Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Boris Böttcher

Netzwerkbasiertes, zuverlässiges low level Wiederherstellungs- und Rekonfigurationsverfahren für Netzwerkgeräte:
Design und Implementierung

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Boris Böttcher

Netzwerkbasiertes, zuverlässiges low level Wiederherstellungs- und Rekonfigurationsverfahren für Netzwerkgeräte: Design und Implementierung

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Technische Informatik
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Ing. Franz Korf
Zweitgutachter: Prof. Dr. rer. nat. Stephan Pareigis

Abgegeben am 28 Mai 2008

Boris Böttcher

Thema der Bachelorarbeit

Netzwerkbasiertes, zuverlässige low level Wiederherstellungs- und Rekonfigurationsverfahren für Netzwerkgeräte: Design und Implementierung

Stichworte

Wiederherstellung, Rekonfiguration, Sicherung, Automation, Remote Labor, Netzwerkgeräte, Router, Switch, Kommandozeile, Konsole, seriell, Zustandsautomaten, XML, SNMP, zuverlässig

Kurzzusammenfassung

Im Rahmen dieser Bachelorarbeit wird für die Wiederherstellung und Konfiguration von Netzwerkgräten, die über eine Kommandozeilenschnittstelle verfügen, das Automationssystem FLASH entwickelt. Der Einsatz erfolgt in einer Remote Labor Umgebung für Netzwerkgeräte. Die Vorteile des Systems liegen in der schnellen und zuverlässigen Bearbeitung der bereitgestellten Aufgaben. Die Aufgaben sind als Automaten beschrieben, deren Verhalten in XML definiert wird. Zur Laufzeit von FLASH können die Definitionen bearbeitet oder hinzugefügt werden. Der Zugriff auf die Funktionen des Systems erfolgt über eine relationale Datenbankschnittstelle.

Boris Böttcher

Title of the paper

Network based, reliable low level recovery- and reconfiguration process for network devices: Design and implementation

Keywords

Recovery, Reconfiguration, Storaging, Automation, Remote Lab, Networkdevices, Router, Switch, Commandline, Console, serial, State Machines, XML, SNMP, reliable

Abstract

Within this bachelor thesis the automation system FLASH is implemented for the configuration of network devices, which are controlled over the command line interface. It is deployed in a Remote Lab environment for network devices. The System advantages are fast and reliable processing of allocated tasks. The tasks are described as state machines, whose behavior is defined in XML. The definitions can be edited or added during the runtime of FLASH. The System offers the features via a relational database interface.

Danksagung

Ich bedanke mich an dieser Stelle bei meinen Kommilitonen, die zu Freunden geworden sind und gemeinsam den Weg durch das Studium gegangen sind.

Mein Dank gilt insbesondere Christian Pflüger, mit dem ich fast das ganze Studium als Team gemeistert habe.

Des Weiteren danke ich Jan Raddatz für die erfolgreiche Zusammenarbeit im letzten Studienjahr.

Den Mitstreitern und Mitdenkern auf der Überholspur Sebastian Gregor, Erik Kunz, Olaf Rempel und Carsten Schulz danke ich ganz besonders für die gemeinsame Bewältigung der Herausforderungen und die hervorragende Zeit.

Professor Franz Korf danke ich für die Betreuung und die Unterstützung dieser Bachelorarbeit, die in der Firma Fast Lane durchgeführt wurde.

Ein außerordentlicher Dank geht an die Korrekturleser dieser Arbeit, die den roten Faden an der einen oder anderen Stelle gerade gezogen haben und so zum Gelingen dieser Bachelorarbeit beigetragen haben.

Vielen Dank!

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

1 MOTIVATION

Die Firma Fast Lane ist ein Schulungsanbieter für High-End-Netzwerkthemen, der weltweit seine Dienstleistungen anbietet. Die Schulungen setzen sich aus zwei Teilen zusammen. Auf der einen Seite wird die Theorie im Vorlesungsstil vermittelt und durch Diskussionen sowie Frage und Antwort Sequenzen abgerundet. Auf der anderen Seite werden praxisorientierte Lerneinheiten angewendet, um das Erlernte umzusetzen. Für den praktischen Anteil wird spezielle Hardware benötigt, die dem jeweiligen Thema entspricht. Die Hardware wird über die sogenannten Remote Labore bereitgestellt.

Diese Remote Labore setzen sich aus verschiedenen Netzwerk Geräten wie Router, Switches und Servern zusammen und werden in 19“ Serverschränken montiert. Für jeden Schulungstyp existiert ein spezielles Remote Labor, das komplett fertig verkabelt und vorkonfiguriert ist. Die individuellen Remote Labore sind in Deutschland an drei Standorten (Hamburg, Berlin und Frankfurt) in firmeneigenen Rechenzentren installiert.

Bis zum Jahr 2005 wurden die Geräte zum Teil in mobilen 19“ Racks verschickt, die in Transportkisten, den sogenannten Roll Cases, montiert waren. Die Vorbereitungen und die Basiskonfigurationen der Geräte wurden vor Ort von den Dozenten der Schulungen durchgeführt. Die mobilen Labore wurden durch die zentrale Installation der Geräte in den Remote Laboren aufgelöst. Der Hauptgrund ist die Versorgung aller eigenen, weltweit durchgeführten Schulungen durch Fast Lane mit diesen Remote Laboren. Ein weiterer Grund ist die weltweite Vermietung der Remote Labore an Partner und andere Schulungsanbieter.

Abbildung 1.1 zeigt den weltweiten Einsatz der Labore, die von jedem Ort über das Internet erreicht werden können. Dieser Einsatz macht es aus zeitlicher Sicht nicht möglich die Geräte nach einer Schulung in Australien abzubauen, dann direkt nach Europa zu verschicken, dort wieder aufzubauen und dann im Anschluss vor Ort für eine Schulung zu nutzen. Die Grenze einer dezentralen, von Transporten abhängigen Lösung ist erreicht gewesen.

Das Betreiben von Laboren in verschiedenen Zeitzonen innerhalb der gleichen Schulungswoche wird als Timesharing bezeichnet. Ein Beispiel für das Timesharing ist der Einsatz derselben Hardware in Sydney (Australien) von 9:00 bis 17:00 Uhr australischer Zeit und im Anschluss in Berlin (Deutschland) von 10:00 bis 17:00 Uhr deutscher Zeit. Dazu wird den Dozenten und Teilnehmern ausschließlich für Ihre Zeitspanne der Zugriff auf die Geräte gewährt.

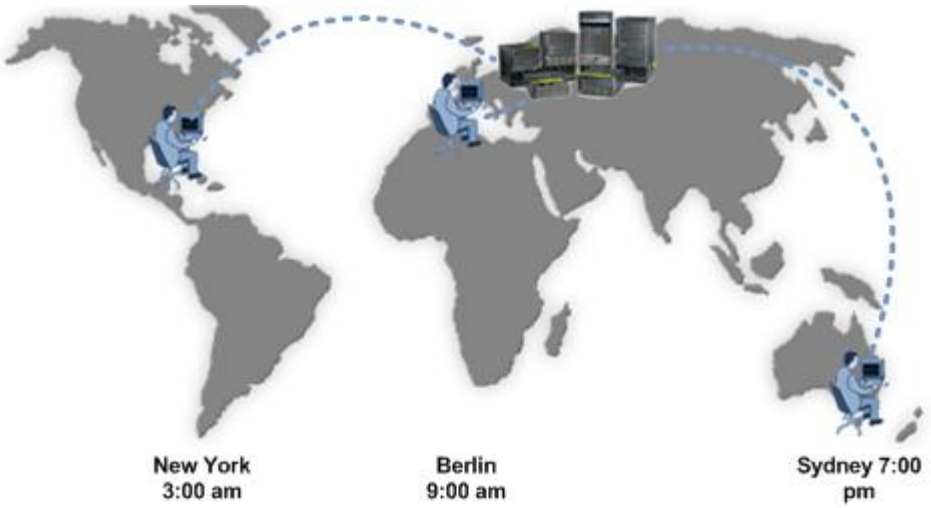

Abbildung 1.1: Globaler Zugriff auf die Remote Labore

Wie in ***Abbildung 1.2*** zu sehen ist, besteht ein Remote Labor aus einer Vielzahl verschiedener Geräte. Dabei kann ein Gerät bis zu 50 kg wiegen und ein einzelnes Labor, wie das in ***Abbildung 1.2*** dargestellte Cisco Certified Internetwork Expert (CCIE) Labor, aus bis zu 100 Geräten zusammengesetzt sein. Diese Tatsache stellt aufgrund der Anzahl und des hohen Gesamtgewichtes der Geräte einen zusätzlichen Hinderungsgrund für die dezentrale, auf Transporten basierende Lösung dar.

Abbildung 1.2: Remote Labor CCIE - Berlin

Der Zugriff auf die Labore und die Kommunikation mit den einzelnen Geräten erfolgt über ein grafisches Webinterface. ***Abbildung 1.3*** zeigt den Zugriff von einem Teilnehmer PC über das Internet auf ein Gerät (Router A) des Remote Labors. Der Teilnehmer arbeitet dabei direkt auf der Konsole[1] des Netzwerkgerätes.

[1] Konsole (Kommandozeile): Eingabebereich für die Steuerung des Betriebssystems der Netzwerkgeräte, welches im Textmodus läuft (engl. Command Line Interface CLI).

Die Dozenten und Teilnehmer haben auf den einzelnen Geräten der Labore volle Root- und Administrationsrechte, da sie sonst zwingend erforderliche Konfigurationsschritte nicht durchführen können. Dies führt dazu, dass sich die Geräte am Ende einer Schulung in einem nicht vorhersagbaren Konfigurationszustand befinden. Soll eine weitere Schulung auf demselben Labor durchgeführt werden, müssen die Geräte vorher in einen definierten Anfangszustand versetzt werden. Dieses Zurücksetzen wird momentan von den Technikern der Rechenzentren manuell nach festen Vorgaben durchgeführt.

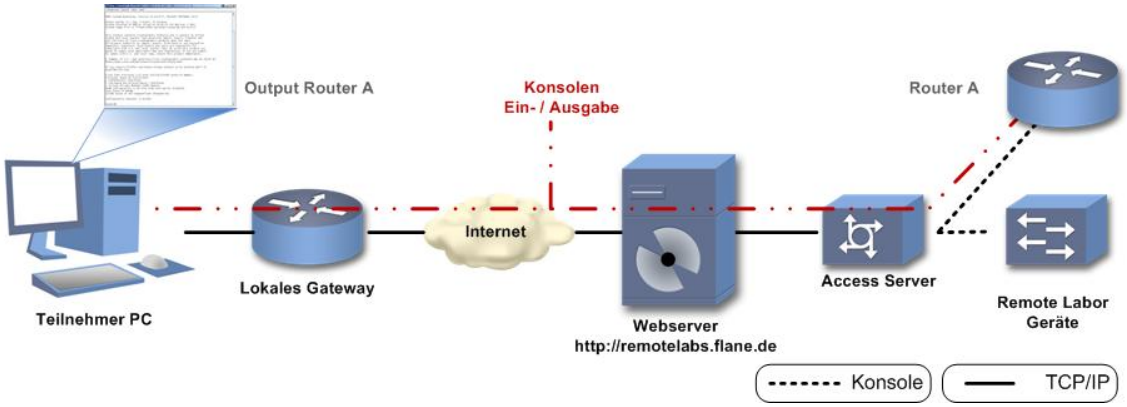

Abbildung 1.3: Teilnehmer Zugriff auf ein Remote Labor Netzwerkgerät

Die kontinuierlich steigende Anzahl der Remote Labore und der damit wachsende Aufwand für die Wartungsaufgaben haben zu der Frage geführt, eine Lösung für die Automation dieser grundlegenden Konfigurationsaufgaben zu finden. Aus diesem Grund fiel die Entscheidung ein Automationssystem zu entwickeln und zu implementieren, das *F*ast *L*ane *A*utomation *S*ystem for Networking *H*ardware (FLASH). Das Design und die Implementierung von FLASH werden im Rahmen dieser Bachelorarbeit umgesetzt.

1.1 ZIELSETZUNG

Das primäre Ziel dieser Arbeit ist die Ausarbeitung des Designs und die Implementierung von FLASH, welches in der Lage ist, die einzelnen oder Gruppen von Netzwerkgeräten der Laborumgebungen automatisch in einen vordefinierten Zustand zurück zu setzten. Des Weiteren soll es eine Möglichkeit bieten, Konfigurationsaufgaben automatisch ablaufen zu lassen. FLASH muss zuverlässig arbeiten und den Technikern entsprechende Statusinformationen bereitstellen. Das System soll schnell arbeiten und die Fehleranfälligkeit durch manuelles Arbeiten vermeiden. Dadurch spart FLASH Arbeitszeit der Techniker ein.

Weitere Ziele ergeben sich aus den entstehenden Möglichkeiten, die die angestrebte Lösung verfügbar macht. Zum einen kann der Status der Konsolenverbindungen (vgl. Abschnitt ***3.2.2.1***) von und zu den Geräten ausgelesen werden. Zum anderen können zusätzliche Wartungsarbeiten durchgeführt werden, wie zum Beispiel das Auslesen der geräteinternen Dateisysteme, um ungewollt verbleibende Sitzungsdateien zu löschen.

Sind die automatisierten Möglichkeiten implementiert, lassen sich diese zudem im laufenden Schulungsbetrieb anwenden. Dadurch kann ein Dozent während der Schulung die entsprechenden Funktionen über das Webinterface ausführen. Das automatische Einspielen einer neuen Konfiguration auf die Geräte, um dann mit der entsprechenden Übung fortfahren,

ist nur ein Beispiel der Möglichkeiten. Dadurch wird den Dozenten die sonst notwendige Zeit für die Durchführung der Konfigurationen erspart.

Des Weiteren soll das zu entwickelnde System herstellerunabhängig auf eine Vielzahl von anwendbar sein, die über eine Kommandozeilenschnittstelle verfügen. Das System ist dabei unabhängig von dem verwendeten Befehlssatz und somit universell einsetzbar. Im Rahmen dieser Arbeit werden die Funktionalitäten exemplarisch für ein Remote Labor implementiert. Dafür wurde das Labor für den Cisco Grundlagenkurs *I*nterconnecting *C*isco *N*etwork *D*evices 1 (ICND1) ausgewählt. Das Labor besteht aus Routern (Modelle C-1841 und C-3640) sowie Switchen (Modelle C-2950 und C-2960). Die Entscheidung für dieses Labor und die verwendeten Geräte ist gefallen, da die Eigenschaften der Geräte alle notwendigen Anforderungen für die Übertragbarkeit auf weitere Kurse abdecken, sodass nach Abschluss dieser Arbeit das Gesamtsystem auch auf die anderen Labore erweitert werden kann. In Kapitel ***4*** werden die Anforderungen detailliert spezifiziert.

1.2 PROBLEMSTELLUNG

Der Zugriff auf die Geräte stellt ein Hauptproblem bei der Entwicklung von FLASH dar. Der einzige, zuverlässige und immer verfügbare Zugriff auf die Netzwerkgeräte ist die Verbindung über den integrierten, seriellen Konsolen Anschluss. Jeder Router und Switch in den Laboren verfügt über diesen Anschluss, über den auf die Kommandozeilenschnittstelle des Betriebssystems zugegriffen wird. Der Anschluss ist für die Grundkonfiguration der Geräte vorgesehen, wenn sie sich im Auslieferungszustand befinden und als Hintertür bei einem Fehlerfall des Systems. Mithilfe einer speziellen Rücksetzprozedur (vgl. Abschnitt ***3.4***) wird der Zugriff auf die Geräte, unabhängig von der aktuell laufenden Konfiguration, hergestellt. Dies kann von den Teilnehmern nicht unterbunden werden, obwohl sie mit Root Rechten auf den Geräten arbeiten. Es ist nicht möglich von den Netzwerkgeräten über den Konsolen Anschluss eine ausgehende Kommunikation herzustellen, da es sich um eine uni-direktionale Verbindung handelt.

Zu den Geräten in den Remote Laboren existiert keine direkte TCP/IP Verbindung. Die einzelnen Labore sind als geschlossene und geschützte Systeme aufgebaut, da die Teilnehmer die Rolle des Administrators übernehmen und alles frei konfigurieren können. Es ist technisch nicht möglich, eine Kommunikation oder Verbindungen aus diesen Systemen heraus aufzubauen. Dies schließt den Einsatz einer kommerziellen Lösung aus.

Kommerzielle Netzwerk Management Software Lösungen wie zum Beispiel Cisco‘s CiscoWorks (siehe (Cisco, 2008)), HP’s OpenView (siehe (HPOpenView, 2008)) oder IBM’s Tivoli (siehe (IBM, 2008)) Software Pakete scheiden als Alternativen aus. Sie sind für den Einsatz in Firmen Netzwerken vorgesehen. Dort besteht immer eine Verbindung zwischen den Geräten auf einer höheren Netzwerkschicht, meist auf TCP/IP Basis. Die Managementstation mit entsprechend installierter Software befindet sich dabei in demselben logischen Netzwerk. ***Abbildung 1.4*** zeigt ein vorhandenes Netzwerk, in das eine kommerzielle Netzwerk Management Software Lösung auf der Management Station integriert wurde.

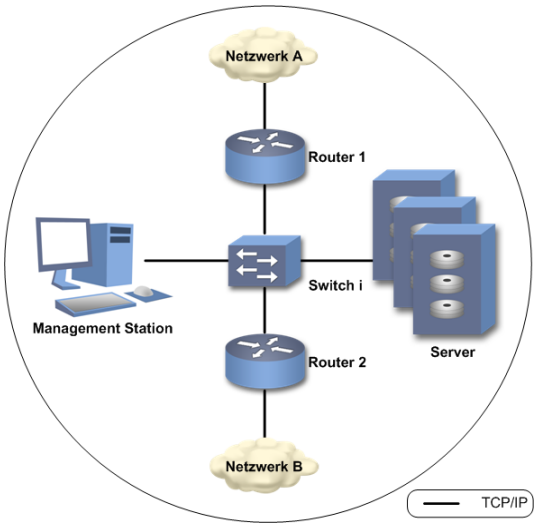

Abbildung 1.4: Typisches Einsatzszenario

Die Labor Geräte werden über ihren Konsolen Anschluss angesprochen. Dafür werden sie an einen sogenannten Access oder Terminal Server (siehe Abschnitt ***3.2.1.3***) angeschlossen. ***Abbildung 1.5*** zeigt diesen Aufbau. Die eingesetzten Terminal Server sind mit den Konsolen Anschlüssen der Netzwerkgeräte verbunden und ermöglichen den Zugriff auf die Netzwerkgeräte über ihren Eigenen, per TCP/IP konfigurierten Ethernet Anschluss. Zu den Terminal Servern besteht daher eine direkte IP Verbindung. Der Zugriff auf die Netzwerkgeräte erfolgt über ein spezielles Protokoll namens Reverse TELNET[2]. Mithilfe dieses Protokolls wird ein gesicherter, uni-direktionaler Zugriff auf die Netzwerkgeräte zur Verfügung gestellt. An einen einzelnen Terminal Server können bis zu 128 Netzwerkgeräten angeschlossen werden. In Kapitel ***3*** wird dies ausführlich erläutert. Bei der Entwicklung muss der Zugriff über diese Server berücksichtigt werden.

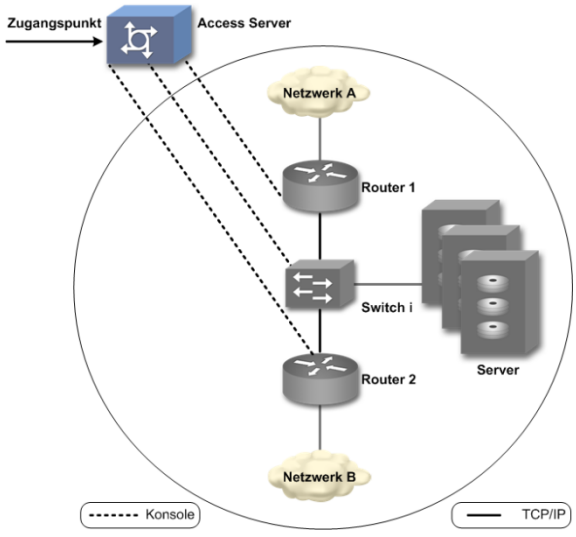

Abbildung 1.5: Spezielles Einsatzszenario

[2] Reverse Telnet: Spezielles Protokoll, das über die Serverseite der Verbindung Daten an eine serielle Verbindung schickt.

1.3 GLIEDERUNG DER ARBEIT

Diese Arbeit gliedert sich in acht Kapitel. In Kapitel *2* wird als Einführung der Einsatz der Remote Labore beschrieben und die damit verbundenen Wartungsarbeiten, die durch das zu entwickelnde System automatisiert werden dargestellt. Im folgenden Kapitel *3* werden zum einen die technischen Grundlagen und Begriffe erläutert. Zum anderen wird eine Gegenüberstellung der Konkurrenzsysteme, die von anderen Remote Labor Anbietern entwickelt wurden, durchgeführt. Abschließend folgt in diesem Kapitel ein Überblick über die Funktionen von kommerziellen Netzwerk Management Software Systemen und deren Relevanz für diese Arbeit. In Kapitel *4* werden die Anforderungen der Firma Fast Lane sowie die einer im Rahmen dieser Bachelorarbeit durchgeführten Dozentenbefragung analysiert und zusammengefasst. Kapitel *5* befasst sich mit der Erstellung des Designs von FLASH, das auf den Analyseergebnissen von Kapitel *4* aufbaut. In Kapitel *6* wird die Implementierung und Realisierung des Designs erläutert und das entwickelte System vorgestellt. Die Ergebnisse der Entwicklung werden in Kapitel *7* ausgewertet. Abschließend fasst Kapitel *8* die einzelnen Abschnitte der Arbeit zusammen und zieht ein Fazit über die Durchführung dieser Arbeit. Mit einem Ausblick auf die Weiterentwicklung von FLASH wird diese Bachelorarbeit abgeschlossen.

2 EINFÜHRUNG - AUSGANGSSITUATION

In diesem Kapitel werden die Abläufe der Schulungen und der damit verbundene Einsatz der Remote Labore erläutert. Dies geschieht auf der einen Seite aus Sicht der Dozenten und Teilnehmer und auf der anderen Seite aus Sicht der Techniker in den Remote Laboren. Des Weiteren werden die Struktur und der Aufbau der einzelnen Labore erläutert. Im Anschluss folgt die Beschreibung des Ablaufs und Zugriffs auf die Remote Labore aus Dozenten und Teilnehmer Sicht. Dies geschieht anhand des Interconnecting Cisco Network Devices 1 (ICND1) Labors, welches für diese Arbeit relevant ist. Zum Schluss wird die Wiederherstellungs- und Rücksetzprozedur für die Netzwerkgeräte am Ende einer Schulung beschrieben.

2.1 ABLAUF DER SCHULUNGEN

Die Firma Fast Lane betreibt Remote Labore verschiedener führender Netzwerkhersteller wie Cisco System und NetApp. Mithilfe dieser Labore ist es möglich den praktischen Teil der Schulungen im Bereich der Netzwerktechnologien durchzuführen. Diese Schulungen werden im Allgemeinen für einen wöchentlichen Turnus eingeplant. Die Kurse beginnen am Montag und haben in der Regel eine Länge von 3-5 Tage mit einer täglichen Dauer von 8 Stunden. Die Hälfte dieser Zeit wird Theorie im Vorlesungsstil unterrichtet und in der anderen Hälfte werden praktische Übungen auf den Remote Laboren durchgeführt.

Die praktischen Übungen sind in einem Übungshandbuch, dem sogenannten Lab Guide, detailliert beschrieben. Dieser Lab Guide ist nach Themen und Aufgabenbereiche unterteilt. In jedem dieser Bereiche werden die Aufgaben in einzelnen Schritten formuliert, die die Teilnehmer zu lösen haben. Für die Bearbeitung dieser Aufgaben greifen die Teilnehmer dann auf die einzelnen Geräte der Labore zu. Die Dozenten und Teilnehmer haben volle Root und Administrationsrechte auf den Geräten, da sie sonst zwingend erforderliche Konfigurationsschritte nicht durchführen können. Dies führt dazu, dass sich die Geräte am Ende einer Schulung in einem nicht vorhersagbaren Konfigurationszustand befinden. Am Ende eines Kurses muss jedes einzelne Gerät des Labors wieder auf einen definierten Zustand zurückgesetzt werden, sodass die Folgeschulungen mit den gleichen Ausgangsbedingungen starten können. Zudem wird so die Funktionsfähigkeit und die Erreichbarkeit der Geräte nach einer Woche Schulungsbetrieb sichergestellt. Diese zeitaufwendigen Wartungsarbeiten werden zurzeit manuell von den Technikern der Remote Labore nach festen Vorgaben durchgeführt.

2.2 STRUKTUR UND AUFBAU DER REMOTE LABORE

Für jeden Kurs gibt es ein spezielles Remote Labor. Diese Labore sind fest in den Rechenzentren der Firma Fast Lane installiert. Beim Installieren werden die Geräte auch

kursspezifisch verkabelt, um so die notwendigen Voraussetzungen für die Übungen zu schaffen. ***Abbildung 2.1*** zeigt einen Ausschnitt eines installierten und verkabelten Labors.

Abbildung 2.1: Remote Labor Installation - Ausschnitt

Ein Labor ist in zwei unterschiedliche Bereiche aufgeteilt. Zum einen in den sogenannten Core und zum anderen in den sogenannten Pair of Delegates (POD) Bereich. Ein POD bezeichnet ein Teilnehmer Paar, welches die Übungen als Team durchführt. Die einzelnen PODs bilden die kleinste funktionale Einheit für ein Teilnehmerpaar.

Der Core Bereich stellt die zentrale Infrastruktur eines Labors dar. Auf die Geräte, die den Core Bereich bilden, hat in den meisten Fällen nur der Dozent Zugriff. Der Dozent konfiguriert die Core Geräte für jede Übung des Lab Guides entsprechend.

Alle PODs eines Labors sind mit dem Core Bereich verbunden. Die PODs sind physikalisch identisch und unterscheiden sich lediglich in der logischen Konfiguration. Bei einigen Laboren sind zudem auch die PODs untereinander verbunden. ***Abbildung 2.2*** zeigt den schematischen Aufbau eines Labors mit einem Core und zwei PODs.

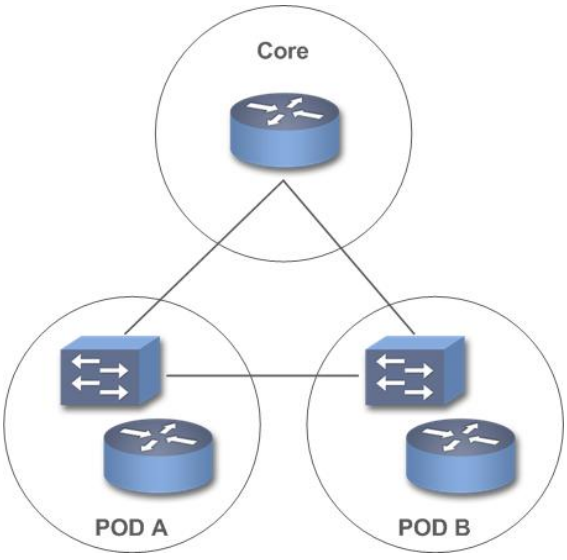

Abbildung 2.2: Core und POD Schema

2.3 ZUGRIFF AUF DIE REMOTE LABORE

Der Dozent und die Teilnehmer sitzen zusammen in einem Seminarraum und bekommen am Anfang der Schulung ein Dokument mit den Zugangsdaten für ihr entsprechend zugeteiltes Labor. Der Zugriff auf das Labor und die Kommunikation mit den einzelnen Geräten erfolgt über ein grafisches Webinterface. ***Abbildung 1.3*** zeigt den schematischen Zugriff von einem Teilnehmer PC über das Internet auf ein Gerät (Router A) des Remote Labors.

Die Teilnehmer starten Ihren lokalen Internet Browser und verbinden sich zu der URL http://remotelabs.flane.de. Dort nutzen sie Ihre Zugangsdaten für die notwendige Anmeldung. Nach erfolgreichem Login erscheint die Web Trainingsoberfläche.

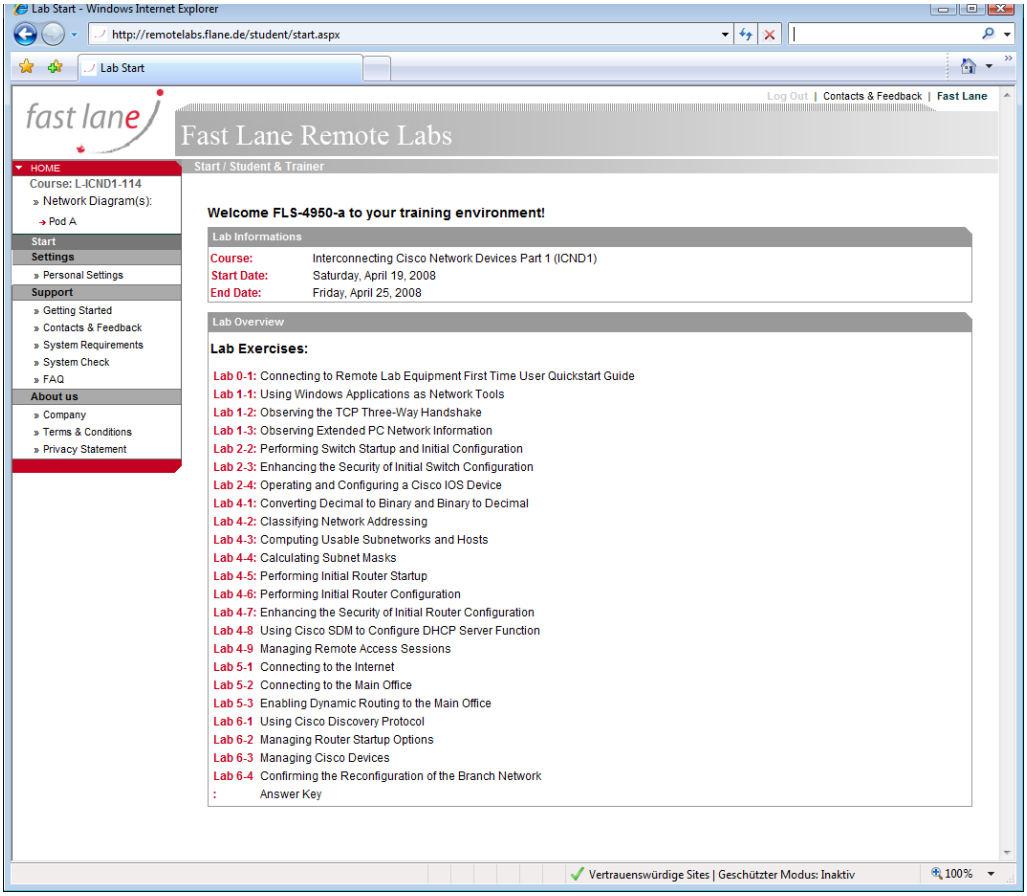

Abbildung 2.3: Teilnehmer Startseite des Webinterface

Die Teilnehmer sehen eine Zusammenfassung der einzelnen Übungen auf der Startseite. Im Menü auf der linken Seite gibt es ein paar Links zu Kontakt- und Supportinformationen, zu den Systemvoraussetzungen und einer Einführung in das System. Der zentrale Menüpunkt hat die Bezeichnung Netzwerk Diagramm. Nach dem Anklicken erscheint ein Diagramm, wie in ***Abbildung 2.4*** zu sehen ist, welches den physikalischen und logischen Aufbau des verwendeten Labors widerspiegelt.

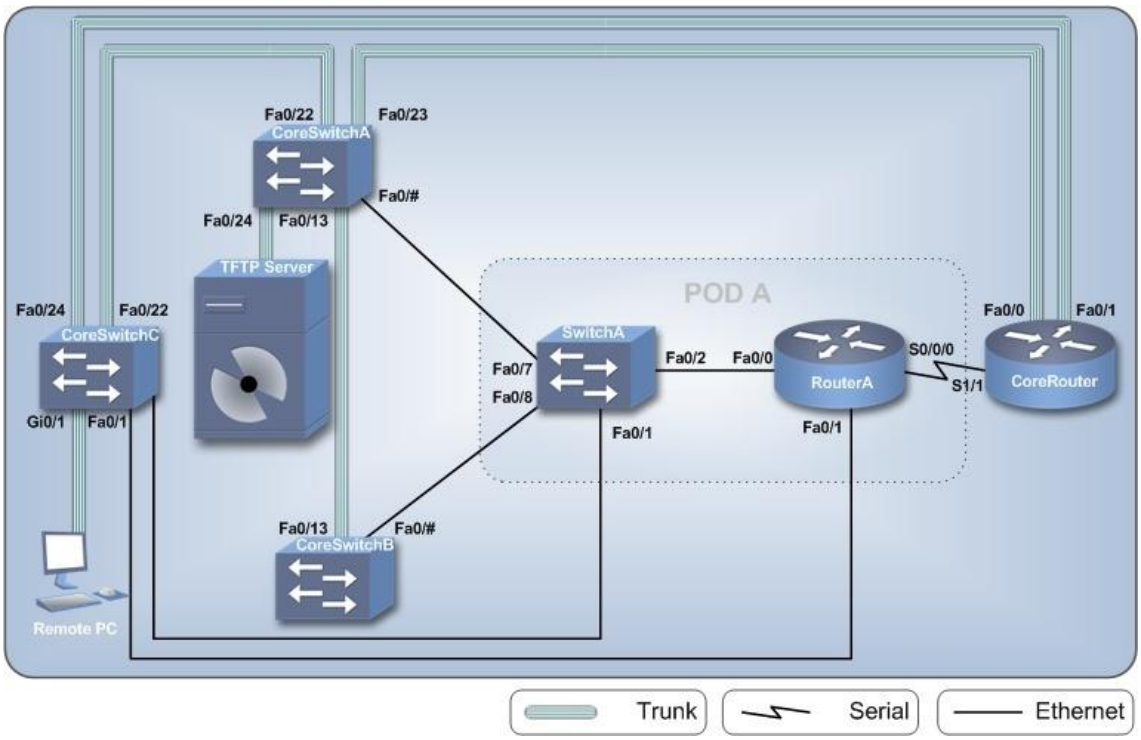

Abbildung 2.4: Netzwerk Diagramm ICND1 Remote Labor

Durch einen Klick auf eines der Gerätesymbole öffnet sich ein weiteres Fenster mit einer webfähigen Java Anwendung, dem Fast Lane TELNET Client. Dieser Client stellt die eigentliche Konsolenverbindung zum Gerät her. Der Teilnehmer kann hier Eingaben zur Konfiguration der Geräte tätigen, um somit seine Übungen erfolgreich abzuschließen.

2.4 BEISPIEL EINER PRAKTISCHEN ÜBUNG

Die praktischen Übungen sind immer nach demselben Schema aufgebaut. Zu jedem theoretischen Themenbereich gibt es eine Übung. Die Übung setzt sich aus mehreren Schritten zusammen.

Ein Schritt aus einer Übung mit dem Namen „Grundlegende Router Konfiguration“ könnte folgendermaßen aussehen:

Schritt 1: Weisen Sie Ihrem POD Router für die Fast Ethernet 0/0 Schnittstelle eine IP Adresse zu.

Der Teilnehmer gibt daraufhin die notwendigen Kommandozeilenbefehle über den TELNET Client ein. Somit kann er die gewünschten Einstellungen für das Gerät festlegen und seine Übung durchführen. Die folgenden Zeilen zeigen die dafür notwendigen Befehle.

```
Router#
Router#configure terminal
Enter configuration commands,one per line. End with CNTL/Z.
Router(config)#interface fastEthernet 0/0
Router(config-if)#ip address 10.0.0.1 255.0.0.0
Router(config-if)#end
Router#
```

2.5 WIEDERHERSTELLUNG UND RÜCKSETZPROZEDUREN AM ENDE EINER SCHULUNG

Da der Teilnehmer jede beliebige Konfiguration für das Gerät einstellen kann, müssen alle Geräte der Labore nach ihrer Nutzung wieder in einen definierten Ausgangszustand versetzt werden. Aktuell wird dies anhand einer stichpunktartigen Anleitung durchgeführt. ***Abbildung 2.5*** zeigt die Überführung der Arbeitsabläufe in ein Ablaufdiagramm. Für jedes Gerät muss dieser Ablauf einmal bearbeitet werden. Zuerst muss sich ein Techniker die Geräteliste des jeweiligen Labors nehmen und sich nacheinander zu den Geräten mithilfe des TELNET Client verbinden. Der Techniker versucht, sich anzumelden. Wurde das Gerät während der Schulung durch einen Teilnehmer mit einem Benutzernamen und Passwort gesichert, das nicht dem Übungsstandard entspricht, muss der Techniker eine bestimmte Rücksetzprozedur für das Gerät durchführen. Diese Prozedur nennt sich „Password Recovery“ und stellt einen Mechanismus zur Verfügung, über den man das Gerät starten kann, ohne die aktuell gespeicherte Konfiguration zu laden. In Abschnitt ***3.4*** wird diese Prozedur genau erläutert. Nach Durchführung der Rücksetzprozedur kann der Techniker zusätzlich eine Konfiguration in das Gerät einspielen, sodass die Ausgangskonfiguration für die entsprechende Schulung hergestellt ist. Zum Abschluss muss das Gerät noch einmal neu gestartet werden und der Techniker kann sich wieder abmelden.

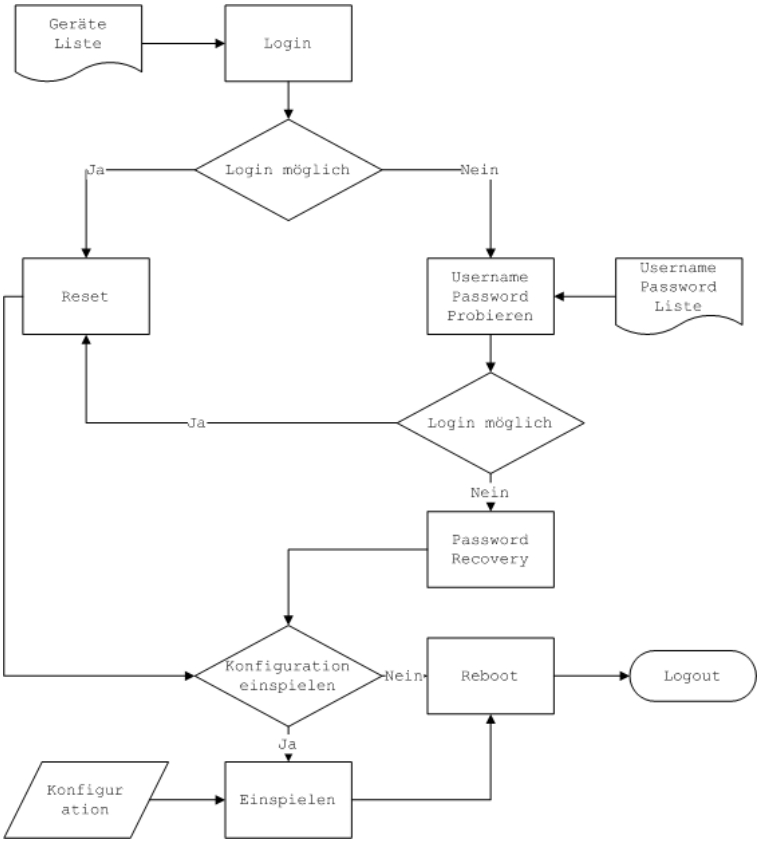

Abbildung 2.5: Arbeitsablauf - Zurücksetzen

Das manuelle Zurücksetzten der Remote Labore hat zwei Nachteile, die in den Abschnitten ***2.5.1 Zeitlicher Aufwand*** und ***2.5.2 Fehleranfälligkeit*** erläutert werden.

2.5.1 ZEITLICHER AUFWAND

Die Gesamtzeit, die benötigt wird, um die Geräte der Labore in einen definierten Ausgangszustand zu versetzten, ist abhängig von der Anzahl der zur Verfügung stehenden Techniker. Um den Zeitaufwand zu verdeutlichen, ist dieser in ***Tabelle 2.1*** für einen Techniker dargestellt, der für das Zurücksetzen eines 8 POD ICND Labors für 16 Teilnehmer notwendig ist.

Tabelle 2.1: Manuelles Zurücksetzen - Zeitaufwand für ein ICND Labor

Gerätetyp	Anzahl pro Labor	Ø Zeitaufwand in Minuten	Gesamtzeit in Minuten
Router C-1841	8	2	16
Router C-3640	1	3	3
Switch C-2950	3	3	9
Switch C-2960	8	3	24
		Summe	**51 Minuten**

Für alle 10 ICND Labore steigt der Zeitaufwand um den Faktor 10, so dass rechnerisch (vgl. ***Tabelle 2.2***) für alle Labore dieses Typs fast 9 Arbeitsstunden notwendig sind. Die parallele Bearbeitung mehrerer Geräte beim manuellen Arbeiten bringt einen zeitlichen Vorteil, sodass sich die Gesamtbearbeitungszeit um bis zu 50% reduzieren lässt. Bei einer wöchentlichen Vollauslastung aller ICND Labore ergibt sich ein Gesamtaufwand von einem halben Arbeitstag und das nur für die Labore des Grundlagenkurses.

Tabelle 2.2: Manuelles Zurücksetzen - Zeitaufwand für alle ICND Labore

Labortyp	Anzahl Labore	Ø Zeitaufwand in Minuten	Gesamtzeit in Minuten
ICND	10	51	510
		Summe	**510 Minuten**

Insgesamt bietet die Firma Fast Lane ca. 65 verschiedene Labor Typen an und unterhält insgesamt fast 100 physikalische Remote Labore mit steigender Tendenz.

Tabelle 2.3 zeigt den Zeitaufwand, der notwendig ist, wenn jede Woche alle Labore in einen definierten Zustand versetzt werden. Unter Berücksichtigung maximaler Parallelarbeit ergeben sich 45 Stunden also fast 6 Manntage Arbeit für die Techniker pro Woche. Der Einsatz von FLASH wird diesen Arbeitsaufwand minimieren.

Tabelle 2.3: Manuelles Zurücksetzen - Zeitaufwand für alle Labore

Anzahl Labortypen	Anzahl Labore	Ø Zeitaufwand in Minuten	Gesamtzeit in Stunden
65	100	60	100
		Summe	**100 Stunden**

2.5.2 FEHLERANFÄLLIGKEIT

Die manuelle Bearbeitung der Rücksetzprozeduren führt zu einer Vielzahl von Fehlern. Zum einen sind dies Flüchtigkeitsfehler bei den sich ständig wiederholenden Arbeitsabläufen und zum anderen Fehler, die durch Zeitdruck entstehen, wenn zwischen zwei aufeinanderfolgenden Schulungen nur ein kleines Zeitfenster liegt. Die Folgen zeigen sich dann bei der Auswertung von Bewertungsbögen, die von den Teilnehmern nach jedem Kurs ausgefüllt werden. Die Teilnehmer zahlen eine sehr hohe Gebühr für die Teilnahme an den Kursen. Daher erwarten sie fehlerfrei vorbereitete Labore für ihre Übungszeiten. Ist dies nicht der Fall, dann spiegelt sich die Unzufriedenheit in den Bewertungsbögen wider. Die Auswertung dieser Bögen wird von den Herstellern der Netzwerkgeräte durchgeführt, da sie für die Qualität der Schulungen zuständig sind. Fallen die Bewertungen über einen bestimmten Zeitraum untern einen vorgebenden Schnitt kann dies weitreichende Konsequenzen für die Firma Fast Lane haben. Im schlimmsten Fall verliert das Unternehmen seine notwendige Lizenz für die Durchführung der Schulungen. FLASH minimiert die Fehleranfälligkeit und stellt somit einen Sicherheitsfaktor für die Firma Fast Lane dar.

2.6 EINSATZ DER REMOTE LABORE

Die Remote Labore werden zum einen für die Versorgung der von Fast Lane durchgeführten Schulungen eingesetzt. Zum anderen werden freie Kapazitäten für die Vermietung an Partner oder konkurrierende Schulungsanbieter angeboten.

Aufgrund des globalen Einsatzes werden die Remote Labore 24 Stunden am Tag an 7 Tagen in der Woche (24 x 7) betrieben. Daher gibt es keinen festen Zeitraum für Wartungen, sogenannte Wartungsfenster, an denen das gesamte Remote Labor nicht im Betrieb ist. Wartungsfenster sind nur auf der Ebene einzelner Labore vorhanden, wenn diese nicht im produktiven Einsatz sind.

Ein zukünftiger Einsatz der einzelnen Remote Labore im Mehrschichtbetrieb, dem sogenannten Timesharing, wird die Auslastung erhöhen und die Zeit für Wartungen weiter verringern. Wird ein Labor in maximal drei Zeitzonen eingesetzt, dann wird jeder Zeitzone für genau sieben Stunden Zugriff auf das Labor gewährleistet. Zwischen zwei Zeitzonen ist dann ein Zeitfenster von einer Stunde für Wartungsaufgaben vorhanden.

3 SYSTEMUMGEBUNG - ANALYSE

Dieses Kapitel beschreibt den Infrastrukturbereich, das sogenannte Backend, der vorhandenen Systemumgebung aus technischer Sicht. In Kapitel *2* wurde dahingegen die Sicht der Dozenten und Trainer auf das sogenannte Frontend beschrieben. Zur Einführung erfolgt eine grundlegende Beschreibung des Remote Labor Systemaufbaus und im Anschluss die Beschreibung der einzelnen Komponenten.

3.1 REMOTE LABOR SYSTEMÜBERBLICK

Zum Zeitpunkt dieser Arbeit sind drei Remote Labore in Hamburg, Frankfurt und Berlin installiert. Das Berliner Remote Labor ist das Zentrallabor, an das die beiden anderen in Sternenform über Virtual Private Networks[3] (VPN) angebunden sind (siehe *Abbildung 3.1*). Das Berliner Labor wird als „zentral“ bezeichnet, da alle für den Betrieb der Remote Labore notwendigen Serversysteme dort installiert sind. In den anderen beiden Remote Laboren sind, mit Ausnahme des Webservers, der ausschließlich in Berlin bereitgestellt wird, für jeden Server ein Back-up-System vorhanden. Alle Remote Labore befinden sich in einem logischen Netzwerk, sodass der Zugriff von und zu allen Systemen über TCP/IP gewährleistet ist.

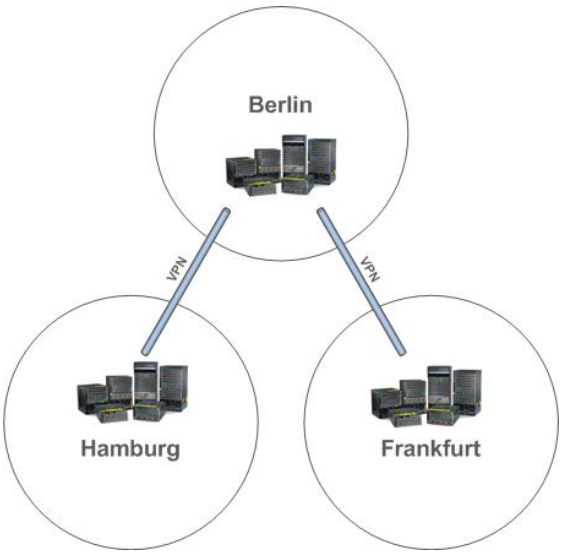

Abbildung 3.1: Remote Labore - VPN

Alle Standorte sind über denselben Provider angebunden. Dadurch wird eine maximale Antwortzeit innerhalb des gesamten logischen Netzwerkes von weniger als 100ms sichergestellt. Dies ist für die Entscheidung relevant, das zu entwickelnde System an einem

[3] VPN: Gesicherte Verbindung über ein unsicheres öffentliches Netzwerk (hier Providernetzwerk)

Remote Labor Standort auszuführen, von dem es Zugriff auf die anderen beiden Standorte hat. Deshalb wird die bestehende Infrastruktur sowohl logisch als auch physikalisch als ein System betrachtet.

Abbildung 3.2 zeigt den Aufbau des zentralen Labors in Berlin und eine Seminarraum Seite. Im Seminarraum sind die Dozenten und Teilnehmer PCs zu sehen. Von ihrem lokalen Gateway wird eine Verbindung über das Internet zum Webserver des Remote Labors aufgebaut. Das Remote Labor ist in die zwei logischen Bereiche Infrastruktur und Labor aufgeteilt.

Im Infrastrukturbereich befinden sich die Serversysteme. Sie stehen in einer, durch zwei Firewalls, die Externe und die Interne, geschützten Demilitarized Zone (DMZ). Der Webserver (WEB) stellt die Webseiten bereit. Dafür wird der Internet Information Server 6.0 (IIS) von Microsoft verwendet. Als Datenbankserver (DB-Server) kommt das Datenbankmanagementsystem (DBMS) Microsoft SQL Server 2005 zum Einsatz. Die für die Authentifizierung notwendigen Daten werden auf dem Lightweight Directory Access Protocol (LDAP) Server bereitgestellt. Dieser nutzt einen Verzeichnisdienst für die Speicherung der Daten. Der vierte Server ist der NoMachine Server (NX) in Version 3.1, dieser ist für die Übertragungen von Bildschirminhalten eines Rechners aus den Remote Laboren zu den Dozenten und Teilnehmer PCs zuständig. Für die Übertragung wird unter anderem das Remote Desktop Protokoll (RDP) verwendet. Des Weiteren gehören zu der Infrastruktur die sogenannten Terminalserver (TS) und die schaltbaren Leistungsverteiler (APC), an die jedes Netzwerkgerät aus den Laboren angeschlossen ist. Die Terminalserver und die Leistungsverteiler sind von allen Systemen im Infrastrukturbereich zu erreichen. Die Netzwerke hinter der internen Firewall sind logisch voneinander getrennt, sodass Querverbindungen ausgeschlossen sind.

Im Laborbereich befinden sich die einzelnen Labore mit ihren Netzwerkgeräten, die für die Durchführung der Übungen notwendig sind.

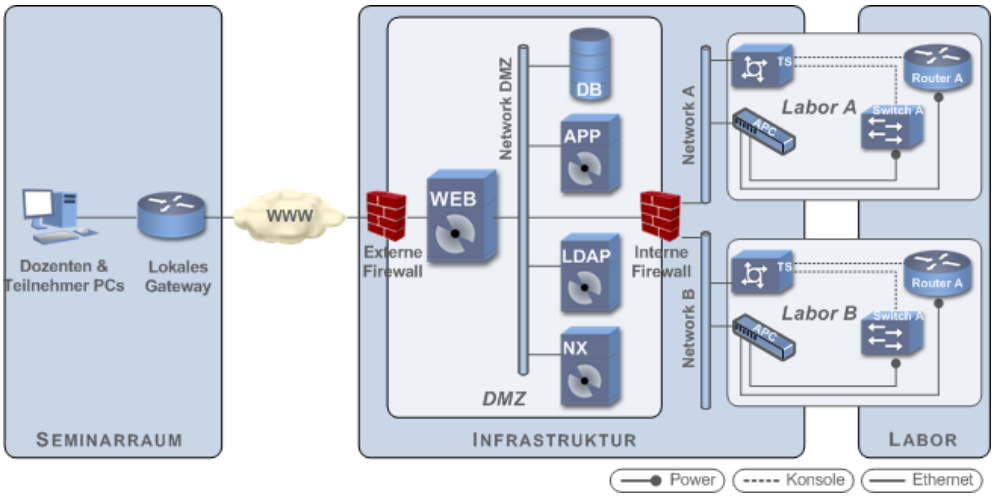

Abbildung 3.2: Remote Labor Systemüberblick

Die für die Umsetzung dieses Projektes relevanten Systeme wurden in ***Abbildung 3.3*** hervorgehoben. Es wird nur der Infrastruktur sowie Laborbereich betrachtet. Die Laborseite wird exemplarisch anhand der Laborkomponenten des projektrelevanten ICND1 Kurses verdeutlicht.

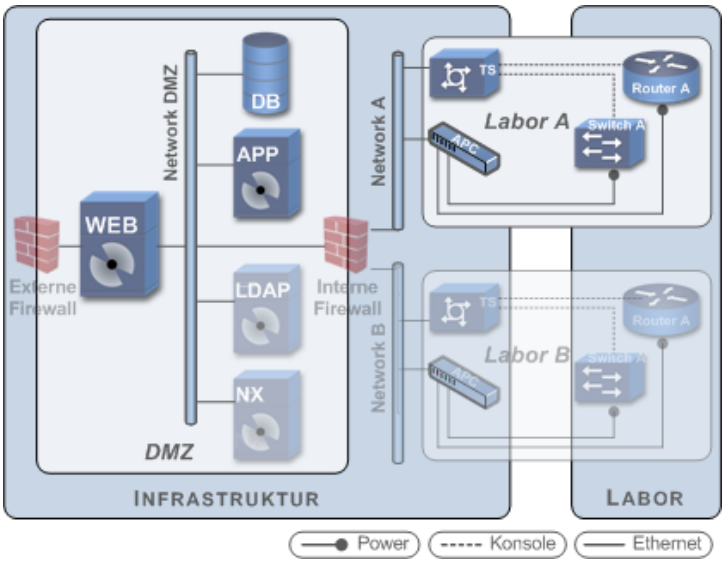

Abbildung 3.3: Projektrelevante Systeme

In den folgenden Abschnitten werden die beiden Systembereiche und die einzelnen Komponenten detailliert erläutert.

3.2 INFRASTRUKTUR VS. LABOR BEREICH

Die für den Betrieb und den Zugriff auf die Laborgeräte notwendigen Systeme bilden den Infrastrukturbereich.

Zu den Labor Komponenten zählen alle Netzwerkgeräte, die in den jeweiligen Laboren installiert sind und von den Dozenten und Teilnehmern für die Durchführung ihrer Übungen benötigt werden. Die Geräte eines Labors bilden zusammen eine geschlossene Einheit. Zugriffe von den Laboren auf die Systeme der Infrastruktur sind nicht möglich. Dies wird zum einen durch die uni-direktionalen Konsolen Verbindungen und zum anderen durch die voneinander physikalisch getrennt installierten Labore erreicht. Verbindungen, die aus den Laboren heraus aufgebaut werden müssen, wie die RDP Verbindungen, werden durch restriktive Regeln der internen Firewall gesichert. Es werden nur eingehende Verbindungen von den Serversystemen der Infrastruktur zugelassen.

3.2.1 INFRASTRUKTUR KOMPONENTEN

In diesem Abschnitt werden die Infrastruktur Komponenten im Einzelnen erläutert.

3.2.1.1 SERVERSYSTEME

Die zentrale Komponente des Fast Lane Remote Labor Systems ist der Webserver. Er stellt die Schnittstelle des Remote Labors nach außen zur Verfügung. Alle Verbindungen werden über ihn aufgebaut und kontrolliert. Seine Aufgaben sind das Bereitstellen der eigentlichen Webseiten und der laborspezifischen Schulungsinhalte für die Webseiten (siehe ***Abbildung 2.3***). Eine weitere Aufgabe ist die Planung und Verwaltung der Zugriffszeiten

auf die einzelnen Labore. Dafür ist der in den Webserver integrierte Scheduler zuständig. Die notwendigen Daten werden auf dem Datenbankserver gespeichert.

Auf dem Anwendungsserver läuft zurzeit ausschließlich der TELNET Server, der die Konsolenverbindungen zu den Netzwerkgeräten herstellt. Nachdem ein Teilnehmer auf ein Gerätesymbol im Netzwerkdiagramm (siehe ***Abbildung 2.4***) der Webseite klickt, wird auf dem lokalen PC ein TELNET Applet geöffnet. Dieses Applet verbindet sich mit den notwendigen Parametern über den TELNET Server auf die einzelnen Netzwerkgeräte der Labore. Der TELNET Server baut dafür eine Reverse TELNET (siehe ***3.5.2***) Verbindung über die Terminal Server (siehe ***3.2.1.3***) zu den Netzwerkgeräten auf. Die Teilnehmer haben ausschließlich Zugriff[4] auf die Geräte ihres PODs. Für die Ausfallsicherheit der gestarteten Anwendungen wird auf dem Anwendungsserver der Microsoft Clusterdienst verwendet (vgl. (Cavale, 2004)). In Abschnitt ***5.1.6*** wird der Einsatz in Verbindung mit FLASH erläutert.

Auf dem Datenbankserver werden die Webinhalte für die Schulungen, die Verbindungsdaten für die Geräte und die Login Daten gespeichert. Zudem stellt er die Schnittstellen für den Datenaustausch der Infrastruktur Serversysteme untereinander zur Verfügung.

3.2.1.2 APCs

American Power Conversion (APC) (siehe (APC.com, 2008)) ist ein Unternehmen, das unter anderem schaltbare und netzwerkfähige Leistungsverteiler herstellt. ***Abbildung 3.4*** zeigt ein Modell vom Typ AP7921. Der Leistungsverteiler verfügt über einen Ethernet Netzwerk Anschluss (1) für den Zugriff per TCP/IP und einen seriellen Anschluss (2), über den die Basiskonfiguration vorgenommen wird. Eine Statusanzeige (3) des aktuellen Leistungsverbrauchs ist ebenfalls vorhanden. Bei diesem Modell können acht 230V-Endgeräte über die Ausgangsanschlüsse (4) angeschlossen werden. Der Eingangsanschluss (5) versorgt den Verteiler mit Spannung. Jedes Netzwerkgerät in den einzelnen Laboren wird über einen APC Anschluss mit Spannung versorgt. Die APCs wurden installiert, um die Netzwerkgeräte von jedem Standort Aus- und wieder Einschalten zu können. Die APCs sind per TCP/IP über den Infrastrukturbereich zu erreichen und lassen sich über ein integriertes Webinterface konfigurieren.

Zusätzlich bieten die APCs eine Unterstützung für das Simple Network Management Protokoll (SNMP). Mithilfe von SNMP (vgl. Abschnitt ***3.5.1***) ist es möglich, den Status jedes einzelnen Anschlusses zu ermitteln und ein- oder auszuschalten. Auf diese Unterstützung wird das zu entwickelnde System zurückgreifen.

Für weitere Informationen zu den APCs siehe (Kasper, A., 2008).

[4] Die notwendige Berechtigungszuordnung zwischen den einzelnen Geräten und den Login Daten der Teilnehmer wird über eine interne Zugriffsmatrix (Login X Gerät) verwaltet. Die notwendigen Daten werden auf dem Datenbankserver gespeichert und mit den Daten des Verzeichnisdienst Servers abgeglichen.

Abbildung 3.4: APC - AP7921

3.2.1.3 TERMINAL SERVER

Ein Terminal Server (TS) wie in ***Abbildung 3.5*** gezeigt, ist ein Router, der mehrere asynchrone serielle Ports (7) zur Verfügung stellt. Über den sogenannten Out-Of-Band [5] (OOB) Zugang wird ein Fernzugriff der angeschlossenen Netzwerkgeräte ermöglicht. Dafür werden an die asynchronen seriellen Ports des Terminal Servers die Konsolen Ports (siehe Konsolenanschluss in ***3.2.2.1***) der Labor Router und Switches angeschlossen. Dadurch ermöglicht der Terminal Server den Zugriff über einen zentralen Punkt auf mehrere andere Geräte. Die Ethernet Schnittstelle (6) wird dafür mit den notwendigen TCP/IP Parametern konfiguriert. Der Zugriff auf die Laborgeräte erfolgt über das Reverse TELNET Protokoll (siehe ***3.5.2***). Eine serielle Verbindung auf dem Terminal Server wird als „Line“ bezeichnet.

Die Terminal Server bieten ebenfalls die Unterstützung für SNMP an, wodurch sich der Status (active / not active) einer Line ermitteln lässt. Des Weiteren kann eine bestehende Verbindung per SNMP abgebrochen werden. Dies wird als „Clear Line“ bezeichnet. Dafür wird der Status der Line auf „not active“ zurückgesetzt und ermöglicht so den Aufbau einer neuen Verbindung. Auf diese Unterstützung wird das zu entwickelnde System ebenfalls zurückgreifen.

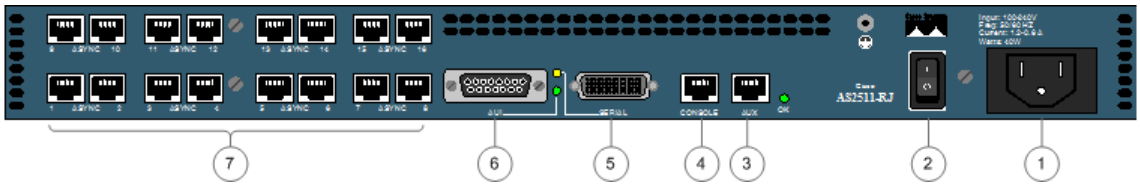

Abbildung 3.5: Cisco Terminal Server - c2511RJ

Des Weiteren verfügt der Terminal Server über einen seriellen (5), Konsolen (4) und AUX (3) Port. Ein Spannungsanschluss (1) und entsprechender Schalter (2) ist ebenfalls vorhanden. Weiterführende Informationen siehe (C2500, 2007).

[5] Out-Of-Band: Bezeichnet die Möglichkeit die Systeme über ihre serielle Schnittstelle, auch im Fehlerfall zu erreichen, unabhängig der laufenden Konfiguration oder eines Systemausfalls.

3.2.2 LABOR KOMPONENTEN

Die in diesem Abschnitt erläuterten Netzwerkgeräte sind Bestandteil des projektrelevanten ICND1 Labors. Die hier erläuterten Systemgrundlagen und Eigenschaften sind für die Modelltypen der anderen Labore identisch. Sie unterscheiden sich in der Art und Anzahl der Schnittstellen und ihrer Systemleistung in Abhängigkeit von Systemkomponenten wie Prozessorleistung, Art und Menge des Arbeitsspeichers.

3.2.2.1 KOMPONENTEN DER NETZWERKGERÄTE

Im Folgenden werden die einzelnen Komponenten detailliert erläutert. Dies ist notwendig, um die von FLASH umzusetzenden Konfigurationsabläufe zu realisieren.

Speicher Komponenten

Die Netzwerkgeräte besitzen verschiedene Speicherbereiche, die in ***Tabelle 3.1*** beschrieben sind.

Tabelle 3.1: Netzwerkgeräte - Speicher Komponenten

Speicher	Beschreibung
RAM	Hält dynamische Systeminformationen zur Laufzeit, das laufende Betriebssystem und die laufende Konfiguration „Running-Config“.
ROM	Enthält den ROM Monitor und den Bootstrap, um das Gerät zu starten und das Betriebssystem zu laden.
Flash	Hält die Image Datei des Betriebssystems.
NVRAM	Enthält die gespeicherte Gerätekonfiguration, die beim Start geladen wird. Diese Konfiguration wird als „Startup-Config“ bezeichnet.
Configuration-Register	Ein 2 Byte großes Register, das unter anderem die Einstellung für die Startvariante des Gerätes und die Geschwindigkeit des Konsolen Anschlusses festlegt.

Konfigurations-Register

Das Konfigurations-Register (vgl. (ConfReg, 2006)) ist zwei Byte groß und wird im NVRAM gespeichert. Es legt unter anderem die Startart des Netzwerkgerätes fest. Das Gerät startet das IOS aus dem Flash Speicher und lädt die „Startup-Config“. Wird der Wert bei einem Router auf 0x2142 (Standardwert ist 0x2102) eingestellt (d. h. gesetztes 6te Bit), ist die Voraussetzung für das Password-Recovery (vgl. ***3.4***) geschaffen.

Software

Die Netzwerkgeräte besitzen verschiedene Software Komponenten, die in ***Tabelle 3.2*** beschrieben sind.

Tabelle 3.2: Netzwerkgeräte - Software Komponenten

SW Komponente	Beschreibung
ROM Monitor	Der ROM Monitor, auch Bootstrap genannt, ist die Firmware des Gerätes und wird beim Einschalten oder Neustarten des Gerätes ausgeführt. Dabei wird die Hardware initialisiert und das IOS geladen. Über den ROM Monitor lassen sich Aufgaben wie das Zurücksetzen eines verlorenen Passwords, auch als „Password-Recovery“ bekannt, durchführen oder ein IOS über den Konsolen Port in den Flash des Gerätes laden. Ist kein IOS Image im Flash gespeichert, läuft das Gerät automatisch weiter im ROM Monitor.
POST	Wird vom ROM Monitor aufgerufen und führt grundlegende Funktionstests der internen Hardware und der verbauten Module durch.
IOS	Das Betriebssystem des Gerätes. Es wird als binäre Datei im Flash gespeichert und beim Starten des Gerätes in den RAM geladen.

Betriebssystem (IOS)

Die Router und Switches verwenden das Cisco eigene Betriebssystem Internetwork Operating System (IOS). Dabei handelt es sich um ein Softwarepaket, dass die verschiedenen Routing- und Switching-Funktionen mit einem Multitasking Betriebssystem verbindet. Gesteuert wird das IOS über die Kommandozeile (siehe Kommandozeilen Schnittstelle in ***3.2.2.1***).

Eine Problematik bei den IOS Versionen ist, dass diese von verschiedenen Entwicklerteams erweitert werden und um neue Funktionalitäten ergänzt werden. Dabei kommt es zu kleinen Veränderungen bei den Ausgaben auf der Kommandozeilen Ebene und auch bei der Befehlssyntax. Ein mögliches Beispiel sieht wie folgt aus:

Alter Befehl: `Router(config)#enable secret`
Neuer Befehl: `Router(config)#enable-secret`

Die alten Befehle werden von der neuen IOS Version nicht mehr erkannt. Bei automatisierten Aufgaben führt dies zu Problemen und Anpassungen der Automatismen.

Kommandozeilen Schnittstelle (CLI)

Um das IOS der Geräte konfigurieren zu können, verfügt es über eine Kommandozeilen Schnittstelle, das Command Line Interface (CLI). Das CLI verfügt über drei Konfigurationsstufen, dem „User Mode“, dem „Priviliged Mode“ und dem „Configuration Mode“. Für ausführlichere Informationen siehe (CLI, 2003).

Eine Kommandozeilenausgabe sieht folgendermaßen aus:

```
Device#
```

In diesem Fall kennzeichnet die „#“, dass wir uns im „Priviliged Mode“ befinden. „Device“ entspricht dem Systemnamen des Gerätes. Mit dem Kommando „configure terminal“ kann in den „Configuration Mode“ gewechselt und dort Konfigurationseinstellungen am System vorgenommen werden. Das entsprechende Kommando sieht wie folgt aus:

```
Device#configure terminal
Device(config)#
```

Die Ausführung von Kommandos kann durch optionale Parameter erweitert werden.

Zu dem gibt es interaktive Abfragen, auf die per Kommandoeingabe oder über bestimmte Steuerzeichen reagiert wird. Die folgenden Zeilen zeigen zwei Abfragen dieser Art. Die Erste wird mit der Eingabe des Kommandos „no“ beantwortet und die Zweite durch das Drücken der „Return“ Taste.

```
Would you like to enter the initial configuration dialog?
[yes/no]: no

Press RETURN to get started!
```

Die Kommandozeilen Schnittstelle reagiert neben den Kommandos auch auf spezielle Steuerzeichen. Das BREAK[6] Signal ist ein Beispiel, welches zwingend notwendig für das Password-Recovery ist.

Konfigurationen

Die Einstellungen der Netzwerkgeräte werden in zwei Konfigurationen gespeichert. Zum einen die Startup-Config und zum anderen die Running-Config. Die Running-Config ist die aktuell gültige Konfiguration für das Gerät. Sie wird im RAM der Geräte gehalten und geht bei einem Neustart oder Ausschalten verloren. Die Startup-Config wird beim Neustart oder Einschalten des Gerätes in dem RAM kopiert und wird somit zur Running-Config.

Die „Startup-Config“ wird im NVRAM der Geräte gespeichert und bleibt auch nach einem Neustart oder Ausschalten erhalten.

Soll die Running-Config gespeichert werden, muss dies über den folgenden Befehl erfolgen:

```
Device#copy running-config startup-config
```

Die Running-Config wird somit zur Startup-Config.

Spannungsversorgung

Der Spannungsanschluss wird mit einem Ausgangsanschluss der APCs verbunden, so kann das Netzwerkgerät über das Netzwerk ein- oder ausgeschaltet werden. Der Spannungsschalter bleibt dafür immer in der Position „Ein“. Dies ist eine Grundvoraussetzung, um die Kontrolle über die APCs zu ermöglichen.

[6] BREAK: Bei dem hier verwendeten BREAK handelt sich nicht um das TELNET Break Signal mit dem ASCII Wert 0x81, sondern um das sogenannte Terminal BREAK Signal mit der Bytefolge 0xFF F3.

Konsolenanschluss

Über den Konsolenanschluss wird das Netzwerkgerät an eine der asynchronen seriellen Terminal Server Ports angeschlossen. So besteht ein permanenter Zugriff auf die Kommandozeilenoberfläche des Betriebssystems. Da es sich um eine uni-direktionale Verbindung handelt, kann keine Verbindung über diesen Anschluss nach draußen hergestellt werden.

3.2.2.2 ROUTER

Das in ***Abbildung 3.6*** gezeigte Router Modell vom Typ c1841 wird im ICND1 POD Bereich eingesetzt.

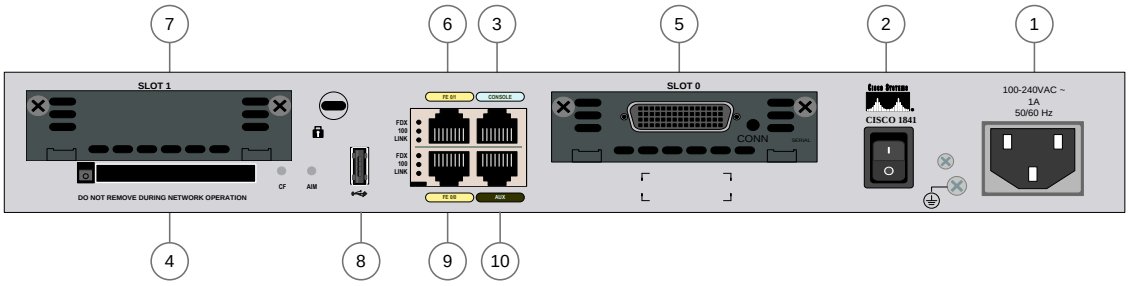

Abbildung 3.6: Cisco Router - c1841

In ***Tabelle 3.3*** sind die einzelnen Schnittstellen, Anschlüsse und Schalter beschrieben. Aus dem Infrastrukturbereich besteht die Möglichkeit des Zugriffs auf diejenigen mit den Nummern von 1 bis 4. Alle anderen können aus dem Laborbereich kontrolliert werden.

Tabelle 3.3: Schnittstellenbeschreibung - c1841

Nummer	Beschreibung
1	Spannungsanschluss
2	Spannungsschalter
3	Konsolen Port
4	Compact Flash (CF) Card Schacht
5	Modulschacht mit einer seriellen Schnittstelle (WIC-1T)
6	Fast Ethernet Port (FE0/1)
7	Modulschacht leer
8	USB Schnittstelle
9	Fast Ethernet Port (FE0/0)
10	AUX (Auxiliary) Port

Für weitergehende Informationen zum Cisco Router c1841 siehe (c1841, 2007) .

Besonderheiten

Bei den Routern vom Typ c1841 gibt es zwei verschiedene ROM Monitor Module. Zum einen den „Readonly“ Rom Monitor, der fest in das ROM des Routers eingebrannt ist und nicht überschrieben werden kann. Zum anderen gibt es den „Upgrade“ Rom Monitor, welches mit neueren Versionen überschrieben werden kann. Mögliche Fehler in der Firmware können durch das Einspielen einer neuen Version behoben werden.

3.2.2.3 SWITCH

Das in ***Abbildung 3.7*** gezeigte Switch Modell vom Typ c2960 wird im ICND1 POD Bereich eingesetzt.

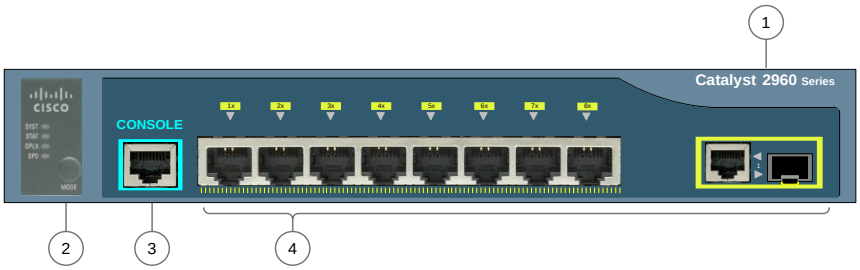

Abbildung 3.7: Cisco Switch - c2960

In ***Tabelle 3.4*** sind die einzelnen Schnittstellen und Anschlüsse beschrieben. Aus dem Infrastrukturbereich besteht die Möglichkeit des Zugriffs auf die mit den Nummern 1,2 und 3. Alle anderen können aus dem Laborbereich kontrolliert werden.

Tabelle 3.4: Schnittstellenbeschreibung - c2960

Nummer	Beschreibung
1	Spannungsanschluss
2	Mode Knopf
3	Konsolen Port
4	Fast Ethernet Ports (FE0/0-7), Gigabit Ethernet Port (Gi0/0) und GBIC Modulschacht

Besonderheit

Die Switches verfügen über eine fest eingebrannte Firmware und können im Gegensatz zu den Routern nicht auf einen anderen Stand gebracht werden.

Für das Password Recovery muss bei diesen Switches der „Mode“ Knopf (2) gedrückt werden, was den physikalischen Zugriff auf das Gerät erfordert und voraussetzt. Über das Setzen des „Enable Break“ Bit im Konfigurationsregister kann man das notwendige „Break“ Signal als Steuerzeichen an den Switch senden. Der „Mode“ Knopf muss dann nicht gedrückt werden, um in den Password Recovery Modus zu gelangen. Die verwendeten Switches weisen an dieser Stelle eine Fehlfunktion auf. Trotz Aktivierung des „Enable Break“ Bits bleibt dies ohne Funktion. Bei der Untersuchung dieses Fehlverhaltens wurden fünf verschiedene Modelle der 2900er Serie miteinander verglichen und zwei von fünf Modellen arbeiteten einwandfrei.

Der Versuch die Ursache für das Fehlverhalten zu finden blieb erfolglos. Dazu wurden verschiedene Betriebssysteme eingesetzt und die unterschiedlichen Modelle miteinander verglichen. Die Ergebnisse sind in ***Tabelle 3.5*** zu sehen.

Die Ergebnisse der Untersuchung wurden zur weiteren Problemlösung an den technischen Support des Herstellers weitergeleitet. Beim Abschluss der Arbeit stand eine Lösung noch aus.

Tabelle 3.5: Switch Modelle - Analyse

Modell	C2960-8TC-L	C296040-8TT-S	C2950T-24	C2960-24TT-L	C2950-24
Status	Fehler	OK	Fehler	OK	Fehler
Boot Loader	12.2(35r)SE2	12.1(13r)AY1	12.1(11r)EA1	12.2(25r)SEE1	12.1(11r)EA1
IOS	c2960-lanbasek9-mz.122-40.SE.bin	c2940-i6q4l2-mz.121-22.EA4.bin	c2950-i6q4l2-mz.121-22.EA1.bin	c2960-lanbase-mz.122-25.SEE2.bin	c2950-i6k2l2q4-mz.121-22.EA10a.bin
Modell Revision	A0	E0	Q0	B0	B0
Motherboard Revision	C0	A0	A0	C0	-
Config-Register	0xF	0xF	0xF	0xF	0xF

Die verwendeten Switches verfügen über keinen Spannungsschalter. Sie werden direkt mit den APCs verbunden und müssen nicht gesondert eingeschaltet werden.

Für weitergehende Informationen zum Cisco Switch c2960 siehe (c2960, 2007).

3.3 KONFIGURATIONSPROZEDUREN

In diesem Abschnitt werden die auszuführenden technischen Abläufe erläutert, um die Netzwerkgeräte zu konfigurieren. Diese sind notwendig, um beim Design eine Lösung für die Umsetzung zu finden. Für die Entwicklung von FLASH ist es nicht relevant, wie die Kommandos für die Netzwerkgeräte und deren Befehlssätze aufgebaut sind. FLASH wird so konzipiert, dass es, unabhängig von den eingesetzten Netzwerkgeräten, seine Aufgaben bearbeiten kann. Dadurch wird versucht eine maximale Kompatibilität zu gewährleistet.

Die Konfiguration der Geräte erfolgt über die Kommandozeilen Schnittstelle. Dafür können folgende Aktionen ausgeführt werden:

1. Steuerzeichen
2. Kommandos (optional mit Parametern)
3. Reaktionen auf interaktive Abfragen
4. Warten auf bestimmte Ausgaben
5. Reaktion in einem bestimmten Zeitfenster.

Diese müssen in einer bestimmten Reihenfolge ausgeführt werden. Daher erfolgt die Darstellung dieser Abläufe in Form von Ablaufdiagrammen. Das Beispiel für das Password Recovery aus Abschnitt ***3.4*** ist zur Veranschaulichung auszugsweise in Anhang ***B*** beigefügt. Die vollständigen Diagramme für die im Rahmen dieses Projektes erstellten Abläufe befinden sich auf der beigefügten CD-ROM.

Der in ***Abbildung 3.8*** dargestellte Ablauf ist für alle Konfigurationsprozeduren identisch und muss vor jeder weiteren Konfiguration ausgeführt werden. Als Erstes wird das Netzwerkgerät ausgeschaltet, um einen definierten Ausgangszustand zu erhalten. Im nächsten Schritt wird die

Line auf dem Terminal Server zurückgesetzt, um den Zugriff auf den Konsolenanschluss des Netzwerkgerätes herstellen zu können. Anschließend wird eine Reverse TELNET (vgl. Abschnitt ***3.5.2***) Verbindung zu dem Gerät vorbereitet. Im letzten Schritt wird das Netzwerkgerät wieder eingeschaltet und die Reverse TELNET Verbindung aufgebaut. Im Anschluss können die entsprechenden Konfigurationsschritte durchgeführt werden.

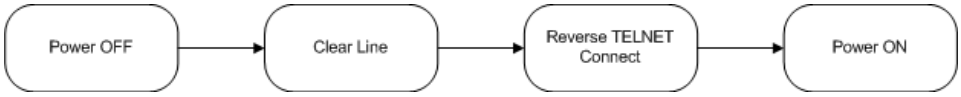

Abbildung 3.8: Allgemeine Konfigurationsprozeduren

Die Konfigurationsprozeduren sind in den folgenden Abbildungen beschrieben. Sie zeigen keine vollständigen Abläufe sondern Auszüge, um einen Überblick zu bekommen. ***Abbildung 3.9*** und ***Abbildung 3.10*** zeigen Abläufe, die durch Kommandoeingaben und das Warten auf entsprechende Ausgaben sequenziell bearbeitet werden können.

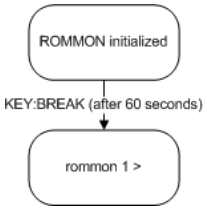

Abbildung 3.9: sequenzieller Ablauf

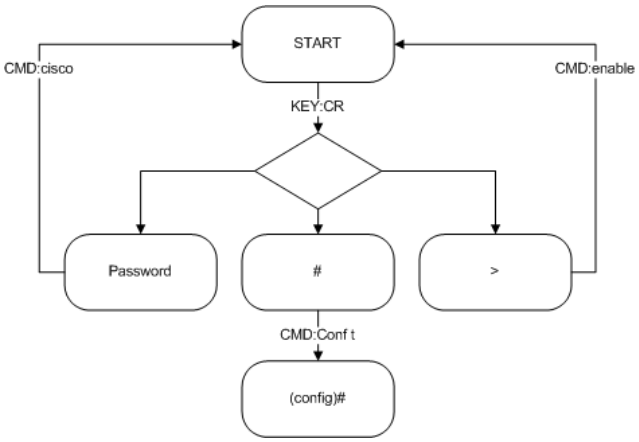

Abbildung 3.10: verzweigter Ablauf

In ***Abbildung 3.11*** ist ein komplexerer Ablauf dargestellt, der nicht durch die Verwendung der oben beschriebenen Aktion bearbeitet wird. Hier soll die Konfiguration des Gerätes ausgelesen werden. Dafür wird das Kommando „show startup-configuration“ abgesetzt. Auf dieses Kommando erfolgt eine Ausgabe, die in Abhängigkeit von der Länge der Konfiguration durch die Ausgabe „-More-„ unterbrochen wird. Um weiter fortzufahren, muss die Taste „Space“ gedrückt werden, bis bei der Beendigung der Ausgabe die Zeichenfolge „Scheduler allocate“ erscheint. Hier muss also die gesamte Ausgabe zwischengespeichert werden, die Zeilen mit „more“ entfernt werden und dann im Anschluss das Ganze gespeichert und zurückgegeben werden, um die Aufgabe auszuführen.

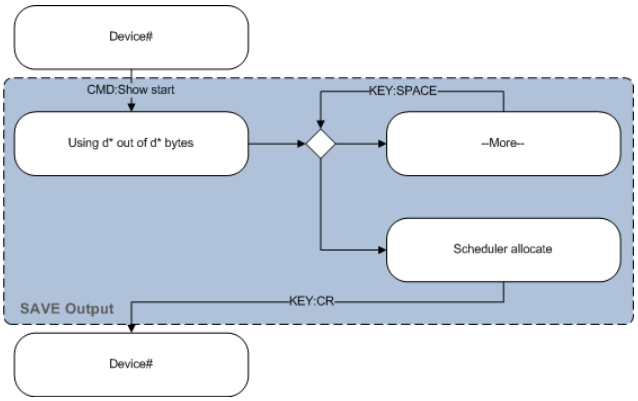

Abbildung 3.11: Komplexer Ablauf

3.3.1 ZEITBEDARF

Der Zeitbedarf wurde anhand von Messungen der Teil- sowie Gesamtabläufe ermittelt. Die im Rahmen dieser Arbeit ermittelten Werte sind in ***Tabelle 3.6*** dokumentiert. Die Werte sind von der Leistungsfähigkeit der Geräte abhängig.

Tabelle 3.6: Dauer der Aktivitäten

Aktivität	Zeitwert in Sekunden
PWD_RECOVERY_1841	300
PWD_RECOVERY_2960	360
PWD_RECOVERY_2950	360
PWD_RECOVERY_3640	540

3.4 PASSWORD RECOVERY

Dieser Abschnitt beschreibt die Rücksetzprozedur der Netzwerkgeräte. Dafür werden die “enable” und „enable secret“ Passwörter[7] umgangen. Dies ist notwendig, wenn Teilnehmer während der Schulung ein frei gewähltes Passwort setzen. Diese Passwörter schützen den Zugriff auf den Privilegierten- und Konfigurationsmodus der Netzwerkgeräte. Im worst-case unterbindet der Teilnehmer so den Zugriff auf das Gerät.

Das „enable“ Passwort kann im Gegensatz zum verschlüsselten „enable secret“ Passwort, welches nur ersetzt werden kann, wiederhergestellt werden. Beide Passwörter sind in der „Startup-Config“ im NVRAM gespeichert. Das „Password Recovery“ muss also das Gerät dazu bringen, ohne die „Startup-Config“ zu booten. Dazu wird das sechste Bit im Konfigurations-Register gesetzt.

Soll die Konfiguration im System erhalten, muss zuerst in den privilegierten Modus gewechselt werden, dann die Startup-Config in die Running-Config kopiert, die Passwörter entsprechend zurückgesetzt und im Anschluss die Running-Config gesichert und das sechste Bit des Konfigurations-Registers wieder zurückgesetzt werden.

Für das Password Recovery gilt die Einschränkung, dass es nur über die Konsolenverbindung zu den Netzwerkgeräten einsetzbar ist.

[7] enable (secret): Lokal gespeicherte Passwörter für den Zugriff auf die Geräte und deren Konfiguration.

3.5 RELEVANTE PROTOKOLLE ZUR KONFIGURATION

3.5.1 SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP)

Das Simple Network Management Protocol (SNMP) (siehe (Case, 1990)) ist ein Netzwerkprotokoll, das von der Internet Engineering Task Force (IETF) entwickelt wurde. Die Hauptaufgabe ist die zentrale Überwachung und Steuerung von Netzwerkgeräten. Das standardisierte Protokoll regelt die Kommunikation zwischen den überwachten Geräten und der Überwachungsstation.

Für die Überwachung und Steuerung werden sogenannte Agenten verwendet. Diese werden auf den überwachten Geräten ausgeführt und sind in der Lage die Einstellungen der Geräte auszulesen und zu verändern. Die Kommunikation erfolgt über fünf grundlegende Befehle, die in *Tabelle 3.7* beschrieben sind.

Tabelle 3.7: SNMP - Befehle

Befehle	Beschreibung
GET	Fordert einen Management-Datensatz an.
GETNEXT	Ruft den nachfolgenden Datensatz ab.
SET	Verändert einen oder mehrere Datensatz/Datensätze des Netzwerkgerätes.
RESPONSE	Antwort auf ein GET, GETNEXT oder SET Paket.
TRAP	Unaufgeforderte Nachricht von einem Agenten an den Manager, dass ein Ereignis eingetreten ist. Dient zur automatisierten Überwachung.

Die beiden GET-Befehle können vom Manager zu einem Agenten gesendet werden, um Daten des Gerätes anzufordern. Dieses antwortet mit einem RESPONSE-Paket, das entweder die angeforderten Daten enthält oder eine Fehlermeldung.

Mit dem SET-Paket kann ein Manager Werte beim Agenten verändern. Damit ist es möglich Einstellungen vorzunehmen, oder Aktionen auszulösen. Der Agent bestätigt die Übernahme der Werte ebenfalls mit einem RESPONSE-Paket.

Wenn der Agent bei der Überwachung des Systems einen Fehler erkennt, kann er diesen mithilfe eines TRAP-Paketes unaufgefordert an die Management Station melden. Diese Pakete werden nicht vom Manager bestätigt. Der Agent kann daher nicht feststellen, ob der TRAP beim Manager angekommen ist.

Damit die Netzwerkbelastung gering bleibt, wird zum Versenden der Nachrichten das verbindungslose User Datagram Protocol (UDP) verwendet. Der Agent empfängt die Anfragen (Requests) auf dem Port 161, während für den Manager der Port 162 zum Empfangen der TRAP-Meldungen vorgeschrieben ist.

Die Werte, die von einem Manager über die gemanagte Netzwerkkomponente ausgelesen und verändert werden können, die so genannten ‚Managed Objects', werden in der Management Information Base (MIB) beschrieben. Dabei handelt es sich um Beschreibungsdateien, in denen die einzelnen Werte tabellarisch aufgeführt werden.

Die Informationen der MIB sind in Form einer Baumstruktur organisiert, deren einzelne Zweige entweder durch Nummern oder alternativ durch alphanumerische Bezeichnungen dargestellt werden können.

Die Cisco MIB ist unter iso.org.dod.internet.priv.ent.cisco zu finden, das ebenso durch die Zahlenreihe 1.3.6.1.4.1.9 eindeutig bestimmt ist (1 für iso, 3 für org usw.). Diese aus Punkten und Zahlen bestehende Zeichenkette nennt man Object Identifier (OID). In den MIBs wird dann weiter verzweigt bis zu den einzelnen Daten, die jeweils auch eine eigene OID besitzen und somit eindeutig identifiziert werden können. Ein Ausschnitt aus der Cisco MIB ist in Anhang ***D*** dargestellt.

Diese Dateien sind in der abstrakten Beschreibungssprache SMIv2 geschrieben, welche auf ASN.1 (siehe (ITU-T, 2007)) basiert. Zu jedem Datum, das vom Agenten abgerufen oder verändert werden kann, wird in diesen Dateien eine Reihe von Informationen angegeben. Mithilfe der Beschreibungsdateien sind die SNMP-Programme in der Lage, den hierarchischen Aufbau der Daten jedes beliebigen SNMP-Agenten darzustellen und Werte von diesem anzufordern.

Für die Sicherheit wird ein sogenannter SNMP Community-String verwendet. Dieser wird als Wertepaar mit einem Zugriffsberechtigungswert (read-only, read-write, not-accessible) gespeichert. Seine Funktion entspricht dem eines Passwortes.

3.5.2 (REVERSE) TELNET

TELNET (siehe (Postel, 1983)) ist ein Klient Server Protokoll, dass einem zuverlässigen und verbindungsorientierten Transport gewährleistet. Standardmäßig wird der TCP Port 23 für den Aufbau der Verbindungen verwendet.

Reverse TELNET ist eine spezielle Anwendung des TELNET Protokolls. Die Server Seite liest und schreibt die Daten an einen seriellen Port (RS-232[8]). Reverse TELNET wird von den Terminal Servern implementiert. Dadurch können die über TCP/IP an den Terminal Server angebundenen Geräte eine TELNET Verbindung zu den seriell angeschlossenen Geräten herstellen. Die Auswahl des richtigen seriellen Ports erfolgt über eine spezielle Port-Nummer. Bei dem Terminalserver in ***Abbildung 3.5*** wird zu der festgelegten Nummer 2000 der entsprechende serielle Port addiert. Eine Beispiel Kommandozeile für eine Verbindung zum ersten angeschlossenen Gerät sieht folgendermaßen aus:

```
telnet 10.1.1.101 2001
```

TELNET verfügt über keinen Mechanismus um seine Verbindung zu schützen. Daher wird im Allgemeinen das Secure Shell Protokoll (SSH) verwendet. SSH ermöglicht im Gegensatz zu TELNET eine sichere, authentifizierte und verschlüsselte Verbindung zwischen zwei Netzwerkgeräten über ein unsicheres Netzwerk. SSH kann in dieser Systemumgebung nicht verwendet werden, da es keine entsprechende Reverse SSH Implementation gibt. Das Netzwerk des Infrastrukturbereichs der Remote Labore kann zudem als sicher bezeichnet werden und erfordert deshalb keine weiteren Sicherungsmaßnahmen.

[8] RS-232: Auch EIA-232 genannt, bezeichnet einen Standard für eine serielle Schnittstelle.

3.6 EXISTIERENDE UND VERWANDTE LÖSUNGEN

In diesem Abschnitt werden die vorhandenen kommerziellen Lösungen, sowie die Automationssysteme der Konkurrenz betrachtet. Insbesondere werden die Funktionalitäten, die diese zur Verfügung stellen, im Hinblick auf eine mögliche Verwendung in FLASH analysiert. Im letzten Abschnitt wird dann auf das Werkzeug, die Skriptsprachen, der Systemadministratoren eingegangen, welches für die Unterstützung ihrer Tätigkeiten eingesetzt wird.

3.6.1 KONKURRENZ

Das Unternehmen Fast Lane teilt sich mit zwei weiteren Remote Labor Anbietern den weltweiten Markt auf. Das slowenische Unternehmen NIL Data Communications Ltd. (siehe (NIL.si, 2008)) auf der einen und das amerikanische Unternehmen Toolwire Inc. (siehe (Toolwire.com, 2007)) auf der anderen Seite. Beide Unternehmen bieten ebenfalls Remote Labore an und verwenden individuelle Systeme, um automatisierte Funktionen für das Betreiben und die Durchführung der Labore bereitzustellen.

Aufgrund von Partnervereinbarungen jedes der Unternehmen mit dem Hersteller Cisco Systems, ist es innerhalb dieser Vereinbarungen möglich, kostenfrei auf die Labore der anderen Unternehmen zuzugreifen. Für einen objektiven Vergleich der Systeme beider Anbieter mit der Firma Fast Lane wurde auf die gleichen Labore für die Schulung ICND1 zugegriffen. Die Ergebnisse sind in ***Tabelle 3.8*** ersichtlich. Die NIL bietet Automatisierungsfunktionen von allen Unternehmen am längsten an und setzt Ihr System auf ca. 20 unterschiedlichen Remote Laboren ein. Toolwire kommt auf ca. 15 bis 20 unterschiedliche Labore und ist noch in der Entwicklungsphase eines entsprechenden Systems. Bei genaueren Tests stellte sich heraus, dass die angebotenen Funktionen nicht zuverlässig arbeiten. Das System setzt voraus, dass die Teilnehmer exakt den Übungsschritten folgen. Wird zum Beispiel das Passwort auf den Geräten verändert, funktioniert das System nicht mehr und quittiert die Ausführung der entsprechenden Funktion mit einer Fehlermeldung.

Die Fast Lane betreibt ca. 65 verschiedene Remote Labor Typen und wird nach Einführung von FLASH die grundlegenden Funktionen der Konkurrenzsysteme bereitstellen.

Tabelle 3.8: Remote Labor Anbieter - Vergleich

Funktion	Fast Lane	NIL	Toolwire
Zurücksetzen in Ausgangszustand	⇧	⇧	⇨
Vordefinierte Konfiguration laden	⇧	⇧	⇨
Speichern der Konfiguration	⇨	⇧	⇨
Konsolenverbindung beenden	⇧	⇧	⇨
Geräte aus- und wieder einschalten	⇧	⇩	⇨
IOS Überprüfung	⇨	⇧	⇨
Konfiguration per E-Mail versenden	⇨	⇩	⇨

⇩ *Wird nicht unterstützt* ⇨ *In Planung* ⇧ *Wird unterstützt*

3.6.2 NETZWERK MANAGEMENT LÖSUNGEN

Die Vorteile einzelner Funktionen der kommerziellen Netzwerk Management Lösungen soll beim Design von FLASH berücksichtigt werden. Grundsätzlich stellen die Lösungen von Cisco,

IBM und HP ähnliche Funktionalitäten zur Verfügung. Daher werden diese am Beispiel von CiscoWorks näher erläutert. CiscoWorks ist in mehrere funktionale Module aufgeteilt. Die relevanten Module sind der Device Fault Manager, der Campus Manager, der Resource Manager, der Internetwork Performance Monitor und das CiscoView. Für eine detaillierte Beschreibung der einzelnen Module vergleiche (Cisco, 2008).

Eine Zusammenfassung der relevanten Funktionalitäten erfolgt in ***Tabelle 3.9***. Die Funktionen sind für die Entwicklung von FLASH interessant und werden in Abschnitt ***4.4*** mit in den Anforderungskatalog aufgenommen oder für die Weiterentwicklung (vgl. Abschnitt ***0***) berücksichtigt.

Tabelle 3.9: Kommerzielle Netzwerk Management Software Funktionen

Funktion	Beschreibung
Inventarisierungs Management	Für die Inventarisierung notwendigen Geräteinformationen können ausgelesen werden.
Konfigurations Management	Das Speichern und Laden von Systemkonfigurationen
Software Image Management	Überprüfen, Update und Sichern der Betriebssysteme
Log	Informationen von den zu überwachenden Systemen und von dem überwachenden System
Geräte Änderungsmanagement	Speicherung der Informationen über Änderungen an der Hardware einzelner Systeme
Topologymanagement	Automatisches auslesen und Erstellung der Systemumgebung
Troubleshooting	Aktive Erkennung von Verbindungsproblemen
Reports	Automatische Erstellung von Berichten und die Möglichkeiten des Exportes

3.6.3 SKRIPTE (SKRIPTSPRACHEN)

Eine mögliche technische Alternative für die Umsetzung der Anforderungen an FLASH ist die Umsetzung in einer Skriptsprache oder die Integration einer Skriptsprache in das Gesamtsystem von FLASH. Netzwerkadministratoren verwenden Skripte in größere Netzwerkumgebungen für die Unterstützung bei sich wiederholenden Wartungsarbeiten und dem Testen von Einstellungen. Dafür wird unter anderem Tool Command Language (TCL, siehe (Libes, 1995)) und Expect (siehe (Don Libes, 2006)) als Weiterentwicklung von TCL eingesetzt. Expect automatisiert ausschließlich Kommandozeilen Befehle und bietet keine Unterstützung für grafische Oberflächen.

Das folgende Expect Beispiel zeigt den Verbindungsaufbau zu einem Netzwerkgerät mit Passwortabfrage sowie den abschließenden Verbindungsabbau.

```
# Variablen $remote_device, $password
# Öffnen der TELNET Verbindung
spawn telnet $remote_device
# Warten auf die Ausgabe nach dem Passwort
expect "password:"
# Senden des Passwortes und warten auf Kommandozeilen
Prompt
send "$password\r"
expect "Device>"
# Beenden der TELNET Verbidnung.
send "exit\r"
```

Expect dient als Schnittstelle zwischen verschiedenen Programmen. Die Grundidee bei der Entwicklung ist gewesen, die vorhandenen Systeme optimal auszulasten und deren vorhandenen Funktionen zu nutzen. Die Probleme sollten nicht ausschließlich versucht werden in Expcet zu lösen. Daher wird Expect vor allem in Zusammenarbeit mit kommerziellen Softwarelösungen eingesetzt. Diese Lösungen haben keine eigenen, ausreichend funktionalen Skriptmodule und nutzen deshalb die Stärken von Expect.

3.7 ENTWICKLUNGSUMGEBUNG

Die vorhandenen Systeme und deren Umsetzung sowie die zugrunde liegende Systemplattform wurden in ***Tabelle 3.10*** zusammengefasst.

Tabelle 3.10: Umsetzung der vorhandenen Systeme

System	Programmiersprache	Betriebssystem
TELNET Client	Java	Windows Server 2003
Scheduler	C#	Windows Server 2003
Webfrontend	C# (ASP.NET)	Windows Server 2003

Für die Entwicklung des TELNET Clients in Java wurde sich entschieden, da der Client auf den Teilnehmer PCs laufen muss und diese nicht von Fast Lane konfiguriert werden. Java bietet dabei die flexibelste Plattformunabhängigkeit und kann auf jedem System mit installierter Java Runtime Environment (JRE) ausgeführt werden.

Das Webfrontend wurde in .NET und C# entwickelt, da die Lizenzen für den Microsoft Webserver IIS und den Datenbankserver auf Microsoft SQL 2005 vorhanden sind. Dadurch wurden die Entwicklungskosten so gering wie möglich gehalten.

Um die Entwicklungskosten für FLASH so gering wie möglich zu halten, soll auf die vorhandenen Software Lizenzen und Systeme zurückgegriffen werden. Die zur Verfügung stehende Software wird in ***Tabelle 3.11*** aufgeführt und die grundlegenden Hardwaredaten des Zielsystems in ***Tabelle*** 3.12 angegeben. Frei verfügbare Software Lösungen können in die Entwicklung mit einbezogen werden.

Tabelle 3.11: Vorhandene Software Lizenzen

Software	Version	Patchlevel	Funktion
Microsoft Windows Server	2003 Enterprise	Service Pack 1	Betriebssystem
Microsoft SQL Server	2005 Standard	Service Pack 4	Datenbankmanagementsystem
Microsoft Visual Studio	2008 Professional	-	Entwicklungsumgebung

Tabelle 3.12: Zielsystem

System	Beschreibung
Anwendungsserver (APP)	- DELL Poweredge 1950 - Zwei Intel® Xeon 5000 Dual-Core-Prozessoren mit 3,0 GHz - 4 GB RAM

4 ANFORDERUNGSKATALOG

In diesem Kapitel werden die Anforderungen aufgelistet, die von der Firma Fast Lane GmbH an das System gestellt werden und die Anforderungen, welche sich aus einer Dozentenbefragung ergeben haben. Im Anschluss folgen eine Zusammenfassung der Anforderungen und deren Bewertungen sowie eine Machbarkeitsanalyse.

4.1 ANFORDERUNGEN DER FIRMA FAST LANE

Die Firma Fast Lane hat an das System die folgenden Anforderungen gestellt. Die Anforderungen sind in die drei Bereiche funktionale Anforderungen, nichtfunktionale Anforderungen und Randbedingungen gegliedert.

4.1.1 FUNKTIONALE ANFORDERUNGEN

1. Anforderung: Zurücksetzen der Netzwerkgeräte in einen definierten Ausgangszustand

Beschreibung: Der Begriff Netzwerkgeräte umfasst alle Geräte, die über eine Kommandozeilen Schnittstelle verfügen. Unter einem definierten Ausgangszustand ist ein Systemzustand des Netzwerkgerätes zu verstehen, der entweder durch das Zurücksetzen in den Auslieferungszustand oder in erweiterter Form durch zusätzliches Einspielen einer Konfiguration erreicht werden kann.

2. Anforderung: Vordefinierte Konfigurationen für die Netzwerkgeräte einspielen

Beschreibung: Diese Anforderung baut auf Anforderung 1 auf und ermöglicht es eine vordefinierte Konfiguration in ein Netzwerkgerät einzuspielen. Dabei kann es unter Umständen vorher notwendig sein, den definierten Ausgangszustand des Gerätes herzustellen.

3. Anforderung: Folgende Cisco Netzwerkgeräte müssen kontrolliert werden:
C-1841 (Router), C-2960 (Switch), C-2950 (Switch), C-3640 (Router)
Das System muss unabhängig vom Hersteller und des Gerätetyps in der Lage sein, weitere Geräte zu kontrollieren, ohne Änderungen am System vornehmen zu müssen.

Beschreibung: Bei den aufgeführten Netzwerkgeräten handelt es sich um Router und Switch Modelle, die über eine Kommandozeilen Schnittstelle verfügen. Die Anforderung wurde auf diese Gerätemodelle bezogen, da sie in dem Remote Labor für den Cisco Grundlagenkurs verwendet sind und dieser als Referenz für dieses Projekt gilt. Des Weiteren lässt sich das darauf entwickelte System

auf weitere Labore übertragen, da die Netzwerkgeräte alle grundlegenden Eigenschaften der Geräte anderer Labore widerspiegeln.

4. Anforderung: Log Funktion für die automatischen Abläufe und möglicher Status Meldungen

Beschreibung: Das System soll automatisch arbeiten. Deshalb muss den Technikern eine Möglichkeit gegeben werden sich über den Status einzelner Abläufe zu informieren. Ebenso soll das System Meldungen über Fehler oder Störungen an die Techniker weitergeben, sodass diese in letzter Instanz noch korrigierend eingreifen können.

4.1.2 NICHTFUNKTIONALE ANFORDERUNGEN

1. Anforderung: Anpassungen teilweise durch Techniker möglich

Beschreibung: Der Teilbereich des Systems, der für das Zurücksetzen und das Einspielen von Konfigurationen zuständig ist, soll auch von nicht Softwareentwicklern angepasst werden.

2. Anforderung: Nebenläufige Ausführung von Aufgaben

Beschreibung: Das System soll für den Einsatz in großen Laborumgebungen einsetzbar sein. Es muss relativ viele, parallel ausgeführte Abläufe bearbeiten. Dadurch dürfen keine Fehlfunktionen auftreten und die Aufgaben müssen fehlerfrei bearbeitet werden.

3. Anforderung: Zurücksetzen aller Geräte eines Labors innerhalb einer Stunde

Beschreibung: Für die Zurücksetzprozeduren steht ein Zeitraum von einer Stunde zur Verfügung. Wenn es technisch möglich ist, muss das Zurücksetzen aller Geräte eines Labors innerhalb dieser Stunde erfolgen. Dadurch soll ein Mehrschichtbetrieb der Labore ermöglicht werden (vgl. Abschnitt ***2.6***).

4. Anforderung: Schnittstellenbereitstellung

Beschreibung: Die Aufträge für FLASH werden vom Scheduler des Webservers bereitgestellt. Die Kommunikation über eine entsprechende Schnittstelle muss bereitgestellt werden.

5. Anforderung: Das zu entwickelnde System soll an allen Standorten der Firma Fast Lane operieren können

Beschreibung: FLASH wird seine Aufgaben an allen drei Standorten bearbeiten. Es muss in der Lage sein, seine Abläufe an jedem dieser Standorte auszuführen. Des Weiteren soll die Erweiterung auf neue Standorte berücksichtigt werden.

6. Anforderung: Betriebssicherheit

Beschreibung: Das System soll für eine garantierte Ausführung der automatisierten Abläufe sorgen. Treten während der Ausführung vom System nicht korrigierbare Störungen auf, muss das System diese erkennen und über ein Log Modul melden.

7. Anforderung: Wartbarkeit

Beschreibung: Das zu entwickelnde System soll nachhaltig einfach zu warten sein. Dies schließt zum einen den Programm Code als auch das fertige System ein. Zu berücksichtigen ist, dass das Unternehmen global aufgestellt ist und Englisch als Sprache für die Dokumentation verwendet wird.

4.1.3 RANDBEDINGUNGEN

1. Anforderung: Anfang des zweiten Quartals 2008 einsatzbereiter Prototyp

Beschreibung: Ein Prototyp des zu entwickelnden Systems soll Anfang des zweiten Quartals fertiggestellt sein.

4.2 DOZENTEN BEFRAGUNG

Im Rahmen dieser Bachelorarbeit wurde ein einfacher und schnell zu beantwortender Fragenkatalog erstellt, um zu verhindern, dass während der Entwicklung neue und geänderte Anforderungen auftreten. Anpassungen der Anforderungen zu einem späteren Zeitpunkt in der Entwicklung werden so vermieden. Der Fragenbogen ist an die Dozenten gerichtet, die ständig im Umgang mit den Teilnehmern und der Durchführung der Schulungen beschäftigt sind. Die folgenden zwei Gründe haben zu der Durchführung der Dozenten Befragung geführt.

Der erste Grund ist die in Abschnitt ***1.1*** erwähnte Tatsache, die neuen technischen Möglichkeiten in den Schulungsbetrieb mit einzubinden. Funktionalitäten wie das Zurücksetzen eines Gerätes in den Ausgangszustand würde den Teilnehmern, bei einer selbst erzeugten fehlerhaften Konfiguration, eine schnelle Möglichkeit zur Verfügung stellen, wieder neu anzufangen.

Der zweite Grund ist die Entwicklung eines objektiven Anforderungsprofils für das zu entwickelnde System. Aussagen und Informationen kommen direkt aus den Erfahrungen der alltäglichen Praxis.

Insgesamt wurden 11 Dozenten befragt, die für das Unternehmen Fast Lane tätig sind. Nicht alle Dozenten haben den gleichen Erfahrungslevel und Kenntnisstand. Daher wurde für die Gewichtung der Antworten ein Feld auf dem Fragebogen eingefügt, welches die Berufserfahrung der befragten Dozenten im Trainingsumfeld widerspiegelt. ***Tabelle 4.1*** zeigt die verschiedenen Erfahrungslevel.

Tabelle 4.1: Dozenten Level

Level	Bezeichnung	Berufserfahrung in Jahren
1	Trainee	1-2
2	Advanced	3-8
3	Expert	>8

Die Fragen Zielen nicht direkt auf die Anforderungen der Firma Fast Lane, da diese aus Sicht der Techniker kommen. Es geht hierbei vielmehr um die Funktionalitäten, die eine automatisierte Lösung möglich macht und deren Einsatz während der Schulungen. Das Ziel ist es, den Dozenten und Teilnehmer einen komfortablen Umgang mit der Trainingsumgebung zur Verfügung zu stellen. Für die Durchführung der Befragung wurden folgende drei Fragen genutzt.

1. Frage: *Welche Funktionalitäten soll ein Geräte Management anbieten?*

Beschreibung: Mit dieser Frage sollen die Funktionen herausgefunden werden, die während der Schulungen eingesetzt werden können und aus der zu entwickelnden automatisierten Lösung hervorgehen.

2. Frage: *Sollen der Dozent und der Teilnehmer die gleichen Funktionalitäten nutzen können?*

Beschreibung: Diese Frage beschäftigt sich nicht mit den technischen Funktionen, sondern mit den Zugriffsrechten und den logischen Abhängigkeiten, denen die technischen Funktionen zugrunde liegen. Man kann Änderungen an den Core Geräten nur schwer durchführen ohne sofort alle POD Geräte damit zu beeinflussen. Das heißt, ein Teilnehmer darf, wenn überhaupt, nur eingeschränkte Rechte auf die Core Geräte haben. Die konkreten Probleme sollen die Antworten auf diese Frage liefern.

3. Frage: *Sonstige Anmerkungen oder Ideen?*

Beschreibung: Die Gedanken und Überlegungen der Dozenten zu der Geräte Automatisierung sollen hier aufgedeckt und dokumentiert werden.

Eine zusammenfassende Auswertung der Ergebnisse erfolgt im nächsten Abschnitt. Die ausführlichen Ergebnisse sind in elektronischer Form auf der beigefügten CD-ROM hinterlegt.

4.3 AUSWERTUNG DOZENTEN BEFRAGUNG

Die Auswertung der Antworten sollte die technischen und systemrelevanten Aussagen von den persönlichen Präferenzen der Dozenten trennen. Die Auswertung der Antworten erfolgt getrennt für die jeweilige Fragestellung und bezieht sich dabei ausschließlich auf die technischen und systemrelevanten Anteile der Antworten.

1. Frage Auswertung: Wie vermutet spiegelten sich die technischen Anforderungen der Firma Fast Lane aus Sicht der Dozenten und Teilnehmer wider. Diese Anforderungen umfassen die grundlegenden Funktionen, wie das Einspielen und Speichern von Konfiguration. Zusätzlich wurden auch neue Anforderungen genannt. Diese betreffen vor allem Punkte für eine optimale Vorbereitung der Geräte auf die Folgeschulung. Zu erwähnen sind dabei vor allem die Folgenden:

    1.1 Prüfen der korrekten Betriebssystem Version auf dem Netzwerkgerät.

    1.2 Überprüfung des geräteinternen Dateisystems. Auf der einen Seite nach ungewollt verbliebenen Sitzungsdateien der Teilnehmer und auf der anderen Seite auf nicht mehr vorhandene Systemdateien, die für die Bereitstellung einer definierten Ausgangssituation notwendig sind.

    1.2 Gruppierungsmöglichkeiten für Geräte eines Labors. Möglichkeit 1-n Geräte eines PODs, Core oder Labors auszuwählen und gleichzeitig eine Aktion zu starten.

2. Frage Auswertung: Bei den Antworten zu dieser Frage ging es um die logische Rechteverteilung zwischen Dozent und Teilnehmer. Diese beziehen sich auf die Ausführungsmöglichkeiten der zur Verfügung stehenden Funktionen. Die relevanten Aussagen sind wie folgt:

    2.1 Die Teilnehmer sollen in der Lage sein, nur im Rahmen ihres zugeteilten PODs bestimmte Funktionen zu starten. Dabei müssen Abhängigkeiten zwischen Core und den anderen PODs berücksichtigt werden. Ein Teilnehmer darf durch seine zur Verfügung stehenden Möglichkeiten nicht ungewollt die Übungen der anderen POD Teilnehmer beeinflussen.

    2.2 Der Dozent muss zu jeder Zeit in der Lage sein, die Aktionen der Teilnehmer zu kontrollieren. Dies betrifft sowohl die Geräte des Core Bereichs als auch die der einzelnen PODs. Der Dozent kann zu jeder Zeit die Kontrolle der Geräte übernehmen.

3. Frage Auswertung: Keine weiteren Aussagen, die nicht schon durch die Auswertung von Frage 1 und / oder Frage 2 abgedeckt wurden.

Die Auswertung der Dozenten Befragung hat ein paar zusätzliche Anforderungen ergeben und einige wesentliche Überlegungen zum Thema logische Abhängigkeiten der Netzwerkgeräte zueinander. Diese Anhängigkeiten sind bei der Bereitstellung automatischer Funktionen für die Dozenten und Teilnehmer im Schulungsbetrieb zu berücksichtigen.

Im Abschnitt *4.4* werden die Anforderungen der Firma Fast Lane und die zusätzlich aus der Dozenten Befragung erhaltenen zusammengefasst und konkret spezifiziert.

4.4 ZUSAMMENFASSUNG DER ANFORDERUNGEN

Folgende finale Anforderungen haben sich aus der Dozenten Befragung und den von der Firma Fast Lane vorgebenden Anforderungen ergeben. Die Anforderungen sind in die drei Bereiche funktionale und nichtfunktionale Anforderungen sowie Randbedingungen gegliedert.

Die Einteilung der Anforderungen erfolgt nach den Kriterien der ISO/IEC 9126 (ISO.ORG). In der ersten Spalte wird ein Index eingeführt, der die Bezugnahme aus anderen Kapiteln erleichtert und darauf hinweist, dass es sich um eine funktionale (Fx) beziehungsweise nichtfunktionale (nFx) Anforderung oder eine Randbedingung (RBx) handelt. In der zweiten Spalte wird die Version der Anforderung angegeben, um Änderungen einer Anforderung während der Bachelorarbeit zu kennzeichnen. Die dritte Spalte spiegelt die möglichen Abhängigkeiten der Anforderungen untereinander wieder. Die Spalte Anforderung beinhaltet eine textuelle Kursbezeichnung der jeweiligen Antwort, wo hingegen die fünfte Spalte eine Beschreibung der Anforderung beinhaltet. Die letzte Spalte gibt die Priorität der Anforderung wieder. 1 – „must-have“, 2 – „should have“ und 3 – „nice-to-have“.

Anforderungen mit Priorität 1 werden im Rahmen dieser Bachelorarbeit umgesetzt. Anforderungen der Prioritätsstufe 2 werden im Anschluss an die Entwicklung des Systems umgesetzt und Stufe 3 Anforderungen nach Abschluss der Umsetzung von Priorität 2 Anforderungen.

Tabelle 4.2: Funktionale Anforderungen

Index	Version	Abhängig	Anforderung	Kurzbeschreibung	Priorität
F1	1.0	-	System_Reset	Vordefinierten Default Zustand eines Netzwerkgerätes herstellen	1
F2	1.0	F1	System_CFG_Load	Vordefinierte Konfigurationen laden	1
F3	1.0	F1	User_CFG_Save	Benutzer Konfiguration sichern	2
F4	1.0	F1	User_CFG_Load	Benutzer Konfiguration laden, die zuvor gesichert wurde	2
F5	1.0	F1	File_Read	Dateien im FLASH Speicher der Netzwerkgeräte auslesen	2
F6	1.0	F1	File_Erase	Dateien aus dem FLASH Speicher der Netzwerkgeräte löschen	2
F7	1.0	F1	File_Load	Dateien über die Konsolen Verbindung laden	3
F8	1.0	F1	File_Save	Dateien über die Konsolen Verbindung sichern	3
F9	1.0	-	Log Funktion	Fehler und Status Meldungen mit loggen und auswerten bzw. ausgeben.	1
F10	1.0	F1	Output_Processing	Um eingespielte Konfigurationen anzeigen zu lassen / Status und Register Informationen auszulesen	2
F11	1.0	F10	IOS_Check	Aktuell laufende Version des Betriebssystems ermitteln	3
F12	1.0	-	Netzwerkgeräte	Folgende Cisco Geräte müssen kontrolliert werden: C-1841 (Router), C-2960 (Switch), C-2950 (Switch), C-3640 (Router)	1

Tabelle 4.3: Nichtfunktionale Anforderungen

Index	Version	Abhängig	Anforderung	Kurzbeschreibung	Priorität
nF1	1.0	-	Anpassungen teilweise durch Techniker möglich	Die Verhaltensmuster müssen auch von einem Techniker ohne Programmierkenntnisse angepasst werden können.	1
nF2	1.0	-	Nebenläufige Ausführung von Aufgaben	In großen Labor Umgebungen eine Vielzahl von Aufgaben parallel verarbeiten.	1
nF3	1.0	-	Zeitkritische Rücksetzprozeduren	Zurücksetzen aller Geräte eines Labors innerhalb einer Stunde	1
nF4	1.0	-	Schnittstellenbereitstellung	Stellt die Funktionen des zu entwickelnden Systems den anderen Modulen des Remotelab Management Systems zur Verfügung.	1
nF5	1.0	-	Standortübergreifend	Standortübergreifender Einsatzmöglichkeiten	2
nF6	1.0	-	Betriebssicherheit	Garantierte Ausführung oder Fehlererkennung bzw. Meldung)	1
nF7	1.0	-	Wartbarkeit	Nachhaltige Wartbarkeit des Programm Codes als auch des laufenden Systems	1
nF8	1.0	-	System Sicherheit	Sicherheit des Gesamtsystems	1

Tabelle 4.4: Randbedingungen

Index	Version	Abhängig	Anforderung	Kurzbeschreibung	Priorität
RB1	1.0	-	Anfang Q2/08 soll das Prototyp System im Betrieb sein		1

Zu den Anforderungen der Firma Fast Lane und der Dozenten Befragung ist die nichtfunktionale Anforderung nF8 hinzu gekommen. Aufgrund der Analyse der zugrunde liegenden Infrastruktur muss die Sicherheit des zu entwickelnden Systems berücksichtigt werden und sichergestellt sein, dass die korrekte Funktionsfähigkeit nicht durch Eingriffe Dritter beeinflusst werden kann.

4.5 ABGRENZUNG

Bei der Analyse der bestehenden Systeme und der Erarbeitung der Anforderungen haben sich auch Anforderungen herausgestellt, die aufgrund ihres Umfangs und aus zeitlicher Sicht nicht im Rahmen dieser Bachelorarbeit umgesetzt werden. Es handelt sich dabei um die folgenden Anforderungen, die als Abgrenzungen beschrieben sind.

1. Abgrenzung: Zu welchem Zeitpunkt die zu entwickelnden Funktionen gestartet werden

Beschreibung: Für den Einsatz der Remote Labore im Mehrschichtbetrieb (vgl. Abschnitt ***2.6***) muss ein entsprechendes Planungs- und Zeitmanagement System vorhanden sein. Die vorhandene Systemumgebung verwendet den Scheduler, der ausschließlich die Zugangszeiten für die Remote Labore regelt. Diese Zeiten werden vorher manuell von einem Planungsteam festgelegt. Für den vollen Einsatz von FLASH muss ein flexibleres, funktionsfähigeres und automatisch arbeitendes Scheduling System entwickelt werden.

2. Abgrenzung: Einbindung in die grafische Oberfläche des Websystems

Beschreibung: Der Zugriff erfolgt, wie in Abschnitt ***2.3*** beschrieben, über ein grafisches Webinterface. Die Integration der Funktionalitäten in dieses System erfolgt im Anschluss an dieser Bachelorarbeit.

3. Abgrenzung: Logische Rechteverwaltung

Beschreibung: Die Auswertungen 1.2, 2.1 und 2.2 der ***Dozenten Befragung*** und der in Abschnitt ***2.2*** beschriebene Aufbau der einzelnen Remote Labore zeigen mögliche Abhängigkeiten zwischen den Core und POD Geräten auf. Diese müssen berücksichtigt werden. Wenn ein Teilnehmer für seinen POD eine andere Konfiguration einspielen möchte und dafür Änderungen an den Core Komponenten notwendig sind, würde diese Aktion auch die anderen PODs beeinflussen. Ob und in wie weit dies zugelassen werden kann, muss von einem zusätzlichen Richtlinien System geregelt werden. Die Firmen Planungen sehen dafür ein zusätzliches System vor oder die Integration eines zusätzlichen Moduls in den Scheduler.

4. Abgrenzung: Physikalische Abhängigkeiten berücksichtigen

Beschreibung: Die für diese Arbeit relevanten Netzwerkgeräte sind aus physikalischer Sicht unabhängig voneinander. Es gibt aber Geräte wie Filer[9] der Firma NetApp, die physikalische Abhängigkeiten aufweisen. In diesem Fall sind zwei an sich unabhängige Geräte in einem Gehäuse verbaut und nutzen Ressourcen wie Netzteile und Plattenspeicher gemeinsam. Für das Ausschalten eines Filers müssen beide Netzteile ausgeschaltet werden. Die Möglichkeit, verschiedene Geräteklassen zu verarbeiten (siehe Abschnitt ***5.2***, Erweiterter Lösungsansatz) wird beim Design berücksichtigt. Die Implementierung dieser Spezialfälle erfolgt nicht im Rahmen dieser Arbeit.

[9] Filer oder auch Network Attached Storage (NAS) Gerät bezeichnet einfach zu verwaltende Dateiserver, bestehend aus einem eigenständigem Host mit eigenem Betriebssystem.

4.6 VORTEILE DES ZU ENTWICKELNDEN SYSTEMS

Die Vorteile, die das zu entwickelnden Systems mit sich bringt, werden in diesem Abschnitt näher erläutert.

Die Existenzberechtigung des neuen Systems lässt sich an den Zeiteinsparungen begründen, wie in Abschnitt ***2.5.1*** detailliert erläutert. Die Automatisierung entlastet die Techniker der Remote Labore und ihre frei gewordene Arbeitskraft kann für Service Tätigkeiten eingesetzt werden. Dadurch lässt sich nachhaltig die Kundenzufriedenheit steigern.

Durch die Vermeidung der in Abschnitt ***2.5.2*** beschriebenen Probleme wird die Qualität der Laborvorbereitung verbessert. Im Ergebnis werden negative Bewertungen durch die Teilnehmer diesbezüglich vermieden. FLASH stellt hier einen grundlegenden Sicherheitsfaktor für das Unternehmen dar (vgl. Abschnitt ***2.5.2***).

Des Weiteren gibt es nicht messbare, qualitative Verbesserung des Schulungsangebotes durch die Bereitstellung von zusätzlichen Möglichkeiten für den Umgang mit den Laboren. Ein Dozent kann zum Beispiel schnell den Zustand des gesamten Labors ändern, wenn er bei aufeinander aufbauenden Lab Übungen einfach eine überspringt und die benötigten Konfigurationen automatisch einspielen lässt. Der gesamte Ablauf der Schulungen wird flüssiger, da der Dozent sich auf das Wesentliche konzentrieren kann und nicht mit Konfigurationsaufgaben beschäftigt ist. Ein ähnliches Szenario, welches während der Dozenten Befragung erwähnt wurde, ist die unterschiedliche Lerngeschwindigkeit der Teilnehmer. Die Funktionen der Automation kann eingesetzt werden, um den Teilnehmer Gruppen während der Schulung verschiedene, Ihrem Lernfortschritt angepasste Konfigurationen einzuspielen.

Zum anderen besteht die Möglichkeit die Labore für eine zeitversetzte parallele Nutzung im Mehrschichtbetrieb (vgl. Abschnitt ***2.6***) zu verwenden. Die produktive Einsatzzeit der Hardware lässt sich dabei mindestens verdoppeln. Das führt dazu, dass die Investitionskosten viel schneller eingespielt werden. Das folgende Beispiel zeigt den Vorteil aus finanzieller Sicht. Das für die Arbeit relevante Labor für den ICND1 Kurs dient auch an dieser Stelle als Grundlage für die Berechnung in ***Tabelle 4.5***. Die Preise für die aufgeführten Netzwerkgeräte beziehen sich auf das Gerät und die optional verbauten Module mit zusätzlichen Schnittstellen. Als Grundlage für die angegebenen Preise wurden die Preislisten der Hersteller mit Stand 03/2008 herangezogen.

Tabelle 4.5: Investitionskosten - ICND1 Labor

Position	Bezeichnung	Einsatzbereich	Menge	Preis in €	Gesamtbetrag
1	Router c1841	POD	8	1.795,00	14.360,00
2	Switch c2960-8	POD	8	895,00	7.160,00
3	Router c3640	Core	1	600,00	600,00
4	Switch c2950-24	Core	2	995,00	1.990,00
5	Switch c2960-24	Core	1	1.295,00	1.295,00
6	Terminal Server c2511-RJ	Access	1	250,00	250,00
7	Switch c2900-24	Access	1	150,00	150,00
8	APC-AP7952	Access	1	830,00	830,00
9	Verbrauchsmaterial	-	1	150,00	150,00
				Summe	26.785,00

Die einmaligen Investitionskosten betragen ca. 26.785,00 €. Durch die Verwendung der Labore für die eigenen Schulungen und die Vermietung der Labore an Partner ergeben sich Einnahmen, die firmenintern mit 1.300,00 € pro Kurswoche für dieses Labor eingerechnet werden. Bei einer einfachen wöchentlichen Nutzung werden die Investitionskosten nach 21 Wochen eingespielt. Wird die Nutzung durch den Mehrschichtenbetrieb gesteigert, verringert sich die Zeit entsprechend.

Die vorangegangene Berechnung ist ohne die laufenden Kosten für Betriebsstrom, Klimatisierung und Technikerzeiten durchgeführt worden und entspricht nicht vollständig den realen Bedingungen. Sie zeigt deutlich den Vorteil des Mehrschichtbetriebes aus Sicht der Investitionskostenamortisierung.

Zusammenfassend sind die Vorteile einer automatisierten Lösung:

1. Interne Arbeitszeiteinsparungen
2. Zeitversetzte parallele Nutzung der einzelnen Remote Labore
3. Verbesserte Kundenzufriedenheit durch
    a. Qualitätssteigerung des aktiven Laborbetriebes
    b. Qualitätssteigerung der Laborvorbereitung
4. Schnellere Amortisierung der Investitionskosten pro Labor

4.7 MACHBARKEITSANALYSE

Alle Anforderungen der Prioritätsstufe 1 sollen im Rahmen dieser Bachelorarbeit umgesetzt werden. Dies beinhaltet die funktionalen Anforderungen F1, F2, F5, F6, F9 und F11 sowie die nichtfunktionalen Anforderungen nF1, nF2, nF3, nF4, nF5, nF7 und nF8. Des Weiteren gilt die Randbedingung RB1 als Zieltermin für die Entwicklung des Systemprototyps. Die Umsetzung dieser Anforderungen ist für den eingeplanten Zeitraum als realistisch anzusehen.

Anhand der zu realisierenden Anforderungen und der zur Verfügung stehenden Mittel (siehe Abschnitt *3.7*) lässt sich ein Grundsystem entwickeln, dass den ursprünglichen Anforderungen der Firma Fast Lane gerecht wird. Darauf aufbauend können dann die weiteren Anforderungen der Prioritätsstufen 2 und 3 entwickelt und implementiert werden.

5 DESIGN

Die Ergebnisse der Anforderungsanalyse bilden die Grundlage für das Design von FLASH. Dabei werden zum einen die Designkriterien, die das Gesamtsystem betreffen, analysiert und entsprechende Lösungswege konzipiert. Zum anderen werden die Konzepte für die einzelnen Umsetzungen der technischen Anforderungen aufgeführt.

Für die Erstellung des Designs sind die nichtfunktionalen und funktionalen Anforderungen entscheidend. Aus den funktionalen Anforderungen lassen sich die technischen Anforderungen ableiten. Die nichtfunktionalen Anforderungen entsprechen den Designkriterien. In den Abschnitten *5.1* und *5.2* werden deshalb die beiden Anforderungsbereiche getrennt betrachtet und in die Designerstellung mit einbezogen.

5.1 DESIGNKRITERIEN

Die Designkriterien werden aus den nichtfunktionalen Anforderungen (vgl. *Tabelle 4.3*) abgeleitet. Im Rahmen dieses Projektes sollen die Anforderungen nF1, nF2, nF3, nF4, nF6, nF7 und nF8 umgesetzt werden. Zusammengefasst sind diese:

nF1: Anpassungen teilweise auch durch Techniker möglich

nF2: Nebenläufige Ausführung von Aufgaben

nF3: Zeitkritische Rücksetzprozeduren

nF4: Schnittstellenbereitstellung

nF7: Wartbarkeit

nF6: Betriebssicherheit

nF8: System Sicherheit

In den folgenden Abschnitten werden die Anforderungen und die Umsetzung jedes Designkriteriums erläutert.

5.1.1 NF1 - ANPASSUNGEN TEILWEISE AUCH DURCH TECHNIKER MÖGLICH

Eine grundlegende Herausforderung stellt die Anforderung nF1 dar. Dabei sollen die von FLASH zu verarbeitenden Verhaltensbeschreibungen der Netzwerkgeräte durch Techniker der Remote Labore angepasst werden. Diese Anpassungen können in Form eines in FLASH integrierten Editors erfolgen oder über selbsterklärende Konfigurationsdateien, die von den Technikern bearbeitet werden. Die Umsetzung dieses Problems ist das erste Ziel dieser Anforderung.

Das zweite Ziel hat sich bei der Analyse der Systemumgebung herausgestellt. Es folgt aus dem Problem, das bei Änderungen der IOS Versionen (vgl. Betriebssystem in ***3.2.2.1***) auftritt. Das folgende Beispiel verdeutlicht die zugrunde liegende Problematik und welche Auswirkungen im laufenden Betrieb auftreten können.

Die Remote Labore sind 24 Stunden am Tag und 7 Tage die Woche (24 x 7) im Betrieb (siehe Abschnitt ***2.6***). Daher läuft FLASH ohne Unterbrechung. Wird ein neues Labor installiert oder eine Wartung auf einem Bestehenden durchgeführt, die zugleich eine Veränderung der von FLASH verwendeten Verhaltensbeschreibungen erfordert, kann dies unter Umständen nicht rechtzeitig erfolgen. Würden die Verhaltensbeschreibungen zur Kompilierzeit festgelegt werden, wäre es nicht möglich, während des Systembetriebes Änderungen an den Beschreibungen vorzunehmen. Zuerst müssten alle laufenden Tasks von FLASH abgeschlossen werden, im Anschluss FLASH beendet, den Code anpasst, erneut kompiliert und FLASH wieder gestartet werden. Werden die Verhaltensbeschreibungen dynamisch zur Laufzeit des Automationssystems geladen, kann FLASH aktiv bleiben. Daher sollen die Verhaltensmuster der Netzwerkgeräte zur Laufzeit von FLASH angepasst werden.

Lösungsansatz

Eine Lösung für die Umsetzung der Konfigurationsabläufe für die Netzwerkgeräte ist zu realisieren. Diese Abläufe werden durch die in Abschnitt ***3.3*** abgebildeten Ablaufdiagramme beschrieben und bilden die Grundlage für den Lösungsentwurf. Die zu lösenden Fragen sind zum einen, wie diese Ablaufdiagramme in konkrete Programmabläufe umgesetzt und zum anderen, wie diese abgelegt werden können. Da die Abläufe während der Laufzeit von FLASH verändert werden sollen, besteht die Möglichkeit, diese auszulagern und entsprechend zur Laufzeit zu laden.

Ablaufdiagramme lassen sich in eine Automatenstruktur überführen bzw. sich daraus ableiten. Im ersten Schritt muss eine Struktur gefunden werden, die eine geordnete Transformation der Diagramme ermöglicht. Im zweiten Schritt wird die Entscheidung getroffen, wie diese Struktur von FLASH interpretiert und abgearbeitet werden soll. Die Umsetzung kann in den zwei Varianten sequenzielle Abarbeitung oder Verarbeitung der abgeleiteten Automatenstrukturen realisiert werden.

Für die Umsetzung stehen vier Alternativen zur Verfügung. Im Folgenden werden diese verglichen und bewertet. Im Anschluss wird die Entscheidung für die eigene Lösung begründet.

Bewertungskriterien

Die Bewertung erfolgt anhand der folgenden fünf Kriterien:

1. Kriterium: *Anpassungen durch Techniker möglich*

Beschreibung: Das System soll Anpassungen und Erweiterungen auch durch Techniker möglich machen.

2. Kriterium: *Nicht fest codiert*

Beschreibung: Dieses Kriterium spiegelt die Haupteigenschaft dieser Anforderung wieder. Siehe dazu die beiden Teilziele in Abschnitt ***5.1.1***.

3. Kriterium: *Abbildung komplexerer Abläufe möglich*

Beschreibung: Nicht alle Verhaltensmuster setzen sich aus einfachen, sequenziellen Abläufen zusammen. Es gibt komplexere Anforderungen (vgl. Abschnitt ***3.3***). Deshalb muss die Möglichkeit gewährleistet sein, auch komplexere Muster umzusetzen.

4. Kriterium: *Übersichtlichkeit / Wartbarkeit*

Beschreibung: Die Übersichtlichkeit und Wartbarkeit bezieht sich nur auf die umzusetzende Anforderung. Die Umsetzung soll so erfolgen, dass auch größere Verhaltensmuster beschrieben werden können, ohne dass die Übersichtlichkeit verloren geht. Die Erstellung neuer Verhaltensmuster soll auf einfache Art und Weise möglich sein. Die Pflege und Wartung bestehender Muster soll ebenso einfach umzusetzen sein.

5. Kriterium: *Erweiterbarkeit*

Beschreibung: Die Umsetzung soll es ermöglichen auch Erweiterungen des Funktionsumfangs an der bestehenden Struktur vornehmen zu können.

Alternativen

Zur Auswahl standen nach eingängiger Recherche folgende drei Alternativen:

1. Alternative: *Skriptsprachen (vgl.* (Ousterhout, 1993)*)*

2. Alternative: *Automatenumsetzung mit „Switch Case“ Sprachelement (vgl.* (Liberty, 2001)*)*

3. Alternative: *Automatenumsetzung mit Pattern „State Pattern“(vgl.* (Bishop, 2008),148-157)

4. Alternative: *Interpretation von Automaten, die in XML definiert sind.*

Folgend werden die Vor- und Nachteile der oben genannten Alternativen kurz erläutert.

1. Skriptsprachen, insbesondere die in ***3.6.3 Skripte (Skriptsprachen)*** erläuterte Variante Expect, hat die Vorteile, dass sich auf einfache Art und Weise Kontrollabläufe für TELNET Anwendungen entwickeln lassen. Des Weiteren können die Skripte über Batch Jobs oder direkt durch FLASH regelmäßig gestartet werden. Außerdem ließe sich der Inhalt der Skripte so auch dynamisch zur Laufzeit des Systems ändern. Die Einarbeitung ist relativ einfach, da die Befehlssyntax überschaubar ist.
   Nachteilig ist, dass aufwendigere Skripte schnell unübersichtlich werden und die Wartbarkeit stark beeinträchtigt wird. Ein weiterer Nachteil ist, dass sich einfache

Abläufe auf diese Weise schnell umsetzen lassen, komplexere Anweisungen, wie das Auslesen einer Konfiguration (vgl. Abschnitt ***3.3***), nur sehr aufwendig.

2. Für die Umsetzung als Automaten bieten sich Switch Case Anweisungen in einer objektorientierten Programmiersprache an. Sie bietet den Vorteil, dass Abläufe relativ schnell in Programmcode umgesetzt werden können. Die Funktionen der Programmiersprache ermöglichen es, auch komplexere Abläufe umzusetzen, wie das in Abschnitt ***3.3*** beschriebene Auslesen einer Konfiguration.
   Von Nachteil ist die statische Eigenschaft einer solchen Lösung, da alles zur Kompilierzeit festgelegt werden muss. Bei Änderungen muss also ein Eingriff in den Programmcode erfolgen. Des Weiteren geht die Übersicht schnell verloren und die Wartbarkeit wird erschwert. Anpassungen durch die Techniker sind nicht möglich.

3. Die Eigenschaften der Unübersichtlichkeit sowie der schlechten Wartbarkeit der Alternativen 1 und 2 haben bei der Recherche zum „State Pattern“ geführt. Der Einsatz des State Patterns würde gerade bei komplexeren Automatenabläufen zu mehr Übersicht führen. Für detaillierte Informationen siehe (Bishop, 2008 S. 148-157). Vorteil bei dieser Alternative ist die Möglichkeit auch komplexere Funktionen zu implementieren (vgl. Abschnitt ***3.3***).
   Nachteil dieser Alternative ist aber auch hier die statische Eigenschaft, die Änderungen zur Laufzeit des Gesamtsystems nicht ohne Weiteres möglich macht. Anpassungen durch die Techniker sind ebenfalls nicht möglich.

4. Da keine der vorherigen Lösungsansätze optimal für die Erfüllung der Anforderung ist, wird eine eigene Lösung entwickelt. Ansatz ist die Interpretation von Automaten, die in XML definiert sind. Zur Laufzeit werden die definierten XML Dateien vom System geladen und interpretiert. Komplexere Anforderungen werden fest codiert, um die Möglichkeiten einer objektorientierten Programmiersprache zu nutzen und können über Schlüsselwörter in den XML Dateien aufgerufen werden.
   So wird die Anforderung erfüllt, dass Änderungen an Verhaltensbeschreibungen zur Laufzeit angepasst und komplexere Funktionen ebenfalls umgesetzt werden können. Die Verhaltensbeschreibungen können aufgrund der standardisierten Beschreibung in XML durch die Techniker bearbeitet werden. Ein weiterer Vorteil ist die Vermeidung von Fehlern bei der Erstellung der Dateien, da sich die XML Dateien an einem fest vorgegebenen Schema orientieren.

Bewertung der Alternativen

In *Tabelle 5.1* werden die Alternativen gegenübergestellt und bewertet. Die Gegenüberstellung zeigt, dass die individuelle Lösung alle notwendigen Kriterien optimal erfüllt.

Tabelle 5.1: Alternativen - Gegenüberstellung

Alternative	Anpassungen durch Techniker	Nicht fest codiert	Komplexere Anforderungen	Übersicht-lichkeit	Erweiter-barkeit
Skripte	⇨	⇧	⇩	⇩	⇨
Switch Case	⇩	⇩	⇧	⇩	⇨
State Pattern	⇩	⇩	⇧	⇧	⇧
Individuelle Lösung	⇧	⇧	⇧	⇧	⇧

⇩ *Mangelnde Unterstützung* ⇨ *Grundlegende Unterstützung* ⇧ *Optimale Unterstützung*

Grundlagen von Automatenstrukturen und Aufbau

Ein Zustandsübergangsdiagramm ist eine grafische Darstellung von endlichen Automaten, d.h. Zuständen und deren Übergangsbedingungen, um die enthaltenen Verknüpfungen möglichst durchschaubar und eindeutig zu visualisieren. Eine Vereinheitlichung wurde durch David Harels (vgl. (Harel, 1984)) Statechart-Notation erreicht. In der objektorientierten Softwareentwicklung ist dies im Rahmen der Unified Modeling Language (UML) normiert als Zustandsdiagramm. Die Notation erlaubt die präzise Spezifikation von zustandsbasierten Systemen. Dafür führte Harel mehrere Notationselemente ein, um die Komplexität großer Systeme mittels endlicher Automaten handhabbar zu machen.

Tabelle 5.2: Harel Automaten - Notationselemente

Element	Beschreibung
Hierarchie	In Verbindung mit Unterzustandsautomaten, in denen in einem Zustand einer höheren Ebene ein weiterer vollständiger Zustandsautomat steckt. Die Unterzustandsautomaten können entweder einen eigenen Startzustand haben, oder aber können Unterzustände direkt angesprungen werden.
Komposition	Für die Darstellung von parallelen Zustandsautomaten. Hierbei sind AND und OR-Komposition möglich, die ein gleichzeitiges oder abwechselndes Schalten der Automaten vorsehen.
Inter-Level-Transitionen	Die auch einen Unterzustand in einen Zustand einer anderen Ebene überführen können und umgekehrt.
History-Konnektor	Der für einen Unterzustandsautomaten bei dessen Verlassen den zuletzt eingenommenen Zustand speichert, um beim Wiedereintritt in den Unterzustandsautomaten diesen Zustand wieder einzunehmen. Der History-Konnektor wird mit einem eingekreisten H notiert.
Condition-Konnektor	Der einen Zustandsübergang (eine Transition) abhängig von einer Bedingung in disjunkt in verschiedene Zielzustände überführt. Der Condition-Konnektor wird mit einem eingekreisten C notiert.
Temporale Logik	Diese kann in den Transitionen verwendet werden, um beispielsweise Timeouts anzugeben.
Entry-, Exit-, Throughout	Bereiche von Zuständen, welche Aktionen angeben, die beim Eintreten, Verlassen bzw. während des Aufenthalts in einem Zustand ausgeführt werden.

Vorhandene Systeme für Automatenumsetzung in XML

Das an der Universität Sherbrooke entwickelte Toolkit RobotFlow (vgl. (Dominic Létourneau, 2008)) im Projekt "Mobile Robotics and Intelligent Systems Laboratory" (LABORIUS) verwendet in XML beschriebene Automaten. Die Entscheidung für XML wurde aufgrund der verbreiteten Verwendung, der sehr guten Lesbarkeit und der einfachen Anwendung getroffen. Ein Anwender braucht nicht die Details von XML zu kennen, um einen Automaten zu definieren. Er muss nur die Verwendung der Tags und deren Hierarchie verstehen. Dies gilt ebenso für die Techniker der Remote Labore.

Der Aufbau der in RobotFlow verwendetet XML Dateien für einen Automaten ist in Anhang ***C*** aufgeführt. Eine Verwendung der von Harel definierten Notationselemente wurde nicht berücksichtigt.

Individuelles Konzept für Automatenumsetzung in XML

Für die Entwicklung der individuellen Automatenstruktur wird der Ansatz für die Umsetzung in XML aus RobotFlow mit dem Aufbau einer Variante, die die Notationselemente von Harel verwendet, kombiniert. In einer Vorlesung zum Thema Prozesslenkung wurde diese erläutert. Für nähere Informationen hierzu siehe (Kaltenhäuser, 2005). Die einzelnen Zustände werden wie in ***Abbildung 5.1*** gezeigt definiert. Für die Umsetzung wird also ein strukturierter Aufbau der Automaten und den darin enthaltenden Aktivitäten benötigt.

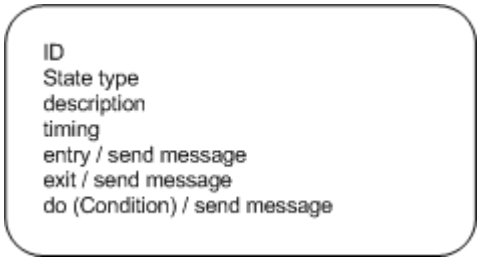

Abbildung 5.1: Automaten Zustandsdefinition

In ***Tabelle 5.3*** werden die Eigenschaften eines Automatenzustandes beschrieben. Die Felder ID, State Type und Description sind Eigenschaften, die bei jeder Automatenzustandsdefinition anwendbar sind. Die Vorteile von diesem Aufbau sind zum einen die Möglichkeit jeden einzelnen Zustand mit einem Timeout zu versehen und somit die zeitliche Gültigkeit zu begrenzen. Des Weiteren ist die Aufteilung in Eingangsbereich (Entry), Ausgangsbereich (Exit) und Hauptbereich für die Abbildung der Aktivitäten hilfreich. So lassen sich bestimmte Kommandos oder Steuerzeichen absetzen, auf dessen Antwort im Hauptbereich gewartet wird und im Anschluss verarbeitet werden kann. Am Ende kann auf eine bestimmte Antwort im Ausgangsbereich eine weitere Aktion folgen. All dies erfolgt logisch zusammenhängend in einem Zustand und muss nicht auf mehrere Zustände aufgeteilt werden.

Tabelle 5.3: Prozesslenkung Zustandseigenschaften

Eigenschaft	Beschreibung
ID	Beschreibt die eindeutige Nummer des Zustandes im Automaten
State Type	Definiert ob es sich um einen Start-, End- oder Zwischenzustand handelt
Description	Beschreibung des Zustandes in Textform
Timing	Legt die maximale Zeit, die der Zustand für sich beanspruchen darf, fest
Entry / send message	Beschreibt was beim Eintreten in den Zustand für eine Nachricht gesendet werden soll
Exit / send message	Beschreibt was beim Verlassen des Zustandes für eine Nachricht gesendet werden soll
Do (condition) / send message	Beschreibt die eigentliche Aktion, die in diesem Zustand ausgeführt werden soll. Kann optional an eine Bedingung gekoppelt werden

Da dieser Aufbau nicht allen Anforderungen der Abläufe der Netzwerkgeräte entspricht, werden beim Design der Aktivitäten Anpassungen an der Struktur vorgenommen. ***Abbildung 5.2*** zeigt die erweiterte Definition eines Automaten Zustandes.

Abbildung 5.2: Erweiterte Automaten Zustandsdefinition

Die Felder ID, Description und Timeout (vorher Timing) werden übernommen. Die Aufteilung in die drei Bereiche Eingang (entry), Hauptteil (expect) und Ausgang (exit) wird ebenfalls übernommen. Es können entweder Kommandos oder spezielle Steuerzeichen gesendet werden. Zudem wurde in jedem der drei Bereiche die Möglichkeit eingebunden, einen weiteren (Sub-) Automaten zu starten. Sich wiederholende Abläufe werden in einzelne Automaten ausgliedern und dann einfach über diese Funktionen aufgerufen. In Abhängigkeit von der erwarteten Antwort wird der entsprechende Folgezustand (nextState) aufgerufen. In ***Tabelle 5.4*** werden die einzelnen Eigenschaften nochmal beschrieben.

Die aufgerufenen Automaten werden sequenziell in den Ablauf des aufrufenden Automaten eingebunden. Parallelität zwischen Aufrufer und Aufgerufenen ist an dieser Stelle nicht vorgesehen. In Abschnitt ***5.1.2*** wird auf die Notwendigkeit von parallel auszuführenden Arbeitsabläufen eingegangen.

Tabelle 5.4: Erweiterte Zustandseigenschaften

Eigenschaft	Beschreibung
ID	Beschreibt die eindeutige Nummer des Zustandes im Automaten
Description	Beschreibung des Zustands in Textform
Timeout	Legt die maximale Zeit fest, die der Zustand maximal aktiv sein darf
entrySendCommandList	Beschreibt 1-n Konfigurationskommandos, die beim Eintreten in den Zustand an das Netzwerkgerät gesendet werden
entrySendKeylist	Beschreibt 1-n Sondersteuerzeichen, die beim Eintreten in den Zustand an das Netzwerkgerät gesendet werden
entryStartFSMList	Beschreibt 1-n Automaten, die beim Eintreten in den Zustand gestartet werden
expectString	Beschreibt den erwarteten Antwortstring vom Netzwerkgerät
expectActionSendCommandList	Beschreibt 1-n Konfigurationskommandos, die beim Zutreffen des erwarteten Antwortstrings an das Netzwerkgerät gesendet werden
expectActionSendKeyList	Beschreibt 1-n Sonderzeichen, die beim Zutreffen des erwarteten Antwortstrings an das Netzwerkgerät gesendet werden
expectActionStartFSMList	Beschreibt 1-n Automaten, die beim Zutreffen des erwarteten Antwortstrings gestartet werden
nextState	Beschreibt den Folgezustand, der beim Zutreffen des erwarteten Antwortstrings aktiviert wird
exitSendCommandList	Beschreibt 1-n Konfigurationskommandos, die beim Verlassen des Zustandes an das Netzwerkgerät gesendet werden
exitSendKeyList	Beschreibt 1-n Sondersteuerzeichen, die beim Verlassen des Zustandes an das Netzwerkgerät gesendet werden
exitStartFSMList	Beschreibt 1-n Automaten, die beim Verlassen des Zustandes gestartet werden

Grundlagen strukturierter XML Dateien

Für die Erstellung von strukturierten XML Dateien existieren zwei Definitionen. Zum einen DTD (Dokumententyp Definition) und zum anderen XML Schema. Beide Definitionen sind detailliert vom W3C (World Wide Web Consortium) beschrieben (Fallside, 2001). Da es sich bei der Definition durch das W3C um einen de facto Standard handelt und das W3C den Einsatz von XML Schema empfiehlt, wird die Umsetzung nach diesem Standard erfolgen. Im Gegensatz zur klassischen DTD Definition wird beim XML Schema die Struktur in einer eigenen XML Datei beschrieben. Dabei werden auch verschiedene Datentypen unterstützt, um eine hohe Datenintegrität, ähnlich einer Datenbank, zu gewährleisten. Daher wird beim Erzeugen der Verhaltensbeschreibungen bereits Typsicherheit erlangt. Variablentypen wie zum Beispiel xs:string und xs:integer definieren die zu erwartenden Daten, welche anschließend mit weiteren Attributen eingeschränkt werden können. Ein XML-Schema wird als XSD (XML-Schema-Definition) bezeichnet und hat in Form einer Datei die Dateiendung ".xsd".

XML Dateien - Namenskonvention

Um die verschiedenen Verhaltensbeschreibungen abzulegen, wird eine einheitliche Namenskonvention für die XML Dateien verwendet. Die Definition sieht folgendermaßen aus:

XML_DeviceType_AutomationTask_IOSVersion_Version.xml

Tabelle 5.5: XML Dateien - Namenskonvention

Feld	Beschreibung
XML	Festes Feld für Dateitypenkennzeichnung
DeviceType	Beschreibt den genauen Gerätetypen (z. B. 1841, 2960-8)
TaskType	Beschreibt die Aufgabe dieser Verhaltensbeschreibung und wird in Kombination mit dem folgenden Feld verwendet, da eine Verhaltensbeschreibung immer für eine bestimmte IOS Version definiert wird
IOSVersion	Name der IOS Version, für die die XML Datei definiert wird. Diese beinhaltet auch immer den genauen Modelltyp des Gerätes
Version	Beschreibt die Version der Datei

Tabelle 5.5 beschreibt die einzelnen Felder der Namenskonvention. Eine fertige Datei sieht wie folgt aus:

XML_1841_PWD_RECOVERY_c1841-advipservicesk9-mz.124-16.bin_Version10.xml

Der Dateiname setzt sich aus festen und variablen Bezeichnern zusammen, die mit Unterstrichen voneinander getrennt werden. *XML* ist fest vorgegeben, dann folgt der Gerätetyp (Bsp.: *1841*) und im Anschluss eine Beschreibung des Automaten (*PWD_RECOVERY*). Die darauf folgende Zeichenkette *„c1841-advipservicesk9-mz.124-16.bin“* ist der Name der verwendeten IOS Version. Um verschiedene Versionen einer Verhaltensbeschreibung zu kennzeichnen, wird noch ein Versionsbezeichner angefügt (*Version10*). Da es sich um XML Dateien handelt, ist die entsprechende Dateiendung „.xml“.

5.1.2 NF2 – NEBENLÄUFIGE AUSFÜHRUNG VON AUFGABEN

FLASH soll mehrere Aufgaben nebenläufig (parallel) bearbeiten, um in relativ großen Laborumgebungen[10] eingesetzt zu werden. Dabei soll FLASH in der Lage sein, Einzelne oder Gruppen von Geräten nebenläufig zu bearbeiten. Die relevanten Netzwerkgeräte für die Entwicklung des Systems sind physikalisch unabhängig voneinander, sodass eine Aufgabe für ein Gerät als unabhängig bezeichnet werden kann.

Lösungsansatz

FLASH wird als Multithreading fähiges System entwickelt. Grundlage für das Design ist das klassische Master/Slave Prinzip. Ein Master verteilt einzelne Aufgaben an seine Slaves und sammelt die Ergebnisse zum Endergebnis zusammen. Die Slaves kommunizieren nur mit dem Master. Über den Master werden die auszuführenden Aufgaben synchronisiert.

Basierend auf dem Master/Slave Prinzip wird ein dreistufiges Controllersystem für die Auftragsbearbeitung eingesetzt, das in *Abbildung 5.3* zu sehen ist. Die erste Stufe bildet der Hauptcontroller (Controller). Er ist dafür zuständig anstehende Aufträge entgegenzunehmen und einen Controller der zweiten Stufe (Controller Worker) für jeden Auftrag zu erzeugen und diesem eine Geräteliste des Auftrages zu übergeben. Die Schnittstelle, über den der Controller

[10] Die Fast Lane Remote Labore in Berlin, Frankfurt und Hamburg bestehen zum Zeitpunkt dieses Projektes aus ca. 2.500 Netzwerkgeräten.

der ersten Stufe die Aufträge erhält, wird in Abschnitt ***5.1.3*** erläutert. Zwischen den beiden Controllern besteht keine weitere Verbindung. Der Controller der zweiten Stufe entspricht in seiner Funktion dem des Masters und erzeugt die sogenannten Slaves, die als Worker bezeichnet werden. Zwischen erzeugendem Controller und den erzeugten Workern besteht die oben erläuterte Master/Slave Beziehung.

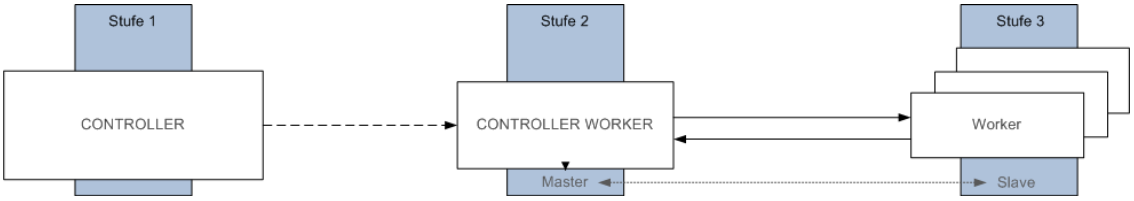

Abbildung 5.3: Dreistufiges Controllersystem

Beispiel

Ein ICND1 Labor soll zurückgesetzt werden. Dafür legt der Scheduler einen Auftrag an, der alle zu bearbeitenden Geräte beinhaltet. Der Controller kann anschließend mit der Bearbeitung des Auftrages beginnen und erzeugt einen Controller Worker, der wiederum pro Gerät einen Worker erzeugt.

Da die einzelnen Geräte unterschiedlich schnell fertig sind, kann eine erfolgreiche Bearbeitung des Auftrages erst gemeldet werden, wenn alle Geräte bearbeitet und die entsprechenden Aufträge erfolgreich beendet wurden. Diese Synchronisation wird durch das Master/Slave Beziehung zwischen Controller Worker und den Workern sichergestellt.

Erweiterter Lösungsansatz

Die im Rahmen dieses Projektes relevanten Netzwerkgeräte sind alle voneinander physikalisch unabhängig. In Abschnitt ***4.5*** wurde auf die Netzwerkgeräte vom Typ Filer eingegangen, bei denen eine physikalische Abhängigkeit besteht. Im Hinblick auf die Weiterentwicklung von FLASH soll dies berücksichtigt werden und eine Struktur vorgesehen werden, die diese Anforderung abdeckt.

Umgesetzt wird dies durch die Definition jeweils einer Strategie pro Netzwerkgeräteklasse. Dafür wurde das sogenannte Strategy Pattern (vgl. (Bishop, 2008), S.139-148) verwendet. In ***Abbildung 5.4*** ist das Prinzip verdeutlicht. Jeder Worker ist einer bestimmten Strategie zugeordnet. Die einzelnen konkreten Worker werden dafür von einem Interface („IworkerStrategy“) abgeleitet. Für die Umsetzung von FLASH im Rahmen dieses Projektes ist nur eine Implementierung in Form der Klasse „FL_RS_WORKER_STRATEGY_PARALLEL“[11] benötigt.

[11] FL_RS_WORKER_STRATEGY_PARALLEL: Das „PARALLEL“ steht hier für eine physikalisch und logisch unabhängige, parallele Bearbeitung einzelnen Aufgaben voneinander.

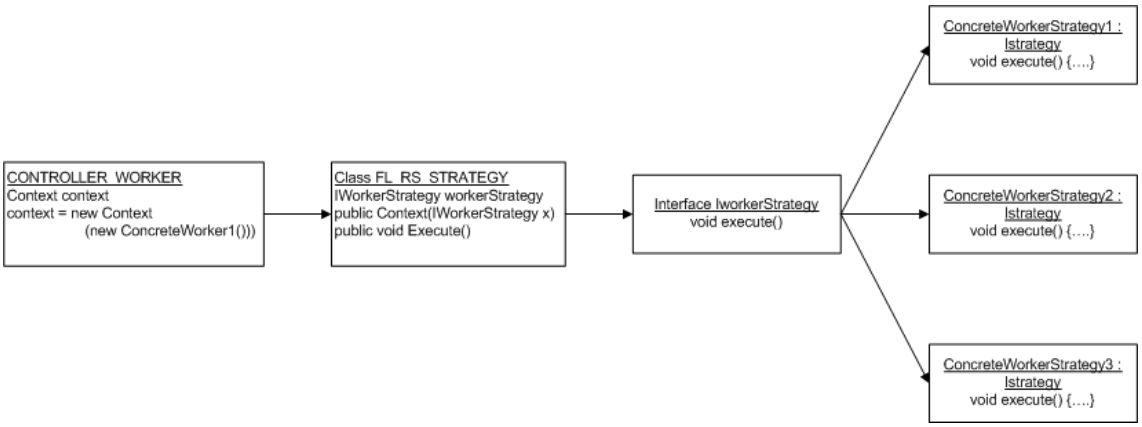

Abbildung 5.4: Strategy Pattern

Konzept für Multithreading

Der Controller der ersten Stufe wird in einer eigenen Thread Instanz ausgeführt. Diese wird direkt beim Start von FLASH ausgeführt. Für jeden Auftrag wird dann zur Laufzeit ein Controller Worker Thread erzeugt, der dann ebenfalls die notwendige Anzahl der Worker Threads erzeugt. Die einzelnen Worker Threads werden über die Controller Worker synchronisiert indem dieser auf die Beendigung der Worker Threads wartet. Wenn alle Threads beendet wurden, sorgt der Controller Worker dafür, dass das System über den aktuellen Status des Auftrages informiert wird. Die dafür notwendige Schnittstelle wird in Abschnitt ***5.1.3*** erläutert.

Besonderheit der Threaderstellung

Aufgrund der hohen Anzahl zu erwartender Threads beim Einsatz von FLASH werden die Möglichkeiten für die Umsetzung betrachtet. Zum einen können Threads individuell erzeugt werden. Der Entwickler muss sich dann um das Management der Threads kümmern. Um die Arbeit bei der Erzeugung und Zerstörung der Threads zu minimieren bietet das .NET Framework eine weitere Möglichkeit an, die sogenannte Threadpools (siehe(MSDN Library - Threading, 2008)). Statt für eine Aufgabe jeweils einen neuen Thread zeitaufwendig zu erzeugen und anschließend wieder zu zerstören, erstellt man hier eine feste Anzahl[12] von Threads. Eingehende Aufträge werden einem freien Thread zugeteilt oder in eine Warteschlange gestellt, wenn kein freier Thread zur Verfügung steht. Der Entwickler braucht sich nicht um das Management der Threads zu kümmern und nutzt einfach die Funktionen der vorhandenen Threadpool Implementierung. Auf die Threads eines Threadpools hat man nicht dieselben Kontrollmöglichkeiten, wie auf individuell erzeugte Threads. Es besteht keine Möglichkeit ein Handle auf den Thread zu bekommen, der einen neuen Auftrag entgegennimmt. Daher kann der Thread weder abgebrochen oder gestoppt werden. Die vorgesehene Synchronisierung der Worker Threads mit dem aufrufenden Controller Worker soll über die Methode Thread.Join()[13] realisiert werden, um im Sinne des Master/Slave Prinzips

[12] Threadpool: Die Anzahl der Threads in einem Threadpool ist beim .NET Framework 3.5 auf 250 Threads pro CPU begrenzt.

[13] Thread.join() Methode: Sie bewirkt, dass der aufrufende Thread solange blockiert bis der angegebene Thread beendet wird.

das Ergebnis des Gesamtauftrages zu verarbeiten. Bei einem Threadpool Thread ist dies nicht möglich, sodass auf die Verwendung der Threadpools verzichtet wird.

Ressourcenbedarf für Threadeinsatz

Beim Einsatz der Threads sind die beiden Ressourcen Arbeitsspeicher und Rechenzeit der CPU zu berücksichtigen. Die Größe des Arbeitsspeichers ist eine Begrenzung für die maximale Anzahl zu erzeugender Threads. Für jeden Thread wird ca. 1MB Arbeitsspeicher benötigt (vgl. (Duffy, 2006)). Auf dem Anwendungsserver ist für die Ausführung von FLASH bis zu 2GB RAM frei, sodass maximal ca. 2000 Threads zur gleichen Zeit ausgeführt werden können. Im Rahmen dieses Projektes wird FLASH mit maximalen 200 nebenläufigen Worker Threads ausgeführt. Die Berechnung erfolgt auf Grundlage der Gesamtanzahl der Netzwerkgeräte der ICND Labore.

Bei den verwendetet Worker Threads handelt es sich um keine rechenintensiven Threads sondern eher um I/O intensive Threads. Sie haben relativ kurze Arbeitszyklen, sodass die auf dem verwendeten Anwendungsserver zur Verfügung stehende Rechenzeit als ausreichend anzusehen ist.

5.1.3 NF4 - SCHNITTSTELLENBEREITSTELLUNG

Im Rahmen dieses Projektes soll der Scheduler (siehe Abschnitt ***3.2.1.1***) seine Aufträge an FLASH übermitteln, sodass mit der Bearbeitung begonnen wird. Über eine Schnittstelle werden alle notwendigen Informationen für die Auftragsbearbeitung bereitgestellt und die Ergebnisse der Auftragsbearbeitung über diese zurückgegeben.

Lösungsansatz

Als Schnittstelle für die Kommunikation von FLASH mit dem Scheduler wird eine Datenbanktabelle verwendet. Die Aufträge werden vom Scheduler bereitgestellt. Dieser schreibt die notwendigen Daten in eine Datenbank. Diese wird vom Hauptcontroller (Controller) in FLASH alle 30 Sekunden abgefragt (Polling). Für die Durchführung der Aufträge sind die in ***Tabelle 5.6*** erläuterten Daten notwendig.

Tabelle 5.6: Informationen für Auftragsbearbeitung

Information	Beschreibung
Auftrags ID	Eindeutige Auftragsnummer, die für das weitere Referenzieren auf und zu diesem Auftrag notwendig ist.
Geräte ID	Eindeutige Gerätenummer des Auftrags.
Auftragsstatus	Kennzeichnet den Status des Auftrages.
Geräte Typ	Modell Typ des Netzwerkgerätes. Notwendig für die Auswahl der entsprechenden XML Datei
Geräte Auftragstyp	Notwendig für die Auswahl der entsprechenden XML Datei
Geräte Terminal Server IP	Notwendig für den TCP/IP Zugriff über den entsprechenden Terminal Server auf das Netzwerkgerät
Geräte Terminal Server Line	Notwendig für den Zugriff auf die Konsole des Netzwerkgerätes über die entsprechende Line des Terminal Server
Geräte APC IP	Notwendig für den TCP/IP Zugriff über den entsprechenden APC auf den Spannungsanschluss des Netzwerkgerätes
Geräte APC Port	Notwendig für den Zugriff auf den entsprechenden Port des APCs
Geräte Konfiguration	Optionale Konfiguration, die auf das Gerät eingespielt werden soll.

Ein Auftrag besteht aus 1 – n beliebigen Geräten. Daher ist ein Gerät immer eine Teilaufgabe eines Gesamtauftrages. Auftrags ID und Geräte ID bilden zusammen den Primärschlüssel[14] der Tabelle. Die Geräte eines Auftrages besitzen die gleiche Auftragsnummer. Die Aufgaben des Automationssystems können die Zustände „waiting“, „running“, „finished“ und „error“ annehmen und werden in *Tabelle 5.7* beschrieben.

Tabelle 5.7: Status Zustände für Aufträge

Status	Beschreibung
waiting	Bereit zur Bearbeitung
running	Wird zurzeit bearbeitet
finished	Erfolgreich beendet
error	Fehler während der Ausführung

Bei neuen Aufträgen, die durch den Status „waiting“ gekennzeichnet sind, startet der Controller die Controller Worker. Bei erfolgreicher Beendigung eines Auftrages wird dies durch das Beenden aller Worker Threads signalisiert. Der Controller Worker nimmt seine Arbeit wieder auf und schreibt den Status des Auftrages direkt in die Datenbank.

Werden alle Teilaufgaben auf den Status „finished“ gesetzt, ist der Auftrag erfolgreich beendet worden. Das Setzen eines neuen Status für alle Teilaufgaben erfolgt atomar, sodass immer der Gesamtauftrag einen eindeutigen Zustand besitzt. Die Auswertung erfolgt im Anschluss durch den Scheduler und ist nicht Bestandteil dieses Projektes.

5.1.4 NF3 - ZEITKRITISCHE RÜCKSETZPROZEDUREN

Die nichtfunktionale Anforderung nF3 beruht auf der Überlegung, dass die Remote Labore in Zukunft durch den Einsatz von FLASH im Timesharing Betrieb (vgl. Abschnitt *2.6*) eingesetzt werden. Daher bleibt ein Zeitfenster von einer Stunde zwischen zwei aufeinanderfolgenden Schulungen.

Abgrenzung

Ist diese Stunde nicht einzuhalten, da das Zurücksetzen aus technischer Sicht nicht gewährleistet werden kann, so muss dies im Scheduler vermerkt sein. Damit wird verhindert, dass die entsprechenden Labore in zu kurzen Zeitabständen hintereinander eingeplant werden und nicht umsetzbare Aufträge für FLASH bereitgestellt werden.

Lösungsansatz

FLASH muss bei der Ausführung der Aufgaben sicherstellen, dass diese zum einen insgesamt nicht länger als eine Stunde dauern und zum anderen darf jeder einzelne Abschnitt einer Aufgabe nicht länger dauern als vorab festgelegt. Die Voraussetzung aus technischer Sicht, dass die Netzwerkgeräte der entsprechenden Labore innerhalb einer Stunde rekonfiguriert werden können, ist gegeben.

[14] Primärschlüssel (eng. Primary Key): Dient in einer Datenbank dazu, eine Zeile einer Tabelle eindeutig zu identifizieren. In diesem Fall werden zwei Spalten als Schlüssel festgelegt.

In Abschnitt *5.1.1* wird der Aufbau der XML Dateien detailliert erläutert. Für jede Aufgabe pro Gerätetyp gibt es einen Ablauf (Automaten), der in einer XML Dateien definiert ist. Innerhalb der Abläufe gibt es zwei Einstellungsmöglichkeiten für das Timing. Zum einen die Gesamtzeit für einen Automaten und zum anderen die Zeit, die jeder einzelne Zustand des Automaten andauern darf.

Für die Einhaltung der Zeiten werden sogenannte „Time-Out-Watchdogs“[15] (Watchdog) verwendet. Für den Gesamtautomaten und die jeweiligen Zustände wird jeweils ein Watchdog erzeugt, der intern einen Timer verwendet. Der Timer wird mit einem bestimmten Wert initialisiert und zählt diesen runter. Sobald der Wert des Timers 0 erreicht, wird der laufende Worker Thread beendet und eine entsprechende Statusmeldung vom Typ „Error“ (vgl. Abschnitt *5.1.6*) erzeugt, da eine korrekte Ausführung des Auftrages nicht mehr gewährleistet werden kann.

Das System erfordert keine kritischen Zeitanforderungen im Millisekundenbereich. Deshalb erfolgt die Implementierung über Software Timer (vgl. (MSDN Library - Timer, 2008)), die vom .NET Framework zur Verfügung gestellt werden.

5.1.5 NF7 - WARTBARKEIT

Für die Erfüllung der nichtfunktionalen Anforderung nF7 ist in erster Linie die eindeutige, vollständige und strukturierte Dokumentation des Quellcodes notwendig. Des Weiteren richtet sich der Stil des Programmcodes nach einem festgelegten Schema.

Lösungsansatz

Die Dokumentation des Quellcodes erfolgt nach Microsoft Richtlinien (Microsoft). C# bietet die Möglichkeit automatisch die Quellcodedokumentation in ein XML Format zu exportieren. In Verbindung mit einer Quellcodedokumentationssoftware, wie zum Beispiel Doxygen (vgl. (Dimitri van Heesch, 2008)), lässt sich aus diesem XML Export eine grafisch aufbereitete Dokumentation erstellen.

Die strukturierte Entwicklung des Quellcodes richtet sich ebenfalls nach Microsoft Konventionen, die in (CSharpProgG_2008) ausführlich beschrieben sind.

Sowohl die Dokumentation des Quellcodes als auch die Namensgebung für Klassen, Variablen, Konstanten und anderer Elemente erfolgt ausschließlich in englischer Sprache,

5.1.6 NF6 - BETRIEBSSICHERHEIT

Die nichtfunktionale Anforderung nF6 steht für eine garantierte Ausführung der Automationsaufgaben. FLASH soll autonom arbeiten und ohne Eingriffe durch die Techniker seine Aufgaben erledigen. Treten Fehler auf, die die korrekte Ausführung der Aufgaben beeinträchtigen, muss das System diese Fehler erkennen und versuchen zu beheben oder bei nicht Korrigierbarkeit melden.

[15] Time-Out-Watchdog: Der Thread muss sich vor Ablauf einer vorgegebenen Zeit beim Watchdog melden. Andernfalls wird der Thread abgebrochen und eine entsprechende Log Meldung erzeugt.

Fehlerquellen

Die in ***Tabelle 5.8*** beschriebenen Fehlerquellen müssen für die Betriebssicherheit von FLASH berücksichtigt werden.

Tabelle 5.8: Fehlerquellen

Fehlerquelle	Beschreibung
Ausführung von FLASH	Betrifft die Ausführung von FLASH, die durch einen Systemabsturz des Anwendungsservers oder von FLASH beeinträchtigt werden kann.
Ausnahmebehandlung während der Ausführung von FLASH	Betrifft Ausnahmen (Exceptions), die während der Programmausführung von FLASH auftreten können.
Fehler im Automatenablauf	Fehler die eine korrekte Ausführung der Automaten nicht möglich macht (z. B. Fehler im Automaten oder in den einzelnen Zuständen).
Fehler bei der Ausführung	Einzelne Ausführungsschritte können nicht korrekt ausgeführt werden (z. B. Verbindung zu einem Netzwerkgerät kann nicht hergestellt werden).

Lösungsansatz

Die Ausführung von FLASH wird auf dem Anwendungsserver durch den Failover[16]-Manager des Microsoft Clusterdienstes (vgl. (Cavale, 2004)) sichergestellt. Die Ausführung von FLASH wird dabei überwacht und bei einem Absturz automatisch neu gestartet. Die maximale Anzahl der automatischen Neustarts innerhalb von fünf Minuten wird auf 10 begrenzt. Im Anschluss wird eine Fehlermeldung vom Anwendungsserver erzeugt und im integrierten System Log als Meldung vom Typ „Error“ gespeichert. FLASH führt bei einem Neustart eine Überprüfung der Auftragsdatenbanktabelle durch, um zu prüfen, ob es nach einem Absturz neu gestartet wurde. Befindet sich dort ein Auftrag im Status „running“ (vgl. ***Tabelle 5.7***), was nur nach einem nicht geplanten Absturz der Fall ist, wird dieser wieder auf den Status „waiting“ zurückgesetzt und so erneut bearbeitet.

Die Ausnahmebehandlung erfolgt im Programmcode von FLASH. Die Codeabschnitte, die zu Ausnahmen führen können, werden in einem try-Block platziert. Der Code zur Behandlung von Ausnahmen im try-Block wird in einen catch-Block geschrieben.

Treten Fehler bei der Bearbeitung der Automatenabläufe auf, führen diese zu einem Verbleiben in einem der Zustände. Dies wird durch die in Abschnitt ***5.1.4*** erläuterten Watchdogs abgefangen und eine entsprechende Fehlermeldung erzeugt.

Fehler, die bei der Ausführung auftreten, wie zum Beispiel das Fehlschlagen eines Verbindungsaufbaus zu einem Netzwerkgerät, werden versucht zu beheben. Es wird versucht, die Ausführungsschritte nach einer kurzen Wartezeit zu wiederholen. Scheitern die Versuche ebenfalls, wird eine entsprechende Fehlermeldung erzeugt.

[16] Failover: Bezeichnet eine Technologie, die Daten und Anwendungen hochverfügbar hält. In diesem Fall wird dabei nicht das gesamte Primärsystem auf ein Back-up-System umgeschaltet sondern die Verfügbarkeit der Anwendung durch einen Neustart nach einem Absturz automatisch gewährleistet.

Lösungsansatz für Fehlermeldung

Für die Meldung von Fehlern, die bei der Ausführung von FLASH auftreten, wird ein zentrales Log Modul integriert. Die Aufgaben und Ziele sind das Protokollieren von Ereignissen und die damit verbundene Vereinfachung der Fehlersuche und Analyse. Das Log Modul wird von allen Modulen in FLASH verwendet und stellt dafür die entsprechenden Funktionen zur Verfügung. Die Log Informationen werden in einem zentralen Verzeichnis auf dem Anwendungsserver abgelegt. Dadurch wird sichergestellt, dass alle Systeme bei Bedarf Zugriff auf die Informationen haben. Das Verzeichnis auf dem Anwendungsserver ist auch vom Webserver zu erreichen, sodass bei der Weiterentwicklung von FLASH die entsprechenden Informationen vom Webserver abgerufen und auf entsprechenden Webseiten bereitgestellt werden können.

Das einheitliche Format der Log Information und Dateien unterstützt die Fehlersuche und Analyse ebenfalls.

Das Log System kennt die vier verschiedenen Log Level, die in *Tabelle 5.9* erklärt werden:

Tabelle 5.9: Log Level

Level	Bezeichnung	Beschreibung
1	Info	Bezeichnet eine allgemeine Statusinformation.
2	Debug	Bezeichnet eine Debug Information, die für die Fehlersuche und Analyse des Automationssystems notwendig ist. Diese Informationen sind für Techniker nicht relevant.
3	Warning	Bezeichnet eine Warnung. Warnungen schließen die erfolgreiche Ausführung der jeweiligen Aufgabe nicht aus, sondern kennzeichnen die Reaktion des Systems auf einen erkannten Fehler, der behoben werden konnte.
4	Error	Bezeichnet eine Fehlerinformation, die eine erfolgreiche Ausführung der jeweiligen Aufgabe ausschließt.

Die Log Informationen sind zum einen für die Techniker gedacht und zum anderen für Entwickler. Daher gibt es zwei verschiedene Log Dateien, die pro Aufgabe angelegt werden, die „Simple“ und „Advanced“ Log Dateien.

Damit die Techniker umgehend über einen Fehler vom Typ „Error“ informiert werden, sodass zeitnah reagiert werden kann, wird ein System Log verwendet. FLASH und der Webserver werden diese Informationen für die Techniker darstellen. Die Darstellung über den Webserver wird im Anschluss an diese Arbeit umgesetzt.

Log Dateien - Namenskonventionen

Die Dateinamen der erzeugten Logdateien richten sich nach der folgenden Konvention:

DATE_TIME_LOG_SESSION_TYPE_TASKID.txt

Tabelle 5.10: Log Dateinamen Format

Feld	Beschreibung
DATE	Aktuelles Datum in Dezimalform (JJJJMMDD)
TIME	Aktuelle Zeit in Dezimalform (hhmmss)
LOG_SESSION	Konstanter String
TaskID	ID des Auftrags für die Zuordnung der Log Dateien

Ein Beispiel für einen Logdateinamen sieht wie folgt aus:

20080403_184713_LOG_SESSION_ADVANCED_3.txt

Über die „TaskID“ kann auf die entsprechende Log Datei zugegriffen werden. Sowohl von FLASH als auch vom Scheduler oder dem Webserver.

Abgrenzung

Die Betrachtung beschränkt sich auf die beeinflussbaren Größen. Fehlerquellen, die durch höhere Gewalt ausgelöst werden, können immer auftreten und sind durch FLASH nicht abzufangen. Der Ausfall des Netzwerkes oder Sicherheitslücken des Infrastrukturbereichs sind nur zwei Beispiele.

5.1.7 NF8 - SYSTEMSICHERHEIT

Die Sicherheit des Gesamtsystems beruht auf der Annahme, dass kein ungewollter Zugriff im Infrastrukturbereich den Betrieb von FLASH beeinträchtigt.

Lösungsansatz

Der Infrastrukturbereich kann als sicheres System bezeichnet werden, sodass für diesen Bereich keine weiteren Maßnahmen getroffen werden. Die neu verwendeten und integrierten Systeme sowie Funktionen werden durch die in ***Tabelle 5.11*** aufgeführten Maßnahmen abgesichert.

Tabelle 5.11: Maßnahmen für die Systemsicherheit

Bereich	Beschreibung
Netzwerk SNMP	Zur Sicherheit der SNMP Zugriffe wird ein SNMP Community String auf den Terminal Servern (TS) und den APCs verwendet.
Datenbank	Die Datenbankzugriffe werden über einen Benutzernamen mit Password gesichert.

5.2 KONZEPTION DER TECHNISCHEN ANFORDERUNGEN

Die technischen Anforderungen werden aus den funktionalen Anforderungen (vgl. ***Tabelle 4.2***) abgeleitet. Im Rahmen dieses Projektes sollen die Anforderungen F1, F2, F9 sowie F12 umgesetzt werden. Zusammenfassend sind diese:

F1: System_Reset

F2: System_CFG_load

F9: Log Funktion

F12 Netzwerkgeräte (Liste)

Lösungsansatz

Die funktionalen Anforderungen F1 bis F11 werden auf technischer Ebene umgesetzt. ***Abbildung 5.5*** zeigt eine Darstellung der funktionalen Anforderungen in einem Anwendungsfalldiagramm (Use Case Diagram), um das Zusammenspiel zwischen System, Funktionen und Zugriffen auf das System zu verdeutlichen. Als Akteure können die Techniker der Remote Labore oder die Teilnehmer in den Schulungen sowie das System selbst agieren. Die Akteure lösen bestimmte Funktionen aus, wie das Zurücksetzen eines Gerätes. Das System greift (nicht als Akteur) auf die dafür notwendigen internen Funktionalitäten zurück. Die Umsetzung der einzelnen funktionalen Anforderungen erfolgt in Form der in Abschnitt ***5.1.1*** beschriebenen XML Dateien, die die einzelnen Konfigurationsabläufe definieren.

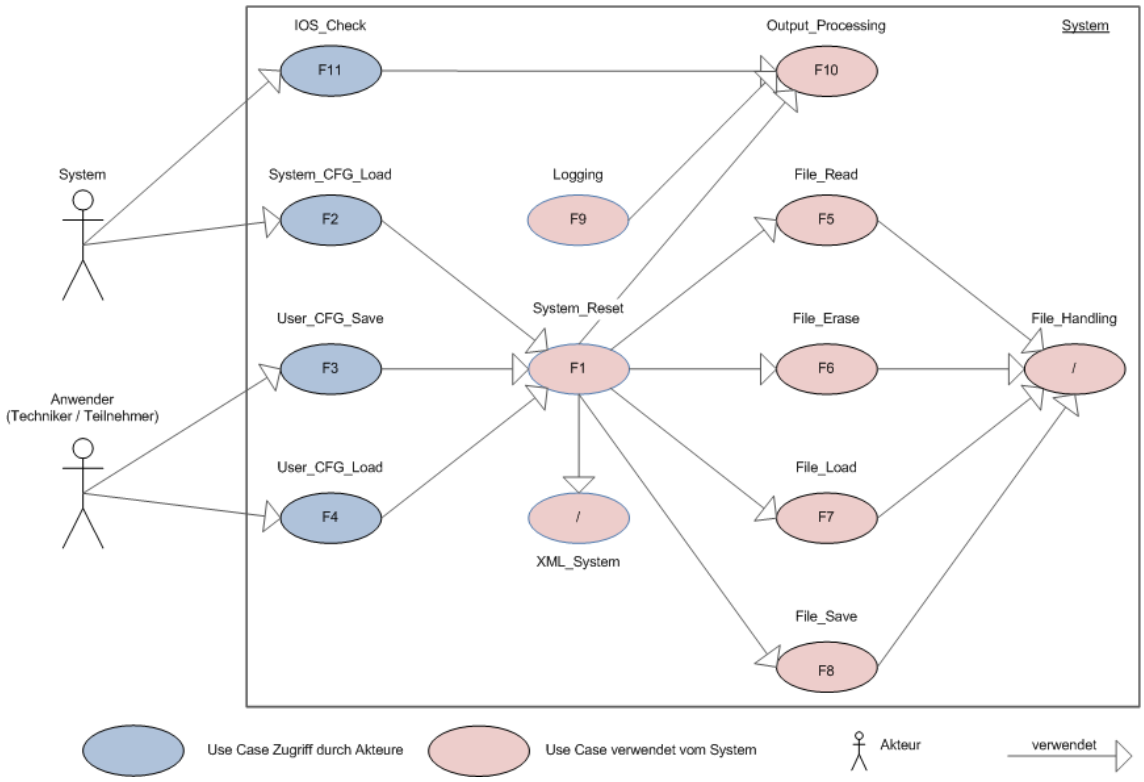

Abbildung 5.5: Use Case Diagramm

Die, für die Durchführung der Konfigurationsabläufe allgemeinen Aufgaben (vgl. Abschnitt ***3.3***), werden über statische Klassen bereitgestellt und können in den XML Dateien als Unterautomaten aufgerufen werden. Eine Liste dieser Unterautomaten ist in ***Tabelle 5.12*** zu sehen. Dafür wird der Name des Unterautomaten als Schlüsselwort in dem dafür vorgesehenen XML Feld angegeben.

Tabelle 5.12: Vordefinierte Unterautomaten

Unterautomat	Beschreibung
STATIC_FSM_POWER_OFF	Zuständig für das Ausschalten des entsprechenden APC Ports per SNMP
STATIC_FSM_CLEAR_LINE	Zuständig für das Zurücksetzen der entsprechenden Terminal Server Line per SNMP
STATIC_FSM_TELNET_CONNECTION	Zuständig für die Reverse TELNET Verbindungserstellung
STATIC_FSM_POWER_ON	Zuständig für das Einschalten des entsprechenden APC Ports per SNMP

Nach Einführung von FLASH lassen sich weitere Funktionen integrieren, indem die Abläufe spezifiziert werden und entsprechende XML Dateien erzeugt werden.

5.3 SNMP MODUL

Für die Ansteuerung der Terminal Server (TS) und der Leistungsverteiler (APCs) wird eine Steuerung über das Simple Network Management Protokoll (SNMP) benötigt. Damit sollen die Funktionen wie das Ein- und Ausschalten der APC Ports und das Line Clear auf den Terminal Servern durchgeführt werden.

Lösungsansatz

Integration eines SNMP Moduls in FLASH. Die Umsetzung richtet sich nach dem SNMPv1 Standard (siehe (Case, 1990)). Für die Entwicklung von FLASH sind die in ***Tabelle 5.13*** beschriebenen SNMP Funktionen erforderlich.

Tabelle 5.13: SNMP Funktionen

Funktion	Beschreibung
APC Power OFF	Zuständig für das Ausschalten des entsprechenden APC Ports per SNMP
APC Power ON	Zuständig für das Einschalten des entsprechenden APC Ports per SNMP
TS Clear Line	Zuständig für das Zurücksetzen der entsprechenden Terminal Server Line per SNMP

Das SNMP Modul wird so implementiert, dass es bei der Weiterentwicklung von FLASH einfach erweitert werden kann. Die Überprüfung des Status einer Terminal Server Line ist ein Beispiel dafür.

5.4 ZEIT

FLASH arbeitet nicht mit der absoluten Zeit, sondern startet die Ausführung der Aufträge spätestens nach 30 Sekunden. Dies entspricht dem Abfrageintervall der Auftragsdatenbank. Eine Berücksichtigung der Systemzeit des Schedulers oder Webservers ist daher nicht relevant.

5.5 RANDBEDINGUNGEN

Bei der Umsetzung und Implementierung ist die Einhaltung des Zeitplanes zu berücksichtigen. Daher erfolgt die Umsetzung des ersten Prototyps von FLASH während dieser Bachelorarbeit umgesetzt.

5.6 ABGRENZUNGEN

Während der Designerstellung von FLASH sind zwei weitere Merkmale sichtbar geworden, die für den Einsatz von FLASH von Bedeutung sind aber im Rahmen diese Bachelorarbeit aus zeitlicher Sicht nicht entwickelt oder berücksichtigt werden. Im Folgenden werden diese Merkmale der Vollständigkeit wegen erläutert.

Abgrenzung - Zugriffsregelung

Stellt der Scheduler einen Auftrag für FLASH bereit, muss sichergestellt werden, dass kein Zugriff auf die Netzwerkgeräte der Labore über das Webinterface erfolgen kann. Sonst würden die Teilnehmer unter Umständen die Auftragsbearbeitung unterbrechen und FLASH kann die Aufträge nicht erfolgreich bearbeiten. Die dafür notwendige Zugriffsregelung muss vom Scheduler kontrolliert werden.

Abgrenzung – Benutzer und System Konfigurationen

Beim Einsatz der Labore im Timesharing Betrieb werden die Konfigurationen der Netzwerkgeräte in zwei unterschiedliche Klassen aufgeteilt. Zum einen die Systemkonfigurationen, die im System gespeichert sind und von FLASH für die Durchführung der einzelnen Aufgaben benötigt werden. Zum anderen die Benutzerkonfigurationen, die zur Laufzeit erstellt werden. Die Benutzerkonfigurationen spielen beim Speichern einer Teilnehmer Sitzung eine entscheidende Rolle, da sie den gegenwärtigen Systemzustand beinhalten. Nur durch das Sichern der Benutzerkonfiguration kann eine vollständige Wiederherstellung der Sitzung gewährleistet werden. Die Benutzerkonfigurationen müssen dem entsprechenden Teilnehmer zugeordnet werden. Beim Design von FLASH wird darauf geachtet, dass eine zu sichernde Konfiguration als Rückgabewert dem Auftrag zugeordnet werden kann. Das endgültige Abspeichern und anschließende Bereitstellen muss vom Scheduler gewährleistet werden.

Des Weiteren besteht eine Voraussetzung für das Speichern der Benutzerkonfiguration, die zwingend erforderlich ist. Das Abspeichern der „Running-Config“ als „Startup-Config“. Dies muss durch den Teilnehmer gewährleistet werden und stellt die einzige Vorbedingung für das automatische Wiederherstellen dar. Das System kann dies nicht bewerkstelligen, da für den definierten Einstieg das Password Recovery (vgl. Abschnitt *3.4*) notwendig ist und damit der Inhalt, der „Running-Config“, verloren geht. Vor dem Ablauf der Teilnehmersitzung muss der Teilnehmer also seine Konfiguration sichern. Dies geschieht über das Kommando:

```
Device#copy running-config startup-config
```

Das direkte Sichern der „Running-Config“ ist nicht möglich, da man sich diese über ein Kommando auf der Konsole ausgeben lassen muss und die Ausgabe durch Debug Nachrichten des IOS auf den Netzwerkgeräten unterbrochen werden kann, da diese asynchron ausgegeben werden.

6 IMPLEMENTIERUNG UND REALISIERUNG

In diesem Kapitel werden die Realisierung der einzelnen Anforderungen und der aktuelle Stand der Entwicklung sowie die Weiterentwicklungsmöglichkeiten erläutert.

Die Dokumentation des Quellcodes wird in der Datei „FLASH_Documentation.xml“ auf der beigefügten CD-ROM abgelegt und in diesem Kapitel nicht weiter berücksichtigt.

6.1 VORBETRACHTUNGEN

Die Entwicklung der Systemarchitektur von FLASH erfolgt objektorientiert in der Programmiersprache C#. Als Entwicklungsumgebung wird Visual Studio 2008 (Professional Edition) auf Basis des .NET Frameworks 3.0 eingesetzt. Die Ausführung von FLASH erfolgt auf dem Anwendungsserver (APP) und die Datenbankschnittstelle wird auf den vorhandenen Datenbankserver implementiert. In Abschnitt ***3.7 Entwicklungsumgebung*** wird die Entscheidung begründet.

Im Anhang ***F*** ist ein vollständiges Klassendiagramm des Entwicklungsprojektes von FLASH abgebildet. Es zeigt die einzelnen Module und ermöglicht einen Überblick über die Aufteilung der Programmstruktur und den Aufbau der einzelnen Module.

6.2 FLASH ALS ANWENDUNG

Dieser Abschnitt dokumentiert die entwickelte grafische Oberfläche von FLASH. Des Weiteren werden die einzelnen Funktionen erläutert. ***Abbildung 6.1*** zeigt die Oberfläche von FLASH mit Auswahl des Status Tabs. Das Status Tab hat die Funktion, den Inhalt des System Logs in FLASH darzustellen. Dadurch kann auf der Administratorkonsole des Anwendungsservers auf kritische Fehler Einsicht genommen werden. Es werden ausschließlich die Log Meldungen vom Typ „Error“ ausgegeben, sodass nur Meldungen zu Aufträgen erscheinen, die fehlgeschlagen sind.

FLASH wird auf dem Anwendungsserver ausgeführt und dem Failover Manager zugeordnet, der für den automatischen Neustart im Fehlerfall zuständig ist (vgl. Abschnitt ***5.1.6***).

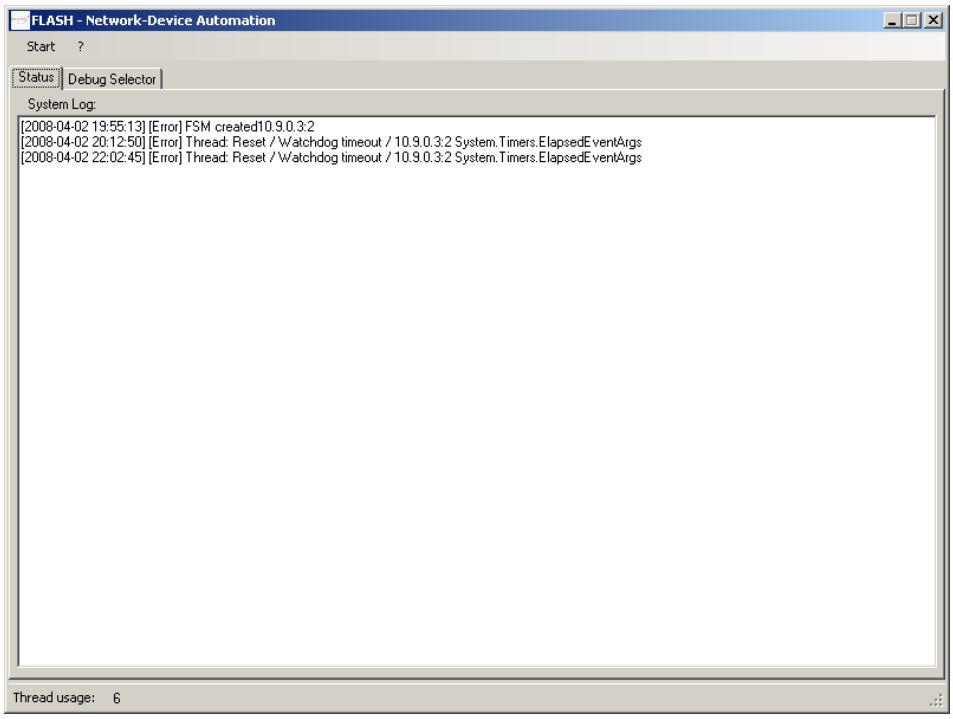

Abbildung 6.1: FLASH - Status Tab

In ***Abbildung 6.2*** ist das Debug Selector Tab zu sehen. Die Funktionen von FLASH können hierüber getestet werden, ohne auf den Scheduler zurückzugreifen, um Aufträge für Testzwecke zu erstellen. Soll eine neue Funktion getestet werden, wählt man im linken Bereich das entsprechende Gerät aus und fügt es über den „Add“ Button der Auftragsanfrageliste zu. Dort wählt man die zu überprüfende Funktion für das Gerät aus und startet die Ausführung über den „Start“ Button.

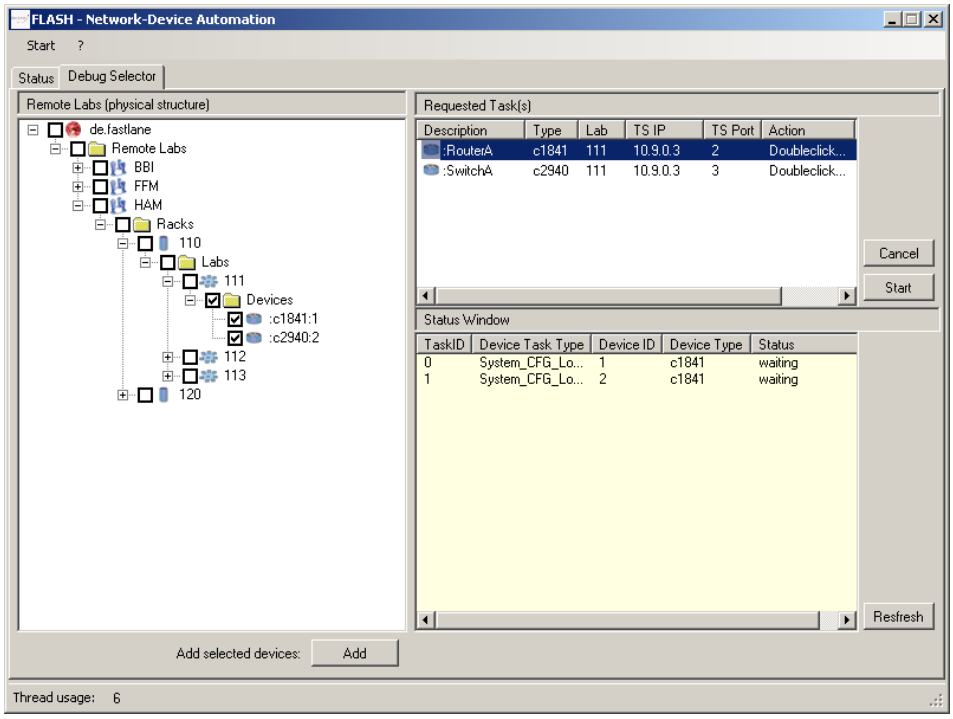

Abbildung 6.2: FLASH - Debug Selector Tab

6.3 DESIGNKRITERIEN

In diesem Abschnitt werden die Implementierung und die Realisierung der jeweiligen Designüberlegungen aus Abschnitt ***5.1*** erläutert.

6.3.1 XML KONZEPT

Während der Durchführung der Konfigurationsprozeduren der Netzwerkgeräte werden die entsprechenden Ablaufdiagramme erstellt. Aus diesen wird dann anschließend eine XML Datei erzeugt und der entsprechende Dateiname über das Datenbanksystem FLASH zur Verfügung gestellt. Die im Rahmen dieser Bachelorarbeit erstellte Diagramm und dazugehörigen XML Dateien sind auf der beiliegenden CD-ROM hinterlegt.

XML Schema Datei - Aufbau

In diesem Abschnitt wird der Aufbau der XML Schema Datei beschrieben. Der Schema Datei Aufbau befindet sich im Anhang ***A FLASH - XML Schema Datei***. Die Erstellung und Wartung der XML Dateien soll so einfach wie möglich sein und keine Abhängigkeiten zwischen den einzelnen Abläufen bestehen, so dass jede Automatenbeschreibung in einer einzelnen XML Datei abgelegt wird. Der Aufbau der XML Konfigurationsdateien gestaltet sich wie folgt. Am Anfang sind Informationen zum Gesamtautomaten enthalten. Zu diesen Informationen zählt die maximale Zeit, die der Automat hat, um vom Startzustand zu einem Endzustand durchzulaufen. Weiterhin gibt es ein Feld für die Angabe des Startzustandes des Automaten. Der Startzustand wird eindeutig anhand der ID 1 festgelegt. Es folgen die einzelnen Zustände des Automaten, von denen es mindestens einen und beliebig viele geben kann. Das Schema beschreibt den in ***Abbildung 5.2*** definierten Zustand. Jeder Zustand bekommt eine ID, eine Beschreibung und optional einen Timeout Wert. Im Entrybereich kann keine oder beliebig viele Kommandos, Steuerzeichen oder weitere Automaten angegeben werden. Der Hauptteil kann einmal oder beliebig oft verwendet werden. Dies ist notwendig um Verzweigungen abzubilden. Für jede Möglichkeit setzt sich aus einem erwarteten Antwortstring und deren Reaktion darauf zusammen. Die Reaktion kann sich aus keinen oder beliebig vielen Kommandos, Steuerzeichen oder Automaten zusammensetzen und wird mit dem Aufruf eines Folgenzustandes beendet. Im Exitbereich kann keine oder beliebig viele Kommandos, Steuerzeichen oder weitere Automaten angegeben werden.

Die Erstellung der Dateien erfolgt über einen XML Editor. Anhand der Definitionen in der Schemadatei werden so korrekte Beschreibungen erzeugt, da die Eingaben gegen das Schema geprüft werden. In einer späteren Phase wird dann ein grafischer Editor für die Erstellung erzeugt. Dort werden dann Definitionen für die Steuerzeichen und bereits vorhandene Automaten vorgegeben. So wird eine Fehlerquelle bei der Erzeugung der XML Dateien vermieden, die aktuell durchaus gegeben ist.

XML Dateien - Verarbeitung

Die Bearbeitung und Auswertung der XML Dateien bilden den Abschluss dieser Lösungsalternative. Eine vorhandene XML Datei wird zur Laufzeit über eine Funktion (`createFSM()`) eingelesen, die die Parser (siehe (MSDN Library - XmlTextReader, 2008)) des .NET Framework verwendet. Dabei werden zwei verschiedene Objekttypen erzeugt. Zum einen ein Automaten Objekt vom Typ „`FL_FSM`“ und

für jeden Zustand des Automaten ein Objekt vom Typ „`FL_FSM_STATE`“. Den Eigenschaften der Objekte werden die entsprechenden Werte aus den XML Dateien zugewiesen.

Ausführung der Konfigurationsaufgaben

Die eigentliche Ausführung der beschriebenen Abläufe startet, nachdem die XML Dateien eingelesen wurden. Dafür wird die Funktion `runFSMEngine()` aufgerufen, die die Verarbeitung der Automatenstruktur vornimmt. Als Erstes wird ein Watchdog für den Gesamtautomaten erzeugt und im Anschluss startet die Bearbeitung mit dem Startzustand. In Abhängigkeit der Parameter werden nun die entsprechenden Zustände des Automaten abgearbeitet.

6.3.2 NEBENLÄUFIGE AUSFÜHRUNG

Basierend auf dem Master/Slave Prinzip wird das Controllersystem implementiert. Das umgesetzte Controller Konzept ist in ***Abbildung 6.3*** dargestellt. Kern dieser Implementierung ist der Hauptkontroller (Controller) (1), er ist für die Annahme der Aufträge (Tasks) vom Scheduler über die in Abschnitt ***5.1.3*** besprochene Datenbankschnittstelle (2) zuständig. Pro Task erzeugt der Kontroller einen Controller Worker (5) und übergibt diesem alle zum Task (3) gehörenden Geräte (Devices) (4). Im Anschluss setzt der Controller den Task bzw. die Devices auf den Status „running“. Der Controller Worker erzeugt seinerseits pro Device einen Worker[17] (6), der dann die eigentliche Bearbeitung erledigt. Der Controller wartet über die Funktion `Thread.join()` auf das Ende der Worker Threads. Während der Bearbeitung schreiben die Worker ihre Statusmeldungen in die entsprechenden Log Dateien.

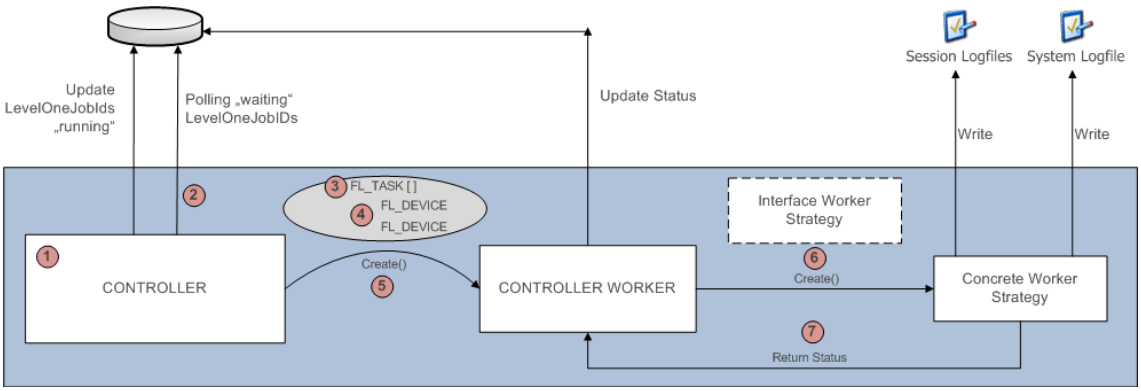

Abbildung 6.3: Controller Konzept

Beim Starten von FLASH wird geprüft, ob noch Aufträge mit dem Status „running“ in der Auftragstabelle vorhanden sind. Diese deuten auf einen Absturz von FLASH hin und konnten nicht beendet werden. Daher werden diese Aufträge auf den Status „waiting“ gesetzt und erneut bearbeitet.

[17] Worker: Im Rahmen dieser Bachelorarbeit können nur Worker der Klasse „FL_RS_WORKER_STRATEGY_PARALLEL“ erzeugt werden.

6.3.3 ZEITKRITISCHE RÜCKSETZPROZEDUREN

Um sicherzustellen, dass eine Aktivität nicht länger dauert, als eigentlich notwendig, wird das Watchdog Modul verwendet. Dies wird als „`FL_WATCHDOG`“ Klasse repräsentiert. Intern wird dann die „`System.Timers.Timer`“ Timer Implementation des .NET Frameworks verwendet.

In den Worker Threads wird beim Starten der Automatenausführung und der einzelnen Zustandsbearbeitungen aus dieser Klasse ein Watchdog Objekt erzeugt. Dabei wird der Initialisierungswert und ein Handle auf den erzeugenden Worker Thread übergeben. Läuft ein Watchdog ab, wird eine Statusmeldung vom Typ „Error“ erzeugt, da an dieser Stelle eine korrekte Durchführung nicht mehr gewährleistet ist. Für die Festlegung der maximalen Zeitwerte des gesamten Automaten gelten die in ***Tabelle 3.6*** auszugsweise beschriebenen Werte. Die Werte für die einzelnen Zustände werden bei der Erstellung der Zustandsübergangsdiagramme ermittelt und festgelegt.

6.3.4 SCHNITTSTELLE

Der Aufbau der Datenbanktabelle ist in ***Abbildung 6.4*** zu sehen und zeigt die Umsetzung der in Abschnitt ***5.1.3*** erläuterten Designüberlegungen. Während der Realisierung sind noch weitere Felder hinzugekommen, die im Folgenden erläutert werden. Das vollständige Datenbankdiagramm befindet sich im Anhang in ***Abbildung G.2***.

Die „ID“ steht für die eindeutige Identifizierung des Eintrages. Die Felder „NodeID“, „LeftID“ und „RightID“ sind aufgrund der Suche nach einer einfachen Gruppierungsmöglichkeit der Aufträge und zugehörigen Geräte entstanden. Mit Hilfe von sogenannten verschachtelten Mengen (Nested Sets), die sich sehr gut in eine Datenbanktabelle umsetzen lassen und zudem sehr komfortabel über SQL Statements abfragen lassen. Für die im Rahmen dieses Projektes entwickelte Version von FLASH haben diese noch keine Funktion.

Das Feld „TaskStatusID“ kennzeichnet den individuellen Status eines Auftrages. Das folgende Feld „DeviceTaskTypeID“ steht für die Art des Auftrages für das jeweilige Netzwerkgerät. Die „DeviceID“ steht für die eindeutige Identifizierung eines Gerätes in einem Auftrag und das Feld „DeviceType“ steht für das Netzwerkgerätemodell. Das Feld „DeviceTaskStatusID“ kennzeichnet den individuellen Status eines Netzwerkgerätes und ist unabhängig vom Status des Gesamtauftrages. Der Status des Gesamtauftrages setzt sich aus den Statuswerten der Geräte zusammen.

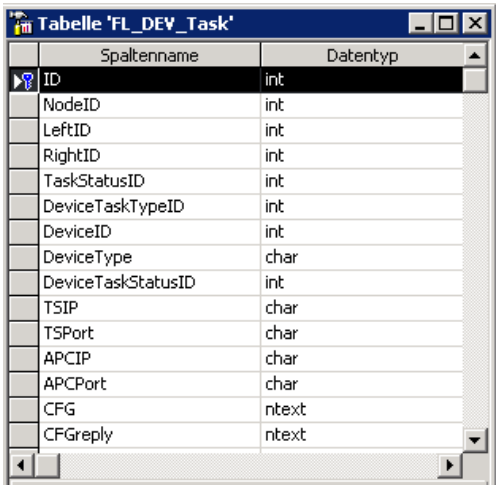

Abbildung 6.4: FLASH - Datenbankschnittstelle

Anhand der Felder „DeviceTaskTypeID“ und „DeviceType“ wird die entsprechende XML Datei von FLASH geladen. Im folgenden Beispiel wird für einen 1841 Router das Password Recovery für die IOS Version c1841-advipservicesk9-mz.124-16.bin in der Version 1.0 aufgerufen.

XML_1841_PWD_RECOVERY_c1841-advipservicesk9-mz.124-16.bin_Version10.xml

Die aktuelle Version der Datenbankschnittstelle lässt sich die Verbindungsdaten der Terminal Server und der APCs als Werte im String Format übergeben. Bei der Weiterentwicklung von FLASH wird dies auf entsprechende IDs umgestellt, sodass über diese auf die Werte der vorhandenen Systeme zugegriffen wird.

Für das Einspielen einer Konfiguration auf die Netzwerkgeräte ist das Feld „CFG“ vorgesehen. Im Design wurde nicht berücksichtigt, dass auch ein Feld für das Speichern einer Konfiguration notwendig ist. Daher wurde das Feld „CFGReply“ hinzugefügt.

Datenbankzugriff

Moderne Softwarearchitekturen bevorzugen beim Datenbankzugriff das verbindungslose Arbeiten, wobei die relevanten Daten vom Datenbank Management System (DBMS) bezogen, in einem lokalen DataSet-Objekt gespeichert, bearbeitet und ggf. anschließend wieder zur Datenbank zurück übertragen werden. Die Datenmengen für dieses System sind im Verhältnis zu der Leistungsfähigkeit des DBMS sehr klein, deshalb kann hier das verbindungslose Arbeiten eingesetzt werden. Beim Durchsuchen sehr großer Datenbestände wäre es weniger sinnvoll, mit einem lokalen Cache zu arbeiten. Für solche Anwendungen bietet das .NET Framework eine Implementierung der Schnittstelle IDataReader (z.B. SqlDataReader) an. Diese Klassen erlauben ein lesendes, sequentielles Durchlaufen der angeschlossenen Datenbank. Im Unterschied zum DataSet-Einsatz würde dann aber eine permanente Verbindung zur Datenbank bestehen. Dies ist hier nicht notwendig.

6.3.5 WARTBARKEIT

Im Programmcode wird über das Tag „///“ ein neuer Kommentar oder ein Dokumentationsabschnitt hinzugefügt. Die Inhalte der Tags werden dann ausgelesen und entsprechend in die XML Datei geschrieben. Alle Klassen, Interfaces und Funktionen sind ausführlich dokumentiert.[18] Ein Beispiel für eine Funktionsdokumentation sieht wie folgt aus:

```
/// <summary>
/// This method returns the Task type in string format
/// for the given Task Type ID
/// </summary>
/// <param name="taskTypeID"> The proper ID of the Task Type </param>
/// <returns> Returns the proper Task Type of the Device </returns>
public string getDeviceTaskType(int taskTypeID)
        {
        …
        }
```

Ein Teil der Quellcodekonventionen beinhaltet die Auslagerung von global gültigen Definitionen und Konstanten in eine separate Hilfsklasse. Dafür ist die Quellcodedatei „FL_DEFINITIONS“ zuständig. Dort sind alle Definitionen und Konstanten nach funktionalen Modulen sortiert abgelegt.

6.3.6 BETRIEBSSICHERHEIT

Das Log Modul wird durch die Klasse „FL_LOG“ repräsentiert. Das Format eines Log Eintrages ist folgendermaßen aufgebaut:

[JJJJ-MM-DD hh:mm:ss] [Log Level] Log Description

In ***Tabelle 6.1*** werden die Felder im Einzelnen beschrieben.

Tabelle 6.1: Format der Log Einträge

Feld	Beschreibung
JJJJ	Das aktuelle Jahr in Dezimalform
MM	Der aktuelle Monat in Dezimalform
DD	Der aktuelle Tag in Dezimalform
hh	Die aktuelle Stunde in Dezimalform
mm	Die aktuelle Minute in Dezimalform
ss	Die aktuelle Sekunde in Dezimalform
Log Level	Der Log Level der Status Meldung (Info, Debug, Warning, Error)
Log Description	Kurze Statusmeldung in Textform

Die „Simple“ Log Datei ist die einfachere Variante und in erster Linie für die Techniker gedacht. Sie enthält kurze und eindeutige Log Level Einträge. Die folgenden Zeilen zeigen einen Ausschnitt einer Logdatei vom Typ „Simple“.

[18] Die finale Quellcodedokumentation befindet sich im Verzeichnis „FLASH\doc“ auf der beigefügten CD-ROM.

```
[2008-04-02 19:56:02] Start logging
================================================
[2008-04-02 19:56:02] [Info] open simple log
[2008-04-02 19:56:02] [Info] FSM created
[2008-04-02 19:56:50] [Info] FSM engine executed
…
================================================
[2008-04-02 19:58:37] Finished logging!
```

Die “Advanced” Log Datei enthält detailliertere Informationen. Zum einen ist das für jeden Teilschritt einer Aufgabe eine entsprechende Statusmeldung und zum anderen wird die komplette Kommunikation mit den Netzwerkgeräten aufgezeichnet. So kann auf sehr einfache Art und Weise bei auftretenden Fehlern nach deren Ursache geschaut werden. Im Folgenden wird ein kurzer Auszug einer „Advanced“ Log Datei gezeigt.

```
[2008-04-03 18:47:13] Start logging
================================================
[2008-04-03 18:47:13] [Info] open simple log
[2008-04-03 18:47:13] [Info] open advanced log
[2008-04-03 18:47:13] [Info] FSM created
[2008-04-03 18:48:33] [Info] FSM engine executed
[2008-04-03 18:48:36] [Info] FSMWatchdog started
StateID:0 / StateDescription:Power OFF started
…
StateID:14 / Endstate / StateDescription:Close Connection
[2008-04-03 18:50:17] [Info] FSM engine executed
[2008-04-03 18:50:19] [Info] FSMWatchdog stopped

System Bootstrap, Version 12.4(13r)T, RELEASE SOFTWARE
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 2006 by cisco Systems, Inc.
…
================================================
[2008-04-02 19:56:50] Finished logging!
```

Zu sehen sind zum einen mehrere Statusmeldungen vom Automationssystem und am Ende auch Teile der Antwortnachrichten von einem Netzwerkgerät. Die Antwortnachrichten werden in Textform ohne Zeitstempel an die Systemnachrichten angehängt. Auf der beigefügten CD-ROM befindet sich jeweils eine „Simple“ und eine „Advanced“ Log Datei, die von FLASH generiert werden.

Das System Log wird fortlaufend mit Log Meldungen vom Typ „Error“ in dem festgelegten Format beschrieben.

6.4 TECHNISCHE ANFORDERUNGEN

Die technischen Anforderungen wurden alle in entsprechenden XML Dateien abgebildet. Aufgrund des Fehlverhaltens der Switches (vgl. Abschnitt ***3.2.2.3***) konnte hierfür keine Umsetzung erfolgen.

Ein Auszug der Umsetzung des System_Resets (funktionale Anforderung F1) für einen Router vom Typ c1841 ist auszugsweise im Anhang ***B*** beigefügt. Die im Rahmen dieser Bachelorarbeit erstellten Verhaltensbeschreibungen befinden sich auf der beigefügten CD-ROM.

6.5 SNMP MODUL

Das SNMP Modul setzt sich aus den drei Klassen „`FL_SNMP`“ (Basisklasse), „`FL_SNMP_TS`“ (Terminal Server) und „`FL_SNMP_APC`“ (APC) zusammen. In der Basisklasse ist die Generierung der SNMP Request Pakete gekapselt. Die Auswertung der SNMP Response Pakete ist abhängig von der entsprechenden MIB Definition und erfolgt deshalb individuell in den abgeleiteten Klassen.

Die Implementierung eines eigenen SNMP Modul ermöglicht eine detaillierte Fehlerbehandlung der SNMP Funktionen, durch die Auswertung der Response Pakete. In Anhang ***E*** ist die Auswertung eines Response Paketes detailliert erläutert.

7 ERGEBNISSE DER ENTWICKLUNG

In diesem Kapitel wird abschließend die Funktionsfähigkeit von FLASH anhand des ersten Prototyps getestet.

7.1 TESTZIELE

Ziel der Tests ist der Nachweis, dass die an FLASH gestellten Anforderungen aus der Anforderungsanalyse in Abschnitt *4.4* und der Machbarkeitsanalyse in Abschnitt *4.7* erfolgreich umgesetzt wurden. Die Durchführung der Test erfolgt in einer speziellen Testumgebung, die alle relevanten Eigenschaften des ICND1 Remote Labors widerspiegelt. Ein Test auf einem der aktiven Labore konnte aufgrund nicht zur Verfügung stehender Ressourcen zum Testzeitpunkt nicht durchgeführt werden.

7.2 TESTUMGEBUNG

Die in *Abbildung 7.1* dargestellte Testumgebung bildet die realen Bedingungen der Remote Labore nach. Im Labor Bereich befinden sich ein Router und ein Switch. Beide Netzwerkgeräte sind an einen Terminal Server und APC angeschlossen. Der Anwendungsserver, auf dem FLASH ausgeführt wird, kann auf den Terminal Server und den APC per TCP/IP zugreifen. Die detaillierten Informationen zu den verwendeten Softwareversionen sind in Anhang *H* aufgeführt.

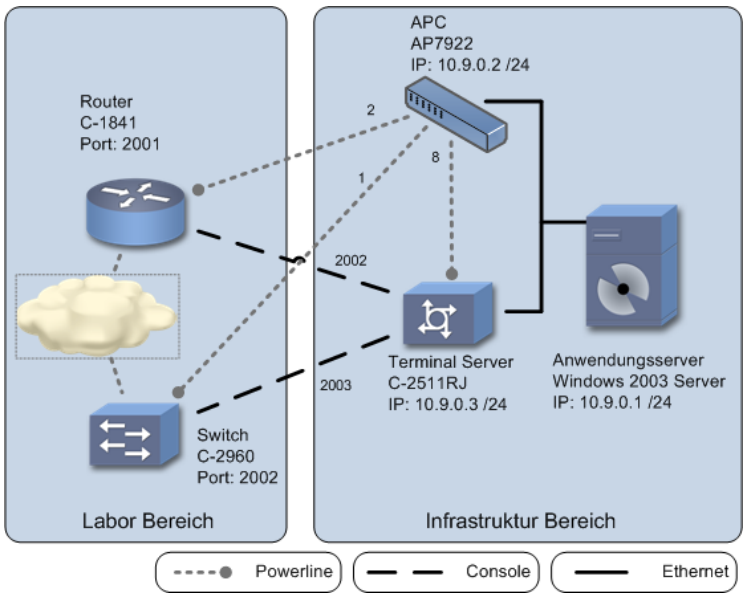

Abbildung 7.1: Testaufbau

Als Datenbank wurde eine lokale Instanz des Microsoft SQL Servers auf dem Anwendungsserver verwendet. Die Aufträge wurden nicht vom Scheduler sondern über die grafische Oberfläche von FLASH erzeugt. Des Weiteren wird der Microsoft Cluster Dienst nicht auf dem Anwendungsserver der Testumgebung ausgeführt, da die notwendigen Hardware Voraussetzungen nicht gegeben sind.

7.3 TESTSZENARIEN

Für die Testdurchführung wurden mehrere Aufträge durch FLASH erstellt. Der Controller hat dann auf über die Datenbankschnittstelle die Aufträge bearbeitet. Obwohl die Auftragserstellung nicht über den Scheduler erfolgt ist, konnte die Funktion der Schnittstellen erfolgreich getestet werden. Im Anschluss wurde FLASH umgehend abgebrochen, um die Wiederaufnahme bereits laufender Aufträge erfolgreich zu testen. Die Integration von FLASH in den Failover Manager des Microsoft Cluster Dienstes wurde in diesem Test nicht durchgeführt sondern erst zu einem späteren Zeitpunkt auf dem produktiven Anwendungsserver. Das automatische Neustarten funktionierte ohne Probleme und eine entsprechende Fehlermeldung auf dem Server wurde erzeugt.

Test SNMP Modul

Im nächsten Schritt wurde die Ansteuerung der Terminal Server und APCs über das SNMP Modul getestet. Die relevanten Funktionen wie Power OFF, Power ON und Clear Line wurden erfolgreich geprüft.

Beim Test der APC SNMP Unterstützung und dem Versuch der gleichzeitigen Überprüfung des Test über das Webinterface des APCs ist ein Problem aufgetreten. Die gleichzeitigen Zugriffe schließen sich gegenseitig aus. In den Dokumentationen zu den APCs konnten darüber keine Hinweise gefunden werden, sodass als Lösung für den Einsatz von FLASH der Zugriff auf das Webinterface der APCs über eine Benutzerkennung gesperrt wird. Das ein- und ausschalten einzelner Geräte kann deshalb nur noch über FLASH und nicht mehr direkt über die APCs erfolgen, da die fehlerfreie Ausführung der Aufträge nicht gewährleistet werden kann.

Test Watchdog Modul

Für den Test des Watchdog Moduls wurde zum einen die Einhaltung der Zeitvorgaben für die einzelnen Aufträge überprüft und zum anderen die Funktionsfähigkeit der Watchdogs auf Automaten und Zustandsebene. Die in *Tabelle 3.6* vorgegebenen Werte wurden eingehalten, somit ist die Anforderung, dass alle Aufgaben innerhalb einer Stunde bearbeitet werden müssen, für dieses Testszenario erfüllt.

Für den Watchdog Test auf Automatenebene wurden eine XML Definition ohne Endzustand gestartet, die entsprechend dem angegebenen Zeitwert abgebrochen wurde und als „Error“ im Log gespeichert wurde.

Test FSM Modul

Genaue XML Datei angeben für Durchführung und die entsprechenden CFGs

Test der maximalen Thread Gesamtzahl muss noch erbracht werden, da die Ressourcen nicht zur Verfügung standen, konnte dieser bis zum Abschluss des Projektes nicht in der Realumgebung durchgeführt werden. Trotzdem Test des Master / Slave Prinzip mit Dummy Threads, die jeder für sich eine bestimmte Teilberechnung durchgeführt haben.

Test der Betriebssicherheit durch simulierte Fehler

Die Betriebssicherheit wurde anhand der folgenden simulierten Fehler getestet und die Ergebnisse durch „Error“ Meldungen im Log dokumentiert.

1. Exception im Programmcode kontrolliert geworfen
2. Fehlerzustand in einer XML Definition
3. Verbindungsabbruch durch die Deaktivierung der TCP/IP Verbindung

Während der durchgeführten Test wurden die jeweiligen Log Dateien erzeugt und dokumentierten eine einwandfreie Funktion des Log Moduls. Auf der beigefügten CD-ROM sind je eine „Simple“ und eine „Advanced“ Log Datei abgelegt.

7.4 ERGEBNISSE

In *Tabelle 7.1* sind die Anforderungen der Prioritätsstufe 1, die im Rahmen dieser Bachelorarbeit umgesetzt wurden, aufgeführt. Den Testergebnissen entsprechend sind diese bewertet. Die nichtfunktionalen Anforderungen wurden alle umgesetzt. Bei den funktionalen Anforderungen fehlt die Integration der Switches aus der geforderten Netzwerkgeräteliste in das System. Die Ursache wurde in Abschnitt *3.2.2.3* erläutert und konnte im Rahmen dieser Bachelorarbeit nicht gelöst werden.

Tabelle 7.1: Ergebniskontrolle - Erledigungsmatrix

Index	Anforderung	Status
F1	System_Reset	⇧
F2	System_CFG_Load	⇧
F9	Log Modul	⇧
F12	Netzwerkgeräte(Liste)	⇨
nF1	Anpassungen teilweise durch Techniker möglich	⇧
nF2	Nebenläufige Ausführung von Aufgaben	⇧
nF3	Zeitkritische Rücksetzprozeduren	⇧
nF4	Schnittstellenbereitstellung	⇧
nF6	Betriebssicherheit	⇧
nF7	Wartbarkeit	⇧
nF8	System Sicherheit	⇧
RB1	Anfang Q2/08 soll das Prototyp System im Betrieb sein	⇧

⇩ *Nicht erfüllt* ⇨ *Teilweise erfüllt* ⇧ *Voll erfüllt*

8 ZUSAMMENFASSUNG

Im Rahmen dieser Bachelorarbeit wurde das *F*ast *L*ane *A*utomation *S*ystem for Networking *H*ardware kurz FLASH erstellt. Es automatisiert zuverlässig die Wiederherstellungs- und Rekonfigurationsaufgaben für Netzwerkgeräte über den seriellen Konsolenanschluss und wird in der Remote Labor Umgebung des Unternehmen Fast Lane eingesetzt.

Einen Einblick über den Einsatz der Remote Labore und die notwendigen Wartungsaufgaben wurde in Kapitel ***2*** gegeben. Der hohe zeitliche Aufwand und die Fehleranfälligkeit des manuellen Arbeitens haben zu der Entscheidung für die Entwicklung von FLASH geführt.

Die Systemumgebung, in der FLASH eingesetzt wird, wurde ebenso wie die technischen Voraussetzungen in Kapitel ***3*** detailliert erläutert.

In Kapitel ***4*** wurden die Anforderungen der Firma Fast Lane sowie die Ergebnisse der im Rahmen dieser Bachelorarbeit durchgeführten Dozentenbefragung zusammengefasst. Die Anforderungsanalyse hat die Ergebnisse zusammengeführt und wurde mit einer Machbarkeitsanalyse abgeschlossen.

Darauf aufbauend wurde in Kapitel ***5*** ein Design für die Entwicklung von FLASH entworfen. In Abschnitt ***5.1.1*** wurde die grundlegende Herausforderung für das Design analysiert. Die Erstellung und Wartung der Konfigurationsbeschreibungen für die Netzwerkgeräte ist von den Technikern der Remote Labore durchführbar. Durch die Transformation der Beschreibungen in Automaten und deren Abbildung in standardisierten XML Dateien, wurde diese Anforderung erfüllt. FLASH ermöglicht die Anpassung der Beschreibungen zur Laufzeit, sodass das System im 24x7 Betrieb eingesetzt wird.

Die Entwicklung eines dreistufigen Controllersystems auf Basis der Master / Slave Prinzips in Abschnitt ***5.1.2*** hat die Grundlage für die multithreading Fähigkeit von FLASH geschaffen, sodass FLASH eine Vielzahl von Konfigurationsaufgaben nebenläufig bearbeitet. Der Einsatz des Strategy Patterns ermöglicht eine komfortable Erweiterung der Funktionalitäten von FLASH.

Der Zugriff auf FLASH erfolgt über die in Abschnitt ***5.1.3*** entworfene Schnittstelle auf Basis eines relationalen Datenbankmanagementsystems. Der bereits vorhandene Scheduler stellt die Konfigurationsaufträge über diese Schnittstelle für FLASH bereit.

Aufgrund des geplanten Einsatzes der Remote Labore im Mehrschichtbetrieb stellt FLASH sicher, dass die Aufträge innerhalb einer Stunde abgeschlossen worden sind. Dafür wurde in Abschnitt ***5.1.4*** ein Watchdog Modul entworfen, um die Einhaltung der zeitlichen Vorgaben zu überwachen.

Für die Wartung und Weiterentwicklung des Systems wurde die Entwicklung des Programmcodes sowie dessen Dokumentation an feste Richtlinien gebunden, die in Abschnitt *5.1.5* beschrieben sind.

In Abschnitt *5.1.6* bestand die Aufgabe, die Betriebssicherheit von FLASH zu garantieren. Dafür wurde der Failover Manager des Microsoft Cluster Dienstes verwendet, um Programm- und Systemabstürze zu überwachen und den automatischen Neustart von FLASH zu gewährleisten. Des Weiteren wurde ein Log System entworfen, dass Status Meldungen im Textformat für die Techniker und Systeme der Remote Labor Umgebung bereitstellt. Bei nicht korrigierbaren Fehlern, die eine Ausführung der Konfigurationsabläufe unterbinden, wird umgehend eine Log Nachricht erzeugt, sodass ein Eingreifen der Techniker ermöglicht wird.

Beim Design wurde zudem auf die Systemsicherheit der Remote Labor Umgebung geachtet, sodass FLASH kein Sicherheitsrisiko darstellt. Daher wurden in Abschnitt *5.1.7* Maßnahmen zum Schutz der Datenbankschnittstelle als auch des notwendigen Konfigurationsprotokolls für die Durchführung der Konfigurationsaufgaben getroffen.

Das Zusammenspiel der technischen Anforderungen sowie das Konzept für die Umsetzung der Konfigurationsabläufe wurden in Abschnitt *5.2* erläutert. Des Weitern wurden die für die Konfigurationsaufgaben notwendigen Vorbedingungen in statische Klassen umgesetzt und in die in Abschnitt *5.1.1* entwickelte Konzept der XML Definitionen integriert.

In Abschnitt *5.3* wurde das Konzept für die Entwicklung des notwendigen SNMP Moduls festgelegt. Das Konzept ermöglicht eine schnelle und einfache Erweiterung des Moduls mit weiteren Funktionen, die bei der Weiterentwicklung von FLASH notwendig sind.

Im Anschluss an das Design wurden die entwickelten Konzepte realisiert. In Kapitel *6* erfolgt die detaillierte Beschreibung der Implementierung. Die Umsetzung der entwickelten Module erfolgte in der objektorientierten Programmiersprache C#. Für die XML Definitionen wurde das XML Schema in Anhang *A* erstellt.

Nach der Entwicklung des ersten FLASH Prototyps wurde in Kapitel *7* die Überprüfung der Entwicklungsergebnisse durchgeführt. Dabei wurden die einzelnen technischen Funktionen, die Betriebssicherheit und die Einhaltung der Zeitvorgaben erfolgreich geprüft.

8.1 FAZIT

Die in Kapitel *4* an die Entwicklung von FLASH gestellt Anforderungen wurden erfüllt. Eine Ausnahme bildet die Integration der Verhaltensbeschreibungen der Switches, die aufgrund einer Fehlfunktion, die in der Systemanalyse aufgedeckt wurde, (noch) nicht automatisiert werden konnten. Eine Fehlerbehebung bzw. Lösung durch den Hersteller stand zum Abschluss dieser Bachelorarbeit noch aus. Sollte es keine Lösung geben, wird die folgende Idee als Alternative umgesetzt. Der Mode Button wird über einen Mikrokontroller angesteuert. Dieser Mikrokontroller wird wie die Terminal Server angesprochen und ermöglicht FLASH die Konfiguration der Switches, ohne dass ein physikalischer Zugriff notwendig ist.

Der entwickelte Prototyp von FLASH wird für die Wiederherstellungs- und Rekonfigurationsaufgaben aller Router der ICND Labore bereits verwendet und minimiert den Arbeitszeitaufwand der Techniker für diese Aufgaben.

Die Minimierung der Fehlkonfigurationen bei den Wartungsarbeiten durch die Techniker stellt für das Unternehmen Fast Lane einen wichtigen, wirtschaftlichen Sicherheitsfaktor dar, der durch den Einsatz von FLASH sichergestellt wird.

Die ausführliche Anforderungsanalyse sowie die durchgeführte Dozentenbefragung in Kapitel *4* haben gezeigt, wie wichtig die Durchführung ist, damit beim Design möglichst viele Aspekte einbezogen werden und spätere Änderungen und damit verbundene Anpassungen am System vermieden werden.

Das flexible Systemdesign auf Grundlage des entwickelten XML Schemas erlaubt auch während des Betriebes eine schnelle Anpassung der Funktionalitäten von FLASH an neue Anforderungen und eine schnelle Implementierung zusätzlicher Funktionen. Diese sind nicht auf die projektrelevante Systemumgebung beschränkt, sondern kann herstellerübergreifend eingesetzt werden.

Der Einsatz von FLASH ist nicht auf das Remote Labor Szenario beschränkt. Szenarien, in denen kommerzielle Lösungen nicht angewendet werden können, da eine Grundkonfiguration der Netzwerkgeräte (noch) fehlt oder der Zugriff über TCP/IP unterbrochen ist, kann auf die Funktionen von FLASH zurückgegriffen werden. Bei der Konfigurationsverteilung werden Fehler vermieden und der Zeitaufwand verringert, sodass die entstehenden Kosten minimiert werden. FLASH übernimmt in diesen Fällen die low level Funktionen unterhalb der Einsatzmöglichkeiten der kommerziellen Systeme.

Mit der Entwicklung von FLASH ist es gelungen ein System für die Automatisierung zu schaffen, dass den Einsatz der Remote Labore im Timesharing Betrieb möglich macht und somit dem Unternehmen eine Verdopplung, teilweise sogar Verdreifachung der Einsatzzeit der vorhandenen Ressourcen ermöglicht. Die Verwendung freier Remote Labor Ressourcen zwischen zwei Zeitzonen kann zudem für kurze, praxisnahe Lerneinheiten verwendet werden. Zum Beispiel als Ergänzung beim Selbststudium oder für die individuelle Nachbereitung am Anschluss einer Schulung. Dafür werden die freien Labor Ressourcen vom Scheduler zur Verfügung gestellt und durch den Einsatz von FLASH werden die Labore zeitnah vorbereitet.

Die Firma Fast Lane hat sich bereits für die Weiterentwicklung von FLASH im Anschluss an diese Bachelorarbeit entschieden. Im ersten Schritt soll die Erweiterung des Systems auf weitere Labore und Netzwerkgeräte erfolgen. Im folgenden Abschnitt wird abschließend ein Ausblick auf die Weiterentwicklung der Funktionen und Möglichkeiten getroffen.

8.2 WEITERENTWICKLUNG VON FLASH

Mit der Weiterentwicklung von FLASH wird im Anschluss an diese Bachelorarbeit begonnen. In *Tabelle 8.1* werden zum einen die Anforderungen der Prioritätsstufe 2 und 3 (vgl. Abschnitt *4.4*), die nicht im Rahmen dieser Bachelorarbeit umgesetzt wurden, aufgeführt. Diese werden als Erstes umgesetzt. In *Tabelle 8.2* sind Funktionen und Weiterentwicklungsmöglichkeiten aufgeführt, die auf der aktuellen Version von FLASH aufbauend umgesetzt werden.

Tabelle 8.1: Anforderungen Prioritätsstufe 2 und 3

Index	Funktion	Beschreibung
F3	User_CFG_Save	Benutzer Konfiguration sichern
F4	User_CFG_Load	Benutzer Konfiguration laden, die zuvor gesichert wurde
F5	File_Read	Dateien im Flash Speicher der Netzwerkgeräte auslesen
F6	File_Erase	Dateien im Flash Speicher der Netzwerkgeräte löschen
F7	File_Load	Dateien über die Konsolen Verbindung laden
F8	File_Save	Dateien über die Konsolen Verbindung sichern
F10	Output_Processing	Um eingespielte Konfigurationen anzeigen zu lassen / Status und Register Informationen auszulesen
F11	IOS_Check	Aktuell laufende Version des Betriebssystems ermitteln

Tabelle 8.2: Funktionen für die Weiterentwicklung von FLASH

Funktion	Beschreibung
Web Einbindung	Im ersten Schritt wird das System Log von FLASH in die Web Oberfläche integriert. Anschließend folgt die Integration der einzelne Funktionen.
Automatische Bestimmung der Systemplattform	Bestimmung der Systemplattform, die sich an einem Terminalserver Anschluss befindet. (z. B. Cisco Router c1841) . Anhand dieser Information und der Information der entsprechenden IOS Version (vgl. F11) kann dann dynamisch die entsprechende XML Datei geladen werden. Es sind dann keine weiteren Informationen außer der physikalischen Verbindungsdaten notwendig.
XML Editor	Für eine komfortablere Bearbeitung der XML Dateien wird ein entsprechender XML Editor für die Techniker erstellt.
Verteiltes System	Erweiterung von FLASH auf weitere Standorte, bei denen keine garantierten Verzögerungszeiten gegeben sind (z. B. Remote Labor in Australien und den USA).
Geräte ID Ermittlung	Für eine automatische Aufnahme der Geräte zwecks Inventarisierung kann bei allen neueren Modellen die entsprechende ID über die Kommandozeile ermittelt werden, sodass diese Aufgabe in FLASH integriert wird und so eine schnelle Aufnahme für die jährliche Inventarisierung möglich ist.
Erweiterung der kompatiblen Netzwerkgeräteliste	Erstellung weiterer XML Definitionen für andere Netzwerkgeräte.

LITERATURVERZEICHNIS

APC.com APC [Online]. - APC, 2008. - 5. März 2008. - http://www.apc.com/.

Bishop Judith C# 3.0 Design Patterns [Buch]. - Sebastopol : O`Reilly, 2008. - 1. Edition.

C# Programming Guide [Online] // C# Programming Guide. - Microsoft. - 16. März 2008. - http://msdn.microsoft.com/en-us/library/67ef8sbd(VS.80).aspx.

c1841 Cisco 1841 Integrated Services Router [Online]. - Cisco Systems, 2007. - 18. März 2008. - http://www.cisco.com/en/US/products/ps5875/index.html.

C2500 Cisco 2500 Series Access Servers, Install and Upgrade Guide [Online]. - Cisco Systems, 26. September 2007. - 14. März 2008. - http://www.cisco.com/en/US/docs/routers/access/2500/software/user/guide/acsvrug.html.

c2960 Cisco Catalyst Switch 2960 with LAN base Software [Online]. - Cisco, 2007. - 18. März 2008. - http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps6406/product_data_sheet0900aecd80322c0c.html.

Case J [Online] // rfc1157 - A Simple Network Management Protocol (SNMP). - Mai 1990. - 6. März 2008. - http://www.ietf.org/rfc/rfc1157.txt.

Case J. [Online] // rfc1901 - Introduction to Community-based SNMPv2. - Januar 1996. - 6. März 2008. - http://www.ietf.org/rfc/rfc1901.txt.

Cavale Mohan Rao Einführung in Microsoft Clusterdienst (MSCS) in Windows Server 2003 [Online]. - Microsoft, 23. Juni 2004. - 10. April 2008. - http://msdn.microsoft.com/de-de/library/ms952401.aspx.

Cisco CiscoWorks LAN Management Solution [Online]. - Cisco Systems, 2008. - 7. März 2008. - http://www.cisco.com/en/US/products/sw/cscowork/ps2425/index.html.

CLI Introduction to Cisco IOS Software [Online]. - Cisco Systems, 31. Oktober 2003. - 18. März 2008. - http://www.ciscopress.com/articles/article.asp?p=101658&seqNum=2.

ConfReg Use of the Configuration Register [Online]. - Cisco, 03. April 2006. - 17. März 2008.

Dimitri van Heesch Doxygen [Online]. - 1. Mai 2008. - 10. Mai 2008. - http://www.stack.nl/~dimitri/doxygen/.

Dominic Létourneau RobotFlow: Open Source Robotics Toolkit for FlowDesigner [Online]. - 5. Oktober 2008. - 25. März 2008. - http://robotflow.sourceforge.net/.

Don Libes The Expect Home Page [Online]. - 20. Oktober 2006. - 22. März 2008. - http://expect.nist.gov/.

Duffy Joe Verwenden der Parallelität für Skalierbarkeit [Online]. - Microsoft, 4. September 2006. - 12. April 2008. - http://www.microsoft.com/germany/msdn/library/net/VerwendenDerParallelitaetFuerSkalierbarkeit.mspx?mfr=true.

Fallside David C. [Online] // XML Schema Part 0: Primer. - 2. Mai 2001. - 13. März 2008. - http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/.

Harel David Statecharts: A visual approach to complex systems [Buch]. - [s.l.] : Department of Applied Mathematics, The Weizmann Institute of Science, 1984.

Hopcroft John E., Motwani Rajeev und Ullman Jeffrey D. Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie [Buch] / Hrsg. WESLEY ADDISON. - München : Pearson Studium, 2002. - 2., überarbeitete Auflage : S. 93-133.

HPOpenView HP OpenView [Online]. - 19. Februar 2008. - 7. März 2008. - http://en.wikipedia.org/wiki/HP_OpenView.

IBM IBM Tivoli NetView [Online]. - IBM, 2008. - 7. März 2008. - http://www-306.ibm.com/software/tivoli/products/netview/.

ISO.ORG ISO.ORG [Online] / Hrsg. (ISO) International Organization for Standardization. - International Organization for Standardization (ISO). - 20. März 2008. - http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39752.

ITU-T Abstract Syntax Notation One [Online]. - 17. Dezember 2007. - 18. März 2008. - http://en.wikipedia.org/wiki/Abstract_Syntax_Notation_One.

Kaltenhäuser IE-Verhaltensspezifikation [Dokument]. - 2005. - 91.

Kasper, A. Switched Rack PDU [Online] // Switched Rack PDU. - APC, 4. April 2008. - 14. März 2008. - http://www.apcmedia.com/salestools/ASTE-6Z6K8G_R0_EN.pdf.

Krause Jörg und Bünning Uwe ASP.NET-Programmierung mit c# [Buch]. - München : ADDISON-WESLEY, 2002. - S. 468-486.

Liberty Jesse Programming C# [Buch]. - Sebastopol : O'reily, 2001. - S. 37-40.

Libes Don Exploring Expect: A Tcl-Based Tool for Automating Interactive Programs [Buch]. - [s.l.] : O'Reilly & Associates, Inc., 1995.

Microsoft XML Documentation Comments (c#) [Online] // XML Documentation Comments (c#). - Microsoft. - 15. März 2008. - http://msdn.microsoft.com/en-us/library/b2s063f7(VS.80).aspx.

MSDN Library - Threading MSDN Library - Threading [Online]. - Microsoft, 2008. - 12. April 2008. - http://msdn.microsoft.com/en-us/library/3dasc8as(VS.85).aspx.

MSDN Library - Timer Timer-Klasse [Online]. - Microsoft, 2008. - 13. April 2008. - http://msdn.microsoft.com/de-de/library/system.timers.timer(VS.80).aspx.

MSDN Library - XmlTextReader XmlTextReader Class [Online]. - Microsoft, 2008. - 24. April 2008. - http://msdn.microsoft.com/de-de/library/system.xml.xmltextreader(en-us,VS.85).aspx.

NIL NIL Data Communications [Online]. - NIL, 1. Januar 2008. - 23. März 2008. - http://www.nil.si/english.

NIL.si NIL Data Communication [Online]. - NIL, 2008. - 5. März 2008. - http://www.nil.si/english.

Ousterhout John K. Tcl and the Tk Toolkit [Buchabschnitt] / Buchverf. Addison-Wesley. - California : Addison-Wesley, 1993.

Postel J. [Online] // rfc854 - TELNET PROTOCOL SPECIFICATION. - Mai 1983. - 7. März 2008. - http://www.ietf.org/rfc/rfc854.txt.

Sommerville Ian Software Engineering [Buch]. - München : Pearson Studium, 2001. - 6. Auflage : S. 269-291.

Toolwire.com Toolwire - The Experiential Learning Company [Online]. - Toolwire, Inc., 2007. - 5. März 2008. - http://www.toolwire.com.

GLOSSAR

.NET
Ist eine von Microsoft entwickelte Softwareplattform. Diese umfasst eine Laufzeitumgebung, eine für Programmierer bestimmte Sammlung von Klassenbibliotheken (API), und angeschlossene Dienstprogramme (Services). Die Plattform ist derzeit in ihrem vollen Umfang nur für Windows verfügbar.

24 x 7
Steht für 24 Stunden und 7 Tage die Woche. Also rund um die Uhr unterbrechungsfrei

Access Server
siehe Terminal Server

Administrationsrechte
siehe Rootrechte

Antwortzeit
Bezeichnet die Zeitspanne zwischen dem Absenden einer Nachricht und dem Empfang der Antwort auf die Nachricht (engl. Response time).

Backend
Bezeichnet das dem Remote Labor zugrunde liegende System, das als Infrastruktur bezeichnet wird.

Backupsystem
Übernimmt die Aufgaben eines Systems, das ausgefallen ist.

Cache
Ein schneller Puffer-Speicher mit dem Ziel die Zugriffszeit zu verringern bzw. eine Verringerung der Anzahl der Zugriffe auf den zu cachenden Speicher.

Core
Bezeichnet die zentrale Komponente eines Labors, die für alle PODs geteilt zur Verfügung steht.

Datenbankmanagementsystem
Verwaltungssoftware einer elektronischen Datenverwaltung.

Demilitarized Zone
Bezeichnet ein Computernetz mit sicherheitstechnisch kontrollierten Zugriffmöglichkeiten auf die daran angeschlossenen Server.

Ethernet
Kabelgebundene Datennetztechnik für lokale Netzwerke (LAN), die einen Datenaustausch der angeschlossenen Geräte mit einer Geschwindigkeit von 10 Mbit/s ermöglicht.

Fast Ethernet
Eine Weiterentwicklung von Ethernet, die mit 100 Mbit/s arbeitet.

Firewall
Stellt eine kontrollierte und gesicherte Verbindung zwischen zwei Netzwerken her.

Frontend
Bezeichnet den Teil des Remote Labors, den die Teilnehmer sehen und auf den sie Zugriff gewährt bekommen.

Gateway
Verbindet Netzwerke miteinander. Daher können auch Netzwerke miteinander kommunizieren, die unterschiedliche Protokolle verwenden.

IP Adresse
Dient zur eindeutigen Adressierung von Rechnern und anderen Geräten in einem IP-Netzwerk.

Kommandozeile
Ein Eingabebereich für die Steuerung eines Betriebssystems.

Konsole
siehe Kommandozeile

Konsolen Anschluss
Schnittstelle eines Netzwerkgerätes um den Zugriff auf die Kommandozeile zu ermöglichen.

Lab Guide
Übungshandbuch für Teilnehmer, die eine Schulung besuchen.

Labor
Eine Einheit von Netzwerkgeräten, auf denen die praktischen Übungen zu einem bestimmten Thema durchgeführt werden können.

Logisches Netzwerk
Bezeichnet ein durch Konfigurationsmaßnahmen getrenntes physikalisch verbundenes Netzwerk.

Managementstation
Ein System auf dem die Netzwerk Management Software ausgeführt wird.

Mehrschichtbetrieb
Ein Verfahren bei dem mehrere Benutzer auf ein und dem selben Labor zeitversetz arbeiten können. Es wird sichergestellt, dass die Laborumgebung immer den entsprechenden Zustand des Nutzers einnimmt.

Netzwerk Management Software
Software, die sich um die Verwaltung und Überwachung von Netzwerken einschließlich der integrierten Geräte, kümmert.

NoMachine
Remote-Desktop-Software, die den Bildschirminhalt eines entfernten Computers auf einen lokalen Rechner übertragen kann.

Pair of Delegates
Beschreibt eine Einheit für die Aufteilung der Geräte eines Labors auf Lerngruppen bestehend aus zwei Teilnehmern.

Password Recovery
Bezeichnet einen Vorgang bei Cisco Netzwerkgeräten, bei dem ein konfiguriertes Password einen neuen Wert zugewiesen bekommt.

Provider
Anbieter von Internetzugängen.

Out-of-Band-Management
Ein Fernwartungskonzept, das mit einem separaten Kanal arbeitet. Über diesen Kanal erfolgt der Zugriff auf den Server. Bei dieser Technik kann die Fernwartung auch dann erfolgen, wenn das Betriebssystem nicht in Betrieb ist oder der Personal Computer abgestürzt ist.

Remote Labor
Besteht aus einer Vielzahl einzelner Labore.

Reverse TELNET
Spezielles Protokoll, das über die Serverseite der Verbindung Daten an eine serielle Verbindung schickt.

Rootrechte
Uneingeschränkte Berechtigung ein System zu bedienen und zu verändern.

Router
Verbindet oder trennt verschiedene Netzwerke.

Rücksetzprozedur
Bezeichnet einen Vorgang, der ein Netzwerkgerät auf einen definierten Zustand versetzt.

Switches
Netzwerk-Komponente zur Verbindung mehrerer Computer (meist in Sternenform).

TELNET
Byteorientiertes Netzwerkprotokoll für eine bidirektionale Kommunikation zweier Systeme.

Terminal Server
System, das den Zugriff auf Netzwerkressourcen über einen zentralen Punkt zur Verfügung stellt.

Timesharing
siehe Mehrschichtbetrieb

uni-direktionale
Bezeichnet eine Verbindung, die eine einseitige Kommunikation zwischen zwei Systemen. Ausschließlich ein System ist in der Lage aktiv zu kommunizieren.

Virtual Private Networks
Technik zur Einbindung von Systemen eines benachbarten Netzes an das eigene (private) Netz, ohne dass die Netzwerke zueinander kompatibel sein müssen.

Wartungsfenster
Bezeichnet ein Zeitfenster, in dem die Wartung an Hard- oder Software stattfindet. Während dieses Zeitfensters wird der Zugriff auf die Hard- oder Software gesperrt.

ABKÜRZUNGSVERZEICHNIS

ASN.1	Abstract Syntax Notation One
APC	American Power Conversion
APP	Anwendungsserver
ASN.1	Abstract Syntax Notation One
AUX	Auxiliary
CCIE	Cisco Certified Internetwork Expert
CD	Compact Disc
CF	Compact Flash
CFG	Configuration
CLI	Command Line Interface
CPU	Central Processing Unit
DB	Datenbank
DBMS	Datenbankmanagementsystem
DMZ	Demilitarized Zone
DTD	Document Type Definition
F	Funktional
FE	Fast Ethernet
FLASH	Fast Lane Automation System for Networking Hardware
FSM	Finite State Machine
GUI	Graphical User Interface
I/O	Input Output
ICND	Interconnecting Cisco Network Devices
ID	Identifikationsnummer
IETF	Internet Engineering Task Force
IIS	Internet Information Server
ILT	Instructor Led Training
IOS	Internetwork Operating System
IP	Internet Protocol
ISO	International Organization for Standardization
ITU	International Telecommunication Union
JRE	Java Runtime Environment
LABORIUS	Mobile Robotics and Intelligent Systems Laboratory
LDAP	Lightweight Directory Access Protocol
MIB	Management Information Base

MSDN	Microsoft Developer Network
nF	Nicht Funktional
NVRAM	Non Volatile Random Access Memory
NX	No Machine
OID	Object Identifier
OOB	Out-Of-Band
PC	Personal Computer
POD	Pair of Delegates
POST	Power On Self-Test
RAM	Random Access Memory
RB	Randbedingung
RDP	Remote Desktop Protocol
RFC	Request for Comments
ROM	Read Only Memory
SNMP	Simple Network Management Protokoll
SQL	Structured Query Language
SSH	Secure Shell Protocol
TCL	Tool Command Language
TCP	Transmission Control Protocol
TELNET	Telecommunication Network
TS	Terminal Server
UDP	User Datagram Protocol
UML	Unified Modeling Language
URL	Uniform Resource Locator
USB	Universal Serial Bus
VPN	Virtual Private Network
W3C	World Wide Web Consortium
XML	eXtensible Markup Language
XSD	XML Schema Definition

ANHANG

A. FLASH - XML SCHEMA DATEI AUFBAU

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <xs:element name ="fsm" type="fsmType" />

  <xs:complexType name="fsmType">
    <xs:sequence>
      <xs:element name="startStateID" type="xs:int" minOccurs="1"
maxOccurs="1"></xs:element>
      <xs:element name="fsmTimeout"   type="xs:int" minOccurs="1"
maxOccurs="1"></xs:element>
      <xs:element name="state"        type="stateType"    minOccurs="1"
maxOccurs="unbounded"></xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="stateType">
    <xs:sequence>
      <xs:element name="id"           type="xs:int"             minOccurs="1"
maxOccurs="1" />
      <xs:element name="description"  type="xs:string"          minOccurs="0"
maxOccurs="1" nillable="true" />
      <xs:element name="timeout"      type="xs:int"             minOccurs="0"
maxOccurs="1" nillable="true" />
      <xs:element name="entry"        type="entryActivityType"  minOccurs="0"
maxOccurs="1" nillable="true" />
      <xs:element name="expect"       type="expectLogic"        minOccurs="0"
maxOccurs="1" nillable="true" />
      <xs:element name="exit"         type="exitActivityType"   minOccurs="0"
maxOccurs="1" nillable="true" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="entryActivityType">
    <xs:sequence>
      <xs:element name="entrySendCommand"  type="xs:string"    minOccurs="0"
maxOccurs="unbounded" nillable="true" />
      <xs:element name="entrySendKey"      type="xs:string"    minOccurs="0"
maxOccurs="unbounded" nillable="true" />
      <xs:element name="entryStartFSM"     type="xs:string"    minOccurs="0"
maxOccurs="unbounded" nillable="true" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="expectActionActivityType">
    <xs:sequence>
      <xs:element name="expectActionSendCommand"  type="xs:string"
minOccurs="0" maxOccurs="unbounded" nillable="true" />
      <xs:element name="expectActionSendKey"      type="xs:string"
minOccurs="0" maxOccurs="unbounded" nillable="true" />
      <xs:element name="expectActionStartFSM"     type="xs:string"
minOccurs="0" maxOccurs="unbounded" nillable="true" />
    </xs:sequence>
```

```
  </xs:complexType>

  <xs:complexType name="exitActivityType">
    <xs:sequence>
      <xs:element name="exitSendCommand"  type="xs:string"    minOccurs="0"
maxOccurs="unbounded" nillable="true" />
      <xs:element name="exitSendKey"      type="xs:string"    minOccurs="0"
maxOccurs="unbounded" nillable="true" />
      <xs:element name="exitStartFSM"     type="xs:string"    minOccurs="0"
maxOccurs="unbounded" nillable="true" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="expectLogic">
    <xs:sequence>
      <xs:element name="expectCase"   type="expectCaseType" minOccurs="0"
maxOccurs="unbounded" nillable="true" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="expectCaseType">
    <xs:sequence>
      <xs:element name="expectString" type="xs:string"
minOccurs="1" maxOccurs="1" />
      <xs:element name="action"       type="expectActionActivityType"
minOccurs="1" maxOccurs="1" />
      <xs:element name="nextState"    type="xs:int"
minOccurs="1" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

B. XML DEFINITION AUSZUG – PASSWORD RECOVERY

```
<?xml version="1.0" encoding="utf-8"?>
<fsm xmlns:xs="http://www.w3.org/2001/XMLSchema"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="c:\flash\xml\schema\FLASH_FSM_XML_Schema_Versio
n10.xsd">

  <!-- GLOBAL MEMBERS-->

  <startStateID>0</startStateID>
  <fsmTimeout>1000000</fsmTimeout>

  <!-- STATES-->

  <state>
    <id>0</id>
    <description>Power OFF</description>
    <timeout>0</timeout>
    <entry>
      <entrySendCommand xsi:nil="true"></entrySendCommand>
      <entrySendKey xsi:nil="true"></entrySendKey>
      <entryStartFSM>STATIC_FSM_POWER_OFF</entryStartFSM>
    </entry>
    <expect>
      <expectCase>
        <expectString>null</expectString>
        <action>
          <expectActionSendCommand></expectActionSendCommand>
          <expectActionSendKey></expectActionSendKey>
          <expectActionStartFSM></expectActionStartFSM>
        </action>
        <nextState>1</nextState>
      </expectCase>
    </expect>
    <exit>
      <exitSendCommand xsi:nil="true"></exitSendCommand>
      <exitSendKey xsi:nil="true"></exitSendKey>
      <exitStartFSM xsi:nil="true"></exitStartFSM>
    </exit>
  </state>

  <state>
    …
  </state>
```

C. ROBOTFLOW – XML BEISPIEL

```
<?xml version="1.0"?>
<FSM name="MyFSMName">

    <Input name="FoundSomething" type="bool"/>
    <Output name="FindSomething" type="float"/>

    <StartState name="WanderAround"/>

    <State name="WanderAround">
        <SetOutput name="FindSomething" value="1.0"/>
    </State>

    <State name="GetSomething">
        <SetOutput name="FindSomething" value="1.0"/>
    </State>

    <Transition from="WanderAround" to="GetSomething">
        <Condition>
            <Left variable="FoundSomething" type="bool"/>
            <Operator type="equals"/>
            <Right constant="true" type="bool"/>
        </Condition>
    </Transition>

    <Transition from="GetSomething" to="WanderAround">
        <Condition>
            <Left variable="FoundSomething" type="bool"/>
            <Operator type="equals"/>
            <Right constant="false" type="bool"/>
        </Condition>
    </Transition>

</FSM>
```

Beschreibung

Eine FSM-Definition kann aus den folgenden FSM-Elementen zusammengesetzt sein: Input, Output, Startstate, State oder Transition. Input repräsentiert den aktuell gültigen Wert eines auslösenden Ereignisses (Trigger). Die Trigger sind für die Durchführung der Übergäne (Transitionen) zwischen den Zuständen notwendig. Ein Output steht für die Änderung, die durch den aktuellen Zustand erzeugt wird. Der StartState definiert lediglich den Namen des Zustandes (State), der als Startzustand des Automaten gilt. Ein State führt beim Aufruf eine Änderung des in SetOutput angegebenen Wertes durch. Der Wert StateOutput wird also dann aktiv, wenn der Zustand es wird. Eine Transition reagiert auf die Änderung der Werte und veranlasst den Wechsel von einem Zustand in einen anderen. Deshalb muss eine Transition mindestens eine Bedingung enthalten. Bei mehreren erfüllten Bedingungen koordiniert der Operator die Logik zwischen den Bedingungen.

D. SNMP MIB UND OID AUFBAU

Tabelle D.1 zeigt einen Auszug aus dem Cisco MIB Modul *OLD-CISCO-TS-MIB***.** Im Folgendem wird der entsprechende MIB Aufbau in ASN.1 Notation gezeigt.

Tabelle D.1: Cisco MIB Auszug

Objekt Name	Object Identifier	Object Type
tsClrTtyLine	1.3.6.1.4.1.9.2.9.10	OBJECT

```
OLD-CISCO-TS-MIB DEFINITIONS ::= BEGIN

IMPORTS
   IpAddress           FROM RFC1155-SMI
   OBJECT-TYPE         FROM RFC-1212
   DisplayString       FROM RFC1213-MIB
   local               FROM CISCO-SMI;

   lts                 OBJECT IDENTIFIER ::= { local 9 }

-- Local cisco Terminal Service Group
-- This group is present in all products which contains
-- asynchronous terminal lines.

{…output ommitted…}

tsClrTtyLine OBJECT-TYPE
      SYNTAX  INTEGER
       ACCESS  read-write
      STATUS  mandatory
      DESCRIPTION "tty line to clear.  Read returns the last
      line cleared.  A value of -1 indicates no lines have been
      cleared."
       ::= { lts 10 }
END
```

E. SNMP RESPONSE PAKET - AUSWERTUNG

Im Folgenden sind alle Status Meldungen die ein SNMP (Version 1) Response Paket beinhaltet zu sehen. Alle Meldungen werden von FLASH ausgewertet und entsprechend weiterverarbeitet. Als erstes wird die Hostadresse (IP Adresse) des Senders angeben, gefolgt von der SNMP Versionsnummern (0 = SNMP Version 1). Darauf folgt der verwendete Community String und die Identifizierung des Paketes als Response Paket durch den PDU Type. Die Request ID wird zufällig von FLASH erzeugt und als Identifizierungswert in den Antwortpaketen weiterverwendet. Es folgen zwei Werte für einen möglichen Fehler und dem dazugehörigen Wert (optional). Abschließend wird die entsprechende OID zurückübertragen mit dem sogenannten Object Value, der in diesem Fall Ausschluss über Erfolg oder Misserfolg der ursprünglichen Anfrage gibt.

```
SNMP Response from:   10.9.0.2
SNMP Version:         0
SNMP Community:       private
SNMP PDU Type:        A2
SNMP Request ID:      18
SNMP Error Status:    00
SNMP Error Index:     00
SNMP OID:             1.3.6.1.4.1.130.62.1.1.12.3.3.1.1.4.2
SNMP Object Value:    01 immediate ON
```

F. FLASH KLASSENDIAGRAMM

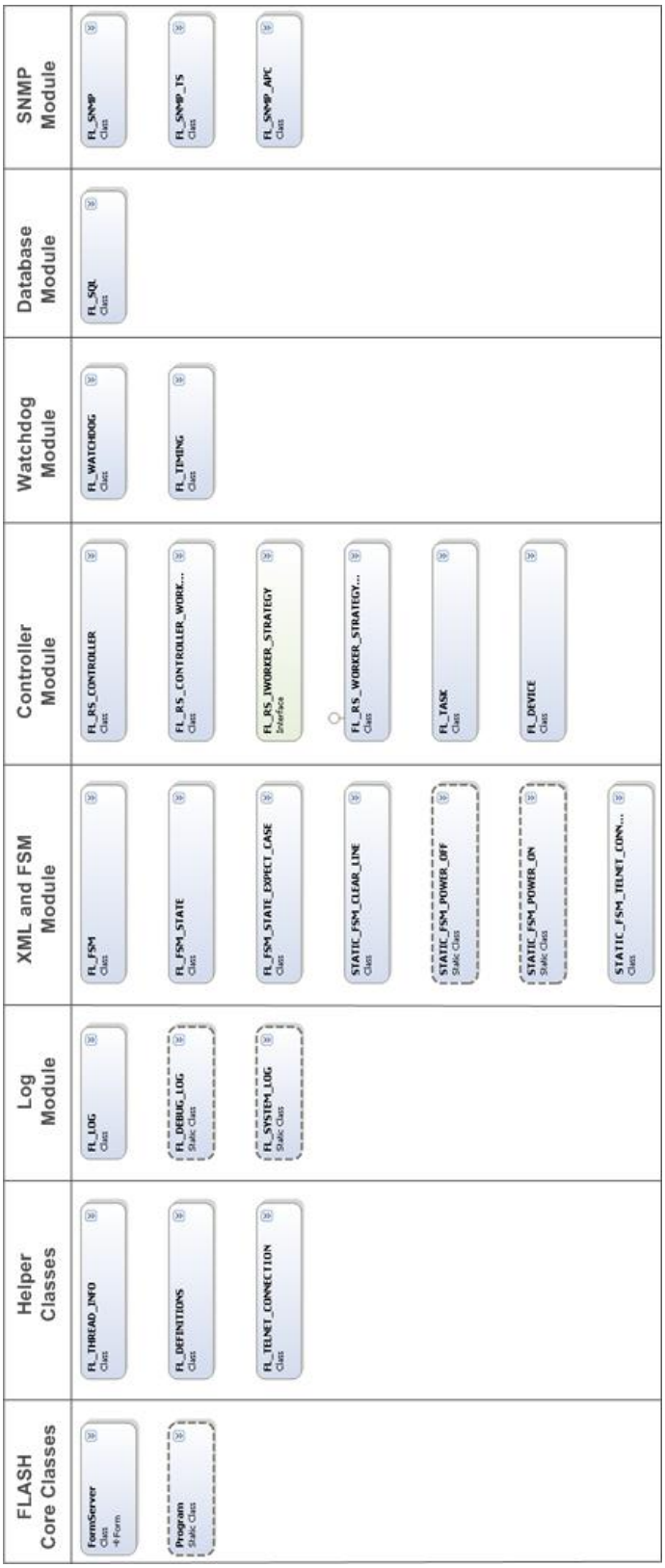

Abbildung F.1: FLASH - Klassendiagramm

G. FLASH – DATENBANKDIAGRAMM

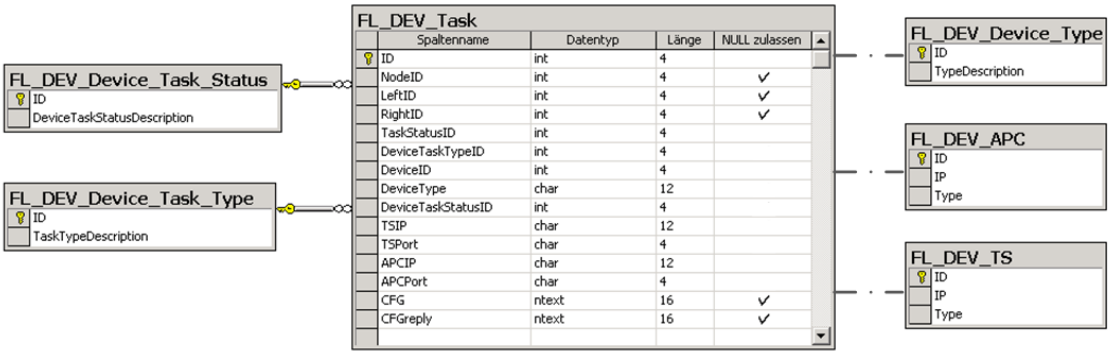

Abbildung G.2: FLASH - Datenbankdiagramm

H. TESTAUFBAU - SOFTWAREVERSIONEN

Die verwendeten Versionen der Betriebssystem und Image Dateien ist in *Tabelle H.2* dokumentiert.

Tabelle H.2: Testaufbau - Software Versionen

Gerät	Software Version
Router C1841	c1841-advipservicesk9-mz.124-16.bin
Switch C2960	c2960- lanbasek9-mz.122-35.SE1.bin
Terminalserver C2511RJ	c2500-c-l.123-24.bin
Anwendungsserver (APP)	Windows 2000 Server (Service Pack 1)
APC AP7922	Network Management Card AOS v3.5.6; Rack PDU APP v3.5.5

I. INHALT DER CD-ROM

Die CD-ROM enthält alle im Rahmen dieser Bachelorarbeit erstellten Dateien und Dokumente.

Tabelle I.3: Inhalt der CD-ROM

Verzeichnis	Inhalt
Bachelorarbeit	Enthält die Bachelorarbeit im .pdf Format
Dozentenbefragung	Ergebnisse der Dozentenbefragung im .pdf Format
FLASH	Enthält das Visual Studio Projekt
FLASH/doc	Enthält die Quellcode Dokumentation im XML Format
Links	Enthält Internetlinks, die im Inhaltsverzeichnis angegeben wurden
Log	Enthält zwei Beispiele für eine „Simple“ und eine „Advanced“ Log Datei
XML/schema	Enthält die XML Schema Datei im .xsd Format
XML/description	Enthält die XML Definition im XML Format

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 28.Mai 2008 _____

Ort, Datum

_____

Boris Böttcher