Systementwicklung eines Usertracking-Systems bei Pirelli Deutschland GmbH

Bachelorarbeit

im Fachgebiet Wirtschaftsinformatik

vorgelegt von:	Denis Hamann
Kurs:	WWI08B
Studienbereich:	Wirtschaftsinformatik
Matrikelnummer:	253977
Wissenschaftlicher Betreuer:	Olaf Rogge
Unternehmerischer Betreuer:	Gerd Hoffarth

Abstract

Von: Denis Hamann
Kurs: WWI08B
Firma: Pirelli Deutschland GmbH
Thema: Systementwicklung eines Usertracking-Systems bei Pirelli Deutschland GmbH

Inhalt:
Noch leer.

Inhaltsverzeichnis

Abkürzungsverzeichnis

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

Abbildungsverzeichnis

1. Einleitung

Noch leer.

2. Theoretische Grundlagen

2.1. Datenbank Entwurf / ERM

Unter dem Datenbank Entwurf 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 im Voraus ist essentiell, da im späteren Prozess nderungen der Datenbankstruktur nicht nur die Datenbank selbst, sondern auch alle mit ihr verbundenen Applikationen betreffen wrden. Um die Beziehungen korrekt darstellen zu können, wird das Entity-Relationship-Modell[1] verwendet. Durch diese Modell lassen sich sogenannte ER-Diagramme zeichnen, z.B. nach der Chen Notation[2]. Ein ER-Diagramm nach Chen stellt die Entittstypen (Klassen), Attribute, sowie Beziehungen (Relationen/Kardinalitäten) dar. Im folgenden Beispiel soll ein ER-Diagramm nach der Chen-Notation kurz erläutertwerden.

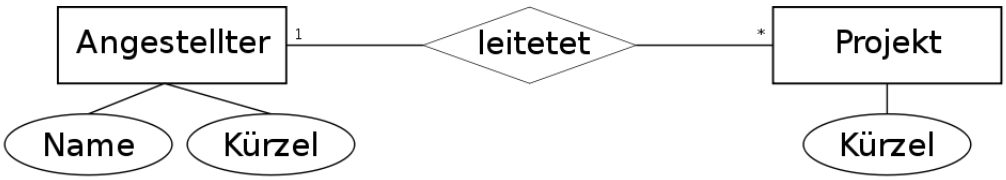

Abbildung 2.1.: Chen Notation 1:N

Das Diagramm beschreibt folgenden Sachverhalt:

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

[1] 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

[2] 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

[2] In Anlehnung an Chen

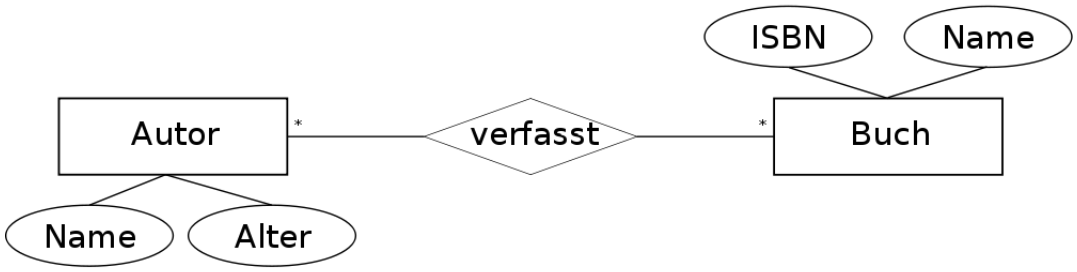

Abbildung 2.2.: Chen Notation N:M

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

2.2. Kardinalitäten

Kardinalitäten beschreiben den Grad einer Verbindung zwischen zwei Objekten[3]. In Abbildung 2.1 ist eine 1:n Kardinalitt gegeben. Diese sagt aus, dass ein Objekt der Relation 1, mehrere Objekte der Relation 2 zugeordnet werden, einem Objekt der Relation 2 jedoch nur einem Objekt der Relation 1.

In der zweiten Abbildung 2.2 ist eine n:m Kardinalitt zu sehen. Diese sagt aus, dass ein Objekt der Relation 1, mehrere Objekte der Relation 2 angehören, einem Objekt der Relation 2 werden ebenfalls mehrere Objekten der Relation 1 zugewiesen. Diese n:m Kardinalitäten müssen bei einem Datenbank Entwurf aufgelöst werden, da hier keine eindeutige Zuordnung möglich ist. Meistens lässt sich eine solche Kardinalitt wie in Abbildung 2.2 durch das Hinzufügen einer zusätzliche Tabelle, welche beide Objekte verknüpft, lösen. Ein Beispiel ist in Abbildung 2.3 zu sehen.

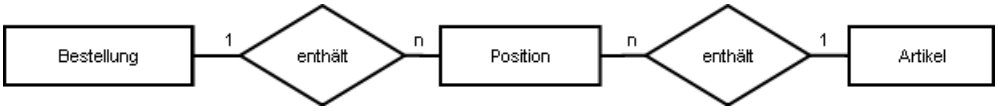

Abbildung 2.3.: Aufgelöste Chen N:M Notation

[2]In Anlehnung an In Anlehnung an Quelle **?**
[3]vgl. Heinz Burnus(2007): Datenbankentwicklung in IT-Berufen, 1. Auflage, S.20

2.3. Normalisierung - Optimierung von Datenbanken

Wenn es bereits bestehende Datenbanken gibt, so muss geprüft werden, ob diese eine optimale Struktur aufweisen. Dies kann anhand der sogenannten Normalisierung durchgeführt werden.[4] Dadurch lässt die Datenbank sich weiter optimieren.[5] Bei der Normalisierung wird abgefragt, ob Tabellen gewisse Eigenschaften besitzen und passt diese wenn nötig an. Hierzu stehen bis zu 5 Stufen der Normalformen zur Verfügung. In der ersten Normalform wird untersucht, ob jedes Attribut atomare Werte besitzt, dass heißt es enthält nur einen Wert und ist frei von Wiederholungen.[6]

	A
1	ReifenDimension
2	195/50R15

Abbildung 2.4.: Keine Normalform angewendet

In Abbildung 2.4 ist eine Verletzung der Normalform 1. zu sehen. Um diese aufzuheben müssen wir die einzelnen Werte trennen wie in Abbildung 2.5 zu sehen.

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

Abbildung 2.5.: 1. Normalform

Wurde die Relation entsprechend angepasst, so ist Normalform 1 erreicht und es kann nun geprüft werden, ob diese die Eigenschaften von Normalform 2 erfllt. Um eine Normalform zu erfüllen, müssen auch alle vorhergehende Normalformen erfllt sein, dass heißt erfllt eine Tabelle die Normalform 3, so erfllt sie auch die Normalform 1 und 2. Die 5 wichtigsten Normalformen beschreiben sich durch folgende Attribute.[7]

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

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

[6] vgl. Matthias Schubert(2007): Datenbanken, Theorie, Entwurf und Programmierung relationaler Datenbanken, 2. Auflage, S.293

[7] vgl. Heinz Burnus(2007): Datenbankentwicklung in IT-Berufen, 1. Auflage, S.292-308

1. Normalform: Alle Attribute enthalten atomare Inhalte, und die Relation hat eine feste Breite
2. Normalform: Jedes Nichtschlüsselattribut ist vom kompletten schlüssel abhängig
3. Normalform: Jedes Nichtschlüsselattribut ist von keinem schlüsselkandidaten transitiv abhängig, dass heißt kein Attribut ist ber 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: Eine weitere Aufspaltung ohne Informationsverlust ist nicht mehr möglich

2.4. DBMS

Ein Datenbankmanagementsystem organisiert die Speicherung der Daten einer Datenbank und legt die Anordnung der Daten fest. Das DBMS legt auch die Art der Beziehung fest, in der die Daten der Datenbank stehen (relational, objektorientiert). Zur Kommunikation mit diesem wird eine Sprache benötigt. Meist wird hier die deskriptive Sprache SQL verwendet.[8]

Es gibt verschiedene Arten von DBMS:

- Hierarchisch
- Relational
- Objektorientiert

Ein hierarchisches DBMS[9] dient vor allem der schnellen Suche in großen Datenbanken. 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 Datenbanken, welche heutzutage den höchsten Verbreitungsrad besitzen. 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 Datenbanken. Diese bieten die Möglichkeit Objekte aller Art zu speichern. Problematisch hierbei sind jedoch die Formulierung von geeigneten Abfragen, weswegen diese

[8] vgl. E. F. Codd(1970): A Relational Model of Data for Large Shared Data Banks in Commun. ACM, Vol 13, Nr. 6, S. 382
[9] vgl. Bernd-Jrgen Falkowski(2002): Business Computing: Grundlagen und Standardsoftware, 1. Auflage, S.235

in der Praxis eher selten und meist im Bereich von Multimedialenanwendungen anzutreffen sind.

Wird versucht verschiedene relationale DBMS in der Praxis zu vergleichen, so ist eine große Anzahl an verschiedenen Systemen zu finden. Eine kleine Auflistung soll einige Bekannte vorstellen.

- Microsoft Jet Engine (Access)
- MS-SQL Server
- Oracle
- MySQL
- PostgreSQL

Die Microsoft Jet Engine ist ein dateibasierendes DBMS, welches dem Benutzer eine einfache Möglichkeit bietet Daten in einer Datenbank zu speichern und passende Oberflchen (Frontends) in der Datenbank zu integrieren. Wie alle anderen dateibasierenden DBMS bietet es die selben Vor- und Nachteile. Bei diesem System 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 eine solchen Datenbank. Dadurch, dass Access für jeden Nutzer bei einer Abfrage die komplette Datenbank durchsucht entsteht eine hohe Auslastung der Festplatte, sowie des Netzwerks. Folglicherweise 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, weil sonst die Daten inkonsistent werden können, wenn mehrere Personen gleichzeitig einen Datensatz schreiben. MS-SQL ist ebenfalls ein relationales Datenbankmanagementsystem und in den verschiedenen Serverbetriebssystemen von Microsoft enthalten.Für Entwickler ohne Lizenz wird eine eingeschrnkte Express Version zur Verfügung gestellt. MS-SQL ist im Gegensatz zu Access kein dateibasierendes DBMS, sondern ein DBMS welches zentral 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.

MySQL ist der Open Source Pendant zu MS-SQL, welcher im Internet eine sehr hohen Verbreitungsgrad aufweist. So wird dieses von Seiten wie Wikipedia[10] oder Youtube[11] verwendet. Im Gegensatz zu MS-SQL erlaubt das Lizenzmodell, dass die Datenbank für Privatanwender kostenlos ist und die Lizenzgebhren für Unternehmen einen Bruchteil der Kosten ausmachen, die für ein Microsoft System bezahlt werden mssten.[12]

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[13].

2.5. 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 mssen.

Um dynamische Webseiten erzeugen zu können, bedarf es einer Skriptsprache. Aktuell haben sich folgende Sprachen etabliert:[14]

- ASP
- JSP
- PHP

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 erstellt wurde. Sie ist die am weitesten verbreitete Scriptsprache zum Erstellen von dynamischen Webseiten.

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

[10] vgl. Mysql, http://www.mysql.com/why-mysql/scaleout/wikipedia.html
[11] 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/
[12] Vgl http://www.mindfactory.de/product_info.php/pid/geizhals/info/p155132
[13] vgl. PostgreSQL: BSD-Lizenz, http://www.postgresql.org/about/licence
[14] vgl. Tiobe Software(2009): TIOBE Programming Community Index for August 2009, http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

- Apache
- IIS

Der Apache Webserver ist ein Opensource Webserver der Apache Foundation. Er kann unter vielen verschiedenen Betriebssystemen eingesetzt werden und untersttzt durch seine Module, alle verbreiteten Scriptsprachen, sowie Datenbanken.

Der IIS von Microsoft läuft ausschließlich unter Windows, zudem ist es auch nur unter dem Serverbetriebsystem von Windows möglich mehr als 10 Verbindungen gleichzeitig aufzubauen.[15]

2.6. Schnittstellen

2.7. Softwareentwicklung

2.8. UML

2.9. MAC - Media Access Control

2.10. VLAN - Virtual Local Area Network

2.11. SNMP

Unter dem Simple Network Management Protocol ist ein Netzwerkprotokoll zu verstehen, welches einem erlaubt Netzwerkgeräte (z.B. Drucker, Router, Switchs, Router) per Netzwerk zu überwachen und zu steuern.[16] Diese Abfragen werden von einem zentralen Punkt aus durchgeführt, dem sogenannten SNMP-Manager, welcher die Daten von den SNMP-Agenten (Netzwerkelementen) abruft.[17]
Bei SNMP handelt es sich um ein Protokoll, welches sich auf der Schicht 7, die Anwendungs-Ebene, des ISO/OSI-Schichtenmodells, ansiedeln lässt. Entwickelt wurde das Protokoll von

[15] vgl. Microsoft: IIS 7.0: Übersicht ber die verfügbaren Features in IIS 7.0, http://msdn.microsoft.com/de-de/library/cc753198%28WS.10%29.aspx
[16] vgl. Essential SNMP, S. 1
[17] vgl. Essential SNMP, S. 3

der IEFT und ist über diverse RFCs definiert. Durch die hohe Modularität ist SNMP auch unabhängig von IP und funktioniert somit auch über IPX oder AppleTalk. Dies ist mitunter auch ein Grund für die Weite Verbreitung von SNMP, welches mittlerweile als Standard gilt.
Die Funktionsweise von SNMP spiegelt sich in der Verwendung der Agenten und Manager wieder. Zunächst gibt es die sogenannten Agenten welche als Dienst auf dem jeweiligen Endgerät laufen und die Informationen zur Verfügung stellen. Diese werden dann auf einem Manager jeweils abgerufen per SNMP. Die Nachrichten werden entweder angefordert vom Manager oder aufgrund eines Ereignisses vom Agent an den Manager selbständig gesendet.
SNMP selbst definiert nicht welche Daten/Werte die Netzwerkkomponenten liefern, sondern gibt nur eine Baumstruktur vor, an die sogennante Management Information Base (MIB) angliedert. Diese beschreibt die jeweils enthaltenen Informationen und sind teilweise ebenfalls über RFCs spezifiziert.[18] 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. Diese sind wiefolgt:
GET
GETNEXT
GETBULK
SET
RESPONSE
TRAP

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.
Um diese Problematik zu lösen gibt es den sogenannten GETNEXT-Befehl, der es ermöglicht den Wert sowie die OID eines daraufolgenden Elementes zu erhalten.

[18] vgl. RFC 1213

//<Beispiel>

Die Abfrage von x Ports erzuegt x+1 Abfragen (Bei einem 48+2 Port Switch somit 51 Abfragen) ist neffektiv , da der Manager nur eine Informationsmenge erhalten möchte aber dazu eine vielzalh an Anfragen durchführen muss. Daher 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.
Das SET-Paket dient zum setzten spezieller Werte, so kann zum Beispiel darüber der Status des Portsvon einem Switch geändert werden, oder es könnte eine Firewall konfiguriert werden. Auf diese bisher genannten Pakete antwortet der Agent mit einem RESPONSE Paket, welcher die benötigten Werte oder eine Fehlermeldung enthält. Sofern beim SNMP-Agent z.B. gewisse Grenzwerte hinterlegt wurden kann dieser sich bei einer Überschreitung mittels eine Trap-Paketes beim Manager melden, ohne das dieser die Information explizit abgefragt hat.
Um möglichst wenig Netzwerklast zu erzeugen kommuniziert SNMP über das UDP Protokoll, da es eine Verbindunglose Kommunikation ermöglicht. Der Agent erhält die Anfragen auf Port 161, während der Manager auf Port 162 die Trap Meldungen empfängt.

2.12. CDP - Cisco Discovery Protokoll

//Auf CDP eingehen Das CDP von Cisco ist ein propäriteres Protokoll, welches dazu dient, Cisco-Geräten zu ermöglichen andere angeschlossene Geräte zu identifizieren und mit deren Details aufzulisten.

3. Praktische Umsetzung

3.1. Situationsbeschreibung

Die bestehende Situation 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. Hierunter fällt die Überwachung der Stati der einzelnen Netzwerk Geräte, im speziellen die Switchs und Router, welche über das Programm "OpenView" von HP erfasst werden. Über dieses Programm können ebenfalls Server und deren Dienste erfasst werden. Diese Funktion wird jedoch aktuell nicht mehr mit Daten gepflegt, da hier ein Transfer zu einem anderen System stattfindet, aufgrund der Tatsache, dass die Lizenz von OpenView ausgelaufen ist. Im Moment dient das OpenView System bei Pirelli Deutschland dazu, dass eine automatische Übersicht über die Relationen zwischen den Switchs und Routern erstellt werden kann.
Für die Überwachung spezieller Hosts (in der Praxis meist Server) und Diensten, kommt ein weiteres System zum Einsatz, welches eine Open Source Software ist und ein hohes Maß an Flexibilität und Modularität bietet. Dieses "Nagios" gennante Programm ermöglicht es diverse Dienste im Nertzwerk auf vorgegebenen Hosts abzufragen und anschließend diese 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.
//Hier nur MRTG oder auch auf Orion eingehen?
Zur Untersuchung von Netzwerk-Traffic zwischen verschiedener Netzwerke kommt das System MRTG zum Einsatz welches hauptsächlich für das anzeigen des Netzwerktraffics auf Routern ausgelegt ist. Dieses wird im Unternehmen hauptsächlich dafür verwendet, um den Traffic der Router bzw. der WAN-Verbindungen zu überprüfen. 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 "Cisco Works" welches dazu dient mit einem speziellen Programmteil die Hosts an den jeweiligen Switchs zu erkennen und zu identifizieren. Hier werden nicht nur simple Zuordnungen zwischen Mac Adresse des Hosts und des Switchs, bzw. des Port des Switches hergestellt, sondern umfangreiche Informationen gesammelt, z.B. über die Geschwindigkeit des Anschlusses oder die IP und sofern vorhanden der DNS-Hostname des Gerätes. Hinzukommen diverse Details wie der Gerätetyp, welcher ei-

nem die Möglichkeit gibt, zu erkennen ob es sich hierbei um ein Switch oder Router handelt. Da bei diesem System es notwendig ist, regelmäßig die Lizenz zu verlängern um auch weiterhing die neusten Router und Switches zu unterstützen, ist hier eine gewisse Abhängigkeit gegeben. Weil von Cisco Works selbst nur ein kleiner Teil genutzt wird und somit die Lizenzkosten unverhältnismäßig sind wurde entschieden diese in Zukunft zu ersetzten.

Die Überwachung des Netzwerks lässt sich bei näherer Betrachtung in folgende Punkte unterteilen:
Überwachung von Diensten/Servern
Überwachung des Netzwerkverkehrs
Darstellung und Überwachung des Netzwerkequipments
Überwachung von einzelnen Hosts im Netzwerk

3.2. Anforderungsdefinition

Um das bestehende System ersetzten zu können muss zunächst untersucht werden, welche der bereits vorhandenen Funktionen genutzt werden und welche in Zukunft benötigt werden.
Im Anschluss dieser Untersuchung müssen auch weitere potentielle Aufgaben berücksichtigt werden um eine Erweiterbarkeit des Systems zu garantieren. Damit das neue System alle Anforderungen gerecht wird empfiehlt es sich zunächst sich mit den betreffenden Personen zusammen zu setzten, welche das bestehende Programm zur Zeit nutzen.
Hierbei wird nicht nur auf die aktuelle Nutzung eingegangen sondern auch diese kritisch hinterfragt, ob eventuelle Anforderungen bereits von anderen Systemen erfüllt werden, oder diese den Umfang einer neuen Lösung sprengen könnten.
Durch Gespräche mit den zuständigen Mitarbeitern haben sich dann diverse Anforderungen ergeben.
Als wichtigster Punkt ist die Erkennung der Hosts, sowie der zugehörigen Ports an den Switchs zu sehen. Diese ist notwendig um zu sehen an welchem Ort im Netzwerk sich ein betreffender Host befindet. Praktische Anwendung hat dies, wenn zum Beispiel ein Host aufgefunden werden muss, wenn der zugehörige Mitschein des Computers ausläuft und dieser zurückgegeben werden muss. Hier ist es nicht selten der Fall, dass der Computer mit seinem dazugehörigen Besitzer bereit die damalige Abteilung verlassen hat und dieser somit nicht mehr auffindbar ist. Eine weiterere Anwendungsmöglichkeit ist es, dass der Helpdesk wissen möchte, wenn es Probleme mit einem Computer gibt, ob die Netzwerk-Hardware bis zum Host einwandfrei funktioniert. Hierfür musss man nicht nur wissen, an welchem Ort der Host sich befindet, sondern

auch die dazugehörigen Ports. Da die Ports als auch die Switchs selbst ebenfalls ausgelesen werden, kann man somit zeitnah eine Rückmeldung geben, dass der Switch, so wie 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 hat dies den Hintergrund, dass es vorkommen kann, dass unerwünschte Netzwerkgeräte angeschlossen wurden. Normalerweise befinden sich im Unternehmen hinter einem Port an einem Switch 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 nciht direkt an den Port des Switches angeschlossen, sondern wird über den Port eines VOIP-Telefons durchgeschleift, welches dann an den Switch angeschossen ist. Dies ist der Soll Zustand, jedoch kann es vorkommen, das 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 werden können und auch gleichzeitig lokalisiert um eine Entfernung zu ermöglichen. Ein weiterer wichtiger Punkt ist die Möglichkeit eine Historie zu haben, 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. Beispielsweise gibt es den Switch S1 und S1. Zu Beginn ist der Host, in unserem Falle ein Nobteook N1, an einen Port P1 an S1 angeschlossen. Nun Wechselt N1 zu P1 an S2. Hierfür soll nun wiederum ein Eintrag erzeugt werden. Wenn N1 jetzt wieder von P1 auf S2 auf P1 auf S1 wechselt, so soll kein neuer Eintrag entstehen, sondern nur der bereits existierende aktualisiert werden. Wechselt N1 jedoch von P1 auf S1 zu P2 auf S1, so soll beenfalls ein Eintrag erzeugt werden.

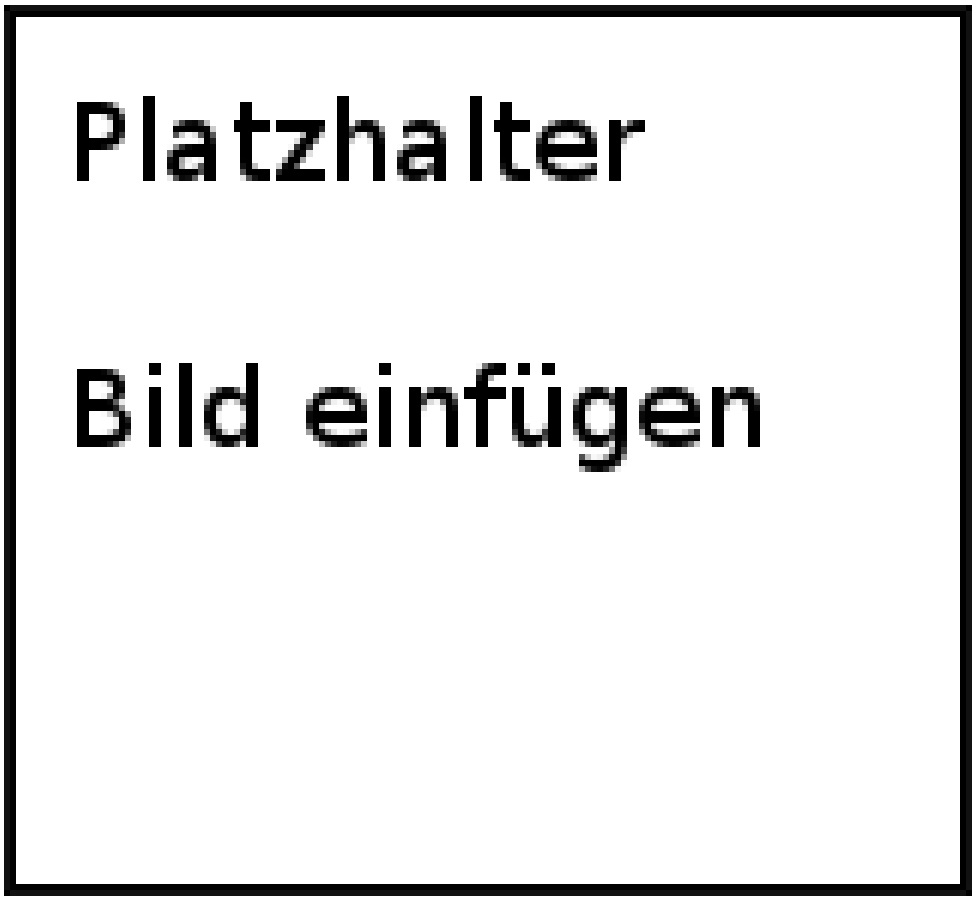

Abbildung 3.1.: Grafik mit S1+S2 sowie jeweils P1 und N1

Für das Auslesen der Switchs soll das Netzwerkprotokoll SNMP verwendet werden, da eine Abfrage per Telnet die Switchs stark belastet und somit es passieren kann das 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 den alten ungefähren Wert handelt, welcher die vorherige Auslese-Geschwindigkeit darstellt.

Eine weitere Anforderung die als optional angesehen wird, ist die Möglichkeit die Verbindungen zwischen den Switchs und Routern zu erfassen. Dies heißt, es soll eine Topologie anhand der Daten erstellt werden können.
Zu den ebenfalls optionalen Anforderungen gehört auch 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 direkt anrufen zu kön-

nen bei Problemen.
Kombiniert man diese Anforderungen so bildet sich eine weitere Anwendungsmöglichkeit und zwar kann es in der Praxis vor kommen, dass ein Switch gewechselt und somit abgeschaltet werden muss, hierfür kann man nun im Voraus kontrollieren, welche Hosts an diesen angeschlossen sind und die dementsprechenden Nutzer im Voraus zu informieren.

Neben den Anforderungen zu den Anwendungsmöglichkeiten wurden auch ein paar Anforderungen an das Programm selbst gestellt. Zu diesen gehören die Betriebsystemunabhängigkeit, das Programm muss sowohl auf Windows als auch Linux Computern laufen können. Sofern Interpretersprachen oder ähnliches verwendet werden, darf keine Versionsabhängigkeit bestehen, d.h. das Programm 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 diese 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 der Materie vertraut machen und die technischen Möglichkeiten mit den Anforderungen abgleichen.
Zunächst einmal muss untersucht werden, dass die im Unternehmen befindlichen Switchs auch das SNMP-Protokoll soweit unterstützen, um das Auslesen möglich zu machen. Dies wurde überprüft und bei vereinzelten Switchs 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 abfragtbar 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 Hersteller eigenen proprietären Protokolle. Bei der Überprüfung der Zuordnung zwischen Mac-Adresse der angeschlossenen Ports wurde festgestellt, dass diese zwar möglich ist, dies aber eine sehr spezielle Vorgehensweise vorrausetzt [vgl. SNMP Cisco PDF].
Ein Problem das sich stellt ist 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 Hierarchie Ebene angehört. Diese Informationen müssten manuell gepflegt werden oder anhand einer automatischen Hierachie Erkennung erhalten werden.

3. Praktische Umsetzung

Die Anforderung alle Hosts hinter einem Port zu finden stellt kein Problem dar, da dies von allen Switchs per SNMP unterstützt wird.
Eine Historie die es ermöglich nur neue Kombinationen aus Mac des Hosts und zugehöriger Port als neuen Eintrag zu sehen ist ohne Probleme möglich und kann durch die demensptrechende Erstellung der Datenbank sichergestellt werden.
Aussagen über die Geschwindigkeit 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. Genauere Aussagen über die Geschwindigkeit lassen sich erst durch Benchmarks machen, welche in Kapitel [Kapitel X] durchgeführt werden.
Um mögliche Verbindungen zwischen den Switchs und Routern zu Erfassen muss es möglich sein aufeinem 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 besitzt.
Die Betriebsystem Unabhängigkeit ist mit einer Vielzahl von Interpreter Sprachen (Perl, Phyton, ...) oder einer Platform unabhänigen Sprache (Java, C Sharp/Mono) gegeben.
Da für alle gängigen Sprachen diverse Datenban Anbindungen 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 hindern aber nicht an der 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 Test 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 beim Auslesen der Daten aufgebracht werden muss, wird die Zeit für die Umsetzung der Weboberfläche relativ kurz geraten. Hier muss dann im laufe des Projektes abgewägt werden zwischen einzelnen Punkten.

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 es das bestehende kommerzielle Produkt zu ersetzten und somit eine monitäre 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 komerzieller Natur als auch Open Source sind.
Bei HP Openview handelt es sich um eine kommerzielle Lösung von

- HP Openview
- Cisco Works -> Usertracking
- Nagios (Abgrenzung)
- tirtih
- Sonstige

//hier ausführen

3.5. Entscheidung zur eigenen Herstellung

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 besonderes 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 wurde aber bereits dies abgewegt und eine Machbarkeit des Systems festgestellt.

3.6. Entwurf

3.6.1. Usecases

Zu Beginn eines Projektes zur Entwicklung eines Systems müssen zuerst die Usecases identifiziert werden und die beteiligten Akteure. Hierzu wird am zu Beginn überlegt welche Personen Zugang zum System haben werden und welche Aufgaben diese am System erfüllen werden. Hierzu zieht am besten nicht nur die Anforderungsdefinition zu Rate sondern 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.

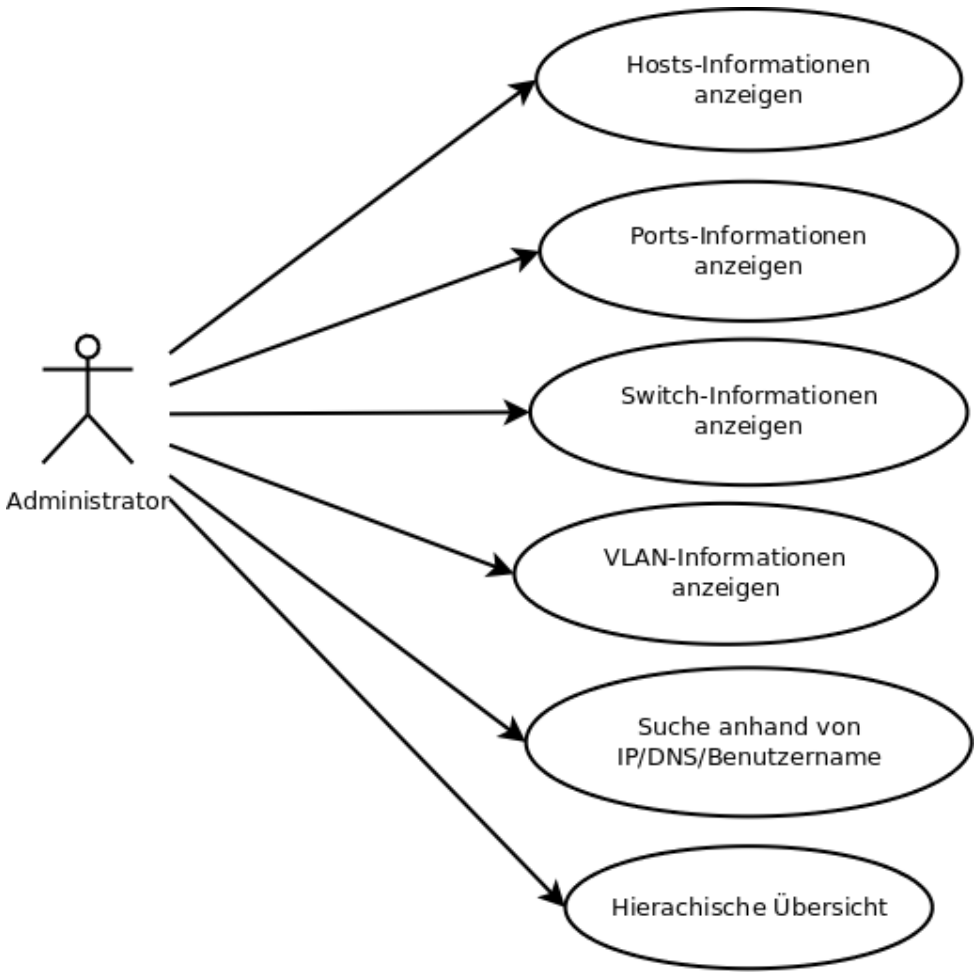

Abbildung 3.2.: Usecasediagramm

3. Praktische Umsetzung

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 Switchs, hier muss neben der Switch IP auch der per SNMP ausgelesene Alias stehen, aber auch die vorhandene IOS-Version und weitere Informationen.
In einer weiteren Sicht solle eine Übersicht auf die Ports eines Switches geben werden. Neben der MAC-Adresse des Ports, so wie des Port Namens (z.B. Fa0/1) sind weitere Informationen sichtbar. Unter anderem der Status (Up/Down), Geschwindigkeit des Ports, sofern zugewiesen die VLAN ID, Duplexmodus, aber auch Infos, 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. Zusätzlich sollen aber auch Informationen angezeigt, welche in Verbindung mit dem Port stehen, aber im Zuge der Historie zusätzlich beim Host gespeichert werden. Sofer es sich um ein CDP-Gerät handeln sollte, werden diese zusätzliche Informationen ebenfalls angezeigt. Neben den Sichten zu den Geräten wird es auch eine Übersicht der VLANs geben, welche es erlaubt eine Verbindnug herzustellen zwischen VLAN ID und des jeweiligen VLANS inkl. Name und VTP-Domain.
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 ferfügbaren 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 detailierten Informationen.
Als optionaler Usecase wir das Anzeigen einer Art Dashboard gesehen. In diesem werden Informationen zusammen gefasst die nicht direkt erkenntlich über die einzelnene Sichten sind. Beispielsweise könnte man hier Dinge anzeigen lassen, wie z.B. Anzahl freier Ports, Switch mit den meisten freien Ports, TOP 5 Switchs mit der höchsten Last, um nur eine Liste der potentiellen Informationen zu nennen.

3.6.2. Klassendiagramme

Für die Umsetzung des Projektes wurde eine objektorientierte Programmiersprache gewählt, welche es erfordert Objekte von Klassen abzuleiten und die dementsprechenden Methoden dieser Klassen bzw. ihrer Objekte zu nutzen. Für die Umsetzung des Projektes wurde mit mehr als 25 Klassen gearbeitet um eine genügend genaue Abstrahierung und Modularität zu erreichen. Im Nachfolgenden wird auf die wichtigsten Klassen und deren beinhaltete Methoden eingegangen.
Das Ausleseprogramm wird über die Klasse *SNMPTrack* initialisiert, welches auch der Name des Skripts darstellt. Diese Klasse ist sozusagen die Hauptklasse von der alle weiteren Objekte erzeugt werden. Zu Beginn des Programms wie in Kapitel X zu sehen werden die Switchs aus der XML-Datei ausgelesen. Um hier später die Möglichkeit zu geben die Informationen auch über eine andere Datenquelle einzulesen wurde eine Spezielle Klasse *SwitchListe* für das Einlesen der Switchs geschaffen. Hierzu muss nur ein Objekt der Klasse erzeugt werden, welches mit dem Befehl *getSwitchList* eine Liste der Switchs zurückliefert. Diese kann dann im Hauptprogramm direkt weiterverwendet werden ohne dort die komplette XML-Logik hinterlegen zu müssen.
Über die Klasse *SNMPConfig* können weitere Informationen ausgelesen werden die benötigt werden. So bietet diese Klasse Methoden an, welche das Debuglevel oder die Maximale Threadanzahl der Switchthreads zurückliefern. Durch diese Methoden können die Werte ohne Probleme von einer zentralen Stelle ausgelesen werden. Diese wiederum können aus einer beliebigen Datenquelle (XML, DB) kommen.
Die nächste wichtige Klasse ist die *SNMPHandler* Klasse. Sie verwaltet alle SNMP Abfragen bzw. vereinfacht den Abruf eines SNMP Wertes oder im Falle eines SNMP-Walk der Abruf einer kompletten Liste. Hierzu muss der jeweiligen Methode nur das Globale SNMP Objekt, welches aus der SNMP4J API stammt, die benötigte OID, die IP auf dem der SNMP-Agent läuft, sowie der passende Community-String zum lesen übergeben werden. Je nach Methode liefert die entsprechende Methode dann ein String oder eine Array List mit Strings als Elemente zurück. Um die Rückgabe Werte des SNMP-Agents besser zu verstehen ist das folgende Beispiel aufgeführt:

<OID>!<Wert>

Mit reellen Daten gefüllt könnte ein Rückgabewert wiefolgt sein:

1.3.6.1.2.1.1.1!Linux WRT54G 2.4.20 2 Thu Sep 9

Durch das Zusammensetzten mit ! kann eine Verarbeitung stattfinden bzw. der Wert einfach abgelesen werden.
Um einen Switch auszulesen wird ein Objekt der Klasse textitSwitchWorkerThread erzeugt und diesem das SNMP Objekt, die IP und die passende Read-Community übergeben. Dieses Objekt Startet dann den intern einen Thread, welcher ein Objekt der Klasse Switch erzeugt und die entsprechenden Initialisierungs Variablen übergibt. Im Anschluss wird die Methode refresh() des Switches aufgerufen, welches alle relevanten Informationen beginnt auszulesen und dabei wie in Kapitel X beschrieben vorgeht. In der refresh() Methode des Switches wiederum wird auf spezielle Klassen zurückgegriffen.
Eines dieser Klassen ist textitOIDL, welche für OID-Library stehen soll und alle Aliase der OID die verwendet werden enthält. Diese Aliase sind per RFC spezifiziert und 1:1 abgebildet. So ergibt z.b. der Aufruf:

OIDL.sysDescr0

den String:

1.3.6.1.2.1.1.1.0

Dies ist vor allem notwendig um eine Übersicht über die verwendeten OID zu behalten bzw. erkennen zu können welcher Funktion hinter der OID steckt. Der nachfolgende Quellcode-Aschnitt verdeutlicht die Wichtigkeit der Text-Äquivalenz zu den IDs:

3. Praktische Umsetzung

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

swCDP=SNMPHandler.getOIDWalk(snmp, OIDL.cdpCacheAddress, sIP, sReadcommunity);
swCDPC=SNMPHandler.getOIDWalk(snmp, OIDL.cdpCacheCapabilities, sIP, sReadcommunity);
swCDPDI=SNMPHandler.getOIDWalk(snmp, OIDL.cdpCacheDeviceId, sIP, sReadcommunity);
swCDPPort=SNMPHandler.getOIDWalk(snmp, OIDL.cdpCacheDevicePort, sIP, sReadcommunity);
swSpeed=SNMPHandler.getOIDWalk(snmp, OIDL.ifSpeed, sIP, sReadcommunity);
swType=SNMPHandler.getOIDWalk(snmp, OIDL.ifType, sIP, sReadcommunity);
swTypeCisco=SNMPHandler.getOIDWalk(snmp, OID.portType, sIP, sReadcommunity);
swSTP=SNMPHandler.getOIDWalk(snmp, OIDL.locIfspanInPkts, sIP, sReadcommunity);

swCiscoPort = SNMPHandler.getOIDWalk(snmp, OIDL.c2900PortIfIndex, sIP, sReadcommunity);

if(swCiscoPort.size()>0){
swCiscoDuplex = SNMPHandler.getOIDWalk(snmp, OIDL.c2900PortDuplexStatus, sIP, sReadcommunity);
```

Abbildung 3.3.: Codebeispiel

Im Programm selbst existieren jedoch auch Klassen die nicht nur zur Hilfe der eigneen Algorithmen Umsetzung dienen, sondern auch, welche externe Datenquellen anbinden. Hierzu gehört die Klasse Nagios, welche per HTTP oder per MySQL Datenbank ermöglicht die Gruppenzuordnung der Switchs auszulesen. Aber auch die Klasse LDAP, welche das auslesen des Active Directories ermöglicht oder aber die Klasse DNSHelperClass, welche es ermöglicht auf mehreren DNS Servern einen DNS Namen anhand einer IP zu finden. Neben diesen Klassen gibt es aber auch speziell erstellte Klassen die es ermöglichen SQL-Befehle die keine Resultsets erwarten, das heißt welche keinen Rückgabewert 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 von Klassen, welche zur verinfahcung der Übersicht und Wartbarkeit dienen, gibt es trotzdem Objekte welche auch die Realtät abbilden. Dazu gehören die Klassen Switch (welche bereits beschrieben wurde), Ports, sowie Hosts. Diese bieten jeweils Methoden an, welche von sich selbst die passenden SQL-Befehle generieren anhand der Daten, welche diese enthalten. Dies macht es vorallem einfach die Objekte zu serialisieren. Das heißt die Objekte im Programm wiederum abzubilden auf eine Textuale-Ebene.

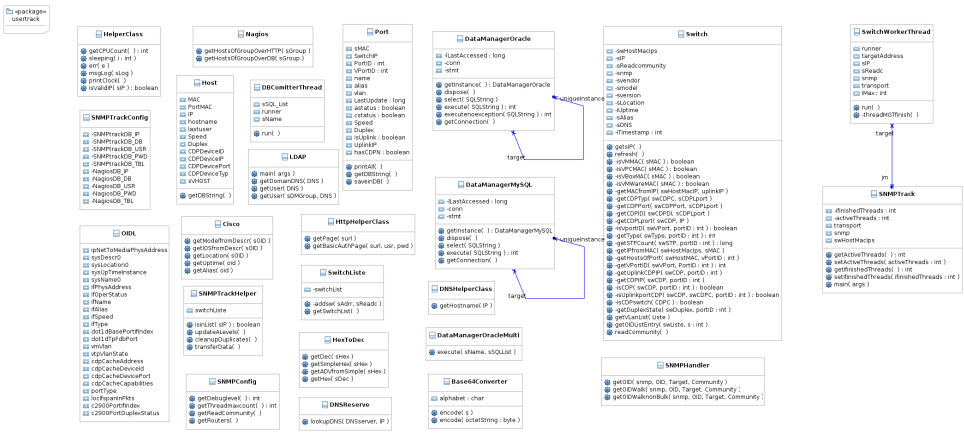

Abbildung 3.4.: Klassendiagramm

3.6.3. Sequenzdiagramme

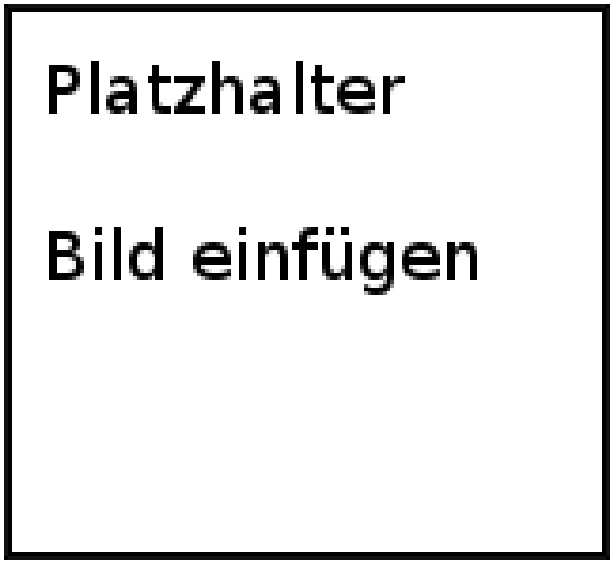

Abbildung 3.5.: Sequenzdiagramm

3.6.4. Aktivitätsdiagramme

Zur Darstellung des Programmablaufs eigenen sich vor allem Aktivitäts-Diagramme von UML. In Abbildung X ist ein solches Diagramm zu sehen.

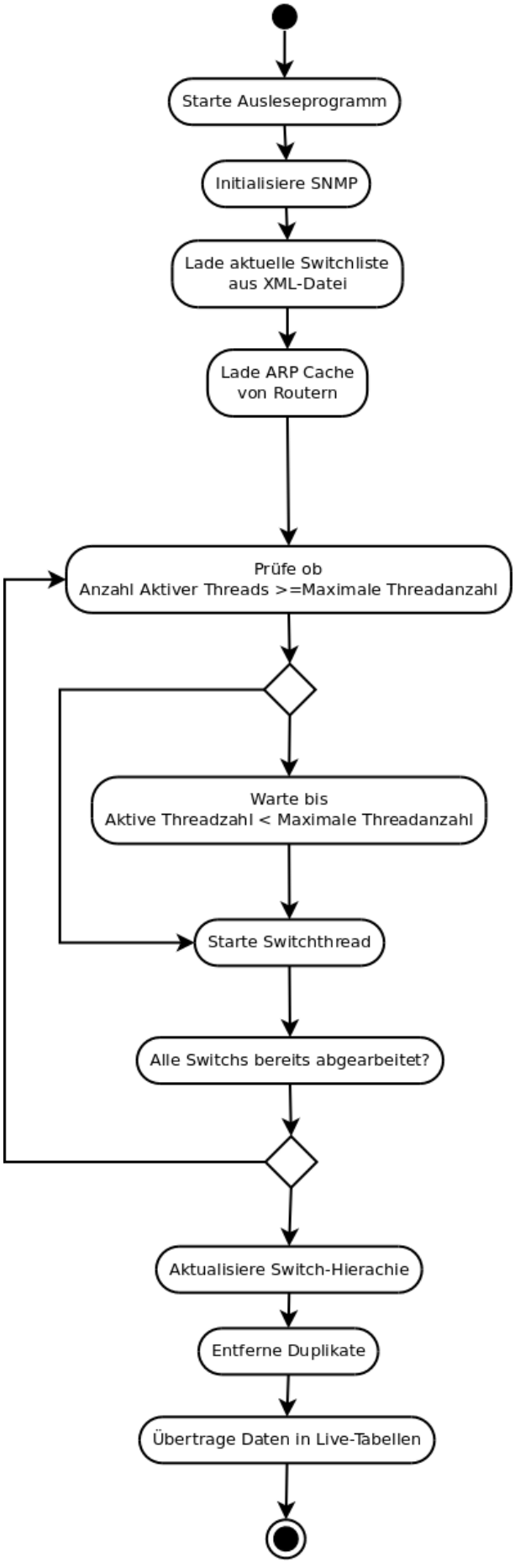

Abbildung 3.6.: Aktivitätsdiagramm1

3. Praktische Umsetzung

In dieser Abbildung ist der Ablauf des Hauptprogrammteils zu sehen. Dieser Programmteil soll im näheren angesprochen werden. Jedoch ist dabei zu beachten, das es sich bei der Darstellung um eine merkliche Vereinfachung des Programmablaufs handelt.
Zu Beginn des Programms wird die SNMP Schnitstelle initialisiert, da diese später allen Switch Threads übergeben werden muss. Im Anschluss wird die XML-Konfigurationsdatei der Switchs ausgelesen. Dies ist notwendig zum einen um die definierte Liste der auszulesenden Switchs zu erhalten, aber auch um zugehörige ReadCommunity auszulesen welche benötigt wird um Zugriff auf die jeweiligen Switchs zu erhalten. Nachdem die Switch Liste eingelesen wurde per XML, wird nun per SNMP der ARP-Cache aller relevanten Router abgefragt. Dieser wird später benötigt in den Switch Threads um eine Zuordnung zwischen MAC-Adresse und IP-Adresse herstellen zu können und anhand derer weitere Informationen wie DNS-Hostname zu erhalten. Anschließend wird in einer Schleife geprüft, ob die Anzahl der aktiven Threads das gleich oder höher der Maximalen Thread Anzahl ist. Ist dies der Fall wird eine Pause eingelegt bis die Anzahl der aktiven Threads gesunken ist und ein Thread gestartet. Dies dient dazu um sicherzustellen, dass nie mehr wie X Threads aktiv sind. Dies ist vor allem in Hinblick auf die während de Benchmarks festgestellten Probleme zurückzuführen. Während der Implementierungsphase wurde der Wert im Bereich 8-32 variiert und schließlich auf 10 gesetzt, welches als Sicherstellung dient, dass keine Verbindungen zum Oracle Server Probleme bekommen. Die Threads starten jeweils einen Switchthread-Code, welcher im nächsten Paragraphen besprochen wird. Nach dem Durchlauf der Schleife wird gewartet bis alle Threads beendet wurden, da zuerst alle Hosts in der Datenbank abgelegt werden müssen um dann eine Duplikatserkennung durchführen zu können. Zuerst werden die Switch Hierachie-Levels anhand des Nagios-Systems aktualisiert, da diese ausschlaggebend für die Einordnung der Hosteinträge sind. Im Anschluss werden die Duplikate anhand eines Algorithmuses entfernt, welcher alle bekannten Duplikate untereinander vergleicht und bei bedarf "falsche" Hosteinträge löscht. Hierbei handelt es um Host die sich auf nicht identifzierten Uplinkports befinden. Die Problematik der Uplinkport Identifizierung wurde bereits in Kapitel X angesprochen. Zum Abschluss werden die von den Duplikaten gesäuberten Daten in die sogenannten Livetabellen übertragen. Diese Übertragung ist notwendig, da es während des Auslesevorganges Inkonsistenzen im Hinblick auf die Verknüpfung der Beziehungen gibt und daher darf erst der "finale" Status in die Livetabellen übertragen werden. Durch diese Übertragung wird auch sichergestellt, dass während des Ausleseprozesses die Datenbank trotzdem nutzbar ist vom User und dieser keinen Ausfall des Systems registriert.
Im folgenden wird nun der Auslese Prozess etwas detaillierter für die Switch Threads beschrieben. Auch in diesem Fall gilt der Hinweis, dass es sich hierbei um eine starke Vereinfachung handelt, da das Abbilden der zusätzlich geschriebenen Algorithmen z.B. für das Erhalten der passenden Host-MAC-Adressen eines Ports. Die Erste Überprüfung die durchgeführt wird, ist

ob SNMP auf der entsprechenden IP verfügbar ist. Ist dies nicht der Fall wird das Auslesen des vermeindlichen Switches mit einer Fehlermeldung abgebrochen und der Thread beendet. Im Anschluss dieser Überprüfung werden alle Informationen bezüglich des Switchs ausgelesen. Das heißt Hersteller, Modell, IOS-Version, Uptime, Alias usw. . Im Anschluss wir die Switch spezifische SQL Abfrage generiert um die Informationen in die Datenbank abzuspeichern. Jedoch wird diese nicht direkt durchgeführt sondern in einem Stapel ersteinmal gesammelt, der Grund hierfür liegt in der Reduzierung der Last des Datenbankservers wie bereits in Kapitel X bei den Benchmarks entdeckt wurde, dass das Absetzten mehrerer Einträge auf einmal die Performence erhöht. Im Anschluss werden alle Port relevanten Informationen abgefragt. Nachdem die Liste der VLANs ausgelesen wurden werden die VLAN spezifischen Listen zur MAC-Adressen Zuordnung geladen. Das heißt, für jedes auf dem Switch bekannten VLAN wird eine Abfrage an den Switch gestellt, welche Host-Adressen in dem jeweiligen VLAN mit den dazugehörigen Virtuellen Ports zurückliefert. Nach dem alle SNMP-Daten nun ausgelesen und zwischengespeichert sind wird überprüft, ob der jeweilige Port eine MAC-Adresse hat und es sich hierbei auch nicht um eine virtuelle Schnittstelle handelt. Sofern dies der Fall sein sollte wird der Port ignoriert und der nächste bearbeitet. Im Normalfall wird der Bearbeitungs-Prozess anschließend fortgesetzt. Hier wird dann überprüft, ob der Port Up oder Down ist und ob dieser, sofern der Status Up ist CDP-Informationen enthält. Diesen CDP Informationen werden bei Bedarf ausgelesen und überprüft per Spanning-Tree-Protokoll ob es sich hierbei um einen Uplink-Port handelt. Im Anschluss wird die jeweilige SQL-Abfrage generiert und im Puffer abgelegt. Nachdem die Informationen des Ports bekannt sind müssen die Hostinformationen ausgelesen werden. Hierzu wird als allererstes geprüft ob ein Endgerät an den Port angeschlossen ist. Ist dies nicht der Fall, wird das Auslesen der Hostinformationen abgebrochen und alle SQL Abfragen an die Datenbank übertragen. Sind jedoch Hosts vorhanden wird jeweils unterschieden, ob es sich hierbei um ein Uplink-Port handelt oder nicht. Im Falle des Uplink-Ports werden die CDP-Daten ausgewertet und diese als Host Eintrag hinzugefügt. Handelt es um keinen Uplinkport, so werden alle Hosts ausgewertet und für jeden Host die passende IP im ARP-Cache gesucht und aufgrund dieser IP der passende DNS-Hostname rekursiv abgefragt. Im Anschluss wird, sofern es sich um ein Host mit Windows handelt das Active Directory befragt, welcher Benutzer zuletzt am genannten Host angemeldet war. Auf Grundlage dieser Daten wird wiederum eine SQL-Abfrage generiert die anschließend dem Puffer hinzugefügt 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 Absetzten der SQL-Befehle kümmert. So kann Der Switch Thread bereits beendet werden und das Auslesen des nächsten Switches beginnen, auch wenn noch nicht alle Datensätze in der Datenbank sind. Dies dient vor allem der Reduzierung der Wartezeit und somit einer Reduzierung der Auslesezeit.

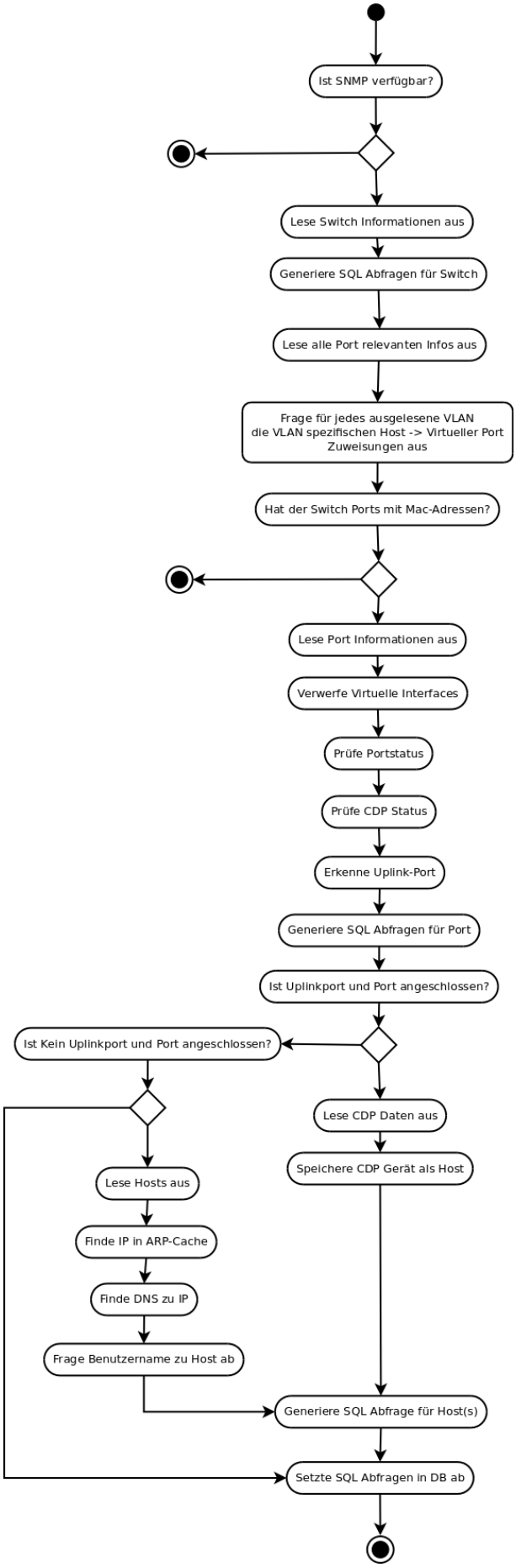

Abbildung 3.7.: Aktivitätsdiagramm2

Im folgenden wird nun der Auslese Prozess etwas detailierter für die Switch Threads beschrieben, wie er in Abbildung X zu sehen ist. Auch in diesem Fall gilt der Hinweis, dass es sich hierbei um eine starke Vereinfachung handelt, da das Abbilden der zusätzlich geschriebenen Algorithmen z.B. für das Erhalten der passenden Host-MAC-Adressen eines Ports. Die Erste Überprüfung die durchgeführt wird, ist ob SNMP auf der entsprechenden IP verfügbar ist. Ist dies nicht der Fall wird das Auslesen des vermeindlichen Switches mit einer Fehlermeldung abgebrochen und der Thread beendet. Im Anschluss dieser Überprüfung werden alle Informationen bezüglich des Switchs ausgelesen. Das heißt Hersteller, Modell, IOS-Version, Uptime, Alias usw. . Im Anschluss wir die Switch spezifische SQL Abfrage generiert um die Informationen in die Datenbank abzuspeichern. Jedoch wird diese nicht direkt durchgeführt sondern in einem Stapel ersteinaml gesammelt, der Grund hierfür liegt in der Reduzierung der Last des Datenbankservers wie bereits in Kapitel X bei den Benchmarks entdeckt wurde, dass das Absetzten mehrerer Einträge auf einmal die Performence erhöht. Im Anschluss werden alle Port relevanten Informationen inkl. der zugehörigen MAC-Host Liste abgefragt.

3.6.5. ERM

Aufgrund der vorherigen Analysen ist ein Enitity Relationship Model 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.

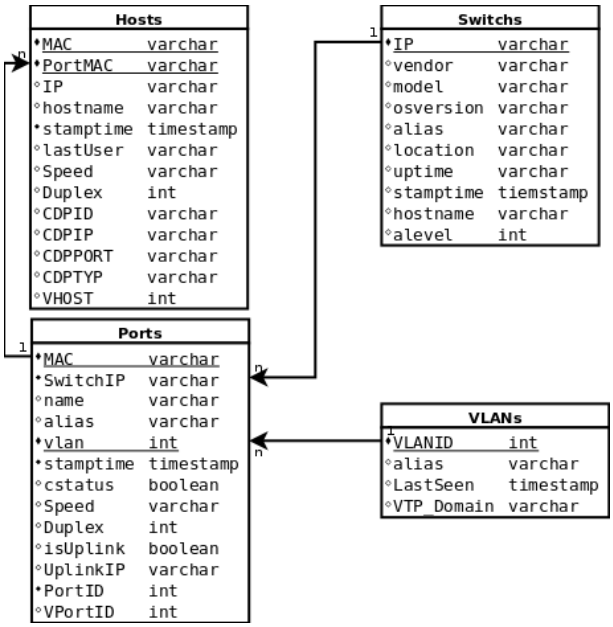

Abbildung 3.8.: ERM

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 Primary Key VLANID, welcher für jedes VLAN im Netwerk eindeutig ist und somit auch als PrimaryKey geeignet ist. Über diesen Schlüssel werden alle VLAN relevanten Informationen abgelegt, 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 Switchs 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 Dinge wie IOS-Version, Modell, Ort, Uptime, aber auch Hostname, sofern vorhanden und die entsprechende Hierachiestufe, welche anhand de Nagios-Systems ausgelesen wurde.
In der Port Tabelle werden alle Ports der Switchs gespeichert. Die Ports selbst sind eindeutig über ihre jeweilige MAC-Adresse identifizierbar, jedoch soll ein neuer Eintrag angelegt werde, sofern sich die VLAN Einstellung ändert. Daher dient nicht nur die MAC-Adresse als Primary Key, sondern auch die VLAN ID. Neben dem Namen des Ports und der MAC-Adresse gibt es auch ein Foreign Key, welcher die IP des angehörigen Switch enthält um eine eindeutige Zuordnung des Ports an einen Switch zu ermöglichen. Zusätzlich sind Attribute des Ports bezüglcih des Status untergebracht. So ist zum Beispiel zu sehen ob der Port gerade Up oder Down ist, mit welcher Geschwindgkeit 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 befindelichen 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 eneuer Eintrag angelegt werden muss, daher wurde nicht nur die MAC-Adresse de Hosts sondenr auch die MAC-Adresse des Ports am Switch als Primary Key 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 Switchs, 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 ein 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 das z.B. das spezielel VMWare Esxi Server betriebsystem selbst über eine virtuelle Schnittstelle kommuniziert, also das physikalische Host-System selbst und nicht die Guest-VMs.

3.7. 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 Programmier Sprache gewählt wird. Nach dem Abgleich der Anforderungen und mit Absprache 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, die einen Switch mit wenig Netzwerk-Traffic zum Zweck eines Benchmarks mit einer Vielzahl von SNMP Abfragen beschäftigt.
Im Benchmark-Programm wird die Zeit gemessen wie lange es dauert um 1000 Abfragen durchzuführen. Der in Milisekunden gemessene Wert wird in die nachfolgende Formel eingesetzt:

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

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

In der nachfolgenden Grafik sind die beide Programmiersprachen aufgezeichnet:

Abbildung 3.9.: Benchmark - Perl, Java

Wie sich in der Grafik feststellen lässt liegt Java ein Bruchteil vor Perl, der geringe Unterschied zwischen den beiden Programmiersprachen lässt sich dadurch erklären, dass die Implementierung des SNMP Abfragen minimal anders sind, der bei beiden fast gleich hohe Wert lässt darauf schließen, dass der Flaschenhals des Benchmarks der Router selbst ist. Dies konnte dadurch validiert werden, dass 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 Switchs abgefragt werden müssen, empfiehlt es sich eine Parallelisierung anzustreben, da jeder Switch nur eine begrenzte Geschwindigkeit an Requests aufweist. Hierfür eignet sich die Verwendung von Threads um den Teil der Abfragen zu parallelisieren der unabhängig voneinander Laufen kann. Um den Nutzen einer Parallelisierung zu

erkennen wurde wiederum ein Benchmark durchgeführt. Diesmal wurde die Anzahl der Threads variiert und jeweils wiederum die Zeit gemessen.

Abbildung 3.10.: Benchmark - Parallelisierung

Wie in der Grafik 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 was ein maximalwert pro Switch später darstellen sollte.
Bei der Durchführung dieses Benchmarks wurde aber zugleich ein Problem erkannt und zwar kommt es zu Problemen bei Thread-Zahlen über 20, da hierbei der UDP-Puffer für die SNMP Abfragen nicht mehr ausreicht und somit der Empfang aller SNMP-Pakete nicht mehr sicher gestellt ist, daher empfiehlt es sich bei der Implementierung ein niedrigeren Wert zu wählen und auf Nummer sicher zu gehen.
Da es bei SNMP-Protokoll der Version 2 ein speziellen Abfrage Modus "BULKGET" gibt musste auch überprüft werden, in welchem Umfang dieser dem normalen GET ein Geschwindigkeitsvorteil gibt. Hierfür wurden der erste Benchmark angepasst und die Liste der Ports von einem Switch abgefragt, einmal alle Ports sequentiell und einmal per BULK.
Die Ergebnisse des Benchmarks lassen sich im nachfolgenden Bild erkennen.

Abbildung 3.11.: Benchmark - SNMP Bulk

Es lässt sich feststellen, dass der BULK Modus einen deutlichen Geschwindigkeitsvorteil gibt. 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 genaustens beachtet werden, wann welche Methode eingesetzt werden kann.

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

Abbildung 3.12.: Benchmark - Oracle Transaktionen

Wie in der Grafik zusehen ist es von Vorteil die Commits zu sammeln und erst dann abzusetzten. Während des Benchmarks wurde jedoch erkannt, dass bei einer Vielzahl von vorgehaltenen Abfragen, die durch die Parallelisierung entstehen, die Anzahl der gleichzeitig offenen Tabellen Einträgen das Limit des Datenbank überschreitet. Zwar wurde anschließend das Limit der maximal offenen Einträge erhöht, jedoch kam es selbst dann zu Überschreitungen, daher wurde der Algroithmus so angepasst, dass er immer in 100 Abfrage-Paketen die Abfragen absetzt. Durch diese Maßnahme ist es trotzdem möglich einen Geschwindigkeitsvorteil zu erhalten, bei gleichzeitiger Einhaltung der Limits.

3.8. Auswahl der Hilfsmittel

Für die Umsetzung des Projektes sind verschiedene Hilfmittel notwendig.
Zum einen wird eine Entwicklungsumgebung benötigt. Da die Wahl der Programmiersprache auf Java gefallen ist und kommerzielle Lösungen nicht zur Auswahl gehören wurde sich für Eclipse entschieden, da es neben NetBeans zu den einzigen Open Source IDEs gehört welche eine Vielzahl von Erweiterungsmöglichkeiten bietet und interaktive Funktionen wie Code-Vervollständigung oder Code-Vorlagen unterstützt aber auch standardmäßige Dinge wie Syntaxhervorhebung.

Zur Softwareversionverwaltung wurde Subversion verwendet. Dies ist eines der in der Praxis sehr weit verbreiteten Systeme. Zusätzlich gibt es noch CVS, Git und viele weitere.

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 fiehl speziell auf Subversion, da es im Gegenzug zu CVS das Versionsschema nicht auf einzelene 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. Somit wäre kein Zusammenhang erkennbar. Bei Subversion hingegen kann dies direkt erkannt werden, da die neue Funktion in der Revision 33 in beiden Dateien erkennbar ist.

3.9. Schnittstellen

Um ein Projekt mit vielen Einlese und Export Funktionalitäten erstellen zu können, wird eine Vielzahl an 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 koordiniert werden.
Im folgenden wird auf die jeweiligen Schnittstellen eingegangen, wie sie sequentiell im Ablauf des Ausleseprogramms verwendet werden, diese sind im folgenden zusammengefasst:
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 anzutreffen. Hierbei muss der Benutzer dem Programm eine Großzahl von Informationen übergeben.
Es existiert eine spezielle XML-Datei in der die Switchs, welche ausgelesen werden müssen, die Router, welche den ARP-Cache enthalten, sowie die notwendigen Zugangsdaten für die Datenbank(en), enthalten sind.
Das Programm selbst bietet dem Nutzer keine Möglichkeit Übergabe Parameter zu definieren, um einen möglichst einfachen Ablauf zu ermöglichen und zusätzliche Fehleingaben des Benutzers zu vermeiden.
Die nächste Schnitstelle ist die Kommunikation mit den Switchs per SNMP. Um nicht die komplette SNMP-Kommnuikation selbst in per UDP implmentieren zu müssen emfiehlt es sich hierbei eien bereits vorhandene API zu nutzen.

Speziell für Java gibt es hierfür mehrere Möglichkeiten (keien vollständige Liste):
SNMP4J
jSNMP
WebNMS SNMP
iReasoning SNMP API
netsnmpj

Schließt man nun alle kommerziellen Lösungen aus, so bleiben lediglich SNMP4J und netsnmpj übrig. Da merhere SNMP Abfragen später gleichzeitig durchgeführt werden müssen ist die Wahl auf SNMP4J gefallen, da dieses Threadsicher ist und auch ein größeren Funktionsumfang wie netsnmpk bei der Implementierung mitsichbringt.
Um anhand der ermittelten IP-Adressen die dementsprechende DNS-Namen zu erhalten ist es notwendig ein Reserve DNS Lookup zu machen. Hierfür muss das sogennante 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 Reserve 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 eigneschränkt 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)

3. Praktische Umsetzung

Hiervon interessiert uns vor allem das LDAP-Verzeichnis, welches uns ermöglichen soll 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 welches es ermöglicht unabhängig von der eingesetzten Datenbank, die Anbindung an das Programm immer auf die gleiche Art zu realisieren zu können. Da diese Unabhängigkeit dadurch erreicht wird, dass Befehle erst in die Datenbankspezifischen umgewandelt werden müssen und somit ein Overhead entsteht, ist diese Lösung meist langsamer wie eine native Lösung. Daher ist es von vorteil spezielle vom Hersteller angebotene JDBC Treiber einzusetzten, 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 einsetzt 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.

3.10. Zeitplan

		Name	Dauer	Start	Ende	Vorgänger
1		Augabenstellung	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 3.13.: Zeitplan - Arbeitspakete

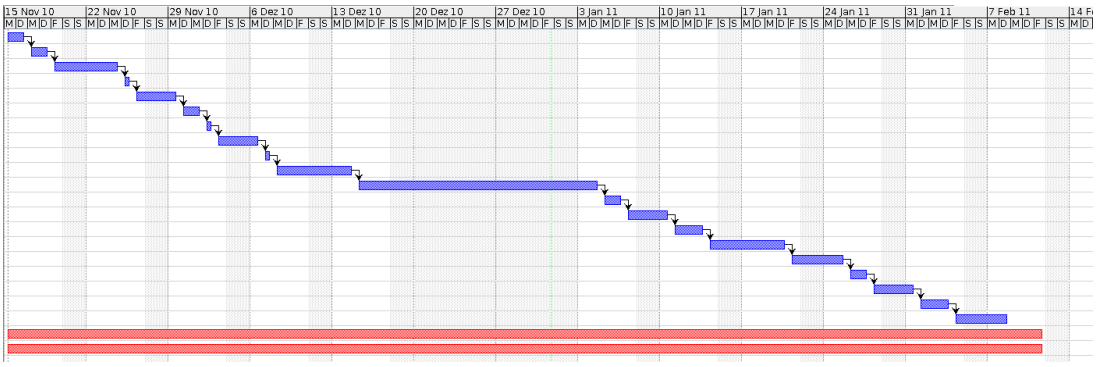

Abbildung 3.14.: Zeitplan - Ganttdiagramm

//Zeitplanung ansprechen

3.11. Realisierung

//hier das Ergebnis eingehen

3.12. Probleme bei der Implementierung

Während der Implementierung traten verschiedene Probleme auf, welche berücksichtigt werden mussten in der Implementierung, um abschließend trotzdem die gewünschten Anforderungen alle Umsetzen zu können. Die Probleme lassen sich vor allem zu den folgenden Punkten Einordnen:
Zuordnung Host –> Switchport
Verwendete VLANs (“unsichtbare 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 erhalten wurden nicht denen gleichen, die lokal auf dem Switch per Telnet über das Cisco System zu erhalten waren. So kann das Cisco System einem direkt anzeigen welche MAC Adressen sich hinter welchem Port verbergen. Über SNMP sind diese Informationen jedoch nicht direkt erhaltbar, hier müssen erst verschiedene “Tabellen” die Abgerufen werden verknüpft werden. Als aller erstes müssen die verwendeten VLANs des Switchs abgefragt werden (zur Problematik der VLAN Abfrage im nächsten Paragraphen mehr). 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 gibt per SNMP eine Liste welche einem 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 fesgelegt 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 dann trotzdem an die MAC Adresse des VOIP-Telefons zu bekommen bleibt einem nichts anderes übrig, als alle im Netzwerk bekannten VLANs an einem Switch abzufrage, 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.
Eine weitere Problematik stellt sich in der Identifizierung der Uplink-Ports an den Switchen. Hier gibt es keine einfache 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 zu Identifizierung aus. Das nächste Attribut welches ausgewählt wurde ist ein spezielles Cisco spezifisches Protrokoll 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 antowrten, welche weder Switchs 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 das dieses nicht vollständig ausreicht, da nicht alle Switchs alle Dinge unterstützen, somit musste nach einer zusätzlichen Identifikationsmöglichkeit gesucht werden. Hier wurde auf das Spannign Tree Protokoll gestoßen, welches zur Kommnuikation zwischen den einzelnen Switchs dient 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 ermöglicht.
Ein weiteres Problem hat die Einordnung der Hierachie Ebenen 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 realen Port. Hierzu muss man wissen, ob der Switch ein Access-Switch ist oder ein Switch einer höheren Ebene.
//das folgende sollte besser zur Situationsbeschreibung
Die Einordnung der Switch in die verschiedenen Hierachie-Ebenen ist wiefolgt:
Access-Switch
Distribution-Switch
Core-Switch

//besser als Grafik

3. Praktische Umsetzung

Die Hosts im Netzwerk sind mit einer geringen Anzahl von Ausnahmen, alle an Access-Switche angeschlossen. Zusätzlich gibt es Endgräte die auch an die Distribution Switche angeschlossen sein können, z.B. das Rechenzentrum.

Um diese Problematik lösen zu können wurde überlegt, die Hierachie Ebenen jeweils numerischen Äquivlanten Zahlen zuzuordnen. Hierzu wurden die Core Switchs mit der Zahl 0, die Distribution Switches mit der Zahl 1 und die Access Switchs mit der Zahl 2 definiert.
Da die Information der Ebene des Switchs nicht direkt von diesen selbst auslesbar ist, gibt es effektiv nur zwei Möglichkeiten. Zum einen kann man die komplette Hierachie anhand der Uplink Ports mit einem Programm logisch ab bilden und manuell einen Hauptknoten auswählen anhanddessen dann eine Einordnung der Switches erfolgt, was jedoch äußerst komplet ist und den Zeitlichen Rahmen de Projektes sprengt, daher wurde auf die zweite Möglichkeit zurückgegriffen, nämlich diese Informationen extern von einer anderen Quelle einzulesen.
Hierzu 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 Gruppen Informationen der zu kontrollierenden Hosts, in unserem Fall der Switchs ausgelesen werden. So sind die Switchs in verschiedene Host-Gruppen einsortiert, unter anderem auch in die benötigten Access, Distribution und Core Hierachien. Über diese Gruppen werden die jeweiligen Hosts bzw. Switchs in diesem Falle erfasst und deren Hierachielevel in der Datenbank eingetragen.
Neben den hauptsächlich logischen Probleme zur Identifikation stellte sich unter anderem noch ein weiteres Problem heraus bei der Abfrage der Informationen per SNMP.
Wie bereits in vorherigen Kapitel angedeutet, werden die auszulesenden Switchs besonders belastet bei SNMP Abfragen, weshalb eine Erhöhung der Anzahl der Abfragen stehts kritisch bewertet wurde in den Ausführungen. In den Beschreibung zur Zuordnung zwischen MAC-Adresse des Hosts und der Mac Adresse des Swicths 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 um ein hundertfaches überschritten wird beim Ausleseprozess. Es muss auch bedacht werden, dass die Switchs nicht für solche Operationen haupsächlich konzipiert wurden, sondern vielmehr für ihre eigentliche Aufgabe, insofern ist das Ziel die Last auf den Switchs möglichst gering zu halten, um den Netzwerk-Betrieb in keinster Weise zu beeinflussen. Aufgrund eines logischen Fehlers im Programmablauf kam es während diverser Tests dazu, dass einer der Core-Switchs übermäßig ausgelastet war und dies dazu geführt hat, dass das Überwachungsystem der Switchs anschließend Warnungen versen-

det hat. Solche Fälle dürfen im Betrieb nicht passieren, da die Ursache für die Hohe Last der Switchs 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.

3.13. Tests

Im Zuge der Implementierung müssen diverse Tests durchgeführt werden. Hierzu kommen neben dem obligatorischen Test der kompletten Implementierung, Test 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 diverse Tests an, welche diverse APIs überprüft haben. Diese dienten dazu sicherzustellen, dass eine Kommunikation mit den benötigten Schnittstellen erfolgreich ist. Darunter sind Tests gefallen, wie die Überprüfung von SNMP4J, aber auch die des Programmcodes zum Reserve-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 dieses Test war, dass die Programmiersprache als solche keinen Einfluss auf die Auslesegeschwindgikeit 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 wirkt im bezug auf Anzahl der verwendeten Threads und Laufzeit. 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 gesetzt wurde. Neben der Parallelisierung wurden auch die Auslesemethoden von SNMP selbst überprüft. So wurde getestet, ob eine Ge-

schwindigkeitsverbesseung erreicht wird, wenn der SNMP-Bulk Modus, welcher in Version 2c spezifiziert wurde, verwendet wird. Das Ergbnis dieses Tests war eine fünf-fache[19] 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
Wobei N wiederum die Anzahl der maximal aktiven Threads darstellt und M die maximale Anzahl der SQL-Befehle die 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

[19] hier korrekten Wert nachtragen

3. Praktische Umsetzung

Zahl mit dem Standard Wert von 50[20], so lässt sich feststellen, dass der Wert einer gesammelten Durchführung den Standard-Wert um ein vielfaches überschreitet. Orcale selbst empfiehlt den Wert zu erhöhen und liefert teilweise [21] 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 ausgeholfen haben wurden während der Implementierung durchgängig Tests durchgeführt den Funktionsumfang sicherzustellen 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. Im groben müssen folgende Programmteile getestet werden:

Webapplikation:
-Hostinformationen anzeigen
-Portinformationen anzeigen
-Switchinformationen anzeigen
-VLANinformationen anzeigen
-Suche anhand von IP/DNS/Benutzername
-Hierachie 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.

[20] vgl. http://download.oracle.com/docs/cd/B19306_01/server.102/b14237/initparams138.htm http://wiki.oracle.com/page/OPEN_CURSORS
[21] vgl.

3.14. Weitere Anwendungsfelder / Datamining

3.15. Wirtschaftliche Betrachtung

4. Fazit

Noch leer.

Literaturverzeichnis

A. Anleitung zur Fehlerbehebung

Noch leer.

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich:
1. dass ich meine Projektarbeit mit dem Thema

Systementwicklung eines Usertracking-Systems bei 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 3. Januar 2011

Denis Hamann