Ein Netzwerk-, User- und Systemmanagementwerkzeug zur Unterstützung der Administration in kleineren bis mittleren Unternehmensnetzen

Diplomarbeit

Alexander Schreiber

Technische Universität Chemnitz

Fakultät für Informatik
Lehrstuhl Rechnernetze und verteilte Systeme

Ein Netzwerk-, User- und Systemmanagementwerkzeug zur Unterstützung der Administration in kleineren bis mittleren Unternehmensnetzen
Schreiber, Alexander. - 2002. - 79 S., 4 Abb.
Chemnitz, Technische Universität Chemnitz, Fakultät für Informatik, Diplomarbeit

eingereicht von:	Alexander Schreiber
	geboren am 23. Mai 1975 in Bad Langensalza
ausgegeben am:	15. März 2002
eingereicht am:	16. September. 2002
Betreuer:	Prof. Dr. Uwe Hübner

Aufgabenstellung:

Im Kontext eines kleinen bis mittleren heterogenen Netzes (Software/Hardware) soll nach Analyse und Bewertung vorhandener Lösungen und Hilfsmittel ein modulares System zur Unterstützung der Administration von Hosts, Nutzern und Software entworfen und implementiert werden. Da aufgrund des zeitlichen Rahmens einer Diplomarbeit nur eine begrenzte Lösung möglich ist, soll das System von vornherein modular und auf einfache Erweiterbarkeit ausgelegt werden. Im Rahmen der Arbeit sollen existierende Lösungsansätze und Hilfsmittel aus dem Bereich der Administrationshilfen untersucht und gegebenenfalls auf ihre Weiterverwendbarkeit in diesem Projekt geprüft werden. Die dabei gewonnenen Erkenntnisse (u.a. in Bezug auf Flexibilität und Wiederverwendbarkeit) sollen in das zu erstellende System einfließen.

Inhaltsverzeichnis

1 Motivation **8**

2 Begriffsklärung **9**
2.1 Admin . . . 9
2.2 Überwachung . . . 9
2.3 Zeitstempel . . . 9

3 Strukturierung eines Administrationssystems **10**
3.1 Aufgabenbereiche . . . 10
3.2 Verwaltung . . . 10
3.3 Konfiguration . . . 11
3.4 Überwachung . . . 11
3.5 Wissensmanagement . . . 11

4 Verwaltung **13**
4.1 Usermanagement . . . 13
4.1.1 Authentisierung via SMB . . . 14
4.1.2 Eingriffe in Windows . . . 14
4.1.3 LDAP und Kerberos . . . 14
4.1.4 Schlußfolgerungen für diese Arbeit . . . 14
4.2 DNS-Management . . . 15
4.2.1 Strukturierung des DNS-Namensraumes . . . 15
4.2.2 Wahl der DNS-Serversoftware . . . 16
4.2.3 Administration des DNS-Servers . . . 16
4.3 DHCP-Management . . . 17
4.4 Software-Lizenzmanagement . . . 17

5 Systemkonfiguration **18**
5.1 Plattformen . . . 18
5.1.1 UNIX . . . 18
5.1.2 Windows . . . 19
5.2 Werkzeuge zur automatisierten Konfiguration . . . 20
5.2.1 PIKT . . . 20
5.2.2 Cfengine . . . 21

6 Überwachung von Systemen **23**
6.1 Kategorien von Überwachungssystemen . . . 23
6.2 Überwachung und Alarmierung . . . 24
6.2.1 Nagios™ . . . 24
6.2.2 Big Brother . . . 25
6.2.3 Big Sister . . . 26
6.3 Überwachung und Graphengenerierung . . . 27
6.3.1 mrtg . . . 27
6.3.2 RRDtool . . . 28
6.3.3 Cricket . . . 28
6.3.4 cacti . . . 29

7 Wissensmanagement **30**
7.1 Ansätze zum Wissensmanagement 30
7.1.1 Grundsätzliche Überlegungen 30
7.1.2 Dokumentation 30
7.1.3 Knowledgebase 30
7.1.4 Webbasierte, teamorientierte Dokumentationssysteme 31
7.1.5 Troubleticket-Systeme 31
7.2 Verschiedene Systeme im Test 32
7.2.1 Request Tracker 32
7.2.2 TWiki 33
7.2.3 unixops 34
7.3 Schlußfolgerungen zum Wissensmanagement 34

8 SCASS - System Configuration and Administration Support System **35**
8.1 Ansatz 35
8.2 Design von SCASS 35
8.2.1 Modulkonzept 35
8.2.2 Strukturierung 36
8.2.3 Konzept Frontendkomponente 36
8.2.4 Konzept Datenspeicherungskomponente 39
8.2.5 Konzept Backendkomponente 40
8.2.6 Modulübergreifende Standards und das Infrastrukturkonzept 41
8.2.7 Konzept DNS-Modul 46
8.2.8 Konzept Softwarelizenzmanagement-Modul 50
8.2.9 Konzept DHCP-Modul 55
8.3 Implementation 59
8.3.1 Implementation der Frontendkomponente 59
8.3.2 Infrastruktur der Frontendkomponente 61
8.3.3 Frontendkomponente DNS-Modul 62
8.3.4 Frontendkomponente DHCP-Modul 62
8.3.5 Frontendkomponente Softwarelizenzmanagement-Modul 63
8.3.6 Implementation der Datenspeicherungskomponente 64
8.3.7 Implementation der Backendkomponente 65
8.3.8 Infrastruktur der Backendkomponente 66
8.3.9 Backendkomponente DNS-Modul 66
8.3.10 Backendkomponente DHCP-Modul 66
8.4 Installation 68
8.4.1 Installationsvorraussetzungen für SCASS 68
8.4.2 Installation von SCASS 68

9 Fazit und Ausblick **72**
9.1 Bewertung 72
9.2 Weiterentwicklung von SCASS 72

10 Erklärungen **78**

Tabellenverzeichnis

1 Visualisierung der Komponenten- und Modulstruktur 36
2 SCASS-Infrastruktur, Datendefinition für die Tabelle `scass_fes_modules` . . 44
3 SCASS-Infrastruktur, Datendefinition für die Tabelle `scass_fes_menu` 44
4 SCASS-Infrastruktur, Datendefinition für Tabelle `scass_table_description` 44
5 SCASS-Infrastruktur, Datendefinition für die Tabelle `scass_config` 44
6 DNS-Modul, Datendefinition für die Tabelle `dns_zone` 48
7 DNS-Modul, Datendefinition für die Tabelle `dns_mapping` 48
8 DNS-Modul, Datendefinition für die Tabelle `dns_alias` 48
9 DNS-Modul, Datendefinition für die Tabelle `dns_special_types` 49
10 DNS-Modul, Datendefinition für die Tabelle `dns_special` 49
11 Softwarelizenzmanagement-Modul, Datendefinition für `lm_platform` 53
12 Softwarelizenzmanagement-Modul, Datendefinition für `lm_vendor` 53
13 Softwarelizenzmanagement-Modul, Datendefinition für `lm_class` 53
14 Softwarelizenzmanagement-Modul, Datendefinition für `lm_license` 53
15 Softwarelizenzmanagement-Modul, Datendefinition für `lm_software` 54
16 Softwarelizenzmanagement-Modul, Datendefinition für `lm_installation` . . . 54
17 DHCP-Modul, Datendefinition für `dhcp_net` 57
18 DHCP-Modul, Datendefinition für `dhcp_host` 57
19 DHCP-Modul, Datendefinition für `dhcp_range` 57

Abbildungsverzeichnis

1 SCASS-Infrastruktur, visuelle Darstellung des Datenbankmodells 46
2 DNS-Modul, visuelle Darstellung des Datenbankmodells 49
3 Softwarelizenzmanagement-Modul, Visualisierung des Datenbankmodells . . . 55
4 DHCP-Modul, grafische Darstellung der Datenbankstruktur 58

1 Motivation

Die vorliegende Arbeit beschäftigt sich mit der Entwicklung eines Systems zur Unterstützung der Systemadministration in kleineren bis mittleren Unternehmensnetzen. Das zu erstellende System richtet sich in seiner Zielstellung primär an Netze ohne vorhandene zentrale Administrationshilfen, aber auch an vergleichbare kleinere bis mittlere Netzstrukturen, wie zum Beispiel den lokalen Netzbereich einer Fakultät.

Oftmals werden gerade diese Netze schrittweise und bedarfsgesteuert ausgebaut, von sehr kleinen, problemlos überschaubaren Anfängen zu zunehmend umfangreicheren Strukturen. Während die anfänglich sehr kleinen Strukturen (ein, zwei Server, ein Dutzend Workstations) meist von einem Mitarbeiter „nebenbei“ administriert werden können, so wird dieser Ansatz mit dem Wachstum der Struktur zunehmend schwieriger und die Gefahr, daß die Übersicht verloren geht, wächst.

Für das Management von Netzen gibt es bereits eine umfangreiche Palette verfügbarer Software, die verspricht, die Administration und die Überwachung der Systeme zu vereinfachen. Dabei werden verschiedene Ansätze verfolgt, um diese Aufgaben weitestgehend zu zentralisieren, zu automatisieren und zu vereinfachen. Während manche dieser Werkzeuge sich noch in frühen Entwicklungsstadien befinden, haben sich andere bereits seit Jahren im Einsatz bewährt. Aufgrund ihrer unterschiedlichen Ansätze eignen sich natürlich nicht alle Werkzeuge für jede Umgebung. Im Rahmen der vorliegenden Arbeit sollen vorhandene Werkzeuge verglichen, ihre Verwendbarkeit im Rahmen eines Administrationssystems für kleinere und mittlere Netze geprüft und gegebenfalls weitere Software zur Erfüllung von Teilaufgaben entwickelt werden.

Bei der Betrachtung vorhandener Softwarelösungen beschränkt sich der Autor aus mehreren Gründen auf den Bereich der freien Software. Zum einen läßt sich freie Software aufgrund der nicht anfallenden Lizenzkosten relativ kostenneutral einsetzen, eine Tatsache, die den an hohem Einspardruck leidenden Einrichtungen und Unternehmen die Entscheidung für den Einsatz erleichtert. Zum anderen besteht in der freien Verfügbarkeit der Quellen der Vorteil der Herstellerunabhängigkeit. Bei kommerzieller Software ist man als Kunde üblicherweise darauf angewiesen, daß der Hersteller die Software auch in Zukunft weiterhin unterstützt, weiterentwickelt oder zumindest zeitnah Fehlerkorrekturen (Bugfixes), insbesondere bei sicherheitsrelevanten Fehlern, bei Bedarf zur Verfügung stellt. Im bisherigen Einsatz hat sich freie Software gerade im Bereich der laufenden Fehlerkorrektur als sehr zuverlässig und reaktionsschnell erwiesen. Während man bei kommerzieller Software mit dem Ende der Verfügbarkeit des Produktes (zum Beispiel weil der Hersteller die Entwicklung eingestellt hat) entweder mit zunehmend veralteter Software arbeiten oder aber auf ein anderes Produkt (kostenintensiv) migrieren muss, kann man bei freier Software im schlimmsten Falle die Pflege der verwendeten Software im eigenen Haus weiterführen. Meist wird sich aber in solchen Fällen ein Entwickler oder ein Entwicklerteam finden, welches die Software weiterpflegt.

2 Begriffsklärung

An dieser Stelle soll die Gelegenheit genutzt werden, verschiedene, im folgenden Text genutzte, Begriffe zu klären und gegebenenfalls zu definieren, da einerseits viele Begriffe im normalen Sprachgebrauch mit mehreren, sich zum Teil überschneidenden oder widersprechenden Bedeutungen belegt sind und andererseits im Folgenden einige Begriffe im Interesse der Lesbarkeit nicht jedesmal voll ausdefiniert werden sollen. Aus Gründen der sprachlichen Vereinfachung wird üblicherweise die männliche Begriffsform verwendet. Wenn im laufenden Text nicht anders erwähnt, ist von den hier ausgeführten Begriffserklärungen auszugehen.

2.1 Admin

Es wird durchweg von „dem Admin“ gesprochen, auch wenn sich dies nicht notwendigerweise auf eine, vollständig mit der Systemadministration beschäftigte, Person bezieht. Der Begriff soll sowohl Adminteams als auch teilweise als Admin agierende Personen einschliessen. Dabei werden die Begriffe „Admin“ und „Sysadmin“ austauschbar verwendet.

2.2 Überwachung

Der Begriff „Überwachung“ ist mit vielen unterschiedlichen Bedeutungen belegt. Im folgenden soll darunter die Überwachung des Betriebs sowie der Betriebsparameter von Netzwerk- und Hostsystemen verstanden werden.

2.3 Zeitstempel

Unter einem Zeitstempel wird eine vollständig Datums- und Uhrzeitangabe verstanden. Das verwendete Darstellungsformat ist dabei kontextabhängig. Wenn der Kontext kein anderes Format nahelegt, ist jedoch von einer Darstellung nach ISO8601 auszugehen.

3 Strukturierung eines Administrationssystems

3.1 Aufgabenbereiche

Die typischen Aufgaben der Systemadministration umfassen Konfiguration, Verwaltung und Überwachung der eingesetzen Systeme, sowie das Management des administrativen Wissens. Weitere Aspekte, wie zum Beispiel Planung und Beschaffung, sind nicht Thema dieser Arbeit.

Unter Konfiguration fallen sowohl die Erstkonfiguration neu aufgesetzter Systeme, als auch die Pflege und Aktualisierung vorhandener Systeme. Die Verwaltung der verwendeten Systeme, Software und weiteren Objekte ist die zweite Teilkomponente. Und schließlich sind die laufenden Systeme in ihren Betriebsparametern zu überwachen, um Abweichungen vom Normalzustand und so die Notwendigkeiten von Eingriffen rechtzeitig festzustellen. All dies erfordert entsprechendes Wissen, welches sich entweder über mehrere Personen verteilt oder zum überwiegenden Teil in einer Person, dem lokalen Sysadmin, zusammengefaßt ist. Mit dem Ausfall einer oder mehrerer solcher Personen, zum Beispiel durch Krankheit oder Wechsel der Arbeitsstelle, geht somit meist ein temporärer oder permanenter Ausfall von für den Betrieb des Netzes wichtigem Wissen einher. Da dies keine akzeptable Situation darstellt, wird eine geeignete Möglichkeit zur Hinterlegung und Verwaltung dieses „Betriebswissens“ benötigt.

3.2 Verwaltung

Die Verwaltung eines Netzes umfaßt zahlreiche verschiedene Aspekte, es sind Nutzer und Zugriffsrechte, Systeme, Software, Namenszuordnungen und einiges mehr zu verwalten.

Zu den offensichtlichen Teilaspekten gehört sicher das Verwalten der Nutzeraccounts. Die Verwaltung von Nutzeraccounts wird sich in den seltensten Fällen auf die Einrichtung oder Löschung der Accounts beschränken, meist wird mit der Einrichtung eines Nutzeraccounts ein ganzer Prozess abgearbeitet, der u.a. das Einrichten von Nutzeraccounts in verschiedenen Systemen (UNIX-Accounts, Windows-Accounts, Lotus Notes Accounts, ...), Vergeben weiterer Zugriffsrechte (z.B. durch Aufnahme in Gruppen mit besonderen Rechten), Einrichten von Homeverzeichnissen, Einrichten von Mailzugängen und, je nach Umgebung, noch verschiedene andere Schritte umfaßt. Diese Schritte sind bei der Löschung des Nutzeraccounts natürlich auch wieder auf geeignete Weise rückgängig zu machen.

Neben der Verwaltung der Nutzer ist die Verwaltung der Hostnamen und IP-Adressen eine der üblichen Aufgaben in einem Netz. Typischerweise wird man zu diesem Zweck einen lokalen Nameserver betreiben um eine zentrale Namensauflösung zur Verfügung zu stellen. Die Zuordnung der IP-Adressen zu den einzelnen Host kann entweder manuell, durch festes Eintragen der IP-Adresse in der Konfiguration des Hostes, oder automatisch, durch den Einsatz eines DHCP[1]-Servers geschehen.

Ein oft vernachlässigter Teil der Verwaltung ist die Softwareverwaltung. Oftmals wird Software bedarfsgesteuert beschafft und installiert, ohne das wirklich eine klare Übersicht darüber besteht, welche Software und Softwarelizenzen bereits vorhanden und welche davon in Verwendung sind. Während bei freier Software dieser Aspekt eher unproblematisch ist und ein solcher Überblick lediglich eine Vereinfachung der Administrationstätigkeit darstellt, kann bei kommerzieller Software ein mangelnder Überblick über den Softwarestand zu finanziellen Konsequenzen führen. Trotz des zunehmenden Einsatzes freier Software sind

[1]Dynamic Host Configuration Protocol, ein Verfahren um einem Host, der sich durch die Hardwareadresse seiner Ethernetkarte identifiziert, automatisch seine Netzwerkkonfiguration zu übermitteln

viele Unternehmen und Einrichtungen nach wie vor auf den Einsatz kommerzieller Software angewiesen. Auch wenn - ab eines entsprechend umfangreichen Einsatzes - firmen- beziehungsweise einrichtungsweite Lizenzen (sogenannte „site licenses“) möglich sind, so wird kommerzielle Software doch meist in Form einzelner oder zumindest abgezählter Lizenzen beschafft. Ein Mangel an Überblick über die eingesetzte Software kann in dem Fall zu zwei Situationen führen: Überlizensierung und Unterlizensierung. Im Falle der Überlizensierung sind mehr Lizenzen vorhanden, als sich im Einsatz befinden, es werden also unter Umständen Mittel unnötig gebunden. Wesentlich unangehmer können die Konsequenzen bei Unterlizensierung, also beim Einsatz von mehr Lizenzen, als überhaupt vorhanden sind, werden. Von den Softwareherstellern wird diese Situation meist als Einsatz von Raubkopien interpretiert und bei Bekanntwerden entsprechend zivilrechtlich verfolgt, was sowohl Kosten verursacht als auch Imageschäden mit sich bringen kann. Die Notwendigkeit, mit geeigneten Mitteln einen Überblick über die eingesetzte Software zu behalten, ist somit offensichtlich.

3.3 Konfiguration

Bei den zu konfigurierenden Einheiten handelt es sich typischerweise um Hosts, die entweder als Workstations oder als Server dienen. Deren Konfiguration läßt sich in drei zeitliche Abschnitte unterteilen:

- initiale Konfiguration bei der Installation,
- Wartung und Anpassung im laufenden Betrieb,
- Neukonfiguration im Rahmen eines Systemupgrades,

3.4 Überwachung

Die Überwachung eines Netzes umfaßt sowohl die Überwachung der Betriebsparameter des Netzwerks selbst (Erreichbarkeit von Hosts, Netzlast, Funktion von Netzdiensten) als auch die Überwachung der Betriebsparameter (Systemlast, Auslastung der Filesysteme, ...) der beteiligten Hosts. Eine solche Überwachung ist im Rahmen der Gewährleistung eines störungsfreien Betrieb des Netzes als Gesamtsystem notwendig, um u.a. Störungen (z.B. ausgefallene Dienste) zu erkennen und bei sich abzeichnenden Problemen (z.B. volllaufende Dateisysteme) rechtzeitig einschreiten zu können. Weiterhin liefert eine solche Überwachung, wenn sie entsprechend ausgelegt ist, auch Informationen für die zukünftige Planung des Systems aus Netzwerk und beteiligten Hosts, z.B. für notwendige Kapazitätserweiterungen.

3.5 Wissensmanagement

Zusätzlich zum normalen „Arbeitswissen“ eines Admins fällt beim Betrieb eines Netzes immer ein für dieses Netz spezifisches Wissen an. Dies umfaßt unter anderem die lokalen Standardprozeduren zur Abarbeitung wiederkehrender Vorgänge (Einrichtung eines neuen Nutzers/Hosts, Behandlung der Backups, ...), lokale Besonderheiten der Konfiguration, erprobte Problemlösungen und vieles mehr. Dieses Wissen ist oftmals nur im Kopf sowie den Notizen der entsprechenden Mitarbeiter „gespeichert“ und selten an geeigneter Stelle zentral hinterlegt. Typische Gründe dafür sind meist:

- „das sind doch nur Kleinigkeiten, die weiß man“,

- die altbekannte Unlust am Schreiben von Dokumentation,
- keine Zeit zum Schreiben von Dokumentationen.

Selbst wenn für den Admin dieses angesammelte Wissen immer in vollem Umfang abrufbar ist - eine angesichts normaler menschlicher Schwächen eher kühne Annahme - so führt dies spätestens beim Verlust der Verfügbarkeit des entsprechenden Mitarbeiters (nicht erreichbar, hat Firma verlassen, ist krank, ...) unter Umständen zu erheblichen Problemen.

4 Verwaltung

Unter Verwaltung sollen hier vorwiegend die Elemente betrachtet werden, die ihrer Natur nach eine zentrale Verwaltung erfordern oder zumindest sehr stark nahelegen. Üblicherweise fallen in diese Kategorie unter anderem die Bereiche:

- DNS-Management,
- IP-Adress-Management,
- Usermanagement,
- Mailverwaltung,
- Profile für Hostkonfiguration.

4.1 Usermanagement

Das Usermanagement spielt in einem Netz eine sehr wichtige Rolle. Der eigene Account ist für die Nutzer der Systeme der Schlüssel zum Zugang, der die Nutzung der Arbeitsplätze und verschiedenen Dienste (z.B. Mail, gesicherte Webseiten, ...) erst ermöglicht. Gerade in kleineren Netzen erfolgt die Verwaltung der Nutzeraccounts oftmals noch von Hand, die Accounts werden vom Sysadmin oder einem Mitarbeiter, der zusätzlich diese Aufgabe erfüllt, von Hand angelegt und gepflegt. Meist umfaßt dies nicht nur das einfache Anlegen eines Accounts auf einem System, sondern z.B. auch das

- Anlegen des Accounts auf einem entsprechenden Server (z.B. NIS[2]-Master, PDC[3]),
- Anlegen von Mailkonfigurationen für den Account,
- Anlegen des Homeverzeichnisses,
- Einbinden in Backupsysteme,
- Anlegen weiterer resultierender Accounts (Zugang zu geschützten Webseiten, Nutzungsaccounts für spezielle Softwarepakete, ...).

Bei einem nicht unüblichen Plattformmix müssen z.B. 4 und mehr Accounts (UNIX-Account, Windows-Account auf PDC, AFS-Account, Accountdaten in Samba, Accounts für spezielle Software) für einen einzigen Nutzer angelegt werden. Dies und die weiteren dazugehörigen Prozeduren zusammenzufassen und zu automatisieren würde eine deutliche Arbeitserleichterung für den damit beschäftigten Mitarbeiter bedeuten. Es gibt auch durchaus einige Werkzeuge, die Teilaufgaben daraus automatisieren und die man geeignet miteinander zu einer Gesamtlösung verbinden könnte.

Doch bei näherer Überlegung zeigt sich, daß dies der falsche Ansatz ist. Zwar kann an dieser Stelle mit einem automatisierten Werkzeug der Arbeitsaufwand reduziert werden, doch dies ist ein Kurieren der Symptome. Die offensichtlich bessere Lösung besteht darin, das Usermanagement möglichst zu vereinheitlichen. Es empfiehlt sich also, Möglichkeiten zur Vereinheitlichung des Usermanagements zu finden und dann über administrative Werkzeuge dafür nachzudenken.

---

[2]Network Information System
[3]Primary Domain Controller

4.1.1 Authentisierung via SMB

Mit der SMB[4]-Authentisierung bestehen mehrere Möglichkeiten zum Aufbau eines zentralen Usermanagements. Ein Ansatz besteht darin, den Windows NT PDC zu nutzen und alle Systeme, sowohl Windows – für welches dies den Normalfall darstellt – als auch UNIX diesen Dienst als Authentisierungsserver zu benutzen zu lassen. In [Gre02] wird ein Aufbau einer solchen Lösung beschrieben. Dabei wird unter Linux das in [Air00] implementierte PAM [5]-Modul pam_smb verwendet um eine Authentisierung von UNIX-Usern gegen einen Windows PDC zu ermöglichen.

Ein anderer Ansatz besteht darin, mit Samba [Tea02] auf Unix einen PDC aufzubauen, gegen den dann die Windows-Maschinen authentisieren. Der Samba-PDC seinerseits benutzt ein geeignetes System zur Authentisierung, wofür es mehrere Möglichkeiten gibt, zum Beispiel LDAP[6], welches dann auch von anderen UNIX-Systemen zur Authentisierung herangezogen werden kann.

4.1.2 Eingriffe in Windows

Weitere Ansätze greifen in das NT Anmeldesystem ein, z.B. ein MS-GINA Ersatz [Sco97] und das darauf aufsetzende NISGINA [Wil97], um NT-Accounts auf Grundlage von UNIX-Accounts zu verwalten. Diese Lösung hat jedoch zahlreiche, auch sicherheitsrelevante Probleme. So wird in der Originalimplementierung [Sco97] auf FTP zur Verifikation der Passwortdaten zurückgegriffen. Noch störender ist die Tatsache, das diese Implementation praktisch „schleichend“ die UNIX-Nutzerdatenbank auf den Windows-Arbeitsplätzen nachbaut, eine wirklich zentrale Lösung ist das also nicht.

4.1.3 LDAP und Kerberos

Dies sind aber letztlich auch nur teilweise taugliche Lösungen. Am erfolgversprechendsten ist nach Ansicht des Autors ein derzeit in Entwicklung befindliches System[Pet02], welches vorraussichtlich auf LDAP und Kerberos setzen wird. Dabei wird darauf hingearbeitet, Nutzer auf Windows 2000/XP gegen AFS-Kerberosserver zu authentifizieren und gegen LDAP zu authorisieren. Dieser Ansatz verspricht nicht nur eine große Flexibilität, sondern auch ein wirklich plattformübergreifendes Usermanagement. Kerberos hat sich praktisch als *das* portable und sichere Authentisierungssystem durchgesetzt. LDAP wiederum bietet eine sehr große Flexibilität, die Authorisierung auch in größeren verteilten Strukturen zu verwalten. Sowohl Kerberos als auch LDAP werden von verschiedenen UNIX-Plattformen gut unterstützt, so daß dieser Ansatz aus einem System sowohl – mittels Kerberos – die Authentisierung als auch – mittels LDAP – die Authorisierung für Windows 2000/XP *und* die UNIX-Plattformen zu verwalten verspricht.

4.1.4 Schlußfolgerungen für diese Arbeit

Anhand dieser Ergebnisse entschied sich der Autor, den Bereich des Usermanagements nicht weiter zu verfolgen. Bei direkter Unterstützung vorhandener Infrastruktur ist u.U. ein sehr

[4] Server Message Block, eine proprietäre von Microsoft für Windows verwendete Protokollsammlung
[5] Pluggable Authentication Modules, eine flexible Authentisierungsarchitektur für UNIX
[6] Lightweight Directory Access Protocol

hoher Aufwand notwendig, um ein neues Managementwerkzeug dafür zu schaffen oder vorhandene Werkzeuge hinreichend weit auf die zahlreichen Besonderheiten anzupassen – die sich oftmals im Betrieb eines Netzes über einen längeren Zeitraum herausbilden. Der Entwurfs-, Implementations- und Testaufwand, der für eine wirklich umfassende Lösung des Usermanagements notwendig wäre, übersteigt die im Rahmen dieser Arbeit – neben den anderen zu bearbeitenden Gebieten – vorhandene Zeit erheblich und kann somit nicht stattfinden. Der Autor verweist mit 4.1.3 sowie den für [Pet02] erwarteten Ergebnissen auf eine Lösung, die seiner Meinung nach ein sehr großes Potential haben kann.

4.2 DNS-Management

DNS-Management ist die Pflege des Domain Name Service, eines verteilten Verzeichnisdienstes, dem unter anderem die Zuordnung von FQDNs[7] zu IP-Adressen („forward mapping“) und umgekehrt („reverse mapping“) obliegt. Mittels DNS können auch weitere Informationen verwaltet werden, zum Beispiel die für einen Host oder eine Domain zuständigen Mailserver. Nicht zuletzt benutzt das DNS-System sich selbst für seine eigene Organisation – im DNS wird auch verwaltet, welcher DNS-Server für welche Domain zuständig ist.

Im Kontext lokaler Netze ist DNS normalerweise in zweifacher Hinsicht relevant:

- als Verzeichnisdienst, der aufgrund der Verwendbarkeit von Hostnamen (z.B. für den Webserver der TU Chemnitz `www.tu-chemnitz.de`) anstelle von IP-Adressen (für den vorgenannten Webserver `134.109.132.109`) die Nutzung des Internet auf einfache Weise erst ermöglicht,
- als lokaler Verzeichnisdienst, der die Zuordnung von Hostnamen zu den IP-Adressen der lokalen Rechner verwaltet und damit den Zugriff auf die lokalen Hosts vereinfacht.

Im Rahmen dieser Arbeit ist nur der zweite Teil, also der lokale Verzeichnisdienst von Interesse. Wenn man sich für das Aufsetzen eines lokalen DNS-Servers für das eigene Netz entschieden hat, sind vor dem Aufsetzen verschiedene Fragen zu klären, die – je nach Entwicklung des Netzes – zum Teil schon vorher abgearbeitet wurden:

- Strukturierung des DNS-Namensraumes,
- Wahl der DNS-Serversoftware,
- Administration des DNS-Servers.

4.2.1 Strukturierung des DNS-Namensraumes

Wie bereits in [Tom01] detaillierter ausgeführt wird, kann eine ungünstige oder falsche Strukturierung von Namensräumen zu erheblichen, zum Teil nur unter großem Aufwand zu beseitigenden, Problemen führen. Daher sollten diese Entscheidungen mit besonderer Sorgfalt getroffen werden.

Für die Strukturierung des lokalen DNS-Namensraumes ist dabei insbesondere zu klären:

- Welcher Domainname soll für das lokale Netz verwendet werden?
- Wie soll diese Domain strukturiert werden?
- Nach welchen Regeln sollen die einzelnen Namen vergeben werden?

[7] FQDN = Fully Qualified Domain Name, vollständiger Hostname, z.B. thishost.example.com

4.2.2 Wahl der DNS-Serversoftware

Als relativ weitverbreiteter Standard für DNS-Serversoftware hat sich der ISC BIND[8] [ISC02a] durchgesetzt. Er wird üblicherweise als Standardpaket für das Aufsetzen von DNS-Servern eingesetzt, befindet sich in sehr vielen Installationen im Einsatz und ist recht umfangreich dokumentiert, u.a. in [Pau01].

Aus verschiedenen Gründen – u.a. ist BIND in der Vergangenheit wiederholt durch Sicherheitslöcher aufgefallen – entstanden im Laufe der Zeit weitere Implementationen von DNS-Serversoftware. Eine der bekannteren ist das djbdns Paket [Ber02a], welches unter anderem einen DNS Server (tinydns) und einen DNS-Cache (dnscache) enthält. Das djbdns Paket wurde – unter dem Eindruck der wiederholten Sicherheits- und Stabilitätsprobleme von BIND zu dieser Zeit – mit dem Ziel der Sicherheit und Zuverlässigkeit entwickelt. Nach den vorliegenden Erfahrungen ist dieses Ziel auch erreicht worden. Störend wirkt beim Aufsetzen der Software aus dem djbdns Paket, daß sie nicht darauf ausgelegt ist, in die üblichen UNIX-Mechanismen zur Verwaltung von Diensten eingebunden zu werden, sondern von einem eigenen System ausgeht.

Eine weitere DNS-Server Implementation is MaraDNS [Tre02]. Die Entwicklungsziele von MaraDNS sind Sicherheit, freie Verfügbarkeit der Software (explizit als Public Domain lizensiert) und minimalistisches Design. Zum derzeitigen Zeitpunkt ist MaraDNS vollständig einsetzbar, sowohl als authoritativer Nameserver als auch als DNS Cache.

Zusätzlich zu den genannten existieren natürlich noch verschiedene andere Implementationen, auf die jedoch hier nicht weiter eingegangen werden soll, da sie sich teilweise in unterschiedlichen Stadien begrenzter Nutzbarkeit befinden: Betaphase oder früher, Entwicklung eingestellt, keine hinreichende Dokumentation.

4.2.3 Administration des DNS-Servers

Schließlich bleibt noch zu entscheiden, auf welche Weise der DNS-Server administriert und seine Konfiguration gepflegt werden soll. Dazu bieten sich zwei Vorgehensweisen an: manuell oder werkzeuggestützt. Die manuelle Variante bietet den Vorteil, daß keine weitere Software (und damit Installation, Einarbeitung und Pflege dieser Software) notwendig ist, trägt aber das Problem in sich, daß die Pflege der Konfiguration, mit zunehmendem Umfang der Konfigurationsdaten, aufwendiger und fehleranfälliger wird. Bei der werkzeugbasierten Pflege ist das Hauptproblem die Auswahl eines geeigneten Werkzeugs zur Unterstützung der Pflege der Konfiguration.

Sinnvollerweise sollte das entsprechende Werkzeug über eine Weboberfläche bedienbar sein, da auf diese Weise lediglich ein, auf einem beliebigem Rechner laufender, Webbrowser zur Bedienung des Werkzeugs benötigt wird.

Es gibt mehrere Administrationswerkzeuge für DNS-Server, die ein Web-Interface verwenden. Der überwiegende Teil dieser Werkzeuge geht sogar soweit, eine Datenbank als Speicher für die Konfigurationsdaten einzusetzen. Leider sind die vorhandenen und dem Autor bekannten Werkzeuge alle mit verschiedenen Mängeln behaftet:

- sie sind überwiegend zu spezifisch, sind nur als Werkzeug für einen DNS-Server geschrieben und unterstützen nur diesen,

[8] Berkeley Internet Name Domain

- sie nutzen die Vorteile einer datenbankbasierten Speicherung der Konfigurationsdaten nicht aus.

Aus diesen Gründen entschloß sich der Autor, im Rahmen dieser Arbeit ein webbasiertes, datenbankgestütztes Administrationswerkzeug für DNS-Server zu entwickeln, welches in seiner Konzeption die vorher genannten Nachteile vermeiden soll.

4.3 DHCP-Management

Zur Vereinfachung der Netzwerkkonfiguration empfiehlt sich der Einsatz von DHCP[9], womit die am Netzwerk beteiligten Hosts ihre Netzwerkkonfiguration automatisch von einem zentralen DHCP-Server beziehen können. Für diesen Dienst gibt es im Wesentlichen zwei freie Implementationen:

- den ISC DHCP Server [ISC02b], als die Standardimplementation dieses Dienstes sowie
- den udhcp Server [Ram02], als sehr kompakte Implementation für embedded Systeme.

Für den ISC DHCP Server existieren verschiedenste Frontends, die auch die vorgenannte Anforderungen an eine Weboberfläche erfüllen. Dabei sind die verschiedenen Administrationswerkzeuge sehr unterschiedlich, von einfachen Statusanzeigen bis hin zur Konfiguration des DHCP Servers über eine Weboberfläche. Aufgrund der Vorherrschaft des ISC DCHP Servers sind alle Administrationswerkzeuge, die direkt die Konfiguration des DHCP Servers ermöglichen, auf diesen Server hin entwickelt, eine Unterstützung für andere DHCP Server ist nicht vorhanden. Die vorhandenen Werkzeuge benutzen als Datenspeicher entweder einfache Textdateien, arbeiten direkt mit den DHCP Server Konfigurationsdateien oder verwenden eigene „Datenbanken“. Die Möglichkeiten der Konfigurationsverwaltung in einer Datenbank nutzt keines der Werkzeuge, so daß sich der Autor entschloß, eine Beispiellösung dafür zu implementieren.

4.4 Software-Lizenzmanagement

Bereits in 3.2 wurde auf die Notwendigkeit und die Vorteile einer Softwareverwaltung hingewiesen. Die Suche nach einfachen, freien vorhandenen Lösungen für diesen Bereich blieb jedoch von wenig Erfolg gekrönt. Es zeigt sich an dieser Stelle, daß das Problem der Verwaltung von Softwarelizenzen im Bereich freier Software aus offensichtlichen Gründen nicht relevant ist – und somit auch keine bekannte Software dafür vorhanden ist. Im typischen Firmennetz dominiert aber eine zumindest gemischte Umgebung aus freier und kommerzieller Software, daher muß das Problem auf geeignete Weise behandelt werden. Eine einfache Verwaltungshilfe für Softwarelizenzen soll daher im Rahmen dieser Arbeit erstellt werden.

[9]Dynamic Host Configuration Protocol [Dro97a] [Dro97b]

5 Systemkonfiguration

5.1 Plattformen

Die üblichen Plattformen, die für eine automatisierte Konfiguration in Frage kommen, lassen sich grundsätzlich in drei Gruppen einteilen:

- UNIX (mit allen seinen unterschiedlichen Implementationen),
- Windows (primär Windows NT, Windows 2000 und Windows XP),

Die vorher genannten Plattformgruppen decken den bei weitem überwiegenden Teil der üblicherweise eingesetzen Systeme ab, daher soll im Rahmen dieser Arbeit auf weitere verbreitete System – wie zum Beispiel MacOS/MacOS X[10] – nicht weiter eingegangen werden.

5.1.1 UNIX

Die verschiedenen UNIX-Plattformen unter einer Kategorie gemeinsam als „UNIX" zusammenzufassen ist an dieser Stelle durchaus zulässig, da trotz aller implementationsspezifischen Unterschiede die zugrundeliegenden Konzepte, Systemstrukturierungen und APIs hinreichend weit übereinstimmen.

Gerade der bei UNIX-Plattformen übliche, uneingeschränkte Remotezugang erleichtert und vereinfacht die Administration der Systeme erheblich. Zudem bieten UNIX-Systeme oftmals schon im Rahmen der Basisinstallation sehr mächtige Werkzeuge für Administration, Automatisierung und Diagnose.

Trotz der großen Bandbreite an UNIX-Implementationen läßt sich der Umfang der zu betrachtenden Systeme auf einige wenige, den überwiegenden Anteil der im Einsatz befindlichen UNIX-Systeme abdeckenden, Plattformen reduzieren:

- Linux, mit verschiedensten Distributionen,
- die freien BSD-Versionen FreeBSD, NetBSD, OpenBSD,
- SUN Solaris,
- HP HP-UX,
- IBM AIX.

Während sich die genannten kommerziellen UNIX-Plattformen (Solaris, HP-UX, AIX) und die freien BSD-Varianten (FreeBSD, NetBSD, OpenBSD) als in sich sehr einheitliche Systeme mit primär versionbedingten Unterschieden darstellen, bietet Linux ein sehr breites Spektrum. Der Grund dafür ist relativ einfach: Im Gegensatz zu den anderen genannten UNIX-Systemen, die jeweils aus einer Hand angeboten werden, gibt es für Linux keine zentrale Instanz. Linux wird in Form von sogenannten Distributionen (Systempaketen bestehend aus dem Linuxkernel, [meist] einem Paketmanagementsystem und einem mehr oder weniger umfangreichen Angebot an Softwarepaketen) eingesetzt. Dabei werden die verschiedenen Distributionen von ihren Entwicklern nach sehr unterschiedlichen Gesichtspunkten strukturiert und gestaltet.

[10] Da MacOS X auf UNIX aufsetzt kann man es allerdings durchaus den UNIX-Plattformen zurechnen.

Trotz der Unterschiede existieren mit dem FHS (Filesystem Hierarchie Standard) und der LSB (Linux Standard Base) Bestrebungen zur Standardisierung wesentlicher Aspekte von Linuxdistributionen, die auch von den am weitesten verbreiteten Linuxdistributionen unterstützt werden. Auch wenn diese Standardisierungen zur teilweisen Vereinheitlichung der verschiedenen Distributionen führen, so sind sie doch von großer Bedeutung um unter anderem die Portierbarkeit von Software zwischen den verschiedenen Distributionen zu vereinfachen. Weiterhin trägt eine solche Standardisierung auch zur Vereinfachung der Administration unterschiedlicher Systeme bei.

5.1.2 Windows

Wesentlich schwieriger gestaltet sich die Situation unter Windows. Trotz nur geringer Unterschiede in der Namensgestaltung unterscheiden sich die verschiedenen Windowsversionen teilweise wesentlich stärker voneinander als UNIX-Implementationen verschiedener Hersteller. Sie lassen sich grob in drei Kategorien einteilen:

- Windows vor Windows95, dies umfaßt Windows 3.11 und dessen Vorläuferversionen. Diese Gruppe von Windowsversionen sollte, schon aufgrund von Inkompatibilitäten mit einigermaßen aktueller Hard- und Software, in den letzten Jahren zunehmend aus dem Einsatz verschwunden sein und kann daher vernachlässigt werden, zudem gelten die im folgenden genannten Nachteile von Windows9X in vollem Maße auch für diese Versionen.
- Windows9X, dies umfaßt Windows95, Windows98, WindowsME und deren verschiedene Versionen. Gemeinsame Merkmale sind unter anderem:
  - Aus Windows 3.11 hervorgegangen, setzen sie so wie dieses auf MS-DOS[11] auf.
  - Es sind single-user Systeme, ein Multiuser-Betrieb ist weder vorgesehen noch - aufgrund technischer Unzulänglichkeiten - möglich.
  - Aufgrund verschiedener Mängel - unter anderem hat der Konsolennutzer jederzeit vollen Zugriff auf das System - sind weder eine sinnvolle Administration noch ein sinnvoller Einsatz in Multiuserumgebungen möglich.
- Windows NT und Nachfolger, dies umfaßt derzeit Windows NT (überwiegend in der Version 4.0, die Versionen 3.51 und älter dürften kaum noch im Einsatz anzutreffen sein), Windows 2000 sowie Windows XP. Für diese Betrachtungen relevante gemeinsame Eigenschaften sind:
  - Neuentwickelte, eigenständige Betriebssystemgeneration.
  - Die Systeme sind multiuserfähig, arbeiten mit preemptivem Multitasking und bieten konzeptuell eine sichere Trennung zwischen System und Benutzer.
  - Sie sind für den Einsatz in Multiuserumgebungen geeignet und bei Einsatz entsprechender zusätzlicher Werkzeuge auch sinnvoll administrierbar.

Die bevorzugte Windowsversion für den Einsatz in Multiuserumgebungen ist derzeit Windows 2000 (Professional), da es gegenüber der Vorgängerversion Windows NT 4.0 zahlreiche Verbesserungen enthält, jedoch noch nicht die Nachteile seines Nachfolgers, Windows XP Professional mitbringt. Nach den Erfahrungen des Autors hat sich Windows 2000 im Einsatz als Windowsplattform im Firmennetz unter verschiedenen Gesichtspunkten bewährt:

[11] Microsoft Disk Operating System, ein singleuser, singletasking Betriebssystem für IBM kompatible PCs

- Stabilität: es hat sich als hinreichend stabil und zuverlässig erwiesen.

- Performance: trotz des, gegenüber Windows NT, deutlich gestiegenen Ressourcenbedarfs für das Basissystem ohne Anwendungen und Last liefert Windows 2000 auf einigermaßen aktueller Hardware eine durchaus akzeptable (ältere, von Windows NT 4.0 auf Windows 2000 aktualisierte Systeme) bis sehr gute (aktuelle Systeme) Performance bei einem Anwendungsspektrum von typischen Büroanwendungen bis hin zur Softwareentwicklung.

- Administration: Windows 2000 enthält gegenüber Windows NT 4.0 zahlreiche Verbesserungen, die die Administration des Systems deutlich vereinfachen.

Zu den typischen Problemen der Systemadministration unter Windows gehören u.a.:

- Es ist standardmäßig keine Remoteadministration vorgesehen und im Rahmen des Systemumfangs ist es nicht möglich, eine volle Remoteadministration - wie unter UNIX üblich - nach der Installation nachträglich zu aktivieren. Der mitgelieferte telnet-Server kann - insbesondere wenn man ihn mit dem gleichnamigen Dienst unter UNIX oder gar der, anstelle von telnet mittlerweile üblichen, ssh vergleicht – bestenfalls als absolut ungeeignet bezeichnet werden.

- Daraus resultierend ein üblicherweise hoher Aufwand für die Installation von Software, da fast alle Software über ein graphisches Installationsprogramm installiert wird, dessen Einsatz ein Login des Administrators an der physikalischen Systemkonsole erfordert.

- Windowsarbeitsplätze verursachen im Supportfall einen relativ hohen Aufwand, da für praktisch alle nichttrivialen Probleme die physikalische Anwesenheit des Administrators (oder zumindest von Unterstützungspersonal mit Administrator-Rechten) vor Ort an der betroffenen Maschine, sowie die Unterbrechung der Arbeit des betroffenen Nutzers notwendig ist.

- Die im Rahmen des normalen Systemumfangs - verglichen mit UNIX-Systemen - sehr geringen Diagnosemöglichkeiten von Windowssystemen erschweren die Problemanalyse und Fehlersuche erheblich, eine Reduzierung der Fehlerbeseitigung auf die oft zitierte Anleitung „Retry, Reboot, Reinstall“ ist zwar nicht akzeptabel, jedoch dadurch in manchen Situationen kaum vermeidbar.

5.2 Werkzeuge zur automatisierten Konfiguration

5.2.1 PIKT

PIKT^{TM}[Ost02], das „Problem Informant/Killer Tool“ ist ein Automatisierungswerkzeug für die Systemadminstration, das sich in einem kurzen Überblick so beschreiben läßt:

- eingebettete Skriptsprache und dazugehöriger Interpreter,

- leistungsfähiger Preprozessor für Skripte und Konfigurationsdateien, der sowohl mit der eigenen, als auch beliebigen anderen Sprachen zusammenarbeiten kann,

- plattformübergreifender, zentral laufender Jobdienst (mit dem UNIX Systemdienst cron vergleichbar),

- ein anpassendes Installationswerkzeug,
- Kommandozeilenerweiterung und
- Skript- und Konfigurationsdateiverwaltung.

Die Hauptaufgaben von PIKT^{TM} bestehen in der Überwachung von Systemen, dem Melden von Problemen und dem – soweit möglich – Beseitigen dieser Probleme. Aufgrund seiner Flexibilität kann es jedoch noch wesentlich umfangreichere Aufgaben erfüllen. Es folgt in seiner Struktur dem üblichen Client-Server Modell mit einem zentralen Server und auf den einzelnen Rechnern installierten Clients. Die Konfigurations- und Skriptsprache von PIKT^{TM} ist als einfache, prozedurale Sprache ausgelegt, welche trotz ihrer besonderen Elemente eine gewisse Ähnlichkeit zu anderen, Sysadmins üblicherweise vertrauten, Sprachen wie C und Perl aufweist. Sie ist unter anderem makrofähig, unterstützt Include-Dateien als Strukturierungsmittel, ist speziell auf die Bedürfnisse der Systemadministration ausgelegt und sehr leistungsfähig. Sie bietet u.a. die Möglichkeiten, komplexe Alarme (zeitgesteuerte Aktionen) aufzusetzen, Konfigurationsdateien in Abhängigkeit von der Systemumgebung und weiteren Parametern zu generieren und das System um weitere Funktionalitäten zu erweitern.

Das System kann nicht nur die Konfiguration der verwalteten Systeme erzeugen und pflegen, zentral gesteuerte regelmäßige Wartungsarbeiten (Logfiles rotieren, Dienste zeitabhängig rekonfigurieren, ...) sowie Probleme erkennen, sondern auch darüber berichten und sie – soweit sinnvoll und möglich – automatisch bearbeiten.

PIKT^{TM} ist unter den Bedingungen der GPL lizensiert und für eine breite Palette an UNIX-Systemen (darunter Linux, Solaris, HP-UX) verfügbar.

5.2.2 Cfengine

Cfengine [Bur02] ist System zur Konfiguration und Administration von Netzwerken und Hosts. Der Kern des Systems ist die Programmiersprache von Cfengine, mit der ein klassenbasierter Entscheidungsbaum aufgebaut wird, über den dann schließlich Konfigurationen berechnet und Aktionen ausgelöst werden. Dabei agiert die Programmiersprache auf sehr hohem Niveau und es wird über das Klassensystem ein gewünschter Soll-Zustand der Konfiguration bestimmt. Cfengine berechnet schließlich aus den Informationen des Klassensystems diesen Sollzustand der Konfiguration, vergleicht ihn mit dem Ist-Zustand und aktualisiert gegebenenfalls die Konfiguration. Über das Klassensystem können Maschinen nach verschiedenen Parametern konfiguriert werden, z.B.

- nach Plattform (SPARC, i386, ...),
- nach Betriebssystem (Solaris 7 oder 8, FreeBSD 4.2/4.4, ...),
- nach Aufgabenklasse (Server, Workstation, Router, ...),
- nach Aufgabe (CAD-Workstation, AFS-Fileserver, Webserver, ...),
- nach Standort (Datacenter, Poolraum, ...),
- nach Datum (z.B. Sonderkonfiguration für eine Ausstellungswoche und Uhrzeit (z.B. für Tages-/Nachtkonfiguration).

Den Möglichkeiten sind dabei erstmal keine Grenzen gesetzt – natürlich muß vorher eine geeignete Strukturierung ausgearbeitet werden, die den Anforderungen der jeweiligen Infrastruktur entspricht. Diese kann dann mit in entsprechende Cfengine-Klassenstrukturen überführt werden. Dabei unterstützt Cfengine den Sysadmins durch die automatische Definition bestimmter Klassen, die von der Systemumgebung (der laufenden Plattform, Datum/Uhrzeit, Hostnamen) abhängig sind. Weitere Klassen ergeben sich dann u.a. daraus, welchen Gruppen der Host zugeordnet wurde.

Zum Ändern der Konfiguration kann Cfengine u.a.:

- ACSII-Files editieren,
- Files kopieren,
- symbolische Links prüfen und einrichten,
- Zugriffsrechte und Eigentümerangaben von Files prüfen und setzen,
- Files löschen,
- Shell-Scripte und Kommandos ausführen,
- Signale an Prozesse schicken,
- Netz-Schnittstellen prüfen und konfigurieren,
- NFS-Filesysteme automatisch montieren,

In der aktuellen Version 2 wurde Cfegine unter anderem um RSA Authentisierung, IPv6-Unterstützung sowie bessere Erkennung von Anomalien erweitert.

Zu den besonderen Möglichkeiten von Cfengine gehört, daß es nicht nur zur automatischen Konfiguration und Administration großer Gruppen von Hosts eingesetzt werden kann, sondern das es sich auch als System zur „Selbstheilung“ von Konfigurationen eignet: im Falle einer (durch Tests, administrative Fehler, Systemschäden oder aus anderen Gründen) beschädigten Systemkonfiguration ist Cfengine in der Lage – vorausgesetzt, das System wird mit Cfengine verwaltet – die korrekte Systemkonfiguration wieder herzustellen.

Aufgrund des Klassenkonzeptes für die Konfiguration eignet sich Cfengine sowohl für die Administration großer, relativ homogener Systemgruppen (z.B. Cluster) als auch hochgradig differenzierter Systemumgebungen.

Cfengine ist ein komplexes Werkzeug mit nicht geringem Einarbeitungsaufwand. Die große Mächtigkeit und Flexibilität von Cfengine belohnt diese Investition jedoch (Cfengine wird mit großem Erfolg vom Universitätsrechenzentrum der TU Chemnitz zur Administrations eines umfangreichen und auch sehr unterschiedlichen Bestands an Systemen eingesetzt).

Cfengine ist unter den Bedingungen der GPL lizensiert und für eine breite Palette an UNIX-Systemen, MacOS X sowie Windows NT/2000 verfügbar.

6 Überwachung von Systemen

Ein wichtiger Bereich der Systemadministration ist die Überwachung von Systemen. Geeignete Überwachungssysteme ermöglichen eine rechtzeitige Erkennung von Problemen und auch die Betrachtung der Entwicklung verschiedener Betriebsparameter und liefern damit wichtige Informationen die in die Planung eines Netzwerksystems einfliessen. Ohne Überwachungssysteme muß sich der Sysadmin darauf verlassen, daß er auftretende Probleme entweder selbst rechtzeitig erkennt oder darauf warten, das die Nutzer der von ihm administrierten Systeme ihn auf Probleme (bzw. deren Symptome) hinweisen – was kein akzeptabler Zustand ist. Aus diesem Hintergrund entschied sich der Autor, die Betrachtung von Überwachungssystemen in den Umfang dieser Arbeit aufzunehmen.

6.1 Kategorien von Überwachungssystemen

Zur Überwachung von Netzwerken und Hosts gibt es zahlreiche Softwaresysteme, die verschiedene Überwachungsansätze verfolgen. Grundsätzlich kann man sie jedoch in zwei große Kategorien einteilen:

- Systeme zur Überwachung und Alarmierung,
- Systeme zur Überwachung und Graphengenerierung.

Erstere überwachen verschiedene Parameter und aktivieren bei Eintreten bestimmter Triggersituationen verschiedene Alarmierungszustände, deren Ausprägung vom Wechsel einer Statusanzeige bis zur direkten Alarmierung entsprechenden administrativen Personals mit geeigneten Mitteln, z.B. EMail, Pager und ähnlichem reicht. Üblich ist auch eine Aufzeichnung der eingetretenen Ereignisse in Form eines Logs. Im Gegensatz dazu wird bei den letztgenannten Systemen eine nahezu kontinuierliche Aufzeichnung der überwachten Parameter durchgeführt, eine Alarmierung bei Erreichen von Schwellwerten ist jedoch meist nicht vorgesehen. Die Aufzeichnung erfolgt nahezu kontinuierlich - oftmals werden die entsprechenden Parameter im Abstand von einer oder fünf Minuten aufgezeichnet - da eine wirklich exakt kontinuierliche Aufzeichnung seltenst erforderlich ist.

Die Systeme der ersten Kategorie eignen sich somit vorrangig zur Überwachung des normalen Betriebs und zur zeitnahen Erkennung von und Reaktion auf im laufenden Betrieb auftretende Probleme, während die Systeme der zweiten Kategorie vor allem zur Überwachung des Betriebs innerhalb der normalen Parameter, zur Erkennung von Trends sowie sich allmählich entwickelnden Problemen eignen.

Aufgrund dieser Eigenschaften ergänzen sich beide Kategorien sehr gut und es empfiehlt sich zur vollen Überwachung der Einsatz von Systemen aus beiden Kategorien.

Natürlich ist die Bandbreite der vorhandenen Lösungen in diesem Bereich sehr umfangreich und es ist weder sinnvoll noch möglich, im Rahmen dieser Arbeit einen repräsentativen Querschnitt, geschweige denn einen überwiegenden Teil der frei verfügbaren Lösungen vorstellen zu wollen. Daher muß die Auswahl auf geeignete Weise eingeschränkt werden. Als Kriterien bieten sich dabei an:

- frei verfügbar (ein in dieser Arbeit mehrfach erwähntes entscheidendes Kriterium),
- Frontend durch ein Webfrontend implementiert – damit ist größtmögliche Unabhängigkeit bezüglich der Nutzung durch Clients zur Informationsabfrage gewährleistet,

- umfangreiche Leistungspalette – Lösungen, die auf einen oder nur sehr wenige Aspekte spezialisiert sind, kann ein guter Sysadmin in akzeptabler Zeit selbst zusammenbauen, wirklich interessant sind Lösungen, die sehr flexibel für unterschiedliche Überwachungsaspekte einsetzbar sind,

- Portabilität – die Software (sowohl Server als auch eventuell auf überwachten Maschinen zu verteilende Sensorsoftware) sollte möglichst portabel sein, dabei ist UNIX-Unterstützung Vorraussetzung, Windows-Unterstützung von Vorteil.

Unter Beachtung der vorgenannten Kriterien möchte der Autor nun im Folgenden einige Systeme näher betrachten.

6.2 Überwachung und Alarmierung

6.2.1 Nagios™

Nagios™[Gal02] ist die Weiterentwicklung des vorher als NetSaint [Gal01] bekannten Netzwerk Monitoring Tools.

Es ist sehr leistungsfähig und kann mit einer großen Anzahl von Sensoren umgehen – und damit sehr viele verschiedene Werte überwachen. Die Konfigurationsmöglichkeiten sind sehr umfangreich, es können u.a. die Optionen zur Objektüberwachung (== ein Dienst) sehr fein eingestellt werden. Die Konfiguration über Templates, mit denen Standardwerte einmal definiert und dann vererbt werden können, vereinfacht den Umgang mit diesen umfangreichen Möglichkeiten.

Als Datenspeicher unterstützt Nagios™ sowohl – als ursprünglichen einzigen, mittlerweile aber nur noch standardmäßig verwendeten Datenspeicher – einfache Textdaten, als auch MySQL und PostgreSQL. Dabei werden – sofern genutzt – in den Datenbanken die gesammelten Überwachungsdaten gespeichert, die Konfiguration erfolgt nach wie vor über Textdateien.

Eine sehr angenehme Fähigkeit von Nagios™ ist, daß es mit Dienstabhängigkeiten umgehen kann. Wenn z.B. Nagios den Ausfall eines Routers feststellt, dann wird es nicht auf dem erfolglosen Versuch bestehen, die „dahinter liegenden“ Dienste immer noch zu überwachen. Die Definition von Dienstabhängigkeiten kann u.a. das Alarmierungssystem deutlich aufräumen - es wird nur noch eine Alarmierung für die Grundursache (z.B. ausgefallener Router) erstellt und nicht mehr für jedes dadurch verursachte Folgeproblem (z.B. nicht erreichbare Server).

Das Webfrontend verfügt über zahlreiche, gut gestaltete (im Sinne von informativ und übersichtlich) grafische Darstellungsmöglichkeiten für den Zustand der überwachten Objekte einschließlich ihrer Beziehungen untereinander. Es ist sogar eine dreidimensionale Darstellung unter Verwendung von VRML integriert. Mit der Fähigkeit, unter Verwendung der aufgezeichneten Überwachungsdaten – je nach überwachtem Objekt – grafische Darstellungen zu historischen Entwicklungen dieses Wertes darstellen zu können, streift Nagios™ den Bereich der graphengenerierenden Überwachungssysteme.

Der Autor hat eine früheren Installation den Vorläufer, NetSaint, erfolgreich eingesetzt und war bereits mit dessen Möglichkeiten sehr zufrieden - mit den Weiterentwicklungen zu Nagios™ hat sich die Leistungsfähigkeit und Flexibilität dieses Werkzeuges noch verbessert.

Nagios™ unterliegt der GPL [Fre01].

6.2.2 Big Brother

Big Brother [Tec02] ist ein weiteres Systemüberwachungswerkzeug mit Weboberfläche und zahlreichen Überwachungsmöglichkeiten. Es basiert auf einer Client/Server-Architektur. Dabei werden die Clients auf den zu überwachenden Maschinen installiert und liefern die zu überwachenden lokalen Systeminformationen (Dateisystemlast, CPU-Auslastung, ...) an den Server, der die Informationen verarbeitet, Netzwerkdienste überwacht und auch für die Darstellung über das Webfrontend zuständig ist. Für die lokale Datenerfassung stützt sich Big Brother auf seine Clients, die über ein eigenes, offengelegtes Protokoll mit dem Server kommunizieren - via TCP/IP und natürlich auf Port 1984. Unterstützung für eine Datenerfassung via SNMP Traps ist – über Plugins – in begrenztem Maße vorhanden. Standardmäßig unterstützt Big Brother die Überwachung der Netzwerkdienste FTP, HTTP, HTTPS, SMTP, POP3, DNS, Telnet, IMAP, NNTP und SSH, sowie mittels lokal installierter Clients die Überwachung von Dateisystemauslastung, CPU-Last, Systemnachrichten, sowie die Aktivität wichtiger Prozesse (z.B. Datenbankserverprozess). Durch Plugins kann dabei die Leistungspalette noch praktisch beliebig erweitert werden.

Als Plattform für den Big Brother Server wird bevorzugt Unix unterstützt, es existiert aber eine reduzierte Version für Windows NT/2000. Die Bigbrother Clients sind (einschließlich Drittentwicklungen durch Anwender) für Unix, Windows NT/2000, VMS, Mac OS sowie weitere Systeme verfügbar.

Über das Webinterface von Big Brother erfolgt sowohl die Darstellung des aktuellen Zustands, als auch der Zugriff auf die archivierten historischen Werte. Dabei wird grundsätzlich auf Basis von Triggerschwellen dargestellt:

- grün - alles in Ordnung, keine Probleme,
- gelb - Warnschwelle ausgelöst,
- rot - Problem oder kritische Schwelle (z.B. für Dateisystemauslastung erreicht),
- lila - kein Report, d.h. seit mehr als 30 min. gibt es keine neuen Überwachungsdaten,
- schwarz - keine Information (administrativ abgeschaltet),
- blau - Benachrichtigung deaktiviert (z.B. wegen Wartung).

Mit diesen Farben wird – als Host-Dienste-Matrix – der Zustand der überwachten Systeme und Dienste dargestellt. Die höchste Alarmierungsstufe (grün/gelb/rot) aller dargestellten Werte bestimmt dabei die Hintergrundfarbe der Webseite. Über die Matrix sind die Seiten der einzelnen überwachten Werte verlinkt, auf denen – soweit verfügbar – detaillierte Informationen zum aktuellen Zustand geliefert werden, sowie ein Zugriff auf die „Geschichte“ des Wertes geboten wird. Diese „historische Übersicht“ erfaßt allerdings nur die Zustandsübergänge des Wertes. Zusätzlich zur Darstellung auf den Webseiten verfügt Big Brother über umfangreiche Benachrichtigungsoptionen, es können bei kritischen Zustandsübergangen Benachrichtigungen an Personen oder Personengruppen per EMail, Pager oder SMS versandt werden wobei die Einstellungen dieser Benachrichtigungen auch an die aktuelle Zeit gekoppelt werden können. So kann man z.B. während der Arbeitszeit die zuständigen Admins per EMail und abends den Mitarbeiter in der Rufbereitschaft per Pager/SMS vom Ausfall eines Datenbankservers informieren.

Trotz aller Leistungsfähigkeit hat Big Brother auch einige Schwächen. Zum einen ist Big Brother überwiegend als Shellscripte implementiert und damit äußerst `fork()`-intensiv, was je nach Plattform mehr oder weniger schnell zu erheblichen Lastproblemen führen kann. Die relativ enge Vermischung von Code und Konfiguration – u.a. stehen Teile der Konfiguration in den Shellscripten – erschweren die Wartung existierender Big Brother Installationen. Auch die Datenhaltung in unzähligen kleinen Dateien im Dateisystem kann zu Problemen führen. Aufgrund der nicht übermäßig klaren Struktur des Gesamtsystems sind Modifikationen und Erweiterungen des Systems deutlich weniger einfach als in der Dokumentation versprochen. Die in [Pet00] angesprochenen Skalierungsprobleme bei Installationen mit mehr als 300 überwachten Systemen dürften allerdings für den Zielbereich dieser Arbeit – kleine bis mittlere Netze – weniger von Belang sein. Der Autor mußte jedoch feststellen, daß Big Brother in bestimmten Problemsituationen selbst eine erhebliche Last auf dem Server verursachen und damit Dienste stark behindern kann (in den konkreten Fällen führte dies zu Ausfällen bei dem auf dem gleichen Host laufenden Mailsystem, da sendmail ab Lastfaktor 12 standardmäßig keine Mails mehr entgegennimmt). Allerdings sind unter Umständen die Lizenzbedingungen von Big Brother zu beachten: die private und nichtkommerzielle Nutzung (dies schließt die Nutzung an Bildungseinrichtungen ein, solange diese es nicht kommerziell einsetzen) ist kostenlos, für die kommerzielle Nutzung – z.B. für die Überwachung von E-Commerceservern – fallen hingegen Lizenzkosten an.

6.2.3 Big Sister

Big Sister [Aeb02] ist eine Reimplementation und Erweiterung von Big Brother in Perl. Der Grund dafür waren verschiedene, zum Teil bereits in 6.2.2 angesprochene Probleme mit Big Brother sowie Erweiterungswünsche:

- zum Teil erhebliche Performanceprobleme bei Big Brother,
- der Wunsch nach mehr Sensoren zur Dienstüberwachung,
- Verbesserung des Benachrichtigungssystems (u.a. die Beachtung von Abhängigkeiten bezüglich der Erreichbarkeit überwachter Systeme),
- Verbesserung der Darstellungsmöglichkeiten des Displayservers (Webfrontend),
- mögliche Integration in existierende HP Openview/SNMP NMS Installationen.

Big Sister verwendet u.a. das Client-Server Protokoll von Big Brother sowie einen Teil von dessen Logfileformat. Es implementiert praktisch vollständig die Fähigkeiten von Big Brother sowie vieles mehr, darunter:

- Verarbeitung von SNMP Traps, Senden von SNMP-Traps als Benachrichtigung,
- grafische Darstellung (unter Verwendung von RRDTool [Oet02] von Sensordaten,
- eine wesentlich umfangreichere Palette an Sensoren, darunter Sensoren für Oracle, NT Eventlog, Tripwire, ...
- flexiblere und bessere Darstellung im Webfrontend.

Natürlich kann es direkt die Big Brother Sensoren und Agents nutzen. Da Big Sister in der Lage ist, Teile von Big Brother zu nutzen, eignet es sich insbesondere für eine Migration von einer existierenden, aber – aufgrund von Skalierungs- oder Performanceproblemen, aus Flexibilitäts- oder anderen Gründen – abzulösenden Big Brother Installation. Big Sister ist als freie Software nach der GPL [Fre01] lizensiert.

6.3 Überwachung und Graphengenerierung

6.3.1 mrtg

Die Abkürzung `mrtg` steht für „Multi Router Traffic Grapher“ und weist bereits auf das Einsatzgebiet hin, für das `mrtg` ursprünglich entwickelt wurde, die Anzeige der Netzwerklast auf Routern. Von `mrtg` werden in regelmäßigen Abständen, standardmäßig alle 5 Minuten, Meßwerte aufgenommen, aus denen dann tägliche, wöchentliche und jährliche Graphen zur Visualisierung generiert werden. Diese Graphen werden als PNG-Dateien erzeugt, die von jedem aktuellen Webbrowser angezeigt werden können. Die Meßwerte werden in Logdateien konstanter Größe für zwei Jahre archiviert, was für viele Anwendungsgebiete dieses Werkzeugs ausreichend sein dürfte. Die Datenerfassung für `mrtg` kann über zwei Mechanismen erfolgen: SNMP und externe Sensoren. Wenn die zu überwachenden Geräte eine Abfrage der relevanten Daten via SNMP gestatten, so kann der in `mrtg` integierte SNMP-Support zum Auslesen der Meßwerte genutzt werden. Falls die zu üeberwachenden Daten nicht via SNMP zugänglich sind oder SNMP – zum Beispiel aus Sicherheitsgründen – nicht genutzt werden soll, so kann man auf externe Sensoren zurückgreifen. Dies sind Programme, die von `mrtg` aufgerufen werden und die Meßdaten in einem vorgebenen Format auf die Standardausgabe schreiben, wo sie dann von `mrtg` erfaßt werden. Dabei können diese Sensoren durchaus vertrauliche Daten über unsichere Strecken übertragen: der Autor verwendet in einer Situation (Serversystem in Colocation) ssh als Transport für die Remotesensoren, womit die Vertraulichkeit der Übertragung gesichert ist.

Der Autor verwendet `mrtg` sowohl im privaten als auch im Firmeneinsatz, teilweise mit selbstgeschriebenen oder überarbeiteten Sensoren, zur graphischen Überwachung verschiedenster Parameter:

- CPU-Last (Lastfaktor),
- Prozessanzahl,
- Filesystemfüllstand,
- Festplatten- und Systemtemperatur,
- Anzahl eingeloggter Nutzer,
- Netzwerklast,
- DNS-Serverlast (Abfragen),
- Mailserverlast (empfangene/versendete Mail),

wobei weitere, auf diese Weise graphisch zu überwachende, Parameter ohne weiteres denkbar sind. Dies beweist unter anderem die „MRTG Companion Sites“ Seite mit ihren Links auf verschiedenste weltweite `mrtg`-Installationen.

Trotz verschiedener Optimierungen hat mrtg aber ein Skalierungsproblem. In einer dem Autor bekannten Installation benötigt bei ca. 130 überwachten Werten ein normaler Durchlauf von mrtg (Sammeln der Überwachungsdaten, Aktualisieren der Datenspeicher, Neuerzeugen der grafischen Darstellung) bereits ca. 2-3 Minuten, und das bei einem Durchlauf alle 5 Minuten. Der Autor sieht dies vor allem in der unglücklichen Designentscheidung begründet, das Backend (Sammeln und Speichern der Überwachungsdaten) und das Frontend (Darstellen der Graphen) in einer Komponente zu integrieren. Andere Tools lösen dieses Problem durch Trennung von Front- und Backend.

6.3.2 RRDtool

RRDtool [Oet02] ist eine Reimplementation des Datenspeicher- und Grafikteils von mrtg. RRD steht für „Round Robin Database", ein Werkzeug zur Speicherung von Meßreihen (Netzwerklast, Temperaturen, Serverstatistiken). Dabei werden die Daten in einem Format konstanter Größe gespeichert. RRDtool selbst verfügt weder über das von mrtg implementierte Frontend noch über die Datensammelfähigkeiten von mrtg, sondern konzentriert sich auf den Bereich der Speicherung und Darstellung der Daten aus Meßreihen, ist in diesem Bereich aber äußerst leistungsfähig und konfigurierbar. Zusammen mit dem u.a. von der RRDtool Homepage [Oet02] verlinkten Frontend können damit sehr leistungsfähige und flexible Systeme zur graphenbasierten System- und Trendüberwachung aufgebaut werden.

6.3.3 Cricket

Cricket [All02] wurde bei WebTV[12] entwickelt, als sich mit der bis dahin verwendeten grafischen Überwachungslösung auf Grundlage von mrtg zunehmende Performanceprobleme abzeichneten. Die Architektur von Cricket basiert auf der üblichen Gliederung in Datensammlerkomponente und Anzeigekomponente. Dabei läuft die Datensammelkomponente – gesteuert vom Systemdienst cron – alle 5 Minuten, was den üblichen Gepflogenheiten dieser Dienste entspricht. Die Anzeigekomponente ist als CGI[13] ausgelegt und kann zum interaktiven Blättern durch die gesammelten Daten verwendet werden. Als Datenspeicher wird das bereits in 6.3.2 beschriebene RRDtool verwendet, das auch zur eigentlichen Graphendarstellung zum Einsatz kommt.

Bezüglich der Datenquellen ist Cricket sehr flexibel, es unterstützt dabei als Quellen:

- SNMP,
- Shellskripte,
- Dateien,
- URLs,
- Perlskripte.

Sowohl die Datensammel- als auch die Anzeigekomponente von Cricket benutzen den sogenannten Konfigurationsbaum als hierarchisches Konfigurationssystem. Dieses System benutzt

[12] WebTV Networks, Inc.

[13] Common Gateway Interface - Programme, die unter einem Webserver ausgeführt werden und dynamisch Webseiten erzeugen

unter anderem Vererbung, um eine Wiederholung von Konfigurationsoptionen zu vermeiden, außerdem kann die Datensammelkomponente dadurch einfach parallelisiert werden.

Zusätzlich zur Graphengenerierung verfügt Cricket auch über eine Alarmierungskomponente, die bei Eintreten von entsprechenden Bedingungen Alarme auslösen kann. Diese Bedingungen können einfache, absolute Wertüberschreitungen sein, aber auch Vergleiche (z.B Netzwerklast von heute ist 20% größer als gestern).

6.3.4 cacti

Auf RRDtool aufsetzende visualisierende Monitoringsysteme gibt es verschiedene, das leistungsfähigste jedoch ist cacti [Ber02b]. Es ist ein in PHP geschriebenes System zur Datensammlung und -darstellung, welches komplett über seine Weboberfläche konfiguriert wird und seine Konfiguration in einer MySQL-Datenbank [AB02] speichert. Für die Datensammlung kann cacti sowohl SNMP als auch externe Sensoren (die im Falle der mitgelieferten Sensoren in Perl implementiert sind) verwenden. Aufgrund der großen Funktionsvielfalt von cacti verfügt es über eine eigene Nutzerverwaltung, mit der Nutzern spezielle, abgegrenzte Rechte am Gesamtsystem gegeben werden können. Aufgrund der Leistungsfähigkeit von cacti empfiehlt der Autor – falls noch kein etabliertes graphisches Monitoringsystem, z.B. auf Grundlage von mrtg im Einsatz ist – den Einsatz von cacti für diesen Zweck.

7 Wissensmanagement

7.1 Ansätze zum Wissensmanagement

7.1.1 Grundsätzliche Überlegungen

Wenn man davon ausgeht, daß die grundsätzlichen Konzepte und Zusammenhänge der Administration eines Netzes als fachspezifisches Grundwissen zu betrachten sind, dann ergeben sich für das Management des dabei anfallenden Wissens zwei unterschiedliche Ansätze:

- Wissensniederlegung als umfassende Dokumentation und
- Wissensniederlegung als Speicherung spezifischer Details.

Keiner der beiden Ansätze ist wirklich in der Lage, alle Anforderungen abzudecken und so wird man normalerweise beide im Einsatz finden, wenn auch in sehr variierender Ausprägung. Beide Ansätze eignen sich für die Lösung unterschiedlicher Probleme des Wissensmanagements. Im folgenden sollen unterschiedliche Ansätze und dazugehörige Werkzeugkonzepte betrachtet werden.

7.1.2 Dokumentation

Der klassische Ansatz zur Verwaltung von fachspezifischem Wissen im Bereich der IT ist das Schreiben von Dokumentationen. Dabei wird üblicherweise ein zu einem bestimmten Zeitpunkt aktueller Zustand eines Systems – eine Art „Schnappschuß" – dokumentiert. Neben den fachlichen Kenntnissen des Autors über das zu dokumentierende System sowie seiner Beteiligung daran hängen Umfang und Qualität der Dokumentation auch sehr stark von der zu ihrer Erstellung zur Verfügung stehenden Zeit ab. Eine, in Form eines festen Textes geschriebene Dokumention hat erfahrungsgemäß das Problem, nur selten aktualisiert zu werden und dadurch gerade bei sehr dynamischem Dokumentationsgegenstand schnell zu veralten. Ein weiteres, oft auftretendes Problem besteht darin, daß nur selten der für eine wirklich gute und umfassende Dokumentation eines Systems notwendige Aufwand investiert wird.

7.1.3 Knowledgebase

Ein bekanntes und schon zu den Standardwerkzeugen zählendes Mittel zum Management von umfangreichem, jedoch gleichzeitig inhaltlich sehr weit gestreutem, oftmals nur auszugweisem Wissen ist das Mittel der Knowlegdebase. Dabei handelt es sich üblicherweise um eine Sammlung zahlreicher, meist eher kurzer Artikel zu einem ganz spezifischen Problem, die durch ein Recherche-, Schlagwort- und Indexsystem miteinander verknüpft und meist in Form einer Datenbank gespeichert sind. Einer der wohl bekanntesten Vertreter dafür ist die Microsoft Knowledgebase [Mic02]. Mit ihrem umfangreichen Bestand an schlaglichtartig ein ganz bestimmmtes Thema behandelnden Artikeln, den Verknüpfungen der Artikel untereinander, dem System der Verschlagwortung, dem integrierten Recherchesystem und der Präsentation als Webanwendung stellt die Microsoft Knowledge Base ein typisches Beispiel für dieses Konzept des Wissensmanagements dar.

Ein Nachteil dieses Ansatzes ist die üblicherweise flache Organisation einer Knowledgebase. Mit der Verschlagwortung als einzigem Strukturmittel kann die Suche nach spezifischen Informationen oftmals zu einer „Odysee im Schlagwortwald" werden, wenn man nicht die richtigen Schlagworte trifft. Eine Einbindung der einzelnen Artikel in eine zusätzliche, tiefe hierarchische Struktur, könnte diese Situation zwar entschärfen, erfordert jedoch – gerade bei umfangreichen Wissensbeständen – einen nicht unerheblichen redaktionellen Aufwand durch fachkundiges Personal.

7.1.4 Webbasierte, teamorientierte Dokumentationssysteme

Ein sich zunehmender Beliebheit erfreuender Ansatz zum Wissensmanagement ist das Konzept webbasierter, teamorientierter Dokumentationssysteme, speziell in der als Wiki Wiki Web [Cun02] bekanntgewordenen Ausprägung. Die Grundidee des Wiki Wiki Web – üblicherweise kurz als „Wiki“ bezeichnet – ist relativ einfach: eine Website, die von ihren Nutzern editiert wird. Jeder Nutzer kann vorhandene Seiten editieren (und löschen) sowie neue Seiten hinzufügen. Das einzige Interface, welches der Nutzer eines Wikis benötigt, ist ein Webbrowser. Die Verwendung eines Wikis ist komplett webbasiert, von der Recherche über das Editieren bis hin zur (soweit von der jeweiligen Wiki-Implementation unterstützt) Administration des Wiki. Aufgrund des freien Zugriffs auch für Änderungen, vermeiden Wikis das „ein Webmaster Problem“[14] und fördern ein dynamisches Arbeiten mit dem System, was wiederum eine sehr hohe Aktualität ermöglicht.

Damit eignen sich Wiki Wiki Webs hervorragend als Teamwerkzeug, da jedes Teammitglied mit minimalem Aufwand zu Inhalt, Struktur und Gestaltung des Wikis beitragen kann.

Wikis werden dementsprechend auch mit großem Erfolg als teamorientierte Werkzeuge eingesetzt und dienen unter anderem als:

- Kommunikationsplattform innerhalb von Teams,
- Wissensbasis („Knowledgebase“),
- fortlaufend aktualisierte Dokumentation,
- dynamische Intranets.

Einige kurze, ausgewählte Beispiele für die erfolgreiche Einführung von Wikis in Firmen sind [Tho02a], [Spa02], [Tho00].

7.1.5 Troubleticket-Systeme

Sysadmins bekommen tagtäglich Anfragen und Anforderungen von verschiedenen Seiten, die bearbeitet und beantwortet werden wollen. Wie in [Tom01] ausgeführt, ist der Einsatz eines Troubleticket-Systems zur Verwaltung dieser Anfragen unbedingt zu empfehlen. Anfragen und Anforderungen werden als Troubletickets gespeichert und ihr aktueller Status sowie ihre Bearbeitung werden dadurch abfrag- und nachvollziehbar. Unter der Vorraussetzung, daß das Troubleticket-System die geschlossenen – also erfolgreich bearbeiteten – Troubletickets mit allen angefallenden Informationen (EMail-Korrespondenz, Notizen, ...) archiviert, sammelt sich im Laufe der Benutzung eines solchen Troubleticket-Systems im System problembezogenes Wissen von erheblichem Umfang an. Dies kann damit auch als Wissensbasis genutzt werden indem z.B. beim erneuten Auftreten bereits erfolgreich bearbeiteter Probleme, der zuständige Bearbeiter auf die bereits vorhandenen Information im Troubleticket-System zurückgreift.

[14] nur eine bzw. wenige berechtigte Personen können zentrale Webseiten aktualisieren und werden somit zum Flaschenhals

7.2 Verschiedene Systeme im Test

Es wurden verschiedene Systeme betrachtet. Aufgrund der großen Varianz der Software in Bezug auf Entwicklungsstand, Leistungsumfang und Zielrichtung, sowie der großen Anzahl der Systeme soll jedoch auf eine detaillierte Erwähnung aller betrachteten Systeme verzichtet werden. In den folgenden Abschnitten sollen lediglich die vom Autor als empfehlenswert betrachteten Softwaresysteme beschrieben werden.

7.2.1 Request Tracker

Neben Request Tracker wurden noch verschiedene andere Troubleticket-System betrachtet. Diese lassen sich in 3 Gruppen einteilen:

- kommerzielle Produkte,
- spezialisiert auf bestimmte Anwendungsbereiche, z.B. als Troubleticket-System für ISPs, für Webhoster, ...,
- nicht hinreichend ausgereift (dies traf für den überwiegenden Teil der betrachteten Systeme zu).

Aus dem Vergleich der verschiedenen Softwarepakete geht Request Tracker als die beste Universallösung hervor, auf eine detaillierte Behandlung der anderen Systeme soll deshalb an dieser Stelle verzichtet werden.

Request Tracker [Bes02] ist ein Troubleticket-System, welches für kleinere bis mittlere Unternehmen konzipiert wurde. Es kann u.a. für Kundensupport, Bugtracking, Troubleticket-Verwaltung verwendet werden. Derzeit ist es weltweit in über 1000 Installationen im Einsatz.

Eine kurze Übersicht über die Merkmale von Request Tracker:

- Nutzung via EMail, Web und Kommandozeile,
- Webinterface für Endnutzer zur schnellen Statusabfrage ihrer Troubletickets,
- SQL-Datenbank als Datenspeicher,
- feingranulare Zugriffssteuerung,
- flexibles Schlagwortsystem,
- umfangreiches System zur Verlinkung innerhalb des Systems sowie zu externen URLs,
- einfache Erweiterbarkeit,
- lizensiert unter der GNU GPL[Fre01].

Request Tracker benötigt als Installationsvoraussetzungen:

- Perl,
- eine Datenbank (derzeit werden MySQL, PostgreSQL und Oracle unterstützt, der Einsatz von Oracle ist aber laut Dokumentation problematisch),
- ein Webserver mit Perl-Unterstützung, empfohlen wird Apache 1.3 mit mod_perl.

Der Arbeitsablauf für den Einsatz von Request Tracker entspricht den üblichen Gepflogenheiten für Troubleticket-Systeme:

- Per EMail an help@example.com wird ein Problem berichtet.
- Request Tracker erzeugt daraus ein neues Troubleticket.
- Request Tracker leitet die EMail an die Gruppe der eingetragenen Mitarbeiter weiter.
- Einer der Mitarbeiter übernimmt die Bearbeitung des Troubletickets.
- Nach erfolgreicher Bearbeitung des Troubletickets wird es geschlossen.

Dabei speichert Request Tracker nicht nur den gesamten, über Request Tracker abgewickelten EMail-Verkehr, sondern auch zusätzliche Notizen, Kommentare und Attachments dazu. Weiterhin können Zusammenhänge zwischen Troubletickets eingetragen und dargestellt werden:

- Abhängigkeiten (Darstellung beider Richtungen),
- Verweise (ebenfalls Darstellung beider Richtungen),
- Eltern-/Kind-Beziehungen zwischen Troubletickets,

Request Tracker erweist sich als sehr leistungsfähiges und umfangreiches Troubleticket-System, welches bereits in [Tom01] empfohlen wird. Diese Empfehlung kann der Autor an dieser Stelle nur wiederholen. Im Bereich der TU Chemnitz ist Request Tracker als Troubleticket-System des Chemnitzer Studentennetzes CSN [CSN02] erfolgreich im Einsatz.

7.2.2 TWiki

Aufgrund der großen Popularität von Wikis entwickelte sich auch eine breite Palette an sogenannten Wikiclones (anderen Implementationen des Wiki-Konzeptes). Die verfügbare Palette an Wikiclones ist relativ umfangreich, allein [(OS02] listet unter dem Suchbegriff „Wiki“ 27 Treffer (bei denen es sich auch überwiegend um Wikiclones handelt) und auf den „WikiWikiClone“- Seiten von [Cun02] finden sich noch *wesentlich* mehr verlinkt.

Als ein Mitglied dieser Softwarefamilie soll an dieser Stelle TWiki [Tho02b] beschrieben werden, da es die Anforderungen an ein Werkzeug zur Wissensverwaltung im Bereich der Systemadministration sehr gut erfüllt.

Übersicht über die Merkmale von TWiki:

- Versionsmanagement – TWiki versioniert alle Wikis, was ein Nachverfolgen von Änderungen (wer/wann/was) ermöglicht,
- eigene Nutzerverwaltung – dieses optionale Feature beseitigt bei Bedarf das Problem der anonymen Änderungen des klassischen WikiWikiWeb,
- Rechteverwaltung – bei Bedarf können damit die Rechte dem Einsatzzweck entsprechend gesteuert werden um z.B. nur bestimmten Nutzern Schreibzugriff auf bestimmte Bereiche zu gewähren,
- File Attachments – in Wikiseiten können downloadbare Dateien eingebunden werden,

- Benachrichtigung per EMail – es ist möglich, sich von Änderungen an Wikis automatisch per EMail benachrichtigen zu lassen,
- weiterhin verfügt TWiki natürlich über die für Wikis üblichen Eigenschaften.

TWiki ist als ein Satz aus Templates, Dokumentation und in Perl geschriebenen CGI-Programmen implementiert. Als Speichersystem für die in den durch TWiki bereitgestellten Wikis wird das „Revision Control System“ RCS verwendet – auf diese Weise implementiert TWiki die Versionierung der Wikis. Die Nutzerverwaltung wird über den `.htaccess` Mechanismus gelöst, die Rechteverwaltung ist im TWiki selbst implementiert.

Der Autor verwendet TWiki im Rahmen seiner derzeitigen Tätigkeit als Sysadmin zur zentralen Bereitstellung administrativen Wissens. Dabei hat sich die Unterstützung von TWiki für Attachments als sehr hilfreich erwiesen, da auf diese Weise Patches, Konfigurationsdateien und ähnliches leicht den dokumentierenden Wikiseiten beizulegen sind und nicht an anderer Stelle extern gespeichert werden müssen. TWiki hat sich im Einsatz als Wissenspeicher für die Systemadministration gut bewährt.

7.2.3 unixops

Systeme nach dem Muster einer Knowledgebase – also nach Schlagworten und Eintragsnummern indizierte Wissensspeicher – lassen sich mit relativ geringem Aufwand lokal selbst implementieren. Dies soll jedoch nicht Gegenstand dieser Arbeit sein, daher wird unixops [uni02] hier als ein einfaches Beispiel eines Knowledgebase-Systems betrachtet. Die Struktur von unixops ist relativ einfach, es werden Artikel (Texte für die Knowledgebase) und Inventareinträge verwaltet. Dabei dienen eine Suche über den Volltext der Artikel und Inventareinträge sowie ein verlinktes Inhaltsverzeichnis der Einträge der beiden Kategorien als Navigationsmittel. Dieser Ansatz mag für sehr einfache Systeme durchaus verwendbar sein, aber bereits bei einer größeren Anzahl von Einträgen geht bei diesem System die Übersicht verloren. Das Problem der Navigation ist allerdings kein spezifisches Problem dieser Lösung, sondern ein grundlegendes Problem des Konzepts einer flachen Knowledgebase – darauf wurde bereits in 7.1.3 hingewiesen. Der einzig erkennbare Vorteil dieses Konzepts ist der sehr geringe Implementations- und Pflegeaufwand. Dafür müssen allerdings erhebliche Abstriche bei der Benutzbarkeit des Systems in Kauf genommen werden.

7.3 Schlußfolgerungen zum Wissensmanagement

Ausgehend sowohl von den Betrachtungen und Analysen im Rahmen dieser Arbeit als auch von seinen praktischen Erfahrungen als Sysadmin kann der Autor als Werkzeug zum Wissensmanagement das Konzept der Wikis, speziell in der Implementation TWiki empfehlen. Im Vergleich zu anderen Ansätzen bietet das Wiki-Konzept die größtmögliche Flexibilität und einfache Anwendbarkeit.

8 SCASS - System Configuration and Administration Support System

8.1 Ansatz

SCASS wurde als modulares Unterstützungswerkzeug für die Systemadministration entworfen. Es besteht aus mehreren, voneinander unabhängigen Modulen und kann relativ einfach erweitert werden. Wie bereits im vorangegangenen Text erwähnt, soll damit die Administration zentraler Dienste sowie die zentrale Verwaltung unterstützt werden. Entgegen der Aufgabenstellung soll in dieser ersten Version von SCASS vorerst auf die Nutzeradministration verzichtet werden, da der notwendige Aufwand für die Implementation eines solchen Systems in der wünschenswerten Implementationstiefe zusätzlich zur weiteren Funktionalität von SCASS nicht im relativ engen zeitlichen Rahmen einer Diplomarbeit zu bewältigen ist.

8.2 Design von SCASS

Im folgenden soll das Designkonzept von SCASS erläutert werden. Dabei wird auf die beiden Grundgedanken des Entwurfs,

- Aufbau aus voneinander unabhängigen Modulen und
- Strukturierung in:
  - einfaches Webfrontend,
  - intelligenten Datenspeicher,
  - Backend

sowie die auf Überlegungen hinter den Entwurfsentscheidungen eingegangen.

8.2.1 Modulkonzept

SCASS ist als Sammlung von Modulen entworfen, die eine gemeinsame Infrastruktur nutzen, sich mit einem gemeinsamen Interface in einem System präsentieren und doch voneinander unabhängig sind. Dabei können die einzelnen Module bei fehlendem Bedarf einfach deaktiviert werden – es können aber auch mit minimalem Aufwand zusätzliche Module in das System eingebunden werden.

Die fehlende Verknüpfung zwischen den einzelnen Modulen verhindert zwar die Implementation bestimmter Features (z.B. Verwendung von Informationen aus dem DNS-Modul zur Validierung von Eingaben im DHCP-Modul), sichert aber die Unabhängigkeit der Module voneinander (so kann man das DHCP-Modul nutzen, ohne das DNS-Modul ebenfalls verwenden zu müssen).

Die Module selbst sind in sich nach einheitlichen Mustern, die u.a. die Strukturierung der Quellen und den Aufbau der Funktionalität umfassen, entwickelt. Dies erleichtert die Erweiterung und Pflege der Module.

Insgesamt sollen im Rahmen dieser Arbeit für SCASS 3 Module implementiert werden:

- DNS-Modul: zur Verwaltung von DNS-Einträgen,
- DHCP-Modul: zur Verwaltung von DHCP-Konfigurationen und

	DHCP-Modul	DNS-Modul	Softwarelizenzverwaltung
Frontendkomponente	ja	ja	ja
Datenspeicherkomponente	ja	ja	ja
Backendkomponente	ja	ja	entfällt

Tabelle 1: Visualisierung der Komponenten- und Modulstruktur

- Softwarelizenzmanagement-Modul: als einfaches Werkzeug zur Verwaltung vorhandener und verwendeter Softwarelizenzen.

8.2.2 Strukturierung

Beim überwiegenden Teil der evaluierten Softwaretools zur Administrationsunterstützung waren die drei Standardebenen Frontend, Datenspeicherung und Backend sehr eng miteinander verbunden und typischerweise in einem Komplex gemeinsam implementiert. Ein weitverbreiteter Ansatz besteht in der Implementation dieser drei Ebenen gemeinsam im den Code des in PHP oder Perl geschriebenen Frontends, wobei die Datenspeicherung entweder in einfachen Textdaten oder (im fortgeschrittenen Fall) in einfachen MySQL-Tabellen erfolgt.

Eine Aufspaltung des Systems in die drei konzeptionellen Elemente Frontend, Datenspeicher und Backend hingegen ermöglicht eine optimale Implementation der jeweiligen Komponente, unabhängig von den anderen Komponenten, während bei einer gemeinsamen Implementation in einer Einheit an den jeweiligen Komponenten Abstriche gemacht werden müssen.

Die Strukturierung von SCASS als Gesamtsystem ist als Matrix vorstellbar. Dabei erfolgt die vertikale Strukturierung durch die Komponenten, die horizontale durch die Module. Damit ergibt sich die in Tabelle 1 dargestellte Komponenten-/Modulstruktur. Weiterhin muß natürlich darauf hingewiesen werden, daß innerhalb der Komponenten ein bestimmter Grundstock an quer durch alle Module genutzter gemeinsamer Funktionalität existiert. Dieser begründet sich darin, daß einfache, in mehreren (und teilweise in allen Modulen) benötigte Funktionalität als gemeinsam nutzbare Infrastruktur implementiert wurde, um unnötige Mehrfachimplementationen von Funktionalität zu vermeiden und die Strukturierung zu verbessern.

8.2.3 Konzept Frontendkomponente

Das Frontend soll gemäß den vorher beschriebenen Anforderungen als Webfrontend ausgelegt werden. Im Sinne der funktionalen Trennung der Komponenten wird das Frontend ausschließlich

- Dateneingabe und
- Datenausgabe

implementieren und sämtliche Eingaben an die Datenspeicherungskomponente weiterleiten beziehungsweise alle auszugebenden Daten von dieser Komponente beziehen. Die für den Anwender sichtbare Funktionalität des Frontends, welche zu implementieren ist, umfaßt dabei

- Hinzufügen von Daten,

- Anzeigen eingetragener Daten,
- Ändern eingetragener Daten,
- Löschen eingetragener Daten.

Als weitere für den Nutzer sichtbare Funktionalität bietet sich je nach Art des gerade laufenden Moduls die Suche über die eingetragenen Daten an.

Dabei wird die Frontendkomponente keinerlei syntaktische oder semantische Datenverifikation über die Eingabedaten implementieren, sondern diese lediglich auf geeignete Weise für die Datenspeicherungskomponente verkapseln und an diese weiterleiten. Zudem arbeitet das Frontend als Webanwendung praktisch transaktionsorientiert:

1. Maske lokal ausfüllen,
2. Daten an Server senden,
3. Ergebnis vom Server lesen,
4. Ergebnis lokal darstellen.

Damit nähert man sich - ganz nebenbei - konzeptuell wieder an die Arbeitsweise transaktionorientierter Terminalanwendungen aus der (hostbasierten) Frühzeit an. Die entscheidenden Vorteile dieses Ansatzes bestehen darin, daß

- man vom Client unabhängig ist - praktisch jeder hinreichend aktuelle Webbrowser eignet sich als Client und er kann praktisch überall laufen,
- man im Frontendsystem nur die minimale Frontend- aber keine Systemlogik implementieren muss und dieses somit deutlich weniger komplex und fehleranfällig wird.

Im Sinne einer möglichst einfachen Nutzung ist die Benutzbarkeit des Webfrontends mit möglichst allen Browserm, auch und insbesondere Texmodusbrowsern, ein Designziel. Dies schließt die Verwendung solcher Elemente wie

- im Browser laufendes Java,
- JavaScript,
- andere aktive Inhalte (z.B. AktiveX),
- DHTML,
- Frames

vollständig aus, da deren Verwendung den Einsatz bestimmter Browser verhindern würde. Weiterhin bedingt dies Zurückhaltung in den Bereichen

- Grafik,
- CSS [Bos99] [Jac98],
- Layoutkomplexität,

- Sprachkomplexität des verwendeten HTML,

um die Benutzung auch durch Webbrowser mit eingeschränkten Möglichkeiten zu erlauben. Da für die Benutzung des Webfrontends die Übertragung sicherheitskritischer Daten notwendig ist (privilegierte Zugangsdaten), muß die Übertragung zumindest bei den Teilen des Webfrontends, bei denen die sicherheitskritischen Daten übertragen werden, mit SSL [Kar96] verschlüsselt werden. Somit muß der zu verwendende Browser auch SSL-fähig sein.

Aus diesen Anforderungen bildeten sich schließlich die Anforderungen an die technische Gestaltung des Webfrontends von SCASS:

1. Verzicht auf die Elemente (Java, JavasScript, aktive Inhalte, DHTML, Frames) durch die der Einsatz bestimmter Browser verhindert würde.

2. Einsatz von CSS lediglich zur optischen Gestaltung, damit leidet bei fehlender CSS-Fähigkeit zwar etwas die Optik, die Funktionalität wird jedoch nicht beeinflußt.

3. Verzicht auf Grafiken zu Navigationszwecken und zur Inhaltsdarstellung. Die im Rahmen dieser Diplomarbeit vorgestellte erste Release-Version von SCASS wird komplett auf Grafiken im Webfrontend verzichten, da diese zur Implementation der Funktionalität derzeit nicht notwendig sind und die im Rahmen dieser Arbeit zu erstellende erste Version von SCASS sich ausschließlich auf die zu implementierende Funktionalität beschränken wird. Nicht direkt die Funktionalität betreffende Maßnahmen wie z.B. eine graphisch verbesserte Gestaltung des Frontends bleiben späteren Überarbeitungen vorbehalten.

4. Die gestellten Anforderungen an den Browser sind minimal:
   - Formulare mit POST,
   - verschachtelte Tabellen,
   - ansonsten einfaches und relativ minimales HTML,
   - Beherrschung von SSL.

5. Funktionalität mit verschiedenen Browsern muß sichergestellt und getestet werden:
   - Netscape,
   - Internet Explorer,
   - Mozilla,
   - Konqueror,
   - Opera,
   - links,
   - lynx,
   - w3m

   womit der überwiegende Teil derzeit verwendeter Browser abgedeckt ist.

Im Interesse einer einfachen Erweiterbarkeit soll dabei die Frontendkomponente möglichst flexibel und modular aufgebaut werden, spätere Erweiterungen, z.B. durch zusätzliche Module, sollen mit möglichst minimalem Aufwand in die Struktur des Frontends eingefügt werden können.

8.2.4 Konzept Datenspeicherungskomponente

Dabei handelt es sich um eine für die Leistungsfähigkeit des Gesamtsystems entscheidende Komponente. Im einfachsten Falle ist sie – ihrer Bezeichnung entsprechend – nur einfacher Datenspeicher, der z.B. als eine Sammlung von Textdateien implementiert werden kann. Diese Komponente mit einer Gruppe von einfachen Tabellen in einer relationalen Datenbank zu implementieren hat den Vorteil, das man ein eigenes Zugriffssystem für Dateien sowie die entsprechenden Parser einspart und sich stattdessen der Abfragesprache der Datenbank – üblicherweise SQL[15] – bedienen kann. Dies ist auch der von einigen der fortgeschritteneren unter den in den vorhergehenden Kapiteln betrachteten Administrationswerkzeugen verfolgte Ansatz. Die Möglichkeiten, die sich mit dem Einsatz eines leistungsfähigen, relationalen Datenbanksystems eröffnen, werden jedoch von diesen Werkzeugen praktisch kaum genutzt.

Dabei wird bereits in [Fin93] der Einsatz relationaler Datenbanksysteme als Werkzeuge zur Systemadministration beschrieben. Das dort beschriebene System namens SIMON basiert allerdings auf Oracle als Datenbank und setzt damit kommerzielle und relativ preisintensive Software voraus. Zudem ist SIMON als in-house Werkzeug speziell für den Anwendungsbereich und die Bedürfnisse einer bestimmten universitären Einrichtung – des Rensselaer Polytechnic Institute – abgestimmt und in seiner Komplexität für den Bereich kleinerer und mittlerer Netze ungeeignet. Die Unterlagen zu SIMON enthalten jedoch bereits ein wichtiges Konzept: die Verwendung der Datenbank nicht nur als reinen Datenspeicher, sondern zum Durchsetzen bestimmter Anforderungen an die Daten.

Eine Grundidee von SCASS ist die Benutzung der Dateinspeicherkomponente als „intelligenter Datenspeicher“. Die Dateinspeicherkomponente soll, soweit sinnvoll und möglich, die syntaktische semantische Korrektheit der in ihr gespeicherten Daten sicherstellen. Damit stellen sich verschiedene relativ hohe Anforderungen an diese Komponente:

- soweit möglich, Sicherstellung der syntaktischen Korrektheit der Daten (Beispiel: syntaktisch korrekte IP-Adresse für DNS-Zonefiles, syntaktisch korrekte MAC-Adresse für DHCP-Konfiguration),
- soweit möglich, Sicherstellung semantischer Korrektheit der Daten und damit Abfangen bestimmer Konfigurationsfehler (z.B. überlappende, widersprüchliche Daten, ungültige MAC-Adressen bei DHCP-Konfiguration),
- Speicherung der Daten in einem neutralen Format, welches nicht direkt von der letztlich verwendeten Implementation des unterstützten Dienstes abhängig ist, d.h. keine 1:1 Abbildung der Konfigurationsdateien einer bestimmten Dienstimplementation auf das Datenspeichermodell,
- Konsistenzsicherung der Daten.

Es soll – soweit sinnvoll und möglich – praktisch die gesamte Logik in die Datenbank verlagert werden. Für diesen Ansatz sprechen im Wesentlichen folgende Gründe:

- Wie bereits in 8.2.3 erwähnt, soll die Frontendkomponente ausschließlich auf die Aufgabe der Interaktion mit dem Nutzer – also zur Ein-/Ausgabe und Änderung von, sowie zur Recherche über Daten – hin ausgelegt werden.
- Zentraler Implementation der Systemlogik an einer Stelle.

[15]Structured Query Language

- Die Datenspeicherungskomponente ist konzeptionell der einzig richtige Ort zur Ablage der Systemlogik. An dieser Stelle laufen alle Daten des Systems zusammen. Wenn die Datenspeicherungskomponente die Systemlogik implementiert und die Einhaltung der Regeln derselben erzwingt ist ein Umgehen (z.B. durch Fehler im Frontend) nicht mehr möglich, da z.B. fehlerhafte Daten (soweit dies erkennbar ist) von der Datenspeicherungskomponente gar nicht erst akzeptiert werden.

8.2.5 Konzept Backendkomponente

Die dritte Komponente von SCASS ist schließlich die Backendkomponente. Sie soll dazu dienen, im Hintergrund bei Bedarf verschiedene, vom System ausgelöste Aktionen auszuführen. Da die Systemlogik als Bestandteil der „intelligenten" Datenspeicherungskomponente ausgeführt werden soll, erfolgt die Auslösung von Backendaktivitäten letztlich über entsprechende Mechanismen dieser Komponente.

Im Rahmen der Aufgaben von SCASS müssen u.a. Konfigurations- und ähnliche Dateien erzeugt werden. Dieses Problem wird u.a. in [Fin02a] behandelt, allerdings besteht der dort verfolgte Ansatz darin, diese Dateiinhalte in der Datenbank selbst zu generieren und mit den Datenbankmitteln zur Ein- und -ausgabe von Dateien und minimaler Backendunterstützung in die entsprechenden Dateien zu übertragen. Dieser Ansatz soll in SCASS nicht verfolgt werden, da ihn der Autor für ungünstig hält und die in [Fin02a] angeführten Gründe (Portierungsprobleme der überwiegend in C implementierten Backends zwischen Plattformen) sich bei SCASS nicht stellen. Die Erzeugung der entsprechenden Dateien soll bei SCASS vom Backend übernommen werden, welches zur Vermeidung und (wo dies nicht möglich ist) Minimierung von Portierungsproblemen auf eine geeignete, möglichst portable Weise implementiert werden soll.

Die Backendkomponente stellt die ausführende Komponente von SCASS dar, die verschiedene Aktionen ausführt. Ihre Aufgaben lassen sich zusammenfassen als:

- Erzeugen von Konfigurations- und anderen Dateien gemäß Vorgaben der Datenspeicherungskomponente,
- Ausführen von Aktionen auf Anstoss der Datenspeicherungskomponenten,
- Rückmeldung entsprechender Informationen an die Datenspeicherungskomponente.

Da eine direkte Auslösung des Backends durch die Datenspeicherungskomponente aus implementationstechnischen Gründen ungünstig ist, wird dafür ein anderer Ansatz gewählt: Die Backendkomponente startet in entsprechend festgelegten, regelmäßigen Abständen, prüft auf eventuelle „Aufträge" vom Backend und führt diese – wenn vorhanden – aus. Dies kann mit auf vielen Plattformen üblichen Systemwerkzeugen realisiert werden.

Der Ansatz, Konfigurationsdateien für Dienste aus Datenbanken zu erzeugen ist natürlich nicht neu, er wird u.a. in [Fin02b] und [Fin93] beschrieben. In [Fin02a] wird der Ansatz verfolgt, aus Portabilitätsgründen (man hatte offensichtlich mit den bis dahin verwendeten Backends zunehmende diesbezügliche Probleme) die Konfigurationsdateien bereits innerhalb der Datenbank mittels stored procedures zu erzeugen und sie dann nur noch als Dateien aus der Datenbank herausschreiben zu lassen. Der Autor hält dieser Ansatz für problematisch und ist der Meinung, daß dieser den aktuellen Möglichkeiten in diesem Bereich nicht vollständig Rechnung trägt. Die Probleme dieses Ansatzes sind nach Meinung des Autors u.a.:

- Höhere Spezialisierung der Entwickler ist notwendig: für die Unterstützung einer anderen Dienstimplementation sind Datenbankprogrammierer notwendig, die sich sowohl mit der Entwicklung von stored procedures für die jeweilige Datenbank (in den zitierten Papers handelt es sich um Oracle) als auch mit der zu unterstützenden Dienstimplemenation auskennen.

- Höherer Entwicklungsaufwand - stored procedures sind typischerweise deutliche schwieriger zu testen und zu debuggen als externer Code, der lediglich aus der Datenbank liest und diese dann in geeignete Formate transformiert.

- Grundsätzlich nicht optimale Wahl des Werkzeuges – gute Sysadmins sind meist mit einer oder mehreren Skriptsprachen (Perl, Python, ...), jedoch nur selten mit der Entwicklung von stored procedures für Datenbanken vertraut.

Aus diesem Grund verfolgt der Autor einen anderen Ansatz. Die Datenbank dient, wie bereits ausgeführt, als intelligenter, bestimmte Anforderungen an die Daten durchsetzender, Datenspeicher und das Backend zur Erzeugung von Dienstekonfigurationen. Konfigurationsdateien für unter UNIX laufende Dienste sind üblicherweise als Textdateien ausgelegt und die Konfigurationsdaten kommen praktisch in einem textähnlichen Format aus der Datenbank. Es empfiehlt sich somit für das Backend eine Implementatiossprache, die sich sowohl sehr gut für die Aufgaben der Texttransformation eignet als auch hochportabel ist. Diese Anforderungen werden von Perl sehr gut erfüllt, dazu kommt die bereits sehr große Verbreitung von Perl als Programmiersprache im Bereich der Systemadministration, was eine Erweiterung und Anpassung des zu erstellenden Backendcodes durch andere Sysadmins – eine geeignete klare und wartungsfreundliche Programmierung natürlich vorausgesetzt – deutlich vereinfachen wird. Daher hat sich der Autor für Perl als Implementierungssprachen für das zu erstellende Backend für SCASS entschieden.

8.2.6 Modulübergreifende Standards und das Infrastrukturkonzept

Trotz der unterschiedlichen Aufgaben und Anforderungen der verschiedenen Module gibt es mehrere Gemeinsamkeiten, die in allen Modulen und allen von ihnen genutzten Komponenten vorhanden sind. Diese sollen nun, gegliedert nach Komponenten, erläutert werden.

Die Präsentation der Benutzeroberfläche in der Frontendkomponente wird - u.a. durch Verwendung eines Kerns einheitlicher Hilfsroutinen - weitestgehend ähnlich gestaltet werden. Es wird eine Menüstruktur geben, die die verfügbaren Aktionen auflistet. Dazu gehören bei jedem Modul das

- Hinzufügen,
- Anzeigen,
- Ändern und
- Löschen

von Daten. Wenn das Modul über eine Backendkomponete verfügt – z.B. zum Erzeugen von Serverkonfigurationsdateien aus den Daten der Datenspeicherungskomponente und dem anschließenen Neuladen dieser Konfigurationsdateien in den Server - so wird es auch einen Eintrag zur Aktivierung dieses Backends geben. Die verschiedenen Eingabe-, Anzeige- und

Editiermasken und -formulare werden – soweit sinnvoll – einheitlich gestaltet. Mit diesen Vorgaben soll die Bedienung der Frontendkomponenten der jeweiligen Module vereinheitlicht und vereinfacht werden.

Da für die vollständige Benutzung von SCASS auch regelmäßig Passworte übertragen werden, muß diese Verbindung natürlich gesichert werden. Dies geschieht auf einfache Weise durch den Einsatz von SSL (für den Einsatz von SCASS ist ein SSL-fähiger Webserver ,z.B. Apache [Fou02] mit mod_ssl [Eng02] oder Apache-SSL [Lau02] notwendig). Die Frontendkomponente von SCASS unterstützt (bei geeigneter Konfiguration des Webservers) sowohl unverschlüsselt HTTP als auch verschlüsselt HTTPS, allerdings mit einer entscheidenden Einschränkung: es wird bei jedem Zugriff das verwendete Protokoll geprüft und bei unverschlüsseltem Zugriff sowohl die Anzeige sämtlicher Formulare (für Hinzufügen/Ändern/Löschen von Daten) als auch die Ausführung entsprechender Requests mit dem Hinweis verweigert, daß diese Funktionalität nur beim Zugriff über HTTPS zur Verfügung steht. Diese Maßnahme dient vor allem dazu, eine unverschlüsselte Verwendung der passwortgeschützten Funktionen „aus Versehen“ zu verhindern. Dieses Verhalten ist nicht abschaltbar.

Die Datenspeicherungskomponente wird durch eine relationale Datenbank implementiert. Die Struktur dieser Datenbank läßt sich in zwei funktionale Bereiche unterteilen:

- modulspezifische Datenbankstrukturen,
- generelle Infrastruktur von SCASS.

Die Teile der Datenbankstruktur die zur generellen Infrastruktur von SCASS gehören, umfassen sowohl gemeinsam modulübergreifend genutzte Funktionalität als auch infrastrukturinterne Funktionalität. Die interne Infrastruktur wird dabei vor allem durch die in der Datenbank gespeicherte Struktur des Menüsystems repräsentiert. Um bei einer Erweiterung von SCASS um weitere Module nicht jedesmal existierenden Code – nämlich u.a. den Code des Menüsystems - umschreiben zu müssen, wird die Struktur des Modul- und Menüsystems in der Datenbank gespeichert. Dabei reduzieren sich die für das Einbinden eines neuen Moduls notwendigen Schritte auf:

- Laden der Datenbankstrukturen des Moduls,
- Ablegen des Codes der Backendkomponente (sofern eine solche existiert) des Moduls an die entsprechende Position im Filesystem,
- Ablegen des Codes der Frontendkomponente des Moduls an der entsprechenden Position im Filesystem,
- Erweitern der in der Datenbank gespeicherten Modul- und Menüstruktur um die Informationen des neuen Moduls.

Dabei führt jeder Modul- und Menüeintrag in der Datenbank eine Information über seinen Zustand mit, die angibt, ob er aktiv ist oder nicht. Auf diese Weise kann durch Setzen der entsprechenden Zustandsinformationen ein Menüeintrag oder ein ganzes Modul deaktiviert – oder bei Bedarf auch wieder aktiviert werden.

Für die Verwaltung der Modul- und Menüstruktur in der Datenbank werden zwei Tabellen benötigt:

- eine Tabelle für die Modulverwaltung, diese enthält:

  - den Namen des Moduls,
  - die Beschreibung des Modules,
  - die Basis-URL zum Aufruf der Frontendkomponente des Moduls,
  - ein Flag, ob die Frontendkomponente des Moduls aktiv sein soll oder nicht,

- eine Tabelle für die Menüverwaltung, diese enthält ihrerseits:
  - einen Verweis auf das Modul, zu dem der Menüeintrag gehört,
  - den Namen des Menüeintrags,
  - die Beschreibung des Menüeintrags,
  - die Basis-URL zum Aufruf des Menüeintrags über die Frontendkomponente des zugehörigen Moduls,
  - ein Flag, ob der Menüeintrag aktiv (also sichtbar) sein soll oder nicht.

Weiterhin werden von verschiedenen Teilen der Infrastruktur der Frontendkomponente die Namen und Beschreibungen der verwendeten Tabellen benötigt. Diese Information wird in einer eigenen Tabelle gespeichert.

Für SCASS muß, wie bei jedem komplexeren System, ein gewisser Satz an Systemkonfigurationsdaten gespeichert werden. Dies kann zum einen auf die allgemein übliche Art über Konfigurationsdateien geschehen, wobei dieser Ansatz den Nachteil hat, daß ein entsprechendes Subsystem zur Unterstützung von Konfigurationsdateien notwendig wird. Andererseits verwendet SCASS bereits eine leistungsfähige Datenspeicherungskomponente. Daher hat sich der Autor entschieden, die Konfiguration von SCASS zweistufig zu verwalten:

- einfache Textdateien, diese enthalten nur die zum Zugriff auf die Datenspeicherungskomponente notwendigen Informationen,
- eine eigene Struktur in der Datenspeicherungskomponente, welche die eigentlichen Konfigurationsdaten enthält.

Um die unterschiedlichen Privilegien der Frontendkomponente und der Backendkomponente besser zu trennen, verwenden beide unterschiedliche Konfigurationsdateien.

Die Speicherung der Konfiguration in der Datenspeicherungskomponente ist als einfache Tabelle in der relationalen Datenbank gelöst. Diese implementiert ein typisches Konfigurationssystem nach dem Ansatz von Schlüsseln und zugeordneten Werten.

Daraus ergibt sich schließlich das Datenbankmodell der SCASS-Infrastruktur, welches im Folgenden sowohl in Form der SQL-Tabellendefinition als auch in Form einer Visualisierung der Tabellenstruktur dargestellt ist. Die vorher beschriebenen Tabellen erhielten dabei die Tabellennamen:

- `scass_config`: die Konfigurationsdaten von SCASS,
- `scas_table_description`: die Namen und Beschreibungen der Tabellen,
- `scass_fes_modules`: die Modulstruktur für die Frontendkomponente,
- `scass_fes_menu`: die Menüstruktur für die Frontendkomponente.

Die einzelnen Datenfelder der Tabellen haben dabei inhaltlich folgende Bedeutung:

Datenfeld	SQL-Definition
id	serial primary key
module_name	varchar unique not NULL
module_description	varchar not NULL
module_url_base	varchar not NULL
active	boolean
comment	text default NULL

Tabelle 2: SCASS-Infrastruktur, Datendefinition für die Tabelle `scass_fes_modules`

Datenfeld	SQL-Definition
id	serial primary key
module	int references scass_fes_modules not NULL
menu_name	varchar not NULL
menu_description	varchar not NULL
menu_url_base	varchar not NULL
active	boolean
comment	text default NULL

Tabelle 3: SCASS-Infrastruktur, Datendefinition für die Tabelle `scass_fes_menu`

Datenfeld	SQL-Definition
id	serial primary key
table_name	varchar unique not NULL
description	varchar
comment	text default NULL

Tabelle 4: SCASS-Infrastruktur, Datendefinition für Tabelle `scass_table_description`

Datenfeld	SQL-Definition
id	serial primary key
key	varchar unique not NULL
value	varchar default NULL
comment	text default NULL
last_update	timestamp not NULL default current_timestamp

Tabelle 5: SCASS-Infrastruktur, Datendefinition für die Tabelle `scass_config`

- Tabelle `scass_fes_modules`:
  - `module_name`: intern verwendeter Name des Moduls,
  - `module_description`: von der Frontendkomponente angezeigter Modulname,
  - `module_url_base`: URL die auf die Einstiegsseite des Moduls in der Frontendkomponente verweist, relativ zur Wurzel der Installation der Frontendkomponente,
  - `active`: Modul ist aktiv (in der Frontendkomponente angezeigt und dort benutzbar) oder nicht,
- Tabelle `scass_fes_menu`:
  - `module`: Verweis auf das Modul, zu dem der Menüeintrag gehört,
  - `menu_name`: interner Name des Menüeintrags, dieser Bezeichner ist modullokal und muß nur innerhalb des Moduls eindeutig sein,
  - `menu_description`: die von der Frontendkomponente als Linktext angezeigte Beschreibung des Menüeintrags,
  - `menu_url_base`: URL mit der die von der Frontendkomponente angezeigte Menübeschreibung als Link hinterlegt ist,
  - `active`: Menüeintrag ist aktiv (in der Frontendkomponente angezeigt und dort benutzbar) oder nicht,
- Tabelle `scass_table_descriptions`:
  - `table_name`: Name der Tabelle in der Datenspeicherungskomponente,
  - `description`: von der Frontendkomponente angezeigte Tabellenbeschreibung,
- Tabelle `scass_config`:
  - `key`: Schlüssel (Konfigurationsvariable),
  - `value`: Wert des Schlüssels.

Weiterhin enthält die Datenbank Infrastrukurinformationen über jedes Modul. Diese wird von verschiedenen gemeinsamen Standardroutinen genutzt, die von SCASS als allgemein verfügbare Funktionalität für alle Module angeboten wird.

Mit Ausnahme der zur Infrastruktur gehörenden Datenstrukturen führen alle Datensätze ein besonderes Feld mit sich, den Zeitstempel ihrer letzten Änderung – Abweichungen von dieser Vorgabe sind Ausnahmen und werden entsprechend begründet. Bei den Ausnahmen handelt es sich typischerweise um Daten, die im laufenden Betrieb von SCASS nicht mehr verändert und nur ein einziges Mal – bei der Installation des Systems – geschrieben werden. Der Zeitstempel wird über geeignete Mechanismen automatisch aktuell gehalten und soll nicht automatisch aktualisiert werden, manuelle Änderungen an diesen Zeitstempeln werden standardmäßig verhindert.

Die Backendkomponenten der Module werden einheitlich strukturiert. Diese Strukturierung soll eine möglichst hohe Flexibilität sicherstellen und Erweiterungen vereinfachen. Es gelten für die Strukturierung die folgenden Vorgaben:

- Jede Backendkomponente eines Moduls enthält ein „Masterscript“, welches die Aufgaben des Backends ausführt.

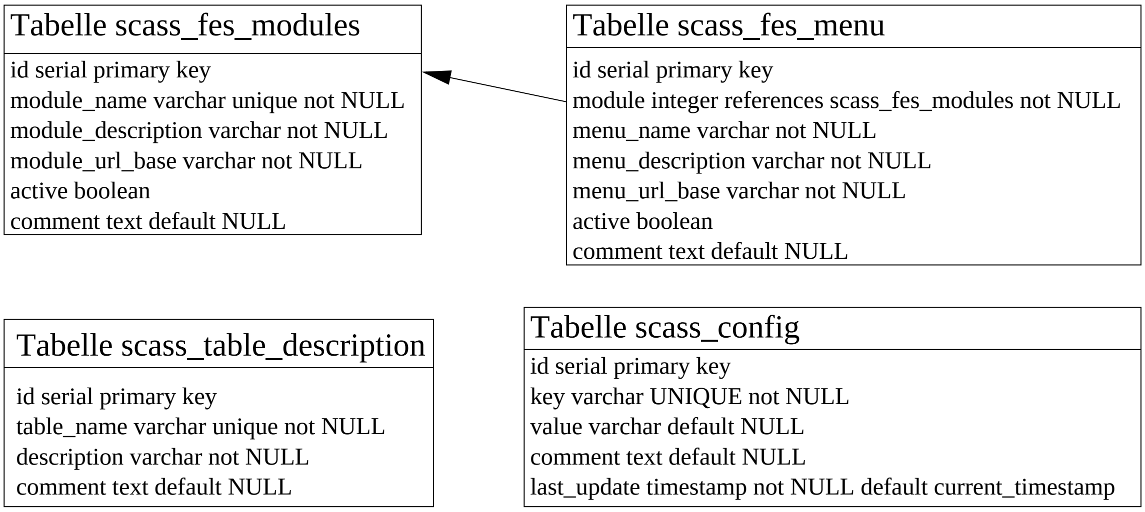

Abbildung 1: SCASS-Infrastruktur, visuelle Darstellung des Datenbankmodells

- Sind Teilaufgaben in stark unterschiedlicher Form implementierbar – z.B. die Erzeugung von Konfigurationsdateien für verschiedene Implementationen eines Dienstes, so sind diese als bedarfsweise zu ladende Codemodule auszulegen. Das jeweils zu ladende Codemodul wird entsprechenden Informationen in der Datenspeicherungskomponenten entnommen.

- Gemeinsam genutzte Funktionalität – z.B. Datenbankzugriffsmodule werden einmal zentral als gemeinsame Infrastruktur in Form ladbarer Codemodule implementiert und bei Bedarf geladen.

8.2.7 Konzept DNS-Modul

Das DNS-Modul dient der einfachen Verwaltung lokaler Nameserver. Die zu verwaltenden Informationen sind dabei:

- Zuordnung von Hostnamen zu IP-Adressen,

- Zuordnung von Aliasnamen zu Hostnamen,

- Strukturierung des Namensraumes in Zonen,

- Infrastrukturinformation der Zonen (u.a. spezielle DNS-Einträge wie z.B. zuständige Mailserver),

die auf geeignete Weise verwaltet werden müssen. Damit ergeben sich für dieses Modul folgende Aufgaben:

- Speicherung der vorher genannten Informationen unter Verwendung der Datenspeicherungskomponente,

- Durchsetzen bestimmter Vorgaben über Struktur und Inhalt der Daten unter Verwendung der „Intelligenz“ der Datenspeicherungskomponente,

- einfache Verwaltung (hinzufügen, anzeigen, ändern, löschen) dieser Informationen mittels der Frontendkomponente,
- Überführung dieser Information aus der Datenspeicherungskomponente in ein für Nameserversoftware verwendbares Format und Anstoßen der Datenübernahme in die Nameserversoftware durch die Backendkomponente.

Da die Daten in einer relationalen Datenbank gespeichert werden sollen, ist zuerst das Datenmodell zu entwerfen. Dazu muß die Struktur der zu speichernden Daten geklärt werden. Die Strukturierung des Domain Name Systems ist u.a. in [Moc87a] [Moc87b] und weiteren niedergelegt. Daraus sowie aus dem Ziel einer möglichst einfachen Administrierbarkeit auch für weniger gute, mit der Problematik vertraute Admins, entstand die folgende Strukturierung des Datenmodells:

- Die Daten werden grundsätzlich nach Zonen strukturiert. Jeder Datensatz ist eindeutig einer Zone zugeordnet. Für jede Zone existiert ein Datensatz, der die Daten der Zone selbst (Domainname, primärer Nameserver, Domainadmin, ...) enthält. Jede Zone hat einen eindeutigen Namen, der nicht mit dem Domainnamen der Zone identisch sein muß. Er dient zur einfacherer Unterscheidung unterschiedlicher Zonen zu administrativen Zwecken. Die Zonendatensätze werden in einer eigenen Tabelle `dns_zone` gespeichert.
- Die Zuordnung von Hostnamen zu IP-Adressen erfolgt in der Tabelle `dns_mapping`, in den Datensätzen sind die IP-Adresse, der Hostname, der Verweis auf die Zone, zu der der Eintrag gehört sowie weitere Daten abgelegt.
- Die Zuordnung von DNS-Aliasnamen zu Hostnamen erfolgt in der Tabelle `dns_alias`, welche in den Datensätzen den Aliasnamen, den Verweis auf den Hostnamen, den Verweis auf die Zone, zu der der Eintrag gehört sowie weitere administrative Daten speichert.
- In einer Tabelle `dns_special_types` wird eine Liste an unterstützten DNS-Recordtypen mit besonderer Bedeutung (MX, NS, ...) gespeichert, die vom DNS-Modul unterstützt werden.
- Diese Tabelle wird von den in der Tabelle `dns_special` abgelegten Datensätzen verwendet. Deren Datensätze enthalten einen Verweis auf ihre Zone, auf den DNS-Recordtyp, den Wert des Eintrag sowie weitere administrative Informationen.

Die folgenden Tabellen enthalten die Definitionen der resultierenden Tabellen in dem für PostgreSQL spezifischem SQL-Dialekt. Eine grafische Darstellung der Tabellenstruktur des DNS-Moduls ist in Abbildung 2 dargestellt.

Weiterhin sollen die Daten noch weitere Bedingungen erfüllen, die sich mit den Mitteln der Tabellendefinition nicht direkt ausdrücken lassen. Diese werden durch Trigger beim Einfügen oder Ändern der Daten überprüft, die nur Daten akzeptieren, welche die Bedingungen erfüllen. Für das DNS-Modul sind diese Bedingungen:

- Für `dns_zone.zone_name`: es werden nur alphanumerische Zeichen (a-z, A-Z, 0-9) sowie Punkt, Unterstrich und Bindestrich akzeptiert. Grund dieser Einschränkung ist die Verwendung dieses Datenfeldes zur Erzeugung von Dateinamen im Backend.

Datenfeld	SQL-Definition
id	serial primary key
zone_name	varchar unique not NULL
domain	varchar not NULL
network	varchar not NULL
serial	integer not NULL default 0
last_update	timestamp not NULL default current_timestamp
primary_ns	varchar not NULL
refresh	int not NULL
retry	int not NULL
expire	int not NULL
minimum	int not NULL
zone_admin	varchar not NULL
comment	text
last_commit	timestamp not NULL default 'epoch'
do_commit	boolean not NULL default false

Tabelle 6: DNS-Modul, Datendefinition für die Tabelle `dns_zone`

Datenfeld	SQL-Definition
id	serial primary key
zone	integer references dns_zone not NULL
last_update	timestamp not NULL default current_timestamp
IP	inet not NULL
hostname	varchar not NULL
comment	text
txt	varchar default NULL
hinfo	varchar default NULL

Tabelle 7: DNS-Modul, Datendefinition für die Tabelle `dns_mapping`

Datenfeld	SQL-Definition
id	serial primary key
zone	integer references dns_zone not NULL
last_update	timestamp not NULL default current_timestamp
name	varchar not NULL
target	integer references dns_mapping not NULL
comment	text

Tabelle 8: DNS-Modul, Datendefinition für die Tabelle `dns_alias`

Datenfeld	SQL-Definition
`id`	`serial primary key`
`type`	`varchar not NULL`

Tabelle 9: DNS-Modul, Datendefinition für die Tabelle `dns_special_types`

Datenfeld	SQL-Definition
`id`	`serial primary key`
`type`	`integer references dns_special_types not NULL`
`weight`	`integer`
`zone`	`integer references dns_zone not NULL`
`value`	`varchar not NULL`
`last_update`	`timestamp not NULL default current_timestamp`
`comment`	`varchar`

Tabelle 10: DNS-Modul, Datendefinition für die Tabelle `dns_special`

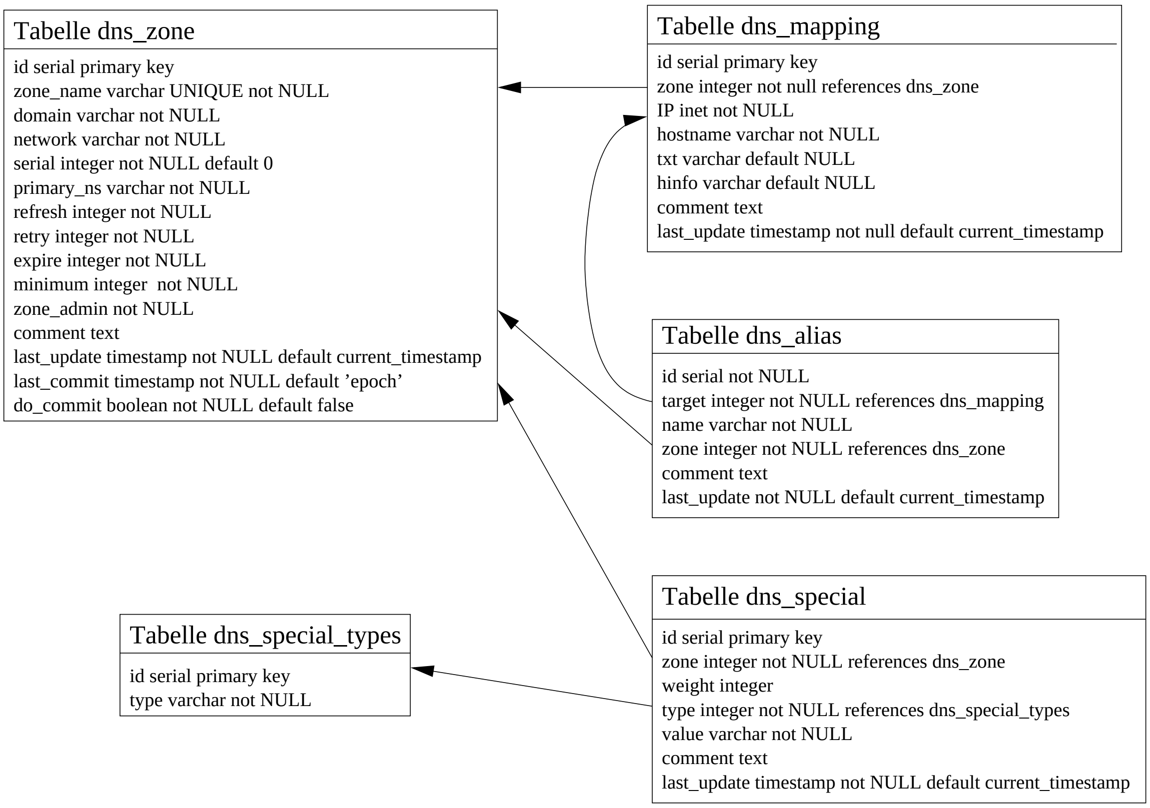

Abbildung 2: DNS-Modul, visuelle Darstellung des Datenbankmodells

- Für `dns_mapping.hostname` sowie `dns_alias.name`: es werden nur alphanumerische Zeichen (a-z, A-Z, 0-9) sowie Punkt und Bindestrich akzeptiert, das erste Zeichen muß ein Buchstabe sein, der Name muß mindestens ein Zeichen lang sein. Grund dieser Einschränkungen sind die Anforderungen nach [Moc87a].

Die Überprüfung dieser Bedingungen erfolgt durch eigene stored procedures, welche durch die bereits erwähnten Trigger ausgelöst werden.

Die Backendkomponente des DNS-Moduls ist darauf ausgelegt, entweder regelmäßig (z.B. durch einen cron-Job) oder nur bei Bedarf ausgeführt zu werden. Ihre Aufgabe besteht darin, aus den in der Datenbankkomponente gespeicherten Daten die Konfigurationsdateien für den konfigurierten Nameserver zu erzeugen sowie diesen zum Neueinlesen dieser Konfigurationsdateien zu veranlassen. Sie wird durch ein Perlscript implementiert, welches – je nach verwendeten Optionen – drei verschiedene Verhaltensmodi aufweist:

- Standardmodus: Überprüfen der Zonen auf Schreibanforderung, Zonen mit gesetzter Schreibanforderung werden bearbeitet, anschliessend wird diese Schreibanforderung gelöscht sowie der Zeitstempel der Ausführung in der Datenspeicherungskomponente vermerkt. Die Anzahl der bearbeiteten Zonen wird zurückgeliefert.

- Testmodus: Überprüfen der Zonen auf Schreibanforderung, die Anzahl der Zonen mit gesetzter Schreibanforderung wird zurückgeliefert, es werden keinerlei Daten verändert.

- Zwangsmodus: Die Zonen werden unabhängig vom Status der Schreibanforderung für die jeweiligen Zonen bearbeitet, die Schreibanforderung wird zurückgesetzt sowie der Zeitstempel der Ausführung in der Datenspeicherungskomponente vermerkt. Die Anzahl der bearbeiteten Zonen wird zurückgeliefert.

Das Backendmodul wird für die Unterstützung beliebiger DNS-Serverimplementationen ausgelegt indem der Code zum Generieren der Konfigurationsdaten sowie zum Neuladen dieser Daten in den DNS-Server, als bei Bedarf ladbares Codemodul ausgelegt wird. Das jeweils zu verwendende Codemodul wird von einer entsprechenden Konfigurationsoption in der Datenspeicherungskomponente der Infrastruktur von SCASS bestimmt. Im Rahmen dieser Arbeit wird aus Gründen der zeitlichen Beschränkung allerdings nur ein Modul für eine einzelne DNS-Serverimplementation entwickelt werden, welches als Implementationsbeispiel für weitere Module dienen, gleichzeitig aber vollständig einsatzfähig sein soll.

8.2.8 Konzept Softwarelizenzmanagement-Modul

Vorhandene kommerzielle Ansätze zum Softwarelizenzmanagement gehen das Problem üblicherweise aus der Sicht des Softwareherstellers an und zielen darauf ab, die Lizenzbestimmungen des Herstellers durchzusetzen sowie den Einsatz von mehr Lizenzen, als der Kunde gekauft hat, zu verhindern. Ein Standardprodukt in diesem Bereich ist FLEX*lm* [Div02]. Im Bereich freier Software gibt es relativ wenig zum Thema Softwarelizenzmanagement, die vorhandenen Lösungen beschäftigen sich meist mit dem Management von Lizenzschlüsseln.

Aus der Sicht des Sysadmins stellen sich jedoch im Bereich des Softwarelizenzmanagements ganz andere Probleme. Hier geht es darum, den Überblick über die Software zu behalten und jederzeit nachvollziehen zu können:

- Welche Software/Softwarelizenzen sind überhaupt vorhanden?

- Welche Software, welche Softwarelizenzen sind im Einsatz?
- Wo sind sie im Einsatz?
- Wo ist die Software als Installationsquelle zu finden?

Sind diese Informationen ersteinmal vorhanden, so können daraus die Antworten auf die folgenden Fragestellungen abgeleitet werden:

- Ist die Firma/Einrichtung unter- oder überlizensiert? Müssen Lizenzen nachgekauft werden oder sind zuviele (ungenutzte) Lizenzen vorhanden?
- Ist die Software überhaupt richtig verteilt, dort wo sie auch benötigt wird?
- Ergeben sich Möglichkeiten zur Konsolidierung der Softwarelandschaft?
- Mit welchen Kosten muß bei einem Upgrade bestimmter Softwarepakete/Lizenzen gerechnet werden?
- Welche Einsparpotentiale ergeben sich beim Umstieg auf andere Lizenzmodelle, andere Software oder durch den Umstieg von kommerzieller auf freie Software?

Dementsprechend versteht sich das Softwarelizenzmanagement-Modul in SCASS als Werkzeug für den Sysadmin um den Überblick über die Softwarelandschaft im Netzwerk zu behalten, was gerade in kleineren Netzen nicht immer gegeben ist.

Was muß überhaupt an Informationen erfaßt werden, um dieses Ziel zu erreichen? Auf jeden Fall die Software selbst, das ist offensichtlich. Verschiedene weitere Informationen, wie zum Beispiel die Plattform, der Hersteller, die Softwarekategorie sind ebenfalls wichtig. Bei freier Software kann die jeweilige Lizenz eine Rolle spielen. Die Anzahl vorhandener und benutzter Lizenzen sind notwendige Informationen. Um die Planung von Installationen zu unterstützen wird der Begriff der „reservierten“ Software eingeführt. Darunter ist Software zu verstehen, die noch nicht installiert, aber dafür bereits verplant und somit bei der Betrachtung der Lizenzsituation zu beachten ist. Damit ergibt sich an zu verwaltenden Informationen:

- die Plattform, auf der die Software läuft, z.B. i386-linux für Linux auf Intel x86 Prozessoren,
- die Softwareklasse, der die Software zuzuordnen ist, dies könnte zum Beispiel „Textverarbeitung“ für Microsoft Word sein,
- der Softwarehersteller, zum Beispiel Microsoft Corporation,
- die Lizenz, unter der die Software lizensiert ist, z.B. die Artistic License für Perl,
- der Name der Software, dieser wird auch den Versionsbezeichner enthalten, da die Praxis zur Versionsbenennung äußerst unterschiedlich ist und eine einheitlich getrennte Erfassung daher nicht als sonderlich sinnvoll erscheint, Beispiele hierfür sind „Microsoft Word 6.0“ und „Microsoft Word 2000“,
- eine kurze Beschreibung der Software,
- die Gesamtanzahl vorhandener Lizenzen,

- die Anzahl installierter Lizenzen,
- die Anzahl reservierter Lizenzen,
- aus obigem ergibt sich die Anzahl noch freier Lizenzen, diese Information muß somit nicht gespeichert werden,
- die Installationsquelle, z.B. ein gemeinsames Netzwerkdateisystem oder (für CD-ROMs) eine Regalposition,
- Installationen (es muß die Anzahl der installierten Lizenzen sowie der Ort der Installation erfaßt werden).

Weiterhin kann bei einigen Werten von bestimmten Annahmen ausgegangen werden, die bereits im Datenmodell mit erfaßt werden sollten:

- Anzahl vorhandener Lizenzen ≥ 0,
- Anzahl reservierter Lizenzen ≥ 0,
- Anzahl installierter Lizenzen ≥ 0.

Ebenfalls gilt es, bestimmte Daten innerhalb des Modells automatisch abzugleichen. Wenn zum Beispiel eine Installation hinzufgefügt, gelöscht oder geändert wird, so muß für die installierte Software automatisch die Anzahl installierter Lizenzen aktualisiert werden. Diese Aufgabe wird von automatisch ausgelösten Datenbankfunktionen (Triggern) gelöst, die als stored procedures in der Datenbank zu implementieren sind.

Damit ergibt sich das, sowohl als SQL-Tabellendefinition als auch in grafischer Form dargestellte, Datenmodell für das Softwarelizenzmanagement-Modul.

Die Daten wurden dabei auf die Tabellen

- `lm_platform`: Plattformen,
- `lm_vendor`: Softwarehersteller,
- `lm_class`: Softwareklassen,
- `lm_license`: Softwarelizenzen,
- `lm_software`: Software,
- `lm_installation`: Installationen

aufgeteilt.

Datenfeld	SQL-Definition
id	serial primary key
platform_name	varchar unique not NULL
comment	text
last_update	timestamp not NULL default current_timestamp

Tabelle 11: Softwarelizenzmanagement-Modul, Datendefinition für lm_platform

Datenfeld	SQL-Definition
id	serial primary key
vendor_name	varchar unique not NULL
comment	text
last_update	timestamp not NULL default current_timestamp

Tabelle 12: Softwarelizenzmanagement-Modul, Datendefinition für lm_vendor

Datenfeld	SQL-Definition
id	serial primary key
class_name	varchar unique not NULL
comment	text
last_update	timestamp not NULL default current_timestamp

Tabelle 13: Softwarelizenzmanagement-Modul, Datendefinition für lm_class

Datenfeld	SQL-Definition
id	serial primary key
license_name	varchar unique not NULL
license_text	text
comment	text
last_update	timestamp not NULL default current_timestamp

Tabelle 14: Softwarelizenzmanagement-Modul, Datendefinition für lm_license

Datenfeld	SQL-Definition
id	serial primary key
software_name	varchar unique not NULL
platform	int references lm_platform not NULL
vendor	int references lm_vendor not NULL
class	int references lm_class not NULL
license	int references lm_license not NULL
description	text
comment	text
total_licenses	int check (total_licenses > -1) not NULL default 0
installed_licenses	int check (installed_licenses > -1) not NULL default 0
reserved_licenses	int check (reserved_licenses > -1) not NULL default 0
install_source	varchar
last_update	timestamp not NULL default current_timestamp
last_notify	timestamp not NULL default 'epoch'

Tabelle 15: Softwarelizenzmanagement-Modul, Datendefinition für `lm_software`

Datenfeld	SQL-Definition
id	serial primary key
software	int references lm_software not NULL
location	varchar
count	int check (count > -1) not NULL default 0
comment	text
last_update	timestamp not NULL default current_timestamp

Tabelle 16: Softwarelizenzmanagement-Modul, Datendefinition für `lm_installation`

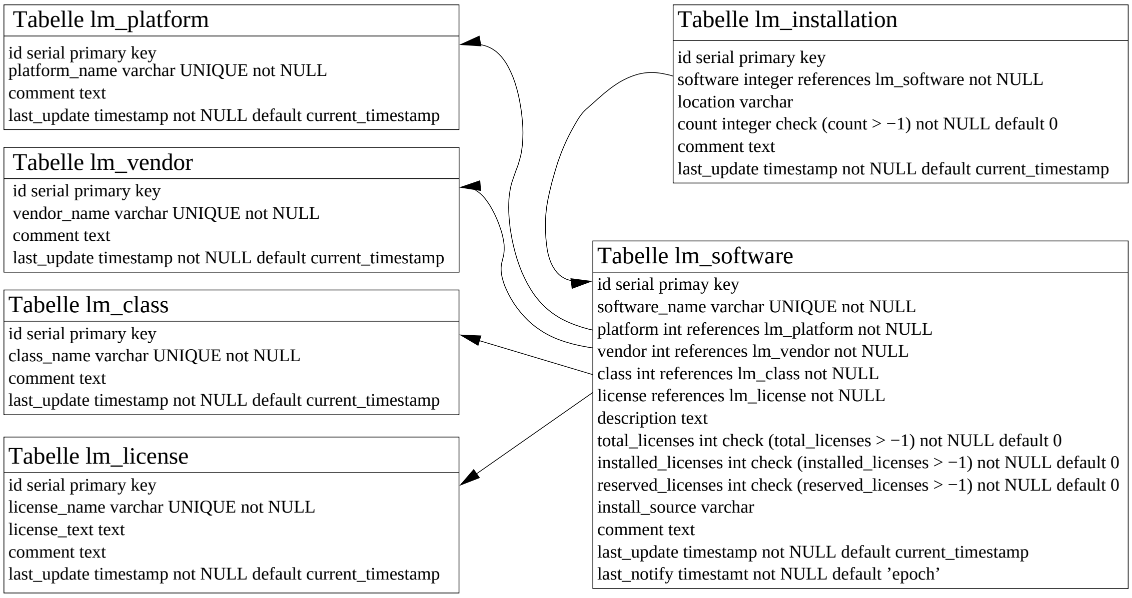

Abbildung 3: Softwarelizenzmanagement-Modul, Visualisierung des Datenbankmodells

8.2.9 Konzept DHCP-Modul

Das DHCP-Modul dient zur webfrontendbasierten Administration von DHCP-Servern. Dabei können beliebig viele DHCP-Server unabhängig voneinander verwaltet werden. Die Konfiguration wird nach Netzen gegliedert (in an Anlehnung an die von DHCP-Servern versorgten Netze) die jeweils eine vollständige Konfiguration eines DHCP-Servers darstellen. Innerhalb einer solchen Konfiguration werden zwei Objekttypen verwaltet:

- Hosts - einzelne, konkret durch die Hardwareadresse ihrer Ethernetkarte spezifizierte Hosts sowie
- Ranges - Netzbereiche, die einen bestimmten IP-Bereich überdecken und die einen „Adressvergabepool“ darstellen, aus dem jedem Host ohne feste Definition eine verfügbare Adresse zugeteilt wird.

Dabei gilt es natürlich, einige Regeln zu beachten. So muß sichergestellt werden, daß innerhalb eines Netzes:

- keine doppelten Hostdefinitionen auftreten - eine Hostdefinition gilt dann als doppelt, wenn die IP-Adresse und die Hardwareadresse der Ethernetkarte für beide Einträge übereinstimmen,
- sich Definitionen von Netzbereichen nicht überlappen,
- keine Kollisionen zwischen Host- und Netzbereichsdefinitionen auftreten, daß also keine Hosts mit Adressen innerhalb von Netzbereichen und keine die IP-Adressen von Hosts überdeckenden Netzbereiche eingetragen werden.

Dies muß mit geeigneten Datenbankmechanismen durchgesetzt werden.

Für die DHCP-Konfiguration werden dabei verschiedene, teils optionale, teils erforderliche Daten erfaßt:

- globale Daten für jedes Netz:
  - der Netzname als eindeutiger Bezeichner des Netzes,
  - die Subnetzmaske,
  - die Standardlebensdauer eines vergebenen DHCP-Eintrags,
  - die maximale Lebensdauer eines vergebenen DHCP-Eintrags,
  - eine Liste von NTP[16]-Servern (optional),
  - eine Liste von Routern (optional),
  - der Domainname (sollte gesetzt werden, ist aber optional),
  - eine Liste von DNS-Servern (unbedingt zu empfehlen, aber ebenfalls optional),
- für die direkte Zuweisung von Einträgen zu Hosts:
  - das zugehörige DHCP-Netz,
  - der symbolische Hostname,
  - die Hardwareadresse der Ethernetkarte,
  - die zugewiesene IP-Adresse,
- für die freie Vergabe von IPs aus vordefinierten Adressbereichen:
  - das zugehörige DHCP-Netz,
  - die Subnetz-Definition,
  - die Startadresse des IP-Bereichs,
  - die Endadresse des IP-Bereichs,
  - eine Liste von Routern.

Anhand der entwickelten Strukturierung ergebt sich eine Aufteilung der Daten in drei Tabellen:

- `dhcp_net`: für die Netzdefinitionen,
- `dhcp_host`: für die Hostkonfigurationen,
- `dhcp_range`: für die IP-Adressbereichsdefinitionen.

[16]Network Time Protocol = ein Dienst zum Abgleichen der Uhrzeit von Computern über ein TCP/IP-Netz

Datenfeld	SQL-Definition
id	serial primary key
net_name	varchar unique not NULL
subnet_mask	inet not NULL
default_lease_time	integer check (default_lease_time > 0)
	not NULL default 600
max_lease_time	integer check (max_lease_time >= default_lease_time)
	not NULL default 7200
ntp_servers	varchar default NULL
routers	varchar default NULL
domain_name	varchar default NULL
domain_name_server	varchar default NULL
comment	text
last_update	timestamp not NULL default current_timestamp
last_commit	timestamp not NULL default 'epoch'
do_commit	boolean not NULL default false

Tabelle 17: DHCP-Modul, Datendefinition für `dhcp_net`

Datenfeld	SQL-Definition
id	serial primary key
net	integer NOT NULL references dhcp_net
hostname	varchar not NULL
ethernet_address	macaddr not NULL
fixed_address	inet not NULL
comment	text
last_update	timestamp not NULL default current_timestamp

Tabelle 18: DHCP-Modul, Datendefinition für `dhcp_host`

Datenfeld	SQL-Definition
id	serial primary key
net	integer NOT NULL references dhcp_net
subnet	cidr not NULL
range_from	inet not NULL
range_to	inet not NULL check (range_from
routers	varchar
comment	text
last_update	timestamp not NULL default current_timestamp

Tabelle 19: DHCP-Modul, Datendefinition für `dhcp_range`

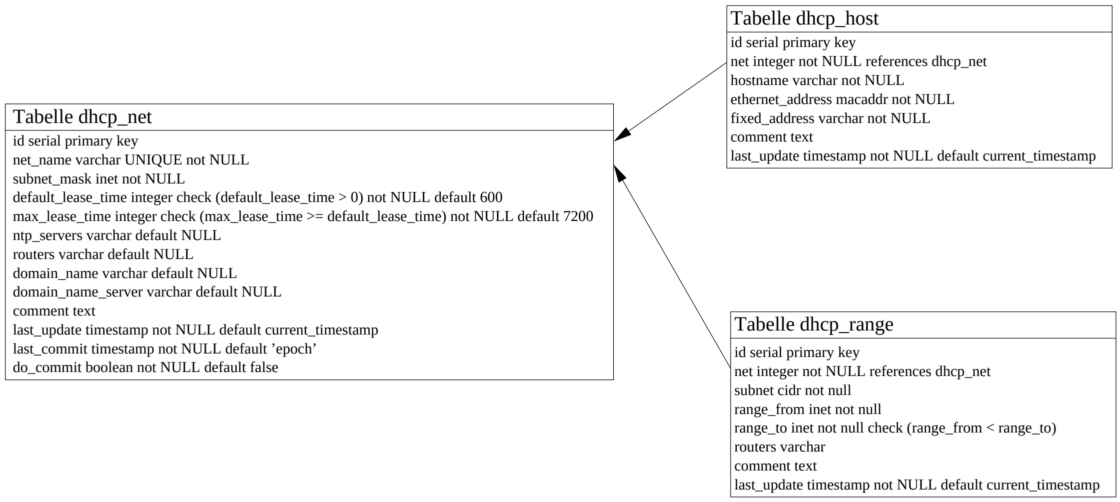

Abbildung 4: DHCP-Modul, grafische Darstellung der Datenbankstruktur

8.3 Implementation

Aufgrund des Umfangs des geschriebenen Codes (ca. 11.000 Zeilen Code) zieht es der Autor vor, im Interesse der Kürze des Textes dieser Arbeit auf detaillierte Zitate aus dem implementierten Code zu verzichten. Es soll lediglich auf die Struktur der Implementation sowie auszugsweise auf konkrete Implementationsdetails eingegangen werden. Da der Code auch mit Hinblick auf spätere Wartung und Weiterentwicklung geschrieben wurde, sollte er – sowohl durch die geeignet „sprechend" gewählten Bezeichner, als auch durch die Kommentare – hinreichend selbst zu seinem Verständnis betragen.

8.3.1 Implementation der Frontendkomponente

Da die Frontendkomponente als Webfrontend implementiert werden soll, galt es, eine dafür geeignete Implementierungsplattform zu finden. Die Anforderungen an die Plattform waren dabei unter anderem:

- Portabilität,
- freie Verfügbarkeit als freie Software,
- umfangreiche Unterstützung der dynamischen Erzeugung von Webseiten,
- gute Unterstützung von Datenbanken,
- gute Performanceerwartungen für die zu implementierende Lösung,
- gute Erweiterbarkeit.

Der Autor entschied sich schließlich für den Einsatz von PHP[Gro02] unter dem Apache-Webserver[Fou02]. Diese Plattform zeichnet sich durch

- umfangreiche Unterstützung für die Entwicklung von Webanwendungen,
- sehr gute Datenbankanbindung, Unterstützung des Zugriffs auf zahlreiche Datenbanken,
- umfangreiche Funktionalität bereits im Standardumfang, gute Erweiterbarkeit mit Zusatzmodulen

aus und eignet sich nach den Erfahrungen des Autors sehr gut zur Implementation dynamischer, datenbankbasierter Webanwendungen und -frontends. Aus Gründen der Aktualität wurde die derzeit aktuelle Version PHP4 zur Implementation der Frontendkomponente von SCASS ausgewählt.

Die übliche Vorgehensweise bei der PHP-Programmierung, insbesondere bei sehr kleinen PHP-Anwendungen, besteht darin, den PHP-Code abschnittsweise in umgebendes HTML einzubetten. Dies mag bei bis zu einem recht kleinen Umfang der Anwendungskomplexität durchaus noch akzeptabel sein, bei größeren Anwendungen ist es jedoch der Klarheit der Struktur und damit der Wartbarkeit und Erweiterbarkeit der enstandenen Anwendung sehr hinderlich. Da zudem der zu implementierende Teil der Frontendkomponente fast nur aus in PHP zu implementierender Frontendlogik (zum Aufbau von Struktur und Darstellung des Frontends sowie zur Darstellung von Daten aus der – und Weitergabe von Daten an die Datenspeicherungskomponente) bestand, wurde das PHP praktisch als reiner Code – mit gelegentlich in Ausgabeanweisungen enthaltenen HTML-Fragmenten – implementiert.

Die Gestaltung des Frontends wurde weitgehend standardisiert und mit einmalig implementierter Standardfunktionalität realisiert.

Dabei wurde, wie allgemein üblich, ein möglichst großer Anteil an wiederkehrend benötigter Funktionalität – soweit es möglich und sinnvoll war – generalisiert und bibliotheksartig als Teil der Infrastruktur der Frontendkomponente allen Modulen zur Verfügung gestellt. Der entsprechende Code findet sich im `include`-Verzeichnis des Frontends von SCASS.

Die Implementation der Frontendkomponenten der einzelnen Module folgt strikten Standardvorgaben:

- Es gibt einen zentralen Funktionsdispatcher, der grundsätzlich als einziger Codeteil direkt im Webfrontend aufgerufen wird und dann je nach ausgewählter Funktionalität in den relevanten Teil des Modulcodes verzweigt, nicht ohne vorher eventuelle Vorbedingungen (z.B. Verwendung von gesicherter Verbindung oder nicht) zu überprüfen. Dieser Dispatcher lädt die von der Frontendkomponente „seines“ Moduls benötigten Codemodule nach, diese umfassen
  - Hilfsmodule aus der Infrastruktur der Frontendkomponente sowie
  - die zur Frontendkomponente des jeweiligen Modules gehörenden Codemodule,
- Es gibt Standardfunktionsgruppen, diese sind:
  - Eintragen neuer Daten,
  - Anzeigen eingetragener Daten,
  - Ändern eingetragener Daten,
  - Löschen eingetragener Daten,
  - weitere Funktionsgruppen nach spezifischer Notwendigkeit,
- die die Standardfunktionsgruppen implementierenden Funktionen benutzen klare, eindeutig funktionsbeschreibende Namen,
- Die Gliederung und Darstellung dieser Funktionsgruppen wird soweit standardisiert wie, nach den Anforderungen des jeweiligen Moduls sinnvoll.

Während die Dateien zur Implementation der Infrastruktur der Frontendkomponete keiner konkreten Namenskonvention unterliegen, gilt für die, die Frontendkomponenten der einzelnen Module implementierenden Dateien, eine strikte Namenskonvention. Diese dient der klaren Strukturierung (und Darstellung dieser Strukturierung) des Codes im Interesse der Wartbarkeit und Erweiterbarkeit der Codebasis. Nach dieser Namenskonvention leitet sich der Name dieser Dateien direkt aus der Modulzugehörigkeit und der implementierten Teilfunktionalität ab: `scass_$MODUL_$FUNTION.php`, wobei `$MODUL` durch den intern verwendeten Modulbezeichner und `$FUNKTION` durch die beschreibende Bezeichnung der implementierten Teilfunktionalität zu ersetzen sind. Zum Beispiel befindet sich der Code zum Anzeigen von Daten aus dem DHCP-Modul in der Datei `scass_dhcp_show.php`.

Aufgrund der Zielgruppe der entwickelten Software (Sysadmins) wird auf ein Abfangen und detailliertes „Übersetzen“ der evtl. von der Datenspeicherungskomponente gelieferten Fehlermeldungen – zum Beispiel bei verletzten Einschränkungen für Datenfelder – verzichtet und es werden direkt die Fehlermeldungen aus der Datenspeicherungskomponente angezeigt.

Sie sind üblicherweise so gestaltet, daß sie das Problem hinreichend deutlich beschreiben, daher betrachtet der Autor dies als unkritisch.

Die Sprachgestaltung des Webfrontends erfolgte vollständig in Englisch, was durchaus den üblichen Gepflogenheiten bei Administrationssoftware entspricht. Lokalisierte (d.h. auf die Landessprache abgestimmte) Versionen dieser Softwareklasse sind oftmals entweder nicht verfügbar oder aufgrund ungeschickter Übersetzungen problematisch. Zudem sollte der Umgang mit englischsprachiger Software für Sysadmins kein Problem darstellen.

Eine Lokalisierung für die Frontendkomponente von SCASS ist jedoch später problemlos nachrüstbar, es wurde im Rahmen dieser Arbeit sowohl aus Zeitgründen als auch aus Mangel an zwingender Notwendigkeit darauf verzichtet.

8.3.2 Infrastruktur der Frontendkomponente

Die Infrastruktur der Frontendkomponente hat im Wesentlichen folgende zwei Aufgaben zu erfüllen:

- Bereitstellung eines Rahmens an Unterstützungsfunktionalität (z.B. Standardfunktionen zur Generierung von Elementen des Webfrontends) für die Frontendkomponenten der einzelnen Module und
- Erzeugung der Basisstruktur des Webfrontsends gemäß den verfügbaren Modulen und ihrer Frontendfunktionalität – d.h. im Wesentlichen die Darstellung der Menüstruktur gemäß den Strukturinformationen aus dem Infrastrukturteil der Datenspeicherungskomponente.

Die modulunabhängige Infrastruktur der Frontendkomponente wird in 4 Dateien implementiert (alle Pfade relativ zum Basisverzeichnis, in dem die Frontendkomponente von SCASS installiert wurde):

- `scass.php`: die Startseite von SCASS, zeigt die Menüstruktur mit der Liste der aktiven Module an,
- `include/sidebar.php`: Includedatei, implementiert die zur Anzeige der Menüstruktur verwendete Funktion `display_sidebar_top()`, wird sowohl von der Startseite von SCASS als auch von den Frontendkomponenten aller Module zu diesem Zweck verwendet,
- `include/nav_sidebar_end.html`: kurzes HTML-Fragment für das Ende der Darstellung Menüstruktur,
- `include/style.html`: enthält das CSS-Stylesheet zur Gestaltung der Darstellung einiger Details der Weboberfläche, wird ebenfalls von den Frontendmodulen aller Komponenten mitgenutzt.

Im in 8.3.1 bereits erwähnten `include`-Verzeichnis befinden sich weitere, die angesprochene modulübergreifende Funktionalität implementierende, Codemodule:

- `db_helpers.php`: verschiedene Datenbankinterfacefunktionen, universelle Standardfunktionen zur Darstellung und Bearbeitung von Daten aus der Datenspeicherungskomponente,

- `form_helper.php`: Standardfunktionen, die mehrfach benötigte Aufgaben zum Umgang mit Webformularen enthalten,
- `textmodules.php`: parametrisierbare Standardtextmodule zum Anzeigen situationsspezifisch angepaßter Standardtexte,
- `helpers.php`: Sammlung sonstiger Hilfsfunktionen, u.a. Funktionen zur Fehlerbehandlung und Debuggingunterstützung für die Entwicklung.

8.3.3 Frontendkomponente DNS-Modul

Die Frontendkomponente des DNS-Moduls implementiert das Webfrontend zum:

- Hinzufügen, Anzeigen, Ändern und Löschen von DNS-Zonen,
- Hinzufügen, Anzeigen, Ändern und Löschen von DNS Hosteinträgen in den eingetragenen Zonen,
- Hinzufügen, Anzeigen, Ändern und Löschen von DNS-Aliaseinträgen in den eingetragenen Zonen, die auf vorhandene DNS-Hosteinträge verweisen,
- Hinzufügen, Anzeigen, Ändern und Löschen von DNS-Sondereinträgen in den eingetragenen Zonen,
- Aktivieren der durchgeführten Änderungen.

Der Code des Frontends verteilt sich dabei auf die Dateien:

- `scass_dns.php`: zentraler Dispatcher, Nachladen der unten aufgeführten Codemodule sowie der Hilfsmodule,
- `scass_dns_activate.php`: Aktivieren der Änderungen,
- `scass_dns_add_zone.php`: Hinzufügen von Einträgen,
- `scass_dns_delete_entries.php`: Löschen von Einträgen,
- `scass_dns_edit_zone.php`: Editieren von Einträgen,
- `scass_dns_select_zone.php`: interne Dispatcherfunktionalität,
- `scass_dns_show_zone.php`: Anzeigen von Einträgen,
- `scass_dns_zoneselect.php`: Auswahl der anzuzeigenden Zone aus den gespeicherten Zonen.

8.3.4 Frontendkomponente DHCP-Modul

Mit der Frontendkomponente für das DHCP-Modul wird das Webfrontend zum:

- Hinzufügen, Anzeigen, Ändern und Löschen von DHCP-Netzen,
- Hinzufügen, Anzeigen, Ändern und Löschen von Hosteinträgen in eingetragenen DHCP-Netzen,

- Hinzufügen, Anzeigen, Ändern und Löschen von IP-Bereichseinträgen in eingetragenen DHCP-Netzen,

implementiert. Dabei verteilt sich der Code auf die Dateien:

- `scass_dhcp.php`: zentraler Dispatcher, Nachladen der unten aufgeführten Codemodule sowie der Hilfsmodule,
- `scass_dhcp_activate.php`: Aktivieren der Änderungen,
- `scass_dhcp_add.php`: Hinzufügen von Einträgen,
- `scass_dhcp_delete.php`: Löschen von Einträgen,
- `scass_dhcp_edit.php`: Editieren von Einträgen,
- `scass_dhcp_show.php`: Anzeigen von Einträgen.

8.3.5 Frontendkomponente Softwarelizenzmanagement-Modul

Da das Softwarelizenzmanagement-Modul konzeptbedingt über kein Backend verfügt, ist die Frontendkomponente der einzige Teil dieses Moduls, mit dem der Anwender in irgendeiner Form in Kontakt kommt. Diese Komponente implementiert die Funktionalität zum:

- Hinzufügen, Anzeigen, Ändern und Löschen von sogenannten Basisdaten:
  - Plattformen,
  - Hersteller,
  - Lizenzen (d.h. die Texte der Lizenzvereinbarungen, z.B. der GPL) und
  - Softwareklassen,
- Hinzufügen, Anzeigen, Ändern und Löschen von Softwareeinträgen,
- Hinzufügen, Anzeigen, Ändern und Löschen von Installationen, die sich auf Softwareeinträge beziehen.

Dabei verteilt sich der Code auf die Dateien:

- `scass_lm.php`: zentraler Dispatcher, Nachladen der unten aufgeführten Codemodule sowie der Hilfsmodule,
- `scass_lm_add.php`: Hinzufügen von Einträgen,
- `scass_lm_delete.php`: Löschen von Einträgen,
- `scass_lm_edit.php`: Editieren von Einträgen,
- `scass_lm_show.php`: Anzeigen von Einträgen.

8.3.6 Implementation der Datenspeicherungskomponente

Aus den in 8.2.4 beschriebenen Anforderungen ergibt sich bereits die Eingrenzung auf ein relationales Datenbanksystem. Die Anforderung an dieses Datenbanksystem, solche wichtigen Fähigkeiten wie

- Trigger,
- stored procedures,
- Constraints,
- References

zu beherrschen schränkt, zusammen mit der Notwendigkeit der freien Verfügbarkeit des ausgewählten relationalen Datenbanksystems, die Auswahl weiter ein. Der Autor hat sich schließlich für PostgreSQL [Gro] entschieden. Diese Entscheidung wurde aus mehreren Gründen getroffen:

- PostgreSQL ist als freie Software unter der BSD-Lizenz [BSD02] verfügbar,
- PostgreSQL implementiert nicht nur alle geforderten Fähigkeiten, sondern ist auch das vollständigste und leistungsfähigste freie relationale Datenbanksystem,
- der Autor hat bereits bei anderen Gelegenheiten sehr gute Erfahrungen mit PostgreSQL als relationalem Datenbanksystem gemacht.

Die Implementation der Datenspeicherungskomponente besteht aus zwei Teilen:

- der Definition der Tabellenstrukturen für SCASS,
- der Implementation der stored procedures.

Die Dateien, in denen dies implementiert wird befinden sich im Unterverzeichnis `SQL` des Backendsystems von SCASS. Dabei handelt es sich um:

- `create.sql`: Anlegen der Datenbank,
- `stored_procedures.sql`: enthält die im Datenbankteil von SCASS verwendeten stored procedures,
- `scass_config.sql`: Definition der Tabellenstruktur für die Konfigurationsdaten von SCASS,
- `scass_config__data.sql`: Konfigurationsdaten von SCASS in Form von SQL INSERT Anweisungen, SCASS-Auslieferungszustand,
- `scass_fes.sql`: Definition der Tabellenstruktur für die von der Frontendkomponente benötigten Konfigurationsdaten (Modul- und Menüstruktur),
- `scass_fes__data.sql`: Modul- und Menüstruktur von SCASS im Auslieferungszustand (mit DNS-, DHCP- und Softwarelizenzmanagement-Modul, alle Module und Menüeinträge aktiv) in Form von SQL INSERT Anweisungen,

- scass_dns.sql: Definition der Tabellenstruktur für das DNS-Modul,
- scass_lm.sql: Definition für das Softwarelizenzmanagement-Modul verwendeten Tabellenstruktur,
- scass_dhcp.sql: Definition der Tabellenstruktur für das DHCP-Modul.

Die weiteren Dateien in diesem Verzeichnis dienen als Hilfsmittel für die Entwicklung und Installation und sind, sofern relevant, dort dokumentiert.

8.3.7 Implementation der Backendkomponente

Die Backendkomponente hat die Aufgabe, aus den in der Datenspeicherungskomponente gespeicherten Daten, Konfigurationsdateien für Dienste zu erzeugen und die Dienste zum Neueinlesen dieser Konfigurationsdateien zu veranlassen. Konfigurationsdateien unter UNIX sind üblicherweise einfache, strukturierte Textdateien und die Manipulation solcher Dateien erfordert vor allem gute Mittel zur Stringbearbeitung. Somit ergeben sich als Anforderungen an das Implementationsmittel der Backendkomponente

- möglichst große Portabilität der Backendkomponente,
- freie Verfügbarkeit,
- gute Datenbankunterstützung,
- gute Unterstützung der Stringverarbeitung.

Damit bietet sich Perl als Implementationssprache für das Backend an. Die Tatsache, daß Perl im Bereich der Systemadministration sehr weit verbreitet ist, unterstützt diese Entscheidung - schließlich ist SCASS erweiterbar ausgelegt und gerade in der Backendkomponente bieten sich Erweiterungen – z.B. in Form weiterer unterstützter Dienstimplementationen – an. Der Einsatz einer weit leistungsfähigen und weit verbreiteten Programmiersprache erleichtert dies natürlich.

Die Backendkomponenten der einzelnen Module (sofern diese über eine Backendkomponente verfügen) sind nach einem Standardmuster modular aufgebaut:

- ein zentrales Skript, welches aufgerufen wird,
- gemeinsam modulübergreifend genutzte Codemodule,
- modullokale, implementationsspezifische Module, welche bestimmte Implementationen der Dienste unterstützen,
- die aktive, zu verwendende Dienstimplementation wird von einer Konfigurationsvariable bestimmt.

Als Grundregel für die Backendkomponente gilt, daß Konfigurationsdateien nicht einfach überschrieben werden dürfen, dies dient zur Vereinfachung der Analyse und Reparatur im Fehlerfall. Existierende Konfigurationsdateien werden (mit einem Zeitstempel im neuen Namen versehen) umbenannt und bleiben somit erhalten. Dies bedeutet aber auch, daß bei häufigem Aktualisieren dieser Dateien auf geeignete Weise die sich ansammelnden Sicherungen der alten Konfigurationsdateien beräumt werden müssen. Dies läßt sich jedoch mit Standardwerkzeugen sehr einfach bewerkstelligen.

8.3.8 Infrastruktur der Backendkomponente

Die Codemodule der Infrastruktur der Backendkomponente sind im Verzeichnis `common` des Backendsystems zu finden. Aufgrund des relativ geringen Umfangs an gemeinsamer Funktionalität ist dies nur relativ wenig Code:

- `db_connect.pl`: Verbindungsaufbau mit der Datenbank,
- `load_config.pl`: Laden der Konfiguration (für den Datenbankzugang) aus der Konfigurationsdatei,
- `load_config_db.pl`: Laden der Systemkonfiguration aus der Datenbank,
- `timestamp.pm`: Erzeugung verschiedener Zeitstempel.

8.3.9 Backendkomponente DNS-Modul

Die Backendkomponente des DNS-Modul wird durch das Perlskript `DNS/scass2zone` und die zugehörigen Codemodule im Verzeichnis `DNS/modules` implementiert.

Das Skript lädt – je nach Konfiguration – das korrekte Codemodul zur Unterstützung des eingestellten DNS-Servers, erzeugt die Konfigurationsdateien des DNS-Servers und veranlaßt den DNS-Server zum Einlesen dieser Konfigurationsdateien.

Derzeit ist – aus Gründen der im Rahmen dieser Arbeit sehr begrenzten Zeit – nur eine Unterstützung für BIND9 implementiert. Dieses Codemodul erzeugt sowohl die Zonefiles (forward- und reverse-Files) als auch ein Includefile für die `named.conf`. Die Integration des DNS-Moduls in eine bestehende BIND9-Konfiguration besteht damit aus dem Eintragen einer einzigen Zeile in die `named.conf` sowie der Konfiguration des DNS-Moduls selbst.

Da das implementationsspezifische Codemodul nur

- das Erzeugen der Konfigurationsdateien sowie
- das Veranlassen des Dienstes zum Neuladen

implementieren muß, läßt sich Support für weitere Nameserver (unter Verwendung der BIND9-Unterstützung als Vorlage) relativ einfach hinzufügen.

Die anderen Aufgaben der Backendkomponente des DNS-Moduls,

- das Aktualisieren der Serial der DNS-Zone,
- des Zurücksetzen des `do_commit`-Flags der DNS-Zone sowie
- das Bestimmen der zu bearbeitenden DNS-Zone(n)

werden als Standardfunktionalität in `scass2zone` implementiert.

8.3.10 Backendkomponente DHCP-Modul

Die Backendkomponente des DHCP-Modul wird durch das Perlskript `DHCP/scass2dhcpconf` und die zugehörigen Codemodule im Verzeichnis `DHCP/modules` implementiert.

Das Skript lädt – je nach Konfiguration – das korrekte Codemodul zur Unterstützung des eingestellten DHCP-Servers, erzeugt die Konfigurationsdateien des DHCP-Servers und veranlaßt den DHCP-Server zum Einlesen dieser Konfigurationsdateien.

Derzeit ist nur eine Unterstützung für den ISC DHCP Server implementiert. Dieses Codemodul erzeugt die `dhcpd.conf` für den DHCP-Server.

Da das implementationsspezifische Codemodul nur

- das Erzeugen der Konfigurationsdateien sowie
- das Veranlassen des Dienstes zum Neuladen

implementieren muß, läßt sich Support für weitere DHCP-Server – unter Verwendung der Unterstützung des ISC DHCP Servers als Vorlage – relativ einfach hinzufügen. Allerdings muß erwähnt werden daß, im Gegensatz zum DNS-Modul, hier eine Erweiterung deutlich weniger wahrscheinlich ist, da der ISC DHCP Server praktisch als Standard gilt und kaum andere freie DHCP-Server verfügbar sind.

Die anderen Aufgaben der Backendkomponente des DHCP-Moduls,

- das Zurücksetzen des `do_commit`-Flags des DHCP-Netes
- das Bestimmen der zu bearbeitenden DHCP-Netze

werden als Standardfunktionalität in `scass2dhcpconf` implementiert.

8.4 Installation

Aus implementationstechnischen Gründen verteilt sich das Gesamtsystem von SCASS auf zwei Subsysteme, die in Form zweier Archive gemeinsam ausgeliefert werden:

- das SCASS-Frontendsystem im Archiv `scass.fes-$VERSION.tar.gz` enthält die Frontendkomponente und
- das SCASS-Backendsystem im Archiv `scass.bes-$VERSION.tar.gz` enthält die Daten zum Aufbau der Datenspeicherungskomponente und die Backendkomponente,

dabei ist `$VERSION` die Versionsnummer (z.B. `1.0`) des jeweiligen Subsystems.

8.4.1 Installationsvorraussetzungen für SCASS

Für die Installation von SCASS müssen auf dem Zielsystem natürlich verschiedene Vorraussetzungen erfüllt sein. Diese umfassen sowohl installierte Softwareversionen als auch laufende Dienste. An installierter Software werden zum Betrieb von SCASS benötigt:

- eine PostgreSQL-Datenbank, Version 7.2 oder aktueller, mit Unterstützung für stored procedures in plpgsql und plperl,
- ein PHP4-fähiger Webserver mit SSL-Unterstützung, bevorzugt Apache mit mod_php in PHP4-Unterstützung,
- Perl in einer Version ≥ 5 (entwickelt wurde mit Perl 5.6),
- das Perl-Datenbankinterface DBI mit Unterstützung für PostgreSQL,

8.4.2 Installation von SCASS

Zuerst muß das Backendsystem installiert werden, da das Frontendsystem zur Läuffähigkeit die Datenspeicherungskomponente vorraussetzt. Dazu muß zuerst vom installierenden Sysadmin ein geeigneter Ort im Filesystem gefunden werden, der Autor empfiehlt als Verzeichnis `/opt/scass.bes`. Dazu wird das Archiv `scass.bes-$VERSION.tar.gz` im übergeordneten Verzeichnis (z.B. `/opt` bei Installation nach `/opt/scass.bes`) entpackt und dann ein symbolischer Link von `/opt/scass.bes` auf das entstandene Verzeichnis `/opt/scass.bes-$VERSION` angelegt.

Als erster Schritt nach dem Auspacken des Archivs muß die Datenbank aufgesetzt werden. Dies erfolgt in mehreren Schritten:

- Anlegen der notwendigen Datenbanknutzer im Datenbanksystem,
- Erzeugen des SQL-Skripts zum Anlegen der Datenbankstruktur,
- Anlegen der Datenbank,
- Anlegen der Datenbankstruktur mit Hilfe des vorher genannten SQL-Skripts.

Es wird davon ausgegangen, daß die Einrichtung der Datenbank unter der User-Id des PostgreSQL-Systemnutzers auf der Kommandozeile erfolgt. Dies ist notwendig, da Datenbanken erzeugt und modifiziert sowie Datenbanknutzer angelegt werden müssen. Dann sind folgende Kommandos auszuführen:

- Anlegen der Datenbank scass:
  - `createdb scass`.
- Aktivieren der Unterstützung für stored procedures in plpgsql und plperl (falls nicht bereits auf dem Datenbanktemplate `template1` aktiviert):
  - `createlang --pglib=$POSTGRESLIB plperl scass`
  - `createlang --pglib=$POSTGRESLIB plpgsql scass`

  dabei ist `$POSTGRESLIB` das Libraryverzeichnis der lokalen PostgreSQL-Installation, üblicherweise unter `/usr/postgresql/lib` oder an ähnlicher Stelle zu finden.
- Anlegen der Datenbanknutzer für SCASS. Es werden für den Einsatz von SCASS mehrere Datenbanknutzer benötigt:
  - der Datenbankadmin `DBADMIN`, dieser benötigt volle Rechte auf der Datenbank und wird benötigt, um die Datenbankstruktur von SCASS zu erzeugen sowie um SCASS über die Datenbank zu konfigurieren,
  - der Dienstnutzer für das Frontendsystem, dieser benötigt SELECT-Rechte auf allen Tabellen der Datenbank und wird von der Frontendkomponente zum Anzeigen von Daten ohne Passwortabfrage benötigt,
  - der Dienstnutzer für das Backendsystem, dieser benötigt SELECT-Rechte auf allen Tabellen sowie UPDATE-Rechte auf bestimmten Tabellen und wird von der Backendkomponente zum Erzeugen der Konfigurationsdateien aus der Datenbank benötigt,
  - die Nutzer von SCASS (die Sysadmins), die schließlich mit dem System arbeiten sollen, sie benötigen sehr weitgehende Rechte auf der Datenbank.

  Sämtliche Nutzer sind mit Passwörtern zu versehen, die Passwörter für die Sysadmins und den Datenbankadmin sind diesen Personen mitzuteilen, die Passwörter für das Frontend- und das Backendsystem sind zu notieren – sie werden später gebraucht. Ebenso sind die angelegten Nutzernamen (und ihre zugedachten Rollen) zu notieren. Die Datenbanknutzer werden mit dem Kommando:
  - `createuser --no-createdb --no-adduser --pwprompt $USERNAME`

  angelegt, dabei ist `$USERNAME` der anzulegende Datenbanknutzer.
- Erzeugen der Datenbankstruktur für SCASS. Dies erfolgt in mehreren Schritten:
  - Wechsel in das Installationsverzeichnis des Backendsystems,
  - Wechsel in das Unterverzeichnis SQL,
  - Aufruf des Kommandos:
    * `make`

    Jetzt wird die Liste der angelegten Datenbanknutzer und ihrer Funktionen, auf die vorher hingewiesen wurde, benötigt. Es werden nun von einem Skript die Datenbanknutzer für die vier Nutzerklassen (BES == Backendsystem, FES == Frontendsystem, USER == Sysadmins, DBADMIN = Datenbankadmin für SCASS) abgefragt, diese sind durch Komma getrennt einzugeben. Anschließend erzeugt der make-Lauf ein SQL-Skript zum Anlegen der Datenbankstruktur von SCASS

– Anlegen der Datenbankstruktur. Dazu muß das erzeugte SQL-Skript (mit dem Dateinamen `scass.sql` in die Datenbank geladen werden. Am einfachsten kann dies erfolgen, indem sich der installierende Sysadmin unter der Datenbanknutzerkennung des Datenbankadmins für SCASS an der Datenbank mit dem Kommandozeilenwerkzeug `psql` anmeldet:

  * `psql -h $DBHOST scass $DBADMIN`

  Dabei ist für den Platzhalter `$DBHOST` der Name des verwendeten Datenbankservers und für `$DBADMIN` der als Datenbankadmin für SCASS vorgesehene Datenbanknutzer zu verwenden. Mit dem Befehl

  * `\i scass.sql`

  wird das soeben erstellte SQL-Skript `scass.sql` geladen und damit die Datenbankstruktur von SCASS angelegt. Der Befehl

  * `\q`

  beendet das `psql` Kommandozeilenwerkzeug.

Damit ist die Datenbankstruktur für SCASS angelegt. Das Konfigurieren von SCASS über die Datenbank schließt die Einrichtung der Datenbank ab. Die Konfiguration von SCASS ist in der Tabelle `scass_config` abgelegt. Um die Standardkonfiguration zu ändern muß sich der Datenbankadmin für SCASS wie soeben beschrieben mittels `psql` an der Datenbank anmelden. Mit UPDATE-Anweisungen in SQL können dann die Werte geändert werden. Die Anweisungen haben dabei in diesem Fall eine einfache Syntax:

– `UPDATE scass_config SET value = '$VALUE' where key = '$KEY';`

Dabei sind für `$KEY` der Name des zu ändernden Konfigurationseintrages und für `$VALUE` der neue Wert dieses Eintrages zu verwenden. Mit dem Kommando

– `SELECT * from scass_config;`

können alle Konfigurationswerte angezeigt werden. Sämtliche Konfigurationswerte enthalten im `comment`-Feld eine kurze Beschreibung ihrer Bedeutung. Die Beschreibungen sind dabei hinreichend deutlich, so daß an dieser Stelle nicht weiter darauf eingegangen werden muß.

Die Datenbankkonfiguration von SCASS ist damit abgeschlossen. Im nächsten Schritt müssen die Konfigurationsdateien eingerichtet werden. Dazu sind die im Installationsverzeichnis das Backendsystems liegenden Vorlagen

- `scass-bes.conf.example` für das Backendsystem und
- `scass-fes.conf.example` für das Frontendsystem

an die richtigen Stellen zu kopieren. Standardmäßig werden die Konfigurationsdateien im Verzeichnis `/etc` gesucht. Damit ist

- `scass-bes.conf.example` nach `/etc/scass-bes.conf` und

- `scass-fes.conf.example` nach `/etc/scass-fes.conf`

zu kopieren. Anschließend müssen die Inhalte angepasst werden. Die Vorlagen sind dafür hinreichend selbstbeschreibend.

Als letzter Schritt ist die Frontendkomponente zu installieren. Dazu muß in einem geeigneten Unterverzeichnis des Webservers das Archiv `scass.fes-$VERSION.tar.gz` ausgepackt werden. Die Startseite der Frontendkomponente von SCASS bildet die Datei `scass.php`.

Für die Einbindung des DNS-Moduls in eine vorhandene BIND9-Installation ist in der Datei `named.conf` dieser Installation die Zeile „`include "$INCLUDE";`" hinzuzufügen. Dabei ist `$INCLUDE` durch den Konfigurationswert `NamedIncludeFile` zu ersetzen.

Das DHCP-Modul legt die erzeugten DHCP-Konfigurationsdateien in dem unter dem Konfigurationswert `DHCPdConffileDir` eingetragenen Verzeichnis ab. Der Dateiname wird dabei aus dem Namen des DHCP-Netzes abgeleitet: `$NETNAME.conf`, dabei ist `$NETNAME` der Name des in der Konfigurationsdatei beschriebenen DHCP-Netzes.

Damit ist die Installation von SCASS abgeschlossen.

9 Fazit und Ausblick

9.1 Bewertung

Mit der vorliegenden Arbeit wurden verschiedene Aspekte der Unterstützung der Sysadmintätigkeit durch Softwarewerkzeuge untersucht, ein Überblick über vorhandene Lösungen gegeben und ein eigenes System zur Unterstützung des Sysadmins entwickelt, welches die Administration und Verwaltung von DNS-Servern, DHCP-Servern und Softwarelizenzen vereinfacht. Der Aspekt der Nutzerverwaltung wurde untersucht, wobei festgestellt wurde, daß sich für diesen Bereich eine potentiell sehr leistungsfähig Lösung in Entwicklung befindet. Daraufhin entschied sich der Autor, auf diese Entwicklung zu verweisen und nach weiterer Betrachtung des Problemkomplexes die Entwicklung einer eigenen Lösung aufgrund zu hoher und damit den Rahmen dieser Arbeit sprengenden Komplexität zurückzustellen.

Es bietet sich an, nach der Veröffentlichung von [Pet02] die dort erzielten Resultate zu analysieren und ggf. auf dieser Grundlage (sofern dies noch nicht in [Pet02] enthalten ist) ein geeignetes Frontendwerkzeug zur Verwaltung der entsprechenden Daten in Kerberos und LDAP zu erstellen.

Im Rahmen der Untersuchungen zu dieser Arbeit stellten sich zwei weitere Bereiche als besonders wichtig für die Unterstützung der Systemadministration heraus: die Überwachung der administrierten Systeme (Netzwerke und Hosts) und die Verwaltung des Sysadminwissens. Aufgrund des engen Bezugs dieser Bereiche zur bearbeiteten Problematik entschloß sich der Autor, sie in den Umfang der Arbeit aufzunehmen.

9.2 Weiterentwicklung von SCASS

Der derzeitige Entwicklungsstand des im Rahmen dieser Arbeit entstandenen Administrationsunterstützungswerkzeugs – SCASS – erfüllt die gestellten Aufgaben und Anforderungen. Es wurden Konfigurationsmodule für DNS und DHCP sowie ein Modul zum Softwarelizenzmanagement aus der Sicht des Sysadmins implementiert. Die Benutzbarkeit des Frontends über eine breite Palette an Webbrowsern hinweg, darunter insbesondere Textmodusbrowsern wurde erreicht und erfolgreich getestet.

Während die Backendkomponente und die Datenspeicherungskomponente ihren Aufgaben auch bei erheblichem Wachstum der verwalteten Daten gerecht werden können, ergeben sich bei der Frontendkomponente Skalierungsprobleme. Dabei handelt es sich nicht um Probleme im Bereiche der Performance, sondern um Probleme im Bereich der Benutzbarkeit. Mit mehreren hundert Einträgen pro Modul verliert das derzeitige Webinterface der Frontendkomponente erheblich an Übersichtlichkeit. Eine weitere Verbesserungsmöglichkeit wäre eine Sortierbarkeit der angezeigten Datensätze nach einem beliebigem Feld – derzeit wird nach dem ersten Datenfeld sortiert.

Das DNS-Modul ist auf die Bedürfnisse eines kleineren bis mittleren Netzes mit einfacher DNS-Struktur orientiert. Mögliche Weiterentwicklungen dieser Komponenten wären u.a.:

- direkte Unterstützung eines hierarchisch gegliederten lokalen DNS-Namensraumes,
- differenzierte Rechtevergabe für die Administration von Teilbereichen des DNS-Namensraumes,
- erweiterte Konfigurationsmöglichkeiten, z.B. Unterstützung weiterer Typen von DNS-Records.

Im Softwarelizenzmanagement wird in der aktuellen Implementation von einem Lizenzmodell mit abzählbaren Lizenzen ausgegangen. Daher bietet sich die Erweiterung auf Unterstützung weiterer Lizenzmodelle, z.B. site licenses, an.

Ebenfalls interessant wäre die Möglichkeit, direkt die aktuelle Softwarekonfiguration eines Hostes von diesem abzufragen (Windows 2000 bietet diesbezüglich gewisse rudimentäre Möglichkeiten via SNMP), mit dem Stand in der Datenbank zu vergleichen und die Abweichungen zum Aktualisieren der Datenbank anzubieten.

Aufgrund der Aufgabenstellung unterstützt das DHCP-Modul mit Hinblick auf kleinere Netze nur grundlegende Informationen. Dies könnte auf eine wesentlich größere Teilmenge der von DHCP unterstützten Konfigurationsdaten erweitert werden. Eine sehr nützliche Erweiterung wäre ein Zugriff auf den aktuellen DHCP-Status um zu ermitteln, welche Hosts gerade welche Konfiguration via DHCP erhalten haben. So könnte, regelmäsig bzw. auf Abruf, der aktuelle DHCP-Status aus den Statusdateien des DHCP-Servers ausgelesen und diese Information geeignete Weise in das DHCP-Modul eingebunden werden.

Das durch SCASS gebildete System soll natürlich auch erweitert werden. In Frage kommen als zusätzliche Module zum Beispiel:

- Apache-Modul: Verwaltung der Konfiguration – sowohl insgesamt als virtuelle Hosts – von Apache Webservern,

- Konfiguration von Netzwerkgeräten,

Aufgrund des auf Modularität und einfache Erweiterbarkeit hin ausgerichteten Designs von SCASS beschränkt sich der Aufwand für zusätzliche Module auf die Implementation der Komponenten des jeweiligen Moduls.

Literatur

[AB02] MySQL AB. *MySQL*, 2002.
URL http://www.mysql.com/

[Aeb02] Thomas Aeby. *The Big Sister Network and System Monitor*, 2002.
URL http://bigsister.graeff.com/

[Air00] Dave Airlie. *pam_smb*, 2000.
URL http://pamsmb.sourceforge.net/

[All02] Jeff R. Allen. *Cricket*, 2002.
URL http://cricket.sourceforge.net/

[Ber02a] D. J. Bernstein. *djbdns: Domain name tools*, 2002.
URL http://cr.yp.to/djbdns.html

[Ber02b] Ian Berry. *cacti*, 2002.
URL http://www.raxnet.net/products/cacti/

[Bes02] Best Practical Solutions, LLC. *Request Tracker*, 2002.
URL http://www.bestpractical.com/rt/

[Bos99] Håkon Wium Lie, Bert Bos. *Cascading Style Sheets, level 1*, Januar 1999.
URL http://www.w3.org/TR/REC-CSS1

[BSD02] *The BSD license*, 2002.
URL http://www.opensource.org/licenses/bsd-license.php

[Bur02] Mark Burgess. *Cfengine - A configuration engine*, 2002.
URL http://www.cfengine.org/

[CSN02] Chemnitzer Studentennetz CSN. *CSN - Chemnitzer StudentenNetz*, 2002.
URL http://www.csn.tu-chemnitz.de/

[Cun02] Ward Cunningham. *Wiki Wiki Web*, August 2002.
URL http://c2.com/cgi/wiki?WikiWikiWeb

[Div02] Globetrotter Software Division. *FLEXlm*, 2002.
URL http://www.globetrotter.com/flexlm/flexlm.shtml

[Dro97a] R. Droms. *RFC2131 - Dynamic Host Configuration Protocol*, März 1997.
URL http://www.ietf.org/rfc/rfc2131.txt

[Dro97b] S. Alexander, R. Droms. *RFC2132 - DHCP Options and BOOTP Vendor Extensions*, März 1997.
URL http://www.ietf.org/rfc/rfc2132.txt

[Eng02] Ralf S. Engelschall. *mod_ssl - The Apache Interface to OpenSSL*, 2002.
URL http://www.modssl.org/

[Fin93] Jon Finke. *Relational Database + Automated SysAdmin = SIMON*, Juli 1993.
URL http://www.rpi.edu/~finkej/FTPPapers/simon-sug-east93.ps

[Fin02a] Jon Finke. *An Improved Approach for Generating Configuration Files from a Database*. Seiten 29–38. New Orleans, LA, USA, Dezember 2002.

[Fin02b] Jon Finke. *An Improved Approach for Generating Configuration Files from a Database*, 2002.
URL http://www.usenix.org/publications/library/proceedings/lisa2000/finke.html

[Fou02] The Apache Software Foundation. *Apache HTTP Server Project*, 2002.
URL http://httpd.apache.org/

[Fre01] Free Software Foundation, Inc. *GNU General Public License*, Juli 2001.
URL http://www.gnu.org/licenses/gpl.html

[Gal01] Ethan Galstad. *NetSaint Network Monitor*, 2001.
URL http://www.netsaint.org/

[Gal02] Ethan Galstad. *Nagios^{TM}*, 2002.
URL http://www.nagios.org/

[Gre02] Anthony E. Greene. *PDC Authentication HOWTO*, 2002.
URL http://www.mindspring.com/~aegreene/linux/PDC-Authentication-HOWTO.html

[Gro] PostgreSQL Global Development Group. *PostgreSQL*.
URL http://www.postgresql.org/

[Gro02] The PHP Group. *PHP: Hypertext Preprocessor*, 2002.
URL http://www.php.net/

[ISC02a] Internet Software Consortium ISC. *ISC BIND*, 2002.
URL http://www.isc.org/products/BIND/

[ISC02b] Internet Software Consortium ISC. *ISC DHCP*, 2002.
URL http://www.isc.org/products/DHCP/

[Jac98] Bert Bos, Håkon Wium Lie, Chris Lilley, Ian Jacobs. *Cascading Style Sheets, level 2*, Mai 1998.
URL http://www.w3.org/TR/REC-CSS2/

[Kar96] Alan O. Freier, Paul C. Kocher, Philip L. Karlton. *SSL 3.0 SPECIFICATION*, November 1996.
URL http://wp.netscape.com/eng/ssl3/

[Lau02] Ben Laurie. *Apache-SSL*, 2002.
URL http://www.apache-ssl.org/

[Mic02] Microsoft Corporation. *Microsoft Knowledgebase*, 2002.
URL http://support.microsoft.com/

[Moc87a] P. Mockapetris. *RFC1034 - DOMAIN NAMES - CONCEPTS AND FACILITIES*, November 1987.
URL http://www.ietf.org/rfc/rfc1034.txt

[Moc87b] P. Mockapetris. *RFC1035 - DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION*, November 1987.
URL http://www.ietf.org/rfc/rfc1035.txt

[Oet02] Tobi Oetiker. *RRDtool*, 2002.
URL http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/

[(OS02] The Open Source Development Network (OSDN). *freshmeat.net*, 2002.
URL http://freshmeat.net/

[Ost02] Robert Osterlund. *PIKT^{TM}*, 2002.
URL http://pikt.org/

[Pau01] Paul Albitz, Cricket Liu. *DNS and BIND*. O'Reilly & Associates, Inc, 4. Auflage, April 2001. ISBN 0-596-00158-4.

[Pet00] Karsten Petersen. *Monitoring heterogener Systeme - Was kommt nach BigBrother?*, April 2000.
URL http://archiv.tu-chemnitz.de/pub/2000/0034/index.html

[Pet02] Karsten Petersen. *unveröffentlichte Studienarbeit*. TU Chemnitz, vorraussichtlich Herbst 2002.

[Ram02] Russ Dill , Matthew Ramsay. *udhcp Server/Client Package*, 2002.
URL http://udhcp.busybox.net/

[Sco97] Doug Scoular. *UNIX Integration GINA replacement v1.01*, 1997.
URL http://www.arch.usyd.edu.au/~doug/gina.html

[Spa02] Michael Sparks. *TWiki Success Story of Inktomi*, Juni 2002.
URL http://twiki.org/cgi-bin/view/Main/TWikiSuccessStoryOfInktomi

[Tea02] The Samba Team. *Samba*, 2002.
URL http://www.samba.org/

[Tec02] BB4 Technologies. *Big Brother*, 2002.
URL http://bb4.com/

[Tho00] Peter Thoeny. *TWiki Success Story at TakeFive Software*, Mai 2000.
URL http://twiki.org/cgi-bin/view/Main/TWikiSuccessStoryOfTakeFive

[Tho02a] Peter Thoeny. *TWiki Success Story of Wind River*, Juli 2002.
URL http://twiki.org/cgi-bin/view/Main/TWikiSuccessStoryOfWindRiver

[Tho02b] Peter Thoeny. *TWiki^{TM}- A Web Based Collaboration Platform*, August 2002.
URL http://www.twiki.org/

[Tom01] Tom Limoncelli, Christine Hogan. *The Practice of System and Network Administration*. Addison-Wesley Pub Co, 2001. ISBN 0-201-70271-1.

[Tre02] Sam Trenholme. *MaraDNS*, 2002.
URL http://www.maradns.org/

[uni02] *unixops*, 2002.
URL http://unixops.no-ip.org

[Wil97] Nigel Williams. *NISGINA*, 1997.
URL http://www.dcs.qmul.ac.uk/~williams/

10 Erklärungen

Haftungsausschluß

Der Autor übernimmt keine Haftung für Schäden, die sich aus der Benutzung dieser Arbeit oder der zugehörigen Software ergeben.

Schutzrechte

Eingetragene Waren- und Markenzeichen sind im Text nur in Ausnahmefällen als solche gekennzeichnet. Das Fehlen der Kennzeichnung bedeutet nicht, daß diese Zeichen frei verwendbar sind.

Mir sind zum Zeitpunkt der Abgabe keine Ansprüche Dritter an dieser Arbeit und der zugehörigen Software bekannt.

Lizenzerteilung

Die zu dieser Arbeit gehörende Software wird nach den Bedingungen der GNU GPL Version 2 als freie Software lizensiert.

Selbständigkeitserklärung

Hiermit versichere ich, daß ich die vorliegende Arbeit selbständig und ausschließlich mit Hilfe der aufgeführten Quellen angefertigt habe.

Software auf CD