisiert* zwischen *Physischen Datenverarbeitungsbausteinen* und *Bausteintypen* gesetzt. Gibt es mehrere *Bausteintypen* zu einem *Physischen Datenverarbeitungsbaustein*, so kann natürlich nur eine zugeordnet werden. Die Entscheidung darüber sollte der Benutzer treffen. Wenn es zu einer Hostgroup schon einen *Bausteintyp* gibt, so wird kein neuer angelegt.

Bemerkung Diese Modul basiert auf Beobachtungen über die Nutzung von Hostgroups in realen Nagiosdaten. Der verwendete Algorithmus ist als Struktogramm in Abbildung 12.2.4 dargestellt.

12.2.6 Modul Contactgroup/Timeperiod

Voraussetzungen der Nagioskonfiguration Es gibt mehr als eine Contactgroup und eine Timeperiod. Außerdem sind die Bezeichnungen von Contactgroup und Timeperiod aussagekräftig.

Vorausgegangene Module Modul Rechnernetz

Grad der Automatisierung automatisch

Inhalt Contactgroup und Timeperiod enthalten sinnvolle Zusatzinformation, die in die Beschreibung von 3LGM²-Modellelementen einfließen können. Allerdings ist das Feld „Beschreibung“ dafür ungeeignet, da hier auch Einträge von Modellierern Platz finden müssen. Zwei eigene benutzerdefinierte Eigenschaftsfelder der Form „Überwachungszeiten“ und „verantwortlicher Personenkreis“ wären besser. Dabei werden *Contactgroup* und *Timeperiod* denjenigen 3LGM²-Modellelementen zugeordnet, die wiederum einem Nagiosobjekt zugeordnet sind, das eine Referenz auf die Contactgroup bzw. Timeperiod hat. Bei mehreren Contactgroups werden diese aufgelistet.

Bemerkung Der verwendete Algorithmus ist als Struktogramm in Abbildung 12.6 dargestellt.

12.2.7 Modul ist_verbunden_mit-Relation

Voraussetzungen der Nagioskonfiguration Wenn Hosts in der Konfiguration das `parents`-Attribut nutzen, dann im von Nagios vorgesehenen Sinne. Das heißt, ein Parent ist ein Rechner oder Switch, von dem die Erreichbarkeit des Hosts abhängig ist. Funktioniert ein Parent nicht mehr, so ist der Host über diesen Parent auch nicht mehr erreichbar.

Vorausgegangene Module Modul Physischer Datenverarbeitungsbaustein

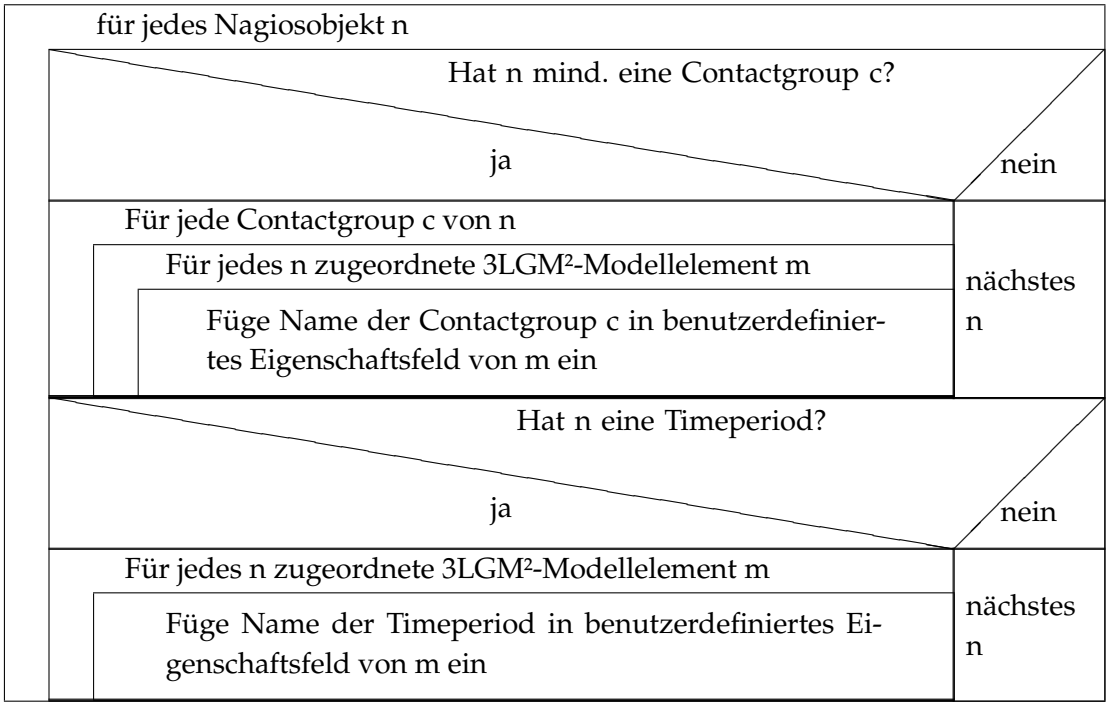

Abbildung 12.6: Das Nassi-Shneiderman Diagramm des Moduls Contactgroup/Timeperiod

Grad der Automatisierung automatisch

Inhalt Besteht eine Parent-Relation zwischen 2 Hosts und ist von jedem Host auch ein *Physischer Datenverarbeitungsbaustein* abhängig, so wird eine ist_verbunden_mit-Relation zwischen den beiden *Physischen Datenverarbeitungsbausteinen* angelegt. Besteht diese Verbindung bereits, so wird sie nicht noch einmal erstellt.

Bemerkung Der verwendete Algorithmus ist als Struktogramm in Abbildung 12.7 dargestellt.

12.2.8 Modul Anwendungsprogramm

Voraussetzungen der Nagioskonfiguration keine

Vorausgegangene Module evtl. Modul Servicegroups

Grad der Automatisierung automatisch

Inhalt Ist für einen Service bekannt, dass er selbst oder seine übergeordnete Servicegroup einem *Softwareprodukt* zugeordnet ist und ist auch bekannt, dass der Host, zu dem der Service gehört, einem *Physischen Datenverarbeitungsbaustein* zugeordnet ist, dann wird der Service zu einem *Anwendungsprogramm* abgeleitet und ein *Rechnerbasierter Anwendungsbaustein* sowie eine *Datenverarbeitungsbaustein-Konfiguration* erstellt. Das entstehende *Anwendungsprogramm* sowie der *Rechnerbasierte Anwendungsbaustein* und die *Datenverarbeitungsbaustein-Konfiguration* werden dabei dem Service zugeordnet. Die *Datenverarbeitungsbaustein-Konfiguration* wird zusätzlich dem Host zugeordnet, auf dem der Service läuft. Es werden anschließend alle Relationen erzeugt, so dass zwischen *Physischem*

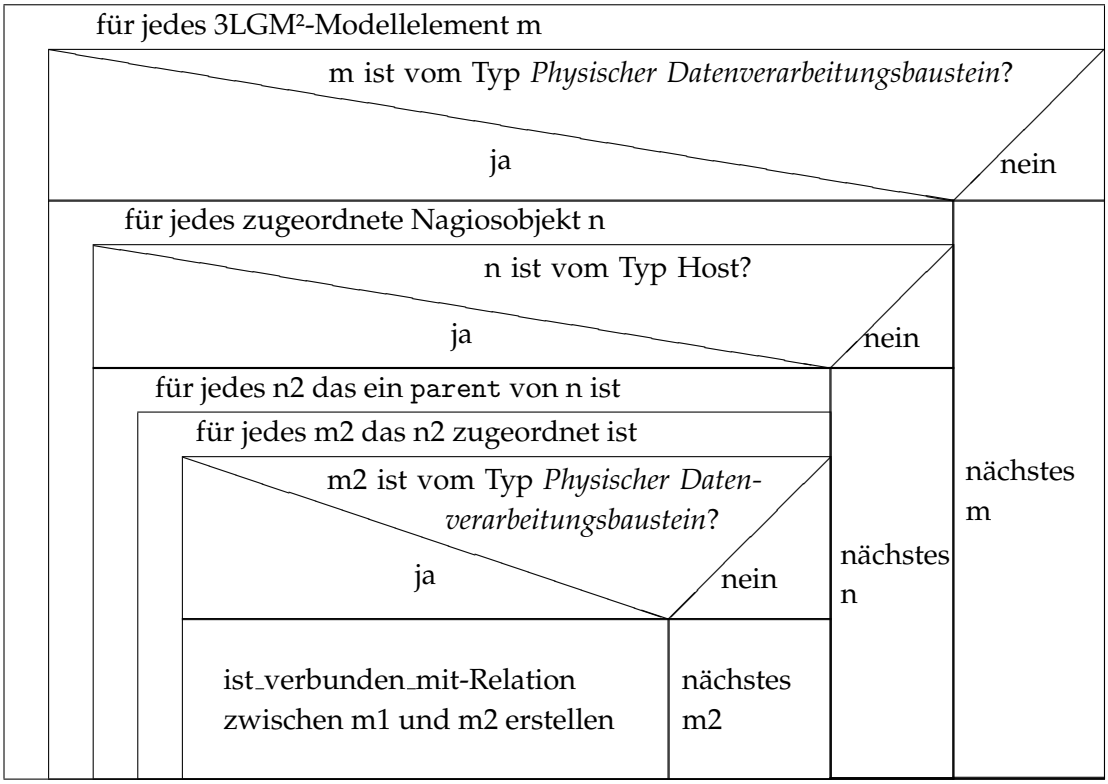

Abbildung 12.7: Das Nassi-Shneiderman Diagramm des Moduls ist_verbunden_mit-Relation

Datenverarbeitungsbaustein und *Softwareprodukt* ein Pfad entsteht. Besteht dieser Pfad bereits, so wird nichts neu angelegt.

Bemerkung In Verbindung mit den Modulen Datenbanksystem und Bausteinschnittstelle können für einen *Rechnerbasierten Anwendungsbaustein* mehrere, möglicherweise redundante, *Datenverarbeitungsbaustein-Konfigurationen* angelegt werden. Diese lassen sich im 3LGM²-Baukasten allerdings wieder vereinen.

Auf ein Nassi-Shneiderman Diagramm wird an dieser Stelle verzichtet, da es sehr unübersichtlich wird. Die Funktion, wurde allerdings im Prototypen realisiert.

12.2.9 Modul Datenbanksystem

Voraussetzungen der Nagioskonfiguration keine

Vorausgegangene Module evtl. Modul Servicegroups

Grad der Automatisierung automatisch

Inhalt Ist für einen Service bekannt, dass er selbst oder seine übergeordnete Servicegroup einem *Datenbankverwaltungssystem* zugeordnet ist und ist auch bekannt, dass der Host, zu dem der Service gehört, einem *Physischen Datenverarbeitungsbaustein* zugeordnet ist, dann wird der Service zu einem *Datenbanksystem* abgeleitet und ein *Rechnerbasierter Anwendungsbaustein* sowie eine *Datenverarbeitungsbaustein-Konfiguration* erstellt. Das entstehende *Datenbanksystem* sowie der *Rechnerbasierte Anwendungsbaustein* und

die *Datenverarbeitungsbaustein-Konfiguration* werden dabei dem Service zugeordnet. Die *Datenverarbeitungsbaustein-Konfiguration* wird zusätzlich dem Host zugeordnet, auf dem der Service läuft. Es werden anschließend alle Relationen erzeugt, so dass zwischen *Physischem Datenverarbeitungsbaustein* und *Datenbankverwaltungssystem* ein Pfad entsteht. Besteht dieser Pfad bereits, so wird nichts neu angelegt.

Bemerkung In Verbindung mit den Modulen Anwendungsprogramm und Bausteinschnittstelle können für einen *Rechnerbasierten Anwendungsbaustein* mehrere, möglicherweise redundante, *Datenverarbeitungsbaustein-Konfigurationen* angelegt werden. Diese lassen sich im 3LGM²-Baukasten allerdings wieder vereinen.

Auf ein Nassi-Shneiderman Diagramm wird auch an dieser Stelle verzichtet, da es sehr unübersichtlich wird. Die Funktion wurde allerdings im Prototypen realisiert.

12.2.10 Modul Bausteinschnittstelle

Voraussetzungen der Nagioskonfiguration Die Beschreibungen von Services (`service_description`) sind nur dann gleich, wenn es sich um den gleichen Test zur Überprüfung des Services handelt.

Vorausgegangene Module Modul Rechnernetz

Grad der Automatisierung automatisch

Inhalt Ist für einen Service bekannt, dass er einer *Bausteinschnittstelle* zugeordnet ist und ist auch bekannt, dass der Host, zu dem der Service gehört, einem *Physischen Datenverarbeitungsbaustein* zugeordnet ist, dann wird ein *Rechnerbasierter Anwendungsbaustein* sowie eine *Datenverarbeitungsbaustein-Konfiguration* erstellt. Der *Rechnerbasierte Anwendungsbaustein* und die *Datenverarbeitungsbaustein-Konfiguration* werden dabei dem Service zugeordnet. Die *Datenverarbeitungsbaustein-Konfiguration* wird zusätzlich dem Host zugeordnet, auf dem der Service läuft. Es werden anschließend alle Relationen erzeugt, so dass zwischen *Physischem Datenverarbeitungsbaustein* und *Bausteinschnittstelle* ein Pfad entsteht. Alle weiteren Services mit der gleichen Beschreibung werden auch zu *Bausteinschnittstellen* abgeleitet und erhalten einen *Rechnerbasierten Anwendungsbaustein* sowie eine *Datenverarbeitungsbaustein-Konfiguration*. Besteht der Pfad zwischen *Physischem Datenverarbeitungsbaustein* und *Bausteinschnittstelle* bereits, so wird nichts neu angelegt.

Bemerkung In Verbindung mit den Modulen Anwendungsprogramm und Datenbanksystem können für einen *Rechnerbasierten Anwendungsbaustein* mehrere, möglicherweise redundante, *Datenverarbeitungsbaustein-Konfigurationen* angelegt werden. Diese lassen sich im 3LGM²-Baukasten allerdings wieder vereinen.

Auf ein Nassi-Shneiderman Diagramm wird an dieser Stelle ebenfalls verzichtet, da es sehr unübersichtlich wird. Die Funktion wurde im Prototypen realisiert.

12.2.11 Modul Servicetemplate

Voraussetzungen der Nagioskonfiguration Servicetemplates werden verwendet wie Servicegroups im Modul Servicegroup.

Vorausgegangene Module Modul Rechnernetz

Inhalt Servicetemplates erhalten den selben Status wie Servicegroups in den Modulen Rechnernetz, Servicegroup, Anwendungsprogramm und Datenbanksystem.

12.2.12 Modul Hosttemplate

Voraussetzungen der Nagioskonfiguration Hosttemplates werden verwendet wie Hostgroups im Modul Hostgroup.

Vorausgegangene Module Modul Rechnernetz

Inhalt Hosttemplates erhalten den selben Status wie Hostgroups in den Modulen Rechnernetz, Hostgroup, ist_verbunden_mit-Relation und Bausteintyp.

12.2.13 Modul Multiservice

Voraussetzungen der Nagioskonfiguration Multidefinitionen von Services werden verwendet wie Servicegroups im Modul Servicegroup.

Vorausgegangene Module Modul Rechnernetz

Inhalt Multidefinitionen von Services erhalten den selben Status wie Servicegroups in den Modulen Rechnernetz, Servicegroup, Anwendungsprogramm und Datenbanksystem.

12.2.14 Modul 3LGM²-Modellelementfelder

Voraussetzungen der Nagioskonfiguration keine

Vorausgegangene Module Modul Rechnernetz

Grad der Automatisierung manuell

Inhalt Die Felder von 3LGM²-Modellelementen (z. B. Name, Beschreibung, benutzerdefinierte Eigenschaftsfelder) können mit dem Inhalt von Feldern der Nagiosobjekte gefüllt werden. Diese Inhalte werden bei einer Synchronisation auch mit aktualisiert.

Bemerkung Beispielsweise kann so die Beschreibung eines Hosts gleichzeitig die Beschreibung eines *Physischen Datenverarbeitungsbausteins* sein.

12.3 Ausblenden von Nagiosobjekten

Die für die einzelnen Module geforderten Voraussetzungen werden von den meisten Nagioskonfigurationen sicherlich teilweise, aber nicht vollständig erfüllt. Um für einen Teil der Nagioskonfigurationen die Anwendung der Module dennoch zu ermöglichen, ist es sinnvoll, dass die Nagiosobjekte, die die Voraussetzungen nicht erfüllen, ausgeblendet werden können. Die Module operieren demnach immer auf den sichtbaren Nagiosobjekten.

Das Ausblenden von Nagiosobjekten verbessert auch den Umgang mit großen Konfigurationen.

12.4 Filtern von Services

Die Dienste sind die größte Klasse von Nagiosobjekten in einem Nagiossystem. Oft hat jeder Host zumindest einen grundsätzlichen PING-Service und zusätzlich dazu die eigentlichen Dienste (FTP, HTTP, SMTP, ...). Viele dieser Dienste, wie der eben genannte PING-Service, liefern aber leider keine verwertbare Information für ein 3LGM²-Modell. Steht für den Modellierer fest, dass er einen bestimmten Dienst nie integrieren wird, dann sollte es möglich sein, dass er diesen Service ausblendet. Voraussetzung für das Anwenden von Filtern ist allerdings, dass die Beschreibung von Services (`service_description`) immer gleich ist, wenn es sich um den gleichen Test handelt.

12.5 Vermeiden von identischen 3LGM²-Modellelementen

Die Erkennung von identischen 3LGM²-Modellelementen ist eine sehr schwierige Aufgabe, da sich die Identität durch ihre Semantik ausdrückt und nicht durch ihre Bezeichnung. Ist jedoch die Bezeichnung zweier 3LGM²-Modellelemente gleich, so liegt die Vermutung nahe, dass es sich um identische 3LGM²-Modellelemente handelt. Günstig ist eine Duplikaterkennung, die den Benutzer schon beim Datenintegrationsprozess auf mögliche Duplikate hinweist.

Ein erster Ansatz basiert auf der Annahme, dass identische 3LGM²-Modellelemente ähnlich benannt sind. Mit einfachen Mitteln des Vergleichs von Zeichenketten (z. B. Levenshtein-Distanz oder anderen Verfahren des Data-Cleaning) können Ähnlichkeiten quantifiziert werden.

Sind Duplikate erkannt worden, kann die „Vereinigungsfunktion“ des 3LGM²-Baukastens genutzt werden. Dabei sollte darauf geachtet werden, dass das vereinigte 3LGM²-Modellelement die Zuordnung zu dem entsprechenden Nagiosobjekt behält, um bei einer Aktualisierung eine erneute Duplikaterkennung zu vermeiden.

12.6 Persistenz des globalen Datenbestandes und der Nagiosschlüssel

Die Referenzarchitektur schlägt vor, den globalen Datenbestand zu speichern und die Anwendungen darauf zugreifen zu lassen. Allerdings bleibt die Frage offen, ob lokale Datenbestände weiterverwendet werden. Die Verwendung des globalen Datenbestandes anstelle der lokalen Datenbestände ist für Nagios und den 3LGM²-Baukasten nicht möglich, da der globale Datenbestand für beide nur eine Teilmenge der lokalen Datenbestände enthält. Wenn der globale Datenbestand persistent ist und die lokalen Datenbestände weitergenutzt werden, so muss sichergestellt werden, dass die Objekte aus den lokalen Datenbeständen, die auch im globalen Datenbestand vorkommen, als ein und das selbe Objekt identifizierbar sind.

Allerdings kann man durch die Anwendung des in Kapitel 9.2 beschriebenen Sonderfalls auf die Persistenz des globalen Datenbestandes verzichten und den Mehrwert der Datenintegration in einem Nagiosschlüssel beschreiben. Der Nagiosschlüssel ist als Zeichenkette konzipiert und kann damit als „benutzerdefiniertes Eigenschaftsfeld“ in den Datenbestand des 3LGM²-

Baukastens einfließen. Auf der Seite von Nagios kann ebenfalls auf den globalen Datenbestand verzichtet werden, da im Moment noch keine Verwendung von Daten aus 3LGM²-Modellen für Nagios besteht.

Objektidentifikationen Die Beziehungen zwischen den 3LGM²-Modellelementen und den Nagiosobjekten, den Nagiosmultidefinitionen und den Nagiostemplates müssen nachvollziehbar bleiben, um eine Aktualisierung der Daten ohne vollständige neue Integration der Datenbestände zu ermöglichen. Dazu kann der Nagiosschlüssel dienen.

Objekte in Nagios haben einen Namen[2], der bei der Konfiguration zur Referenzierung der jeweiligen Objekte verwendet wird. Dieser Name kann geändert werden. Die Änderung muss aber dann auch an allen referenzierenden Stellen nachgeführt werden, was sehr aufwendig sein kann. Weiterhin ist davon auszugehen, dass die Änderung des Namens mit irgendeiner anderen tiefgreifenden Änderung des Objekts einhergeht (z. B. eine neue Verwendung). Daher soll der Name eines Nagiosobjekts zur Identifikation und Wiedererkennung dieses Objekts verwendet werden. Gegen die Möglichkeit, eine IP-Nummer als Identifikator zu verwenden, spricht, dass die Änderung allein der IP Nummer eines Objekts noch nicht darauf schließen lässt, dass sich auch beispielsweise funktionell etwas verändert hat.

Schwieriger ist die Identifizierung der von der Datenintegration bewusst ausgeschlossenen Elemente. Der 3LGM²-Baukasten bietet keine Möglichkeit, Elemente zu erstellen, die dann nicht im Modell vorkommen. Eine Möglichkeit, diese Information in den integrierten 3LGM²-Modellelementen zu speichern ist, wäre die ausgeschlossenen Nagiosobjekte geeignet mit in den Schlüsseln der aufgenommenen Nagiosobjekte unterzubringen. Dabei entsteht die Schwierigkeit, solche Daten sinnvoll zu verteilen.

Selbsterstellte Objektidentifikationen Die Möglichkeit, selbst Identifikationsnummern für Nagiosobjekte zu erstellen, ist für Nagios im Allgemeinen nicht möglich. Grund ist die bei größeren Netzen aufwendigere Konfiguration. Um diese beherrschbar zu machen, werden Werkzeuge eingesetzt, die die Konfigurationsdatei auslesen und dem Benutzer in „benutzerfreundlicherer“ Form präsentieren. Die mit diesen Werkzeugen getätigten Änderungen werden anschließend wieder in die Konfigurationsdateien ausgeschrieben. Es ist zwar möglich, Kommentare in die Konfigurationsdateien zu schreiben und damit Nagiosobjekte zu markieren, man kann aber leider nicht annehmen, dass diese Kommentare von den Konfigurationswerkzeugen beachtet werden und erhalten bleiben.

Syntax des Nagiosschlüssels Der Nagiosschlüssel soll als Zeichenkette realisiert werden. Dafür ist es sinnvoll, ein Trennzeichen zu definieren, das in Nagioskonfigurationsdaten sonst nicht vorkommt. Die Dokumentation wird diesbezüglich leider auch nicht konkret. Es gibt allerdings zwei Zeichen, die auch in Nagios Sonderzeichen sind und somit niemals in Namen vorkommen. Es handelt sich um „*“ und „,“. Das Komma wird als besonderes Trennzeichen bei der Identifikation von Services verwendet, um `host_name` und `service_description` voneinander zu trennen. Ein Problem, das bisher noch nicht bearbeitet wurde, ist die Identifikation

[2] Eine Ausnahme stellen Services dar. Sie werden über eine Kombination von `host_name` und `service_description` (durch Komma getrennt) referenziert.

von Nagiosmultidefinitionen. Ihr identifizierender Name besteht meist aus einem regulären Ausdruck, der spätestens im Nagiosschlüssel die Syntax gefährdet. Ein einfacher Ansatz, um dieses Problem zu lösen, ist das Ersetzen des regulären Ausdrucks durch seine hexadezimale Repräsentation im ASCII-Format.

Die Syntax des Nagiosschlüssels wird durch die Grammatik G festgeschrieben. Die Grammatik G besteht dabei aus einer endlichen Menge N von Nichtterminalsymbolen, dem Alphabet $\sum$ bei dem es sich um eine endliche Menge von Terminalsymbolen handelt, einer endlichen Menge von Produktionsregeln P und einem Startsymbol S. ϵ ist das leere Wort.

$N :=$ {SCHLÜSSEL, EINTRAG, OBJEKT, VARIABLE, RELATION, ABHÄNGIGKEIT, NAGIOS-ID, VARIABLE-NAGIOS, RELATION-NAGIOS, VARIABLE-METAMODELL, RELATION-METAMODELL, NAGIOS-TYP, NAGIOS-NAME}

$\sum :=$ alle in Nagios verwendeten Zeichen inklusive „*“

$S :=$SCHLÜSSEL

$P :=$ {
SCHLÜSSEL$\rightarrow \epsilon$
SCHLÜSSEL$\rightarrow$EINTRAG
EINTRAG$\rightarrow$EINTRAG*EINTRAG

EINTRAG$\rightarrow$OBJEKT
EINTRAG$\rightarrow$VARIABLE
EINTRAG$\rightarrow$RELATION

OBJEKT$\rightarrow$NAGIOS-ID*null*null*ABHÄNGIGKEIT
VARIABLE$\rightarrow$NAGIOS-ID*VARIABLE-NAGIOS*VARIABLE-METAMODELL*ABHÄNGIGKEIT
RELATION$\rightarrow$NAGIOS-ID*RELATION-NAGIOS*RELATION-METAMODELL*ABHÄNGIGKEIT

NAGIOS-ID$\rightarrow$NAGIOS-TYP*NAGIOS-NAME
ABHÄNGIGKEIT-ID$\rightarrow$integriert
ABHÄNGIGKEIT-ID$\rightarrow$nicht_integriert }

Die Nichtterminalsymbole VARIABLE-NAGIOS, RELATION-NAGIOS, VARIABLE-METAMODELL, RELATION-METAMODELL, NAGIOS-TYP und NAGIOS-NAME werden durch das entsprechende Objekt realisiert. Bei einem Nagiostyp „Nagiosmultitemplate“ ist zu beachten, dass der NAGIOS-NAME mit ASCII-Nummern umschrieben ist, z. B. wird aus „*“ die 42

Mit der oben beschriebenen Syntax hat der Nagiosschlüssel die folgenden Funktionen:

- Zu einem 3LGM²-Modellelement werden 0 bis n Nagiosobjekte zugeordnet.
- Zu einem 3LGM²-Modellelement werden bestimmte Relationen eines bestimmten Nagiosobjekts zu bestimmten Assoziationen dieses 3LGM²-Modellelements zugeordnet.

- Zu einem 3LGM²-Modellelement werden bestimmte Relationen eines bestimmten Nagiosobjekts zu bestimmten Assoziationen dieses 3LGM²-Modellelements nicht zugeordnet.
- Nicht integrierte Nagiosobjekte können aufgelistet werden. (Diese werden dabei auf die vorhandenen Nagiosschlüssel verteilt)

Der letzte der oben aufgeführten Punkte ist nicht möglich, wenn kein 3LGM²-Modellelement ein zugeordnetes Nagiosobjekt hat, da dann auch kein Nagiosschlüssel existiert. Dieser „pathologische“ Fall kommt aber einer nicht durchgeführten Datenintegration gleich, weswegen dies kein Problem darstellt.

Diese Funktionalität des Nagiosschlüssels ermöglicht es, bei einer Synchronisation festzustellen, welche Nagiosobjekte und Nagiosrelationen neu hinzugekommen und welche weggefallen sind.

12.7 Synchronisation

Die Strukturen eines Netzwerkes und damit die Strukturen, die Nagios überwacht, unterliegen im Laufe der Zeit einer steten Veränderung. Der 3LGM²-Baukasten muss dieser Veränderung Rechnung tragen und eine Aktualisierung der Datenintegration ermöglichen. Zu diesem Zweck wurden Nagiosschlüssel eingeführt, die es ermöglichen, einmal zugeordnete Nagiosobjekte bei einer Synchronisation wiederzufinden und Veränderungen festzustellen. Eine erstmalige Datenintegration ist ein Sonderfall der Synchronisation, bei dem alle Nagiosobjekte integriert werden müssen. Es können für ein 3LGM²-Modell auch mehrere Nagioskonfigurationen zur Synchronisation genutzt werden.

Eine Synchronisation könnte nach dem in Abbildung 12.8 dargestellten Struktogramm ablaufen. Dabei werden „veraltete“ 3LGM²-Modellelemente und „neue“ Nagiosobjekte gefunden.

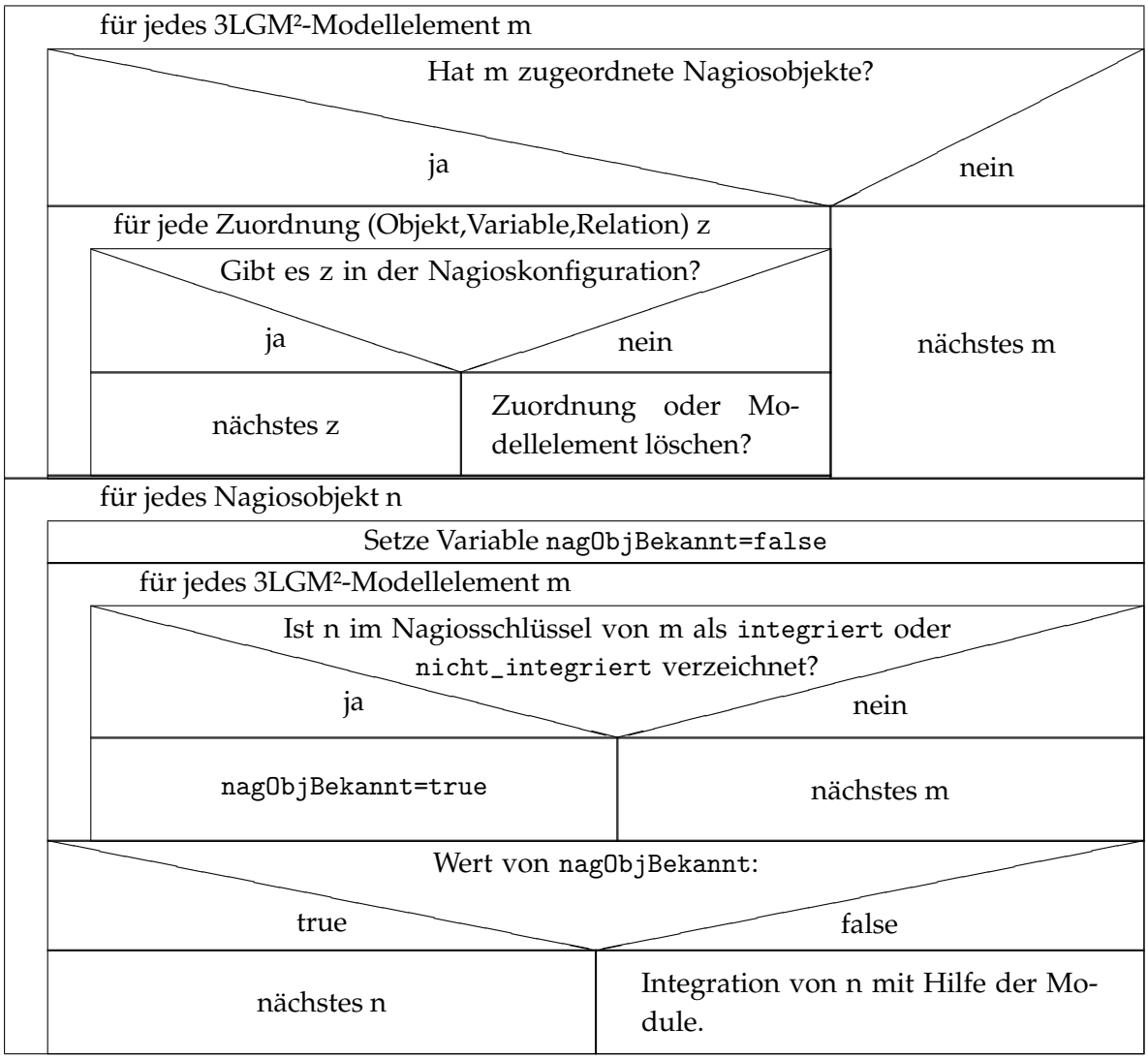

Abbildung 12.8: Das Nassi-Shneiderman Diagramm der Synchronisation

12.8 Implementierung

Da Nagioskonfigurationen keine Dokumente sind, die sich durch reines Lesen erschließen, musste ein Werkzeug entwickelt werden, das Nagioskonfigurationen auswertet und visualisiert. Weiterhin mussten die entwickelten Methoden, um 3LGM²-Modelle durch Nagioskonfigurationen zu verbessern, auch praktisch validiert werden, wozu dieses Werkzeug ebenfalls benötigt wurde. Das anfänglich gesetzte Ziel (Z1.4), eine vollständige Schnittstelle zu implementieren, war im Zuge dieser Arbeit nicht mehr möglich.

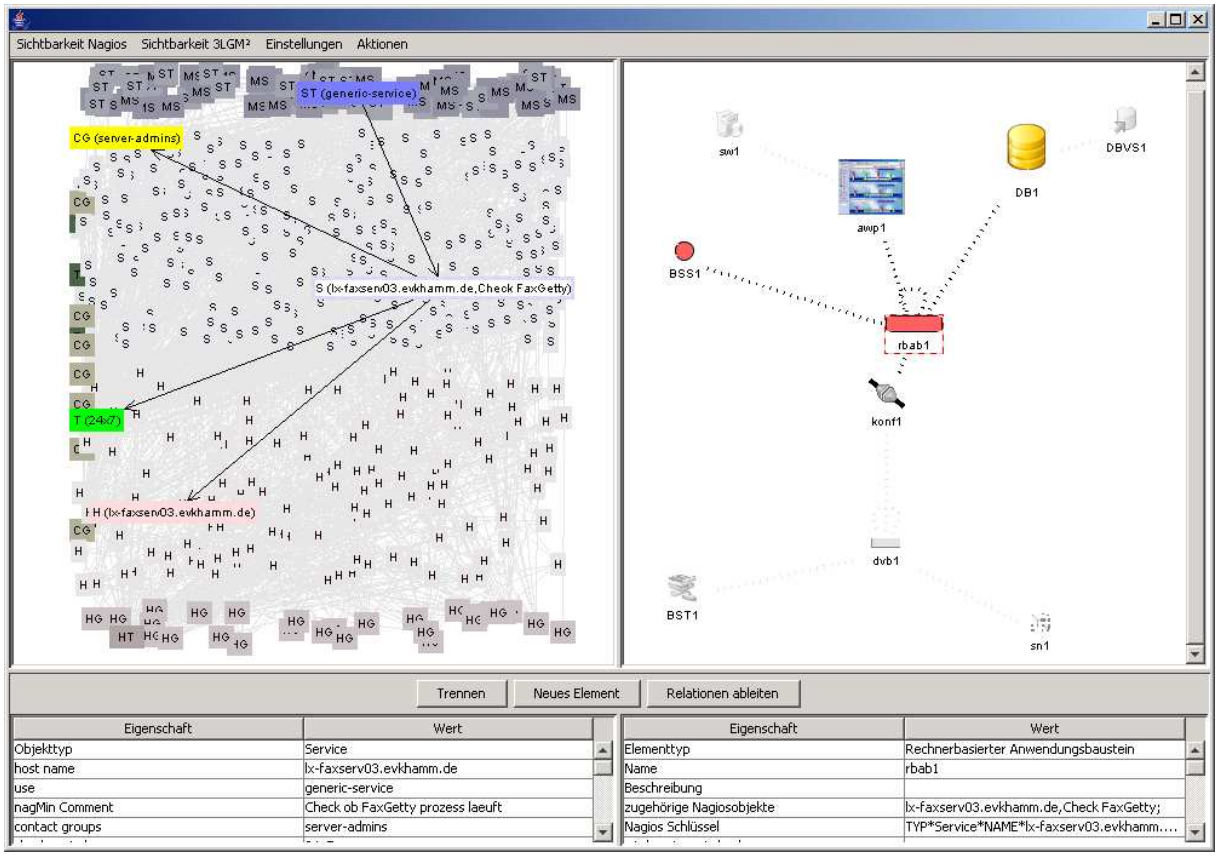

Abbildung 12.9: Ein Screenshot des Prototypen der Schnittstelle

Der als Werkzeug dienende Prototyp ist vollständig in Java entwickelt und stellt, wie in Abbildung 12.9 zu sehen, zwei Graphen dar. Links ist der Graph der Nagioskonfiguration zu sehen und rechts ein Teilgraph des 3LGM²-Modells[3]. Im unteren Teil der Ansicht sind Listen mit den verschiedenen Attributen der ausgewählten Elemente zu sehen, sowohl auf der Nagiosseite als auch auf der 3LGM²-Seite. Mit dem Knopf „Trennen“ kann eine bestehende Zuordnung entfernt werden. Sind zwei markierte Elemente nicht zugeordnet, so ermöglicht der selbe Knopf das Zuordnen dieser Elemente und heißt dann auch „Zuordnen“. Anstelle einer Zuordnung zu einem bestehenden 3LGM²-Modellelement kann mit dem Knopf „Neues Element“ ein neues 3LGM²-Modellelement erstellt und zugeordnet werden. Der Knopf „Relation

[3] 3LGM²-Modellelemente werden nicht wie im Baukasten dargestellt, sondern als Icons und auf die Darstellung der Ebenen wurde ganz verzichtet.

ableiten“ ermöglicht es zugeordneten 3LGM²-Modellelementen, die ableitbaren Relationen in das 3LGM²-Modell zu übernehmen.

Im Prototypen ist es möglich, verschiedene Elemente sowohl aus dem 3LGM²-Modell als auch aus dem Nagiosgraph auszublenden. Diese Filter arbeiten entweder für Gruppen von Elementen (z. B. alle Services) oder für einzelne Elemente. Diese Funktion hat insbesondere den Nutzen, weniger nützliche Nagiosobjekte ausschließen zu können.

12.9 Umsetzung der Anforderungen

12.9.1 Funktionelle Anforderungen

Konsistenzbedingungen für HSKs Die Konsistenzbedingungen werden durch die Module eingehalten. Dort, wo sie nicht explizit durch Module gewahrt bleiben, muss die Implementierung die Konsistenzbedingungen sichern.

Einbindung vorhandener 3LGM²-Modellelemente Das Einbinden vorhandener Elemente geht nur auf manuellem Wege. Nach Anwendung der Module Rechnernetz und bei Bedarf der Module Hostgroup sowie Servicegroup können solche manuellen Zuordnungen stattfinden.

Transportierbarkeit Die Transportierbarkeit ist gegeben, da kein globaler Datenbestand existiert, der mittransportiert werden müsste.

Unabhängigkeit des Datenintegrationsprozesses vom 3LGM²-Baukasten Alle Informationen über die Datenintegration sind im Nagiosschlüssel enthalten, der wiederum zu 3LGM²-Modellelementen gehört, die im 3LGM²-Modell enthalten sind. Keine Information ist an eine konkrete 3LGM²-Baukasteninstallation gebunden.

12.9.2 Nicht funktionelle Anforderungen

Aufwand Wenn automatische Module verwendet werden, so kann man sicherlich von einem verringerten Aufwand sprechen. Können nur manuelle Module verwendet werden, so ist der Aufwand ähnlich dem, der betrieben werden müsste, um das Netzwerk ohne Datenintegration im 3LGM²-Modell zu modellieren. Dann allerdings mit dem Vorteil, dass man nicht überlegen muss, aus welchen Komponenten das Netzwerk besteht, da dies bekannt ist. Kann keines der Module verwendet werden, dann bietet die Datenintegration keine Unterstützung. In diesem Fall besteht ein zu großes Problem der „Different Perspectives“, das im Kapitel 4.6.1.3 angesprochen wurde. Dann sollte aber auch darüber nachgedacht werden, ob Nagios „sinngemäß“ verwendet wurde.

Modellierstile Modellierstile, wie sie in 3LGM²-Modellen zu beobachten sind, werden nicht eingehalten. Man kann hier argumentieren, dass 3LGM²-Modelle, die teilweise durch Datenin-

tegration erzeugt wurden, sich durch ihre feinere Granularität auch von bisherigen Stilen auf der Physischen Werkzeugebene unterscheiden dürfen.

Datenqualität Die Datenqualität hinsichtlich der Aktualität von Daten ist in Nagios sehr gut, da alle in den Konfigurationsdateien beschriebenen Nagiosobjekte auch für die Überwachung genutzt werden.

Datenmenge Die Datenmenge kann durch das Wählen von Filtern oder rein manuelle Zuordnung und Erstellung von 3LGM²-Modellelementen begrenzt werden.

Visualisierung Es wurden keine neuen Visualisierungskonzepte für den 3LGM²-Baukasten entwickelt. Für die Visualisierung der Nagioskonfigurationen ist die Integrationskomponente zuständig, die entsprechende Konzepte verwirklichen kann. Der Prototyp mach dahingehend einen Vorschlag (vergleiche Abbildung 11.1).

Automatisierung Um ein Maximum an Automatisierung zu erreichen, wurden die Module eingeführt, die bei entsprechenden Daten eine automatische Datenintegration ermöglichen.

12.10 Erweiterung der Umsetzung

Die vorliegende Umsetzung der Datenintegration für 3LGM²-Modelle und Nagios ließe sich noch weiter ausbauen. Allerdings wurde dies in diesem Kapitel nicht durchgeführt, da sich der Nutzen für 3LGM²-Modelle in Grenzen hält und die Komplexität der Umsetzung überschaubar gehalten werden sollte.

Insbesondere bei den Contactgroups ließen sich die enthaltenen Contacts als menschliche *Physische Datenverarbeitungsbausteine* interpretieren und entsprechend zuordnen. Jedoch stellen die Administratoren, die als Contacts beschrieben werden, nur einen (kleinen) Teil aller menschlichen Handlungsträger dar. Werden sie mit integriert, entsteht ein Ungleichgewicht zwischen der Beschreibungsqualität der Administratoren und der sonstiger Handlungsträger eines Informationssystems. Die Umsetzung, wie sie in diesem Kapitel gezeigt wurde, versucht, den Bezug zu den Gruppen der Administratoren (Contactgroups) herzustellen ohne dabei ein Ungleichgewicht der Beschreibung zu erzeugen.

12.11 Zusammenfassung

In diesem Kapitel wurde gezeigt, dass es viele Möglichkeiten gibt, Daten aus Nagios für 3LGM²-Modelle zu nutzen. Leider zeigte sich aber auch, dass diese Datenintegration nicht mit jeder Nagioskonfiguration zufriedenstellend funktioniert, weswegen eine Modularisierung vorgeschlagen wurde. Der Nutzen dieser Datenintegration, auch auf der Seite von Nagios, wird Gegenstand des nächsten Kapitels sein.

13 Nutzen der 3LGM²-Datenintegration

Inhaltsangabe

13.1 Nutzen von Nagioskonfigurationen für 3LGM²-Modelle 109
13.1.1 Feinere Granularität 109
13.1.2 Bessere Aktualität 110
13.1.3 Verminderter Arbeitsaufwand 110
13.1.4 Reflexion in Nagios 110
13.2 Nutzen von 3LGM²-Modellen für Nagios 110
13.2.1 Konfigurationsunterstützung 110
13.2.2 Reflexion in 3LGM²-Modelle 111
13.3 „Best Practice“ für Nagios 111
13.3.1 Objekte 111
13.3.2 Hierarchien 112
13.4 Zusammenfassung 112

Ganz gleich wie gut Datenintegration funktioniert, ohne einen realen Nutzen bleibt sie sinnlos. In diesem Kapitel soll daher überprüft werden, inwieweit die in dem vergangenen Kapitel realisierte Datenintegration den Nutzern und Modellierern von 3LGM²-Modellen und Nagios einen Vorteil verschafft. Einige der hier aufgeführten Themen ergeben sich aus der Erfüllung der Anforderungen, die in Kapitel 8 beschrieben sind. Sie wurden daher in anderen Zusammenhängen schon erwähnt, sollen hier aber nochmals in Hinblick auf ihren Nutzen betrachtet werden.

13.1 Nutzen von Nagioskonfigurationen für 3LGM²-Modelle

3LGM²-Modelle werden genutzt, um das taktische und strategische Informationsmanagement zu unterstützen, weswegen die Verbesserung von 3LGM²-Modellen eine Verbesserung für das Informationsmanagement darstellt.

13.1.1 Feinere Granularität

Die Granularität, mit der in Nagios Netzwerke beschrieben werden, ist typischerweise deutlich feiner als die, mit der Netzwerke in 3LGM²-Modellen beschrieben sind. Nach einer Datenintegration können in 3LGM²-Modellen Rechnernetze ähnlich genau modelliert sein wie in Nagios.

13.1.2 Bessere Aktualität

Da bei einer Synchronisation mit einer Nagioskonfiguration Veränderungen auffallen, können in 3LGM²-Modellen diese Änderungen schnell nachgeführt werden, während sie ohne die Synchronisation erst viel später oder nie aufgefallen wären.

13.1.3 Verminderter Arbeitsaufwand

Einige Module der Datenintegration mit Nagios erstellen automatisch 3LGM²-Modellelemente und Relationen zwischen diesen Elementen. Diese Elemente müssen dann nicht mehr vom Benutzer erstellt werden, was zur Verringerung des Arbeitsaufwandes führt. Der verminderte Arbeitsaufwand ist auch von Interesse für Anwender, die den 3LGM²-Baukasten testen möchten. Sie können über die Datenintegration schnell zu verwendbaren Modellen mit realen Daten ihres Informationssystems kommen.

13.1.4 Reflexion in Nagios

Es ist relativ einfach, in der Nagiosbenutzerschnittstelle (also im Webbrowser) gezielt Nagiosobjekte über einen bestimmten Uniform Resource Locator (URL) aufzurufen. Es ist möglich, im 3LGM²-Baukasten URLs von Nagiosobjekten für 3LGM²-Modellelemente zu hinterlegen, somit können diese aus dem 3LGM²-Baukasten heraus in Nagios aufgerufen werden.

Hierbei handelt es sich genau genommen um funktionelle Integration, die aber als Voraussetzung einen URL benötigt, der wiederum durch die Datenintegration verfügbar wird.

13.2 Nutzen von 3LGM²-Modellen für Nagios

Nagios ist ein Werkzeug des operativen Informationsmanagement, weswegen die hier erreichten Vorteile im Gegensatz zu dem letzten Kapitel Vorteile für das operativen Informationsmanagement sind.

13.2.1 Konfigurationsunterstützung

Betrachtet man den Datenintegrationsgedanken nicht so einseitig wie in den vergangenen Kapiteln, sondern beidseitig, dann stellt sich die Frage, welche Daten Nagios aus 3LGM²-Modellen nutzen könnte. Hinsichtlich der momentanen Verwendung sind 3LGM²-Modelle viel gröber und deutlich umfangreicher. Teilweise beschreiben 3LGM²-Modelle Teile eines Informationssystems, die für Nagios ohne Bedeutung sind, wie beispielsweise *konventionelle Anwendungsbausteine*. Andererseits enthält ein 3LGM²-Modell viele 3LGM²-Modellelemente, die auch in einer Nagioskonfiguration eine Rolle spielen. Das sind genau diejenigen, auf denen die letzten Kapitel aufbauten. Contacts, Commands, Service Escalations, Service Dependencies, Host Escalations, Host Dependencies, Extended Host Information und Extended Service Information sind allerdings Nagiosobjekte, die keine Entsprechung im 3LGM²-Baukasten finden.

Die Konfiguration von Nagios ist zunächst sehr benutzerunfreundlich. Schnell verliert man die Übersicht über die Definitionen und die daraus resultierenden Objekte. Verschiedene „Configuration Tools“ für Nagios versuchen, dem Benutzer eine bessere Sicht auf die Konfigurationsdaten zu ermöglichen.

In der Reihe dieser Werkzeuge wäre der 3LGM²-Baukasten eines der interessantesten, da viele lediglich Eingabemasken für die Konfiguration der einzelnen Objekte bereitstellen und der 3LGM²-Baukasten eine grafische Ansicht in Zusammenhang mit dem gesamten IS bietet. Um eine Konfiguration in vollem Umfang zu ermöglichen, müssten alle weiter oben genannten Nagiosobjekte im 3LGM²-Baukasten in irgendeiner Art ersichtlich werden. Ob es möglich ist, allein durch benutzerdefinierte Eigenschaftsfelder alle Nagiosobjekte abzubilden, oder ob es unbedingt nötig ist, neue 3LGM²-Metamodellklassen einzuführen, kann hier nicht beantwortet werden. Vermutlich ist aber die reine Datenintegration über benutzerdefinierte Eigenschaftsfelder ohne besondere Funktionalitäten auf 3LGM²-Baukastenseiten für einen Benutzer sehr unübersichtlich.

Auf 3LGM²-Modelle hätte die gleichzeitige Nutzung des 3LGM²-Baukastens als „Configuration Tool“ den Nebeneffekt, dass 3LGM²-Modelle deutlich feiner und umfassender auf der Physischen Werkzeugebene werden müssten als in Kapitel 7.2 gesehen.

13.2.2 Reflexion in 3LGM²-Modelle

Der Nagiosschlüssel ermöglicht nicht nur, die zu einem 3LGM²-Modellelement zugeordneten Nagiosobjekte zu identifizieren, sondern auch, die einem Nagiosobjekt zugeordneten 3LGM²-Modellelemente. Dies kann genutzt werden, um beispielsweise bei einem Rechnerausfall die entstehenden Beeinträchtigungen für das IS nicht nur in Nagios einzuschätzen, sondern auch im 3LGM²-Baukasten. Eine Funktion, den 3LGM²-Baukasten zu starten und sofort ein bestimmtes 3LGM²-Modellelement auszuwählen, soll im neuen 3LGM²-Baukasten verfügbar sein.

Auch hier handelt es sich um funktionelle Integration.

13.3 „Best Practice“ für Nagios

Im folgden Abschnitt sollen Ideen vorgestellt werden, wie eine Nagioskonfiguration aussehen sollte, um eine bestmögliche Datenintegration mit dem 3LGM²-Baukasten zu gewährleisten.

13.3.1 Objekte

Um Nagiosobjekte, Nagiosmultidefinitionen und Nagiostemplates in einem 3LGM²-Modell richtig zu integrieren, muss der Modellierer eine Vorstellung davon haben, um was es sich handelt. Diese Vorstellung gewinnt er insbesondere über die Beschreibung der Nagiosobjekte, Nagiosmultidefinitionen und Nagiostemplates. Da Nagiosdaten hier über den Personenkreis der Nagiosadministratoren hinaus verständlich sein müssen, sollten die Beschreibungen ausführlicher gestaltet werden als bisher. Bei der Vergabe der Namen für Objekte wird häufig

der auch als Rechnername verwendete Name genutzt. Die Vergabe von Namen kann daher kaum beeinflusst werden, da sonst die Zuordnung von realem Rechner zum Nagiosobjekt für die Administratoren erschwert werden würde.

13.3.2 Hierarchien

Eine Beobachtung während der Arbeit war, dass die Strukturen, die durch externe Relationen entstehen können, ungenutzt bleiben. In den meisten Fällen werden nur zwei Nagiostemplates verwendet, die bereits in der zu Nagios mitgelieferten Beispielkonfiguration vorhanden sind. Die Strukturierung durch Host- bzw. Servicegroups ist dagegen recht ausgereift, was darauf zurückzuführen ist, dass diese Daten auch zur Strukturierung der Ansichten der Benutzerschnittstelle verwendet werden. Eine zusätzliche Strukturierung durch Nagiostemplates ist aber sinnvoll, wenn „Gruppen“ gebildet werden sollen, die für Administratoren unsichtbar bleiben, aber im 3LGM²-Modell gebraucht werden. Günstig an der Nutzung von Templates ist, dass sie, solange sie keine Objekte definieren[1], keinerlei Attribute aufweisen müssen und dadurch das Verhalten von Nagios nicht beeinflussen. Ein Nagiostemplate, das nur einen Namen und eine Beschreibung trägt, ist gut zur zusätzlichen Gruppierung geeignet. Nagiostemplates können leider im Gegensatz zu Host- bzw. Servicegroups nur disjunkte Gruppen bilden.

Die Abbildung von Topologien durch das `parents`-Attribut sollte dort, wo sie noch nicht durchgeführt wurde, nachgeholt werden. Dabei sollte dieses Attribut aber auch wirklich nur zur Abbildung der Topologie benutzt werden und nicht zur Abbildung von anderen Abhängigkeiten, wie es leider teilweise empfohlen wird.

13.4 Zusammenfassung

Dass es einen Nutzen der hier vorgestellten Datenintegration sowohl für Nutzer des 3LGM²-Baukastens als auch für Nutzer von Nagios gibt, zeigte dieses letzte Kapitel. Es zeigte aber auch, dass der Nutzen durch eine angepasste Erstellung der Nagioskonfigurationen verbessert werden kann.

[1] d.h. `register 0`

Z Abschluss

Inhaltsangabe

Z.1 Diskussion **113**
Z.2 Zielerfüllung **113**
Z.3 Ausblick **116**

Z.1 Diskussion

Diese Arbeit zielte darauf ab, dem Modellierer eine Menge von Möglichkeiten darzulegen, Daten über Hard- und Softwarebestände für 3LGM²-Modelle zu nutzen und dies für eine Datenquelle zu realisieren. Leider hat sich gezeigt, dass es zwar viele Ansatzpunkte gibt, aber keine verlässlichen Informationen über Güte und Umfang solcher Daten. Selbst für das konkrete Produkt Nagios ließen sich keine allgemeingültigen Aussagen für die Datenqualität treffen. Um diese Individualisierung von Datenintegration beherrschen zu können, wurde daraufhin versucht, diese Arbeit so weit wie möglich für andere Datenintegrationsszenarien adaptierbar zu gestalten. Dafür wurde eine Referenzarchitektur für Datenintegration beschrieben und die Module für Nagios entwickelt, die es ermöglichen, flexibel auf verschiedene Qualitäten von Nagioskonfigurationen zu reagieren.

Es wurde deutlich, dass Datenintegration eine sehr problemspezifische Aufgabe ist und der 3LGM²-Baukasten mit seinen Anforderungen nur schwer mit bestehenden Konzepten der Datenintegration vereinbar ist.

Z.2 Zielerfüllung

Ziel Z1.1

Überblick über Softwareprodukte, die für 3LGM²-Modelle relevante Hard- oder Softwarebestände eines Unternehmens enthalten

Frage F1.1.1 Welche Softwareprodukte gibt es, die als Quellen für Informationen dienen können?

Frage F1.1.2 Welche potenziell nutzbaren Schnittstellen bieten diese Softwareprodukte?

Frage F1.1.3 Welche für 3LGM²-Modelle relevanten Daten und Schnittstellen bietet der „DX-Union Asset Assistant“?

Frage F1.1.4 Welche für 3LGM²-Modelle relevanten Daten und Schnittstellen bietet der „IBM Tivoli NetView“?

Frage F1.1.5 Welche Daten können/können nicht integriert werden?

Frage F1.1.6 Welche Daten sollten/sollten nicht integriert werden?

Frage F1.1.7 Ist es denkbar, den 3LGM²-Baukasten als Datenquelle für andere Softwareprodukte zu nutzen?

In Kapitel 5 wurden Klassen von Softwareprodukten betrachtet, die für 3LGM²-Modelle relevante Daten enthalten. Leider kann nie mit Sicherheit bestimmt werden, welche Daten in Softwareprodukten dieser Klassen zu finden sind, weshalb in Kapitel 6 konkrete Produkte auf ihre Daten und Schnittstellen hin untersucht wurden (F1.1.1).

Hinsichtlich der Schnittstellen ließen sich keine allgemeinen Aussagen treffen. Sind die Produkte allerdings „groß“ genug, so benutzen sie oftmals eine Datenbank, die als mögliche Schnittstelle dienen kann. Bei den konkreten Softwareprodukten konnten sowohl standardisierte als auch herstellerspezifische Schnittstellen gefunden werden (F1.1.2). Es wurde unter anderem auf den „DX-Union Asset Assistant“ (F1.1.3) und auf „IBM Tivoli NetView“ eingegangen (F1.1.4), wobei letzteres zu umfangreich ist, um hier abschließend beschrieben werden zu können.

Die Hoffnung, auf der Ebene von Produktklassen verlässliche Informationen über vorhandene Hard- und Softwarebestände zu finden, wurde nicht erfüllt und die abschließende Einzelbetrachtung jedes der Softwareprodukte war im Rahmen dieser Arbeit nicht realisierbar (F1.1.5). Allerdings ist umrissen worden, welche Daten in konkreten Softwareprodukten vorhanden sein können und welche davon eine vorstellbare Repräsentation in 3LGM²-Modellen besitzen (F1.1.6)(siehe Kapitel 6). Für die Repräsentation in 3LGM²-Modellen konnten 3LGM²-Metamodellklassen identifiziert werden, die aus Daten über Hard- und Softwarebeständen erstellt werden können (Kapitel 7). Wie der 3LGM²-Baukasten und 3LGM²-Modelle als Datenquelle verwendet werden kann, wurde in Kapitel 13 am Beispiel von Nagios untersucht (F1.1.7).

Ziel Z1.2

Konsistenzbedingungen für 3LGM²-Modelle, die externe Daten enthalten

Frage F1.2.1 Welche zusätzlichen Konsistenzbedingungen gelten für 3LGM²-Modelle mit externen Daten?

Frage F1.2.2 Wie kann man die Konsistenz des 3LGM²-Modells mit externen Daten sicherstellen?

Frage F1.2.3 Müssen externe Daten im 3LGM²-Modell besonders gekennzeichnet werden?

Es mussten keine zusätzlichen Konsistenzbedingungen innerhalb von 3LGM²-Modellen eingeführt werden. Um aber die Konsistenz nach außen zu den integrierten Ursprungsdaten zu halten, wurden Fremdschlüssel benötigt (F1.2.1) und für die Datenintegration mit Nagios ein Fremdschlüssel im Kapitel 12.6 definiert. Dieser Nagiosschlüssel ermöglicht das Erkennen entfernter und hinzugekommener Nagiosobjekte und bildete die Grundlage für die Synchronisa-

tion, die auf Veränderungen der externen Daten eingehen kann (F1.2.2). Da integrierte 3LGM²-Modellelemente im 3LGM²-Modell nicht besonders behandelt werden mussten, war es auch nicht nötig, sie zu kennzeichnen. Andererseits wurden sie bei der Datenintegration mit Nagios automatisch durch den Nagiosschlüssel gekennzeichnet (F1.2.3).

Ziel Z1.3

Strategien, wie externe Daten in ein 3LGM²-Modell integriert werden

Frage F1.3.1 Welche Anforderungen gibt es bei der Integration von Daten aus externen Quellen?

Frage F1.3.2 Müssen bei unterschiedlichen Quellen unterschiedliche Anforderungen an die Integration gestellt werden?

Frage F1.3.3 Inwieweit können externe Informationen über Krankenhausinformationssysteme automatisch in 3LGM²-Modelle abgebildet werden?

Frage F1.3.4 Wie erfolgt dies einmalig?

Frage F1.3.5 Wie erfolgt dies kontinuierlich?

Auf allgemeine Anforderungen für Datenintegration wurde in Kapitel 8 eingegangen (F1.3.1). Die Realisierung für Nagios in Kapitel 12 hat dann gezeigt, dass diese Anforderungen nicht nur für die Referenzarchitektur, sondern auch für eine konkrete Anwendung ausreichend sind (F1.3.2). Dennoch soll diese Liste von Anforderungen nicht abschließend sein und bei Bedarf erweitert werden können.

Die Automatisierung ist abhängig von den externen Daten. Für Nagios wurden daher Module entwickelt, die je nach Datenqualität angewendet werden können und zum Teil automatisch Daten integrieren (F1.3.3). In Kapitel 12 ist wiederum für Nagios die initiale Datenintegration sowie das Vorgehen für eine Synchronisation integrierter Daten beschrieben (F1.3.4 und F1.3.5). Eine allgemeine Beschreibung dieser Vorgänge war nicht möglich.

Ziel Z1.4

Implementierung einer Schnittstelle zwischen einem Softwareprodukt und dem 3LGM²-Baukasten

Frage F1.4.1 Welches Softwareprodukt bietet sich an, mit dem 3LGM²-Baukasten gekoppelt zu werden und wie?

Frage F1.4.2 Welcher Grad der Kopplung wird realisiert und warum?

Frage F1.4.3 Entspricht die Realisierung den Erwartungen?

Für das Softwareprodukt Nagios wurde prototypisch eine Schnittstelle implementiert. Die darin enthaltenen Daten wurden aus den Konfigurationsdateien gewonnen (F1.4.1). Wegen der Eigenschaften des 3LGM²-Baukastens konnte nur eine asynchrone und benutzerinitiierte Daten-

integration erreicht werden, die aufgrund der unterschiedlichen Datenqualität von Nagioskonfigurationen auch nur semiautomatisch realisierbar war (F1.4.2). Das Ziel, eine fertige Schnittstelle zu implementieren, ist leider nicht erreicht worden. Der entwickelte Prototyp enthält aber alle Konzepte, die zur Realisierung einer Datenintegration mit Nagios benötigt werden (F1.4.3).

Z.3 Ausblick

Da dies die theoretische Ausarbeitung einer praktischen Anwendung ist, sollten die hier vorgeschlagenen Konzepte zu einem dauerhaften Bestandteil des 3LGM²-Baukastens werden. Diese Entwicklung ist aber nicht unwesentlich von der Entwicklung des neuen 3LGM²-Baukastens abhängig und kann sich daher noch hinauszögern. Zwischenzeitlich könnten die Konzepte dieser Arbeit an weiteren Nagioskonfigurationen validiert werden.

Auch die Datenintegration mit anderen Netzwerkmanagement-Werkzeugen, die hier vorbereitet wurde, ist ein Betätigungsfeld für die Zukunft. wobei aufgrund des sehr individuellen Charakters von Datenintegrationen der Arbeitsaufwand nicht unerheblich ist.

Auch noch nicht berücksichtigt sind die Daten, die Nagios für das operative Informationsmanagement erhebt, die Verfügbarkeitsdaten. Dafür wurde in der Arbeit ein Vorschlag für eine funktionelle Integration gemacht, mit der diese Daten aus dem 3LGM²-Baukasten heraus in Erfahrung gebracht werden können. In diesem Zusammenhang muss aber auch beleuchtet werden, inwieweit ein Werkzeug des taktischen und strategischen Informationsmanagements, wie es der 3LGM²-Baukasten ist, operative Funktionalität bieten sollte.

Eine synchrone Datenintegration könnte auch erreicht werden, wenn man dafür in Kauf nimmt, einige der in der Arbeit formulierten Anforderungen aufzugeben.

Abkürzungen

CAFM Computer Aided Facility Management

CGI Common Gateway Interface

CI Configuration Item

CLI Command Line Interface

CMC Computer Multi Control

CMDB Configuration Management Database

CORBA Common Object Request Broker Architecture

DHCP Dynamic Host Configuration Protocol

DIN Deutsches Institut für Normung e.V.

FM Facility Management

FQDN Fully Qualified Domain Name

GEFMA German Facility Management Association

GNU *G*NU's *N*ot *U*nix

GPL General Public License

HGB Handelsgesetzbuch

HSK Für die Abbildung von Hard- und Softwarebeständen relevante 3LGM²-Metamodellklasse

IBM International Business Machines Corporation

IDL Interface Description Language

IEC International Electrotechnical Commission

IETF Internet Engineering Task Force

IMISE Institut für Medizinische Informatik, Statistik und Epidemiologie

IS Informationssystem

ISO International Organization for Standardization

ITIL IT Infrastructure Library

itSMF IT Service Management Forum

LDAP Lightweight Directory Access Protocol

LLDP Link Layer Discovery Protocol

MCC Medical Control Center

MIB Management Information Base

MIT Management Information Tree

MO Management Object

MOC Management Object Class

NNM Network Node Manager

OGC Office of Government Commerce

OID Object Identifier

OSI Open Systems Interconnection

POSIX *P*ortable *O*perating *S*ystem *I*nterface for Uni*x*

RFC Requests for Comment

RUP Rational Unified Process

SAM Software Asset Management

SAP SAP AG - Systems, Applications, and Products in Data Processing

SMI Structure of Management Information (Informationsmodell)

SNMP Simple Network Management Protocol

SQL Structured Query Language

TILAK Tiroler Landeskrankenanstalten GmbH

TMR Tivoli Management Regions

UKL Universitätsklinikum Leipzig AöR

UML Unified Modeling Language

URL Uniform Resource Locator

USV Unterbrechungsfreie Stromversorgung

WMI Windows Management Instrumentation

Literaturverzeichnis

[APPELRATH et al. 00] Appelrath H J, Ludewig J (2000): Skriptum Informatik. BG Teubner: Stuttgart, Leipzig, 194

[BARTH 05] Barth W (2005): Nagios: System und Netzwerk-Monitoring. Open Source Press GmbH: München, 243

[BATINI et al. 86] Batini C, Lenzerini M, Navathe S B (1986): A Comparative Analysis of Methodologies for Database Schema Integration. ACM Comput. Surv. 18(4), 323-364

[CALVANESE et al. 98] Calvanese D, De Giacomo G, Lenzerini M, Nardi D, Rosati R (1998): Description Logic Framework for Information Integration. Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'98), 2-13

[COOK et al. 99] Cook C, Darmawan B, Foster M, Gillardo S, Gucer V et.al. (1999): An Introduction to Tivoli Enterprise. International Technical Support Organization (IBM): Austin, Texas(USA), 43

[ELSÄSSER et al. 05] Elsässer W (2005): ITIL einführen und umsetzten. Hanser Verlag: München Wien, 6

[GALSTAD 06] Galstad E (2006): Nagios Website, www.nagios.org, aufgerufen 08/2006

[GEFMA 100-1 02] GEFMA (Deutscher Verband für Facility Management e.V.) (2002): GEFMA 100-1 Facility Management Grundlagen. Bonn: GEFMA

[GEFMA 400 02] GEFMA (Deutscher Verband für Facility Management e.V.) (2002): GEFMA 400 Computer Aided Facility Management CAFM. Bonn: GEFMA

[HAUX et al. 04] Haux R, Winter A, Ammenwerth E, Brigl B (2004): Strategic Information Management in Hospitals: An Introduction to Hospital Information Systems. Springer: New York, 127

[HEERTEN 96] Heerten E (1996): Einsatz integrierter Standard DV-Systeme beim Objektmanagement. Shaker Verlag: Aachen

[HEGERING et al. 99] Hegering H-G, Abeck S, Neumair B (1999): Integriertes Management vernetzter Systeme. dpunkt-verlag: Heidelberg

[HGB] Bundesrepublik Deutschland (2005): Deutsches Handelsgesetzbuch. Deutscher Taschenbuchverlag: München

[HP 06] HEWLETT-PACKARD (2006): HP OpenView solution and product guide

[ISO 06] ISO: ISO/IEC 19770-1 Information Technology - Software Asset Management - Part 1 Processes. www.iso.org/iso/en/commcentre/pressreleases/2006/Ref1006.html, 15.6.2006

[MICROSOFT 06] Microsoft: Software Asset Management - Ein Überblick. www.microsoft.com/germany/sam/default.mspx, 15.6.2006

[JUNG 06] Jung R (2006): Architekturen zur Datenintegration. Deutscher Universitätsverlag: Wiesbaden, 156

[KEMPER et al. 04] Kemper A, Eickler A (2004): Datenbanksysteme. Oldenbourg Wissenschaftsverlag: München, 21

[LAMPING et al. 94] Lamping J, Rao R (1994): Laying out and Visualizing Large Trees Using a Hyperbolic Space. UIST 94: Proceedings of the 7th annual ACM symposium on User interface software and technology, 13-14

[LENZERINI 02] Lenzerini M (2002): Data integration: a theoretical perspective. ACM PODS 02 June 3-6, 233

[MANOEL et al. 04] Manoel E, Colantuono C et. al. (2004): Implementing Tivoli Data Warehouse 1.2.Redbook, IBM

[NAUMANN 03] Naumann F (2003): Qualitätsgesteuerte Anfragebearbeitung für Integrierte Informationssysteme. it - Information Technology 45, 55

[OESTEREICH 05] Oestereich B (2005): Analyse und Design mit UML 2. Oldenbourg Verlag: München Wien, 273

[PERSON 01] Person R-D (2001): Instrumente für ein DV-gestütztes Gebäudemanagement an Hochschulen. Hochschul-Informations-System B2/2001, 4

[RAHM et al. 00] Rahm E, Do H-H (2000): Data Cleaning: Problems and Current Approaches. IEEE Bulletin of the Technical Committee on Data Engineering, Vol 23 No. 4, 1-11

[RAHM 94] Rahm E (1994): Mehrrechner-Datenbanksysteme: Grundlagen der verteilten und parallelen Datenbankverarbeitung. Addison-Wesley: Bonn, Paris, Kapitel 12

[SAGER 05] Sager F (2005): Konzeptentwicklung für Configuration Management in einem Rechenzentrum nach ITIL und MOF. Diplomarbeit, Institut für Informatik, Technische Universität München, 52

[STAT. BUNDESAMT 06] Statistisches Bundesamt Deutschland: Gesundheitswesen Grunddaten der Krankenhäuser 2005 (Fachserie 12 / Reihe 6.1.1). http://www.destatis.de/basis/d/gesu/gesutab29.php, 02.01.2007

[STRÜBING 05] Strübing A (2005): Entwicklung und Umsetzung eines Softwareassistenten für die 3LGM²-Modellierung. Diplomarbeit, IMISE, Universität Leipzig

[TERPLAN 95] Terplan K (1995): Client/Server Management. Datacom: Bergheim, 57

[WENDT et al. 04] Wendt T, Häber A, Brigl B, Winter A (2004): Modeling Hospital Information Systems (Part 2): Using the 3LGM² Tool for Modeling Patient Record Management. Methods of Information in Medicine. 43, 256-267

[WENDT 06] Wendt T (2006): Modellierung und Bewertung von Integration in Krankenhausinformationssystemen. Universität Leipzig, Hochschulschrift

[WINTER et al. 99] Winter Al, Winter An, Becker K, Bott O J, Brigl B, Gräber S, Hasselbring W, Haux R, Jostes C, Penger O-S, Prokosch H-U, Ritter J, Schütte R, Terstappen A (1999): Referenzmodelle für die Unterstützung des Managements von Krankenhausinformationssystemen. Informatik. Biometrie und Epidemiologie in Medizin und Biologie 30(4), 173-189

[WINTER et al. 03] Winter A, Brigl B, Wendt T (2003): Modeling Hospital Information Systems (Part 1): The Revised Three-layer Graph-based Meta Model 3LGM². Methods of Information in Medicine 42 (5), 544-551

[WINTER et al. 06] Alfred Winter und weitere Projektmitarbeiter: 3LGM²-Projekt Webseite, www.3lgm2.de, aufgerufen 07/2006

Abbildungsverzeichnis

2.1 Die Kapitel der Arbeit dargestellt im Zusammenhang mit den Themen und Teilen. 5

3.1 Das UML-Diagramm der Fachlichen Ebene des 3LGM²-Metamodells (aus [WINTER et al. 06]) . . . 10
3.2 Das UML-Diagramm der Logischen Werkzeugebene des 3LGM²-Metamodells (aus [WINTER et al. 06]) . . . 11
3.3 Das UML-Diagramm der Physischen Werkzeugebene des 3LGM²-Metamodells (aus [WINTER et al. 06]) . . . 12

4.1 Die Phasen der Datenintegration, dargestellt in Anlehnung an ein UML-Aktivitätsdiagramm. Graue Aktivitäten fassen dabei die schwarzen Aktivitäten zusammen. Die Datenintegration besteht aus der Struktur- und der Datenbestandsintegration, die sich wiederum in 2 Varianten aufteilen. . . . 18
4.2 Die Abhängigkeiten der globalen Datenstruktur bei der Datenintegration durch einen Bottom-Up-Ansatz . . . 20
4.3 Die Abhängigkeiten der globalen Datenstruktur bei der Datenintegration durch einen Top-Down-Ansatz . . . 20
4.4 Die verschiedenen Möglichkeiten einen globalen Datenbestand zu realisieren . . 23

5.1 Zusammensetzung des Inventars: Grundstücke und Gebäude, Maschinen, Fuhrpark und Geschäftsausstattung gehören zum Anlagevermögen. Verbrauchsmaterialien, Bankguthaben, Bargeld und anderes gehören zum Umlaufvermögen. Anlagevermögen und Umlaufvermögen zusammen bilden das Vermögen. . . . 28

6.1 Ein Screenshot des Asset Assistants . . . 40
6.2 Die NetView Management Console ist eine grafische Benutzerschnittstelle zum NetView System . . . 41
6.3 Ein Ausschnitt der openNMS-Oberfläche. Angezeigt werden Verfügbarkeitsdaten eines Servers. . . . 43

8.1 Links sind physische DV-Bausteine in einer Teil-von-Beziehung aufgeklappt dargestellt. Rechts sind die selben Bausteine zugeklappt dargestellt. . . . 53

9.1 Die Abhängigkeiten der globalen Datenstruktur bei der Datenintegration des 3LGM²-Baukastens . . . 56
9.2 Referenzarchitekturvorschlag für die Datenintegration . . . 58

10.1 Aufbau einer Nagios Installation; der Dämon führt Tests mit Hilfe der Plugins durch und die Benutzerschnittstelle ist über einen Webserver zugänglich. Common Gateway Interface (CGI) ist die Schnittstelle, mit der die HTML-Seiten erzeugt werden. Der Webserver und die CGI-Benutzerschnittstelle sind optional. . 63
10.2 Die eingeführte Terminologie als Klassendiagramm dargestellt. 73
10.3 Die Vererbungs-Relation ist in Teilbild a) und die Inklusions-Relation ist in Teilbild b) dargestellt. D = Definition, T = Nagiostemplate, MD = Nagiosmultidefinition 74
10.4 Schematische Darstellung einer Nagioskonfiguration mit internen Relationen (im Kreis) und externen Relationen (außerhalb des Kreises). S = Service, H = Host, HG = Hostgroup, SG = Servicegroup, D = Definition, T = Nagiostemplate, MD = Nagiosmultidefinition 75
10.5 Beispiel einer „Statusmap“ aus Nagios. Jede Verbindung ist eine `parents`-Relation. Je weiter „innen“ ein Knoten ausfällt („DOWN“) um so mehr äußere Knoten sind nicht mehr überprüfbar („UNKNOWN“).
(Quelle: Ev. Krankenhaus Fördergesellschaft Hamm) 77

11.1 Graphische Darstellung der Konfigurationsdaten aus Hamm durch den Prototyp. 80

12.1 Die lokale Datenstruktur des 3LGM²-Baukastens, reduziert auf die HSKs. 88
12.2 Das UML-Klassendiagramm der lokalen Datenstruktur von Nagios, abgeleitet aus den Nagiosobjekten 89
12.3 Die globale Datenstruktur für die Datenintegration. Sie ist aus dem zum Testen erstellten Prototypen entnommen. 90
12.4 Das Nassi-Shneiderman Diagramm des Moduls Physischer Datenverarbeitungsbaustein 93
12.5 Das Nassi-Shneiderman Diagramm des Moduls Physischer Bausteintyp 94
12.6 Das Nassi-Shneiderman Diagramm des Moduls Contactgroup/Timeperiod . . . 96
12.7 Das Nassi-Shneiderman Diagramm des Moduls ist_verbunden_mit-Relation . . . 97
12.8 Das Nassi-Shneiderman Diagramm der Synchronisation 104
12.9 Ein Screenshot des Prototypen der Schnittstelle 105

Tabellenverzeichnis

5.1 CI-Attribute für eine CMDB (vgl. [SAGER 05] S.52) 34

12.1 Zuordnung von Nagiosobjekttypen zu 3LGM²-Metamodellklassen 92
12.2 Ableitungen der Relation Host ↔ Hostgroup 93
12.3 Ableitungen der Relation Service ↔ Servicegroup 94

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Leipzig, 02.01.2007

Thomas Trommer