

# Systementwicklung eines Usertracking-Systems bei der Pirelli Deutschland GmbH

## Bachelorarbeit

im Studiengang Wirtschaftsinformatik



Verfasser: Denis Hamann

Kurs: WWI08B

Studienbereich: Wirtschaftsinformatik

Matrikelnummer: 253977

Wissenschaftlicher Betreuer: Olaf Rogge

Unternehmerischer Betreuer: Gerd Hoffarth

Abgabetermin: 14. Februar 2011

## Abstract

Verfasser: Denis Hamann  
Kurs: WWI08B  
Unternehmen: Pirelli Deutschland GmbH  
Thema: Systementwicklung eines Usertracking-Systems  
bei der Pirelli Deutschland GmbH

In der vorliegenden Arbeit wird ein System entwickelt, welches zum Erfassen aller im Netzwerk befindlichen Geräte dient. Hierfür wird zunächst auf die grundlegenden Technologien und Werkzeuge eingegangen, die für eine Umsetzung des Systems notwendig sind und in der Literatur behandelt werden. Neben Methoden zur Softwareentwicklung, wird auch auf die technischen Aspekte der Kommunikation im Netzwerk, sowie auf Datenbanken eingegangen. Im anschließenden Teil wird die aktuelle Situation bei Pirelli Deutschland GmbH untersucht. Anhand dieser werden Anforderungen an das System in Absprache mit den späteren Benutzern formuliert und diese in Hinblick auf die Realisierbarkeit überprüft. Im Hauptteil wird der Entwurf für das System erstellt. Dies erfolgt anhand der Modelle aus der UML. Bevor dieser Entwurf umgesetzt wird, findet eine Auswahl an Untersuchungen statt, die Ansätze aus der Theorie überprüfen. Diese Ansätze sind für die optimale Umsetzung des Projektes vorteilhaft und werden daher näher untersucht. Gegen Ende wird auf die Probleme, die bei der Implementierung auftraten, und deren Lösung eingegangen. Zum Abschluss der Arbeit werden weitere Anwendungsmöglichkeiten erörtert, sowie auf die Wirtschaftlichkeit der Umsetzung eingegangen.

# Inhaltsverzeichnis

<b>Abkürzungsverzeichnis</b>	<b>III</b>
<b>Abbildungsverzeichnis</b>	<b>IV</b>
<b>1. Problemstellung und Zielsetzung der Arbeit</b>	<b>1</b>
1.1. Zielsetzung . . . . .	1
1.2. Motivation . . . . .	2
1.3. Vorgehensweise . . . . .	2
<b>2. Theoretische Grundlagen</b>	<b>4</b>
2.1. Datenbanken . . . . .	4
2.1.1. Datenbankentwurf . . . . .	4
2.1.2. Kardinalitäten . . . . .	5
2.1.3. Normalisierung - Optimierung von Datenbanken . . . . .	6
2.1.4. Datenbankmanagementsysteme . . . . .	8
2.2. Webserver . . . . .	10
2.3. Schnittstellen . . . . .	11
2.4. Softwareentwicklung . . . . .	12
2.5. Unified Modeling Language . . . . .	16
2.6. Media Access Control . . . . .	21
2.7. Virtual Local Area Network . . . . .	23
2.8. Simple Network Management Protocol . . . . .	24
2.9. Cisco Discovery Protokoll . . . . .	26
<b>3. Analyse der jetzigen Situation der Pirelli Deutschland GbmH</b>	<b>28</b>
3.1. Situationsbeschreibung . . . . .	28
3.2. Anforderungsdefinition . . . . .	30
3.3. Anforderungsanalyse . . . . .	33
3.4. Betrachtung bereits existierender Lösungen . . . . .	35
3.5. Entscheidung zur eigenen Programmierung . . . . .	37

<b>4. Entwurf und Implementierung</b>	<b>38</b>
4.1. Entwurf . . . . .	38
4.1.1. Usecases . . . . .	38
4.1.2. Klassendiagramme . . . . .	40
4.1.3. Sequenzdiagramme . . . . .	43
4.1.4. Aktivitätsdiagramme . . . . .	46
4.1.5. Entity Relationship Model . . . . .	51
4.2. Design Entscheidungen . . . . .	54
4.3. Auswahl der Hilfsmittel . . . . .	58
4.4. Schnittstellen . . . . .	59
4.5. Zeitplan . . . . .	62
4.6. Realisierung . . . . .	64
4.7. Probleme bei der Implementierung . . . . .	67
4.8. Tests . . . . .	70
4.9. Weitere Anwendungsfelder / Datamining . . . . .	74
4.10. Wirtschaftliche Betrachtung . . . . .	75
<b>5. Fazit</b>	<b>79</b>
<b>Literaturverzeichnis</b>	<b>V</b>
<b>A. Anhang</b>	<b>VI</b>
A.1. Konfiguration SNMP-Track . . . . .	VI
A.2. Benchmarkwerte . . . . .	VI
<b>Ehrenwörtliche Erklärung</b>	<b>VII</b>

## Abkürzungsverzeichnis

<b>API</b>	Application Programming Interface
<b>BANF</b>	Bestellanforderung
<b>DB</b>	Datenbank
<b>DB2</b>	Ein DBMS von IBM
<b>DBS</b>	Datenbanksystem
<b>DBMS</b>	Datenbankmanagementsystem
<b>ERM</b>	Entity-Relationship-Modell: ein Gegenstands-Beziehungs-Modell
<b>GUI</b>	Graphical User Interface: grafische Benutzerschnittstelle
<b>MS</b>	Microsoft
<b>PC</b>	Personal Computer
<b>PD</b>	Pirelli Deutschland GmbH
<b>SAP</b>	SAP R/3 Software: Eine ERP-Software
<b>SQL</b>	Structured Query Language: Eine Datenbanksprache
<b>UNIX</b>	Multiuser Betriebssystem

## Abbildungsverzeichnis

Abbildung 1.	Chen Notation 1:N . . . . .	5
Abbildung 2.	Chen Notation N:M . . . . .	5
Abbildung 3.	Aufgelöste Chen N:M Notation . . . . .	6
Abbildung 4.	Keine Normalform angewendet . . . . .	7
Abbildung 5.	1. Normalform . . . . .	7
Abbildung 6.	Usecase-Diagramm . . . . .	18
Abbildung 7.	Klassendiagramm . . . . .	19
Abbildung 8.	Aktivitätsdiagramm . . . . .	20
Abbildung 9.	Sequenzdiagramm . . . . .	21
Abbildung 10.	Netzwerkarchitektur PD . . . . .	30
Abbildung 11.	Grafik mit Switches, Ports und Host . . . . .	32
Abbildung 12.	Usecasediagramm . . . . .	39
Abbildung 13.	Codebeispiel . . . . .	42
Abbildung 14.	Sequenzdiagramm 1 . . . . .	44
Abbildung 15.	Sequenzdiagramm 2 . . . . .	45
Abbildung 16.	Aktivitätsdiagramm 1 . . . . .	47
Abbildung 17.	Aktivitätsdiagramm 2 - Teil 1 . . . . .	49
Abbildung 18.	Aktivitätsdiagramm 2 - Teil 2 . . . . .	51
Abbildung 19.	ERM . . . . .	52
Abbildung 20.	Benchmark - Vergleich zwischen Perl und Java . . . . .	55
Abbildung 21.	Benchmark - Parallelisierung . . . . .	56
Abbildung 22.	Benchmark - SNMP Bulk . . . . .	57
Abbildung 23.	Benchmark - Oracle Transaktionen . . . . .	58
Abbildung 24.	Zeitplan - Arbeitspakete . . . . .	63
Abbildung 25.	Zeitplan - Gantt-Diagramm . . . . .	63
Abbildung 26.	Programm - Ausleseprogramm . . . . .	65
Abbildung 27.	Programm - Weboberfläche . . . . .	66

# 1. Problemstellung und Zielsetzung der Arbeit

## 1.1. Zielsetzung

Der Verfasser war während der 5. Praxisphase in der IT-Abteilung im Bereich Infrastruktur der Pirelli Deutschland GmbH (PD) eingesetzt. Zu den Aufgaben der Infrastruktur gehört die Sicherstellung des Betriebes der Anwendungssoftware. Um den Einsatz dieser Software zu ermöglichen, kommt eine immense Anzahl an Hardware zum Einsatz. Ein Teil der Hardware stellt unter anderem das Netzwerk dar. Neben dem Aufbau und der Konfiguration des Netzwerkes zählt auch der Betrieb zu den Kernaufgaben. Um den Betrieb des Netzwerks sicherzustellen, ist es notwendig, einen Überblick über dieses zu behalten. Im Netzwerkmanagement gibt es hierfür einen speziellen Part, welcher sich unter dem Begriff des Netzwerkmonitoring zusammenfassen lässt. Beim Netzwerkmonitoring gibt es die Möglichkeit sowohl Hardware als auch Anwendungen zu überwachen. Ein spezieller Bereich konzentriert sich auf das Erfassen aller Geräte im Netzwerk. Dieses wird z.B. von dem Hersteller Cisco als 'Usertracking' bezeichnet. Hierfür soll in der Arbeit ein Entwurf erstellt werden, der auf Basis der Anforderungen des Unternehmens erstellt wird. Um alle Anforderungen zu erfassen, soll im Voraus eine ausgiebige Analyse der Ist-Situation stattfinden. An diese folgt dann die Implementierung des Systems. Das Ziel soll es sein, Konzepte und Technologien aus der Literatur zu überprüfen und diese gezielt anzuwenden, um eine optimale Umsetzung zu erreichen. In der Arbeit selbst wird nicht im Detail auf die implementierte Lösung eingegangen, da dies den Umfang der Bachelorarbeit überschreitet. Daher entfallen auch Erläuterungen zu eingesetzten Algorithmen und dem Quellcode. Vielmehr wird sich auf den Entwurf und dessen Entscheidungsfindung konzentriert, da diese das Grundgerüst für die Implementierung liefern. Im Mittelpunkt soll das Vorgehen, hilfreiche Konzepte aus der Theorie und eine kritische Betrachtung der Ausgangslage stehen, um eine optimale Umsetzung zu garantieren.

## 1.2. Motivation

Da bei PD, aufgrund der ausgelaufenen Wartung, eine Aktualisierung der bestehenden 'CiscoWorks' Lösung ansteht, ist eine Untersuchung der aktuellen Situation notwendig. Zunächst ist es von Bedeutung, die bestehenden Lizenzkosten der Software, zu senken. Besonders im Zuge des immer weiter verbreiteten Einsatzes von OpenSource-Software im gewerblichen Bereich muss untersucht werden, ob es für die bisher eingesetzte Lösung ein Ersatz gibt bzw. ob eine eigene Lösung mit der Verwendung bereits existierender OpenSource-Software bewerkstelligt werden kann. Neben den monetären Aspekten spielen aber auch technische Argumente eine Rolle. So können bei eigenen Entwicklungen beispielsweise Anpassungen vorgenommen werden, die bei Standardsoftware nicht möglich sind. Z.B. besteht dann die Möglichkeit, Switches von anderen Herstellern ebenfalls auszulesen, aber auch neuere Geräte ohne größere Verzögerung von dem System zu unterstützen. Hinzukommt, dass durch das Auffinden der Hosts im Netzwerk die entsprechenden Switches bekannt sind, an die diese angeschlossen sind. Dadurch ergibt sich nicht nur die Möglichkeit einer Inventarisierung, sondern auch Diagnosemöglichkeiten. So kann für den Helpdesk die Möglichkeit bestehen, zu überprüfen, mit welcher Geschwindigkeit ein Host aktuell an das Netzwerk angeschlossen ist. Dies ist vor allem sinnvoll, wenn die Verbindung zum jeweiligen Host sehr langsam ist, dann kann überprüft werden, ob es sich eventuell um ein Duplex-Mismatch oder ähnliches handelt.

## 1.3. Vorgehensweise

Die Arbeit gliedert sich in mehrere Abschnitte. Zunächst wird auf wichtige Grundlagen eingegangen werden, welche für den Entwurf und die Implementierung eines passenden Systems notwendig sind. Im Anschluss findet die Untersuchung der Ausgangssituation statt, sowie einer entsprechenden Anforderungsdefinition und Anforderungsanalyse. Hierbei wird das bestehende System untersucht und anhand dessen in Zusammenarbeit der Mitarbeiter eine Anforderung für ein Ersatzsystem definiert. Diese Anforderungen werden im Hinblick auf die technische und zeitliche Machbarkeit hin untersucht und mit bereits auf dem Markt existierenden Lösungen abgeglichen. Im Anschluss folgt der, in der Softwareentwicklung typische, nächste Schritt, der Entwurf. Durch diesen entsteht das Modell des späteren Systems auf Grundlage der Anforderungen mit Hilfe von Techniken aus der Theorie. Daraufhin wird vom Verfasser eine Vielzahl von Untersuchungen



vorgenommen, um die Designentscheidungen im Bezug auf deren Auswirkung auf die Umsetzung der Anforderungen sicherzustellen. In diesem Zusammenhang wird auch auf den Zeitplan, sowie die Probleme bei der Implementierung eingegangen. Zum Ende der Arbeit werden zusätzlich mögliche Anwendungsfelder der implementierten Lösung angesprochen und eine wirtschaftliche Betrachtung durchgeführt, welche überprüfen soll, ob die Implementierung des eigenen Systems eine ökonomisch sinnvolle Entscheidung ist.

## 2. Theoretische Grundlagen

### 2.1. Datenbanken

#### 2.1.1. Datenbankentwurf

Unter dem Datenbankentwurf ist der Prozess zur Erstellung eines Schemas zu verstehen, welches die spätere Datenbank abbilden wird. Hierunter fallen unter anderem die Analyse der Anforderungen, aber auch die grafische Darstellung der Tabellen, in denen die Daten gespeichert werden. Der Entwurf der Datenbank vor der Implementierung ist essentiell, da im späteren Prozess Änderungen der Datenbankstruktur nicht nur die Datenbank selbst, sondern auch alle mit ihr verbundenen Applikationen betreffen werden. Um die Beziehungen zwischen den einzelnen Tabellen korrekt darstellen zu können, wird das Entity-Relationship-Modell<sup>1</sup> verwendet. Durch dieses Modell lassen sich sogenannte ER-Diagramme zeichnen, z.B. nach der Chen Notation<sup>2</sup>. Ein ER-Diagramm nach Chen stellt die Entitätstypen (Klassen), Attribute, sowie Beziehungen (Relationen/Kardinalitäten) dar. Im Folgenden Beispiel soll ein ER-Diagramm nach der Chen-Notation erläutert werden.

Das nachfolgende Diagramm beschreibt folgenden Sachverhalt:

- Ein Angestellter leitet mehrere Projekte.
- Ein Projekt wird von einem Angestellten geleitet.

---

<sup>1</sup>vgl. Peter Pin-Shan Chen(1976): The Entity-Relationship Model–Toward a Unified View of Data. In: ACM Transactions on Database Systems, Vol 1, No 1, S.10

<sup>2</sup>vgl. Peter Pin-Shan Chen(1976): The Entity-Relationship Model–Toward a Unified View of Data. In: ACM Transactions on Database Systems, Vol 1, No 1, S.19

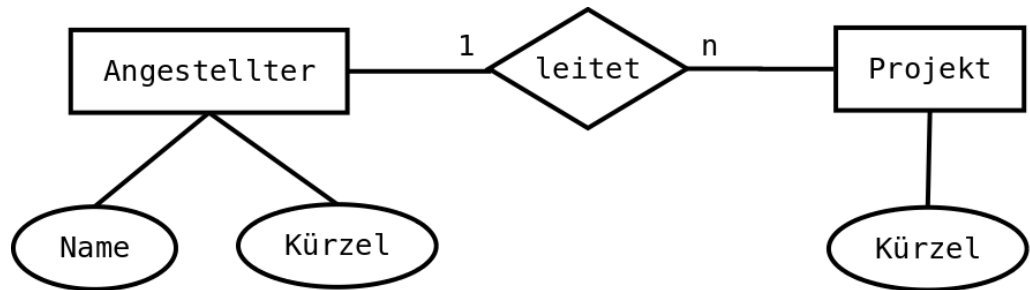


Abbildung 1.: Chen Notation 1:N

- Ein Autor verfasst mehrere Bücher.
- Ein Buch wird von mehreren Autoren verfasst.

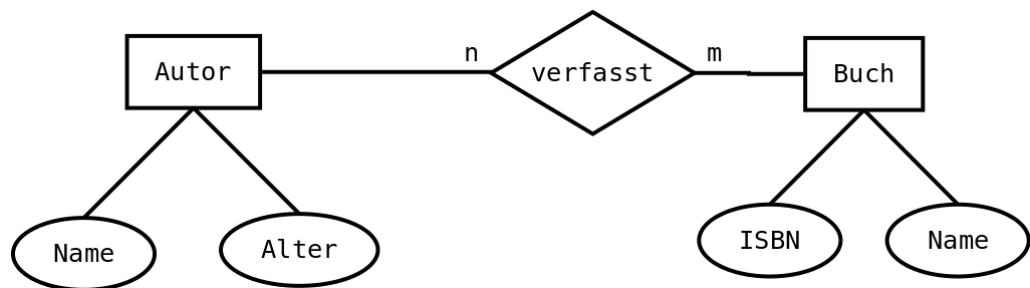


Abbildung 2.: Chen Notation N:M

### 2.1.2. Kardinalitäten

Kardinalitäten beschreiben den Grad einer Verbindung zwischen zwei Objekten.<sup>3</sup> In Abbildung 1 ist eine 1:n Kardinalität gegeben. Diese sagt aus, dass einem Objekt der Relation 1, mehrere Objekte der Relation 2 zugeordnet werden, einem Objekt der Relation 2 jedoch nur ein Objekt der Relation 1.

In Abbildung 2 ist eine n:m Kardinalität zu sehen. Diese sagt aus, dass einem Objekt der Relation 1, mehrere Objekte der Relation 2 angehören, einem Objekt der Relation

<sup>2</sup>In Anlehnung an

<sup>3</sup>vgl. Heinz Burnus(2007): Datenbankentwicklung in IT-Berufen, 1. Auflage, S.20

2 werden ebenfalls mehrere Objekten der Relation 1 zugewiesen. Diese n:m Kardinalitäten müssen bei einem Datenbankentwurf aufgelöst werden, da hier keine eindeutige Zuordnung möglich ist. Meistens lässt sich eine solche Kardinalität wie in Abbildung 2 durch das Hinzufügen einer zusätzlichen Tabelle, welche beide Objekte verknüpft, lösen. Ein Beispiel ist in Abbildung 3 zu sehen.



Abbildung 3.: Aufgelöste Chen N:M Notation

### 2.1.3. Normalisierung - Optimierung von Datenbanken

Wenn es bereits bestehende Datenbanken gibt, so muss geprüft werden, ob diese eine optimale Struktur aufweisen. Eine Optimierung der Struktur ergibt sich nicht nur, um eine bessere Übersichtlichkeit zu haben, sondern auch aus Gründen der Redundanz und der damit verbundenen Probleme. Durch eine nicht benötigte Redundanz kommt es zu Integritäts- und Platzverlusten innerhalb der Datenbank. Da ein Wert an mehreren Stellen in der Datenbank steht, kann es zu Problemen bei der Aktualisierung und Löschungen kommen, da alle Werte in der Datenbank geändert werden müssten und nicht nur ein Wert. Dies kann anhand der sogenannten Normalisierung verhindert werden, sodass eine Optimierung stattfindet.<sup>45</sup> Bei der Normalisierung wird überprüft, ob Tabellen gewisse Eigenschaften erfüllen und sofern dies nicht der Fall ist, wird versucht diese zu erreichen. Hierzu stehen bis zu fünf Stufen der Normalformen zur Verfügung. Diese fünf Normalformen lassen sich durch die folgenden Attribute beschreiben.<sup>6</sup>

1. Normalform: Alle Attribute besitzen einen atomaren Wertebereich
2. Normalform: Jedes Nichtschlüsselattribut ist vom kompletten Schlüssel abhängig

<sup>4</sup>vgl. E. F. Codd(1970): A Relational Model of Data for Large Shared Data Banks in Commun. ACM, Vol 13, Nr. 6, S. 381

<sup>5</sup>vgl. Prof. Dr. Paul. Alpar(2001): Vorlesung, Datenorganisation und Datenbanken, <http://www.tekinci.de/skripte/DBDM/DB-SS2001.pdf>

<sup>6</sup>vgl. Heinz Burnus(2007): Datenbankentwicklung in IT-Berufen, 1. Auflage, S.292-308

3. Normalform: Jedes Nichtschlüsselattribut ist von keinem Schlüsselkandidaten transitiv abhängig, dass heißt kein Attribut ist über ein anderes vom Hauptschlüssel abhängig
4. Normalform: Es darf in einer Relation nicht mehrere, voneinander unabhängige, 1:n-Beziehungen zu einem Schlüsselwert geben
5. Normalform: Es existieren nur noch Einzel-Abhängigkeiten

In der ersten Normalform wird untersucht, ob jedes Attribut atomare Werte besitzt, dass heißt es enthält nur einen Wert und ist frei von Wiederholungen.<sup>7</sup>

	A
1	ReifenDimension
2	195/50R15

Abbildung 4.: Keine Normalform angewendet

In Abbildung 4 ist eine Verletzung der 1. Normalform zu sehen. Um diese aufzuheben, müssen die einzelnen Werte, wie in Abbildung 5 zu sehen, getrennt werden.

	A	B	C	D
1	Reifenbreite	Flankenhöhe	Reifenbauart	Durchmesser
2	195	50	R	15

Abbildung 5.: 1. Normalform

Wurde die Relation entsprechend angepasst, so ist die 1. Normalform erreicht und es kann nun geprüft werden, ob diese die Eigenschaften der 2. Normalform erfüllt. Um eine Normalform zu erfüllen, müssen auch alle vorhergehenden Normalformen erfüllt sein, dass heißt, erfüllt eine Tabelle die 3. Normalform, so erfüllt sie auch die 1. und 2. Normalform.

<sup>7</sup>vgl. Matthias Schubert(2007): Datenbanken, Theorie, Entwurf und Programmierung relationaler Datenbanken, 2. Auflage, S.293

### 2.1.4. Datenbankmanagementsysteme

Ein Datenbankmanagementsystem (DBMS) organisiert die Speicherung der Daten einer Datenbank und legt die Anordnung der Daten fest. Zur Kommunikation mit diesem wird eine Sprache benötigt. In diesem Zusammenhang wird die deskriptive Sprache SQL verwendet.<sup>8</sup>

Das DBMS legt auch die Art der Beziehung fest, in der die Daten der Datenbank stehen.

Es gibt verschiedene Arten von DBMS:

- Hierarchisch
- Relational
- Objektorientiert

Ein hierarchisches DBMS dient vor allem der schnellen Suche in großen Datenbanken.<sup>9</sup> Der Nachteil liegt darin, dass nur eine sequentielle Abarbeitung möglich ist und somit die Art der Abfragen mehr oder weniger schon im Voraus bestimmt sein muss. Im Gegensatz hierzu stehen die relationalen Datenbanksysteme, welche heutzutage den höchsten Verbreitungsgrad besitzen.<sup>10</sup> Diese bieten eine flexible Auswertung der Daten durch die deklarative Abfragesprache SQL. Es muss lediglich die Verknüpfung zwischen den Tabellen durch sogenannte JOINS hergestellt werden. Somit werden Primär- und Fremdschlüssel miteinander verknüpft. Hinzu kommen die objektorientierten Datenbanksysteme. Diese bieten die Möglichkeit Objekte von beliebiger Art und Weise abzuspeichern. Problematisch hierbei sind jedoch die Formulierung von geeigneten Abfragen, weswegen diese in der Praxis eher selten und meist im Bereich von Multimedialen-Anwendungen anzutreffen sind. Wird versucht, verschiedene relationale DBMS in der Praxis zu vergleichen, so ist eine große Anzahl an verschiedenen Systemen zu finden. Im Anschluss soll auf die, in der Praxis verwendeten relationalen DBMS eingegangen werden.

- Microsoft Jet Engine (Access)

---

<sup>8</sup>vgl. E. F. Codd(1970): A Relational Model of Data for Large Shared Data Banks in Commun. ACM, Vol 13, Nr. 6, S. 382

<sup>9</sup>vgl. Bernd-Jürgen Falkowski(2002): Business Computing: Grundlagen und Standardsoftware, 1. Auflage, S.235

<sup>10</sup>Quelle

- MS-SQL Server
- Oracle
- MySQL
- PostgreSQL

Die Microsoft Jet Engine (Access) ist ein dateibasierendes DBMS, welches dem Benutzer eine einfache Möglichkeit bietet, Daten in einer Datenbank zu speichern und passende Oberflächen (Frontends) in der Datenbank zu integrieren. Bei diesem System, wie bei allen anderen dateibasierenden DBMS, steht meist die einfache Konfigurierbarkeit im Vordergrund. Die Datenbanken sind meist für einen Einzeluser-Betrieb ausgelegt und spielen hier auch ihre Stärken aus. Wird eine dateibasierende Datenbank von mehreren Usern benutzt, so zeigen sich die Nachteile einer solchen Datenbank. Dadurch, dass Access für jeden Nutzer bei einer Abfrage die komplette Datenbank durchsucht, entsteht eine hohe Auslastung der Festplatte, sowie des Netzwerks. Daher nimmt Geschwindigkeit bei mehreren Anwendern exponentiell ab, da sich alle Benutzer die Bandbreite der Festplatte sowie des Netzwerkes teilen. Auch beim Speichern müssen zusätzlich Datensätze gesperrt und organisiert werden, da sonst die Daten inkonsistent werden können, wenn mehrere Personen gleichzeitig einen Datensatz schreiben.

MS-SQL ist ein relationales DBMS, welches von Microsoft erwerbbar und in den verschiedenen Windows-Serverbetriebssystemen enthalten ist. Für Entwickler ohne Enterprise Lizenz wird eine eingeschränkte Express Version zur Verfügung gestellt. MS-SQL ist im Gegensatz zu Access kein dateibasierendes DBMS, sondern ein zentralisiertes, dass auf einem Server läuft. Hierdurch werden die Nachteile des dateibasierenden zu den Vorteilen des serverbasierenden Systems. Da der Server selbst die Abfragen verwaltet und zusätzlich Abfragen im Arbeitsspeicher ablegt, sowie dem Benutzer nur die Daten sendet, die er auch angefordert hat und nicht die komplette Datei, werden Zugriffszeiten und Netzwerk/Festplattenlast optimiert.

Ein weiteres DBMS stellt der Datenbankserver 'Oracle Database' von Oracle dar. Die Lizenzstruktur ähnelt der von Microsoft. So gibt es auch hier kostenfreie und kostenpflichtige Varianten. Im Gegenzug zum MS-SQL Server bietet Oracle ein breiteres Spektrum an Funktionalitäten und eine größere Konfigurationsmöglichkeit. Dieses wiederum

macht sich im höheren Preis bemerkbar. Weitere Vorteile der Oracle-Datenbank sind die weitgehende Betriebssystemunabhängigkeit, gute Dokumentation und der Support.

MySQL ist der Open Source Pendant zu MS-SQL, welches im Internet eine sehr hohen Verbreitungsgrad aufweist. So wird dieses von Seiten wie Wikipedia<sup>11</sup> oder Youtube<sup>12</sup> verwendet. Im Gegensatz zu MS-SQL erlaubt das GPL Lizenzmodell, dass die Datenbank für Privatanwender kostenlos ist und die Lizenzgebühren für Unternehmen einen Bruchteil der Kosten ausmachen, die für ein Microsoft System bezahlt werden müssten.<sup>13</sup>

Eine weitere Datenbank stellt PostgreSQL dar, welches unter der BSD-Lizenz zur Verfügung gestellt wird und somit auch für kommerzielle Projekte ohne Kosten nutzbar ist<sup>14</sup>.

## 2.2. Webserver

Ein Webserver dient zum Bereitstellen von statischen, sowie dynamischen HTML Seiten. Der Vorteil bei Webservern liegt darin, dass nur Informationen ausgetauscht werden, die der Nutzer auch angefordert hat. Weiterhin bietet es den Vorteil, diese zielgerichteten Informationen einer größeren Menge an Benutzern zur Verfügung zu stellen, ohne dass spezielle Vorkehrungen zur späteren Skalierung getroffen werden müssen.

Um dynamische Webseiten erzeugen zu können, bedarf es einer Skriptsprache. Aktuell haben sich folgende Sprachen etabliert:<sup>15</sup>

- ASP - Active Server Pages
- JSP - JavaServer Pages
- PHP - PHP: Hypertext Preprocessor

<sup>11</sup>vgl. Mysql, <http://www.mysql.com/why-mysql/scaleout/wikipedia.html>

<sup>12</sup>vgl. University of Maryland: How YouTube scales MySQL for its large databases, <http://ebiquity.umbc.edu/blogger/2007/12/28/how-youtube-scales-mysql-for-its-large-databases/>

<sup>13</sup>Vgl [http://www.mindfactory.de/product\\_info.php/pid/geizhals/info/p155132](http://www.mindfactory.de/product_info.php/pid/geizhals/info/p155132)

<sup>14</sup>vgl. PostgreSQL: BSD-Lizenz, <http://www.postgresql.org/about/licence>

<sup>15</sup>vgl. Tiobe Software(2009): TIOBE Programming Community Index for August 2009, <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>



ASP ist eine Skriptsprache der Firma Microsoft und basiert grundlegend auf der Syntax von Visual Basic. JSP dient dem selben Zweck, wurde jedoch von Sun entwickelt und besitzt die Syntax von Java. PHP ist eine Skriptsprache, welche sich hauptsächlich an der C Syntax orientiert und speziell für das Erstellen von dynamischen Webseiten entwickelt wurde. Sie ist die am weitesten verbreitete Skriptsprache zum Erstellen von dynamischen Webseiten.

Da alle der aufgeführten Skriptsprachen weitgehend Webserver-/Plattformunabhängig sind, kann man frei zwischen den meist benutzten Webserverprogrammen wählen, hierunter fallen unter anderem:

- Apache
- IIS - Internet Information Services

Der Apache Webserver ist ein Open Source Webserver der Apache Foundation. Er kann unter vielen verschiedenen Betriebssystemen eingesetzt werden und unterstützt durch seine Module alle verbreiteten Skriptsprachen, sowie Datenbanken.

Der IIS Webserver von Microsoft läuft ausschließlich unter Windows, zudem ist es auch nur unter dem Serverbetriebssystem von Windows möglich, mehr als zehn Verbindungen gleichzeitig aufzubauen.<sup>16</sup>

## 2.3. Schnittstellen

Um eine Verbindung zwischen den einzelnen Komponenten herzustellen, wird eine passende API benötigt. Idealerweise enthält jedes DBMS und jeder Webserver passende Treiber, obwohl dieser Fall nicht immer gegeben ist.

Zurzeit haben sich verschiedene Varianten etabliert:

- Native Treiber
- ODBC (Open Database Connectivity)

---

<sup>16</sup>vgl. Microsoft: IIS 7.0: Übersicht über die verfügbaren Features in IIS 7.0, <http://msdn.microsoft.com/de-de/library/cc753198%28WS.10%29.aspx>

- JDBC (Java Database Connectivity)
- ADO (ActiveX Data Objects)

Jedes etablierte DBMS bietet für Entwickler eine native Schnittstelle zum Ansprechen der Datenbank an. Diese wird meist in verschiedenen Programmiersprachen angeboten, unter anderem C/C++, PHP usw.. Die native Methode besitzt den Vorteil, dass sie den vollen Umfang der Funktionen der Datenbank ermöglicht. Dahingegen existieren diese keine einheitlichen Design Patterns, was die Verwendungen von verschiedenen DBMS erschwert.

Aufgrund dessen hat sich in der Vergangenheit unter Windows die sogenannte Open Database Connectivity-Schnittstelle (ODBC) etabliert, welche es ermöglicht, ohne Veränderung des Quellcodes, das Datenbanksystem zu wechseln. Hierzu muss lediglich der ODBC-Eintrag angepasst werden. Ein weiterer Vorteil, der sich durch die Schnittstelle ergibt, ist die Möglichkeit, alle Abfragen im standardisierten SQL absetzen zu können, unabhängig vom verwendeten DBMS. Neben der ODBC-Schnittstelle gibt es auch die JDBC-Schnittstelle bei Java. Zwar kann Java auch die ODBC-Schnittstelle über eine Treiber-Brücke nutzen, aber laut Oracle<sup>17</sup> soll dies nur im experimentellen Gebrauch stattfinden. Da der Einsatz eines nativen JDBC-Treibers verhindert, dass unerwünschten Zustände auftreten, welche z.B. bei einer JDBC-ODBC Anbindung möglich sind, ist die Verwendung des nativen Treibers vorzuziehen. Zusätzlich gibt es noch ADO von Microsoft, welches auch als Schnittstelle für Webserver mit IIS bzw. ASP dient. Diese ähnelt in der Verwendung JDBC, da ebenfalls ein ODBC-Treiber angesteuert werden kann. Der Verbindungsaufbau ist ähnlich einfach gestaltet und ermöglicht auch für Webanwendungen eine einfache Anbindung an die Datenbank.

## 2.4. Softwareentwicklung

Unter der Softwareentwicklung ist die Herstellung und Entwicklung von Software, sowie die dazugehörige Planung und Modellierung dieser zu verstehen. Die Softwareentwicklung umfasst eine Vielzahl von Teilgebieten. Die Entwicklung einer komplexen Software wird anhand eines strukturierten Projektplans vorgenommen, welcher den En-

---

<sup>17</sup>Vgl. Oracle: JDBC-ODBC Bridge Driver, <http://download.oracle.com/javase/1.3/docs/guide/jdbc/getstart>.

twicklungsprozess inhaltlich und zeitlich abgrenzt.<sup>18</sup> Die Software wird anhand von bestimmten Schritten fertiggestellt, welche miteinander eng verzahnt sind. Unterschieden wird bei der Softwareentwicklung zwischen Individualsoftware und Standardsoftware. Bei der Standard-Software handelt es sich um Software, welche einen klar definierten Anwendungsbereich abdeckt und als vorgefertigtes Produkt erworben werden kann. Standardsoftware zeichnet sich somit aus, dass diese über mehrere Kunden hinweg ohne Anpassung einsetzbar ist.<sup>19</sup> Ein Beispiel hierfür sind branchenunabhängige Software, wie z.B. das Office-Paket, aber auch Branchensoftware, welche zielgerichtet für eine Branche ist und in dieser übergreifend eingesetzt werden kann.

Bei der Individualsoftware handelt es sich um Software, welche individuell für einen Kunden angefertigt wurde.<sup>20</sup> Typisch für Individualsoftware ist es, dass zuvor keine passenden Lösungen an Standardsoftware existiert haben. Es kann aber auch sein, dass die Entwicklung einer Individualsoftware trotz existierender Standardsoftware Sinn macht, sofern es monetär vorteilhaft ist. Eine weitere Überlegung könnte der Versuch einen Wettbewerbsvorteil gegenüber den Wettbewerbern zu erhalten oder die Optimierung einer vorhandenen Lösung sein.

Die Umsetzung eines Projektes findet entweder intern oder von einem externen Dienstleister statt. Eine wichtige Rolle spielen ebenfalls die Vorgehensweisen bei der Umsetzung eines Projektes. Hier gibt es die Wahl zwischen stark strukturierten Herangehensweisen, wie das Wasserfallmodell bis hin zu sehr flexiblen, z.B. der Agilen Softwareentwicklung.<sup>21</sup> Im Folgenden soll auf die wichtigsten Kernprozesse bei der Umsetzung eines Systems in einem Projekt eingegangen werden.

## Planung

Zu Beginn einer Systementwicklung steht die Planung, in der die Anforderungen erhoben

<sup>18</sup>Softwareentwicklung kompakt und verständlich: Wie Softwaresysteme entstehen, Hans Brandt-Pook, Rainer Kollmeier, Verlag: Vieweg+Teubner; Auflage: 1 (27. Mai 2008)

<sup>19</sup>Integrales Logistikmanagement. Planung und Steuerung der umfassenden Supply Chain, Paul Schönsleben, S. 435, Verlag: Springer, Berlin; Auflage: 4., überarb. u. erw. A. (März 2007)

<sup>20</sup>Integrales Logistikmanagement. Planung und Steuerung der umfassenden Supply Chain, Paul Schönsleben, S. 436f, Verlag: Springer, Berlin; Auflage: 4., überarb. u. erw. A. (März 2007)

<sup>21</sup>Praxishandbuch IT im Gesundheitswesen. Erfolgreich einführen, entwickeln, anwenden und betreiben, Christian Johnner, S. 5-7, Peter Haas, Hanser Fachbuch (5. März 2009)

werden.<sup>22</sup> Hierbei handelt es sich um das Sammeln aller Anforderungen, die seitens des Kunden oder aufgrund von externen Einflüssen (z.B. Gesetze) gegeben sind. Währenddessen ist vor allem der Dialog mit dem Kunden, aber auch mit den späteren Benutzern sowie den fachlichen Experten notwendig. In dieser Phase wird neben dem Lastenheft (Anforderungsdefinition) auch das Pflichtenheft erstellt. Es erfolgt auch eine Aufwandseinschätzung, sowie die Wahl des Vorgehensmodells.

### Analyse

Im Analyse-Prozess findet die Auswertung der zuvor gesammelten Anforderungen statt. Bei dieser Auswertung kommt es auch zur Analyse der Prozesse und des Systems. Bei der Systemanalyse kommt es bereits zum ersten Modellentwurf, wobei dieser explizit ohne "Maschinen", d.h. ohne systemspezifische Inhalte, ist und somit technische Details noch nicht ins Modell aufgenommen werden. Sofern die Möglichkeit besteht, wird in diesem Prozess auch ein Mock-up erstellt. Bei einem Mock-up handelt es sich um ein Modell bzw. einer Nachbildung, welche meist eine Attrappe darstellt. In der Softwareentwicklung wird darunter ein Prototyp verstanden, welcher rudimentär die Benutzerschnittstelle widerspiegelt. Er wird vor allem zu Beginn des Projektes eingesetzt, um eine bessere Zusammenarbeit mit dem Auftraggeber und dem späteren Anwender zu erlangen. Somit können die Anforderungen an die Benutzeroberfläche direkt besprochen werden und die Beteiligten sich ein besseres Bild über die spätere Anwendung machen.

### Entwurf

Beim Entwurfsprozess geht es um die Planung der Software-Lösung. Zur Planung von dieser werden unterschiedliche Sprachen zur Modellierung verwendet. Die wichtigste Sprache hierbei ist UML, welche unter anderem auch die Modellierung von Klassen und Objekten, sowie deren Beziehungen untereinander ermöglicht. Auf UML wird in Kapitel X detailliert eingegangen. Zum Entwurf gehören ebenfalls System- bzw. Designentscheidungen, die später in die Programmierung einfließen.

---

<sup>22</sup>CSCW-Kompodium: Lehr- und Handbuch zum computerunterstützten kooperativen Arbeiten, S.98, Gerhard Schwabe, Norbert Streitz, Rainer Unland, Springer, Berlin; Auflage: 1 (1. August 2001)

### Programmierung

Bei der Programmierung geht es letztendlich um die Umsetzung des zuvor entworfenen Systems. Hierbei wird je nach Vorgehensweise die strukturierte oder objektorientierte Programmierung angewandt.

### Validierung und Verifizierung

Bei der Validierung und Verifizierung geht es vor allem um Tests. Hierbei wird unterschieden zwischen Low-Level-Tests und High-Level-Tests.<sup>23</sup> Unter Low-Level-Tests sind solche Test zu verstehen, die während der Implementierung an Teilen des Systems stattfinden. Bei High-Level-Tests wird das komplette System getestet. Einer der Low-Level-Tests ist der Modultest. Bei diesem werden einzelne Module im Programm getestet. Diese Tests werden regelmäßig während der Entwicklung durchgeführt. Ein weiterer Low-Level-Test ist der Integrationstest. Bei diesem werden verschiedene Module in Kombination getestet. Für jede Verbindung zwischen zwei Komponenten wird ein Test erstellt, welcher überprüft, ob diese ordnungsgemäß nach der Spezifikation funktionieren. In kleineren Projekten findet der Integrationstest meist während der Implementierung durch die Programmierer statt. Der so genannte Systemtest ist ein High-Level-Test bei dem das gesamte Programm gegen die zuvor definierten Anforderungen gegen geprüft wird. Dieser Systemtest findet meist in einer Testumgebung statt und erhält meist simulierte Testdaten um die bestehende Produktivumgebung nicht weiter zu beeinträchtigen. Die simulierten Testdaten können trotz Allem den reellen Daten entsprechen, sollen jedoch verhindern, dass das System direkt in die Produktivumgebung einwirkt. Der letzte High-Level-Test dient der Abnahme und wird auch als Akzeptanztest bezeichnet. Bei diesem Test geht es um den Test der Software im produktiven Einsatz beim Kunden. Der Test selbst stellt ein Black-Box-Test dar und dient meist zu Rechnungsstellung bzw. Abnahme in Verbindung mit den Testprotokollen.

---

<sup>23</sup>Software Testing: A Guide to the Tmap(r) Approach: A Guide to the TMap Approach, S.16 f., Martin Pol, Ruud Teunissen, Erik Van Veenendaal, Verlag: Addison-Wesley Longman, Amsterdam; Auflage: illustrated edition (20. November 2001)

## 2.5. Unified Modeling Language

Die Unified Modeling Language (UML) ist eine standardisierte graphische Modellierungssprache im Bereich der Softwareentwicklung. Der Standard wird von der Object Management Group verwaltet und wurde auch von dieser geschaffen.<sup>24</sup> UML enthält verschiedene Notationstechniken um visuelle Modelle von softwareintensiven System zu erzeugen. UML selbst ist auch von der ISO standardisiert und zählt heutzutage zu einer der bedeutendsten Modellierungssprachen bei der Softwareentwicklung. Durch die Sprache wird nicht nur eine grafische Notation festgehalten sondern ebenfalls die Begriffe und die jeweiligen Beziehungen zwischen diesen. Somit bilden die Diagramme nur einen Teil dessen ab, was unter der UML zu verstehen ist. Die UML wird seit 1997 weiterentwickelt und ist seitdem in mehrere Versionen erschienen. Die in der UML verwendeten Modelle lassen sich in verschiedene Kategorien unterteilen, die wie folgt lauten:<sup>25</sup>

-Struktur-Diagramme

-Verhaltens-Diagramme

In diese Kategorien lassen sich wiederum die in UML verwendeten Diagramme einordnen:

-Strukturdiagramme:

- \* Klassendiagramm
- \* Kompositionsstrukturdiagramm
- \* Komponentendiagramm
- \* Verteilungsdiagramm
- \* Objektdiagramm
- \* Paketdiagramm
- \* Profildiagramm

-Verhaltensdiagramme:

- \* Aktivitätsdiagramm
- \* Anwendungsfalldiagramm
- \* Interaktionsübersichtsdiagramm

<sup>24</sup>Das UML Benutzerhandbuch. Aktuell zur Version 2.0, S. 20, Grady Booch, James Rumbaugh, Ivar Jacobson, Verlag: Addison-Wesley, München (2006)

<sup>25</sup>MDA: Effektives Softwareengineering mit UML2 und Eclipse, S. 447+464, Volker Gruhn, Daniel Pieper, Carsten Röttgers, Verlag: Springer, Berlin; Auflage: 1 (Juli 2006)

- \* Kommunikationsdiagramm
- \* Sequenzdiagramm
- \* Zeitverlaufsdiagramm
- \* Zustandsdiagramm

Im Folgenden soll nun auf die wichtigsten Diagramme eingegangen werden, welche für die Umsetzung des Projekts selbst verwendet wurden. Das erste verwendete Diagramm, ist das Anwendungsfalldiagramm (im Folgenden als Usecase-Diagramm benannt).<sup>26</sup> Dieses dient dazu einen Überblick über die Funktionen des Systems, aber auch über die beteiligten Personen (Akteure) zu erhalten. Das Usecase-Diagramm stellt keine Beschreibung der Abläufe dar, sondern die Beziehung zwischen Akteur und den jeweiligen Funktionen, die in diesem Fall als Anwendungsfall bezeichnet werden. Akteure können im Diagramm Anwender, Administratoren, aber auch Systeme selbst darstellen, welche von extern auf das System zugreifen. Im Diagramm selbst werden diese als 'Strichmännchen' dargestellt und haben jeweils einen Namen. In einem Usecase-Diagramm muss immer mindestens ein Akteur vorhanden sein. Anwendungsfälle hingegen werden als Ellipsen dargestellt und enthalten eine Beschreibung. Um beide unterschiedlichen Elemente in einer Gruppe zusammenzufassen, wird ein Rahmen um alle beteiligten Elemente gebildet. Dieser wird im Systemkontext genannt und bildet somit die Systemgrenze. Neben den normalen Assoziationen (z.B. Benutzer -> Drucken) besteht die Möglichkeit der Generalisierung. Das bedeutet, dass zwei spezifische Akteure oder Anwendungsfälle zu einem generellen zusammengefasst werden können. Ein Beispiel eines Usecase-Diagramms ist in Abbildung X zu sehen.

---

<sup>26</sup>Analyse und Design mit UML 2.3: Objektorientierte Softwareentwicklung, S. 245f. ,Bernd Oestereich, Stefan Bremer, Verlag: Oldenbourg; Auflage: 9., aktualisierte und erweiterte Auflage. (15. September 2009)

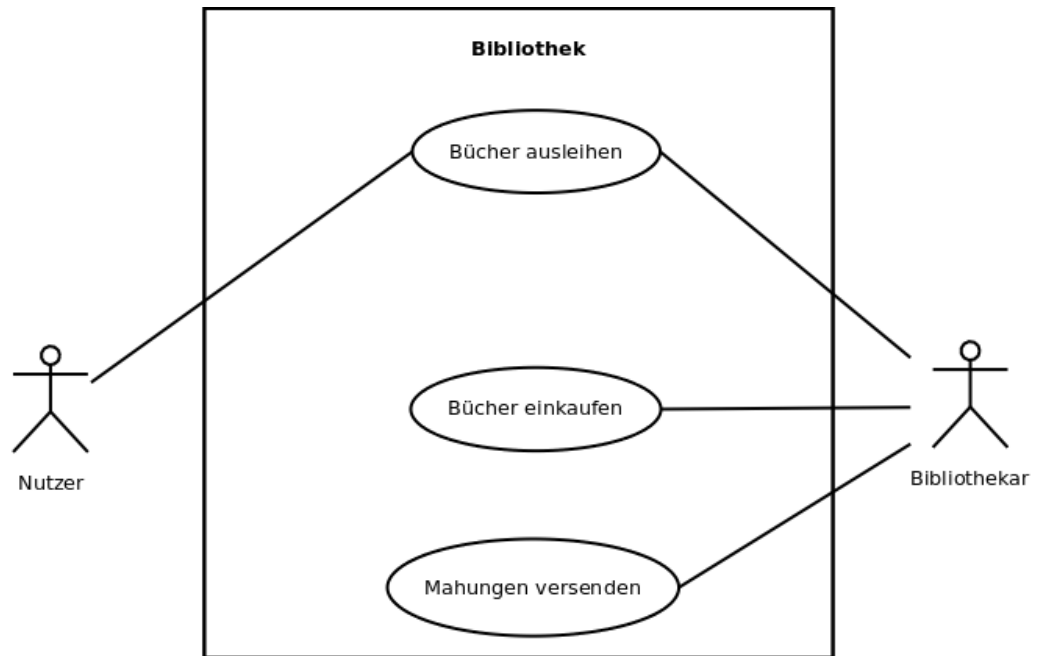


Abbildung 6.: Usecase-Diagramm

Ein weiteres wichtiges Diagramm stellt das Klassendiagramm dar. Es dient zur Beschreibung einzelner Klassen sowie deren Beziehungen untereinander. Klassen dienen zur Beschreibung der Objekte, welche von diesen instanziiert werden. Im Klassendiagramm wird eine Klasse als Rechteck dargestellt und neben dem Klassennamen enthält diese ein Bereich für die Attribute, sowie für die Methoden. Um die Sichtbarkeit der Attribute und Methoden darzustellen, werden verschiedene Symbole verwendet:

+ für public, unbeschränkter Zugriff

# für protected, Zugriff nur von der Klasse sowie von Unterklassen (geerbte Klassen)

- für private, nur innerhalb der Klasse selbst sichtbar

Ähnlich wie bei einem Usecase-Diagramm bietet sich beim Klassendiagramm die Möglichkeit einer Generalisierung. Beispielsweise sind die Klassen *PKW* und *LKW* Unterklassen von der Klasse *Fahrzeuge*. In Abbildung X ist ein Beispiel eines Klassendiagramms zu sehen.



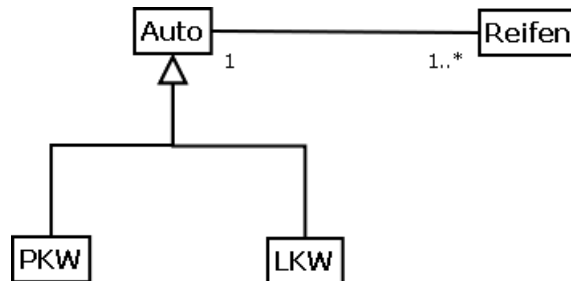


Abbildung 7.: Klassendiagramm

Ebenfalls relevant für die Umsetzung des Projektes sind die Aktivitätsdiagramme. Sie dienen dazu, Abläufe von Prozessen abzubilden.<sup>27</sup> Beim Aktivitätsdiagramm befinden sich die Elemente in abgerundeten Rechtecken, welche zusätzlich den Namen der Aktivität neben den Elementen enthalten. Den Start bzw. Endpunkt bilden jeweils ein gefüllter Kreis, wobei der Endpunkt nur teilweise gefüllt ist. Aktivitäten werden in abgerundeten Rechtecken aufgelistet und miteinander in Flussrichtung verbunden. Entscheidungsfälle werden durch eine Raute symbolisiert. Je nach Entscheidungsfall verläuft der Pfad bei 'Ja' weiter in Flussrichtung nach unten, oder bei einer Abweichung zur Seite ab. Zusätzlich bietet sich die Möglichkeit, Aktivitäten parallel ablaufen zu lassen. Dies kann durch Aktivitäten, die zwischen zwei Balken liegen, symbolisiert werden. Im Folgenden ist in Abbildung X ein beispielhaftes Aktivitätsdiagramm zu sehen.

<sup>27</sup>Analyse und Design mit UML 2.3: Objektorientierte Softwareentwicklung, S. 335f. ,Bernd Oestereich, Stefan Bremer, Verlag: Oldenbourg; Auflage: 9., aktualisierte und erweiterte Auflage. (15. September 2009)

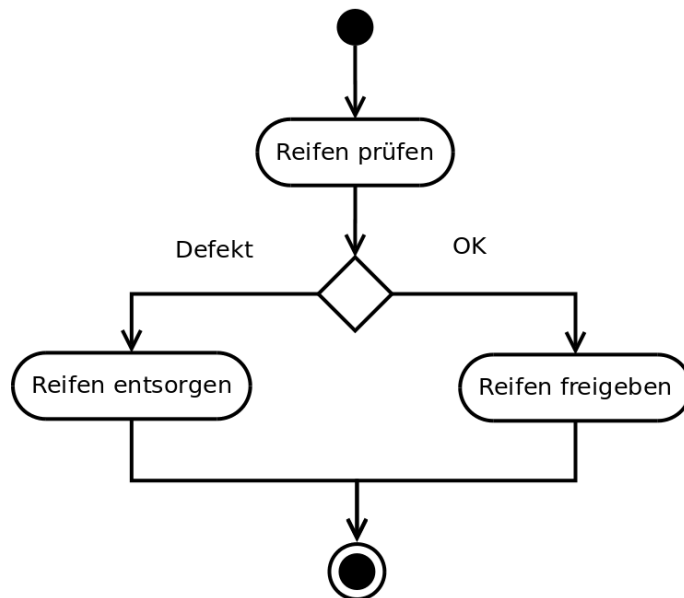


Abbildung 8.: Aktivitätsdiagramm

Für das bessere Verständnis der späteren Implementierung ist das Sequenzdiagramm ebenfalls von größerer Bedeutung.<sup>28</sup> Es dient dazu, einen Überblick über die Lebensdauer und Interaktion zwischen den einzelnen Klassen bzw. deren Objekte zu erhalten. Bei diesen wird nicht nur auf die bei den Klassendiagrammen gezeigte Beziehung, sondern auch auf den Nachrichtenaustausch zwischen den Objekten eingegangen. Bei einem Nachrichtenaufruf mit Antwort bzw. mit einer daraus folgenden Aktion, wird es notwendig, zu unterscheiden, welche Art von Kommunikation stattfindet. Hierbei gibt es die synchrone, als auch asynchrone Kommunikation. Bei der synchronen Kommunikation handelt es sich um einen Nachrichtenaustausch, bei dem das der Sender wartet, bis der Empfänger die Nachricht bearbeitet hat. Beispielsweise könnte ein Browser eine Webseite angefordert haben. Dieser muss nach dem Absenden seiner Anforderung abwarten bis er eine Antwort erhalten hat und kann erst im Anschluss die Webseite dem Benutzer darstellen. Bei der asynchronen Kommunikation handelt es sich um einen Austausch, welcher es nicht erfordert, dass ein Teilnehmer auf die Bestätigung des Anderen warten muss. Beispielsweise beim Versand einer Email. Hier muss der Sender weder warten bis der Empfänger online ist, noch muss er den Empfang der Email abwarten. Somit können Nachrichten gesendet werden, ohne dass für den Sender ein Ergebnis zur Laufzeit

<sup>28</sup> Analyse und Design mit UML 2.3: Objektorientierte Softwareentwicklung, S. 361f. ,Bernd Oestereich, Stefan Bremer, Verlag: Oldenbourg; Auflage: 9., aktualisierte und erweiterte Auflage. (15. September 2009)

erwartet wird. Eine Nachricht wird im Sequenzdiagramm durch Pfeile dargestellt. Synchrone Nachrichten werden mit gefüllten Pfeilspitzen, asynchrone Nachrichten mit offenen Pfeilspitzen gekennzeichnet. In der nachfolgenden Abbildung ist ein beispielhaftes Sequenzdiagramm zu sehen.

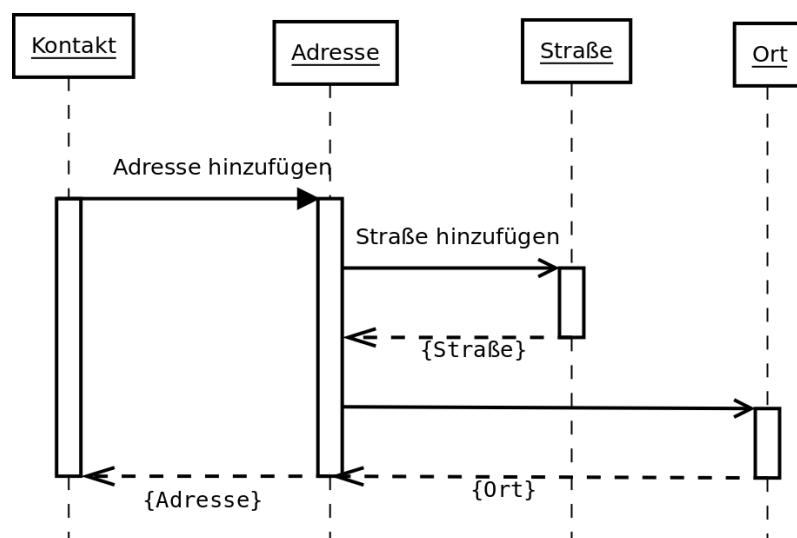


Abbildung 9.: Sequenzdiagramm

## 2.6. Media Access Control

Bei der Media-Access-Controll-Adresse (MAC-Adresse) handelt es sich um eine Adresse eines Netzwerkadapters, die zur eindeutigen Identifizierung in einem Netzwerk dient. MAC-Adressen werden in einer Vielzahl von Netzwerk Protokollen verwendet, unter anderem wird sie im Ethernet-Protokoll (IEEE 802.3), aber auch vielen anderen Netzwerktechnologien genutzt die unter den IEEE 802 Gruppen zu finden sind. Das MAC-Protokoll steuert die Adressierung auf Hardwareebene, sowie die Zugriffsart. Im ISO/OSI-Modell ist das MAC-Protokoll auf Schicht 2, der Sicherungsschicht, angesiedelt. Somit hat das MAC-Protokoll zwei Aufgaben. Zum Einen die Adressierung eines Gerätes und zum Anderen, wie dieses Gerät auf das Medium zugreift.<sup>29</sup> Zur eindeutigen Adressierung der

<sup>29</sup>Business Data Communications and Networking, S.119f. , Jerry Fitzgerald , Alan Dennis, Fitzgerald, Verlag: John Wiley & Sons; Auflage: 10. (19. Januar 2009)

Netzwerkadapter dient die sogenannte MAC-Adresse. Hierbei handelt es sich um einen 48-bit Wert, der einzigartig auf der ganzen Welt sein sollte.<sup>30</sup> Durch die Änderbarkeit der MAC-Adresse kann es jedoch vorkommen, dass dies nicht immer gegeben ist. Die Wahrscheinlichkeit eine doppelte MAC-Adresse im Netzwerk zu haben, ist jedoch extrem gering, da der Adressraum nach aktuellen Kalkulationen noch bis in das Jahr 2100 ausreicht.<sup>31</sup> Die ersten 24-bit kennzeichnen den Hersteller des Netzwerkadapters, die anschließenden 24-bit sind beliebig vom Hersteller vergebbar. Einige bestimmte MAC-Adressen können jedoch nicht reserviert werden, da es sich hierbei handelt um Broadcast-Adressen und Multicast-Adressen handelt. Eine MAC-Adresse kann wie folgt dargestellt werden:

01-23-45-67-89-ab oder 01:23:45:67:89:ab oder 0123.4567.89ab

Anhand der Herstellerkennung lassen sich wiederum Rückschlüsse auf das Netzwerkgerät ziehen. Die Hersteller, welche sich jeweils hinter der Herstellerkennung verbergen, sind öffentlich bei der IEEE einsehbar, z.B. per Internet.<sup>32</sup> Das MAC-Protokoll regelt ebenfalls den Zugriff auf das Transportmedium. Hierbei wird in zwei verschiedene Zugriffsarten unterteilt. Zum Einen den kontrollierten Zugriff und zum Anderen der konkurrierende Zugriff.

Beim kontrollierten Zugriff wird darauf geachtet, dass keine Kollision auftritt. Somit kommuniziert keines der Netzwerkgeräte gleichzeitig über einen Kanal, sondern es ist immer nur ein Gerät aktiv.

Beim konkurrierenden Zugriff hingegen darf jedes Gerät auf das Medium zugreifen, jedoch gibt es bestimmte Regeln, wenn eine Kollision auftritt. In diesen wird geregelt, in welcher Art und Weise die Kollisionen behandelt werden. In der Praxis gibt es unter anderem das Protokoll Carrier sense multiple access with collision detection (CSMA/CD). Dieses stellt bei einer Kollision durch ein Stör-Signal sicher, dass alle beteiligten Geräte die Kollision ebenfalls erkennen.<sup>33</sup> Zusätzlich versendet das Gerät das Paket nach einiger Zeit wiederum erneut, bis dieses ankommt oder die Anzahl der maximalen Versuche

<sup>30</sup>Designing Large Scale LANs, S.135 , Kevin Dooley, Verlag: O'Reilly Media; Auflage: 1 (November 2001)

<sup>31</sup><http://standards.ieee.org/develop/regauth/tut/eui.pdf>

<sup>32</sup><http://standards.ieee.org/develop/regauth/oui/public.html>

<sup>33</sup>Rechnernetze, S.157, Wolfgang P. Kowalk , Manfred Burke, Verlag: Teubner Verlag; Auflage: 1 (1994)

überschritten wurde. Die Wartezeit steigt exponentiell anhand der Versuche. Hierzu wird eine Zufallszahl aus dem Bereich 0 und  $(2^i)-1$  und diese mit einem festen Intervall (z.B.  $50\mu s$ ) multipliziert gewählt.<sup>34</sup> Wobei  $i$  für den  $i$ -ten Versuch steht.

## 2.7. Virtual Local Area Network

Bei einem VLAN (Virtual Local Area Network) handelt es sich um ein logisches Netzwerk innerhalb eines physikalischen Netzwerkes. Dieses logische Netz beinhaltet meist nur einen gewissen Teil des physikalischen Netzwerkes. VLANs können über einen oder mehrere Switches ausgedehnt werden und müssen sich nicht auf einen speziellen Port beziehen. Um ein Netzwerk durch VLANs in mehrere Teilnetze zu unterteilen, gibt es den Ansatz Ports bzw. die einzelnen Datenpakete jeweils VLANs zuzuweisen. Durch diese Zuweisung findet eine logische Trennung statt, da nur die Geräte in einem VLAN untereinander kommunizieren können. VLANs können über verschiedene Arten realisiert werden. VLANs lassen sich in portbasierte VLANs und tagged VLANs unterscheiden. Bei den portbasierten VLANs gehört ein Port je einem oder mehreren VLANs an.<sup>35</sup> Beim sogenannten tagged VLAN hingegen findet die VLAN-Zuordnung durch eine Kennzeichnung des Datenpaketes (im Ethernet-Frame) statt. Das markieren der Datenpakete kann sowohl von VLAN fähigen Endgeräten als auch von Switches geschehen. Jedoch muss sichergestellt werden, dass Geräte ohne VLAN-Unterstützung keine markierten Pakete erhalten.

Neben der Art der Zuweisung gibt es eine Unterscheidung zwischen statischen und dynamischen VLANs.<sup>36</sup> Bei einem statischen VLAN ist ein Port immer einem VLAN zugeordnet.

Das dynamische VLAN hingegen ist nicht portbasierend und wird je nach Inhalt der Datenpakete festgelegt. Die Zugehörigkeit zu einem VLAN kann auf Basis der Adressierung (MAC- oder IP-Adresse) geschehen oder aber auf Basis der Anwendung (TCP

---

<sup>34</sup>IEEE 802.3

<sup>35</sup>Computernetzwerke. Von den Grundlagen zur Funktion und Anwendung, S.131f., Rüdiger Schreiner, Hanser Fachbuch; Auflage: 3., überarb. Aufl. (21. April 2009)

<sup>36</sup>Cisco Networking Academy Program 3. und 4. Semester: Autorisiertes Kursmaterial zur Bildungsinitiative Networking, S. 292f., Cisco, , Verlag: Markt und Technik; Auflage: 3. A.

Port 80 - HTTP). Dadurch ergibt sich z.B. die Möglichkeit ein mobiles Endgerät (Notebook, Handscanner), das per WLAN an unterschiedliche Accesspoints und somit auch an verschiedenen Ports angeschlossen wird, immer dem gleichen VLAN zuzuordnen. Die Gründe für die Verwendung eines VLANs lassen sich in drei Punkte aufteilen. Zuerst macht eine Verwendung wie oben bereits erwähnt Sinn, wenn eine flexible Zuordnung eines Endgerätes immer zum selben VLAN gemacht werden muss.

Ein weiterer Grund für VLANs sind Performance-Aspekte. Neben der Priorisierung von speziellen Daten (z.B. VOIP), dient ein VLAN meist der Verkleinerung der Broadcast-Domänen. Durch die Verkleinerung der Domänen, wird ein Broadcast nicht über das gesamte Netzwerk hinweggeschickt.

Neben diesen Aspekten spielt die Sicherheit ebenfalls eine wichtige Rolle. Um zu verhindern, dass das Netzwerk abgehört wird, kann es sinnvoll sein, VLANs einzusetzen, da diese gegenüber Layer-2-Attacken architekturbedingt unempfindlich sind.

## 2.8. Simple Network Management Protocol

Unter dem Simple Network Management Protocol (SNMP) ist ein Netzwerkprotokoll zu verstehen, welches einem erlaubt, Netzwerkgeräte (z.B. Drucker, Router, Switches, Router) per Netzwerk zu überwachen und zu steuern.<sup>37</sup> Diese Abfragen werden von einem zentralen Punkt aus durchgeführt, dem sogenannten SNMP-Manager, welcher die Daten von den SNMP-Agenten (Netzwerkelementen) abrufen.<sup>38</sup>

Bei SNMP handelt es sich um ein Protokoll, welches sich auf der Schicht 7, die Anwendungsebene, des ISO/OSI-Schichtenmodells, ansiedeln lässt. Entwickelt wurde das Protokoll von der IETF und ist über diverse RFCs definiert. Durch die hohe Modularität funktioniert SNMP nicht nur über IP und sondern auch über IPX oder AppleTalk. Dies ist unter anderem auch ein Grund für die weite Verbreitung von SNMP, welches mittlerweile als Standard gilt. Die Funktionsweise von SNMP ist einfach gehalten, um eine größtmögliche Portabilität zu erreichen. Zum Einen gibt es die sogenannten 'Agenten' die Daten zur Verfügung stellen. Zum Anderen gibt es die sogenannten Manager, welche die Daten abfragen. Die Informationen der Agenten können von einem Manager über das

---

<sup>37</sup>vgl. Essential SNMP, S. 1

<sup>38</sup>vgl. Essential SNMP: Help for System and Network Administrators, , S. 3

SNMP-Protokoll abgefragt werden. Zusätzlich besteht die Möglichkeit, dass ein Agent eine Nachricht an einen Manager sendet, sofern ein vordefinierter Wert überschritten wurde. Neben dem Anfordern von Informationen kann der Manager auch Werte auf dem Agenten setzen. SNMP selbst enthält keine Vorschrift, welche Daten/Werte die Netzwerkkomponenten liefern, sondern gibt nur eine Baumstruktur vor. Die Art der Werte wird über die sogenannte Management Information Base (MIB) definiert, welche die einzelnen Knoten des Baums beschreibt. Diese MIBs sind teilweise über RFCs spezifiziert.<sup>39</sup> Zusätzlich gibt es herstellerspezifische MIBs z.B. von Cisco, die in einem speziellen Punkt im Baum hinterlegt werden können. Diese MIBs werden unter dem Object Identifier (OID) 1.3.6.1.4.1 (iso.org.dod.internet.private.enterprises) bei der IANA registriert.

Bei der Kommunikation untereinander werden verschiedene Paket-Typen verwendet.

## GET

Bei den GET-Paketen handelt es sich um jeweils unterschiedliche Arten der Anforderung die vom Manager an den Agent gesendet werden. Bei einem normalen GET-Paket wird ein einzelnes Attribut vom Agenten angefordert. Jedoch gibt es Abfragen, bei denen nicht im Voraus bekannt ist, wie viele Attribute abgefragt werden müssen. Beispielsweise der Status mehrerer Ports an einem Switch. Da dem SNMP-Manager jedoch keine Informationen vorliegen wie viele Ports der Switch hat, kann er nicht im Voraus die entsprechende Abfrage starten.

## GETNEXT

Um diese Problematik zu lösen gibt es den sogenannten GETNEXT-Befehl, der es ermöglicht anhand einer OID, die OID und den Wert des darauffolgenden Elementes zu erhalten.

Die Abfrage von einer Anzahl von  $n$  Ports erzeugt  $n+1$  Abfragen die vom Manager ausgehen. Bei einem 48 Port Switch müssten somit 49 Abfragen abgesetzt werden. Dies ist aber ineffektiv, da der Manager nur eine Informationsmenge erhalten möchte aber dazu eine Vielzahl an Anfragen durchführen muss.

---

<sup>39</sup>vgl. RFC 1213

## GETBULK

Aus diesem Grund wurde mit SNMP v2 der GETBULK Befehl eingeführt. Dieser ermöglicht es mehrere Werte mit einer Abfrage zu erhalten, die am Knoten im Baum hinterlegt sind.

## SET

Das SET-Paket dient zum Setzen spezieller Werte. So kann zum Beispiel darüber der Status des Ports von einem Switch geändert werden, oder es könnte eine Firewall konfiguriert werden.

## RESPONSE

Auf diese bisher genannten Pakete antwortet der Agent mit einem RESPONSE Paket, welcher die benötigten Werte oder eine Fehlermeldung enthält.

## TRAP

Sofern beim SNMP-Agent z.B. gewisse Grenzwerte hinterlegt wurden, kann dieser sich bei einer Überschreitung mittels eines Trap-Paketes beim Manager melden, ohne dass dieser die Information explizit abgefragt hat. Um möglichst wenig Netzwerklast zu erzeugen, kommuniziert SNMP über das UDP Protokoll, da es eine verbindungslose Kommunikation ermöglicht. Der Agent erhält die Anfragen auf Port 161, während der Manager auf Port 162 die Trap-Meldungen empfängt.

## 2.9. Cisco Discovery Protokoll

Das Cisco Discovery Protokoll (CDP) von Cisco ist ein proprietäres Protokoll, welches dazu dient, dass sich Cisco-Geräten untereinander mit Informationen austauschen kön-



nen. Die Cisco Geräte senden jeweils zur Multicast-Adresse<sup>40</sup> "01-00-0c-cc-cc-cc", um durch das Versenden eines einzigen Paketes alle Cisco Geräte im Netzwerk zu erfassen. Dies geschieht in einem Intervall von 60 Sekunden auf allen Netzwerk-Schnittstellen. Jedes der CDP-fähigen Geräte führt intern eine Tabelle mit Informationen über die Geräte, welche im Netzwerk gefunden wurden. Diese können per internen CDP oder per SNMP abgefragt werden. Bei jedem Empfang von CDP Daten, werden die internen Tabellen aktualisiert. Geräte die sich nicht mehr melden, werden nach einer bestimmten Zeit (standardmäßig nach 180 Sekunden) aus den Tabellen entfernt. Die Informationen welche übertragen werden sind einfach erweiterbar, da diese auf dem "Type-Length-Value" Format basieren. Das heißt in einer Nachricht wird zuerst der Typ des Attributs bestimmt (z.B. String, Zahl, Datum) danach die Zeichenlänge des Wertes und der Wert selbst.

Hersteller, wie HP, distanzieren sich zunehmend von diesem proprietären Protokoll und unterstützen das durch die IEEE spezifizierte offene Protokoll LLDP, welches Hersteller unabhängig ist und den selben Funktionsumfang beinhaltet.

---

<sup>40</sup>Designing Large Scale LANs, S.136 , Kevin Dooley, Verlag: O'Reilly Media; Auflage: 1 (November 2001)

## 3. Analyse der jetzigen Situation der Pirelli Deutschland GbmH

### 3.1. Situationsbeschreibung

Die bestehende Situation der PD spielt für die spätere Umsetzung eine große Rolle und muss somit im Näheren betrachtet werden. Zu Beginn des Projektes wurden bereits diverse Systeme zur Überwachung des Netzwerkes eingesetzt. Die Überwachung des Netzwerkes lässt sich bei näherer Betrachtung in folgende Punkte unterteilen:

Darstellung und Überwachung des Netzwerkequipments

Überwachung von Diensten/Servern

Überwachung des Netzwerkverkehrs

Überwachung von einzelnen Hosts im Netzwerk

Zur Überwachung der Stati der einzelnen Netzwerkgeräte, im Speziellen die Switches und Router, wird das Programm "OpenView Network Node Manager" von HP verwendet. Zurzeit wird das Programm verwendet um eine automatische Übersicht über die Relationen zwischen den Switches und Routern und eine grafische Übersicht über den Status der Switchs und Router zu haben zu erhalten. Über dieses Programm können auch Server und deren Dienste erfasst werden. Diese Funktion wird jedoch aktuell nicht mehr mit Daten gepflegt, da hier aktuell ein Transfer zum Nagios System stattfindet, aufgrund der Tatsache, dass die Wartung von OpenView ausgelaufen ist.

Für die Überwachung spezieller Hosts (in der Praxis meist Server) und Diensten, kommt das Open Source System Nagios zum Einsatz, welches ein hohes Maß an Flexibilität und Modularität bietet. Dieses Programm ermöglicht es, diverse Dienste im Netzwerk auf vorgegebenen Hosts abzufragen und diese anschließend zu visualisieren. Zusätzlich können verschiedene Warn- und Fehlerlevels für einen Service definiert werden, die im Falle einer Störung Warnungen an vordefinierte Personen verschicken. Dieses System ist

somit hauptsächlich für die Überwachung der Dienste zuständig.

Zur Untersuchung von Netzwerk-Traffic zwischen verschiedenen Netzwerken kommt das System MRTG zum Einsatz, welches hauptsächlich für das Anzeigen des Netzwerktraffics auf Routern ausgelegt ist. Dieses wird im Unternehmen hauptsächlich zum Erfassen der Bandbreitennutzung der Router bzw. der WAN-Verbindungen verwendet. Dies ist vor allem wichtig, um den aktuellen Traffic zwischen den einzelnen Standorten in Deutschland im Überblick zu behalten.

Abschließend gibt es noch ein System von Cisco, mit dem Namen "CiscoWorks", welches dazu dient, mit einem speziellen Programmteil die Hosts an den jeweiligen Switches zu erkennen und zu identifizieren. Hier werden nicht nur simple Zuordnungen zwischen MAC-Adresse des Hosts und des Switches, bzw. des Port des Switches hergestellt, sondern umfangreiche Informationen gesammelt, z.B. über die Geschwindigkeit des Anschlusses, die IP und der DNS-Hostname des Gerätes. Hinzu kommen Details, wie der Gerätetyp, welcher einem die Möglichkeit gibt, zu erkennen, ob es sich hierbei um ein Switch oder Router handelt.

Da es bei diesem System notwendig ist, regelmäßig die Lizenz zu verlängern, um auch weiterhin die neusten Router und Switches zu unterstützen, ist hier eine gewisse Abhängigkeit gegeben. Weil von CiscoWorks selbst nur ein kleiner Teil genutzt wird und somit die Lizenzkosten unverhältnismäßig zur Nutzung sind, wurde entschieden, dieses in Zukunft zu ersetzen.

Neben der eingesetzten Software, ist es auch wichtig ein Überblick über die Netzwerk-Architektur zu haben. Auf diese wird im nachfolgenden Abschnitt eingegangen. Das Netzwerk bei PD ist in verschiedenen Hierarchieebenen konzipiert. In der untersten Ebene stehen die Switches an denen die Hosts angeschlossen sind. Diese Ebene wird als Access-Layer bezeichnet. Über dieser Ebene wiederum befindet sich der Distribution-Layer. In diesem sind Switches zu finden, welche alle Access-Layer Switches verbinden und als Schnittstelle zu den Layer 3 Switches, den sogenannten Core-Switches, dienen. Die Core-Switches können zusätzlich anhand von OSI-Layer-3 Informationen Forwarding betreiben. Die komplette Anordnung ist in Abbildung X zu sehen.

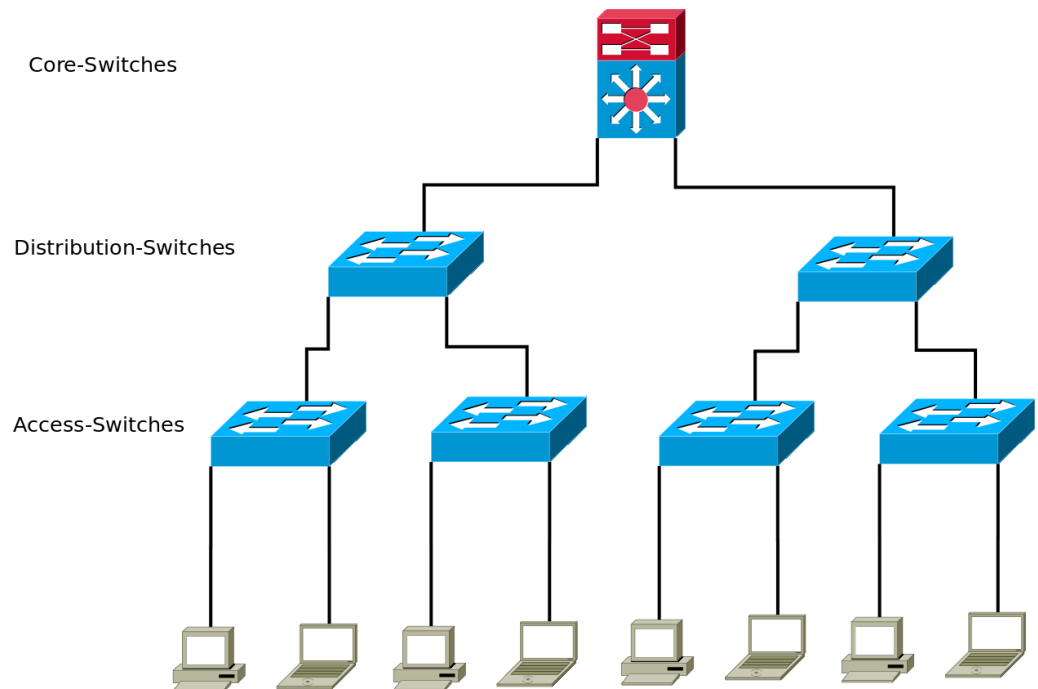


Abbildung 10.: Netzwerkarchitektur PD

### 3.2. Anforderungsdefinition

Um das bestehende System ersetzen zu können, muss zunächst untersucht werden, welche der bereits vorhandenen Funktionen im aktuell verwendeten System genutzt werden und welche in Zukunft benötigt werden. Im Anschluss an diese Untersuchung müssen auch weitere potentielle Aufgaben berücksichtigt werden, um eine Erweiterbarkeit des Systems zu garantieren. Damit das neue System allen Anforderungen gerecht wird, empfiehlt es sich zunächst, sich mit den Personen zusammen zu setzen, die das bestehende Programm zurzeit nutzen.

Hierbei wird nicht nur auf die aktuelle Nutzung eingegangen, sondern auch diese kritisch hinterfragt, ob eventuelle Anforderungen bereits von anderen Systemen erfüllt werden. Durch Gespräche mit den zuständigen Mitarbeitern wurden diverse Anforderungen erörtert.

Als wichtigster Punkt ist die Erkennung der Hosts, sowie der zugehörigen Ports an den Switches zu sehen. Diese ist notwendig um den Ort eines Hosts im Netzwerk festzustellen.

Praktische Anwendung hat dies, wenn zum Beispiel ein Host aufgefunden werden muss, wenn der zugehörige Mietschein des Computers ausläuft und dieser zurückgegeben werden muss. Hier ist es nicht selten der Fall, dass der Computer mit seinem dazugehörigen Besitzer bereits die damalige Abteilung verlassen hat und dieser somit nicht mehr auffindbar ist. Eine weitere Anwendungsmöglichkeit ist, dass der Helpdesk wissen möchte, wenn es Probleme mit einem Computer gibt, ob die Netzwerk-Hardware bis zum Host einwandfrei funktioniert. Hierfür muss man nicht nur wissen, an welchem Ort der Host sich befindet, sondern auch die dazugehörigen Ports. Da die Ports als auch die Switches ausgelesen werden, kann man somit zeitnah eine Rückmeldung geben, dass der Switch, sowie dessen Port einwandfrei funktionieren, oder falls dies nicht der Fall ist, den Fehler besser lokalisieren. Somit ist selbst bei keinerlei Kenntnis des Hosts außer der DNS oder der IP, eine direkte Lokalisierung und Bewertung des Status möglich.

Ein weiterer wichtiger Aspekt ist die Möglichkeit genau herauszufinden, welche Hosts hinter einem Port angeschlossen sind. In der Praxis ist dies wichtig, da es vorkommen kann, dass unerwünschte Netzwerkgeräte angeschlossen wurden. Normalerweise befinden sich im Unternehmen an einem Port eines Switches der untersten Ebene maximal zwei Hosts. Der Großteil der Ports hat nur einen Host zum Beispiel den Computer eines Mitarbeiters oder ein Netzwerkgerät. In einem Teil der Fälle ist dieser Computer nicht direkt an den Port des Switches angeschlossen, sondern wird über den Port eines VOIP-Telefons durchgeschleift, welches dann an den Switch angeschlossen ist. In diesem Fall sind auf dem Port zwei Hosts zu sehen. Dies ist der Soll Zustand. Jedoch kann es vorkommen, dass Mitarbeiter im Werk ohne Erlaubnis fremde Netzwerkgeräte anbringen, sei es ein Hub, ein eigener WLAN-Router oder andere Geräte. Da diese Geräte die Sicherheit des Netzwerkes beeinflussen, müssen diese identifiziert und gleichzeitig lokalisiert werden können um eine Entfernung zu ermöglichen. Hinzu kommt der Punkt einer möglichen Historie, wann welcher Host an welchem Port bzw. Switch angeschlossen war. Hierbei soll erkennbar sein, an welchem Port der Host war, jedoch ist es nicht notwendig zu wissen wie oft dieser dort angeschlossen war. Daraus lassen sich zwei Regeln definieren. Erstens, sofern ein Host an ein Port angeschlossen wird, so entsteht ein Datensatz. Die zweite Regel besagt, dass wenn ein Host an einen Port angeschlossen wird, an dem er bereits zuvor angeschlossen war, so wird der alte Datensatz für diese Host-Port-Kombination nur aktualisiert. In der nachfolgenden Abbildung X wird dies durch die beiden Switches S1 und S2 dargestellt. Bei den angeschlossenen Hosts handelt es sich jeweils um das

Notebook N1, das an jedem Port des jeweiligen Switches ein Datenbank-Eintrag erzeugt.

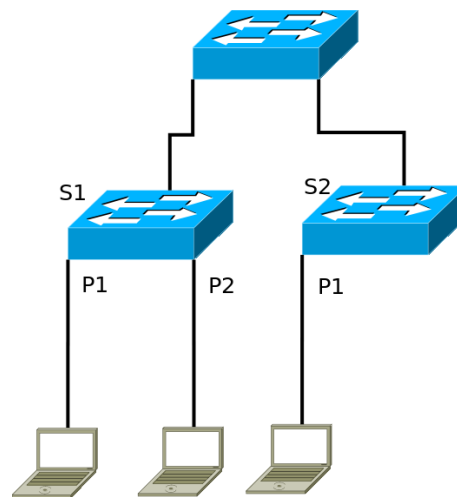


Abbildung 11.: Grafik mit Switches, Ports und Host

Für das Auslesen der Switches soll das Netzwerkprotokoll SNMP verwendet werden, da eine Abfrage per Telnet die Switches stark belastet und daher es passieren kann, dass der Betrieb eingeschränkt wird. Hinzu kommt, dass ein Auslesen per Telnet sehr langsam ist und die Ausgabe selbst nicht standardisiert ist und somit sehr viele Einzelfälle behandelt werden müssten.

Im Hinblick auf die Geschwindigkeit des Auslese-Prozesses wurde vorausgesetzt, dass das Sammeln der Daten mindestens vier Mal am Tag möglich sein muss und sofern möglich sollte eine Auslesezeit unter 60 Minuten angestrebt werden, da es sich hierbei um einen alten Wert handelt, welcher bei früheren Auslesevorgängen erreicht wurde.

Eine weitere Anforderung die als optional angesehen wird, ist die Möglichkeit die Verbindungen zwischen den Switches und Routern zu erfassen. Das heißt, es soll eine Topologie anhand der Daten erstellt werden können.

Zu den optionalen Anforderungen gehört die Möglichkeit einer Benutzerzuordnung zu den jeweiligen Hosts. Die Zuordnung soll über das bestehende Active Directory passieren und somit die Möglichkeit bieten den entsprechenden Nutzer bei Problemen direkt anrufen zu können.

Kombiniert man diese Anforderungen, so bildet sich eine weitere Anwendungsmöglichkeit und zwar kann es in der Praxis vorkommen, dass ein Switch gewechselt und somit abgeschaltet werden muss. Hierfür kann man nun im Voraus kontrollieren, welche Hosts an diesen angeschlossen sind und welche Nutzer somit im Voraus informiert werden müssen.

Neben den Anforderungen zu den Anwendungsmöglichkeiten wurden auch Anforderungen an das Programm selbst gestellt. Zu diesen gehört die Betriebssystemunabhängigkeit. Das Programm muss sowohl auf Windows-, als auch auf dem Linux-Betriebssystem funktionieren. Sofern Interpretersprachen oder ähnliches verwendet werden, darf keine Versionsabhängigkeit bestehen. Das bedeutet für das Programm, es muss z.B. sowohl unter Java 1.6.0.10 als auch unter 1.6.0.33 funktionieren.

Als bestehende Datenbank wurde Oracle vorgegeben, da hierfür bereits Lizenzen existieren und dieses bereits auf den entsprechenden Servern läuft und kein weiteres DBMS installiert und gepflegt werden muss.

### 3.3. Anforderungsanalyse

Um die Anforderungen bewerten zu können, muss man sich zunächst mit den Technologien vertraut machen und die technischen Möglichkeiten mit den Anforderungen abgleichen.

Zunächst einmal muss untersucht werden, ob die im Unternehmen befindlichen Switches auch das SNMP-Protokoll soweit unterstützen, um das Auslesen zu ermöglichen. Dies wurde überprüft und bei vereinzelten Switches musste die Konfiguration angepasst werden, da hier teilweise SNMP deaktiviert oder nur für spezielle Hosts zugelassen war.

Im nächsten Schritt galt es zu überprüfen, ob die benötigten Informationen per SNMP überhaupt abfragbar sind. Problematisch hierbei ist vor allem, dass Hersteller zwar eigene MIBs bei SNMP verwenden, über diese lassen sich jedoch nicht die gleichen Informationen abfragen, wie über die proprietären Protokolle der Hersteller. Bei der Überprüfung der Zuordnung zwischen MAC-Adresse der angeschlossenen Ports wurde festgestellt, dass

diese zwar möglich ist, aber eine sehr spezielle Vorgehensweise voraussetzt.<sup>41</sup>

Problematisch ist vorallem die genaue Zuordnung der Hosts an den korrekten Switch. So kann es sein, dass die MAC-Adresse eines Hosts sowohl auf einem Access-Switch als auch auf einem Distribution-Switch zu finden ist. Hierfür muss eine eindeutige Identifizierung stattfinden, welcher Switch welche Hierarchieebene angehört. Diese Informationen müssten manuell gepflegt werden oder anhand einer automatischen Hierachieerkennung erhalten werden.

Die Anforderung alle Hosts hinter einem Port zu finden, stellt kein Problem dar, da dies von allen Switches per SNMP unterstützt wird.

Eine Historie, die es ermöglicht, nur neue Kombinationen aus MAC-Adresse des Hosts und des zugehörigen Ports als neuen Eintrag zu sehen ist, ohne Probleme möglich und kann durch die demensptrechende Erstellung der Datenbank sichergestellt werden.

Neben den Anforderungen die sich auf die Systemstruktur beziehen, gibt es auch solche, die sich auf den Ausleseprozess konzentrieren. In diesem Zusammenhang wurde die Auslesegeschwindigkeit und die Last auf den Switches angeführt. Aussagen über die Auslesegeschwindigkeit können leider im Voraus nicht getroffen werden, jedoch ist anzunehmen, dass die Geschwindigkeit über der des manuellen Auslesens per Telnet liegen muss, da SNMP nicht verbindungsorientiert ist und somit die Endgeräte weniger belastet werden und der Overhead reduziert ist. Genauere Aussagen über die Geschwindigkeit lassen sich erst durch Benchmarks machen, welche in Kapitel [Kapitel X] durchgeführt werden.

Um mögliche Verbindungen zwischen den Switches und Routern zu erfassen, muss es möglich sein, auf einem Port einen Switch zu identifizieren, was per SNMP in Verbindung mit STP und CDP möglich ist.

Eine Zuordnung der einzelnen Nutzer lässt sich über das Active Directiry per LDAP bewerkstelligen, hierfür muss aber ein spezieller Benutzer eingerichtet werden, der vollständige Leserechte für alle Hosts im Netzwerk besitzt.

---

<sup>41</sup>vgl. SNMP Cisco PDF



Die Betriebssystemunabhängigkeit ist mit einer Vielzahl von Interpreter-Sprachen (Perl, Python, ...) oder einer Plattform unabhängigen Sprache (Java, C Sharp/Mono) gegeben.

Da für alle gängigen Sprachen diverse Datenbankverbindungen angeboten werden, gibt es bei der Wahl des DBMS auf Oracle keine Probleme.

Insgesamt betrachtet gibt es von der technischen Seite zwar gewisse Schwierigkeiten bei der Implementierung, diese verhindern aber nicht eine Umsetzung des Projektes.

Ein weiterer wichtiger Punkt sind auch die Ressourcen, da für die komplette Implementierung nur 11 Wochen zur Verfügung stehen. Für eine Implementierung eines Systems mit diesem Umfang und einer ausreichenden Testphase bei nur einem Entwickler ist dies ein zu geringer Zeitraum. Da vor der Implementierung und den Tests zuerst einmal der Entwurf und gewisse Design Entscheidungen stehen müssen, wird es hier zu Engpässen kommen, welche nur durch Kompromisse im Hinblick auf die Implementierung und den Tests gelöst werden können.

Da ein Großteil der Zeit für das Auslesen der Daten aufgebracht werden muss, wird die Zeit für die Umsetzung der Weboberfläche deutlich geringer angesetzt, wie die für eine umfassende Umsetzung notwendig wäre. Hier muss dann im Laufe des Projektes zwischen einzelnen Funktionen der Weboberfläche abgewägt werden.

Im Hinblick auf die Kosten entstehen bei der Umsetzung keine weitere Kosten bzw. bei der Wahl eines existierenden Open Source-Systems. Ziel des Projektes ist das bestehende kommerzielle Produkt zu ersetzen und somit eine monetäre Einsparung, aber auch der Anforderung einer gewissen Modularität gerecht zu werden.

### 3.4. Betrachtung bereits existierender Lösungen

Betrachtet man die bereits existierenden Lösungen zum Überwachen von Switches und Hosts, so lassen sich einige Produkte finden, die kommerzieller Natur als auch Open Source sind. Bei HP Openview handelt es sich um eine kommerziell Lösung von HP, die dazu dient eine komplette Netzwerk und Systemmanagement-Lösung anzubieten.

Diese Software-Umgebung besteht nicht nur aus HP eigenen Modulen sondern auch von verschiedenen Fremdherstellern. Die wichtigste Komponente des HP Openview für die Netzwerküberwachung stellt der sogenannte 'HP OpenView Network Node Manager' dar. Dieser ermöglicht es, neben dem Überblick über die Netzwergeräte und deren Stati zusätzlich detaillierte Informationen zu den einzelnen Geräten anzuzeigen. Diese Daten können bei Bedarf als Histogramme angezeigt werden, z.B. für den aktuellen Daten-Durchsatz an. Ebenso bietet sich die Möglichkeit einer Art Übersichtskarte die automatisch anhand der vorhandenen Netzwergeräte generiert wird. Die notwendigen Informationen hierfür werden per SNMP und CDP ausgelesen. Zusätzlich wird zur Statusüberprüfung auf Funktionen wie ein Ping zurückgegriffen

Eine weiteres kommerzielles System ist CiscoWorks von Cisco. Dieses bietet unter anderem ein spezielles Usertracking-Modul, welches einem ermöglicht, alle Hosts in einem Netzwerk zu identifizieren, aber auch die dazugehörigen Switches, an die jeweils die Geräte angeschlossen sind anzuzeigen. Diese Informationen können teilweise auch per CDP (Cisco Discovery Protokoll) ausgelesen werden. Über die per CDP bereitgestellten Daten wird auch ein automatisches sequentielles Auslesen aller Switches im Netzwerk ermöglicht, ohne zuvor diese zu kennen, jedoch müssen die notwendigen Zugangsdaten trotzdem zur Verfügung gestellt werden.

Das Open Source Programm Nagios dient vor allem zum Überwachen von verschiedenen Geräten im Netzwerk, sowie deren Dienste.<sup>42</sup> Hierbei arbeitet es ebenfalls per SNMP, jedoch ist es aber nicht auf dieses allein angewiesen. Sofern eingerichtet, kann es auch einfach per TCP oder UDP die Dienste überprüfen. Es können aber auch Remote Skripts per SSH aufgerufen werden. Jedoch ist hierbei zu beachten, dass Nagios nur das Überwachen von bekannten und zuvor konfigurierten Hosts und Diensten unterstützt und somit für die gewünschte Anforderung nicht in Frage kommt.

Ein System, welches die Anforderungen teilweise erfüllt, ist Tirthi. Dieses Programm wird vom Fraunhofer Institut eingesetzt im Kompetenzzentrum Netzwerkmanagement und ist Open Source. Es unterstützt das Auslesen von diversen Cisco Switches sowie deren jeweiligen Hosts. Zusätzlich liest Tirthi eine Großzahl an Informationen über die Switches sowie das bestehende VLAN aus. Funktionen die Tirthi nicht bietet, sind ein

---

<sup>42</sup>Nagios: System and Network Monitoring, S. 66+96, Wolfgang Barth, No Starch Press; Auflage: 2nd Edition. (7. Oktober 2008)

paralleles Auslesen der Switches und es verfügt auch nicht über eine Historie, wenn ein Port an einem Switch getauscht werden sollte.<sup>43</sup>

### 3.5. Entscheidung zur eigenen Programmierung

Vergleicht man alle aufgeführten Systeme, so lässt sich feststellen, dass keine der genannten Lösungen den Anforderungen entspricht. Zwar bietet Tirtih einen Teil der gewünschten Funktionen, jedoch müsste dies erst auf die speziellen Bedürfnisse angepasst werden, die das Unternehmen benötigt. Aufgrund dieser Tatsache empfiehlt sich eine eigene Implementierung, da diese die Möglichkeit gibt, besonders auf die Anforderungen einzugehen und keine Einarbeitungszeit in eine fremde System-Architektur notwendig macht. Selbstverständlich sind mit einer eigenen Implementierung gewisse Risiken verbunden, darunter auch der knappe Zeitrahmen. In der Anforderungsanalyse wurden diese aber bereits abgewegt und eine Machbarkeit des Systems festgestellt.

---

<sup>43</sup>[http://www.gambitcomm.com/site/products/mgmt\\_apps/interopnet98.htm](http://www.gambitcomm.com/site/products/mgmt_apps/interopnet98.htm) [http://www.fernuni-hagen.de/BWLPIT/LADV\\_/PDF\\_Dateien/Hinweise\\_Anfertigung\\_wiss\\_Arbeiten.PDF](http://www.fernuni-hagen.de/BWLPIT/LADV_/PDF_Dateien/Hinweise_Anfertigung_wiss_Arbeiten.PDF)

## 4. Entwurf und Implementierung

### 4.1. Entwurf

#### 4.1.1. Usecases

Zu Beginn einer Systementwicklung müssen die Usecases und die beteiligten Akteure identifiziert werden. Hierfür wird zu Beginn überlegt, welche Personen Zugang zum System haben und welche Aufgaben diese am System erfüllen werden. Für den Entwurf wird nicht nur die Anforderungsdefinition zu Rate gezogen, sondern der Verfasser bespricht die notwendigen Funktionalitäten, die das Programm später beinhalten soll, mit den Mitarbeitern selbst. Dadurch kommen meist weitere Punkte auf, die zuvor nicht angesprochen wurden, jedoch wichtig sind für den späteren Entwurf des Systems, sowie die Umsetzung. In Abbildung X ist das Usecase-Diagramm für das System zu sehen.

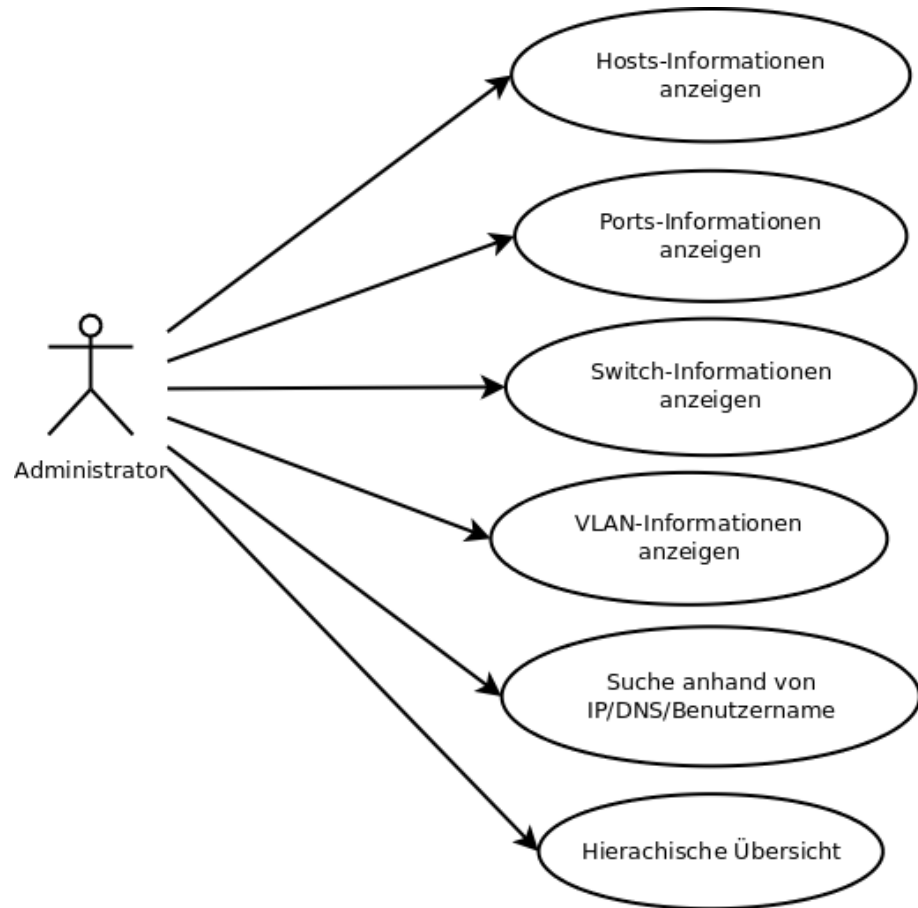


Abbildung 12.: Usecasediagramm

Zum einen ist festzustellen, dass es nur einen Akteur gibt. Das liegt daran, dass keine Einschränkung vorgesehen ist für die aktiven Nutzer des Systems. Bei der Implementierung wird jedoch eine mögliche spätere Einführung von verschiedenen Benutzerrechten bedacht und diese Funktionalität als Möglichkeit offen gelassen. Zur Einfachheit halber wird der Netzwerkdaminstrator im Laufe des Textes als Benutzer beschrieben. Die erste Sicht, die dem Nutzer zur Verfügung stehen muss, ist eine Übersicht über alle Switches, hier muss neben der Switch IP auch der per SNMP ausgelesene Alias stehen, aber auch die vorhandene IOS-Version und die Seriennummer des Switches.

In einer weiteren Sicht soll eine Übersicht auf die Ports eines Switches geben werden. Neben der MAC-Adresse des Ports, sowie des Port Namens (z.B. Fa0/1) sind weitere Informationen sichtbar, unter Anderem der Status (Up/Down), Geschwindigkeit des Ports, Duplexmodus, sofern zugewiesen die VLAN ID, aber auch Informationen, ob es

sich hierbei um ein Uplinkport handelt. Zusätzlich wird eine Sicht benötigt, die Informationen über den angeschlossenen Host anzeigt. Hierunter fallen Informationen, wie MAC-Adresse, IP, DNS, sowie der zuletzt angemeldete Benutzer an einer Workstation. Zusätzlich sollen aber auch Informationen angezeigt werden, welche in Verbindung mit dem Port in Verbindung stehen. Diese werden im Zuge der Historie zusätzlich beim Host gespeichert. Sofern es sich um ein CDP-fähiges Gerät handeln sollte, werden diese zusätzlichen Informationen ebenfalls angezeigt. Neben den Sichten zu den Geräten wird es auch eine Übersicht der VLANs geben, welche es erlaubt, eine Verbindung zwischen der VLAN ID und des jeweiligen VLANs herzustellen. Somit kann nicht nur der Alias sondern auch die entsprechende VTP-Domäne abgelesen werden.

Der Benutzer soll weiterhin auf jeder dieser Sichten eine Möglichkeit zur Suche eines Hosts/Switches haben, wie auch eine Option zur Sortierung der Einträge. Neben diesen grundlegenden Funktionen wurde empfohlen, zusätzlich eine Art hierarchische Übersicht zu ermöglichen. Das heißt, der Nutzer kann über die Switchübersicht einen Switch selektieren und erhält dann die Übersicht der dort verfügbaren Ports. Wählt er nun einen Port auf diesem Switch aus, so erhält er alle Hosts die an diesen Port angeschlossen sind. Sofern es sich dabei um mehrere Hosts handelt, wählt er wiederum einen aus und landet schließlich in der Übersicht über den Host mit den detaillierten Informationen.

Als optionaler Usecase wird das Anzeigen einer Art Dashboard gesehen. In diesem werden Informationen zusammen gefasst, die nicht direkt erkenntlich über die einzelnen Sichten sind. Beispielsweise könnte man hier Informationen anzeigen lassen, wie z.B. Anzahl freier Ports, Switch mit den meisten freien Ports, Switchs mit der höchsten Auslastung, um nur eine Liste der potentiellen Informationen zu nennen.

### 4.1.2. Klassendiagramme

Für die Umsetzung des Projektes wurde eine objektorientierte Programmiersprache gewählt, welche es erfordert Klassen zu verwenden. Diese dienen dazu, Objekte abzuleiten und die jeweiligen Methoden von diesen zu nutzen. Um das System zu entwerfen wurden 25 Klassen verwendet, da eine möglichst hohe Abstraktion erwünscht war. Zusätzlich muss auch sichergestellt werden, dass die Beziehungen zwischen den Klassen modular sind, damit ein einfacher Zugriff und eine Austauschbarkeit gegeben ist. Um ein Ein-

blick in die Struktur des Projektes zu erhalten wird in diesem Kapitel auf die wichtigsten Klassen des Projektes eingegangen. Die Abbildungen der Klassen befindet sich aus Platzgründen im Anhang.

Das Ausleseprogramm wird als Objekt der Klasse *SNMPTrack* instanziiert, welche auch den Namen des Auslese-Programms repräsentiert. Hierbei handelt es sich um die Ausgangsklasse, welche die Steuerung des Programms beinhaltet. Von dieser Klasse aus, wird, wie in Kapitel X beschrieben, ein Objekt der Klasse *SwitchListe* erzeugt. Dieses stellt alle Switches inklusive deren Passwörter in einer Liste zur Verfügung. Das Einlesen dieser Daten erfolgt anhand einer existierenden XML-Datei oder einer bestehenden Datenbank die in der Konfiguration hinterlegt wurde. Somit wird sichergestellt, dass für das Auslesen im Hauptteil des Programms nur ein Befehl benötigt wird und die Logik separat liegt.

Neben den Zugangsdaten für die Datenbank werden über die Klasse *SNMPConfig* weitere Informationen zur Verfügung gestellt. Sie liest anhand der XML-Konfigurationsdatei aus, welche Anzahl von Threads gleichzeitig maximal aktiv sein darf und enthält auch Informationen über das Debug-Level, welches den Detailgrad der Meldungen reguliert. Durch diese Klasse stehen die Informationen in allen anderen Klassen und Methoden zur Verfügung.

Die nächste wichtige Klasse ist die *SNMPHandler*, welche alle SNMP Abfragen verwaltet. Diese Klasse dient dazu, den Abruf eines SNMP Wertes oder sofern notwendig eines ganzen Unterbaums durchzuführen. Hierfür stellt diese eine Methode bereit, die anhand der gewünschten OID, der IP des SNMP-Agenten, sowie des notwendigen Community-String, die benötigten Informationen zurückliefert. Die Rückgabewerte werden in einem speziellen Format zurückgeliefert, um eine spätere Bearbeitung zu vereinfachen. Zum besseren Verständnis dient das nachfolgende Beispiel:

<OID>!<Wert>

Mit reellen Daten gefüllt, könnte ein Rückgabewert wie folgt sein:

1.3.6.1.2.1.1.1!Linux WRT54G 2.4.20 2 Thu Dec 9

Durch das Zusammensetzen mit ! kann eine einfachere Verarbeitung stattfinden bzw. der Wert auch manuell besser gelesen werden. Für das Auslesen eines Switches wird jeweils ein Objekt der Klasse *SwitchWorkerThread* erzeugt. Hierbei handelt es sich jeweils um einen separaten Thread, dem jeweils das *Switch*-Objekt aus der Liste des *Switch-Liste* Objekts übergeben wird. Der Thread ruft anschließend die Methode *refresh()* der Klasse *Switch* auf. Diese Methode beginnt daraufhin alle relevanten Informationen eines Switches über die *SNMPHandler* Klasse, wie in Kapitel X beschrieben, auszulesen. Hierzu verwendet die Methode eine programminterne OID-Bibliothek. Diese wird über die Klasse *OIDL* bereitgestellt. Im Detail enthält die Klasse verschiedene Strings, deren Variablenname dem Namen der OIDS im RFC X übereinstimmt.

Somit kann auf einfache Art und Weise ein übersichtlicher Befehl in den notwendige String übersetzt werden, wie im folgenden Beispiel:

```
OIDL.vtpVlanName  
den String:  
1.3.6.1.4.1.9.9.46.1.3.1.1.4
```

Dies ist vor allem notwendig um eine Übersicht über die verwendeten OID zu behalten bzw. erkennen zu können welcher Wert hinter der jeweiligen OID steckt. Der nachfolgende Quellcode-Abschnitt verdeutlicht die Wichtigkeit der Aliase:

```
swMACs=SNMPHandler.getOIDWalknonBulk(snmpp, OIDL.ifPhysAddress, sIP, sReadcommunity);  
swStatus=SNMPHandler.getOIDWalk(snmpp, OIDL.ifOperStatus, sIP, sReadcommunity);  
swVLANs=SNMPHandler.getOIDWalk(snmpp, OIDL.vtpVlanState, sIP, sReadcommunity);  
swVLANPorts=SNMPHandler.getOIDWalk(snmpp, OIDL.vmlan, sIP, sReadcommunity);  
swPortname=SNMPHandler.getOIDWalknonBulk(snmpp, OIDL.ifName, sIP, sReadcommunity);  
swPortalias=SNMPHandler.getOIDWalk(snmpp, OIDL.ifAlias, sIP, sReadcommunity);
```

Abbildung 13.: Codebeispiel

Neben den zuvor genannten Klassen, gibt auch eine Menge an Klassen, die ausschließlich als Schnittstelle zu anderen Datenquellen dienen. Hierzu gehört die Klasse *Nagios*, die es ermöglicht Switches, inklusive Zugangsdaten und deren Gruppenzugehörigkeit, anhand einer existierenden Datenbank mit Daten eines Nagios-Systems auszulesen.



Eine weitere Schnittstelle stellt die Klasse *LDAP* dar. Diese ermöglicht das Auslesen der Active-Directories der Windows Domäne. Die Klasse *DNSHelper* ermöglicht es den DNS Namen anhand einer IP zu finden, in dem jeweils der DNS-Server angesprochen wird.

Für die Kommunikation mit der Datenbank wurde eine spezielle Klasse *DataManagerOracleMulti* erstellt, die es ermöglicht SQL-Befehle die keine Rückgabe erwarten in Stapel auszuführen. Dies bedeutet, dass die Datenbank nicht jeden Befehl einzeln erhält sondern eine große Anzahl (>100 Stück) auf einmal und somit weniger Overhead für den Verbindungsaufbau entsteht.

Neben der Vielzahl an Klassen, die einer Vereinfachung des Programmablaufs dienen, gibt es trotzdem Objekte welche auch die Realität abbilden. Dazu gehören die Klassen *Switch* (welche bereits beschrieben wurde), *Port*, sowie *Host*. Diese bieten jeweils Methoden an, welche von sich selbst die passenden SQL-Befehle anhand der enthaltenen Daten generieren. Dies macht es einfacher die Objekte zu serialisieren. Das heißt, die Objekte im Programm wiederum abzubilden auf eine Textuale-Ebene, um sie in die Datenbank übertragen zu können.

### 4.1.3. Sequenzdiagramme

Um die Beziehungen der einzelnen Klassen des Programms zu visualisieren, reicht es nicht aus, die Klassendiagramme zu zeichnen, sondern es müssen auch Sequenzdiagramme erstellt werden. Im Folgenden soll auf die Beziehungen ausgehend von der Hauptklasse des Programms eingegangen werden. Zu Beginn des Programms erzeugt die Klasse *SNMPTrack* ein Objekt der Klasse *SwitchListe*. Bei diesem Objekt wird die Methode *getSwitches()* aufgerufen. Diese Funktion liefert dann eine Liste aller Switches, die später anhand einer vorgegebenen Datenquelle ausgelesen werden müssen. Zum aktuellen Zeitpunkt ist dies eine XML-Datei, in naher Zukunft werden diese Daten anhand des existierenden Nagios Systems ausgelesen. Im Anschluss wird ein Objekt der Klasse *SwitchWorkerThread* erstellt, um einen separaten Thread zu starten, der unabhängig und nebenläufig arbeiten kann. Dieser Thread nimmt das übergebene Switch Objekt

und ruft bei diesem die `refresh()` Methode auf, welche wiederum für das Einlesen aller Switchinformationen zuständig ist. Dieser Vorgang ist in Abbildung X beschrieben.

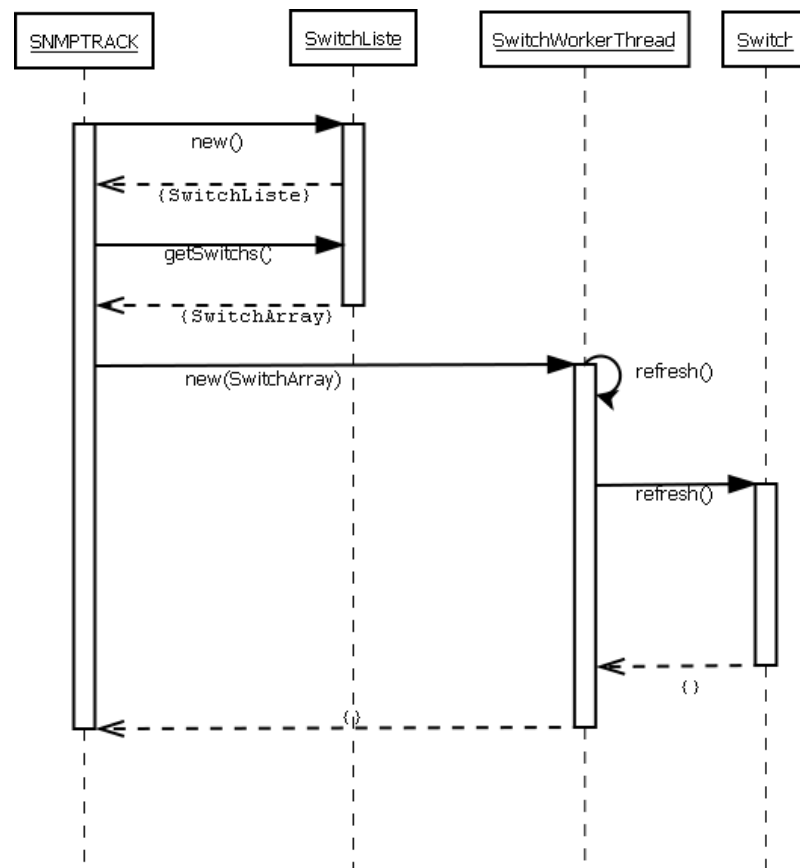


Abbildung 14.: Sequenzdiagramm 1

Während des `refresh()` Methoden-Aufrufs des `Switch` Objektes werden verschiedene andere Objekte erzeugt. Zu Beginn wird die Klasse `SNMPHandler` verwendet, welche verschiedene Methoden zur SNMP-Kommunikation zur Verfügung stellt. Über diese werden `SNMP-GET` und `SNMP-BULK` Befehle abgesetzt. Zusätzlich bietet die Klasse Funktionalitäten, welche SNMP selbst nicht zur Verfügung stellt. Möchte man z.B. den kompletten Unterbaum eines Knoten erhalten und überschreitet die Anzahl der Einträge das Maximum von `GETBULK`, so können diese nur durch `GET` und `GETNEXT` ausgelesen werden. Diese Problematik behandelt die Klasse in einer `SNMP-WALK` Methode, welche die gleiche Funktionsweise wie ein `GETBULK` hat, aber es ermöglicht auch größere Listen an Werten abzufragen, die an einem Knoten hängen. Nachdem die jeweiligen SNMP Methoden die Werte zurückgegeben haben, werden diese ausgewertet.

Im Anschluss daran, wird ein Objekt der Klasse Port abgeleitet. Dieses Objekt dient dazu die portspezifischen Informationen zu speichern und im Anschluss den passenden SQL-Befehl zur Speicherung zu generieren. Dies wird durch die Methoden `setValues()` und `saveInDB()` realisiert. Beim Aufruf der `saveInDB()` Methode kommt es zur Verwendung der Datenbank-Klasse `JDBC-Oracle`. Diese überträgt den SQL-String an die Datenbank, nachdem die Methode `executeSQL()` aufgerufen wurde. Die gleiche Abfolge findet auch mit dem Objekt der Klasse `Host` statt. Hierbei werden anstatt wie beim Objekt der Klasse `Port` portspezifische Informationen abgelegt, sondern in diesem Fall die Informationen, welche den jeweiligen Host betreffen, abgespeichert. Nachdem alle Ports und Hosts abgespeichert wurden, ist die Methode `refresh()` des Switch-Objektes beendet.

Jedoch muss bedacht werden, dass während der Verarbeitung der Daten eine Vielzahl von Klassen aufgerufen werden, welche speziell für das Programm geschrieben wurden. Ein Beispiel stellt die Klasse mit dem Namen `Cisco` dar, welche Methoden zur Verfügung stellt, die speziell auf SNMP-Daten von Cisco-Geräten anwendbar sind. Darunter fallen Umformatierungen, die für das Auslesen der Uptime und des Switch Modells notwendig sind.

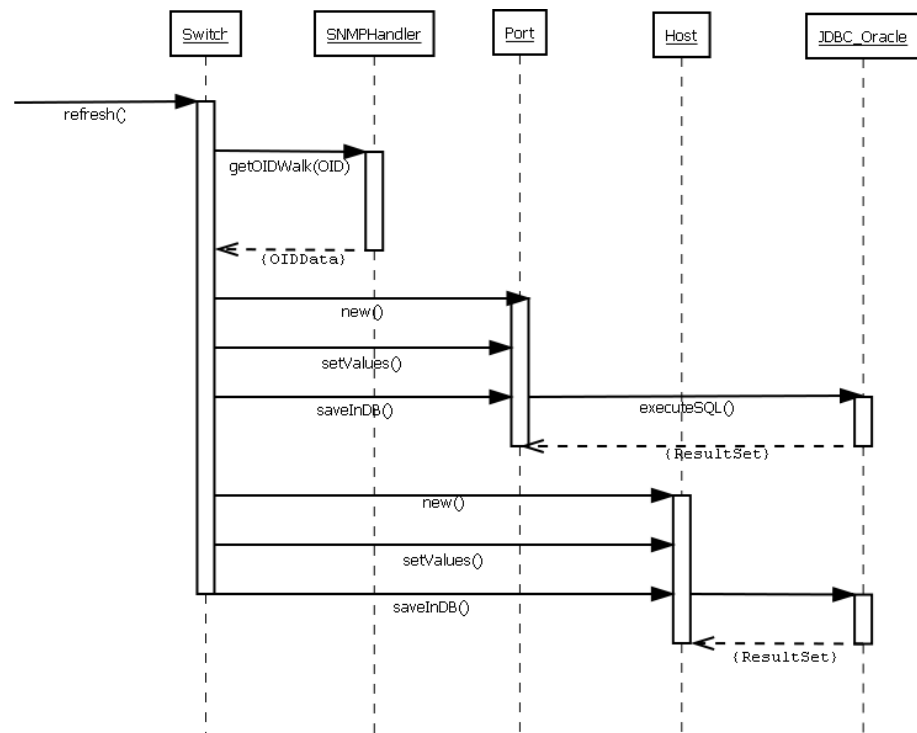


Abbildung 15.: Sequenzdiagramm 2

#### **4.1.4. Aktivitätsdiagramme**

Zur Darstellung des Programmablaufs eignen sich vor allem Aktivitäts-Diagramme von der UML. In Abbildung X ist das Diagramm, welches das Hauptprogramm beschreibt, zu sehen..

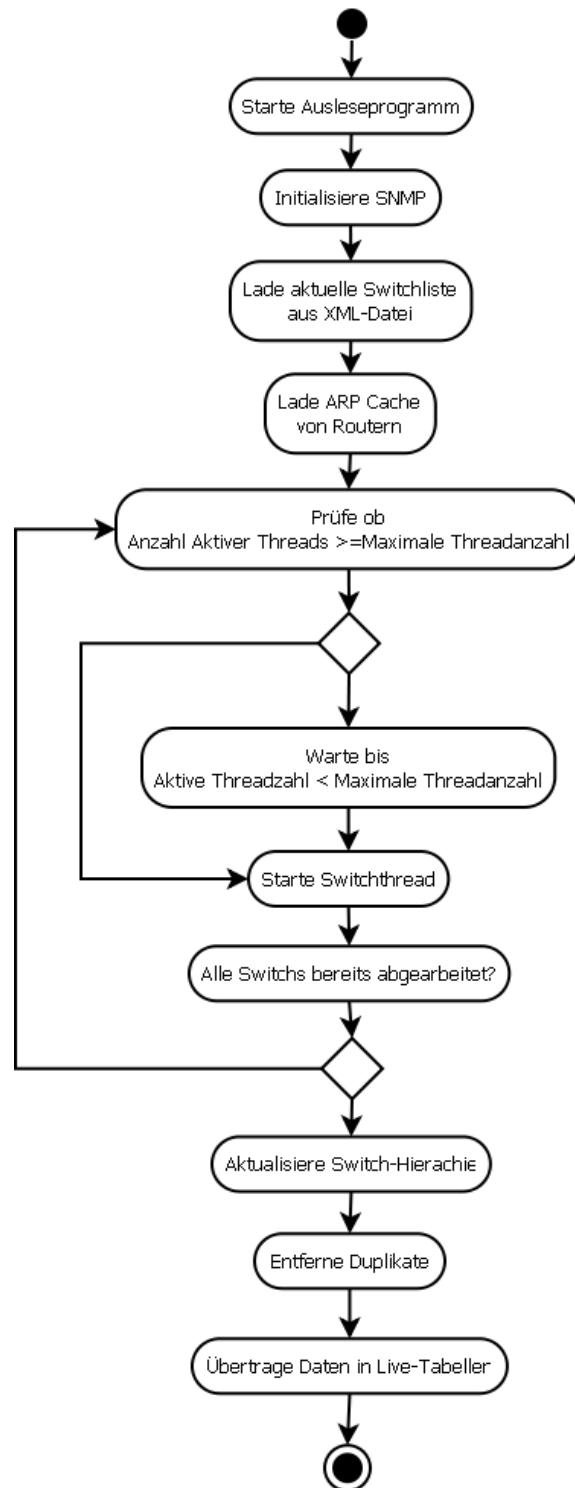


Abbildung 16.: Aktivitätsdiagramm 1

In Abbildung X ist der erste Teil des Aktivitätsdiagramm des Ausleseprogramms zu sehen. Die Notation ist wie in Abbildung X angepasst. Der Auslesevorgang lässt sich in drei Teile aufteilen. Zuerst werden die Informationen des Switches, dann die der Ports und letztendlich die der Hosts ausgelesen. Zuerst wird überprüft, ob der Switch SNMP unterstützt, ansonsten wird das Programm beendet. Ist dies der Fall werden die Informationen über den Switch ausgelesen und anhandessen der SQL-Befehl erzeugt, der zwischengespeichert wird, um die Vorteile der Datenbank-Transaktion zu nutzen, welche in Kapitel X näher erläutert werden.

Im nächsten Teil Werden alle Informationen über die Ports ausgelesen. Dann wird im Anschluss für jeden Port überprüft, ob dieser eine MAC-Adresse hat und kein virtuelles Interface ist, trifft dies nicht zu wird der jeweilige Port verworfen. Anschließend wird anhand der Informationen von CDP und STP entschieden, ob es sich hierbei um ein Uplink-Port handelt und diese Informationen hinterlegt. Daraufhin wird wiederum der passende SQL-Befehl generiert und zwischengespeichert.

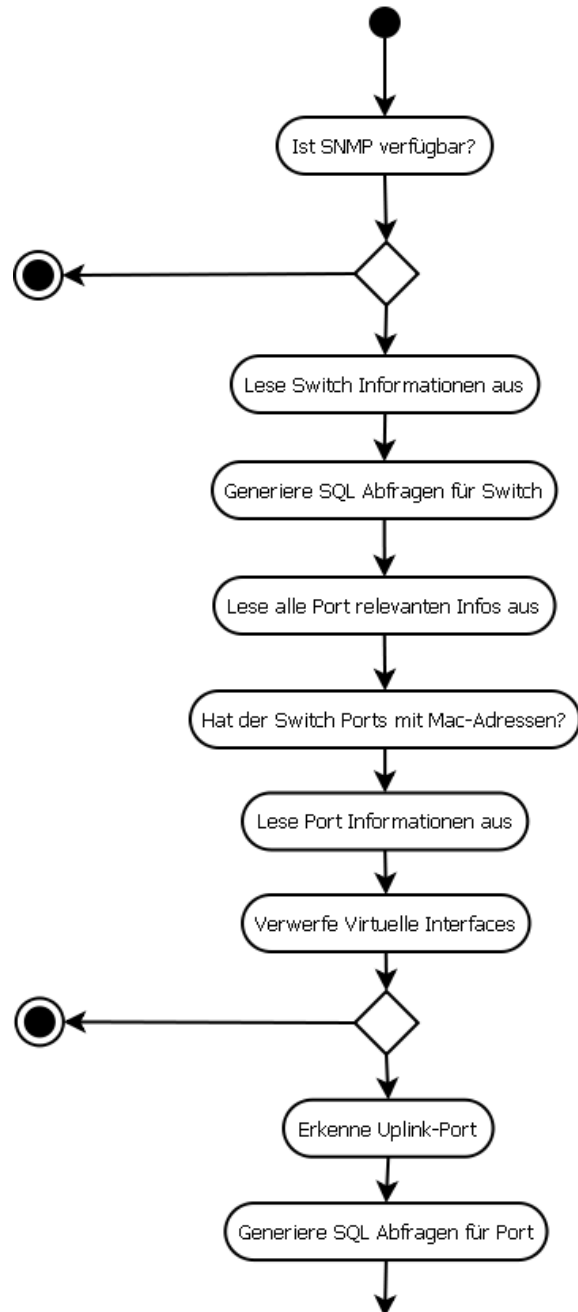


Abbildung 17.: Aktivitätsdiagramm 2 - Teil 1

Im letzten Teil des Auslesevorgangs werden wie in Abbildung X zu erkennen die Hosts erfasst. Zu Beginn wird überprüft, ob am jeweiligen Port ein Gerät angeschlossen ist, sollte keins angeschlossen sein wird der Port übersprungen. Ist jedoch ein Gerät angeschlossen, so wird unterschieden ob es sich hierbei um ein Uplink-Port handelt. Ist dies der Fall,

so werden die per CDP ausgelesenen Informationen verwendet um den Hostenintrag zu erstellen. Handelt es sich bei dem Gerät um ein normales Gerät, dass keine CDP Informationen enthält, so wird der Hostenintrag anhand der per SNMP ausgelesenen Informationen erzeugt. Hierzu wird die MAC-Adresse mit dem zuvor ausgelesenen IP-Adressen des ARP-Cache verknüpft. Anhand dieser IP-Adresse wird wiederum der DNS-Name vom DNS-Server angefordert und sofern es sich um ein Computer des Active Directories handelt, wird zusätzlich der Benutzer ausgelesen. Auf Grundlage dieser Daten wird wiederum ein SQL-Befehl generiert, der anschließend zwischengespeichert wird.

Bevor der Switch-Thread beendet wird, werden die Abfragen aus dem SQL-Puffer an einen speziellen Datenbank Thread übergeben der sich anschließend um das Absetzen der SQL-Befehle kümmert. So kann der Switch Thread bereits beendet werden und im Hauptprogramm das Auslesen des nächsten Switches begonnen werden, auch wenn noch nicht alle Datensätze in der Datenbank sind. Dies dient vor allem der Reduzierung der Wartezeit und somit einer Reduzierung der Auslesezeit.



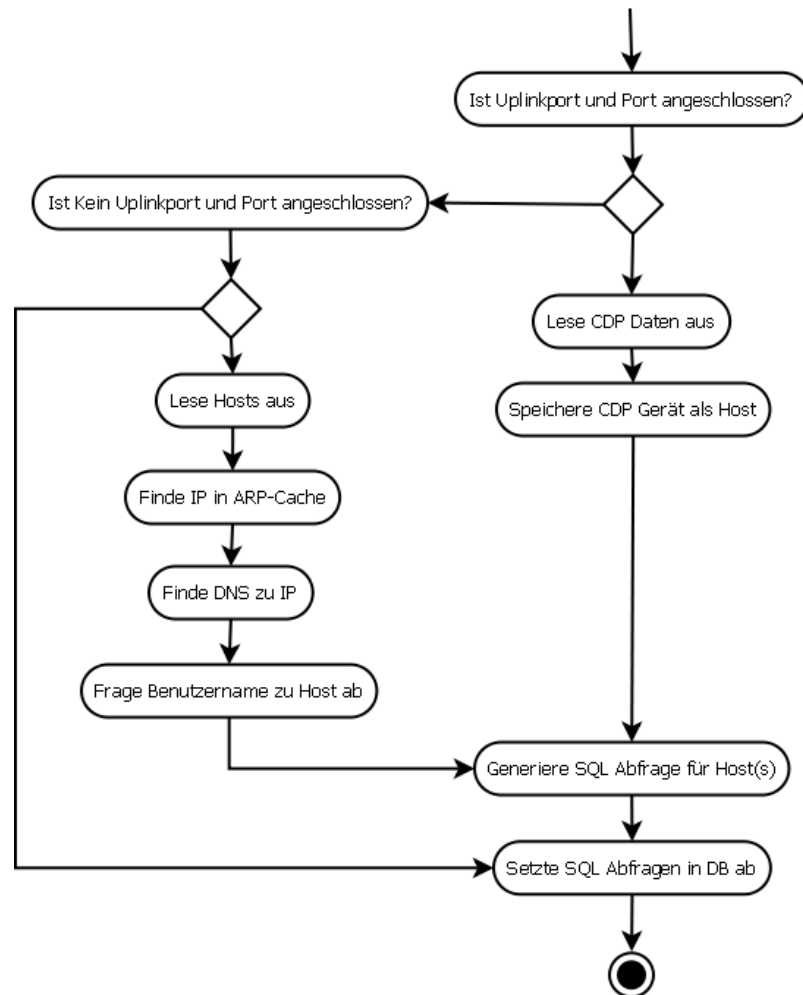


Abbildung 18.: Aktivitätsdiagramm 2 - Teil 2

#### 4.1.5. Entity Relationship Model

Aufgrund der vorherigen Analysen ist ein Entity Relationship Model (ERM) erstellt worden. Zum Erstellen wurde das Programm Dia verwendet, welches nicht der Chen Notation folgt, aber eine einfache und schnelle Erstellung einer schemenhaften Abbildung ermöglicht und auch flexibel gegenüber Veränderungen ist. Der Entwurf der Datenbank als ERM ist in Abbildung X zu sehen.

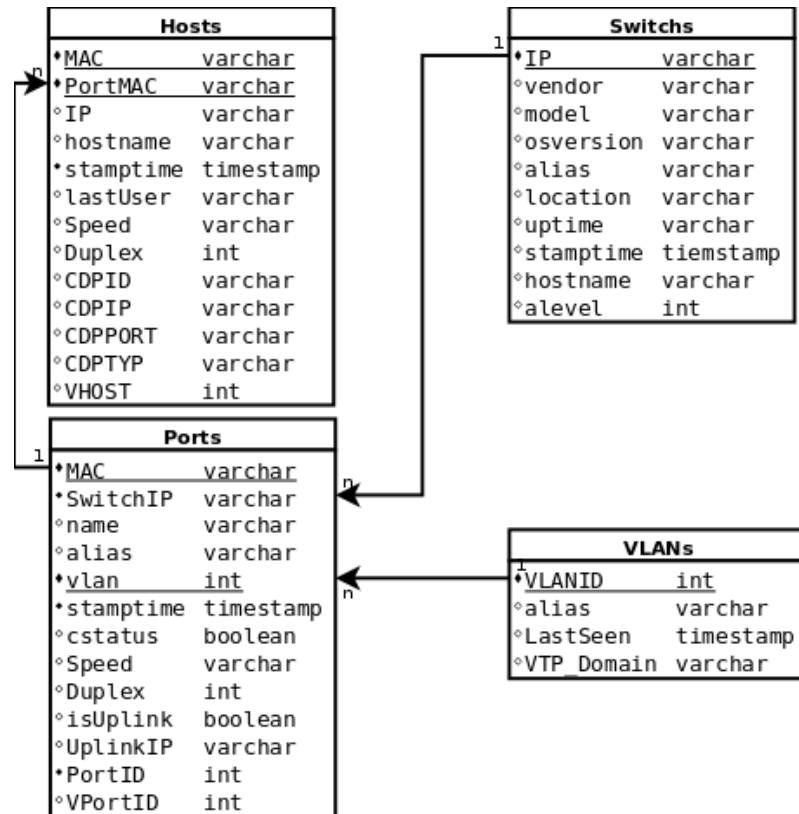


Abbildung 19.: ERM

Um alle Daten verwalten zu können, werden vier verschiedene Tabellen benötigt. Zunächst wird eine Tabelle zum Speichern der VLAN-Informationen benötigt. Hierzu existiert die Tabelle VLANs mit dem Primärschlüssel VLANID, welcher für jedes VLAN im Netzwerk eindeutig ist und somit auch als Primärschlüssel geeignet ist. Über diesen Schlüssel werden alle VLAN relevanten Informationen abgelegt. Dazu gehören neben dem Namen bzw. Alias des VLANs auch die VTP-Domain und ein Zeitstempel, um unterscheiden zu können, welche VLANs eventuell in der Vergangenheit aktiv waren und in der Gegenwart nicht mehr existieren.

Die Tabelle Switches dient dazu, alle Switch relevanten Informationen zu speichern. Um den Switch eindeutig zu identifizieren reicht die IP Adresse des Switches aus. Über diese werden anschließend Informationen über den Switch gespeichert. Dazu gehören IOS-Version, Modell, Ort, Uptime, aber auch Hostname, sofern vorhanden und die entsprechende Hierarchiestufe, welche anhand des Nagios-Systems ausgelesen wurde.

In der Port Tabelle werden alle Ports der Switches gespeichert. Die Ports selbst sind eindeutig über ihre jeweilige MAC-Adresse identifizierbar, jedoch soll ein neuer Eintrag angelegt werden, sofern sich die VLAN Einstellung ändert. Daher dient nicht nur die MAC-Adresse als Primärschlüssel, sondern auch die VLAN ID. Neben dem Namen des Ports und der MAC-Adresse gibt es auch ein Fremdschlüssel, welcher die IP des angehörigen Switches enthält, um eine eindeutige Zuordnung des Ports an einen Switch zu ermöglichen. Zusätzlich sind Attribute des Ports bezüglich des Status untergebracht. So ist zum Beispiel möglich, zu sehen, ob der Port gerade Up oder Down ist, mit welcher Geschwindigkeit auf dem Port kommuniziert wird und welcher Duplex Modus aktiv ist. Auch wird gespeichert, ob es sich bei dem Port um einen Uplinkport handelt.

In der Host-Tabelle werden alle im Netzwerk befindlichen Hosts gespeichert. Zur eindeutigen Identifikation der Hosts genügt die MAC-Adresse, jedoch geht aus der Anforderungsdefinition hervor, dass sofern der Host an einen anderen Port angeschlossen wird ein neuer Eintrag angelegt werden muss, daher wurde nicht nur die MAC-Adresse des Hosts, sondern auch die MAC-Adresse des Ports am Switch als Primärschlüssel genommen. Neben den üblichen Informationen wie IP und DNS-Hostname, werden auch Teile des Port Status abgespeichert, da man, sofern der Host an einem anderen Switch angeschlossen wird (z.B. ein Notebook der den Accesspoint wechselt), immernoch wissen möchte, mit welcher Geschwindigkeit dieser PC ursprünglich angeschlossen war. Es reicht aber auch schon das Suchen eines PCs der gerade ausgeschaltet ist. Um trotzdem herauszufinden, mit welcher Geschwindigkeit dieser angeschlossen war, ist diese Differenzierung notwendig. Zusätzlich dazu werden Informationen abgespeichert, sofern CDP Informationen über den Host bekannt sind. Hier werden dann CDP-Gerätetyp und der CDP-Port gespeichert, dies ist vor allem hilfreich bei Switches, Routern und Firewalls die das CDP Protokoll unterstützen, um eine bessere Einordnung zu erhalten. Neben diesen Informationen wird im Feld VHOST zusätzlich vermerkt, ob es sich bei diesem Host um einen physikalischen Host oder um einen logischen Host handelt. Das heißt, handelt es sich bei dem Host um einen virtuellen Server wird dieser explizit als Vhost markiert. Erkannt werden alle gängigen virtuellen Hosts. Dazu zählen mit VMWare, Virtualbox, VirtualPC, aber auch mit Parallels (Virtual Desktop, Server, Virtruzzo) oder Xen erzeugte Hosts. Zu beachten ist jedoch, dass z.B. das spezielle VMWare ESXi Server Betriebssystem selbst über eine virtuelle Schnittstelle kommuniziert, also das Host-System selbst und nicht die Gast-VMs.

## 4.2. Design Entscheidungen

Für die Umsetzung des Projektes müssen verschiedene Entscheidungen bezüglich der Umsetzung getroffen werden. Einige der Entscheidungen sind entweder durch die Anforderungen spezifiziert oder müssen unter Abwegung der Vor- und Nachteile ausgewählt werden.

Zuerst muss eine Entscheidung fallen, welche Programmiersprache gewählt wird. Nachdem Abgleich der Anforderungen und der Absprache mit der Fachabteilung, standen Perl und Java zur Auswahl. Da einer der Anforderungen auch die Geschwindigkeit betrifft, wurde zuerst eine Test Applikation in beiden Sprachen geschrieben. Dieses hat jeweils einen Switch mit wenig Netzwerk-Traffic mit einer Vielzahl von SNMP Abfragen beschäftigt, um Benchmarkwerte zu erhalten.

Im Benchmark-Programm wird die Zeit gemessen, wie lange es dauert um 1000 Abfragen durchzuführen. Der in Millisekunden gemessene Wert wird in die nachfolgende Formel eingesetzt:

Anzahl der Requests\*1000/Benötigte Zeit in ms= Abfragen pro Sekunde

Daraus resultiert die jeweilige Anzahl der Abfragen pro Sekunde, welche es ermöglicht einen Vergleich durchzuführen.

In der nachfolgenden Grafik sind die beide Programmiersprachen aufgezeichnet:

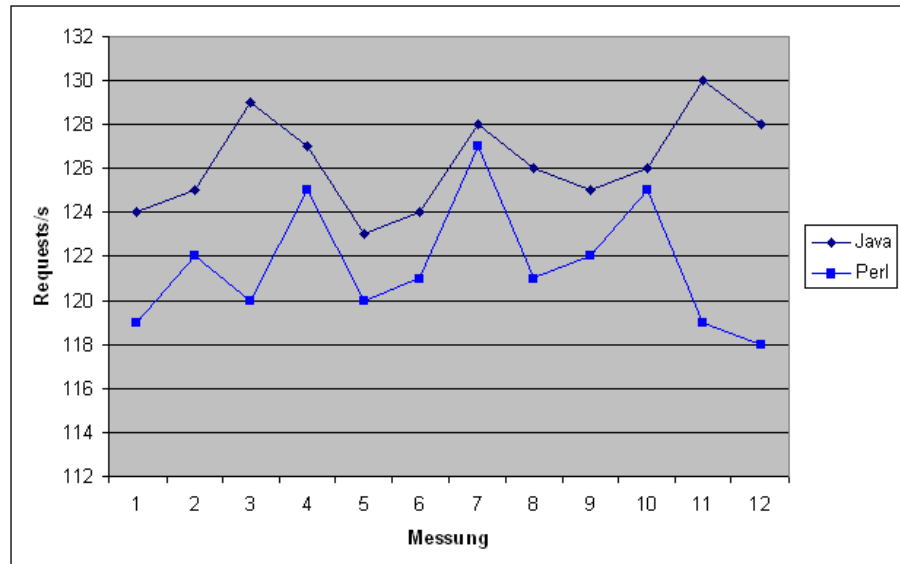


Abbildung 20.: Benchmark - Vergleich zwischen Perl und Java

Wie in der Abbildung zu erkennen, liegt Java ein Bruchteil vor Perl. Der geringe Unterschied zwischen den beiden Programmiersprachen lässt sich dadurch erklären, dass die Implementierung der SNMP-Abfragen minimal anders sind, der bei beiden fast gleiche Wert lässt darauf schließen, dass der Flaschenhals des Benchmarks der Switch selbst ist. Dies konnte dadurch validiert werden, wenn beide Benchmarks gleichzeitig gestartet wurden und dann eine Halbierung beider Benchmark Werte erfolgte.

Da es somit kein Unterschied macht, welche der beiden Sprachen zum Einsatz kommt, wird die Entscheidung anhand der Möglichkeiten beider Sprachen und deren Modularität gefällt und somit fällt die Wahl auf Java.

Da unter anderem eine Vielzahl an Switches abgefragt werden muss, empfiehlt es sich, eine Parallelisierung anzustreben, da jeder Switch nur eine begrenzte Geschwindigkeit aufweist. Hierfür eignet sich die Verwendung von Threads, um den Teil der Abfragen zu parallelisieren, der unabhängig voneinander ablaufen kann. Um den Nutzen einer Parallelisierung zu validieren, wurde wiederum ein Benchmark durchgeführt. Bei diesem Benchmark wurde die Anzahl der Threads variiert und jeweils die Zeit gemessen.

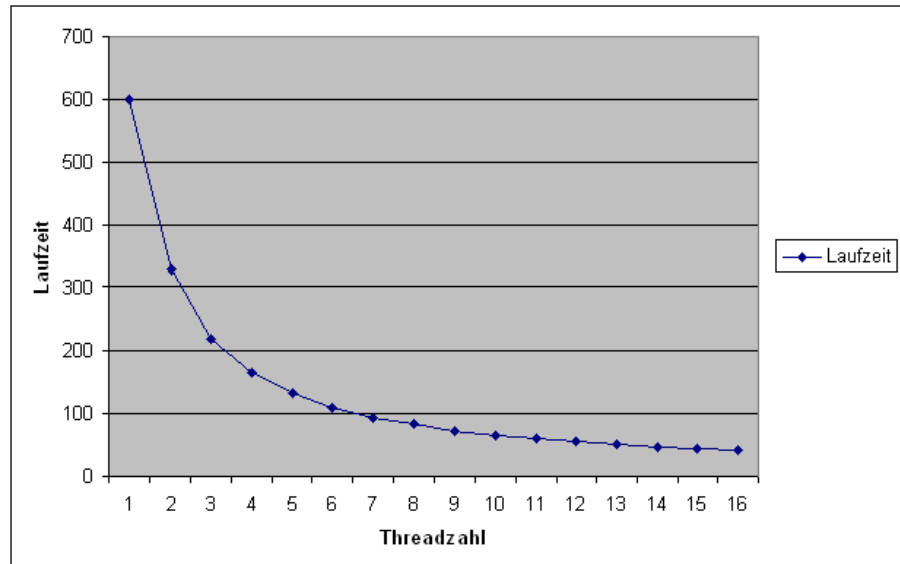


Abbildung 21.: Benchmark - Parallelisierung

Wie in der Abbildung 21 zu erkennen ist, lässt sich feststellen, dass durch die Parallelisierung ein bedeutender Geschwindigkeitszuwachs zu messen ist. Für die Abfragen pro Switch wurden jeweils 2000 Abfragen angenommen, das ein geschätzter Maximalwert pro Switch später darstellen sollte.

Bei der Durchführung des Benchmarks wurde ist ein Problem aufgetreten und zwar kam es zu Fehlern bei Thread-Zahlen über 20. Hierbei reicht der UDP-Puffer für die SNMP-Abfragen nicht mehr aus und somit kann der Empfang aller SNMP-Pakete nicht mehr sicher gestellt werden. Daher empfiehlt es sich bei der Implementierung einen niedrigeren Wert zu wählen und auf Nummer sicher zu gehen.

Bei der Version 2 des SNMP-Protokolls gibt es einen speziellen Abfrage Modus "BULKGET". Bei diesem musste auch überprüft werden, in welchem Umfang dieser dem normalen GET ein Geschwindigkeitsvorteil bringt. Hierfür wurden der erste Benchmark angepasst und die Liste der Ports von einem Switch abgefragt, ein Mal alle Ports sequentiell und ein Mal per BULK-Befehl.

Die Ergebnisse des Benchmarks lassen sich im nachfolgenden Bild erkennen.

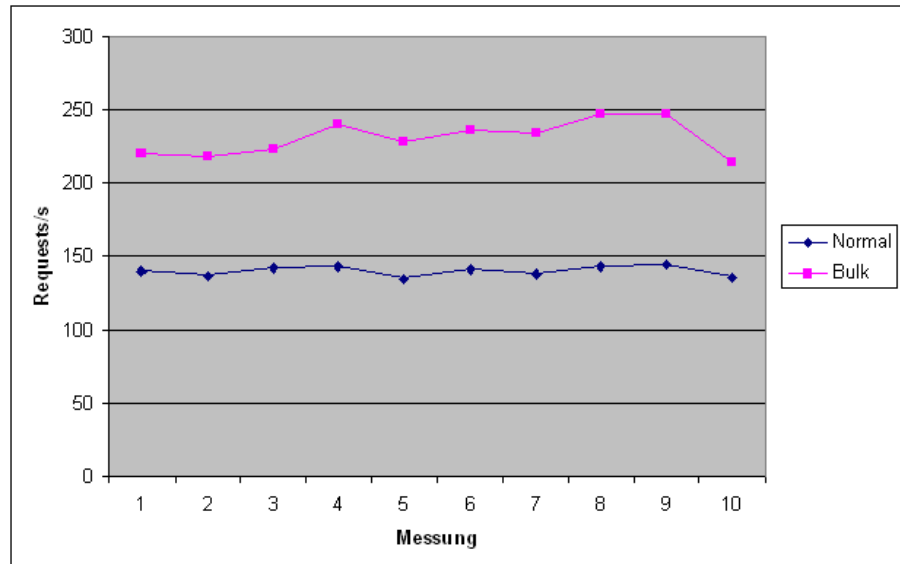


Abbildung 22.: Benchmark - SNMP Bulk

Es lässt sich feststellen, dass der BULK Modus einen bis zu dreifachen Geschwindigkeitsvorteil bringt. Jedoch muss beachtet werden, dass im Falle einer größeren Anzahl von Antworten (ab 50) diese nicht vom BULK Modus zurückgegeben werden, daher ist der Einsatz nur partiell sinnvoll.

Hierbei muss bei der Implementierung dann genauestens beachtet werden, wann welche Methode eingesetzt werden kann.

Eine weitere Sache, die überprüft werden muss, ist die Abwegung, ob es sinnvoller ist die SQL-Befehle, welche neue Datensätze hinzufügen, zuerst zu sammeln und mit einem Commit abzusetzen oder aber jeden einzelnen Datensatz separat abzusetzen. Hierfür wurde ein weiterer Benchmark durchgeführt, um eine Entscheidung in dieser Hinsicht treffen zu können.

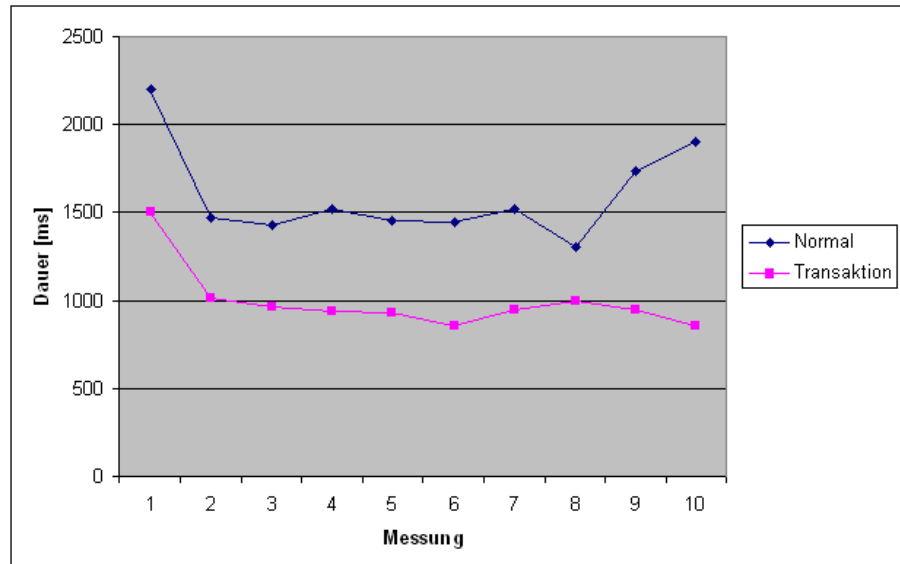


Abbildung 23.: Benchmark - Oracle Transaktionen

In Abbildung 23 ist zu erkennen, dass es von Vorteil ist, die Commits zuerst zu sammeln und dann abzusetzen. Während des Benchmarks wurde jedoch erkannt, dass durch eine Vielzahl von vorgehaltenen Abfragen, die durch die Parallelisierung entstehen, die Anzahl der gleichzeitig offenen Tabelleneinträge sich stark erhöht und somit das Limit der Datenbank überschritten wird. Zwar wurde anschließend das Limit der maximal offenen Einträge erhöht, jedoch kam es selbst dann zu Überschreitungen. Daher wurde der Algorithmus so angepasst, dass er immer in Paketen zu je 100 Befehlen arbeitet. Durch diese Maßnahme ist es trotzdem möglich, einen Geschwindigkeitsvorteil zu erhalten, bei gleichzeitiger Einhaltung der Limits.

### 4.3. Auswahl der Hilfsmittel

Für die Umsetzung des Projektes sind verschiedene Hilfsmittel notwendig. Zum einen wird eine Entwicklungsumgebung benötigt. Da die Wahl der Programmiersprache auf Java gefallen ist und kommerzielle Lösungen nicht zur Auswahl stehen, fiel die Wahl auf Eclipse, da es neben NetBeans zu den wenigen Open Source IDEs gehört, welche eine Vielzahl von Erweiterungsmöglichkeiten bietet und interaktive Funktionen wie Code-Vervollständigung oder Code-Vorlagen, aber auch standardmäßige Funktionen wie Syn-



taxhervorhebung unterstützt.

Zur Softwareversionverwaltung wurde Subversion verwendet. Die Auswahl wurde auf ein Open-Source System gelegt und zusätzlich auf ein zentrales System, um ein großes Maß an Interpolität zu erreichen. Die Wahl fiel speziell auf Subversion, da es im Gegenzug zu CVS das Versionsschema nicht auf einzelne Dateien sondern auf das ganze Projekt bezieht. Das hat den Vorteil, dass das Hinzufügen einer neuen Funktion nicht in der Hauptklasse Version 50 und in der Methodenklasse Version 70 gespeichert wird, sondern in einer gemeinsamen Version. Somit ist es für dne Entwickler möglich den Zusammenhang zwischen den einzelnen Dateien direkt zu erkennen, da die neue Funktion z.B. in der Revision 60 in beiden Dateien erkennbar ist.

## 4.4. Schnittstellen

Um ein Projekt mit vielen Einlese- und Exportfunktionalitäten erstellen zu können, werden viele Schnittstellen benötigt. Zum Einen wird bei diesem Projekt eine Schnittstelle für die SNMP-Abfragen benötigt, zum Anderen wird eine Anbindung an das Active Directory angestrebt. Zusätzlich müssen aber auch alle Datenbankzugriffe sowohl vom Einleseprogramm, als auch von der Website bewerkstelligt werden.

Im Folgenden wird auf die jeweiligen Schnittstellen eingegangen, wie diese sequentiell im Ablauf des Ausleseprogramms verwendet werden. Diese lassen sich wie folgt zusammenfassen:

- Einlesen aller benötigten Konfigurationsdaten
- Auslesen per SNMP
- Rekursive DNS Abfragen
- Auslesen per LDAP
- Datenbankverbindung Java

- Datenbankverbindung Webserver

Die erste Schnittstelle ist beim Start des Programmes zum Auslesen der Daten angesiedelt. Hierbei muss der Benutzer dem Programm eine Großzahl von Informationen übergeben. Es existiert eine spezielle XML-Datei in der die Switches, welche ausgelesen werden müssen, die Router, welche den ARP-Cache enthalten, sowie die notwendigen Zugangsdaten für die Datenbanken, enthalten sind.

Das Programm selbst bietet dem Nutzer keine Möglichkeit Übergabe-Parameter zu definieren, damit ein möglichst einfacher Ablauf ermöglicht und zusätzliche Fehleingaben des Benutzers vermieden werden.

Die nächste Schnittstelle ist die Kommunikation mit den Switches per SNMP. Um nicht die komplette SNMP-Kommunikation selbst per UDP implementieren zu müssen, empfiehlt es sich hierbei, eine bereits vorhandene API zu nutzen. Speziell für Java gibt es hierfür unter anderem folgende Möglichkeiten:

- SNMP4J
- jSNMP
- WebNMS SNMP
- iReasoning SNMP API
- netsnmpj

Schließt man nun alle kommerziellen Lösungen aus, so bleiben lediglich SNMP4J und netsnmpj übrig. Da mehrere SNMP-Abfragen später gleichzeitig durchgeführt werden müssen, ist die Wahl auf SNMP4J gefallen, da dieses threadsicher ist und auch ein größeren Funktionsumfang als netsnmpk bei der Implementierung mit sich bringt.

Um anhand der ermittelten IP-Adressen die dementsprechende DNS-Namen zu erhalten, ist es notwendig, ein Reverse-DNS-Lookup zu machen. Hierfür muss das sogenannte JNDI verwendet werden, welches einem erlaubt, selbstdefinierte DNS-Abfragen zu erstellen. Um beispielsweise den DNS-Namen der IP 192.168.0.1 herausfinden zu können, muss dieser aber erst in ein spezielles Format gebracht werden. Hierzu wird die IP Adresse

umgekehrt zu 1.0.168.192 und der Zusatz “.in-addr.arpa” angehängt. Daraus folgt dann:

1.0.168.192.in-addr.arpa

Diese Adresse wird inklusive dem Modus “PTR”, welche für einen Reverse-DNS-Lookup steht, an den DNS Service Provider übermittelt und das Resultat wiederum zurückgegeben.

Für das Auslesen des zugehörigen Benutzers muss das Active Directory befragt werden. Hierzu gibt es eine Mehrzahl von Möglichkeiten, die jedoch dadurch eingeschränkt werden, dass diese nur unter Windows lauffähig sind. Daher muss auf Ansätze zurückgegriffen werden, bei denen es möglich ist plattformunabhängig zu agieren. Hierzu muss man die Architektur vom Active Directory von Windows genauer untersuchen.

Generell lässt sich das Active Directory in folgende Komponenten unterteilen:

- LDAP-Verzeichnis
- Kerberos-Protokoll
- Common Internet File System
- Domain Name System (DNS)

Hiervon ist vor allem das LDAP-Verzeichnis bedeutend, welches es ermöglicht den Benutzer des jeweiligen Computers herauszufinden. Da LDAP per RFC genaustens spezifiziert ist (aktuell im RFC 4511) und Microsoft diesen Standard ebenfalls nutzt kann mit einer LDAP-API auf das Verzeichnis zugegriffen werden. Hierfür bietet Java mit seinen enthaltenen Bibliotheken ebenfalls eine Schnittstelle ähnlich der DNS-Abfragen an. Über diese können die Attribute eines Objektes im Verzeichnis abgefragt und gesetzt werden. In diesem Verzeichnis befinden sich auch die einzelnen Computer als Objekte. Diese wiederum haben diverse Attribute, welche unter anderem auch den aktiven Nutzer enthalten.

Für die Verbindung des Java-Programms zur Oracle und MySQL-Datenbank gibt es verschiedene Möglichkeiten. Hierzu zählen die verschiedenen Arten von Treibern, die eine Datenbankverbindung ermöglichen. Zu allererst ist die ODBC-Schnittstelle zu nennen, welche es ermöglicht, unabhängig von der eingesetzten Datenbank, die Anbindung an das Programm immer auf die gleiche Art realisieren zu können. Da diese Unabhängigkeit dadurch erreicht wird, dass die SQL-Befehle erst in die datenbankspezifischen umgewandelt werden müssen und somit ein Overhead entsteht, ist diese Lösung meist langsamer als eine native Lösung. Daher ist es von Vorteil spezielle vom Hersteller angebotene JDBC-Treiber einzusetzen, welche anstelle des JDBC-ODBC Brücken-Treibers den zusätzlichen Overhead meiden und direkt mit dem DBMS kommunizieren.

Bei der Verbindung mit dem Webserver kann wiederum ein ODBC-Treiber eingesetzt werden oder speziell für die Programmiersprache PHP geschriebene Bibliotheken, die es erlauben, ähnlich wie bei Java direkt mit dem DBMS zu kommunizieren um unnötigen Overhead zu vermeiden.

## 4.5. Zeitplan

Für die Umsetzung eines Softwareprojektes ist nicht nur ein Entwurf notwendig, sondern auch die Planung über den zeitlichen Ablauf. Der Faktor Zeit spielt eine wichtige Rolle, da er die beiden Variablen Qualität und Kosten begrenzt. Da bei diesem Projekt keine externen Kosten anfallen werden, kann dieser Punkt des magischen Dreiecks außen vor gelassen werden. Aufgrund der sehr knappen Zeit war es vor allem wichtig, einen zuvor definierten Plan zu haben, welcher die genauen Schritte spezifiziert. Um einen Überblick über den Umfang des Projektes zu bekommen, lohnt sich wiederum ein Blick auf die Usecases in Kapitel X, sowie die Anforderungsdefinition in Kapitel Y. Diese bilden eine gute und wichtige Basis, um einen Projektstrukturplan zu erstellen, welcher im Projektmanagement eine wichtige Rolle spielt. Aufgrund dieses Planes ist es möglich, die einzelnen Arbeitspakete zu definieren. Eine Zuordnung der Arbeitspakete zu einer Person muss nicht erfolgen, da die Software nur von einer Person entwickelt und umgesetzt wird. Nachdem die Arbeitspakete definiert wurden, mussten diesen jeweils eine Dauer zugewiesen und die jeweiligen Abhängigkeiten angegeben werden. Bei den Zeitangaben wurden Näherungswerte von bereits umgesetzten Projekten verwendet.

Normalerweise kann eine Ressource (in diesem Fall ein Mitarbeiter) nur immer einem Arbeitspaket aktiv zugewiesen werden, alle anderen Arbeitspakete können immer nur einzeln und nacheinander abgearbeitet werden, nie aber parallel. Aus diesem Grund wurde auch auf eine ausgiebige Planung der Ressourcen verzichtet. Viel wichtiger war hingegen die Terminplanung, welche aufgrund der Angaben (Dauer und Abhängigkeiten) anhand der Arbeitspakete erstellt werden kann. Im nachfolgenden sieht man in Abbildung X den Terminplan für das Projekt.

	⊙	Name	Dauer	Start	Ende	Vorgänger
1		Aufgabenstellung	2 tage	15.11.10 08:00	16.11.10 17:00	
2		Analyse der Situation	2 tage	17.11.10 08:00	18.11.10 17:00	1
3		SNMP Tests	4 tage	19.11.10 08:00	24.11.10 17:00	2
4		SVN Einrichtung	1 tag	25.11.10 08:00	25.11.10 17:00	3
5		SNMP Benchmarks	2 tage	26.11.10 08:00	29.11.10 17:00	4
6		Switch Auslesevorgang	2 tage	30.11.10 08:00	01.12.10 17:00	5
7		DNS Abfrage	1 tag	02.12.10 08:00	02.12.10 17:00	6
8		Oracle DB	2 tage	03.12.10 08:00	06.12.10 17:00	7
9		Switch Klasse	1 tag	07.12.10 08:00	07.12.10 17:00	8
10		Port Ausleseprozess	5 tage	08.12.10 08:00	14.12.10 17:00	9
11		Host Ausleseprozess	5 tage	15.12.10 08:00	04.01.11 17:00	10
12		Ausleseskript Optimierung	2 tage	05.01.11 08:00	06.01.11 17:00	11
13		Erstellen der Views auf die DB	2 tage	07.01.11 08:00	10.01.11 17:00	12
14		Zusätzliche SQL Abfragen (Webseite)	3 tage	11.01.11 08:00	13.01.11 17:00	13
15		Implementierung der Sichten in PHP	5 tage	14.01.11 08:00	20.01.11 17:00	14
16		Implementierung der Top Down Funkt	3 tage	21.01.11 08:00	25.01.11 17:00	15
17		Implementierung der Suche	2 tage	26.01.11 08:00	27.01.11 17:00	16
18		Implementierung der Sortieroptionen	2 tage	28.01.11 08:00	31.01.11 17:00	17
19		Test Ausleseprogramm	3 tage	01.02.11 08:00	03.02.11 17:00	18
20		Tests Weboberfläche	3 tage	04.02.11 08:00	08.02.11 17:00	19
21		Dokumentation	55 tage	15.11.10 08:00	11.02.11 17:00	
22		Bachelor-Arbeit	55 tage	15.11.10 08:00	11.02.11 17:00	

Abbildung 24.: Zeitplan - Arbeitspakete

Anhand des Terminplans kann wiederum ein Gantt-Diagramm erzeugt werden, welches einen grafischen Überblick über alle Arbeitspakete, sowie deren zeitliche Einordnung, ermöglicht. Das passende Gantt-Diagramm zum Projekt ist in Abbildung X zu sehen

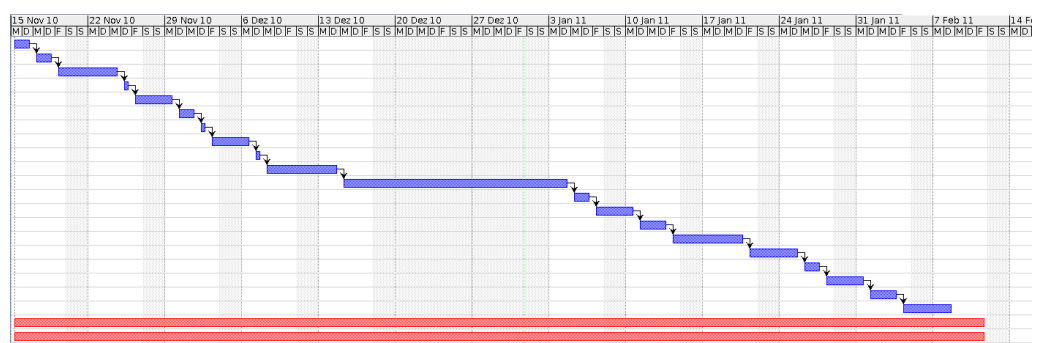


Abbildung 25.: Zeitplan - Gantttdiagramm

In diesem ist erkennbar, dass das Projekt einen konstanten sequentiellen Ablauf hat. Zwar hätte die Möglichkeit bestanden, ein paar einzelne Arbeitspakete parallel auszuführen, jedoch in Verbindung mit dem bereits angesprochenen Problem, dass nur eine Person an der Umsetzung arbeitet und diese ein Arbeitspaket nachdem anderen abarbeitet, keine zeitliche Möglichkeit für eine Parallelisierung gegeben ist. Zu beachten ist auch der rot gefärbte Balken am unteren Ende des Diagramms, welcher als kritischer Pfad angedeutet ist. Hierbei handelt es sich um die Dokumentation. Diese wird zwar als kritischer Pfad angezeigt, aber da die Umsetzung des Projektes sequentiell voranschreitet und immer ein Teil der Dokumentation bearbeitet wird, ist diesem keine weitere Betrachtung zu schenken. Die Zeiten der einzelnen Arbeitspakete sind flexibler anzusehen als diese im Detail definiert wurden, jedoch gibt es hierbei die Einschränkung, dass diese in Summe nicht den Endtermin überschreiten dürfen.

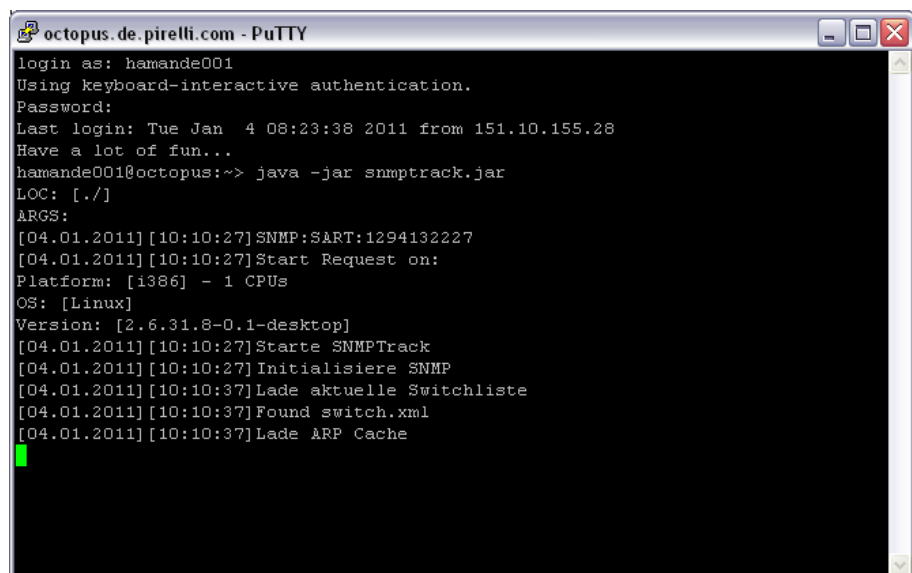
## 4.6. Realisierung

Nachdem sowohl die Umsetzung als auch der zeitliche Ablauf im Detail geplant wurden, konnte mit der Implementierung und somit mit der Realisierung des Projektes begonnen werden. Zuerst wurde, wie im Zeitplan definiert, das Ausleseprogramm umgesetzt und im Anschluss die dazugehörige Webseite. Realisiert wurde das System auf einem virtuellen VMWare Server mit dem Betriebssystem OpenSuse (Linux). Als Webserver diente Apache 2 in Verbindung mit PHP 5.3.2. Die PHP Version wurde speziell mit einer Oracle Datenbank Anbindung kompiliert. Auf diesem Server lief zusätzlich neben dem Webserver ein Oracle Datenbankserver in der Enterprise Edition in Version 11g R2. Das Ausleseprogramm selbst läuft betriebssystemunabhängig und mit einer beliebigen Java Version ab dem Jahre 2006. Zusätzlich ist es unerheblich, ob das Programm mit der original Sun Java VM ausgeführt wird oder mit dem quelloffenen OpenJDK, welches bevorzugt in Linux-Distributionen eingesetzt wird. Beim Start des Ausleseprogramms wird zunächst überprüft, ob die Datei Switch.xml existiert und bei Bedarf diese ausgelesen, welche wiederum die Switches inklusive deren Read-Community enthält. Ist die Datei nicht existent, werden die auszulesenden Switches anhand der Nagiosdatenbank ausgelesen. Die Daten für die Nagiosdatenbank, sowie für die Oracle Datenbank, in der später die Informationen abgelegt werden, befinden sich in der config.xml. Diese enthält neben der maximalen Threadanzahl auch die Einstellung für den SNMP Intervall, der in Kapitel X angesprochen wurde. Dieser Intervall dient zur Reduzierung der Last auf

den Switches während des Auslesevorgangs. In der Konfigurationsdatei ist ebenfalls ein Parameter zu finden, welcher es ermöglicht, ein Debuglevel zu setzen. Je nach Höhe dieses Levels werden nicht nur Fehler, sondern auch Warnungen, Hinweise oder auch Meldungen ausgegeben. Die Levels lassen sich wie folgt einordnen:

- Level 0: Anzeige von Fehlern
- Level 1: Zusätzliche Anzeige von Warnungen
- Level 2: Zusätzliche Anzeige von Hinweisen und Warnungen
- Level 3: Alle Meldungen

Zusätzlich werden alle Meldungen, die per Programm ausgegeben werden, auch in der Log-Datei gespeichert. Diese Meldungen sind abhängig vom Loglevel.

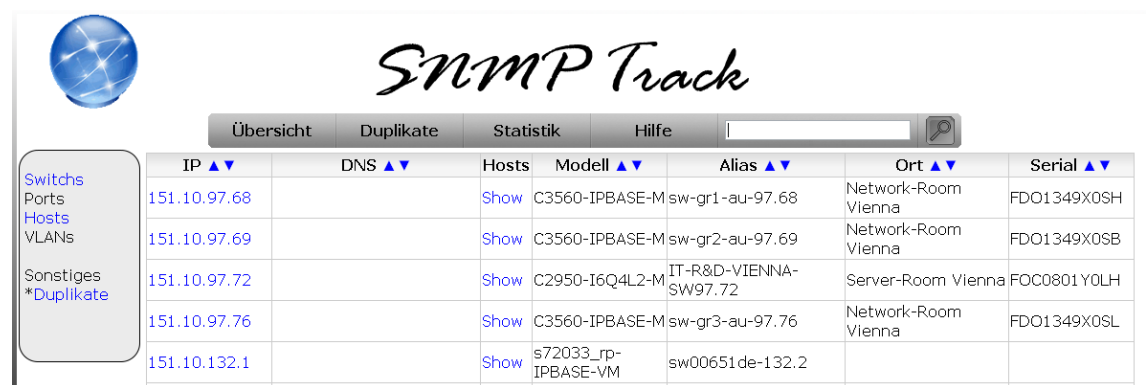


```
octopus.de.pirelli.com - PuTTY
login as: hamande001
Using keyboard-interactive authentication.
Password:
Last login: Tue Jan  4 08:23:38 2011 from 151.10.155.28
Have a lot of fun...
hamande001@octopus:~> java -jar snmptrack.jar
LOC: [./]
ARGS:
[04.01.2011][10:10:27]SNMP:SART:1294132227
[04.01.2011][10:10:27]Start Request on:
Platform: [i386] - 1 CPUs
OS: [Linux]
Version: [2.6.31.8-0.1-desktop]
[04.01.2011][10:10:27]Starte SNMPTrack
[04.01.2011][10:10:27]Initialisiere SNMP
[04.01.2011][10:10:37]Lade aktuelle Switchliste
[04.01.2011][10:10:37]Found switch.xml
[04.01.2011][10:10:37]Lade ARP Cache
```

Abbildung 26.: Programm - Ausleseprogramm

In Abbildung X ist der Start des Programms zu sehen. In diesem ist auch zu erkennen, dass keinerlei Parameter übergeben wurden. Dies wurde in der Art realisiert, um Bedienfehler zu vermeiden und die Ausführung per Skripte zu erleichtern. Zusätzlich werden fehlerhafte Einträge innerhalb der XML-Dateien einfach ignoriert.

Die Weboberfläche des Systems wurde per PHP realisiert. Bei dieser kommen Cascading Style Sheets und Java Script zum Einsatz. Um die Zeit für die Implementierung zu verringern, wurde auf die freie CMS Tints zurückgegriffen.<sup>44</sup>



IP ▲▼	DNS ▲▼	Hosts	Modell ▲▼	Alias ▲▼	Ort ▲▼	Serial ▲▼
151.10.97.68		Show	C3560-IPBASE-M	sw-gr1-au-97.68	Network-Room Vienna	FDO1349X0SH
151.10.97.69		Show	C3560-IPBASE-M	sw-gr2-au-97.69	Network-Room Vienna	FDO1349X0SB
151.10.97.72		Show	C2950-I6Q4L2-M	IT-R&D-VIENNA-SW97.72	Server-Room Vienna	FOC0801Y0LH
151.10.97.76		Show	C3560-IPBASE-M	sw-gr3-au-97.76	Network-Room Vienna	FDO1349X0SL
151.10.132.1		Show	s72033_rp-IPBASE-VM	sw00651de-132.2		

Abbildung 27.: Programm - Weboberfläche

Die Oberfläche in einem frühen Stadium der Entwicklung ist in Abbildung X zu sehen. Neben dem Grundgerüst für die Darstellung der Tabellen müssen spezielle SQL-Abfragen erstellt werden, da später eine Sortierung aller Spalten möglich sein muss. Diese Sortierung wird durch die Übergabe der GET-Parameter realisiert. Klickt der Benutzer auf eine Spalte wie in Abbildung X zu sehen, so wird der GET-Parameter 'sort' in folgendem Format übergeben:

sort=model\_A

Hierbei stellt 'model' 1:1 den Spaltennamen der zu sortierenden Spalte dar und 'A' gibt die Sortierrichtung an. In diesem Fall steht A für das englische Wort 'ascending', d.h. aufsteigend. Für eine absteigende Sortierung wird 'D' angegeben für 'descending'. Zusätzlich gibt es in der Weboberfläche die Möglichkeit, sich Duplikate in der Datenbank anzeigen zu lassen. Hierbei handelt es sich um Rechner, die auf zwei unterschiedlichen Ports im Netzwerk auf der selben Switch-Ebene gefunden wurden. Bei diesen Einträgen handelt es sich um 90% der Fälle um Hosts, welche den Port im Netzwerk gewechselt

<sup>44</sup><http://sourceforge.net/projects/tints-system/>



haben. In der Praxis bedeutet das, dass ein Notebook, das per WLAN verbunden ist, die Access Points oder ein Computer die Abteilung gewechselt.

## 4.7. Probleme bei der Implementierung

Während der Implementierung traten verschiedene Probleme auf, welche berücksichtigt werden mussten, um abschließend trotzdem alle gewünschten Anforderungen umsetzen zu können. Die Probleme lassen sich vor allem zu den folgenden Punkten einordnen:

- Zuordnung Host → Switchport
- Verwendete VLANs
- Uplinkport-Identifizierung
- Differenzierung der Switchlevels
- SNMP-Abfragen und CPU-last auf den Switches

Als erstes Problem ist die genaue Zuordnung zwischen Host und Switchport zu nennen. Hierbei hat sich während der Implementierungsphase gezeigt, dass die Daten, welche per SNMP auslesbar sind, nicht denen gleichen, die lokal auf den Switches per Telnet über das Cisco System zu erhalten sind. So kann das Cisco System einem direkt anzeigen, welche MAC-Adressen sich hinter welchem Port verbirgt. Über SNMP sind diese Informationen jedoch nicht direkt erhaltbar, hier müssen erst verschiedene abrufbare "Tabellen" verknüpft werden. Als erstes müssen die verwendeten VLANs des Switches abgefragt werden (die Problematik der VLAN Abfrage wird im anschließenden Absatz erläutert). Danach muss eine Liste der MAC-Adressen auf den virtuellen Ports pro VLAN abgefragt werden. Diese einzelnen Listen müssen miteinander kombiniert werden und alle Duplikate entfernt werden. Im Anschluss muss diese Liste mit einer weiteren Liste, welche die Zuordnung zwischen Port und virtuellen Port enthält, verknüpft werden. Aufgrund dieser Liste kann dann eine Verbindung zwischen MAC-Adresse des Ports und der MAC-Adresse des Hosts hergestellt werden. Durch diese Komplexität steigt die Anzahl der Abfragen deutlich, um die gleichen Informationen zu erhalten.

Eine weitere Problematik stellte sich beim Auslesen der verwendeten VLANs. Es besteht

die Möglichkeit per SNMP eine Liste abzufragen, welche ermöglicht, das verwendete VLAN für den jeweiligen Port zu erhalten. Ports die mehrere VLAN Zuordnungen haben, werden nicht angezeigt und sind Trunk-Ports, welche eigentlich Uplink-Ports sind bzw. an solche angeschlossen sind. In diesem Zusammenhang stellt sich die Problematik, dass es passieren kann, dass zwar ein Port für eine spezielles VLAN festgelegt wurde, jedoch sich dahinter noch ein VOIP-Telefon befindet, dass über ein "nicht sichtbares" VLAN kommuniziert, welches über die Liste im SNMP nicht erkennbar ist. Um trotzdem die MAC-Adresse des VOIP-Telefons zu erhalten, müssen alle im Netzwerk bekannten VLANs an einem Switch abgefragt werden, was wiederum die Anzahl der Abfragen erhöht, aber auch eine Problematik mit sich bringt. In diesem Zusammenhang gibt es spezielle Cisco spezifische VLANs, die sich im Bereich 1002-1005 befinden, welche ausgeschlossen werden müssen, da eine Abfrage der MAC-Adressen dieser VLANs zu einem Timeout führt und unnötig den Ausleseprozess verzögert.

Die Identifizierung der Uplink-Ports an den Switches stellte ebenfalls ein Problem dar. Hier gibt es keine einfache anwendbare Regel. Zuerst wurde angenommen, dass jeder Trunk Port ein Uplink Port ist. Diese Annahme war jedoch falsch, da in dem vorher aufgeführten Beispiel sich dahinter auch ein durchgeschleifter Computer an einem VOIP-Telefon befinden kann. Daher scheidet das Attribut "Trunk-Port" als solches zur Identifizierung aus. Das nächste Attribut, welches ausgewählt wurde, ist ein spezielles Cisco spezifisches Protokoll mit dem Namen CDP. Es dient dazu Cisco Geräte an einem Port zu identifizieren. Um einen Uplink Port zu identifizieren, wurde dann angenommen, dass sofern das Protokoll CDP vorhanden ist, es sich hierbei um einen Switch handelt. Hier stellt sich jedoch die Problematik, dass es auch Geräte gibt, die auf CDP antworten, welche weder Switches noch Router sind. Ein Beispiel sind VOIP-Telefone, die es auch von Cisco gibt. Daher wurde zusätzlich die Typerkennung des CDP Gerätes abgefragt, welche Auskunft über diverse Eigenschaften des Gerätes gibt. Über diese kann überprüft werden, ob es sich hierbei um einen Switch handelt. Leider hat sich in der Praxis gezeigt, dass dieses nicht vollständig ausreicht, da nicht alle Switches sämtliche Abfragen unterstützen, somit musste nach einer zusätzlichen Identifikationsmöglichkeit gesucht werden. Hier ist der Verfasser auf die Möglichkeit gestoßen, dass Spanning Tree Protokoll per SNMP abzufragen. Dieses ermöglicht die Kommunikation zwischen den einzelnen Switches, um unter anderem Pfadkosten der einzelnen Verbindungen auszutauschen. Nun wurde per SNMP die Anzahl der ausgehenden STP-Pakete auf dem jeweiligen Port ausgelesen und dieses als weiteres Kriterium eingeführt, über welches eine Identifikation des Portes

ermöglicht wird und zusammen in Kombination mit CDP eine sehr verlässliche Identifikation zur Verfügung stellt.

Ein weiteres Problem hat die Einordnung der Hierarchieebenen der Router dargestellt. Um Duplikate reduzieren zu können, ist es notwendig, zu wissen, ob die MAC-Adresse eines Hosts auf einem Uplink/Downlink Port eines Switches erkannt wurde oder an seinem normalen Port. Hierzu muss man wissen, ob der Switch ein Access-Switch ist oder ein Switch einer höheren Ebene.

Die Hosts im Netzwerk sind, mit einer geringen Anzahl von Ausnahmen, alle an Access-Switches angeschlossen. Zusätzlich gibt es Endgeräte, die auch an die Distribution-Switches angeschlossen sein können, z.B. die Server im Rechenzentrum. Um diese Problematik lösen zu können, wurden die Hierarchieebenen jeweils numerischen äquivalenten Zahlen zugeordnet. Hierzu wurden die Core Switches mit der Zahl 0, die Distribution Switches mit der Zahl 1 und die Access Switches mit der Zahl 2 definiert. Da die Information der Ebene des Switches nicht direkt von diesen selbst auslesbar ist, gibt es effektiv nur zwei Möglichkeiten. Zum Einen kann man die komplette Hierarchie anhand der Uplink Ports mit einem Programm logisch abbilden. Dazu müsste man einen Hauptknoten auswählen und anhanddessen eine Einordnung der Switches erfolgen lassen, was jedoch äußerst komplett ist und den Zeitlichen Rahmen des Projektes sprengt. Zum Anderen besteht die Möglichkeit diese Informationen von einer externen Quelle einzulesen, wovon auch Gebrauch gemacht wurde. In diesem Zusammenhang wird das bereits existierende Nagios System im Unternehmen verwendet. Zu diesem wird mittels JDBC zu einer MySQL Datenbank eine Verbindung aufgebaut, über diese dann im Anschluss die Gruppeninformationen der zu kontrollierenden Hosts, in diesem Fall der Switches, ausgelesen werden. Die Switches sind in verschiedene Host-Gruppen einsortiert, unter anderem auch in die benötigten Access, Distribution und Core Hierarchien. Über diese Gruppen werden die jeweiligen Hosts bzw. Switches erfasst und deren Hierarchielevel in der Datenbank eingetragen.

Neben den hauptsächlich logischen Problemen zur Identifikation stellte sich unter anderem noch ein weiteres Problem bei der Abfrage der Informationen per SNMP heraus. Wie bereits in vorherigen Kapiteln angedeutet, werden die auszulesenden Switches bei SNMP-Abfragen besonders belastet, weshalb eine Erhöhung der Anzahl von Abfragen stets kritisch in den Ausführungen bewertet wurde. In der Beschreibung zur Zuordnung zwischen MAC-Adresse des Hosts und der MAC-Adresse des Switches von Cisco, wird

diese Problematik nicht angesprochen, jedoch gibt es von Cisco weitere Informationen zu der Problematik, speziell bei einem Artikel zum Auslesen der CPU-Last per SNMP. In diesem wird angeführt, dass die Abfrage des Wertes selbst zu einer Verfälschung des Wertes führt. Zudem wird empfohlen nicht mehr als 1 Abfrage pro Sekunde per SNMP auf einen Switch zu machen, da geringere Intervalle bereits zu Beeinträchtigungen führen. Vergleicht man dies mit den Benchmarks aus Kapitel X, so lässt sich feststellen, dass diese Grenze beim Ausleseprozess um ein hundertfaches überschritten wird. Es muss auch bedacht werden, dass die Switches nicht für solche Operationen konzipiert wurden, sondern vielmehr für ihre eigentliche Aufgabe, insofern ist das Ziel die Last auf den Switches möglichst gering zu halten, um den Netzwerk-Betrieb in keinsten Weise zu beeinflussen. Aufgrund eines logischen Fehlers im Programmablauf kam es während diverser Tests dazu, dass einer der Core-Switches übermäßig ausgelastet war und dies dazu geführt hat, dass das Überwachungssystem der Switches anschließend Warnungen versendet hat. Solche Fälle dürfen im Betrieb nicht passieren, da die Ursache für die hohe Last der Switches nicht erkennbar ist und neben der Störung des Betriebes auch zusätzlich für Verwirrung sorgt. Daher ist es wichtig, die Anzahl der Abfragen per SNMP auf ein Minimum zu halten und im späteren Verlauf der Implementierung weiter zu optimieren, sofern sich Möglichkeiten ergeben.

## 4.8. Tests

Im Zuge der Implementierung müssen diverse Tests durchgeführt werden. Hierzu kommen neben dem obligatorischen Test der kompletten Implementierung, Tests welche Teile des Programm testen oder aber auch Tests die als Basis für Entscheidungen dienen. Im Nachfolgenden soll darauf eingegangen werden, welche Tests explizit durchgeführt werden müssen und welche Ergebnisse diese Tests hatten bzw. welche Konsequenz daraus gezogen wurden. Hierbei wird chronologisch vorgegangen, um die Tests in der Reihenfolge aufzuführen, in der sie für die Realisierung benötigt wurden.

Zu Beginn des Projektes standen mehrere Tests an, welche diverse APIs überprüft haben. Diese dienten dazu, sicherzustellen, dass eine Kommunikation über die benötigten Schnittstellen erfolgreich ist. Darunter sind Tests gefallen, wie die Überprüfung von SNMP4J, aber auch die des Programmcodes zum Reverse-DNS-Lookup, Abfrage des Active

Directory oder die Überprüfung der Datenbank-Anbindung. Das Ergebnis der Überprüfungen hat dazu geführt, dass eventuell eine andere API verwendet werden musste, sofern ein Test nicht erfolgreich war oder spezielle Einstellungen gemacht werden mussten.

Nachdem die APIs und Grundfunktionalitäten überprüft wurden, kam es zu den nächsten Tests. Diese dienten zur Entscheidungsfindung über die Nutzung von speziellen Algorithmen oder Design-Entscheidungen, wie sie in Kapitel X angeführt wurden. Die erste Entscheidung, die getroffen werden musste, war die Wahl der Programmiersprache. Hier zu wurde ein Benchmark des SNMP Aufrufs durchgeführt und die Anzahl der maximalen Abfragen pro Sekunde miteinander verglichen. Das Ergebnis war, dass die Programmiersprache als solche keinen Einfluss auf die Auslesegeschwindigkeit hat, da der Flaschenhals der Switch selbst darstellt. Der nächste Test der durchgeführt wurde, war die Überprüfung, ob eine Parallelisierung den gewünschten, zuvor prognostizierten Geschwindigkeitsvorteil erbringt. Das Ergebnis dieses Tests war, dass die gleichzeitige Ausführung des Programmcode antiproportional im Bezug auf Anzahl der verwendeten Threads und Laufzeit wirkt. Jedoch traten bei einer bestimmten Anzahl zunehmend Fehler aufgrund des Pufferüberlaufs auf. Dies führte dazu, dass die Anzahl der gleichzeitig aktiven Threads auf ein Maximum begrenzt wurde. Neben der Parallelisierung wurden auch die Auslesemethoden von SNMP selbst überprüft. So wurde getestet, ob eine Geschwindigkeitsverbesserung erreicht wird, wenn der SNMP-Bulk Modus, welcher in Version 2c spezifiziert wurde, verwendet wird. Das Ergebnis dieses Tests war eine dreifache<sup>45</sup> Leistungssteigerung, aber auch die Erkenntnis, dass per SNMP-Bulk nur eine begrenzte Anzahl von Tabelleneinträgen abfragbar ist. Diese Information war wiederum wichtig für die Implementierung des Ausleseskripts und hat zur Vermeidung von potentiellen Fehlern im Betrieb geführt. Ebenfalls überprüft wurde die Möglichkeit, mehrere Datensätze gesammelt an die Datenbank zu übertragen. Ziel war es, zu überprüfen, ob dies den Overhead der einzelnen Verbindungen reduziert und generell für einen Geschwindigkeitszuwachs sorgt. Aufgrund des Benchmark wurde festgestellt, dass dies der Fall ist, jedoch Probleme auftreten, da die Anzahl der gleichzeitig geöffneten Tabelleneinträge stark erhöht wird. Im Fall der einzelnen Abarbeitung und bei einer maximalen Anzahl aktiv arbeitender Switchthreads von  $N$  beläuft sich die maximale offene Tabelleneintragszahl auf  $N$ .

Bei einer gesammelten Übertragung beträgt die Anzahl der offenen Tabelleneinträge:  
 $N * M$

---

<sup>45</sup>Vgl. Kapitel X

Wobei N wiederum die Anzahl der maximal aktiven Threads darstellt und M die maximale Anzahl der SQL-Befehle die vom Switch abgesetzt werden müssen. Eine Näherungszahl für M ist:

$$1+52+52*H$$

Wobei hier H die Anzahl der Hosts hinter einem Port ist. Diese liegt in der Praxis durchschnittlich zwischen 1 und 2. Nimmt man hier den Wert 2 an, so ergibt sich für M der Wert:

$$1+52+52*2=157$$

Dieser wiederum eingesetzt in die ursprüngliche Formel ergibt:

$$N*157$$

Für N kann der Wert angenommen werden, der auf Grundlage der Benchmarks in Kapitel X empfohlen wurde. Das heißt, es wird für N der Wert 10 angesetzt und führt somit zu:  $10*157=1570$

Vergleicht man dies mit dem Wert der gleichzeitig offenen Tabelleneinträge bei einzelner Ausführung von 10, ist ein merklicher Unterschied zu erkennen. Vergleicht man nun die erhaltene Zahl mit dem Standard Wert von 50,<sup>46</sup> so lässt sich feststellen, dass der Wert einer gesammelten Durchführung den Standard-Wert um ein vielfaches überschreitet. Oracle selbst empfiehlt den Wert zu erhöhen und liefert teilweise <sup>47</sup> die Oracle Datenbank mit dem Wert 300 aus. Eine Erhöhung des Wertes bringt keinerlei Performencenachteile mit sich, jedoch dient die Begrenzung dazu, um Applikationen, welche die Tabelleneinträge nicht korrekt schließen, daran zu hindern, ein Großteil der Tabellen zu blockieren. Daher wurde der Wert auf 2000 erhöht, um auch die Spitzen im Ausleseprozess abzufangen.

Neben diesen Tests, welche zu Design Entscheidungen dienten, wurden während der Im-

<sup>46</sup>vgl. [http://download.oracle.com/docs/cd/B19306\\_01/server.102/b14237/initparams138.htm](http://download.oracle.com/docs/cd/B19306_01/server.102/b14237/initparams138.htm)  
[http://wiki.oracle.com/page/OPEN\\_CURSORS](http://wiki.oracle.com/page/OPEN_CURSORS)

<sup>47</sup>vgl.

plementierung durchgängig Tests durchgeführt, die den Funktionsumfang sicherstellen bzw. Funktionen an Ort und Stelle zu überprüfen.

Der abschließende Test, welcher alle Funktionen des Programms überprüfen soll, wird anhand der Usecases bzw. der Anforderungen abgeleitet. Von diesen wiederum kommt man auch auf die technischen Einzelheiten, wie z.B. das Ausleseprogramm. Es müssen folgende Programmteile getestet werden:

Webapplikation:

- Hostinformationen anzeigen
- Portinformationen anzeigen
- Switchinformationen anzeigen
- VLANinformationen anzeigen
- Suche anhand von IP/DNS/Benutzername
- Hierarchie Übersicht

Ausleseprogramm:

- Auslesen der SNMP-Informationen
- Zuordnung zwischen Host und Port
- Zuordnung zwischen MAC und IP
- Auflösung von IP zu DNS
- Abfrage des Benutzers
- Entfernung von Duplikaten

Nachdem diese jeweils auf Funktionalität und Korrektheit überprüft wurden, kann das Programm für die Nutzung übergeben werden.

Ergebnis der finalen Tests

Bei den finalen Tests wurde das komplette zusammengesetzte System als Black-Box-Test getestet. Dabei wurden alle Usecases durchgeführt und überprüft, ob das erwartete Ergebnis mit dem Erhaltenen übereinstimmt. Es wurde währenddessen festgestellt, dass es passieren kann, dass zwei DNS-Hostnamen einer IP zugewiesen sind. Zuerst wurde vermutet, dass es sich hierbei um ein programminterner Fehler handelt und ein Puffer verwendet wurde, der eventuell nicht geleert wurde. Nach der Durchsicht des Codes kon-

nte jedoch kein Fehler entdeckt werden. Im Anschluss wurden mit Betriebssystemmitteln (nslookup) die DNS-Auflösung überprüft. Hierbei wurde festgestellt, dass das Problem beim DNS-Server selbst liegt. Dieser behält über einen festen Zeitraum die Zuordnung von DNS zu IP. Jedoch prüft dieser im Gegensatz zum DHCP-Server nicht, ob eine IP bereits vergeben wurde oder doppelt in der Namensauflösung vorliegt. Durch diesen Umstand kann es passieren, dass Hosts die keinen DNS-Namen am Name-Server gemeldet haben einen alten Eintrag übernehmen.

## 4.9. Weitere Anwendungsfelder / Datamining

Neben den Anforderungen die von der Abteilung für das neue System angebracht wurden, gibt es weitere Anwendungsmöglichkeiten. Neben dem Anzeigen der jeweiligen Sichten (VLAN, Switch, Ports, Hosts) und der Suche in diesen ergeben sich weitere Nutzungsmöglichkeiten. Eine bereits erwähnte Möglichkeit ist die sogenannte Top-Down Sicht, welche es ermöglicht die Elemente hierarchisch zu durchsuchen. Ein Klick auf einen Switch öffnet somit die Liste mit allen dessen enthaltenen Ports. Diese sind wiederum auswählbar und führen zu den Hosts, welche an dem jeweiligen Port angeschlossen sind. Da es sich hierbei um mehrere Hosts handeln kann, da eventuell ein WLAN-Access-Point an einen Port angeschlossen ist, muss eventuell einer der Hosts explizit noch einmal ausgewählt werden, um dessen Detailinformationen zu erhalten. Die Möglichkeit, die sich zusätzlich bietet, sieht es vor, das Modell in die umgekehrte Richtung zu erweitern, so dass auch ein Ablauf der Hierarchie von unten nach oben ermöglicht wird. So soll der Benutzer über den Host, zu dessen zugehörigen Port und von diesem wiederum zum Switch gelangen.

Eine weitere Anwendungsmöglichkeit stellt das Anzeigen von statistischen Daten dar. So könnte berechnet werden, wie viele Switch-Ports aktiv belegt sind oder welche Switches am meisten freie Ports haben oder die wenigsten. Solche oder ähnliche Abfragen könnte man dazu verwenden, wenn neue Hosts im System an Switches angeschlossen werden müssen.

Zusätzlich bietet sich auch die Möglichkeit potentiell nicht erwünschte Hardware zu erkennen. Dies ist durch den Abgleich diverser Listen möglich.



Eine weitere Anwendungsmöglichkeit ergibt sich, wenn die Hosts in Verbindung mit den Ports gespeichert und deren Verknüpfung inklusive des Zeitstempels genutzt werden. So ist es anhand von diesen Daten machbar, eine Historie zu erstellen, die zeigt, an welchen Switches bzw. deren Ports der Host angeschlossen war. Diese Historie ist chronologisch sortierbar und gibt den "Pfad" des Hosts wieder. Im Beispiel könnte dies ein Notebook sein, welcher an verschiedenen Accesspoints angeschlossen war. Somit ist nicht nur dessen aktuelle Position erkennbar, sondern auch die vorherig genutzten Ports. Sofern man alle Switches auf einen Geländeplan grafisch darstellen würde, könnte man farblich den Weg des Notebooks darstellen. Neben der grafischen Darstellung der Switches auf dem Werksgelände bietet sich durch die, anhand von CDP ausgelesenen, Informationen die Möglichkeit, die Hierarchien zwischen den Switches grafisch dazustellen. D.h. durch die jeweils gespeicherten Uplink IPs auf den Ports kann somit eine Verbindung zu dem jeweiligen Nachbar-Switch hergestellt werden. Nimmt man nun die Core Switches als Hauptknoten, so kann man einen grafischen Baum bilden, welcher in den Blättern, in diesem Fall den Access Switches, endet. Theoretisch gesehen könnte man aus den Blättern einen weiteren Knoten machen und den Baum bei den Hosts enden lassen, jedoch leidet in der Praxis vor allem die Übersicht an einer solchen Darstellungsweise. Es wird daher abgeraten diese zu verwenden.

## 4.10. Wirtschaftliche Betrachtung

Untersucht man das Projekt hinsichtlich seiner Wirtschaftlichkeit, so muss zunächst die Ausgangssituation betrachtet werden. In dieser existiert die bereits bestehende Lösung CiscoWorks, welche in einer veralteten Version vorliegt. Für diese wurde bereits ein Angebot eingeholt, welches sich auf ungefähr 8000€ bewegt. Sofern hier kein Update durchgeführt wird, können alle neueren Switches nicht unterstützt werden. Ein weiterer Aspekt ist, dass nur ein Bruchteil der Funktionen, die von CiscoWorks angeboten werden, auch tatsächlich genutzt werden.

Im Vergleich dazu hat eine selbst entwickelte Lösung keinerlei Lizenzkosten und kann auch neuere Switches unterstützen. Betrachtet man beide Möglichkeiten von der monetären Seite, so lässt sich eine Einsparung von 8000€ erreichen. Jedoch muss bedacht werden, dass eine selbstentwickelte Lösung hingegen mit Arbeitszeit verbunden ist. Diese müssen als Kosten angerechnet werden. Jedoch handelt es sich bei diesem Projekt um

ein Sonderfall, da die Aufgabenstellung während der Praxisphase des Studenten mit der Bachelorarbeit einhergeht, somit fällt keine zusätzliche Mehrarbeit an, da die betreffende Person in jedem Fall mit dem Projekt beschäftigt sein muss. Trotzdem ist es sinnvoll zu kalkulieren, in welcher Höhe Kosten bei der eigenständigen Entwicklung entstehen. Da für eine detaillierte Berechnung die Personalkosten eingeplant werden müssen, ist es notwendig zuerst erst ein Stundensatz zu definieren. Um einen Vergleich erstellen zu können, ist es ratsam die Kosten zu berechnen, wenn das System von einer externen Person und wenn es von einer internen Person erstellt wird. Vergleicht man die Zahlen für einen Software-Entwickler im externen Umfeld, so lässt sich ein Durchschnittswert von 65€/Stunde festlegen.<sup>48</sup> Neben den Kosten spielt natürlich der Faktor Zeit ebenfalls eine wichtige Rolle. Für die Umsetzung des Projektes wurde in Kapitel X eine Annahme getroffen über die Zeit, die benötigt wird, das Projekt umzusetzen. Jedoch muss bei dieser Schätzung bedacht werden, dass hierbei, neben einer Einarbeitungszeit, gleichzeitig eine Bachelorarbeit geschrieben wird und die Erfahrung des Entwicklers nicht auf dem selben Level eines externen Spezialisten ist. Grob geschätzt ist also die Annahme von 50 Werktagen auf ca. die Hälfte reduzierbar. Bei einer Annahme von 30 Tagen (zu je 8 Stunden pro Tag) ergibt sich eine Stundenzahl von 240 für das Projekt. Verrechnet man diese Stundenzahl mit dem Stundenlohn, so erhält man 15.600€. Dieser Wert liegt deutlich über dem Wert einer neuen Lizenz. Daher ist von einer Umsetzung des Projektes von einem externen Entwickler abzuraten. Für die Umsetzung durch einen internen Entwickler liegen leider keine genauen Zahlenwerte vor, daher wird ein beispielhafter Wert angenommen, wie er in jedem Unternehmen gültig sein könnte. Für einen Mitarbeiter, der mit allen Kosten (inkl. betrieblicher Anteil der Versicherungen) 5000€ pro Monat Kosten verursacht, lässt sich bei einer durchschnittlichen Zahl von 20 Werktagen pro Monat und einer 40 Stunden Woche, ein Stundenlohn wie folgt berechnen:

$$5000/(20*8)=31,25\text{€}$$

Berechnet man nun die Kosten für das Projekt für einen internen Mitarbeiter, so lässt sich feststellen, dass die Kosten deutlich gesunken sind:

$$31,25\text{€}*240=7500\text{€}$$

---

<sup>48</sup>Vgl. Diverse Stundensätze in der Software-Entwicklung

Bei anderen Kosten für interne Mitarbeiter schwankt dieser Wert natürlich. Generell lässt sich jedoch feststellen, dass der Wert, welcher die Kosten für einen Mitarbeiter beschreibt nicht über einen Wert X liegen darf, der sich wie folgt bestimmen lässt:

$$(X/(20*8))*250=8000$$

$$(X/160)*250=8000$$

$$X*(25/16)=8000$$

$$X=8000/(25/16)$$

$$X=5120$$

Somit lässt sich feststellen, dass ab einem Kostenpunkt von ca. 5120€ für einen Mitarbeiter pro Monat dieser die Kosten eines Updates der Cisco Software übersteigt. Jeder Wert unter diesem Grenzwert kann als unproblematisch angesehen werden.

In diesem Zusammenhang muss auch betrachtet werden, wie oft ein Update einer solchen Software ansteht. Generell gibt es bei CiscoWorks verschiedene Versionen. Sofern es sich um ein Major-Release handelt (1.0 -> 2.0, 2.0->3.0), muss eine neue Lizenz gekauft werden. Um nun eine Schätzung machen zu können, welche Kosten auf die PD zukommen, wenn fortlaufend Updates für CiscoWorks gekauft werden müssen, ist die Versionsgeschichte von CiscoWorks zu Rate zu ziehen.<sup>49</sup> Diese sieht wie folgt aus:

CiscoWorks 2000 -> 1999

CiscoWorks LMS 1.0 -> April 2000

CiscoWorks LMS 2.0 -> März 2001

CiscoWorks LMS 3.0 -> Juni 2007

CiscoWorks LMS 4.0 -> September 2010

Diese Daten ergeben einen Durchschnittswert von einer Version in 2,5 Jahren. Setzt man diesen auf drei Jahre, ist dies dann auch ein realistischer Wert, wenn die beiden letzten Versionen im Bezug gesehen werden. Somit kann man von einem Kostenpunkt von fortlaufend von  $8000/3=2667\text{€}$  pro Jahr ausgehen. Mit diesen Werten lässt sich eine grobe Schätzung für die kommenden Jahre bewerkstelligen. Um ebenfalls der Inflation gerecht zu werden wird eine Inflation von 2% angenommen, basierend auf der Inflation von 3,06% der letzten 36 Jahre <sup>50</sup> und der aktuellen Situation (vgl. stat. Bundesamt).

<sup>49</sup>Vgl. diverse Cisco Artikel auf [cisco.com](http://cisco.com)

<sup>50</sup>vgl. stat. Bundesamt

Kosten die bei der Wahl der Updates für die nächsten fünf Jahre entstehen:

$$2667 \cdot (1,02)^{(1-1)} + 2667 \cdot (1,02)^{(2-1)} + 2667 \cdot (1,02)^{(3-1)} + 2667 \cdot (1,02)^{(4-1)} + 2667 \cdot (1,02)^{(5-1)} = 13879,18 \text{€}$$

Würde man sich für die Selbsterstellung der Software entscheiden muss man zum Einen die Entwicklungskosten von 240 Stunden und zum Anderen zusätzlich für jeden neuen nicht unterstützen Router eine Anpassungszeit von maximal sechs Stunden berechnen. Da im Netzwerk nur eine geringe Anzahl von unterschiedlichen Switches im Betrieb ist, ist die Anzahl der Anpassungen pro Jahr im Durchschnitt maximal auf zwei zu sehen. Die Inflation wird ebenfalls auf den Lohn angerechnet. Für die Kosten auf 5 Jahre gesehen ergibt sich somit:

$$31,25 \cdot 240 + 2 \cdot 6 \cdot 31,25 \cdot (1,02)^{(1-1)} + 2 \cdot 6 \cdot 31,25 \cdot (1,02)^{(2-1)} + 2 \cdot 6 \cdot 31,25 \cdot (1,02)^{(3-1)} + 2 \cdot 6 \cdot 31,25 \cdot (1,02)^{(4-1)} + 2 \cdot 6 \cdot 31,25 \cdot (1,02)^{(5-1)} = 9451,52 \text{€}$$

Diese Zahl ist zwar höher als die Kosten für eine neue CiscoWorks Version, aber es besteht auch die Möglichkeit fremdartige Switches zu unterstützen. Beachtet man nun die realen Kosten, so kann man feststellen, dass für die initale Entwicklung keine Kosten entstanden sind, so lässt sich die Rechnung wie folgt aktualisieren:

$$2 \cdot 6 \cdot 31,25 \cdot (1,02)^{(1-1)} + 2 \cdot 6 \cdot 31,25 \cdot (1,02)^{(2-1)} + 2 \cdot 6 \cdot 31,25 \cdot (1,02)^{(3-1)} + 2 \cdot 6 \cdot 31,25 \cdot (1,02)^{(4-1)} + 2 \cdot 6 \cdot 31,25 \cdot (1,02)^{(5-1)} = 1951,52 \text{€}$$

Wie sich erkennen lässt, stellt dies nur ein Bruchteil der Kosten eines Updates wieder. Somit ist die Entscheidung für die Entwicklung eines eigenen wirtschaftlich Systems sinnvoll. Jedoch sollte auch bedacht werden, dass die eigene Umsetzung gewisse Risiken mit sich bringt, z.B. könnten unvorhergesehene Probleme bei der Implementierung auftreten und somit die Implementierung verzögern, was zu einer Erhöhung der Stundenzahl der Initialimplementierungsphase (240 Stunden) führt.

## 5. Fazit

Zum Abschluss der Arbeit lässt sich bei einem Vergleich des Resultats mit der Ausgangssituation feststellen, dass die Aufgabenstellung erfüllt wurde. Darüberhinaus wurden Ansätze aus der Literatur kritisch anhand von diversen Tests untersucht und aufgrund dessen die Entscheidungen getroffen, um die optimalsten Ergebnisse zu erzielen. Während den Tests wurde festgestellt, dass sich ein Großteil der Aussagen in der Literatur mit der Praxis abdeckt. Jedoch gibt es auch teilweise Unterschiede. Zum Beispiel bei der Parallelisierung von Programmen. In der Literatur wird hauptsächlich auf Threadzahlen im hohen Bereich, sowie die Verwaltung von gemeinsamen Ressourcen eingegangen. Es wird jedoch auf die optimale Threadzahl, bei der es "ökonomisch" nicht mehr sinnvoll ist diese zu erhöhen, weniger Aufmerksamkeit geschenkt. Dies ist sehr gut in der Abbildung des Benchmarks zur Parallelisierung zu sehen, in der Werte über 10 keinen nennenswerten Geschwindigkeitsvorteil mehr bringen. Hierbei konzentriert sich die Literatur auch vermehrt auf einzelne Punkte, wenn der Hauptaugenmerk vielmehr auf das komplette System gelegt werden sollte. Daher empfiehlt es sich, nicht unreflektiert Ansätze aus der Theorie zu übernehmen, sondern diese selbstständig zu überprüfen, ob die diskutierten Ansätze für ein Projekt passend sind und in wiefern diese eine Auswirkung haben. Zudem sind Empfehlungen von Herstellern nicht einfach als gegeben hinzunehmen. So ist es korrekt, dass bei der Abfrage eines Switches die CPU Last erhöht wird, jedoch ist eine Begrenzung der SNMP Abfragen auf eine Abfrage in der Sekunde sehr zurückhaltend formuliert, da je nach Switch bis zu über 450 Abfragen in einer Sekunde bearbeitet werden können. Somit kann mit einer Limitierung auf 20 Abfragen pro Sekunde trotzdem ein Kompromiss zwischen Last und Geschwindigkeit getroffen werden.

Interessant ist auch, dass in der Literatur eine scheinbare Lücke zwischen den abstrakten Vorgehensweisen und den einzelnen implementierten Algorithmen zu finden ist. Hier wird sehr detailliert meist auf Einzelfälle eingegangen, jedoch nicht so stark auf Aspekte, die eine Anwendung als Ganzes betrachten.

In diesem Zusammenhang ließ sich feststellen, dass für die Umsetzung des Systems eine beachtliche Menge an Technologien zusammenspielt, um letztendlich die gewünschten

Informationen zu erhalten. Neben Dingen wie MAC-Adresse, IP, DNS, müssen auch Details wie der ARP-Cache, SNMP, LDAP/Active Directory beachtet werden. Diese muss man für die Implementierung eines Systems mit diesem Ausmaß nicht nur kennen und deren Funktionsweise verstanden haben, sondern auch die Transferleistung erbringen wie diese in Kombination einzusetzen sind.

Bei der Auswertung der Tests wurde vermehrt festgestellt, dass es zur Speicherung von nicht eindeutigen Daten kommen kann. Ein Beispiel hierfür war die umgekehrte Auflösung einer IP Adresse zu einem DNS-Hostnamen. Hier kommt es vor, dass der DNS-Server für zwei DNS-Hostnamen die gleiche IP Adresse hinterlegt hat. In solchen Fällen muss abgewegt werden was mit den Daten passiert. Diese Entscheidung muss jeweils im Einzelfall getroffen werden. Wichtig in diesem Zusammenhang ist vor allem, dass es in der Realität vorkommen kann, dass Daten uneindeutig sind und diese Uneindeutigkeit nicht durch einen Algorithmus lösen ist bzw. zu einer Eindeutigkeit gebracht werden kann. Hierbei ist es wichtig im Dialog zu stehen mit den betreffenden Personen und eine Lösung zu finden.

Ein weitere Aspekt, der als Ergebnis gesehen werden kann, ist in der Aufwandsschätzung zu sehen. Durch Erfahrungen, die durch frühere Projekte gesammelt wurden, konnten für das Projekt sehr genaue Zeitschätzungen erstellt werden und damit auch eine bessere Einhaltung der vorgegeben Parameter gegeben werden.

Im Zusammenhang mit der Planung lässt sich auch feststellen, dass durch das vorherige Modellieren anhand von ERM und UML bedeutend Zeit bei der Implementierung eingespart werden konnte und auch eine deutlich bessere Übersicht über die zu implementierenden Funktionen und der Zusammenhänge der einzelnen Programmteile möglich ist.

Zum Abschluss lässt sich feststellen, dass durch eine gekonnte Kombination aus Theorie und praktischer Erfahrung ein sehr effizientes System erstellen lässt, jedoch muss bedacht werden, dass nur durch eine solide Basis dies ermöglicht wird. Die Basis für ein System liegt immer auf Grundlage der Gespräche und Abstimmungen mit den Beteiligten. Ohne diese wird zwar das vermeintliche Ziel erreicht, nicht aber die Anforderungen der Benutzer erfüllt.

# Literaturverzeichnis

## **A. Anhang**

### **A.1. Konfiguration SNMP-Track**

Noch leer.

### **A.2. Benchmarkwerte**

Noch leer.



## Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich:

1. dass ich meine Projektarbeit mit dem Thema

*Systementwicklung eines Usertracking-Systems bei der Pirelli Deutschland GmbH*

ohne fremde Hilfe angefertigt habe;

2. dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe;

3. dass ich meine Projektarbeit bei keiner anderen Prüfung vorgelegt habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Höchst, den 5. Februar 2011

---

DENIS HAMANN