

Diplomarbeit

Softwareverteilung für Windows

ausgeführt an der
Höheren Abteilung für Informationstechnologie
der Höheren technischen Bundeslehranstalt Wien 3 Rennweg

im Schuljahr 2006/2007

durch

Markus Dietler
Christoph Gratzner
Thomas Pani

unter der Anleitung von
Dipl.-Ing. Herbert Sasshofer

Wien, 10. Mai 2007

1 Einleitung

1.1 Kurzfassung

Diese Diplomarbeit beschäftigt sich mit der Aufgabenstellung, ein System zur Softwareverteilung über das Netzwerk zu planen und zu implementieren. Diese Lösung ist primär für den Einsatz an Schulen und kleinen und mittleren Unternehmen konzipiert und unterstützt Microsoft Windows Betriebssysteme. Die Software arbeitet unabhängig von vorhandenen Verzeichnisdiensten und Datenbanken. Ein weiteres Planungsziel war die möglichst einfache und intuitive Bedienung des Produktes.

Um ein möglichst transparentes und erweiterbares System anzubieten, wird ausschließlich freie Software zur Implementierung verwendet und das resultierende Produkt wiederum unter einer freien Softwarelizenz veröffentlicht.

1.2 Abstract

The aim of this project is to plan and implement a system for software distribution over the network. This solution is primarily built for use at schools and in small and medium-sized enterprises. The target platform is the Microsoft Windows family of operating systems. The software works independently of existing directory services and databases and has been designed to be easily and intuitively manageable.

In order to offer a transparent and extendable system, only free software has been used for the implementation. The resulting product is again released under a free software license.

1.3 Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt und mich auch sonst keiner unerlaubten Hilfe bzw. Hilfsmittel bedient zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden.

Wien, 10. Mai 2007

Markus Dietler

Christoph Gratzler

Thomas Pani

1.4 Präambel

Die Inhalte dieser Diplomarbeit entsprechen den Qualitätsnormen für "Ingenieurprojekte" gemäß §29 der Verordnung des Bundesministers für Unterricht und kulturelle Angelegenheiten über die Reife- und Diplomprüfung in den berufsbildenden höheren Schulen, BGBl. Nr. 847/1992, in der Fassung der Verordnungen BGBl. Nr. 269/1993, Nr. 467/1996 und BGBl. II Nr. 123/97.

Diese Diplomarbeit wurde unter der Anleitung unseres Hauptbetreuers Dipl.-Ing. Herbert Sasshofer und der Nebenbetreuer Mag. Thomas Angerer und Dipl.-Ing. Franz Breunig durchgeführt.

1.5 Vorwort

1.5.1 Motivation

Markus Dietler

Auch in Heimnetzen sollte Software immer up-to-date bleiben. Das ist bei herkömmlichen Netzen ziemlich anstrengend. Man muss sich auf jedem PC anmelden und die Programme aktualisieren. Die Programminstallation ist genauso mühselig wie das Aktualisieren. Außerdem passiert es oft, dass nicht alle PCs auf den selben Stand aktualisiert werden. Hier möchte ich mit meinen Freunden eine Programmlösung schaffen, die diese Probleme nicht mehr aufweist.

Eine Diplomarbeit ist die erste Möglichkeit zu beweisen, dass wir das erworbene Wissen auch für nützliche Zwecke einsetzen können. Weiters sehen viele Firmen in Absolventen mit einer Diplomarbeit bessere Arbeitskräfte, da diese meist selbständiger sind.

Ich freue mich darauf, in einer kleinen Gruppe an einem größeren Projekt arbeiten zu können, vor allem mit meinen Kollegen Thomas Pani und Christoph Gratzler. Zusammen haben wir schon diverse Projekte erfolgreich durchgeführt und wollen uns nun den Herausforderungen eines neuen, großen Projekts stellen.

Christoph Gratzner

In einem privaten Netz stellt man sich oft die Frage, wie man am besten die Updates bzw. Programme auf mehreren Rechnern installiert. Es erfordert sehr viel Zeit, von einem PC zum anderen zu gehen und alles manuell zu installieren. Es ist außerdem schwer zu kontrollieren, welche Programme auf einem PC schon installiert wurden und welche noch fehlen. Wir wollen mit unserer Diplomarbeit eine einfach zu bedienende Software schreiben, die genau diese Funktionen enthält und außerdem gratis ist.

Bei einer Diplomarbeit wird man das ganze Jahr über gefordert. Außerdem arbeitet man als Team und nicht alleine. Diese Teamfähigkeit wird von vielen Firmen gefordert. Daher bringt eine Diplomarbeit bei einer Bewerbung Vorteile gegenüber Mitbewerbern, die keine gemacht haben.

Außerdem freue ich mich auf die Zusammenarbeit mit meinen Kollegen Markus Dietler und Thomas Pani. Wir sind schon seit fünf Jahren in der gleichen Klasse und kennen uns sehr gut. Beide sind fachlich kompetent und es macht Spaß, mit ihnen zusammenzuarbeiten. Wir haben auch schon gemeinsam viele Projekte erfolgreich abgeschlossen.

Thomas Pani

Schon bei der Administration des Heimnetzes taucht immer wieder die mühselige Tätigkeit des Updatens vorhandener Software und Installierens neuer Software auf. Wie ungleich anstrengender muss es dann noch sein, vorinstallierte Arbeitsplätze in EDV-Sälen „schnell“ um die gerade benötigte Software zu ergänzen? Hier liegt meine Freude am Programmieren: Software zu erstellen, die man sich immer schon gewünscht hat, und die auch anderen Leuten ihre Arbeit erleichtern wird.

Außerdem ermöglicht mir diese Diplomarbeit, mein – sowohl in der Schule als auch privat – erworbenes Wissen einzusetzen, um den Herausforderungen, die ein Programmier-Projekt in der Größe einer Diplomarbeit mit sich bringt, zu begegnen.

Die größte Motivation liegt aber in der Zusammenarbeit in einer kleinen Gruppe mit meinen Kollegen Markus Dietler und Christoph Gratzler, mit denen mir die Arbeit besonders großen Spaß macht, und die ich sowohl fachlich als auch persönlich sehr schätze.

1.5.2 Persönliche Erfahrungen

Markus Dietler

Die gesammelten Erfahrungen durch das Projekt waren durchaus neue Erkenntnisse für mich. Es war spannend, den Unterschied zwischen Einzelarbeiten und Projektarbeiten kennenzulernen. Außerdem wurde uns bald bewusst, was es bedeutet, wenn mehrere Leute an einer kritischen Ressource arbeiten müssen. Es wurde uns wirklich klar, was Projektmanagement bedeutet. Endlich konnten wir Theorie in die Praxis umsetzen.

Ich denke, es gibt keine "negativen Erfahrungen". Eben diese sind nämlich die wichtigen Erfahrungen und daher eher als positiv anzusehen. Wenn alles perfekt läuft, entwickelt man sich um einiges weniger weiter, als wenn man neue Erfahrungen sammelt. Als gutes Beispiel nehme ich "Überschneidung von Schule und Diplomarbeit". Meistens kann man es sich einteilen, wie man es benötigt, doch das geht nicht immer. Die zweite Zwischenpräsentation der Diplomarbeit wurde kurzfristig verschoben. Natürlich genau in die Woche, in der wir bis auf die Matura den meisten Stress des Jahres hatten. Ich denke, gerade solche Erfahrungen bereiten uns gut auf das spätere Berufsleben vor.

Die Zeit, die für die Diplomarbeit benötigt wurde, entspricht in etwa unseren Vorstellungen. Wir wurden von ehemaligen Diplomanden und Lehrern gut auf die tatsächliche Länge der Diplomarbeit hingewiesen. Mit den Erfahrungen unserer doch schon länger andauernden Schulzeit gelang es uns, die Zeit ziemlich genau einzuschätzen.

Christoph Gratzner

Vor ungefähr einem Jahr haben wir drei uns für eine Diplomarbeit entschieden. Jeder hat einen Bereich bekommen, für den er sich interessiert und in dem er viel Neues lernen konnte. Es waren alle von der Projektidee begeistert, daher musste niemand von uns zusätzlich motiviert werden. Außerdem gab es das ganze Projekt über keine Streitigkeiten und die Termine wurden von jedem eingehalten.

Wir haben uns alle paar Wochen zusammengesetzt und den Fortschritt besprochen. Dabei wurden Protokolle erstellt; damit war jedem klar, wie der Fortschritt des Projektes war. Wir haben einander bei Problemen geholfen, dadurch sind wir auf viele neue Ansätze und Lösungsmöglichkeiten gestoßen. Wir wohnen zwar verteilt in Wien und Niederösterreich, waren aber in der schulfreien Zeit immer per Telefon oder E-Mail miteinander in Kontakt. Ich bin froh, dass dieses Projekt so reibungslos und erfolgreich durchgeführt werden konnte.

Wir haben uns die Arbeiten sehr gut eingeteilt, trotzdem war es eine anstrengende Zeit, da das Projekt außerhalb der Schulzeit durchgeführt werden musste. Daher haben wir speziell in den verschiedenen Ferien die Hauptarbeit geleistet. Wir haben alle unsere Ziele erfüllt und auch manche der Kann-Ziele implementieren können. Daher bin ich mit unserer Arbeit sehr zufrieden und würde dieses Projekt mit den gleichen Teamkollegen wieder machen. Außerdem finde ich es gut, dass man das theoretische Projektmanagement in einem konkreten und langfristigen Projekt anwenden konnte. Ich habe in diesem Jahr sehr viel dazugelernt und bin mir sicher, dass mir dieses Wissen im späteren Berufsleben weiterhelfen wird.

Thomas Pani

Ich habe mich vor allem deshalb für die Mitarbeit an diesem Projekt entschieden, da mich die Herausforderung reizte. Zwar bedeutet ein solches Projekt (im Umfang von etwa 200 Stunden pro Teammitglied) vor allem im Matura-Jahrgang eine zusätzliche Belastung, doch die Erfahrungen, die ich in dieser Umgebung sammeln konnte, machen diesen Aufwand mehr als wett.

Besonders interessant war, dass ich in diesem Projekt nicht nur meine technischen Interessen einbringen konnte – dies ist das bei weitem umfangreichste Programmierprojekt, an dem ich bis jetzt mitarbeiten durfte – sondern auch die Position des Projektleiters übernahm. Da eine Diplomarbeit nicht nur ein Projekt auf der „grünen Wiese“ im Rahmen des Projektmanagement-Unterrichts bedeutet, sondern vor allem die Matura-Note auf dem Spiel steht, ist diese Rolle umso interessanter. Eine große Erleichterung war, dass meine Teammitglieder seit fünf Jahren die gleiche Klasse besuchen und wir unsere Stärken – aber auch Schwächen – kennen und so auch auf diese eingehen konnten. Außerdem haben wir schon häufiger zusammen an größeren Aufgabenstellungen gearbeitet, eine anfängliche Gruppenbildungsphase konnte also entfallen und die Arbeit sofort aufgenommen werden.

Die schönste Erinnerung an dieses Projekt wird aber wohl die ungetrübte Motivation aller drei Teammitglieder sein, in diesem Projekt die jeweiligen Stärken einzusetzen und so aus dem – im Rahmen der HTL-Ausbildung und zusätzlich privat erworbenen – Wissen ein nicht nur einsetzbares, sondern sogar konkurrenzfähiges Produkt herzustellen.

1.5.3 Danksagung

Wir möchten auf diesem Wege unseren Betreuern danken, die sich mit viel Engagement für dieses Projekt eingesetzt und durch ihr Wissen zum Gelingen dieses Projekts beigetragen haben.

1.5.4 Kapitelzuordnung

Markus Dietler:

- Implementierung
 - o Datenbank
 - o Verzeichnisdienst
 - o Webinterface

Christoph Gratzler:

- Implementierung
 - o Infrastruktur
 - o Pakete
- Tests

Thomas Pani:

- Projektmanagement
- Implementierung
 - o Anmerkungen
 - o Serverapplikation
 - o Clientapplikation

Inhaltsverzeichnis

1	Einleitung.....	3
1.1	Kurzfassung.....	3
1.2	Abstract.....	3
1.3	Ehrenwörtliche Erklärung.....	4
1.4	Präambel.....	4
1.5	Vorwort.....	5
1.5.1	Motivation.....	5
1.5.2	Persönliche Erfahrungen.....	8
1.5.3	Danksagung.....	11
1.5.4	Kapitelzuordnung.....	11
	Inhaltsverzeichnis.....	13
	Tabellenverzeichnis.....	16
	Auflistungsverzeichnis.....	17
	Abbildungsverzeichnis.....	19
2	Projektmanagement.....	21
2.1	Problemstellung.....	21
2.2	Lösungsansatz.....	22
2.3	Team.....	23
2.4	Planung.....	23
2.4.1	Pflichtenheft.....	24

2.4.2	Projektablauf	25
2.4.3	Ressourcenplanung	27
2.5	Umsetzung	28
2.6	Projekt abrechnung	29
3	Implementierung	31
3.1	Anmerkungen	31
3.2	Infrastruktur	31
3.3	Datenbank.....	34
3.3.1	Datenbankkonzept	35
3.3.2	Entity Relationship Modell (ER-Modell).....	36
3.3.3	Create Statement der Datenbank.....	39
3.4	Verzeichnisdienst	40
3.4.1	Installation von LDAP	40
3.4.2	Konfiguration von LDAP.....	41
3.4.3	Erzeugung der LDAP Struktur	43
3.5	Webinterface	44
3.5.1	Gestaltung des Webinterface	45
3.5.2	Sicherheit	46
3.5.3	Kommunikation mit der Datenbank	48
3.5.4	Navigation	50
3.5.5	Style des Webinterface.....	53
3.6	Pakete.....	56

3.6.1	Virenschanner Updates.....	57
3.6.2	Office	61
3.6.3	Microsoft Updates	69
3.6.4	Andere Pakete	78
3.7	Serverapplikation	80
3.7.1	Protokoll.....	80
3.7.2	Serverimplementierung	84
3.8	Clientapplikation.....	95
3.8.1	Clientimplementierung.....	95
4	Tests	105
5	Quellenverzeichnis	107

Tabellenverzeichnis

Tabelle 1 Betreuer	23
Tabelle 2 Projektteam	23
Tabelle 3 Erlaubte Paket-Codes	83
Tabelle 4 Erlaubte Status-Codes	84
Tabelle 5 Erklärung der Parameter in yasds.ini / yasds.conf	89
Tabelle 6 Erklärung der Parameter in yasdc.ini	102

Auflistungsverzeichnis

Auflistung 1 create statement database	39
Auflistung 2 /etc/ldap/slapd.conf	41
Auflistung 3 Auszug samba.schema	42
Auflistung 4 Authentication1.ldif	43
Auflistung 5 dcObject.....	43
Auflistung 6 organization	44
Auflistung 7 Authentication2.ldif	44
Auflistung 8 Authentication3.ldif	44
Auflistung 9 main_frame.inc.php	46
Auflistung 10 auth.php	47
Auflistung 11 logout.php.....	48
Auflistung 12 db.php	48
Auflistung 13 CSS Einbindung	54
Auflistung 14 style1.css	55
Auflistung 15 update_skript.cmd	58
Auflistung 16 vdf.links.....	58
Auflistung 17 %windir%\system32\drivers\etc\hosts	60
Auflistung 18 winupdate.cmd - Teil 1	71
Auflistung 19 winupdate.cmd - Teil 2.....	72
Auflistung 20 winupdate.cmd - Teil 3.....	73
Auflistung 21 yasds.ini / yasds.conf	88
Auflistung 22 yasds.py: create_daemon()	90
Auflistung 23 yasds_service.py	93
Auflistung 24 setup.py	94
Auflistung 25 Deklaration der per P/Invoke aufzurufenden Funktionen.....	97

Auflistung 26 yasdc.ini102

Abbildungsverzeichnis

Abbildung 1 Terminplan	25
Abbildung 2 Projektstrukturplan	26
Abbildung 3 Entity Relationship Model	36
Abbildung 4 yasd Webinterface	50
Abbildung 5 Vorbereiten von Office für die Verteilung	63
Abbildung 6 Custom Installation Wizard - Komponenten entfernen	66
Abbildung 7 Flussdiagramm - Ablauf einer Windows Update Installation	75
Abbildung 8 Flussdiagramm der Client-Server-Kommunikation.....	81

2 Projektmanagement

2.1 Problemstellung

Die Idee zu diesem Projekt entstand aus einer Beobachtung aus dem Schulalltag: Nachdem die HTL Wien 3 Rennweg eine Abteilung Informationstechnologie führt, sind entsprechend viele EDV-Säle und – speziell für den jeweiligen Ausbildungsschwerpunkt – diverse Medientechnik- bzw. Netzwerktechnik-Labors vorhanden.

Allerdings gestaltet sich die Administration dieser PCs relativ kompliziert: Um die entsprechenden Übungen praktisch umsetzen zu können, haben die Schüler mit Administrator-Rechten Zugriff auf die Computer. Das ist zwar für eine sinnvolle Vermittlung der Inhalte unumgänglich – leider kommt es hier aber zu vielen ungewollten Fehlkonfigurationen, die dazu führen, dass entweder Windows neu aufgesetzt und alle Anwendungen neu installiert werden müssen, oder zumindest ein vorher angefertigtes Backup zurückgespielt werden muss. Da die Kontrolle dieser Vorgänge notwendig ist, verbringen die Kustoden einen großen Teil ihrer Zeit allein mit dem Erhalten vorhandener Infrastruktur, anstatt sie zu verbessern.

Ein anderes Beispiel, das nicht nur auf Schulen, sondern auch auf KMUs und größere Heim-Installationen zutrifft: Soll eine neue Anwendung – oder gar nur ein Update bereits installierter Software – auf allen PCs installiert werden, so muss der Administrator mit Installations-CD oder USB-Stick „bewaffnet“ von PC zu PC gehen und sich dort durch die Setup-Programme klicken. Das hält nicht nur den Administrator selbst, sondern vor allem auch die Anwender von ihrer eigentlichen Arbeit ab.

In großen Unternehmen wird diese Aufgabe durch den Einsatz kommerzieller Software-Management Lösungen – wie dem „System Management Server“ von Microsoft, „Tivoli“ von IBM oder „NetInstall“ des deutschen Unternehmens Enteo – gelöst. Diese Lösungen setzen aber zwingend den Einsatz eines Microsoft Windows Server Betriebssystems (für das zusätzliche Lizenzgebühren entfallen) voraus; außerdem ist ein ernst zu nehmender Einarbeitungsaufwand zu leisten, bevor das System produktiv genutzt werden kann, da die Möglichkeiten weit über die einfache Softwareverteilung hinausgehen.

Wünschenswert wäre daher ein freies System zur Softwareverteilung unter Windows, das auf einer „normalen“ (Windows 2000/XP) Windows-Installation läuft und die Funktionalitäten eines reinen Softwareverteilungs-Systems erfüllt. Genau ein solches System soll im Rahmen dieses Projekts entwickelt werden.

2.2 Lösungsansatz

Der vom Projektteam erarbeitete Lösungsansatz sieht folgende Teilbereiche vor: Infrastruktur, Pakete, Backend, Webinterface, Client-/Server-Implementierung.

Zuerst wird die notwendige Infrastruktur (Hard- und Software) geschaffen. Auf dieser Basis wird anschließend die eigentliche Softwareverteilungslösung erarbeitet. Diese besteht aus einer Clientapplikation, die auf jedem Computer, der Softwarepakete beziehen soll, installiert ist. Sie kommuniziert über ein zu spezifizierendes Protokoll mit der Serverapplikation. Inhalt dieser Kommunikation sind Paket-Informationen; diese Daten bezieht die Serverapplikation aus dem Backend (Datenbank und Verzeichnisdienst) und bereitet sie für die Clientapplikation auf. Die Administration erfolgt über ein im Browser bedienbares Webinterface. Zur Verwendung mit dem System werden Beispiel-Softwarepakete bereitgestellt.

Client- und Serverapplikation sowie das Webinterface werden vom Projektteam programmiert, hinsichtlich der anderen Komponenten sollen vorhandene Produkte evaluiert und eventuell entsprechend angepasst werden.

2.3 Team

Die Struktur und Organisation des Projektteams wurde wie folgt gestaltet:

Name	Funktion
Dipl.-Ing. Herbert Sasshofer	Hauptbetreuer Fach „Netzwerkmanagement“
Mag. Thomas Angerer	Hauptbetreuer-Stv. Fach „Lokale Netzwerke“
Dipl.-Ing. Franz Breunig	Nebenbetreuer Fach „Netzwerkprogrammierung“

Tabelle 1 Betreuer

Name	Funktion	Aufgabengebiet
Markus Dietler	Dokumentations- Verantwortlicher	Backend, Webinterface
Christoph Gratzler	Projektleiter-Stv.	Infrastruktur, Pakete
Thomas Pani	Projektleiter	Client- und Serverapplikation

Tabelle 2 Projektteam

2.4 Planung

Die Planung des Projektes, auf die ein erheblicher Anteil des Gesamtaufwandes entfiel, wird in den folgenden Kapiteln dargelegt. Ziel dieser Planungsphase war es, den Projektaufwand möglichst genau abzuschätzen, auf die Teammitglieder aufzuteilen, und Abhängigkeiten zwischen den einzelnen Aufgabengebieten aufzuzeigen.

Um während der Umsetzungsphase jedem der Mitarbeiter einen Überblick über den Gesamtstatus geben zu können, wurden in regelmäßigen Abständen Sitzungen abgehalten. Weiters wurden in diesen Treffen Schnittstellen zwischen den Einzelanwendungen geschaffen, um die spätere Kompatibilität sicherzustellen.

2.4.1 Pflichtenheft

Aus den oben angeführten Kriterien des Softwareverteilungs-Systems, das im Rahmen dieser Diplomarbeit erarbeitet werden soll, ergeben sich folgende Zielsetzungen:

Muss-Ziele

- Es wird eine Softwareverteilungslösung bereitgestellt, die aus einer Serverapplikation, die die Software verteilt, und dem zugehörigen Client, der die Software bezieht und installiert, besteht.
- Um die Berechtigung zum Beziehen der Software zu erhalten, muss sich der Client gegenüber dem Server authentifizieren.
- Die Implementierung wird in einem der EDV-Säle der HTL 3 Rennweg getestet.
- Folgende Software kann über das System verteilt werden:
 - Antiviren-Updates des zum Einsatz kommenden Virenschanners
 - Microsoft-Updates
 - ein Office-Paket
- Die Softwareverteilungslösung (sowohl Server als auch Client) ist unter Microsoft Windows XP lauffähig.
- Pakete lassen sich in Prioritäts-Stufen unterteilen, die die Reihenfolge der Paket-Installation beeinflussen.
- Das Produkt wird unter einer von der Open Source Initiative anerkannten Lizenz freigegeben.

- Über die technischen Ausprägungen der Lösung wird eine Studienphase geführt.

Kann-Ziele

- Zusätzliche Pakete werden bereitgestellt.
- Die Authentifizierung findet gegen einen Verzeichnisdienst (LDAP) statt.
- Die Implementierung des Servers ist betriebssystemunabhängig.
- Es wird eine Prioritätsstufe eingeführt, die dem User das Aufschieben der Installation bis zu einer festlegbaren Deadline gestattet.
- Die Client-PCs lassen sich zu Gruppen zusammenfassen, die jeweils die gleichen Pakete erhalten.
- Ein System zur Verwaltung von Softwarelizenzen wird eingebaut.

2.4.2 Projektablauf

Um das Projekt terminlich abzugrenzen wurden ein Terminplan und ein Projektstrukturplan erstellt.

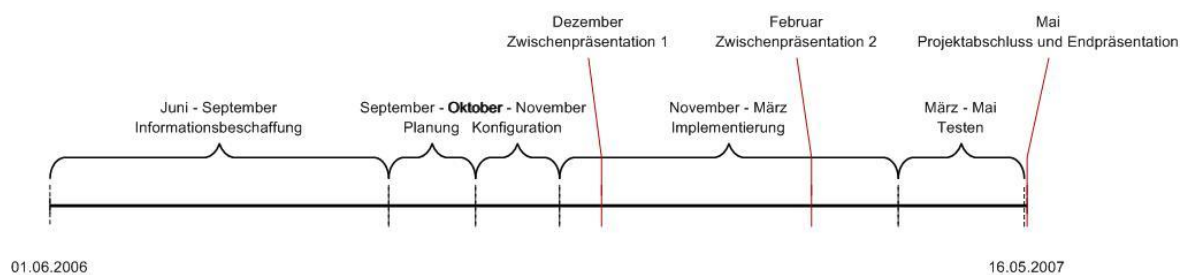


Abbildung 1 Terminplan

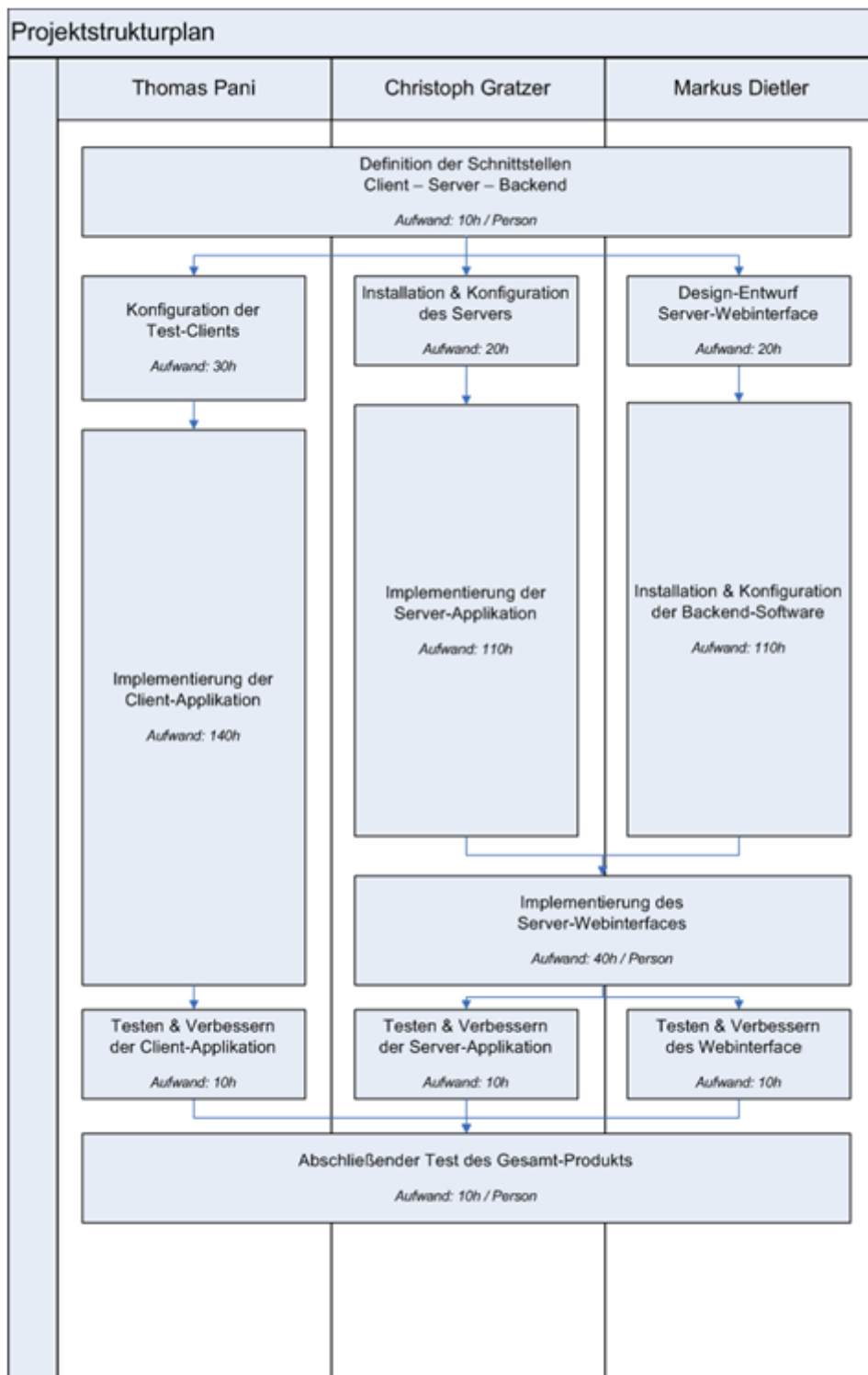


Abbildung 2 Projektstrukturplan

2.4.3 Ressourcenplanung

Hardware

- ein PC pro Teammitglied (vorhanden, privat von den Teammitgliedern bereitgestellt)
- ein Server mit entsprechend großer Festplatte für die Software-Pakete
- ein Server zum Einsatz als LDAP-Server
- Testclients (vorhanden in den EDV-Sälen NWTK1 und NWTK2)
- diverse Netzwerkhardware (Switch, Kabel, ...)

Software

- Fedora Core 5
- Windows XP Lizenzen für die Testclients und den Server
- GIMP
- Microsoft Visual Studio 2005
- Microsoft Office Powerpoint, Project, Visio, Word
- Notepad++
- wget aus den UnxUtils
- xampp lite
- an die Testclients zu verteilende Software

Raumbedarf

Zugang zu den beiden Netzwerktechnik-Labors.

2.5 Umsetzung

Basierend auf dem unter 2.2 Lösungsansatz und 2.4 Planung vorgestellten Ansatz wurde ein Softwareverteilungssystem entwickelt, welches den Anforderungen entspricht. Folgende Details bezüglich der technischen Realisierung wurden während der geführten Studienphase festgelegt:

Das Backend dient zur Speicherung der verwendeten Daten. Dazu werden ein MySQL-Datenbankserver und ein OpenLDAP-Server gewählt. Die Datenbank wird als ER-Modell entworfen, anschließend normalisiert und schließlich in SQL-Statements abgebildet, mit denen die Datenbank angelegt wird. Für den LDAP-Server wird das Daten-Schema aus dem Samba-Projekt verwendet, um Information über Clients abzulegen.

Das Webinterface stellt eine Administrationsoberfläche für die Softwareverteilung zur Verfügung. Es wird in der Skriptsprache PHP realisiert und dient zur Manipulation der Daten im Backend.

Als nötige Infrastruktur wird ein Apache Webserver zum Bereitstellen des Webinterface verwendet. Außerdem werden Test-Clients und -Server mit den Windows-Versionen 2000, XP, XP mit Service Pack 1 sowie XP mit Service Pack 2 installiert.

Der Server-Dienst wird in der Skriptsprache Python realisiert. Er kommuniziert mit den einzelnen Clients über ein selbst entwickeltes Netzwerk-Protokoll und bezieht seine Informationen aus dem Backend. Die Paketverteilung erfolgt über das Standard-Protokoll HTTP bzw. Windows-(SMB)-Freigaben. Da zwar Software an Windows-Clients verteilt wird, der Server aber möglichst flexibel sein soll, wird der Server-Dienst plattformunabhängig implementiert, d.h. er läuft auch unter anderen Systemen wie Linux oder den diversen Unix-Varianten.

Grundsätzlich werden alle Komponenten auch auf einem Nicht-Windows-Betriebssystem laufen, es wird aber nur der Server-Dienst auf Linux-Systemen getestet.

Der Client wird in der Programmiersprache C# mit Unterstützung von Microsofts .NET Framework und Platform SDK geschrieben. Der Client kommuniziert für den Benutzer unsichtbar mit dem Server, führt notwendige (De-)Installationen durch, anschließend wird der Benutzer von der Oberfläche über die durchgeführten Änderungen informiert.

2.6 Projektabrechnung

Die materiellen Ressourcen, die im Rahmen dieses Projekts bis zur Fertigstellung benötigt wurden, stimmen mit den unter 2.4.3 Ressourcenplanung vorgestellten Ansätzen überein.

Die zur Umsetzung benötigte reine Arbeitszeit betrug schätzungsweise 160 Stunden pro Person.

3 Implementierung

3.1 Anmerkungen

Für das Gesamtprodukt wurde der Name „yasd“ gefunden. Dieser Name geht auf ein in der OpenSource-Welt übliches Akronym zurück, das immer mit den Buchstaben „ya“ – also „Yet Another“ – beginnt. Ausgeschrieben bedeutet yasd „Yet Another Software Distribution“. Der Name wird auch im Folgenden als Bezeichnung für das im Rahmen dieser Diplomarbeit verwirklichte Gesamtsystem verwendet.

3.2 Infrastruktur

In diesem Abschnitt wird erklärt, warum wir uns für Windows XP als Serverplattform entschieden haben, wie es eingerichtet wird, und welche Dienste auf dem Server laufen.

Wir wollten unsere Diplomarbeit für Schulen, kleine Firmen und natürlich für Privathaushalte machen. Windows XP ist in diesem Umfeld stark verbreitet und ist auf jedem neu gekauften Rechner vorinstalliert. Außerdem wollten wir, dass unsere Softwareverteilungslösung auch auf Systemen ohne Serverbetriebssystem läuft, da Lizenzen dafür sehr teuer sind.

Als Server kann man außerdem Windows 2000 verwenden. Auf anderen Windows Betriebssystemen sollte er grundsätzlich auch laufen, was wir aber nicht getestet haben. Bei der Auswahl unserer Serverdienste haben wir darauf geachtet, dass die verschiedenen Dienste frei verfügbar sind. Außerdem sollten die Dienste ohne lange Einrichtung und Installation lauffähig sein. Daher haben wir uns für XAMPP lite entschieden. Diese Sammlung von Diensten ist speziell für unsere Bedürfnisse

geeignet. Gepackt hat dieses Paket weniger als 20MB. Das war für uns sehr wichtig, da wir eine größere Datei nicht über das Internet verteilen hätten können. Auf unserem Server laufen folgende Dienste:

- Webserver (Apache)
Dieser wird für die Bereitstellung des Webinterface und zur Freigabe von Virens Scanner-Updates und Paketen benötigt.
- Datenbank (MySQL)
Die Datenbank ist für unser Projekt sehr wichtig, da darin alle nötigen Informationen für die Verteilung gespeichert sind. Die Datenbank wird in einem eigenen Kapitel näher beschrieben.
- Update Dienste
Diese Dienste bestehen aus mehreren Skripten und dienen zur Aktualisierung der Virens Scanner Updates und zum Downloaden der neuesten Windows Updates. Auch diesen Skripten werden eigene Bereiche in der Diplomarbeit gewidmet.
- yasd Server Dienste
Der yasd Server kann entweder als Dienst im Hintergrund oder als normales Konsolenprogramm gestartet werden. Er dient zur Verteilung der Updates und Programme an yasd Clients.

Einrichtung

Zuerst lädt man sich von der offiziellen Homepage XAMPP lite herunter: <http://www.apachefriends.org/de/xampp-windows.html>.

Neue Versionen dieses Paketes werden in unregelmäßigen Abständen veröffentlicht. Man sollte das Paket regelmäßig aktualisieren, da in neuen Paketen entweder Sicherheitslücken geschlossen werden oder neue Funktionen enthalten

sind. Die heruntergeladene Datei kann man anschließend einfach in einen beliebigen Ordner auf dem yasd Server entpacken.

Das Webinterface steht in einer gepackten Datei zur Verfügung. Nach dem Extrahieren der Datei erhält man einen Ordner, den man nun in das `htdocs` Verzeichnis von XAMPP kopiert. Standardmäßig heißt der Ordner zur einfachen Erkennung `yasd`. Danach ist das Webinterface über einen beliebigen Webbrowser unter folgenden Adressen erreichbar: `<http://IP_des_yasd_Servers/yasd>` oder `<http://Name_des_yasd_Servers/yasd>`.

Auch der AntiVir Updater und die Windows Update Skripte sind in einer gepackten Datei verfügbar. Die extrahierten Ordner werden hier aber direkt in das XAMPP-Verzeichnis kopiert. Anschließend müssen noch die Datenbankbefehle in die Datenbank importiert werden. Das kann man zum Beispiel über die Weboberfläche `phpmyadmin` machen.

Das Hauptprogramm am Server ist der yasd Server Dienst. Dieser wird als ausführbare Setup Datei zur Verfügung gestellt. Nach der Installation des yasd Server Dienstes findet man Einträge im Startmenü. Mit diesen kann man den Dienst starten oder stoppen. Außerdem kann der Dienst wieder entfernt oder die Logdatei für eine Fehlersuche angesehen werden. Weiters kann über das Startmenü die Konfigurationsdatei geöffnet werden. Hier müssen der Datenbankname und die Authentifizierungsdaten für die yasd Datenbank angegeben werden. Man kann zudem auswählen, ob die Anmeldung der Clients über einen LDAP-Server erfolgen soll. Diese Konfigurationsdatei sollte nach der Installation angepasst und anschließend abgespeichert werden. Damit sich der Administrator nicht um den Dienst kümmern muss, wird er standardmäßig automatisch gestartet.

Freigabe

Auf dem Server gibt es einen Ordner `freigabe`, der über eine Windows Freigabe für Clients zugänglich ist. Mit diesem Ordner werden die verschiedenen Programme und Updates an Clients verfügbar gemacht. Das hat den Vorteil, dass ein Client die Installation über das Netzwerk startet und so die übertragene Dateimenge auf ein Minimum verringert wird. Manche Installationen, wie zum Beispiel die Office Installation, würden sehr viel länger dauern, wenn man die gesamten 900MB vor einer Installation auf den Client PC übertragen müsste. Daher haben wir uns für diese Möglichkeit entschieden. Kleinere Dateien können aber genauso über den Webserver freigegeben werden. Der Client Dienst kopiert sich dann die benötigte Datei, entpackt sie wenn nötig und führt anschließend die Installation durch.

Letzte Schritte

Zum Abschluss der Einrichtung des yasd Servers muss man nur noch die Pakete/Programme in den Freigabe-Ordner kopieren und den yasd Server Dienst über das Startmenü starten. Nun kann man die Pakete, die man verteilen will, über das Webinterface hinzufügen. Der yasd Server ist nun über das Webinterface steuerbar und kann für die Verteilung eingesetzt werden.

3.3 Datenbank

Schon am Anfang des Projektes war klar, dass große Datenmengen auftreten würden, die verwaltet werden müssen. Im Speziellen geht es hier um die Daten, die bezüglich der Softwareverteilung gespeichert werden müssen. Zum Beispiel: Wie heißt der Client? Wo finde ich die Installationsdatei für dieses Paket? Ist der Client befugt diese Pakete zu bekommen?

Die einfachste Lösung diese Daten sinnvoll zu verwalten, ist die Benutzung einer Datenbank. Mit einer guten Struktur ist es möglich, diese Daten einfach abzuspeichern und gezielt abzufragen.

Eine der elementaren Entscheidungen unseres Projekts war die Auswahl der Datenbank. Wir wollten eine Datenbank, die folgende Eigenschaften aufweist:

- billig beziehungsweise gratis
- klein und leicht in Windows zu integrieren
- gut dokumentiert
- leicht mit dem Webinterface zu verbinden

Vor allem die ersten zwei Punkte gaben den Ausschlag, dass die Entscheidung auf MySQL fiel. Da yasd unter der GPL veröffentlicht wird, kann auch MySQL unter den Bedingungen dieses Lizenzmodells verwendet werden ([MySQL2006]). Außerdem gibt es MySQL in einem Paket mit anderen benötigten Komponenten für Windows. Das Paket heißt XAMPP und beinhaltet alle für uns benötigten Ressourcen. Apache 2.0 als Webserver, MySQL als Datenbank und PHP für Kommunikation zwischen Webinterface und Datenbank.

Schlussendlich hat die gute und umfangreiche Dokumentation die Entscheidung für MySQL zusätzlich beeinflusst.

3.3.1 Datenbankkonzept

Es dauerte relativ lange, bis der erste Entwurf der Datenbank fertig war. Es war am Anfang noch nicht absehbar, wie viele und welche Attribute der Server benötigen wird. Dennoch war das erste Modell schon beinahe identisch mit dem des fertigen Produktes. Die einzige nötige Änderung war das Hinzufügen eines Attributes, um ein Kann-Ziel erfolgreich in Angriff zu nehmen.

3.3.2 Entity Relationship Modell (ER-Modell)

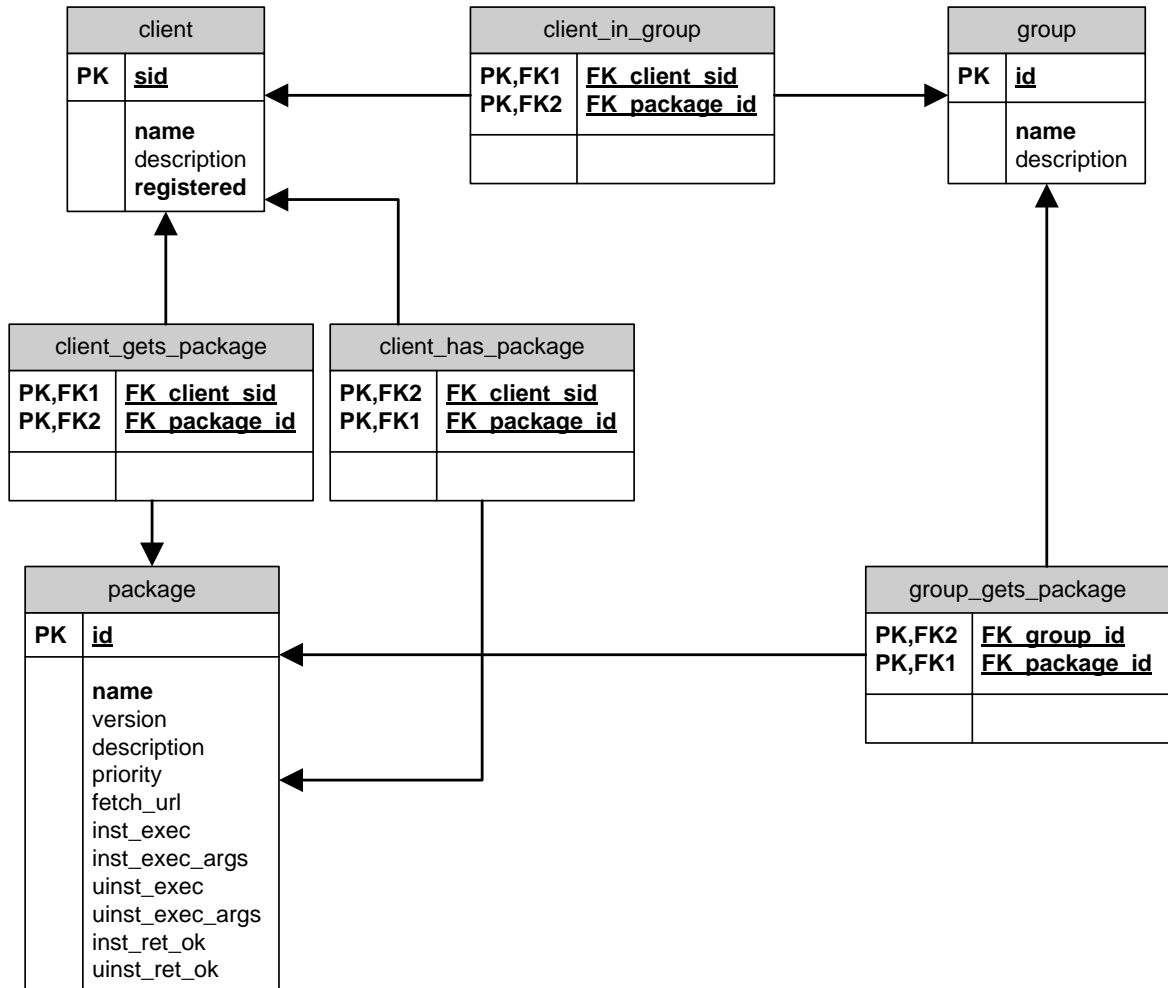


Abbildung 3 Entity Relationship Model

Die Datenbank besteht aus 3 wesentlichen Entitäten.

Client

Ein Client ist ein PC, der Pakete beziehen soll.

- **sid**:

Die SID steht für Security Identifier. Jeder PC hat eine eindeutige SID, die es nur einmal gibt. Anhand dieses Wertes werden Clients identifiziert.

- `name`:
Der Name des Clients.
- `description`:
Unter `description` kann man eine kurze Beschreibung des Clients angeben.
- `registered`:
Nur registrierte Clients haben die Befugnis Pakete verteilt zu bekommen.
Hier wird der Client durch `1` für registriert und `0` für nicht registriert gekennzeichnet.

Gruppe

Um die Administration der Pakete klein zu halten, ist es möglich Gruppen zu erstellen. Pakete, die dieser Gruppe zugeteilt sind, werden an alle Gruppenmitglieder verteilt.

- `id`:
Dient zur eindeutigen Kennzeichnung der Gruppe.
- `name`:
Der Name der Gruppe.
- `description`:
Unter `description` kann man eine kurze Beschreibung der Gruppe angeben.

Package

Unter dem Namen `package` sind Pakete zu verstehen. Jegliche Software, die von unserem yasd-Server bereitgestellt wird, ist in Pakete eingeteilt.

- `id`:
Dient zur eindeutigen Kennzeichnung des Paketes.
- `name`:
Der Name des Paketes.

- `version:`
Die aktuelle Version des Paketes.
- `description:`
Unter `description` kann man eine kurze Beschreibung über des Paketes angeben.
- `priority:`
Dient zum Festlegen der Priorität des Paketes. Desto höher der Wert, desto dringlicher ist es, das Paket zu installieren.
- `fetch_url:`
Gibt eine Datei an, die vor der Installation oder Deinstallation heruntergeladen werden soll.
- `inst_exec:`
In `inst_exec` steht der Pfad, unter dem das Programm für die Installation des Paketes bereitgestellt wird.
- `inst_exec_args:`
Hier werden die Parameter für das oben genannte Programm angegeben.
- `uninst_exec:`
Wie `inst_exec`, nur wird hier der Pfad des Deinstallations-Programmes angegeben.
- `uninst_exec_args:`
Hier werden die Parameter für die Deinstallation angegeben.
- `inst_ret_ok:`
Eine mit Strichpunkten getrennte Liste von Rückgabewerten einer erfolgreichen Installation.
- `uninst_ret_ok:`
Eine mit Strichpunkten getrennte Liste von Rückgabewerten einer erfolgreichen Deinstallation.

3.3.3 Create Statement der Datenbank

```
CREATE TABLE client (
  sid          varchar(60) NOT NULL Primary Key,
  name        text NOT NULL,
  description  text NULL,
  registered   tinyint(1) NOT NULL DEFAULT '0'
);
CREATE TABLE client_gets_package (
  FK_client_sid  varchar(60) NOT NULL,
  FK_package_id  int(11) NOT NULL,
PRIMARY KEY (FK_client_sid , FK_package_id)
);
CREATE TABLE client_has_package (
  FK_client_sid  varchar(60) NOT NULL,
  FK_package_id  int(11) NOT NULL,
PRIMARY KEY (FK_client_sid , FK_package_id)
);
CREATE TABLE client_in_group (
  FK_client_sid  varchar(60) NOT NULL,
  FK_group_id    int(11) NOT NULL,
PRIMARY KEY (FK_client_sid , FK_group_id)
);
CREATE TABLE group (
  id            int(11) AUTO_INCREMENT NOT NULL Primary Key,
  name          text NOT NULL,
  description   text NULL
);
CREATE TABLE group_gets_package (
  FK_group_id    int(11) NOT NULL,
  FK_package_id  int(11) NOT NULL,
PRIMARY KEY (FK_group_id, FK_package_id)
);
CREATE TABLE package (
  id            int(11) AUTO_INCREMENT NOT NULL Primary Key,
  name          text NOT NULL,
  version       text NULL,
  description   text NULL,
  priority      int(11) NULL,
  fetch_url     text NULL,
  inst_exec     text NULL,
  inst_exec_args text NULL,
  uninst_exec   text NULL,
  uninst_exec_args text NULL,
  inst_ret_ok   text NULL,
  uninst_ret_ok text NULL
);
```

Auflistung 1 create statement database

Mit diesem Skript wurde die Datenbank `yasd` erzeugt.

3.4 Verzeichnisdienst

Ein Kann-Ziel der Diplomarbeit war eine sichere Authentifizierung. Um diese zu gewährleisten, wurde ein Verzeichnisdienst benötigt. Der Verzeichnisdienst sollte folgende Eigenschaften besitzen. Er sollte eine Schnittstelle zur Verfügung stellen, die es ermöglicht, zwischen Linux und Windows zu kommunizieren. Außerdem soll der Verzeichnisdienst frei erhältlich und relativ klein und leicht zu konfigurieren sein. Unter diesen Voraussetzungen fiel die Wahl auf OpenLDAP. LDAP (steht für „Lightweight Directory Access Protocol“ und der Name spricht schon für sich) war genau das Gesuchte, klein und kompakt. Außerdem erfüllt OpenLDAP auch alle anderen oben genannten Punkte. OpenLDAP fällt unter das Lizenzierungsmodell GPL.

3.4.1 Installation von LDAP

Wir entschieden uns für die Linux-Distribution „Fedora Core 5“. Nach der Installation von Fedora musste OpenLDAP eingerichtet werden. Dafür waren folgende Pakete notwendig: `openldap-clients-2.3.19-4`, `openldap-2.3.19-4`, `openldap-servers-2.3.19-4`.

Außerdem wird die Datei `samba.schema` benötigt. Diese kann man sich aus dem Internet herunterladen oder das Paket `samba` installieren. Falls die Installation gewählt wird, findet man diese Datei unter `<sambadirectory>/examples/LDAP/`.

Pakete werden durch den Befehl `yum install <package>` installiert, wobei statt `package` das gewünschte Paket zu wählen ist. Oben genannte Versionen sind die verwendeten Versionen. Mit dem Befehl `yum list | grep ldap` werden alle aktuellen LDAP Pakete, inklusive Versionen, angezeigt. Bei Bedarf können diese aktualisiert werden.

3.4.2 Konfiguration von LDAP

Nachdem jetzt alle benötigten Pakete installiert sind, kann das System nun konfiguriert werden. Zuerst wird der LDAP Server konfiguriert.

Um den LDAP Server gegen unbefugte Zugriffe abzusichern, muss dieser durch ein Passwort gesichert werden. Der Befehl `slappasswd` erzeugt dieses Passwort. Nach Eingabe und Bestätigung des Passwortes, wird der berechnete SHA-1 Hash des Passwortes ausgegeben, zum Beispiel `{SSHA}CnfYq+nmP0A/l2aDGbwLDsDF53G5B0ac`.

Dieses Passwort wird im Anschluss in der Konfigurationsdatei `/etc/openldap/slapd.conf` eingetragen. Die folgende Auflistung zeigt die notwendigen Änderungen:

```
include                /etc/openldap/schema/samba.schema

access to attrs=userPassword
    by self write
    by anonymous auth
    by * none
access to * by * none

suffix                 "dc=yasd,dc=at"
rootdn                 "cn=ldapadmin,dc=yasd,dc=at"

rootpw                 {SSHA}CnfYq+nmP0A/l2aDGbwLDsDF53G5B0ac

index sambaSID                eq
```

Auflistung 2 /etc/openldap/slapd.conf

Das `include` Statement dient zur Einbindung des samba-Schemas. Dieses Schema wird benötigt, um die `sambaSID` (Security Identifier) in LDAP speichern zu können. Wenn es noch kein passendes Schema gibt, muss das Schema selbst geschrieben werden. In diesem Fall gibt es jedoch das samba-Schema, welches ein Schema bereit stellt, das den gewünschten Anforderungen genau entspricht. Es hat ein Muss-Feld, nämlich die `sambaSID`. Genaueres wird weiter unten erläutert.

Bei dem nächsten Block handelt es sich um eine Access Control List (`acl`). Der Benutzer mit dem `userPassword` darf schreiben. Alle anonymen Anfragen werden gezwungen sich zu autorisieren. Ansonsten ist nichts erlaubt. Die letzte Zeile `access to * by * none` sorgt dafür, dass sonst keiner das Recht hat, Daten anzusehen oder zu verändern. Das heißt, allen anderen ist es nicht möglich, Daten abzufragen oder zu manipulieren.

Unter `suffix` versteht man den höchsten Pfad, in den man Einsicht hat. In dem Fall ist es möglich alles anzusehen, was unter dem Verzeichnis `yasd.at` liegt, da LDAP hierarchisch aufgebaut ist.

Mit `rootdn` wird der Administrator für LDAP bestimmt. Die Syntax sieht so aus, dass zuerst der „canonical name“ (`cn`) vom Administrator angegeben werden muss. Anschließend muss das `suffix` angegeben werden. In diesem Fall heißt der Administrator `ldapadmin`. Unter `rootpw` muss das Passwort des Administrators eingetragen werden. Dieses wurde vorher mit dem Befehl `slappasswd` erzeugt. Dieser Hash Wert muss nun an dieser Stelle eingetragen werden. Dabei muss man auf die Abstände achten, diese werden nämlich zu dem Hashwert dazu interpretiert. Daher ist es sehr empfehlenswert, nur Tabulatoreinrückungen zu verwenden.

`index sambaSID eq` wird nur aus Performancegründen benötigt. Da Indizes für die `sambaSID` angelegt werden, werden Suchanfragen viel schneller gehandhabt.

Wie schon weiter oben erwähnt, wird das samba Schema benötigt, um die `sambaSID` in das LDAP Verzeichnis zu schreiben. Folgendes spezielle Schema der Datei `samba.schema` ist für diesen LDAP Server vonnöten:

```
attributetype ( 1.3.6.1.4.1.7165.2.1.20 NAME 'sambaSID'  
  DESC 'Security ID'  
  EQUALITY caseIgnoreIA5Match  
  SUBSTR caseExactIA5SubstringsMatch  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{64} SINGLE-VALUE )
```

Auflistung 3 Auszug `samba.schema`

`attributetype` wird von einer hierarchischen Zahl definiert, welche dann den Namen des Schemas bekommt, nämlich `sambaSID`. `DESC` steht für Description und erklärt nur das Schema mit wenigen Worten. Mit `EQUALITY` und `SUBSTR` werden die Parameter für Suchergebnisse definiert, wobei `EQUALITY` für den ganzen Suchstring gilt und `SUBSTR` nur für einen Teilstring.

- `caseIgnoreIA5Match`: Groß- und Kleinschreibung wird nicht beachtet
- `caseExactIA5SubstringsMatch`: Groß- und Kleinschreibung wird beachtet

`1.3.6.1.4.1.1466.115.121.1.26{64}` ist die Datentypdefinition, auch OID genannt. Die `{64}` steht für die Länge des Strings, nämlich 64 Bit. `SINGLE-VALUE` bedeutet, dass nur ein Wert erlaubt ist.

3.4.3 Erzeugung der LDAP Struktur

Nun kann die eigentliche Erzeugung des Verzeichnisses beginnen. Dafür werden Dateien, die dem LDAP Directory Interchange Format (LDIF) entsprechen, benötigt. Zuerst einmal muss die Hauptdomain erzeugt werden.

```
dn: dc=yasd,dc=at
objectClass: dcObject
objectClass: organization
o: Yasd
dc: yasd
```

Auflistung 4 Authentication1.ldif

`dn` steht für „distinguished name“, mit dem jedes Objekt identifiziert werden kann. Jeden `dn` kann es nur einmal in der ganzen Verzeichnisstruktur geben. `objectClass` definiert, um welchen Typ es sich bei dem Objekt handelt. Dieses Objekt leitet sich von `dcObject` und `organization` ab.

```
objectclass ( 1.3.6.1.4.1.1466.344 NAME 'dcObject'
  DESC 'RFC2247: domain component object'
  SUP top AUXILIARY MUST dc )
```

Auflistung 5 dcObject

```
objectClass ( OpenLDAPobjectClass:3
  NAME 'OpenLDAPorg'
  DESC 'OpenLDAP Organizational Object'
  SUP organization
  MAY ( buildingName $ displayName $ labeledURI ) )
```

Auflistung 6 organization

Außerdem wird jedes Objekt von der Klasse `top` abgeleitet. In `Authentication1.ldif` wird ein neues `dcObject` in `yasd.at` erzeugt. Es ist das Objekt `yasd` in der Domain `yasd`.

```
dn: cn=ldapadmin,dc=yasd,dc=at
objectClass: organizationalRole
cn: ldapadmin
description: "LDAP Manager"
```

Auflistung 7 Authentication2.ldif

Mit diesem Skript wird der Administrator für das Verzeichnis `yasd.at` erzeugt. Das Objekt leitet sich von `organizationalRole` ab.

```
dn: sambaSID=xx,dc=yasd,dc=at
objectClass: sambaSidEntry
sambaSID: xx
```

Auflistung 8 Authentication3.ldif

Der distinguished name besteht aus der `sambaSID` und dem Suffix. Das Objekt leitet sich von der Klasse `sambaSidEntry` ab. Durch diese Klasse kann jeder PC eindeutig in das Verzeichnis geschrieben werden. Dies ermöglicht eine sichere Authentifizierung.

3.5 Webinterface

Das Webinterface dient der Verwaltung der Daten aus der Datenbank. Es benötigt relativ viel Aufwand und Vorkenntnisse, die Daten der Datenbank durch Datenbankmanipulationen zu verändern. Aus diesem Grund gibt es ein Webinterface, welches alle nötigen Datenmanipulationen grafisch zur Verfügung

stellt. Anders ausgedrückt, das Webinterface ist die zentrale Verwaltung der Softwareverteilung. Im Webinterface ist es möglich, neue Gruppen anzulegen, Clients diesen Gruppen zuzuweisen, Pakete mit Clients und Gruppen zu verbinden, sowie alle anderen nötigen Einstellungen zu treffen. Auf die Optionen wird in der Folge genauer eingegangen. Das Webinterface ist auf Userfreundlichkeit ausgelegt, das heißt es ist intuitiv zu bedienen.

3.5.1 Gestaltung des Webinterface

Nachdem die Funktionen des Webinterface ausgearbeitet waren, kam die Frage, mit welchen „Werkzeugen“, im Sinne von Programmiersprachen, die notwendigen Funktionen implementiert werden können. Ein wichtiger Aspekt war, dass der Webserver keine zusätzlichen Pakete benötigen sollte. Aus diesem Grund konnten zahlreiche strategisch günstige Programmiersprachen, wie Perl und Python, nicht verwendet werden.

Unter diesen Voraussetzungen fiel die Wahl auf folgende Programmiersprachen:

- Hyper Text Markup Language (HTML) ist noch immer die Programmiersprache, an der man im Bereich WEB nicht vorbei kommt. Es war von Beginn an klar, dass der Hauptteil des Webinterface in HTML verfasst sein wird.
- Hypertext Preprocessor (PHP): Nachdem es mit HTML nicht möglich ist, mit einer Datenbank zu kommunizieren, wurde dieser Teil mit PHP realisiert. Außerdem ist es mit PHP möglich, die Seite dynamisch zu gestalten und auf Eingaben des Benutzers zu reagieren.
- Cascading Style Sheet (CSS) wird im Zusammenhang mit HTML verwendet. Dank CSS ist es möglich, die Formatierung des gesamten Webinterface mit geringem Aufwand zu verändern. Eine Veränderung in der CSS Datei bewirkt die Veränderung auf jeder Seite des Webinterfaces.

- Javascript (JS) wurde ausgewählt um die Navigation des Webinterface dynamisch zu gestalten. Da HTML die Einschränkung hat, nur statische Objekte bereit zu stellen, wurde hier mit Javascript ein Ausweg gefunden.

3.5.2 Sicherheit

Wie schon erwähnt, erfüllt das Webinterface eine zentrale Aufgabe, daher ist es wichtig, dass maximale Sicherheit gewährleistet ist. Es soll ausschließlich dem Administrator vorbehalten sein, Daten anzusehen und zu verändern. Aus diesem Grund ist das ganze Webinterface mit einem Passwortschutz versehen. Es ist nicht möglich, irgendwelche Daten anzusehen oder gar zu verändern, ohne angemeldet sein.

```
<?php
  if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    session_start();

    $username = $_POST['username'];
    $passwort = $_POST['passwort'];
    $hostname = $_SERVER['HTTP_HOST'];
    $path = dirname($_SERVER['PHP_SELF']);

    // Benutzername und Passwort werden überprüft
    if ($username == 'yasd' && $passwort == 'yasdyasd') {
      $_SESSION['angemeldet'] = true;

      // Weiterleitung zur geschützten Startseite
      if ($_SERVER['SERVER_PROTOCOL'] == 'HTTP/1.1') {
        if (php_sapi_name() == 'cgi') {
          header('Status: 303 See Other');
        }
        else {
          header('HTTP/1.1 303 See Other');
        }
      }

      header('Location: http://'.$hostname.($path == '/' ? '' :
$path).'/startseite.php');
      exit;
    }
  }
?>
```

Auflistung 9 main_frame.inc.php

Dieses Skript behandelt die Passwortabfrage. Wenn der Benutzer seine Angaben, Benutzername und Passwort, eingegeben hat, werden diese Daten an dieses Skript geschickt. Die gesendeten Daten werden dann mit den richtigen Daten abgeglichen. Sind sie richtig, so beginnt eine neue „Session“ und der Benutzer gilt als angemeldet. Außerdem wird er automatisch zur eigentlichen Startseite des Webinterface weitergeleitet. Sind die Angaben inkorrekt, wird der Benutzer auf diese Seite neu verbunden. Er hat nun die Möglichkeit seine Angaben erneut einzugeben.

```
<?php
    session_start();

    $hostname = $_SERVER['HTTP_HOST'];
    $path = dirname($_SERVER['PHP_SELF']);

    if (!isset($_SESSION['angemeldet']) || !$_SESSION['angemeldet']) {
        header('Location: http://'.$hostname.($path == '/' ? '' :
$path).'/main_frame.inc.php');
        exit;
    }
?>
```

Auflistung 10 auth.php

Dieses Skript wird von jeder Seite des Webinterface inkludiert. Das heißt, jede PHP-Datei des Webinterface beginnt mit folgendem Code:

```
<?php include('auth.php'); ?>
```

Das ist gleichbedeutend mit einem Einfügen des Skriptes `auth.php` in jede Datei. Das Skript selbst überprüft, ob der Benutzer bereits angemeldet ist oder nicht. Ist er angemeldet, wird die gewünschte Seite aufgerufen. Ist der Benutzer jedoch nicht angemeldet, wird er zur Login-Seite weitergeleitet.

Wenn man seine Sitzung beenden will, muss man links auf den Knopf „Logout“ klicken. Dadurch wird die Datei `logout.php` aufgerufen, welche folgende Auflistung zeigt:

```
<?php
    session_start();
    session_destroy();

    $hostname = $_SERVER['HTTP_HOST'];
    $path = dirname($_SERVER['PHP_SELF']);

    header('Location: http://'.$hostname.($path == '/' ? '' :
$path).'/main_frame.inc.php');
?>
```

Auflistung 11 logout.php

Mit `session_start()` wird nicht nur eine neue Session erzeugt, sondern auch eine bereits bestehende weitergeführt. Anschließend wird durch `session_destroy()` die aktuelle Session gelöscht. Das heißt alle relevanten Variablen, die mit dieser Session zu tun haben, werden gelöscht. In `$hostname` steht der Hostname des Servers, zum Beispiel localhost, wenn die Abfrage auf dem Server stattfindet, auf dem auch der Webserver läuft. In der Variable `$path` steht der relative Pfad der aktuellen PHP Datei. Mit der abschließenden Zeile `header('Location: http://'.$hostname.($path == '/' ? '' : $path).'/main_frame.inc.php')` wird man zurück zur Hauptseite geleitet, da man keine gültige Session mehr besitzt.

3.5.3 Kommunikation mit der Datenbank

Der Hauptbestandteil des Webinterface ist, die Inhalte der Datenbank leicht verwaltbar darzustellen. Um dies zu gewährleisten, muss zuerst ein Datenbankzugriff ermöglicht werden. Der Datenbankzugriff erfolgt über PHP in folgendem Skript:

```
<?php
    $database = "yasd";
    mysql_connect ("localhost", "root", "yasd") or die ("Keine Verbindung zum
Server");
    mysql_select_db ($database) or die ("Keine Verbindung zur Datenbank");
?>
```

Auflistung 12 db.php

Zuerst wird der Variablen `$database` der Wert `"yasd"` zugeordnet. Anschließend werden die Parameter für den Datenbankserver übergeben. Dies geschieht mit

- Host: `localhost`
- Datenbankuser: `root`
- Datenbankuserpasswort: `yasd`

War der Aufbau nicht erfolgreich, erscheint die Meldung „Keine Verbindung zum Server“. Da auf dem Datenbankserver mehrere Datenbanken installiert sein können, muss man jetzt die richtige Datenbank auswählen. Dies geschieht mit `mysql_select_db ($database)`. Auch hier wird bei einem Fehler die Meldung „Keine Verbindung zur Datenbank“ ausgegeben.

Diese Datei wird in jeder Datei, die Datenbankzugriffe tätigt, am Anfang inkludiert. Am Ende der Datei muss der Datenbankzugriff wieder „sauber“ beendet werden.

Das wird mit `mysql_close()` realisiert.

Der Vorteil der zentralen Definition der Datenbank ist die zentrale Verwaltung. Egal, ob der Server gewechselt wird, der Username, das Passwort oder die Datenbank umbenannt wird, alles muss nur in einer Datei geändert werden und ist automatisch überall richtig implementiert.

3.5.4 Navigation

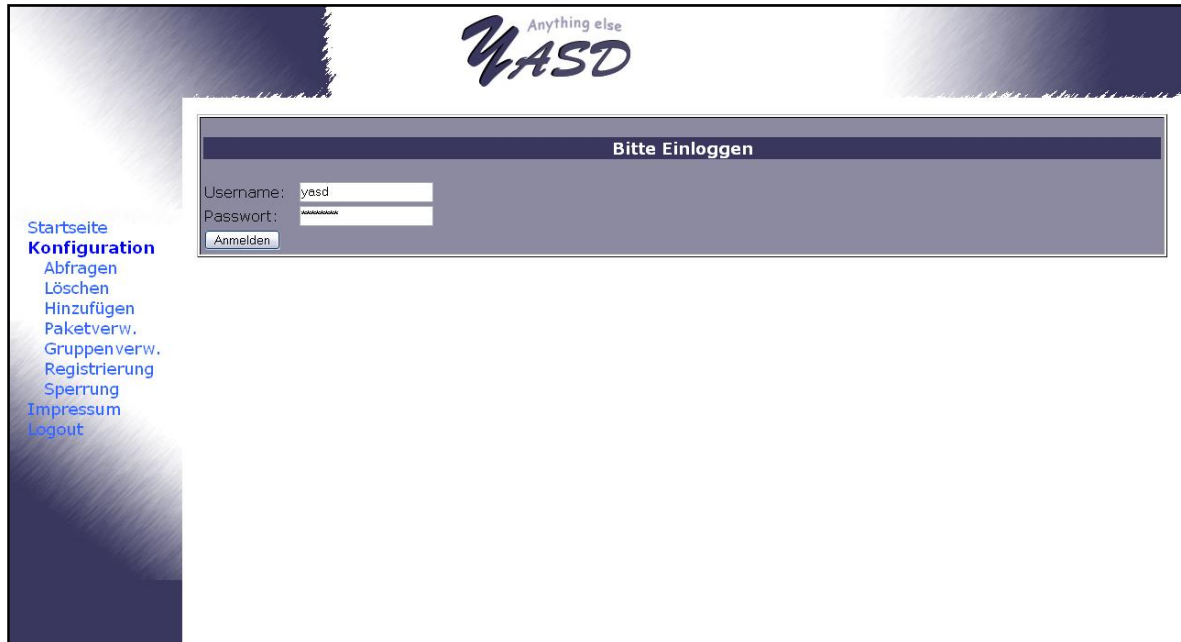


Abbildung 4 yasd Webinterface

Die Navigation befindet sich links und ist in zwei wesentliche Punkte aufgeteilt, in Allgemeines und in die Konfiguration. Während im allgemeinen Teil Erklärungen des Webinterface und rechtliche Dinge, wie das Impressum, vorhanden sind, befasst sich die Konfiguration rein mit der Verwaltung der Datenbank. Weiter unten folgt eine Erklärung der Konfiguration.

Zuvor noch eine kurze Erklärung zur Datenbank: Ein Datensatz in einer Tabelle wird auch Zeile genannt. Wenn von den 3 Haupttabellen gesprochen wird, sind die Tabellen `client`, `group` und `package` gemeint.

- Abfragen:
Hier ist es möglich, die Daten der 3 Haupttabellen anzusehen.
- Löschen:
Unter Löschen bekommt man eine Auflistung aller Daten der 3

Haupttabellen. Die Checkboxes ermöglichen einem, die Daten, die gelöscht werden sollen, auszuwählen. Drückt man unten auf den Button „Löschen“, werden diese Zeilen gelöscht.

- Hinzufügen:

Hier bekommt man zuerst die Wahl, welchen Typ (Tabelle) der neue Datensatz haben soll. Nach der Auswahl muss man nur noch die entsprechenden Felder ausfüllen und mit „Erstellen“ wird die neue Zeile erzeugt.

- Paketverwaltung:

Sie dient dem Verbinden und Trennen von Daten. Auf den ersten Blick sieht man alle Clients und Gruppen. Man kann nun eine Gruppe oder einen Client auswählen und eine „Neue Verknüpfung“ erstellen. Nun werden alle Pakete angezeigt, die zurzeit nicht mit der Gruppe / dem Client verbunden sind. Hier wählt man alle Pakete aus, die man verknüpft haben will und drückt auf „Verknüpfung“. Wenn das erfolgreich war, bekommt man die Message „Datensatz erfolgreich verknüpft“. Um Datensätze voneinander zu trennen, wählt man wieder eine Gruppe / einen Client und klickt auf „Verknüpfung aufheben“. Nun werden alle Pakete angezeigt, die gerade mit dieser Gruppe / diesem Client verbunden sind. Hier wählt man jetzt alle Pakete aus, die davon wieder getrennt werden sollen und drückt anschließend auf „Deinstallieren“. War die Deinstallation erfolgreich, bekommt man die Meldung „Datensatz erfolgreich deinstalliert“.

- Gruppenverwaltung:

In der Gruppenverwaltung kann man Gruppen Clients zuweisen oder entfernen. Will man einer Gruppe einen neuen Client hinzufügen, so wählt man diese Gruppe aus und drückt auf „Client zuweisen“. Man bekommt jetzt alle Clients angezeigt, die aktuell nicht der Gruppe zugewiesen sind.

Nun kann man alle Clients, die man der Gruppe zuweisen will, auswählen und auf „Zuweisen“ klicken. Die Meldung „Client wurde der Gruppe hinzugewiesen“ bestätigt den Erfolg der Zuweisung. Außerdem kann man Clients aus Gruppen wieder entfernen. Dafür wählt man die gewünschte Gruppe aus und drückt auf „Client entfernen“. Nun sieht man alle Clients, die der Gruppe zugewiesen sind. Man kann jetzt die gewünschten Clients auswählen und anschließend auf „Entfernen“ drücken. War die Entfernung erfolgreich, bekommt man die Statusmeldung „Client wurde aus der Gruppe entfernt“.

- Registrierung:

Um überhaupt Pakete empfangen zu können, muss der Client registriert sein. Die Registrierung passiert mit einem Klick. Man wählt alle Clients aus, die man registrieren will und drückt anschließend auf „Registrieren“. War man erfolgreich, bekommt man die Meldung „Der Client wurde erfolgreich registriert“.

- Client sperren:

Das Gegenstück zur Registrierung. Hier können Clients wieder gesperrt werden. Die Prozedur ist der Registrierung ähnlich, der einzige Unterschied ist der Knopf „Client sperren“ und die Meldung „Der Client wurde erfolgreich gesperrt“ bei erfolgreicher Sperrung.

Der allgemeine Part wird in folgende Punkte unterteilt:

- Startseite:

Das ist die Seite, zu der man nach dem Login weitergeleitet wird. Hier findet man eine kurze Erklärung zu allen Punkten der Navigation um sich besser zurecht zu finden.

- Konfiguration:

Wenn man die Konfiguration anwählt, wird diese sichtbar. Klickt man ein

zweites Mal darauf, verschwindet sie wieder. In der Konfiguration stehen die weiter oben erklärten Punkte.

- Impressum:

Das Impressum muss aus rechtlichen Gründen auf jeder Homepage vorhanden sein, daher ist es auch in diesem Webinterface integriert. Das Impressum enthält Name, Anschrift und Kontakt jedes Diplomarbeitverfassers.

- Logout:

Zum Beenden der Session dient der Knopf „Logout“. Nach dem Drücken von „Logout“ werden alle relevanten Variablen dieser Sitzung gelöscht. Es ist anschließend nicht mehr möglich, die geschützten Seiten anzusehen.

3.5.5 Style des Webinterface

Die Formatierung des Webinterface ist ein wichtiger Aspekt um die Verwaltung möglichst leicht zu gestalten. Es erleichtert das Lesen, wenn Schriftart, Schriftfarbe, Schriftgröße, Hintergrund usw. zueinander passen. All das lässt sich mit HTML realisieren. Der einzige große Nachteil ist, dass eine Änderung auf allen Seiten erfolgen muss. Das heißt, dass kleine Veränderungen sehr aufwändig sind.

Mit Cascading Style Sheets (CSS) gibt es dieses Problem nicht. Hier wird ein Muster für alle Seiten definiert. Dank dieser Hilfe ist es leicht, die ganze Formatierung mit wenig Aufwand zu ändern. Außerdem wird einem viel Tipparbeit erspart, da eine einzige Definition reicht.

Die Implementierung der CSS Datei auf einer Seite kann auf zwei verschiedenen Wegen stattfinden. Entweder es gibt eine Datei, in der CSS definiert ist und implementiert wird, oder man schreibt den CSS-Code direkt auf die Seite. Es wurde die Implementierung durch separate Datei gewählt, da nur so eine sofortige Konvergenz gewährleistet ist. CSS direkt auf einer HTML Seite zu

implementieren, wird in der Praxis bei nur einmalig benötigtem CSS-Code verwendet. Da dies jedoch in diesem Webinterface nicht der Fall ist, wurde durchgehend die Implementierung in einer extra Datei gewählt.

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Löschen</title>
<link rel="stylesheet" href="../styles/style1.css" type="text/css" />
</head>
```

Auflistung 13 CSS Einbindung

Die vorangehende Auflistung zeigt die Einbindung der CSS Datei. Das Einbinden geschieht im `head` Bereich. Die relevante Zeile der Einbindung ist `<link rel="stylesheet" href="../styles/style1.css" type="text/css" />`.

Das Attribut `rel` gibt die Art der Sprache an. Da auch andere Dateien wie zum Beispiel Javascript ausgelagert werden können, muss hier der Name der Sprache angegeben werden. In diesem Fall ist dies `stylesheet`. `href` gibt den Pfad der CSS Datei an. `type` gibt den Multipurpose Internet Mail Extension Typ (MIME-Typ) an. Im beschriebenen Fall muss immer `text/css` gewählt werden.

```
body {
    font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
    margin-left: 0px;
    margin-top: 0px;
    margin-right: 0px;
    margin-bottom: 0px;
}

td {
    font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
}

th {
    font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
}

td.body_ {
    background-color: #8c8aa0;
    font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
}

td.head{
```

```
height: 18px;
font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
font-size: 18px;
font-weight: bold;
color: #FFFFFF;
text-align: center;
background-color: #39375e;
}
```

Auflistung 14 style1.css

Um CSS zu verstehen, muss man Grundkenntnisse vom Aufbau von HTML haben: HTML basiert auf Tags. Es gibt von diesen Tags im Großen und Ganzen nur zwei verschiedene Gruppen. Gruppe 1 hat einen Anfangstag und einen Endtag. Gruppe 2 besteht nur aus einem Tag. Tags geben an, wie der folgende Text interpretiert werden soll. Zum Beispiel `` bedeutet, dass der Text bis zum abschließenden `` fett gedruckt wird.

Im `body` Teil steht `font-family`, welches die Schriftart definiert. Die Schriftarten werden von links beginnend gewählt. Wenn die gewünschte Schrift nicht vorhanden ist, wird die nächste genommen. Anschließend kommen die `margin` Befehle. Diese dienen der Ausrichtung der Seite. Die Seiten in HTML beginnen nicht genau links oben sondern mit 1-2 Pixel nach rechts beziehungsweise unten versetzt. Um diesen „Fehler“ zu beheben und die Ausrichtung der folgenden Elemente zu gewährleisten, ist dieser Befehl essentiell. `td` und `th` sind unter anderem die Kennzeichnung für Tabellen. Auch hier wird die Schrift bestimmt um diese einheitlich auf allen Seiten zu gewährleisten.

In CSS kann man auch eigene Klassen definieren, dank der man Ausnahmen in das Muster fügen kann. Zum Beispiel gibt es fünf Tabellenelemente die eine andere Hintergrundfarbe benötigen. So ein Problem wird mit `td.body_` gelöst. Wenn man jetzt auf den Seiten `<td style="body_">` schreibt, wird nicht die normale `td` Formatierung gewählt sondern die `td.body_` Formatierung. `background-color`

definiert die Hintergrundfarbe des Tabellenelements. Außerdem wird wieder die Schriftart angegeben.

Man kann mehrere der oben erwähnten Klassen erzeugen. Je nach Namen wird eine andere gewählt. So gibt es noch eine dritte `td` Klasse. Die Schlüsselwörter stehen für folgende Formatierungen:

- `height`: Zellenhöhe
- `font-family`: Schriftart
- `font-size`: Schriftgröße
- `font-weight`: Schriftdarstellung (kursiv, fett, unterstrichen)
- `color`: Schriftfarbe
- `text-align`: Textausrichtung (linksbündig, mittig, rechtsbündig)
- `background-color`: Hintergrundfarbe.

Es sei noch erwähnt, dass die Schlüsselwörter nicht immer die gleichen wie in HTML, sondern oftmals verschieden sind.

3.6 Pakete

Der folgende Abschnitt beschäftigt sich mit den verschiedenen Typen von Paketen, die vom Server verteilt werden, die man in drei Bereiche einteilen kann: Virens Scanner Updates, Office Paket und Windows Updates. Da wir am Ende unserer Diplomarbeit noch Zeit hatten, beschlossen wir auch noch zusätzlich andere Pakete zu verteilen. Die Pakete werden auf dem Server bereitgestellt und können danach von Clients bezogen werden. Die einzelnen Pakettypen werden nun näher beschrieben.

3.6.1 Virens Scanner Updates

Eines unserer Muss-Ziele war es, Updates für einen Virens Scanner zu verteilen. Wir haben lange überlegt, welchen Virens Scanner wir dazu verwenden werden. Schließlich haben wir uns für das Programm AntiVir von Avira entschieden. Man könnte natürlich auch mit anderen Virens Scannern verwenden, das haben wir aber nicht gemacht. yasd ist aber diesbezüglich leicht erweiterbar. Die Auswahl von AntiVir hatte mehrere Gründe:

- **Gratis**
AntiVir ist für Privatanwender frei verfügbar und weist keine Einschränkungen auf.
- **Verbreitung**
AntiVir ist in Privathaushalten eines der am häufigsten eingesetzten Virens Scannerprogramme. Aber auch in vielen Schulen wird es eingesetzt. Zum Beispiel ist es in unserer Schule auf fast allen Windows PCs bereits installiert.
- **Erkennungsleistung**
AntiVir ist zwar gratis, trotzdem weist es eine sehr gute Erkennungsleistung bei Viren auf. Auch die Reaktionszeiten für neue Updates sind beachtlich. Diesen Punkt kann man in regelmäßigen Abständen in Computerzeitschriften nachlesen, in deren Tests AntiVir immer auf einem der vordersten Plätze zu finden ist.

Nachdem klar war, dass wir AntiVir verwenden werden, habe ich im Internet gesucht, ob es überhaupt Lösungen für einen internen Updateserver gibt. Nach langer Suche habe ich festgestellt, dass es dafür noch keine funktionierenden Programme gibt und ich eine eigene Lösung dafür finden muss. Zuerst habe ich mit einem Netzwerkniffer und durch Ausprobieren herausgefunden, welche

Updates von AntiVir benötigt und heruntergeladen werden. Danach hat sich für mich die Frage gestellt, wie ich AntiVir dazu bringe, sich auf einen anderen Updateserver als die offiziellen zu verbinden. Anschließend musste ich ein Skript schreiben, das vollautomatisch die neuesten Updates herunterlädt und danach für yasd-Clients verfügbar macht. Dieses Skript könnte auch für Linux portiert werden.

Skript

Nachfolgend wird das Update Skript aufgelistet und anschließend erklärt.

```
@echo off
echo Download wird gestartet
echo idx
wget -N -t 5 -P      ./upd/idx -i links/idx.links
echo vdf
wget -N -t 5 -P      ./upd/vdf -i links/vdf.links
echo engine-nt-de
wget -N -t 5 -P      ./upd/engine/nt/de -i links/engine_nt_de.links
echo engine-nt
wget -N -t 5 -P      ./upd/engine/nt/ -i links/engine_nt.links
echo engine
wget -N -t 5 -P      ./upd/engine -i links/engine.links
echo basic-nt
wget -N -t 5 -P      ./upd/winwks/de/basic-nt -i links/basic_nt.links
echo classic-nt
wget -N -t 5 -P      ./upd/winwks/de/classic-nt -i links/classic_nt.links
echo basic-nt-2k
wget -N -t 5 -P      ./upd/winwks/de/basic-nt/2k -i links/basic_nt_2k.links
echo basic-nt-nt
wget -N -t 5 -P      ./upd/winwks/de/basic-nt/nt -i links/basic_nt_nt.links
echo basic-nt-xp
wget -N -t 5 -P      ./upd/winwks/de/basic-nt/xp -i links/basic_nt_xp.links
```

Auflistung 15 update_skript.cmd

```
http://dl3.avgate.net/upd/vdf/antivir0.vdf.gz
http://dl3.avgate.net/upd/vdf/antivir1.vdf.gz
http://dl3.avgate.net/upd/vdf/antivir2.vdf.gz
http://dl3.avgate.net/upd/vdf/antivir3.vdf.gz
```

Auflistung 16 vdf.links

Das Skript heißt `update_skript.cmd` und kann durch einen einfachen Doppelklick gestartet werden. Als Voraussetzung muss das Tool wget im selben Ordner wie das Skript abgelegt werden, damit die Updates heruntergeladen werden können.

Die Downloadlinks sind zur Wahrung der Übersichtlichkeit in andere Dateien ausgelagert worden (zB: `vdf.links`). Die Dateien werden in den Ordner `upd` heruntergeladen. Außerdem wird die Verzeichnisstruktur des offiziellen Servers beibehalten. Dieser Ordner wird anschließend auf dem Webserver des yasd-Servers freigegeben (`http://yasd/upd`). Updates werden nur heruntergeladen, wenn sie auf dem Server noch nicht vorhanden sind oder nur in einer veralteten Version vorliegen (`-N` bewirkt das). Damit läuft das Skript auch schneller. Das Updateskript wurde jetzt über ein halbes Jahr getestet. Es wird noch immer die ursprüngliche Version verwendet und es waren keine Änderungen daran notwendig.

Umleiten der Updateanfrage

Nachdem der Updater fertiggestellt war, stellte sich die Frage, wie man AntiVir dazu bringen konnte, auf den yasd Server statt auf die offiziellen Server zuzugreifen.

Bei manchen Virenschannern kann man das über die Einstellungen oder in Konfigurationsdateien ändern, doch AntiVir bietet diese Möglichkeit nicht. Deshalb wurde zu einem Trick gegriffen. Unter Windows gibt es wie unter Linux eine `hosts` Datei. Diese findet man im Windowsverzeichnis im Ordner `%windir%\system32\drivers\etc`. In dieser kann man die Namensauflösung statisch eintragen. Hier werden alle AntiVir Servernamen eingetragen und die IP des yasd-Servers umgeleitet. Der Benutzer merkt davon nichts und die Updates werden nun vom yasd-Server bezogen. AntiVir greift zufällig auf einen der neun Downloadserver zu, daher muss man alle in die `hosts` Datei eintragen.

```
127.0.0.1      localhost
#Anmerkung: 192.168.1.100 steht für die IP des yasd Servers
192.168.1.100 dl.antivir.de
192.168.1.100 dl1.avgate.net
192.168.1.100 dl2.avgate.net
192.168.1.100 dl3.avgate.net
```

192.168.1.100	dl4.avgate.net
192.168.1.100	dl5.avgate.net
192.168.1.100	dl6.avgate.net
192.168.1.100	dl7.avgate.net
192.168.1.100	dl8.avgate.net

Auflistung 17 %windir%\system32\drivers\etc\hosts

Damit dieser Vorgang einfacher wird, haben wir ein Skript erstellt, das den Server automatisch einträgt. Dieses Skript ruft man ganz einfach mit `antivir_hosts.exe install <ip des yasd servers>` auf.

Startmöglichkeiten des Skriptes

Unser Skript kann auf zwei Arten aufgerufen werden: manuell oder automatisch.

Die manuelle Methode kann man wählen, wenn man die Updates nur in unregelmäßigen Abständen durchführt und kontrollieren will, wann ein Update erfolgt.

Für die meisten Anwendungszwecke wird man die automatische Methode wählen. Dabei kann man selbst festlegen in welchem Intervall man das Skript starten will.

Dazu öffnet man den Taskplaner: `Start -> Programme -> Zubehör -> Systemprogramme -> Geplante Tasks`.

Hier klickt man auf „Geplanten Task hinzufügen“, danach führt ein Assistent durch den weiteren Einrichtungsvorgang. Als auszuführendes Programm gibt man dann unser AntiVir Update Skript an. Nach Abschluss des Vorganges ist das automatische Update konfiguriert und das Skript wird in den festgelegten Intervallen nach Aktualisierungen suchen.

Vorteile unserer Implementierung

Unser AntiVir Updater weist einige wichtige Vorteile gegenüber einem normalen Update auf.

- Geschwindigkeit

Die Updates werden nicht aus dem Internet bezogen, sondern werden im LAN bereitgestellt. Daher ist die Übertragungszeit sehr gering und auch größere Updates können in kurzer Zeit an Clients geschickt werden. Damit kann man in sehr kurzer Zeit veraltete AntiVir Installationen auf den neuesten Stand bringen.

- Werbebanner

Durch unsere Lösung wird das Werbefenster nach einem Update nicht angezeigt. Das ist sehr gut, da man es sonst wegklicken müsste und viele Benutzer durch dieses Fenster gestört oder irritiert werden.

- Downloadkosten

Ein weiterer wichtiger Punkt ist, dass die Downloadkosten gesenkt werden, da die Updates für AntiVir nur einmal heruntergeladen werden und danach für beliebig viele Clients zur Verfügung stehen. Schon bei zwei Clients kann man hier die Downloadmenge verkleinern. Bei mehreren Clients ist dieses Einsparungspotenzial noch viel größer.

- Unterstützung für mehrere Plattformen

Unsere Updatelösung kann nicht nur AntiVir auf Windows XP Clients versorgen, sondern auf allen von AntiVir unterstützten Windows Versionen. Es gibt durch unsere Lösung keine Einschränkung gegenüber der offiziellen Updatemöglichkeit.

3.6.2 Office

Ein weiteres Ziel war ein beliebiges Office Paket zu verteilen. Wir haben uns nach kurzem Überlegen für Microsoft Office 2003 entschieden. Wir wollten außerdem, dass man nicht nur das ganze Office verteilt, sondern es sollten auch nur Teile

davon verteilt werden (zB: Client A bekommt nur Excel). Warum wir uns für Microsoft Office 2003 entschieden haben:

- Verbreitung

Microsoft Office 2003 ist weltweit das am häufigsten eingesetzte Office Paket. Es wird laufend erweitert und Sicherheitslücken werden schnell behoben und durch Updates geschlossen.

- Dokumentation

Office 2003 ist sehr gut dokumentiert und wird in vielen Firmen über das Netzwerk verteilt. Daher gibt es viele Tools von Microsoft und anderen Anbietern, die Office für eine Softwareverteilung vorbereiten.

Bevor eine Verteilung von Office 2003 über das Netzwerk durchgeführt werden kann, müssen ein paar Vorbereitungen getroffen werden. Dazu müssen die Inhalte der Installations-CD auf die Festplatte kopiert und ein paar Einstellungen für die Verteilung vorgenommen werden. Anschließend müssen diese Dateien am Server für die Clients verfügbar gemacht werden. Im Weiteren wird diese Vorgehensweise näher beschrieben.

Office kopieren

Im ersten Schritt wird Office von der Installations-CD auf die Festplatte kopiert. Dazu führt man das Office Setup in einem speziellen „Vorbereitungsmodus“ aus. Man öffnet eine Kommandozeile oder „Ausführen“ und startet das Setup von der Installations-CD mit folgendem Parameter: `setup.exe /a`.

Dadurch kommt man in einen Modus, der Office speziell für die Verteilung einrichtet. Zuerst muss man den Namen der Organisation angeben. Außerdem gibt man den CD-Key und den gewünschten lokalen Speicherort von Office an. In diesen Ordner wird die CD extrahiert, daher sollte dieser ca. 1 GB Platz bieten.

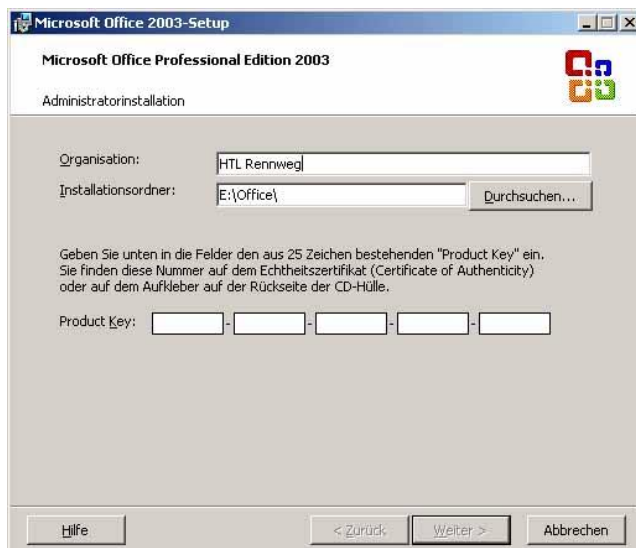


Abbildung 5 Vorbereiten von Office für die Verteilung

Im nächsten Fenster muss man noch den Endbenutzer-Lizenzvertrag akzeptieren. Danach werden die Dateien auf die Festplatte kopiert, die Dauer hängt von der Geschwindigkeit des Laufwerkes und des PCs ab. Wenn das Kopieren erfolgreich war, erscheint eine Meldung und man kann mit dem nächsten Schritt fortfahren.

Service Pack integrieren

Office 2003 ist schon ein paar Jahre auf dem Markt und Microsoft hat schon unzählige Fehler und Lücken darin ausgebessert. Eine Zusammenfassung der Updates wird in speziellen Service Packs angeboten. Damit man das verteilte Office nicht sofort mit diesen SPs versorgen muss, kann man diese in das nun lokal vorhandene Office integrieren. Zur Zeit der Diplomarbeit war das Service Pack 2 das aktuellste. Man kann es kostenlos von der Microsoft Homepage beziehen (<<http://www.microsoft.com/downloads/details.aspx?FamilyID=57e27a97-2db6-4654-9db6-ec7d5b4dd867&DisplayLang=de>>). Die Datei hat zirka 100 MB. In der Zwischenzeit sind weitere Updates erschienen, auch diese kann man separat in die Installation integrieren. Dabei ist die Vorgehensweise die gleiche.

Zuerst muss man das SP extrahieren. Das geschieht mit folgendem Befehl, wobei man den Zielordner nach dem Parameter `/T` angeben muss:

```
Office2003SP2-KB887616-Client-DEU.exe /C /T:"D:\OfficeSP2"
```

Nun muss man noch den Lizenzbestimmungen zustimmen, danach wird das Service Pack in den Ordner `D:\OfficeSP2` entpackt. Wenn dieser Vorgang erfolgreich war, erscheint eine Meldung.

Nun muss man das Service Pack für eine lokale Integration vorbereiten. Dazu öffnet man die Datei `ohotfix.ini` im Ordner des entpackten SPs mit einem beliebigen Texteditor. Hier sucht man den Eintrag `IsNormalUpdate=1` und ändert die „1“ in eine „0“. Die Zeile lautet nun `IsNormalUpdate=0`. Anschließend speichert und schließt man die Datei. Nun führt man die Datei `ohotfix.exe` aus. Im erscheinenden Fenster muss man nun in den extrahierten Office Ordner wechseln und die Datei `OWC11.MSI` auswählen. Anschließend klickt man auf Öffnen und nach einer kurzen Installation sollte eine Erfolgsmeldung erscheinen. Nun öffnet man ein weiteres Mal die Datei `ohotfix.exe` und wählt diesmal die Datei `PR011.MSI` aus. Diese Datei kann je nach verwendeter Office Version auch `STD.MSI` oder `PROPLUS.MSI` heißen. Nachdem auch diese Datei in die Installation integriert wurde, ist das Office auf dem neuesten Stand und kann verteilt werden.

Office Resource Kit

Nun würde die Installation von Office schon über das Netzwerk klappen, aber sie würde noch immer Benutzereingaben erfordern. Das wäre aber für unsere Lösung nicht praktikabel, da wir für unsere Verteilung eine automatische Installation benötigen. Daher kann man mithilfe zweier Programme Installationsvorlagen für Office erstellen. Anhand dieser erfolgt dann eine automatische Installation. Diese Programme sind kostenlos im Office Resource Kit 2003 (ORK) enthalten. Man kann es vom Microsoft Server herunterladen (<http://www.microsoft.com/downloads/>

details.aspx?FamilyID=4bb7cb10-a6e5-4334-8925-3bcf308cfbaf&DisplayLang=en>). Nach der Installation findet man mehrere nützliche Tools für die Wartung und Verteilung von Office im Startmenü vor. Für unser Vorhaben interessieren uns aber nur zwei Programme:

- Custom Installation Wizard

Mit diesem Programm des Resource Kits kann man Vorlagen für die Installation von Office erstellen. Man kann danach mit dieser Vorlage eine automatische Installation von Office oder Teilen davon durchführen.

- Custom Maintenance Wizard

Dieses Tool dient zur Veränderung einer verteilten Office Installation. Wir benötigen es, da wir ein verteiltes Office mit weiteren Programmen erweitern wollen oder eine Deinstallation einzelner Teile durchführen wollen.

Die komplette Deinstallation von Office funktioniert einfach mit dem Befehl `setup.exe /x PR011.msi /qn`, daher ist für diesen Vorgang nichts weiter zu tun. Für die Installation von Office muss man aber eine Vorlage erstellen. Dazu startet man den Custom Installation Wizard über das Startmenü. Im zweiten Schritt fragt der Assistent nach der MSI-Datei, die geöffnet werden soll. Hier gibt man die `PR011.MSI` aus dem Office Ordner an. Nun wählt man aus, dass man eine neue MST-Datei erstellen will und gleich darauf muss man angeben, wie diese heißen und wo sie gespeichert werden soll. Im nächsten Schritt gibt man den Ordner an, in dem das Office auf dem Client PC installiert werden wird. Außerdem fragt der Assistent die Organisation ab. Diese beiden Werte kann man in der Grundeinstellung belassen. Auch im nächsten Schritt kann man den Standardwert beibehalten. Nun folgt der wichtigste Schritt. In diesem Fenster muss man auswählen, welche Teile von Office man installieren will. Standardmäßig sind alle Programme von Office ausgewählt. Um einen Eintrag zu ändern, klickt man ihn an

und wählt im erscheinenden Menü „Nicht verfügbar“ aus. Das bewirkt, dass diese Komponente nicht installiert wird.

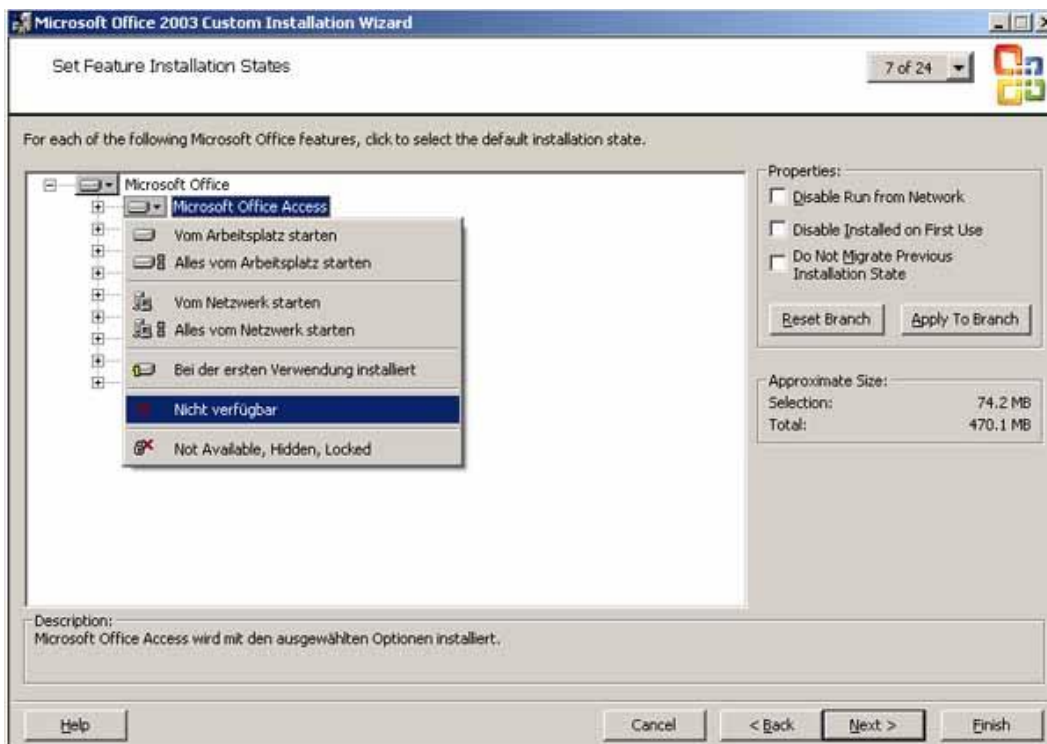


Abbildung 6 Custom Installation Wizard - Komponenten entfernen

Die nächsten Schritte des Assistenten sind für unsere Zwecke nicht wichtig, daher kann man rechts unten auf Finish klicken. Die Vorlage ist nun erstellt und kann für eine automatische Installation verwendet werden. Für die Einzelkomponenten (Word, Excel, PowerPoint) startet man den Assistenten ein weiteres Mal und kann die gerade eben erzeugte MST-Datei als Vorlage öffnen. Im Schritt 7 des Assistenten wählt man nun alle Teile außer der gewünschten Komponente ab. Die Deinstallation dieser Einzelkomponenten funktioniert mit dem gleichen Befehl wie für das gesamte Office, daher braucht man für diesen Punkt keine eigene Vorlage zu erstellen.

Für den zweiten Teil der Vorlagen benötigt man nun das zweite Programm des Resource Kits: den Custom Maintenance Wizard. Damit kann man Änderungen an Office Installationen vornehmen. Das ist für unser Ziel, Komponenten von Office hinzuzufügen beziehungsweise sie zu entfernen, wichtig. Man startet das Programm wieder über das Startmenü. Ein Assistent führt wieder durch den Vorgang, der der Erstellung von MST-Dateien ähnelt. Zuerst gibt man wieder den Pfad zur `PR011.MSI` an. Diesmal wird eine CMW-Datei erstellt, dafür muss man wieder einen Namen und einen Speicherort angeben. Im sechsten Schritt werden alle Komponenten von Office aufgelistet und man muss angeben welche Änderungen an der Installation durchgeführt werden sollen. Dazu klickt man auf den blauen Kreis neben der zu ändernden Komponente und wählt „Nicht verfügbar“ aus, wenn man diesen Teil entfernen will oder „Vom Arbeitsplatz starten“, wenn man die Komponente installieren will. Anschließend kann man auf Finish klicken und den Assistenten beenden. Man muss diesen Schritt nun für alle Möglichkeiten wiederholen (zB: Word hinzufügen und Word entfernen). Insgesamt kommt man nun auf sechs CMW Dateien und vier MST Dateien. Damit ist dieser Schritt abgeschlossen und Office kann automatisiert installiert werden.

Office Skript

Damit das Aufrufen der Vorlagen einfacher wird, wurden zuerst Shell Skripte geschrieben. Für den Aufruf aller Vorlagen waren dazu elf Skripte nötig. Damit die Anzahl kleiner wird und die Handhabung leichter, wurde ein Skript erstellt, das alle Einzelskripte vereint. Das hat den Vorteil, dass man nur ein Skript hat und durch die Übergabe von Parametern kann man bestimmen, welche Aktion ausgeführt werden soll.

Die Syntax des Skriptes lautet: `office.cmd [-d,-i] [full, word, excel, powerpnt]`

Die ersten beiden Parameter dienen zur Bestimmung, ob eine Installation oder Deinstallation durchgeführt wird. Anschließend wird abgefragt, ob Office komplett (full) installiert/deinstalliert wird, oder ob nur Teilprogramme (Word, Excel oder PowerPoint) geändert werden sollen. Anhand der jeweiligen Parameter werden die richtigen Vorlagen für die Installation verwendet.

Wenn man eine Teilkomponente installieren will (zB: `office.cmd -i word`), wird zuerst abgefragt, ob auf dem Client bereits Excel oder PowerPoint installiert ist. Wenn bereits eines der beiden Programme verfügbar ist, wird nur mehr Word hinzugefügt. Ist das nicht der Fall, wird Word mithilfe der MST-Datei neu installiert. Das Gleiche wird bei der Deinstallation durchgeführt. Dabei wird überprüft, ob das zu deinstallierende Programm das letzte Office Programm ist. Wenn ja, wird das Office inklusive der Hilfsprogramme und der Verknüpfungen im Startmenü gelöscht. Wenn nicht, wird nur das Programm entfernt und die restlichen stehen dem Benutzer weiter zur Verfügung. Für die korrekte Funktion des Skriptes muss man nur am Anfang der Datei der Variablen `SRV` die richtige IP oder den Namen des yasd Servers zuweisen.

Freigabe der Office Installation

Das Vorbereiten von Office für die Verteilung ist nach dem vorigen Schritt abgeschlossen. Nun muss man die Installation noch auf dem yasd Server für Clients zur Verfügung stellen. Dazu kopiert man den gesamten Office Ordner (inklusive des Skriptes und aller Vorlagen) auf den Server in den Freigabe-Ordner.

Damit steht der Verteilung von Office nichts mehr im Wege. Man muss es nur noch im Webinterface mit den richtigen Einträgen hinzufügen. Anschließend kann es an Clients zugewiesen werden. Während unserer Diplomarbeit hat Microsoft Office 2007 veröffentlicht. Der Zeitpunkt war aber für uns zu spät und daher haben wir für unsere Lösung nur Office 2003 verwendet. Man muss dabei aber

ähnlich wie oben beschrieben vorgehen. Es gibt aber ein neues Resource Kit und die Parameter wurden teilweise verändert.

Die Verteilung von Office über unsere Lösung bringt für Administratoren viele Vorteile. Die Installation auf einem Client funktioniert ohne Probleme, man braucht nur das Paket im Webinterface einem Client zuweisen und schon wird es auf dem PC installiert. Das erspart viel Zeit, da man sonst mit der Installations-CD von PC zu PC gehen müsste. Außerdem geschieht die Installation automatisch, da sie keine Benutzereingaben erfordert. Man kann Office nur teilweise auf Clients installieren und zu einem späteren Zeitpunkt über das Webinterface einfach weitere Programme von Office installieren. Die Installation geschieht sehr schnell (das Hinzufügen von Excel dauert auf einem normalen PC nur 20 Sekunden). Bei den Installationen ist schon das Service Pack 2 integriert. Es müssen keine großen Daten auf die Clients kopiert werden, da die Installation direkt von der Serverfreigabe gestartet wird. Das sind die wichtigsten Vorteile der Office Verteilung über yasd, die Liste könnte aber noch entsprechend erweitert werden.

3.6.3 Microsoft Updates

Dieser Abschnitt beschäftigt sich mit der Verteilung von Windows Updates. Dafür wurde ein Skript geschrieben, das am Microsoft Server anfragt, ob es neue Updates gibt und diese dann herunterlädt. Diese Patches werden am yasd Server freigegeben und können danach an die Clients verteilt werden. Dazu gibt es von Microsoft das Programm WSUS (Windows Server Update Services), welches genau diese Funktion unterstützt, und auch sehr einfach zu bedienen ist. Der einzige Nachteil daran ist, dass ein Server Betriebssystem von Windows erforderlich ist. Auf einem normalen PC mit Windows XP lässt es sich nicht installieren. Da auf den meisten Heimcomputern kein Serverbetriebssystem installiert ist, sollte unsere Lösung ähnlich funktionieren wie WSUS, aber unter Windows XP.

Dazu gibt es auf den Microsoft Servern eine Datei mit dem Namen `wsusscan.cab`. Diese enthält eine XML-Datei mit allen Windows Updates. Beim Erscheinen eines neuen Patches oder Programmes wird diese Datei automatisch aktualisiert und zum Download angeboten.

Diese Datei wurde auch für den ersten Entwurf des Windows Update Skriptes verwendet, aber am Jahresanfang hat Microsoft die Einstellung dieser Datei mit März 2007 bekanntgegeben. Ab diesem Zeitpunkt gilt nur noch die neue Datei `wsusscn2.cab`. Diese hat ein neues Format und alle Programme sollen auf diese neue Datei umgestellt werden. Die Umstellung war nötig, da die maximale Anzahl der Dateien in der `wsusscan.cab` erreicht wurde. Die neue Datei enthält nun mehrere CAB-Dateien und ist daher für die Zukunft gerüstet.

Auch wir mussten unsere Lösung auf diese neue Datei umschreiben. Mit unserer Lösung kann man Windows XP Installationen aktualisieren. Dabei spielt es keine Rolle, ob schon ein Service Pack installiert wurde, oder ob schon andere Patches auf dem System vorhanden sind. Im Folgenden werden nun die wichtigsten Teile des Windows Update Skriptes näher beschrieben:

```
rem reg.exe
if exist .\..\..\freigabe\winupdates\reg.exe goto SkipReg
if exist %SystemRoot%\system32\reg.exe (copy %SystemRoot%\system32\reg.exe
.\..\..\freigabe\winupdates)
:SkipReg

rem ifmember.exe
if not exist .\..\..\freigabe\winupdates\ifmember.exe (
  wget.exe -N
  http://download.microsoft.com/download/win2000platform/ifmember/1.00.0.1/NT5/EN-
  US/IfMember_Setup.exe -P "%TEMP%\ifmember"
  "%TEMP%\ifmember\IfMember_Setup.exe" /T:"%TEMP%\ifmember" /C
  copy msiextract.exe "%TEMP%\ifmember"

  pushd "%TEMP%\ifmember"
  msiextract.exe ifmember.Msi
  popd

  move "%TEMP%\ifmember\ifmember.exe" .\..\..\freigabe\winupdates > nul
  rd /S /Q "%TEMP%\ifmember"
)
```

```
rem Neue WSUS Update Datei herunterladen
wget -N
http://download.windowsupdate.com/microsoftupdate/v6/wsusscan/wsusscn2.cab -P
.\..\..\freigabe\winupdates

rem Update Datei extrahieren
echo Extracting update catalog file package.xml...
if exist "%TEMP%\package.cab" del "%TEMP%\package.cab"
if exist "%TEMP%\package.xml" del "%TEMP%\package.xml"
%SystemRoot%\system32\expand.exe .\..\..\freigabe\winupdates\wsusscn2.cab -
f:package.cab "%TEMP%"
%SystemRoot%\system32\expand.exe "%TEMP%\package.cab" -f:package.xml
%TEMP%\package.xml
del "%TEMP%\package.cab"

echo Updates ermitteln wxp deu...
if exist "%TEMP%\DownloadLinks-wxp-x86-deu.txt" del "%TEMP%\DownloadLinks-wxp-
x86-deu.txt"
msxsl.exe "%TEMP%\package.xml" ExtractDownloadLinks-wxp-x86-deu.xml -o
"%TEMP%\DownloadLinks-wxp-x86-deu.txt"

echo Updates ermitteln win deu...
if exist "%TEMP%\DownloadLinks-win-x86-deu.txt" del "%TEMP%\DownloadLinks-win-
x86-deu.txt"
msxsl.exe "%TEMP%\package.xml" ExtractDownloadLinks-win-x86-deu.xml -o
"%TEMP%\DownloadLinks-win-x86-deu.txt"

rem package.xml löschen
del "%TEMP%\package.xml"
```

Auflistung 18 winupdate.cmd - Teil 1

In den ersten Zeilen wird überprüft, ob alle nötigen Tools vorhanden sind. In unserem Fall sind das `reg.exe` und `ifmember.exe`. Danach wird überprüft, ob es auf dem Microsoft Server eine neue Version der Datei `wsusscn2.cab` gibt. Wenn ja, wird diese heruntergeladen und ersetzt die alte Version. Diese Datei enthält eine `package.cab`, in der dann die benötigte XML-Datei mit den Namen der Updates vorhanden ist. Dazu wird mit dem Programm `expand.exe`, das auf jedem Windows PC schon vorhanden ist, die `wsusscn2.cab` in das Temp-Verzeichnis entpackt.

Anschließend wird aus der so entstandenen `package.cab` die Datei `package.xml` extrahiert. Nun werden aus dieser Datei mit dem Programm `msxsl.exe` und einer selbst geschriebenen Vorlage die Downloadlinks für die Updates ermittelt. Dabei werden zwei Listen erstellt: Die erste enthält alle Windows XP Updates in

deutscher Sprache und die zweite enthält alle Pakete, die unabhängig von der Windows Version sind. Zu guter Letzt werden die erstellten Dateien im Temp-Verzeichnis wieder gelöscht. Übrig bleiben nun die zwei Dateien, die die Downloadlinks für die benötigten Updates enthalten.

```
echo Statische Pakete downloaden...
rem Windows XP SP2 DEU
wget.exe -N http://download.microsoft.com/download/9/6/4/96442257-721a-4cd5-9006-10a40cbb45cb/WindowsXP-KB835935-SP2-DEU.exe -P .\..\..\freigabe\winupdates\wxp

wget.exe -N http://download.microsoft.com/download/9/d/a/9dad25e-a010-4fb4-a711-83c9457eec97/WindowsXP-Windows2000-Script56-KB917344-x86-deu.exe -P
.\..\..\freigabe\winupdates\wxp

rem Windows Installer herunterladen
wget.exe -N http://download.microsoft.com/download/1/4/7/147ded26-931c-4daf-9095-ec7baf996f46/WindowsInstaller-KB893803-v2-x86.exe -P .\..\..\freigabe\winupdates

rem Windows Update Agent herunterladen
wget.exe -N
http://download.windowsupdate.com/v6/windowsupdate/redirect/standalone/WindowsUpdateAgent20-x86.exe -P .\..\..\freigabe\winupdates

rem Windows Script herunterladen
wget.exe -N http://download.microsoft.com/download/9/d/a/9dad25e-a010-4fb4-a711-83c9457eec97/WindowsXP-Windows2000-Script56-KB917344-x86-deu.exe -P
.\..\..\freigabe\winupdates
```

Auflistung 19 winupdate.cmd - Teil 2

In diesem Abschnitt des Skriptes werden zuerst statische Pakete von den Microsoft Servern heruntergeladen. Dazu zählt zum Beispiel das aktuelle Service Pack. Beim Erstellen der Diplomarbeit war gerade das SP 2 für Windows XP aktuell. Die restlichen Tools dienen dazu, ein System auf das Empfangen von Windows Updates vorzubereiten. Die Downloadlinks für diese Pakete sind statisch eingetragen und werden nur beim ersten Aufruf des Skriptes heruntergeladen. Bei jedem weiteren Aufruf erkennt das Skript, dass die Dateien bereits heruntergeladen wurden und überspringt diese Schritte. Bei Erscheinen von neueren Versionen dieser statischen Pakete müsste man die Links manuell aktualisieren.

```
echo Downloading updates for wxp deu...
wget.exe -nv -N -i "%TEMP%\DownloadLinks-wxp-x86-deu.txt" -P
.\..\..\freigabe\winupdates\wxp
rem Links löschen
del "%TEMP%\DownloadLinks-wxp-x86-deu.txt"

echo Downloading updates for win deu...
wget.exe -nv -N -i "%TEMP%\DownloadLinks-win-x86-deu.txt" -P
.\..\..\freigabe\winupdates\wxp
rem Downloadlinks löschen
del "%TEMP%\DownloadLinks-win-x86-deu.txt"

echo Reminding build date...
date /T > builddate.txt

:EOF
Endlocal
```

Auflistung 20 winupdate.cmd - Teil 3

In diesem letzten Teil des Skriptes werden die zwei erstellten Downloadlisten abgearbeitet und die Updates heruntergeladen. Dabei werden auch hier schon vorhandene Dateien berücksichtigt und nicht mehr heruntergeladen. Nach dem erfolgreichen Abarbeiten der Listen werden diese wieder vom System entfernt. Zuletzt wird in die Datei `builddate.txt` das Datum des letzten Skriptaufrufes hineingeschrieben. Damit kann man auf einen Blick in der Datei sehen, wann das Skript das letzte Mal aufgerufen wurde. Am Ende des Skriptes werden alle Variablen entladen und das Programm beendet sich.

Installationskript

Damit die am yasd Server bereitgestellten Updates an die Clients verteilt werden, musste ein weiteres Skript geschrieben werden. Im Laufe unserer Internetrecherchen sind wir auf ein Projekt von Torsten Wittrock gestoßen: c't Offline Update.

Wittrock hat eine Sammlung von Skripten geschrieben, die aus Windows Updates eine CD erstellt, mithilfe der man einen Windows PC auf den aktuellsten Stand bringen kann. Dieses Projekt hat er unter der GPL veröffentlicht. Das heißt, jeder

kann die Skripte verwenden und sie außerdem verändern. Damit mussten wir „das Rad nicht neu erfinden“ und konnten die Skripte für die Installation der Updates auf den Clients für unsere Zwecke verwenden. Dazu mussten hauptsächlich die Pfade zu den Dateien angepasst werden, da unsere Updates über das Netzwerk freigegeben und nicht auf einer CD bereitgestellt werden. Das folgende Flussdiagramm veranschaulicht schematisch, wie die Installation der Windows Updates auf den Client PCs erfolgt. ([Viol2006])

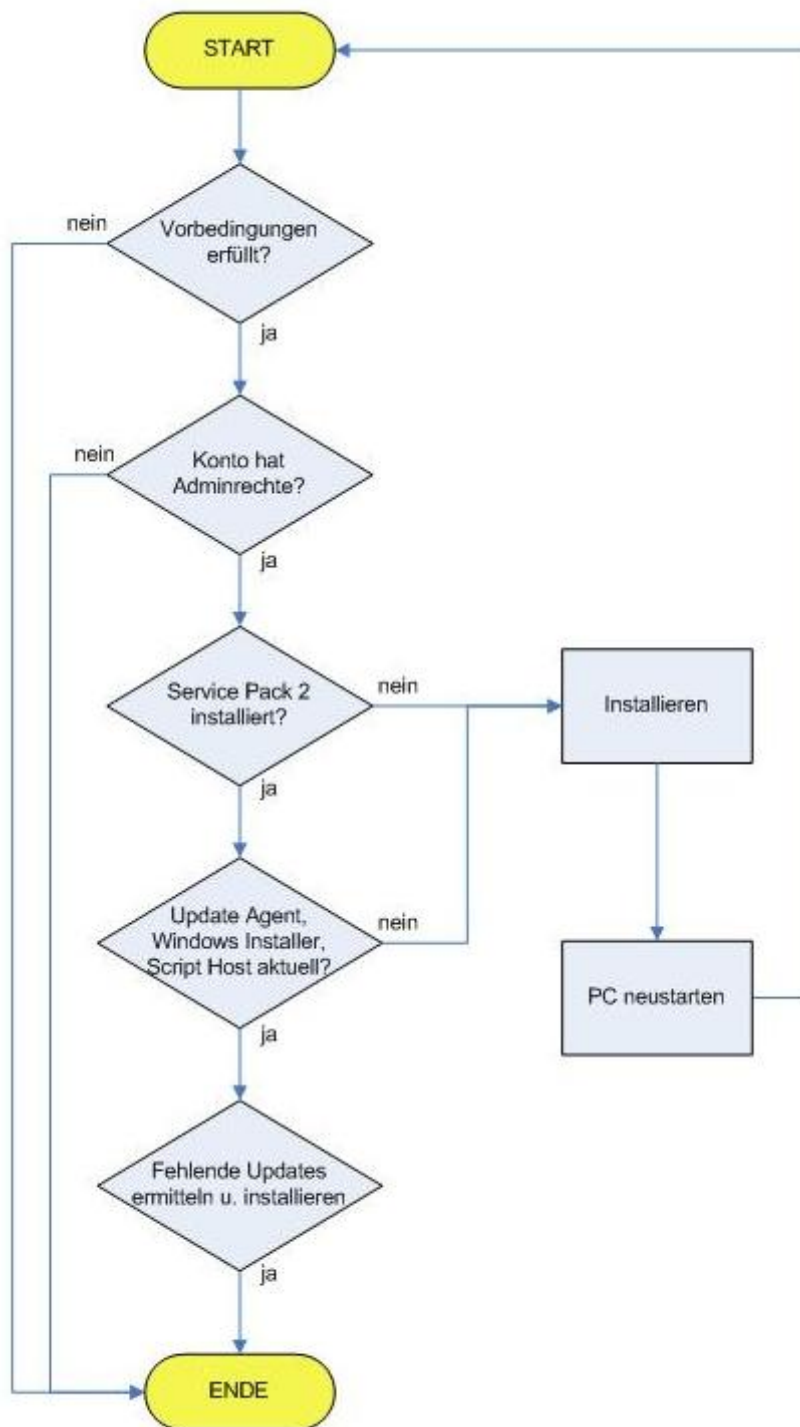


Abbildung 7 Flussdiagramm - Ablauf einer Windows Update Installation

Wenn das Installationskript auf einem Client PC ausgeführt wird, wird zuerst kontrolliert, ob die Vorbedingungen erfüllt werden. Darunter fällt hauptsächlich die Frage, ob benötigte Programme (`reg.exe`, `cscript.exe`, ...) vorhanden sind. Danach wird geprüft, ob das Skript von einem Benutzer ausgeführt wird, der Administratorrechte besitzt, da nur mit diesen eine korrekte Installation der Windows Updates möglich ist. Wenn einer dieser Punkte nicht erfüllt wird, kann das Skript nicht weiterlaufen und wird daher beendet. Danach wird überprüft, ob das Service Pack 2 auf dem Client PC installiert ist. Wenn nicht, wird es installiert und der PC neu gestartet. Anschließend wird das Skript neu ausgeführt. Nach dem SP2 wird überprüft, ob der Windows Update Agent, der Windows Installer und der Windows Script Host in der aktuellsten Version auf dem PC verfügbar sind. Wenn das nicht der Fall ist, werden die fehlenden Programme vom yasd Server geholt und installiert. Anschließend wird der PC neu gestartet und das Installationskript beginnt wieder von vorne. Im nächsten Schritt werden die am Client fehlenden Updates ermittelt und der Reihe nach installiert. Die Installation erfolgt im Hintergrund und der Benutzer merkt nichts davon. Nach der Fertigstellung ist der PC auf dem neuesten Stand und das Skript beendet sich.

Das Installationskript besteht aus mehreren Dateien. Für eine einfache Handhabung wurden sie deshalb in eine Zip-Datei zusammengefasst. Für eine Installation der Windows Updates muss diese Datei nur vom yasd Server kopiert und anschließend extrahiert werden. Außerdem gibt es eine `exclude-list.txt`, in die man Pakete eintragen kann, die man nicht an Clients verteilen will. Zum Beispiel könnte man hier den Internet Explorer 7 oder das Tool zum Entfernen bössartiger Software eintragen, wenn man diese nicht auf den Client PCs haben will.

Für die korrekte Funktion der Windows Update Verteilung benötigt man folgende Dateien am yasd Server:

- `wget.exe`
Dieses Programm ist ein bekanntes Unix-Tool und wird zum Herunterladen der Updates verwendet.
- `msxsl.exe` + zwei XSL-Vorlagen
Mit diesem Programm und den beiden XSL-Vorlagen kann man mithilfe der `packages.xml` eine Liste mit allen Downloadlinks der Updates erstellen.
- `msiextract.exe`
Damit kann man die benötigte Datei `ifmmember.exe` aus einem MSI-Archiv entpacken.
- `winupdate.cmd`
Dieses Skript wurde schon oben beschrieben und benötigt die vorher genannten Programme um Windows XP Updates zu ermitteln und sie anschließend herunterzuladen.
- `win-updateclient.unpack.zip`
Diese Sammlung von Skripten wird für die Installation der Updates auf den Client PCs benötigt.

Microsoft bringt jeden Monat neue Patches heraus, die Probleme oder Fehler von Windows beheben. Dazu gibt es einen Patchday an jedem ersten Dienstag im Monat. Wenn kritische Sicherheitslücken auftauchen, werden die Patches sofort veröffentlicht. Daher ist es wichtig, regelmäßig nach Windows Updates mithilfe des Skriptes `winupdate.cmd` zu suchen. Das kann man entweder manuell machen oder man fügt einen geplanten Task auf dem yasd Server hinzu, der nach einem bestimmten Intervall das Windows Update Skript ausführt. Ein Richtwert wäre ein- bis zweimal pro Woche ein Update durchzuführen. Ein regelmäßiges Ausführen des Skriptes bringt keine Nachteile mit sich, da nur neue Dateien heruntergeladen werden und der Internetzugang dadurch nicht unnötig belastet wird.

Unsere Implementierung der Windows Update Funktion bringt viele Vorteile mit sich. Die Dateien werden nur einmal heruntergeladen und stehen danach einer unbegrenzten Anzahl von Clients sofort zur Verfügung. Dadurch spart man Downloadkosten, da das Herunterladen der Patches nur einmal geschieht. Außerdem kann man ein neu installiertes System sehr schnell auf den neuesten Stand bringen. Das verhindert auch das Risiko eines Virenbefalls während des Herunterladens der Windows Updates mittels eines ungeschützten Windows Betriebssystems. Man kann außerdem Testclients einrichten, an die die Updates verteilt werden und nach einem ausgiebigen Test werden dann die Updates auch an die restlichen Clients verteilt.

3.6.4 Andere Pakete

Nachdem die anderen Aufgaben abgeschlossen waren und noch Zeit blieb, entschied ich mich ein weiteres Kann-Ziel zu verwirklichen. Dieses lautete weitere Programme über yasd verteilen zu können. Dazu habe ich zuerst ein paar frei verfügbare Programme gesucht. Danach habe ich recherchiert, welche Möglichkeiten es gibt, um eine Installation ohne Benutzerinteraktion durchzuführen. Ich bin dabei auf drei Möglichkeiten gestoßen.

- Silent Schalter

Diese Programme wurden schon für die automatische Installation entwickelt. Hierzu muss man die Installationsdatei mit einem speziellen Schalter (meist `/s`) starten. Danach wird das Programm im Hintergrund ohne Benutzerinteraktion auf dem Rechner installiert. Die Schwierigkeit bei solchen Programmen ist, die richtigen Schalter herauszufinden.

- MSI

Manche Programme werden als msi-Dateien ausgeliefert. Auch diese sind sehr leicht zu verteilen. Man muss dem Programm `msi.exe` nur die richtigen

Parameter und die MSI-Datei liefern. Danach läuft die Installation wieder vollautomatisch ab.

- AutoIt

Trifft keine der zwei Möglichkeiten auf ein Programm zu, dann kann man das Programm AutoIt verwenden. Mit dieser Freeware kann man Skripte erstellen. Das Installationsprogramm bleibt wie bei einer normalen Installation sichtbar, aber das Skript führt die nötigen Eingaben durch und die Installation verläuft halbautomatisch.

Die Bereitstellung von Paketen erfolgt in mehreren Schritten. Zuerst muss man herausfinden, mit welcher der drei Methoden eine automatische Installation erfolgen kann. Danach wird diese Methode ausprobiert. Anschließend muss man ausprobieren, wie man ein Programm wieder entfernen kann. Dabei gibt es unzählige Möglichkeiten. Meistens wird eine Uninstall-Datei im Programmverzeichnis angelegt und mit dieser wird die Deinstallation gestartet. Auch diese Dateien bieten oft Silent-Schalter. Bei MSI-Paketen ist die Deinstallation sehr leicht, da dazu die gleiche MSI-Datei wie zur Installation benötigt wird. Diese wird nur mit einem anderen Parameter aufgerufen und erfordert keine weitere Vorgehensweise. Falls diese beiden Möglichkeiten nicht funktionieren, muss man wieder ein AutoIt Skript schreiben, dass die Deinstallation durchführt. Damit diese neuen Pakete verteilt werden können, müssen sie zuerst in den Freigabe-Ordner kopiert werden und anschließend das neue Paket im Webinterface hinzugefügt werden. Danach kann es an Clients verteilt werden.

Wir haben schon mehrere Programme zusätzlich hinzugefügt: Mozilla Firefox, Mozilla Thunderbird, VLC, AntiVir, 7-Zip, GVim, Adobe Reader. yasd ist auf diesem Gebiet sehr flexibel. Bei genügend Zeit könnte man noch viele weitere Pakete mittels unserer Lösung verteilen.

3.7 Serverapplikation

Dieser Abschnitt beschäftigt sich mit der Implementierung der von yasd bereitgestellten Serverapplikation. Die Aufgabe dieses Programms ist die Kommunikation mit der in Kapitel 3.8 beschriebenen Clientapplikation über ein selbst entwickeltes Protokoll. Inhalt dieser Kommunikation sind Informationen über zu installierende und zu deinstallierende Pakete sowie erfolgte Installationen oder Deinstallationen. Diese Daten werden aus der Datenbank abgerufen bzw. dort gespeichert.

3.7.1 Protokoll

Zur Kommunikation zwischen Client- und Serverapplikation wurde ein Protokoll spezifiziert, welches im folgenden Abschnitt näher erläutert wird.

Anforderungen

Das Protokoll wurde entworfen, um den folgenden Ansprüchen zu genügen:

- Die Kommunikation zwischen Client und Server soll über eine zuverlässige Verbindung erfolgen.
- Um die Software-Pakete vor unberechtigtem Zugriff zu schützen, muss eine Authentifizierung der Clients erfolgen.
- Der Client kann alle notwendigen Informationen (wie in der Datenbank vorhanden) zur Installation und Deinstallation eines Pakets erfahren.
- Der Client kann dem Server die erfolgreiche Installation oder Deinstallation von Paketen mitteilen.
- Der Server antwortet dem Client bei allen diesen Aktionen mit einer Status-Meldung über den Ausgang.

Ablauf

Dieses Flussdiagramm zeigt einen Kommunikationsablauf, der die oben aufgeführten Bedingungen erfüllt:

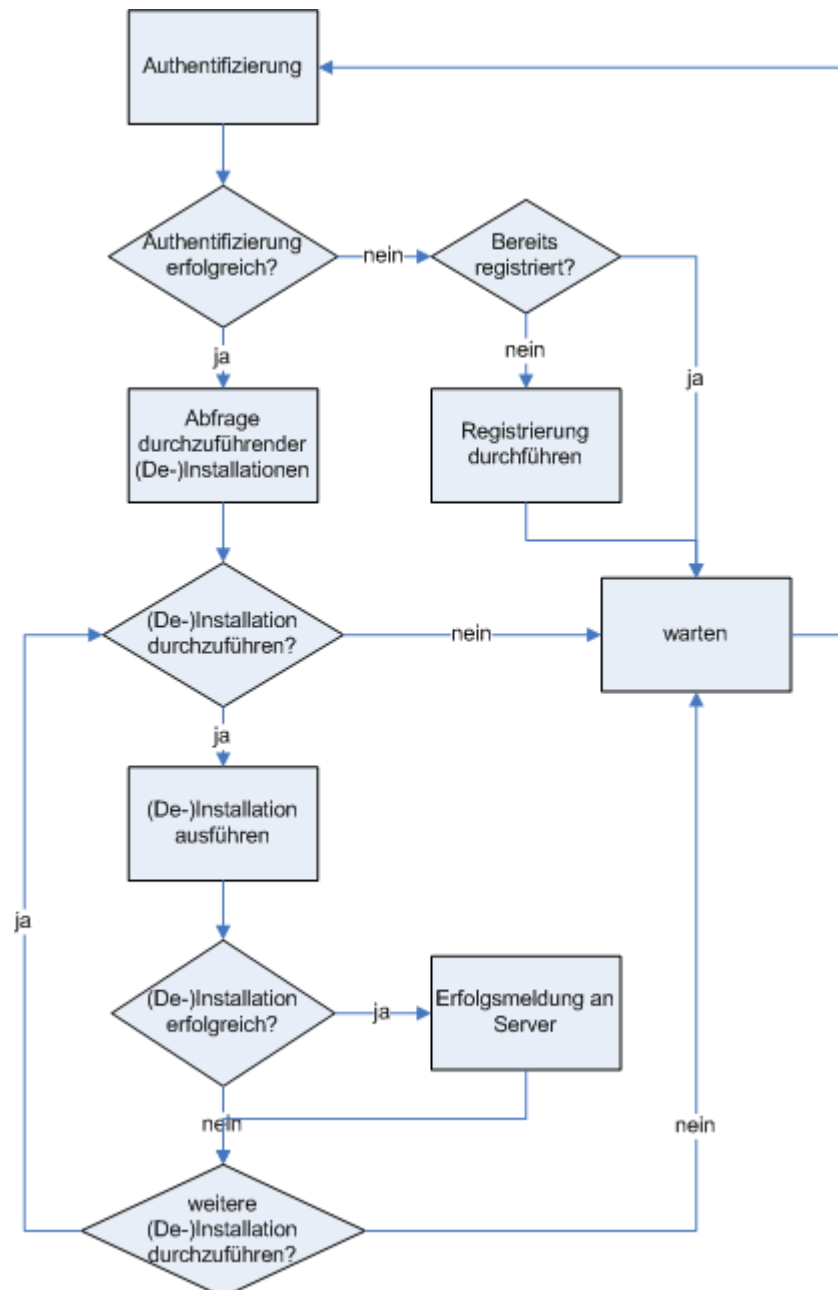


Abbildung 8 Flussdiagramm der Client-Server-Kommunikation

Paketaufbau

Das für yasd entwickelte Protokoll arbeitet am Application-Layer des von der ISO spezifizierten OSI-Modells. In den unteren Layern greift es auf die Protokolle TCP und IP des TCP/IP-Protokoll-Stacks zurück.

Am Application-Layer ist der Aufbau durch folgende Eigenschaften festgelegt:

- Das Paket ist unterteilt in beliebig viele Felder.
- Alle Felder werden als Zeichenkette interpretiert.
- Diese Felder werden mit dem Zeichen `\0` terminiert.
- Pakete werden mit den Zeichen `\r\n` terminiert.
- Bei Anfragen vom Client zum Server muss das erste Feld einen gültigen Paket-Code enthalten. Dieser gibt die auszuführende Aktion an.
- Bei Antworten vom Server zum Client muss das erste Feld einen gültigen Status-Code enthalten, der angibt, ob die Aktion erfolgreich ausgeführt wurde bzw. welcher Fehler aufgetreten ist.

Die folgende Tabelle beschreibt alle definierten Paket-Codes, wie viele und welche zusätzlichen Felder übergeben werden müssen, und wozu sie dienen.

Paket-Code	zusätzliche Felder	Beschreibung
auth (authentication)	Windows-SID des Clients	Führt die Authentifizierung des Clients durch.
reg (registration)	Windows-SID des Clients, Computernamen des Clients	Registriert einen Client am Server.
pkgqi (package query install)	(keine)	Ruft zu installierende Pakete vom Server ab.
pkgqu (package query uninstall)	(keine)	Ruft zu deinstallierende Pakete vom Server ab.
pkgui (package update install)	Paket-ID	Informiert den Server über eine erfolgreiche Installation.

pkguu (package update uninstall)	Paket-ID	Informiert den Server über eine erfolgreiche Deinstallation.
quit	(keine)	Beendet die Verbindung.

Tabelle 3 Erlaubte Paket-Codes

Die bei Antworten verwendeten Status-Codes sind an die des HTTP-Protokolls angelehnt. Obwohl auf jeden Status-Code beliebig viele Felder folgen dürfen, ist meist nur ein weiteres Feld, nämlich eine Beschreibung des Status-Codes in Textform, vorhanden.

Status-Code	Textform	Beschreibung
200	OK	Die Anfrage wurde erfolgreich bearbeitet; es sind keine Fehler aufgetreten.
206	Partial content	Die Anfrage wurde erfolgreich bearbeitet; es folgt ein Teil der angeforderten Daten.
400	Bad request	Allgemeiner, nicht näher bestimmbarer Fehler, verursacht durch den Client.
401	Bad number of arguments	Die Anfrage enthielt nicht die erforderliche Anzahl zusätzlicher Felder.
402	Not authenticated	Der Client hat keine Authentifizierung durchgeführt oder diese nicht erfolgreich abgeschlossen.
403	Not registered	Die Authentifizierung konnte nicht durchgeführt werden, da der Client nicht im System registriert ist.
404	Authentication failed	Der Client konnte gegen keines der konfigurierten Backends authentifiziert werden.
405	Already registered	Die Registrierung konnte nicht durchgeführt werden, da der Client bereits registriert ist.
500	Internal error	Allgemeiner, nicht näher bestimmbarer Fehler, verursacht durch den Server.

501	Unsupported method	Die vom Client per Paket-Code angeforderte Aktion wird vom Server nicht unterstützt.
------------	--------------------	--

Tabelle 4 Erlaubte Status-Codes

3.7.2 Serverimplementierung

Wahl der Programmiersprache

Bei der Wahl der Programmiersprache zur Implementierung der Serverapplikation musste vor allem auf die Anbindung der beiden Backends (MySQL und OpenLDAP) geachtet werden. Ein weiteres Kriterium war, den Server möglichst flexibel zu gestalten, um auf etwaige Änderungen rasch reagieren zu können. Außerdem musste das Kann-Ziel den Server plattformunabhängig zu gestalten hier bereits berücksichtigt werden.

Die Scriptsprache Python wurde gewählt, weil sie alle oben angeführten Kriterien erfüllt und zusätzlich das nötige Hintergrundwissen über die Sprache bereits vorhanden war. Zusätzlich zu den Modulen der Python Standard Library wurden MySQL-python für den Datenbankzugriff, python-ldap für den LDAP-Zugriff, ConfigObject sowie validate aus den PythonUtils zum Parsen der Konfigurationsdatei, pywin32 für die Implementierung des Windows-Dienstes sowie die Distutils-Erweiterung py2exe verwendet.

Kernbereich yasd-Server

Die wichtigste Funktion der Serverapplikation ist die oben beschriebene Kommunikation mit dem Client. Dazu wurden die Server-Klasse und der eigentliche Request-Handler von den Klassen `SocketServer.TCPServer` und `SocketServer.StreamRequestHandler` abgeleitet. Zusätzlich sorgt ein selbst geschriebenes MixIn, nämlich `DaemonThreadingMixIn`, dafür, dass mehrere Clients gleichzeitig bedient werden können.

Der Request-Handler führt die vom Client angeforderten Aktionen – die mit den oben angeführten Paket-Codes bezeichnet werden – aus, und antwortet mit einer Erfolgs- bzw. Fehlermeldung. Dazu verwendet er die Status-Codes des yasd-Protokolls und eine Beschreibung des Status-Codes in Textform.

Nur den Requests `pkgqi` und `pkgqu` – also den Abrufen von (De-)Installationsinformationen – folgen Pakete mit dem Status-Code 206; diese enthalten weitere Informationen über die einzelnen Software-Pakete. Pro Paket wird eine Antwort geschickt; diese enthält sämtliche in der Datenbank über das Paket abspeicherbare Daten. In der Datenbank nicht vorhandene Informationen (`null`) werden durch die Zeichenfolge `"None"` ersetzt. Diese Übertragungen werden mit einem Paket mit dem Code 200 abgeschlossen.

Authentifizierung

Um garantieren zu können, dass die freigegebene Software nicht an Dritte verteilt wird, muss sich der Client am Server authentifizieren. Lediglich die Aktionen `reg`, `quit` und natürlich `auth` selbst erfordern keine vorherige Authentifizierung. Versucht der Client eine andere als diese Aktionen ohne vorherige Authentifizierung durchzuführen, erhält er eine Antwort mit dem Fehler-Code 402 – „not authenticated“.

Die Authentifizierung geschieht mit einer Anforderung mit dem Paket-Code `auth`. Zusätzlich wird die Windows-SID des Computers übergeben, auf dem der yasd-Client installiert ist. Über diesen „Security Identifier“ kann jedes Gerät eindeutig identifiziert werden. Zuerst wird die übermittelte SID in der Datenbank gesucht. Ist der Client noch gar nicht im System registriert, erfolgt eine Antwort mit dem Fehler-Code 403 – „not registered“. Kann der Client in der Datenbank nicht gefunden werden, ist aber die LDAP-Authentifizierung aktiviert, so wird hier ein Eintrag, bei dem das Attribut `sambaSID` mit der SID des Clients übereinstimmt,

gesucht. Wird der Client in keinem der beiden Backends gefunden (oder nur in der Datenbank, wenn die LDAP-Authentifizierung nicht aktiviert wurde), wird eine Antwort mit dem Fehler-Code 404 – „authentication failed“ – gesendet. Diese Reaktion erfolgt auch, wenn der Client zwar in der Datenbank abgelegt, das Feld `registered` aber noch nicht auf `TRUE` gesetzt wurde, der Client vom Administrator also noch nicht freigegeben wurde. Andernfalls ist der Client authentifiziert, er erhält sofort die Nachricht 200 – „OK“.

Registrierung

Um dem Administrator zu ersparen, jeden Client einzeln im Webinterface anlegen müssen, können sich Clients am Server registrieren. Dies geschieht mit dem Paket-Code `reg`, als Parameter werden die SID und der Computer-Name übergeben, um den Client später leichter identifizieren zu können. Der Server versucht daraufhin den Client in der Datenbank anzulegen. Ist die SID bereits in der Datenbank vorhanden, erfolgt eine Fehler-Antwort mit dem Code 405 – „already registered“, sonst war die Registrierung erfolgreich und wird mit dem Code 200 beantwortet. Natürlich kann der Client noch nicht sofort Pakete beziehen, dazu muss er zuerst vom Administrator über das Webinterface freigegeben werden.

Austausch von Paketinformation

Die Anfragen nach neu zu installierenden bzw. zu deinstallierenden Paketen mittels `pkgqi` und `pkgqu` werden mit mehreren Paketen beantwortet. Dazu werden im Fall von `pkgqi` in der Datenbank Pakete gesucht, die dem Client selbst oder einer der Gruppen, in die er aufgenommen wurde, zugeordnet sind, von denen aber noch keine erfolgreiche Installation vom Client gemeldet wurde. Bei `pkgqu` werden Pakete gesucht, die der Client installiert hat, die ihm aber nicht mehr – direkt oder indirekt über eine Gruppe – zugeordnet sind. Daraufhin wird pro

gefundenem Software-Paket eine Antwort mit dem Status-Code 206 – „partial content“ gesendet. In dieser Antwort werden zusätzlich zum Paket-Code die Informationen aus der Datenbank über das Software-Paket – jeweils ein Datenbank-Feld pro Feld in der Antwort – übertragen. Die Reihenfolge hierbei ist: `id`, `name`, `version`, `description`, `priority`, `fetch_url`, `inst_exec`, `inst_exec_args`, `inst_ret_ok`, `uninst_exec`, `uninst_exec_args`, `uninst_ret_ok`. Da die Reihenfolge der (De-)Installation von Paketen mit dem `priority`-Attribut geordnet werden kann, wird diese Reihenfolge bereits beim Auslesen aus der Datenbank berücksichtigt und vor der Übertragung nicht mehr verändert. Abgeschlossen wird die Kommunikation mit einer Antwort 200 – „OK“.

(De-)Installationsmeldungen

Hat ein Client ein Softwarepaket erfolgreich (de-)installiert, so erfolgt eine Meldung an den Server mit den Paket-Codes `pkgui` respektive `pkguu`. Übergeben wird dabei nur mehr die Paket-ID. Der Server aktualisiert daraufhin die Datenbank entsprechend und sendet ein 200 – „OK“.

Verbindungsabbau

Der Paket-Code `quit` ermöglicht den Verbindungsabbau per Kommando. Dieser Paket-Code wurde allerdings nur für Debug-Zwecke implementiert und wird von der Endversion nicht mehr verwendet.

HTTP-Server

Zusätzlich zum Server zur Kommunikation mit dem Client wurde ein HTTP-Server implementiert, über den Pakete freigegeben werden können. Die Server-Klasse und der Request-Handler wurden von den Klassen `BaseHTTPServer.HTTPServer` und `SimpleHTTPServer.SimpleHTTPRequestHandler` aus der Python Standard Library abgeleitet. Der Zugriff auf den Server ist an die Authentifizierung mittels `auth` geknüpft: Sobald die Authentifizierung erfolgreich abgeschlossen wurde, darf der

Client eine konfigurierbare Zeit lang auf die Ressourcen des HTTP-Servers zugreifen. Die Überprüfung erfolgt bei jedem GET-Request.

Konfiguration

Die Konfiguration der Serverapplikation ist über eine INI-Datei möglich. Diese wird unter Microsoft Windows im Programmverzeichnis als `yasds.ini` und auf allen anderen Plattformen unter `/etc/yasds.conf` erwartet.

Im Folgenden wird die Syntax der Konfigurationsdatei gezeigt:

```
[HttpServer]
Port = 8000
RootDir = .
AuthValid = 30

[YasdServer]
Port = 8001

[DatabaseConnection]
Host = localhost
User = root
Password = password
Database = yasd

# Werte die Beistriche enthalten, müssen von Anführungszeichen umschlossen werden
[LDAPConnection]
URL = ldap://localhost
BindDN = "cn=ldapadmin,dc=yasd,dc=at"
BindPW = ldappass
SearchBase = "ou=Computers,dc=yasd,dc=at"

[Authentication]
LDAP = False
```

Auflistung 21 yasds.ini / yasds.conf

Die nachfolgende Tabelle zeigt alle möglichen Optionen, deren Default-Werte und eine kurze Beschreibung.

Option	Default-Wert	Erklärung
[HttpServer]		
Port	8000	Port, an den der HTTP-Server gebunden werden soll

RootDir	Ausführungsverzeichnis	Verzeichnis, in dem der HTTP-Root liegen soll
AuthValid	30	Gültigkeit einer Authentifizierung für den HTTP-Server in Minuten
[YasdServer]		
Port	8001	Port, an den der yasd-Server gebunden werden soll
[DatabaseConnection]		
Host	localhost	Hostname oder IP, auf der der MySQL-Server läuft
User	root	Benutzername für die MySQL-Authentifizierung
Password	password	Passwort für die MySQL-Authentifizierung
Database	yasd	Datenbankname
[LDAPConnection]		
URL	ldap://localhost	LDAP-URL des OpenLDAP-Servers
BindDN	"cn=ldapadmin, dc=example,dc=com"	Distinguished Name für den LDAP-Bind
BindPW	ldappass	Passwort für den LDAP-Bind
SearchBase	"ou=Computers, dc=example,dc=com"	Basis für LDAP-Suchen
[Authentication]		
LDAP	False	Gibt an, ob die LDAP-Authentifizierung benutzt werden soll

Tabelle 5 Erklärung der Parameter in yasds.ini / yasds.conf

Anmerkungen:

- Werden einzelne Optionen nicht angegeben, so werden automatisch die Default-Werte benutzt.

- Werte, die Beistriche enthalten (`BindDN`, `SearchBase`), müssen in Anführungszeichen eingeschlossen werden.
- Die LDAP-Suche nach authentifizierten Clients wie unter „Authentifizierung“ beschrieben, erfolgt mit der in `SearchBase` konfigurierten Basis; das Scope ist immer `SUBTREE`.

Unix-Daemon

```
def create_daemon():
    if os.geteuid():
        logging.error("Must be started with root privileges to daemonize.")
        sys.exit(1)
    logging.info('Daemonizing ...')

    try:
        pid = os.fork()    # fork first child
        if pid > 0:
            os._exit(0)    # parent exits
    except OSError, error:
        logging.error("Daemonizing failed: %s" % error.strerror)
        os._exit(1)        # fork error

    os.setsid()    # obtain a new process group

    try:
        pid = os.fork()    # do second fork
        if pid > 0:
            os._exit(0)
    except OSError, error:
        logging.error("Daemonizing failed: %s" % error.strerror)
        os._exit(1)        # fork error

    os.chdir('/')

    for fd in range(0, 3):
        try:
            os.close(fd)
        except OSError:
            pass

    os.open(os.devnull, os.O_RDWR)    # standard input (0)
    os.dup2(0, 1)                      # standard output (1)
    os.dup2(0, 2)                      # standard error (2)
```

Auflistung 22 yasds.py: create_daemon()

Unter allen Nicht-Windows-Betriebssystemen wird standardmäßig versucht, den yasd-Server als UNIX-Daemon zu starten. Dazu dient die oben aufgelistete Funktion `create_daemon()`, die im Weiteren näher erläutert werden soll.

Um die nötigen Rechte zum Aufruf der System-Calls, die den Daemon erzeugen, zu haben, wird überprüft, ob der Prozess im Sicherheitskontext des `root`-Users läuft. Dies wird anhand der „effective user id“ des laufenden Prozesses überprüft, ist diese Null (die user id des `root`-Users), wird fortgefahren, ansonsten mit einer Fehlermeldung abgebrochen.

Danach erfolgt der Aufruf von `fork()`, der eine Kopie des aufrufenden Prozesses erzeugen soll. Ist dieser Aufruf erfolgreich, beendet sich der aufrufende – auch „Parent“ genannte – Prozess und gibt die Kontrolle an die Shell zurück. Weiters wird mit dem System-Call `setsid()` dem Prozess eine neue Prozessgruppe und -session zugewiesen (die er von seinem Parent-Prozess geerbt hat), um zu verhindern, dass er Signale vom aufrufenden Terminal erhält.

Mit einem weiteren `fork()`-Aufruf verwaist der Kind-Prozess und wird vom Betriebssystem als Kindprozess an `init`, den zu allererst startenden Prozess, angehängt. Dadurch wird der `init`-Prozess für die Kontrolle und Aufräumarbeiten zuständig und sogenannte „Zombies“ – Prozesse die ihre Ausführung abgeschlossen aber noch immer in der Prozesstabelle aufscheinen – werden vermieden. Damit die Umgebung, in der der Prozess nun läuft, erhalten bleibt, wird das Arbeitsverzeichnis nach `/` verlagert.

Da der Prozess auch keine der drei üblichen Ein- und Ausgabestreams (`stdin`, `stdout`, `stderr`) mehr besitzt, werden die zugehörigen „file descriptors“ geschlossen und mit dem Null-Device des Betriebssystems als Ziel wieder geöffnet.

([Kara2001], [Schr2005])

Soll der Prozess im Vordergrund bleiben, muss das Programm mit der Option `-f` gestartet werden.

Windows-Dienst

Um den `yasd` Server unter Windows als Dienst zu starten, wird die Modulsammlung `pywin32` verwendet. Der Code zur Interaktion mit dem Windows Dienste-Manager lautet wie folgt:

```
import os
import sys
import threading

import win32serviceutil
import win32service
import win32event

import yasds

class YasdServiceThread(threading.Thread):

    def __init__(self):
        threading.Thread.__init__(self)
        self.stopevent = threading.Event()

    def run(self):
        yasds.main(self.stopevent)

    def stop(self):
        self.stopevent.set()

class yasds_service(win32serviceutil.ServiceFramework):
    _svc_name_ = "yasdsd"
    _svc_display_name_ = "yasd Server Service"
    _svc_deps_ = ["EventLog"]

    def __init__(self, args):
        win32serviceutil.ServiceFramework.__init__(self, args)
        self.stopevent = threading.Event()

    def SvcStop(self):
        self.ReportServiceStatus(win32service.SERVICE_STOP_PENDING)
        self.stopevent.set()

    def SvcDoRun(self):
        self.ReportServiceStatus(win32service.SERVICE_START_PENDING)

        yasd_thread = YasdServiceThread()
        yasd_thread.start()
```

```
self.ReportServiceStatus(win32service.SERVICE_RUNNING)

self.stopevent.wait()

yasd_thread.stop()

self.ReportServiceStatus(win32service.SERVICE_STOPPED)

if __name__ == '__main__':
    win32serviceutil.HandleCommandLine(yasds_service)
```

Auflistung 23 yasds_service.py

Die Dienst-Parameter selbst werden in der Klasse `yasds_service` implementiert, die von `win32serviceutil.ServiceFramework` abgeleitet ist. Sie reagiert auf Anweisung des Windows Dienste-Managers für die Ereignisse „DoRun“ und „Stop“ – also Starten und Stoppen des Dienstes. Beim Starten des Dienstes wird die Funktion `SvcDoRun` aufgerufen. Hier wird der eigentliche yasd-Server in einem eigenen Thread gestartet und der Dienststart als erfolgreich gemeldet. Daraufhin wird auf das Stoppen des Dienstes gewartet, die Information darüber erhält die Funktion über ein Threading-Event, welches in der Funktion `SvcStop`, die wiederum vom Dienste-Manager aufgerufen wird, gesetzt wird. Der yasd-Server wird über ein weiteres Threading-Event zum Stoppen aufgefordert, die erfolgreiche Durchführung wieder gemeldet.

Logging

Um dem Benutzer zu erlauben, dem Programmablauf auch zu folgen, wenn der yasd-Server als UNIX-Daemon bzw. als Windows-Dienst läuft, werden in diesen Ausführungsfällen Protokoll-Dateien geschrieben.

Dazu wird das Modul `logging` aus der Python Standard Library verwendet. Die Logdateien sind unter Windows im Ausführungsverzeichnis des Programms als `yasds-log.txt` und unter allen anderen Betriebssystemen unter `/var/log/yasds.log` zu finden.

Packaging

Um den Server leicht verteilen und installieren zu können, wurde mit dem „Nullsoft Installer System“ – kurz NSIS – ein Setup-Programm erstellt. Dieses übernimmt das Kopieren der benötigten Dateien, legt Startmenü-Einträge fest, installiert den yasd-Server als Windows-Dienst und erstellt einen passenden Uninstaller, der diese Vorgänge bei der Deinstallation wieder rückgängig macht.

Um den yasd-Server, der ja in Python implementiert wurde, auch auf Windows-Systemen ohne vorhandene Python-Installation verwenden zu können, wurde die Distutils-Erweiterung `py2exe` verwendet. (Die Distutils sind ein Modul der Standard Python Library und werden zur Installation und Vorbereitung von Python-Modulen verwendet.) Folgendes Skript erstellt aus den vorhandenen Python-Quelldateien unter Windows ausführbare EXE-Dateien:

```
from distutils.core import setup
import py2exe

setup(options={"py2exe" : {"bundle_files" : 1, }},
      zipfile=None,
      console=['yasds.py'],
      service=['yasds_service'],
      )
```

Auflistung 24 setup.py

Aufgerufen wird das Skript mit `setup.py py2exe`. Daraufhin sucht und kopiert `py2exe` verwendete Module und die Dateien, die Python zur Laufzeit benötigt, um sie auf dem Zielsystem verfügbar zu machen und erstellt eine EXE-Datei, die nicht nach einer Python-Installation sucht, sondern diese Kopien verwendet. Die Option `bundle_files` mit dem Parameter `1` bewirkt, dass die gesuchten Dateien in einer einzigen Zip-Datei gebündelt werden. Durch den Aufruf mit `zipfile=None` wird diese Datei zusätzlich an die EXE-Datei angelagert und erst beim Starten entpackt.

Durch die beiden weiteren Optionen `console=` und `service=` werden zwei EXE-Dateien erstellt, eine startet als normale Konsolenanwendung den eigentlichen Python-Server aus `yasds.py`, die zweite ist ein Windows-Dienst, der wie oben vorgestellt `pywin32` zur Kommunikation mit dem Windows Dienste-Manager verwendet.

Daher müssen nur mehr zwei ausführbare Dateien, die den Python-Interpreter und alle benötigten Laufzeitabhängigkeiten bereits enthalten, vom Setup installiert werden. Diese sind die Konsolenanwendung (`yasds.exe`) und der Windows-Dienst (`yasds_service.exe`). Außerdem werden eine Default-Konfigurationsdatei (`yasds.ini`) und ein Batch-Skript zur Kontrolle des Windows Dienstes (`yasdsd.cmd`) mitgeliefert.

3.8 Clientapplikation

Dieses Kapitel beschreibt Design und Implementierung der für dieses Projekt entwickelten Clientapplikation. Die Aufgabe dieses Programms ist die Kommunikation mit der in Kapitel 3.7 beschriebenen Serverapplikation.

3.8.1 Clientimplementierung

Wahl der Programmiersprache

Wichtig für die Wahl der Programmiersprache zur Implementierung der Clientapplikation war, dass die Anwendung sowohl als Desktop-Anwendung in der Konsole als auch als Windows-Dienst laufen kann. Soll ein neues Softwarepaket getestet werden, so kann der Administrator zuerst die Installation über die Konsolenanwendung mit verfolgen. Im Produktivbetrieb sollte der `yasd`-Client dann als Windows-Dienst starten. Dadurch ergeben sich zwei Vorteile: erstens bemerkt der Anwender nichts von der eigentlichen Softwareinstallation; zweitens

haben aus einem Windows-Dienst heraus gestartete Programme automatisch die nötigen Rechte um eine Installation durchzuführen.

Die beiden Sprachen C++ und C# erfüllen die Anforderung an eine direkte Integration als Windows-Dienst. Die Wahl fiel auf C#, da dies die Standardsprache für Microsofts .NET Framework ist, welches durch seine umfangreiche Klassenbibliothek die Programmierung weiter vereinfacht. Gleichzeitig wurde eine optionale Verwendung von C++ – für den Fall, dass die Mittel des .NET Frameworks für die Implementierung nicht ausreichen sollten – festgelegt, da das „Microsoft Windows Platform SDK“ (die Windows Programmierschnittstellen) nur mit dieser Sprache genutzt werden kann.

Zusätzlich zur Klassenbibliothek des .NET Frameworks wurden die Bibliotheken Nini zum Auslesen von INI-Dateien, NLog für die Protokollierung und SharpZipLib zum Zugriff auf Dateiarhive verwendet.

Kernanwendung yasd-Client

Auf Seite der Clientapplikation ist die wichtigste Funktion die Kommunikation mit dem Server. Dazu wurden die Sende- und Empfangsmethoden in einer Klasse `YasdServerClient` gekapselt, in der die Klasse `System.Net.Sockets.TcpClient` aus dem .NET Framework instanziiert und zur eigentlichen Netzwerkkommunikation verwendet wird. Eine eigene Klasse `YasdClient`, die den logischen Ablauf der Kommunikation steuert, greift auf diese Schnittstellen zurück. Zusätzlich wurden einige Hilfsfunktionen zur internen Verwendung definiert. Alle in diesem Abschnitt genannten Funktionen liegen in Form einer einzelnen DLL-Datei vor. Darauf setzen die beiden separaten EXE-Dateien – Konsolenanwendung und Windows-Dienst – auf. Somit ist die eigentliche Programmlogik getrennt vom Quellcode der jeweiligen Ausführungsform und kann über eine definierte Schnittstelle angesprochen werden.

Authentifizierung

Um andere Befehle ausführen zu können und den Zugriff auf den im yasd-Server integrierten HTTP-Server freizuschalten, muss sich der Client authentifizieren.

Um die zur Authentifizierung verwendete SID auszulesen, werden die System-Aufrufe `LookupAccountName` und `ConvertSidToStringSid` verwendet. `LookupAccountName` wird verwendet, um die SID eines beliebigen Kontos in binärer Form auszulesen, `ConvertSidToStringSid` wandelt diese anschließend in eine Zeichenkette um. Nachdem ab Windows NT nicht nur Benutzerkonten sondern auch jeder Computer durch einen Account identifiziert wird, kann auch dessen SID nachgeschlagen werden. Definitionen dieser Funktionen sind nur in Microsofts Platform SDK vorhanden und können nur aus C/C++-Code angesprochen werden. Da die Entwicklung des Clients aber nach Möglichkeit in .NET/C# erfolgen sollte, wurde P/Invoke (Platform Invoke) verwendet, um die Funktionen direkt aus dem C#-Quellcode in der mit Microsoft Windows mitgelieferten nativen DLL aufzurufen.

Dazu muss die entsprechende Funktion als extern vorhanden deklariert werden, die DLL, in der die Implementierung der Funktion zu finden ist, bekannt gegeben werden, und eine aus .NET aufrufbare Funktionssignatur geschrieben werden. Für die beiden genannten Funktionen entsteht folgender Quellcode:

```
[DllImport("advapi32.dll", CharSet = CharSet.Auto, SetLastError = true)]
public static extern bool LookupAccountName(
    [In, MarshalAs(UnmanagedType.LPCTSTR)] string systemName,
    [In, MarshalAs(UnmanagedType.LPCTSTR)] string accountName,
    IntPtr sid,
    ref int cbSid,
    StringBuilder referencedDomainName,
    ref int cbReferencedDomainName,
    out int use);

[DllImport("advapi32.dll", CharSet = CharSet.Auto, SetLastError = true)]
internal static extern bool ConvertSidToStringSid(
    IntPtr sid,
    [In, Out, MarshalAs(UnmanagedType.LPCTSTR)] ref string pStringSid);
```

Auflistung 25 Deklaration der per P/Invoke aufzurufenden Funktionen

Der Aufruf der zwei Funktionen wurde in eine eigene Funktion `GetSID` gekapselt, die auch eine entsprechende Fehlerbehandlung durchführt. Anschließend wird die SID des lokalen Computers als Zeichenkette zurückgegeben. Diese wird dann in einem Authentifizierungs-Request (`auth`) an den Server gesendet.

Antwortet der Server mit dem Erfolgs-Code 200, wird die Routine mit dem Abrufen von Paketinformationen fortgesetzt. Erhält der Client die Fehlermeldung 403 – „not registered“, so versucht er die Registrierung wie im nächsten Abschnitt beschrieben durchzuführen. Mit der Meldung 404 – „authentication failed“ meldet der Server einen anderen Fehler bei der Authentifizierung. Der Client reagiert auf diesen, indem er die Authentifizierungs-Anfrage nach einer konfigurierbaren Zeit erneut sendet.

Registrierung

Die zur Registrierung notwendigen Informationen sind die SID des Clients und sein Computernamen. Sie werden verwendet, um dem Administrator das manuelle Anlegen von neuen Clients im Webinterface zu erleichtern, indem die notwendigen Informationen vom Client selbst eingetragen werden. Anschließend muss ihn der Administrator nur mehr freischalten.

Die SID wird wie im vorherigen Abschnitt beschrieben ausgelesen; der Computernamen findet sich unter `.NET` in der Variable `System.Environment.MachineName`. Der Computernamen hat aber keinerlei Auswirkungen auf die Authentifizierung (und kann daher problemlos geändert werden), er dient nur zur Anzeige im Webinterface.

Sind diese Informationen gesammelt, werden sie in einem Registrierungs-Request (`reg`) an den Server gesendet. Daraufhin wartet der Client eine konfigurierbare Zeitspanne, bevor er einen Authentifizierungsversuch startet, da er ja zuerst vom Administrator freigeschaltet werden muss.

Austausch und Ausführung von (De-)Installationsanweisungen

War die Authentifizierung erfolgreich, versucht der Client nun Informationen über Pakete zu beziehen, die er installieren oder deinstallieren muss. Dazu schickt er die Abfrage-Requests `pkgqi` und `pkgqu` an den Server. Zuerst werden Installationsanweisungen verarbeitet, danach folgen die Deinstallationen.

Pro Paket werden folgende Informationen unabhängig von der auszuführenden Aktion (Installation/Deinstallation) empfangen: `id`, `name`, `version`, `description`, `priority`, `fetch_url`, `inst_exec`, `inst_exec_args`, `inst_ret_ok`, `uninst_exec`, `uninst_exec_args`, `uninst_ret_ok`. Pro Softwarepaket werden diese in einem Paket mit dem Code 206 übertragen. Wird die Übertragung mit dem Empfang des Erfolgscode 200 abgeschlossen, beginnt die Verarbeitung der Informationen.

Das Feld `priority` kann ignoriert werden, da die Übertragung bereits nach dieser Information geordnet erfolgt. Je nach auszuführender Aktion werden die zu startenden Programme aus `inst_exec` oder `uninst_exec`, die Parameter aus `inst_ret_ok` oder `uninst_exec_args` und die zulässigen Rückgabewerte aus `inst_ret_ok` oder `uninst_ret_ok` zwischengespeichert.

Sowohl im Pfad des auszuführenden Programms und in den Parametern werden enthaltene Umgebungsvariablen durch ihre Werte ersetzt. Das ermöglicht zum Beispiel den transparenten Zugriff auf das Windows-Verzeichnis unabhängig vom Laufwerk über `%SYSTEMROOT%` oder `%WINDIR%`. Außerdem wird die Zeichenkette `<YASDS>` durch den Namen oder die IP-Adresse des am Client konfigurierten yasd-Servers ersetzt. `fetch_url` ermöglicht den Download oder das Kopieren einer Datei in ein konfigurierbares Verzeichnis. Um dies möglichst transparent zu gestalten, ist auch hier die Ersetzung von `<YASDS>` – wie oben beschrieben – und die Ersetzung von `<YASDHTTPS>` durch den am Client konfigurierten yasd-Server und dessen HTTP-Port in der üblichen Form `hostname:port` möglich.

Ist das Feld `fetch_url` definiert (also nicht `"None"`), wird die definierte Datei in das konfigurierte Verzeichnis für temporäre Dateien heruntergeladen. Unterstützt werden die Protokolle `http:`, `https:`, `ftp:`, `file:` über die .NET-Klasse `System.Net.WebClient` und deren Methode `DownloadFile()` und alle lokalen und UNC-Pfade über die Methode `System.IO.File.Copy()`. Nach dem Download wird die Zeichenkette `<FETCHFILE>` im auszuführenden Pfad und in den Argumenten durch den lokalen Dateinamen ersetzt.

Ist die heruntergeladene Datei eine ZIP-Datei und endet der Dateiname auf `.unpack.zip`, wird die Datei zusätzlich entpackt und im auszuführenden Pfad und in den Argumenten wird die Zeichenkette `<UNPACKDIR>` durch das Verzeichnis, in das entpackt wurde, ersetzt. Dadurch können auch mehrere Dateien über die Fetch-Funktion heruntergeladen werden. Für die Archiv-Unterstützung wird die Bibliothek `SharpZipLib` von `IC#Code`, die unter der GPL-Lizenz verfügbar ist, verwendet.

Ist der auszuführende Pfad definiert, wird dieser anschließend ausgeführt und gewartet, bis sich der Prozess beendet. Ist eine Liste mit erlaubten Rückgabewerten definiert und enthält diese nicht den vom Prozess zurückgegebenen Wert, so wird die Aktion ignoriert und beim nächsten Installationslauf wiederholt. Andernfalls wird die erfolgreiche Aktion an den Server gemeldet (`pkgui` oder `pkguu`).

Abschließend wird eine per `fetch_url` heruntergeladene Datei und eventuell das Verzeichnis, in das sie entpackt wurde, wieder gelöscht. Der Client wartet eine konfigurierbare Zeitdauer, bis er die nächste Abfrage – beginnend bei der Authentifizierung – startet.

Fehlerbehandlung

Einmal gestartet, läuft der Client in einer Endlosschleife, die nur durch Senden von Ctrl-C in der Konsole oder vom Microsoft Dienste-Manager, wenn der Client als Dienst läuft, unterbrochen werden kann. Dementsprechend werden alle Fehler abgefangen und rufen zwar eine Warnung oder Fehlermeldung hervor, der normale Programmablauf wird aber nach einer definierbaren Wartezeit wieder aufgenommen. Dadurch ist sichergestellt, dass der Client immer läuft und überprüft, ob neue (De-)Installationen durchzuführen sind.

User-Interface

Nachdem der Client im Idealfall als Dienst gestartet wird, ist der Programmablauf für den Benutzer nicht mitzuverfolgen. Um ihn trotzdem über durchgeführte Installationen zu informieren, wurde ein separates User-Interface programmiert.

Dieses startet als grafische Anwendung bei der Anmeldung des Benutzers und legt ein Icon im Systemtray ab. Es verbindet sich mit dem bereits gestarteten Client-Dienst und fragt in regelmäßigen Abständen nach Neuigkeiten. Da diese Programme als eigenständige Prozesse laufen, muss dieser Informationsaustausch über einen Mechanismus zur Interprozesskommunikation (IPC) erfolgen. Dazu wurde die Remoting-Technik von Microsofts .NET Framework gewählt. Dies ist kein explizites IPC-Verfahren, Microsoft stellt aber das sogenannte ipc-Protokoll bereit, das die Kommunikation über named pipes abwickelt und speziell für diesen Zweck entwickelt wurde.

Schließt der Server also eine Installation oder Deinstallation erfolgreich ab, so legt er eine entsprechende Nachricht an, die vom Client abgerufen werden kann. Sobald der Client diese Nachricht erhält, zeigt er sie in einem Popup-Fenster am rechten unteren Bildschirmrand an.

Konfiguration

Die Konfiguration der Clientapplikation ist über eine INI-Datei möglich. Diese wird im Programmverzeichnis mit dem Namen `yasdc.ini` erwartet.

Eine Beispiel-Konfiguration, in der für alle Optionen die Standard-Werte gesetzt wurden, sieht wie folgt aus:

```
[YasdServer]
Host = localhost
HttpPort = 8000
YasdPort = 8001

[Delays]
PollDelay = 30
RegDelay = 30

[Directories]
FetchTemp = C:\TEMP
```

Auflistung 26 yasdc.ini

Die nachfolgende Tabelle erklärt alle möglichen Optionen und führt ihre Standard-Werte an:

Option	Default-Wert	Erklärung
[YasdServer]		
Host	localhost	Hostname oder IP-Adresse des yasd-Servers
HttpPort	8000	Port, auf dem der HTTP-Server läuft
YasdPort	8001	Port, auf dem der yasd-Server läuft
[Delays]		
PollDelay	30	Wartezeit zwischen den Installations-Zyklen
RegDelay	30	Wartezeit nach erfolgter Registrierung
[Directories]		
FetchTemp	C:\TEMP	Verzeichnis für temporäre Dateien

Tabelle 6 Erklärung der Parameter in yasdc.ini

Anmerkungen:

- Werden einzelne Optionen nicht angegeben, so werden automatisch die Default-Werte benutzt.
- Ist das in `FetchTemp` angegeben Verzeichnis nicht vorhanden, wird es gegebenenfalls angelegt.

Logging

Für die Protokollierung wurde auf die Bibliothek NLog zurückgegriffen. Diese erlaubt als Ziele sowohl die Konsole als auch eine Datei. Erstere wird verwendet, wenn der yasd-Client als Konsolenanwendung gestartet wird, letztere wenn er als Dienst gestartet wird. Die Protokolldatei wird im Ausführungsverzeichnis als `yasdc-log.txt` abgelegt.

Packaging

Um den Client in Form einer Setup-Datei anbieten zu können, wurde wie schon beim Server das „Nullsoft Installer System“ verwendet. Zwar werden vom C#-Compiler unter Windows ausführbare DLLs und EXE-Dateien erstellt, diese enthalten aber keinen nativen Code, sondern vorkompilierten Byte-Code. Zur Ausführung dieser Dateien wird daher der sogenannte Interpreter aus dem .NET Framework benötigt. Das Setupprogramm prüft, ob diese Vorbedingung bei der Installation bereits erfüllt ist; andernfalls leitet es den Benutzer auf die Downloadseite des .NET Frameworks weiter.

Ausgeliefert werden daher die DLL-Dateien der benötigten Bibliotheken, eine DLL mit den gemeinsamen Routinen des yasd-Clients (`yasdc.dll`), die Konsolenanwendung (`yasdcc.exe`) und der Windows-Dienst (`yasdc.d.exe`), das User-Interface (`yasdci.exe`) und die von .NET Remoting benötigten Interfacedefinitionen (`yasdremoting.dll`), sowie eine Default-Konfigurationsdatei (`yasdc.ini`) und ein Batch-Skript zur Kontrolle des Windows Dienstes (`yasdc.d.cmd`).

4 Tests

In einem umfangreichen Projekt wie einer Diplomarbeit sind Tests sehr wichtig. Das war uns von Anfang an bewusst und wir haben daher schon sehr früh mit ersten Tests begonnen. Trotzdem war es schwer, da es keinen unabhängigen Tester gab, wir alle an der Diplomarbeit arbeiteten und daher „betriebsblind“ waren.

Die Tests erfolgten in mehreren Schritten. Nach dem Schreiben eines Skriptes oder der Implementierung einer neuen Teilfunktion testete jeder seinen Teil ausführlich auf Fehler. Danach trafen wir uns alle paar Wochen und diskutierten den Fortschritt und mögliche Probleme. Dabei wurden den anderen neue Funktionen gezeigt. Anschließend bekam jeder die neue Version und konnte sie so für sich testen. Dadurch konnten wir sehr viele Fehler aufdecken, die ein einzelner nicht gefunden hätte.

Außerdem wurden die yasd Lösungen nach Erscheinen des ersten funktionsfähigen Prototyps in unseren Heimnetzwerken getestet, um Livebedingungen zu simulieren. Damit jeder über die gleiche Version des yasd Servers verfügte, wurde eine Windows XP Installation mit VMWare erzeugt. Dadurch konnten Neuerungen sofort eingebaut und den anderen zur Verfügung gestellt werden. Außerdem war kein eigener PC nötig und man könnte das Image mit dem kostenlosen VMWare Player starten. Außerdem wurden mehrere virtuelle Testclients mit verschiedenen Betriebssystemen (darunter Windows XP, Windows XP mit SP1, Windows XP mit SP2 und Windows 2000) erzeugt, um den yasd Client in mehreren Umgebungen zu testen.

Im März wurde dann ein richtiger PC mit dem yasd Server aufgesetzt und anschließend in der Schule aufgebaut. Dadurch konnten wir testen, ob es

Unterschiede zwischen einer VMWare Umgebung und einem Standard-PC gibt. In der HTL Rennweg ist das Netzwerk in mehrere VLANs unterteilt. Diese Konstellation findet man auch in den meisten Firmen. Nach ein paar Konfigurationsänderungen funktionierte unser yasd Server auch unter diesen Umständen und man konnte in der ganzen Schule AntiVir Updates über unseren Server durchführen oder sich Updates holen.

Im nächsten Schritt wollten wir unsere Lösung in einem richtigen Computerraum testen. Mit Mag. Andreas Fink fanden wir einen Saal-Administrator, der sich für unsere Lösung begeistern ließ. Er stellte uns das Medientechniklabor zur Verfügung, wo wir unseren Server aufbauten. Danach installierten wir unseren yasd Client probeweise auf den PCs und konnten Updates und Programme an die PCs ausliefern.

5 Quellenverzeichnis

[Kara2001] Karakas, Levent:

Unix Daemon Server Programming [online],
aktualisiert am 16. Mai 2001 [zitiert am 03. April 2007],
verfügbar im Internet:
<<http://www.enderunix.org/docs/eng/daemon.php>>

[MySQL2006] MySQL AB:

MySQL Licensing Policy [online],
aktualisiert am 16. März 2006 [zitiert am 10. Mai 2007],
verfügbar im Internet:
<<http://www.mysql.com/company/legal/licensing/>>

[Schr2005] Schroeder, Chad J.:

Creating a daemon the Python way [online],
aktualisiert am 03. Oktober 2005 [zitiert am 03. April 2007],
verfügbar im Internet:
<<http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/278731>>

[Viol2006] Violka, Karsten und Wittrock, Torsten:

Selfmade Service Pack
Windows-Updates ohne Internet-Verbindung installieren,
2006, c't magazin für computer technik, Nr. 23/2006, S. 202-206