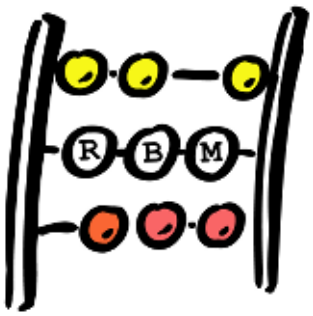

Universität Paderborn
Fakultät für Elektrotechnik, Informatik und Mathematik (EIM)
Institut für Mathematik

Studienarbeit

Konzeption und Implementation eines Wake-Up-Managers

vorgelegt von
Maximilian Wilhelm
max@rfc2324.org
Dr. Rörig-Damm 146
33102 Paderborn

vorgelegt bei
Dr. Dietmar Guhe
und
Prof. Dr. Gerd Szwillus

Paderborn, im September 2008

Für Franz, der so früh von uns gehen musste.

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen (einschließlich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind ausnahmslos als solche kenntlich gemacht.

Paderborn, 1. September 2008

Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Motivation und Einleitung

Das Institut für Mathematik der Universität Paderborn betreibt ungefähr 110 Arbeitsplatz- und 55 Poolrechner, die kontinuierlich laufen. Jeder Mitarbeiter, der einen Rechner mit dem Betriebsystem Linux besitzt, kann sich rund um die Uhr lokal oder via SSH[1] über das Netzwerk an seinem Rechner anmelden, Berechnungen starten, Ergebnisse abrufen oder auf — möglicherweise lokal[2] — abgelegte Daten zugreifen. Der Zugriff auf Windowsrechner über das Netzwerk ist nicht möglich.
Sämtliche Poolräume des Institutes stehen Studierenden ebenfalls durchgehend zur Verfügung. Eine lokale Anmeldung ist überall möglich, der Zugriff über das Netzwerk jedoch nur auf die Rechner des Linuxpools.

Die Regelung, das sämtliche Rechner kontinuierlich laufen, ist historisch gewachsen und an der Universität verbreitet. Neben reiner Bequemlichkeit ist sie mitunter darin begründet, dass sie auch für die Adminstration der Client-Rechner vorteilhaft ist, da Systemverwalter ebenfalls rund um die Uhr auf die Rechner zugreifen können und somit die Möglichkeit besitzen — ggf. automatisierte — Wartungsarbeiten vorzunehmen.

Der offensichtliche Nachteil dieser Regelung ist, dass der Großteil der PCs außerhalb der Arbeitszeiten der Mitarbeiter bzw. Hauptnutzungszeiten der Poolräume ungenutzt bleibt und somit unnötigerweise läuft. Dies verbraucht eine nicht unwesentliche Menge an Energie

[1] vgl. http://www.openssh.org/de/features.html

[2] Aus Performance- oder Platzgründen werden Berechnungserbenisse bzw. private Daten meist auf der lokalen Festplatte eines Rechners abgelegt.

und ist nicht im Sinne des Klimaschutzes, von Wirtschaftlichkeit und "Green IT" (vgl. [Gre08], [Sta07], [Bis08]).

Bei 220 Arbeitstagen im Jahr mit jeweils 8 Stunden Arbeit pro Tag (d_A) ergibt sich das im folgenden berechnete Einsparpotential. Die Arbeitszeit wird auch als Nutzungszeitraum für die Poolräume des Institutes herangezogen.

Ein volles Jahr entspricht 365 Tagen

$$t_J = 1\,\mathrm{a} = 365\,\mathrm{d} * 24\frac{\mathrm{h}}{\mathrm{d}} = 8760\,\mathrm{h} \tag{1.1}$$

Berechnet man nun die jährliche Arbeitszeit (t_A) in Stunden

$$t_A = 220\,\mathrm{d_A} * 8\frac{\mathrm{h}}{\mathrm{d_A}} = 1760\,\mathrm{h} \tag{1.2}$$

und legt man einen dauerhaft konstanten Energieverbrauch zugrunde, so ergibt sich durch Abschalten der Rechner außerhalb der Arbeitszeten ein Einsparungspotential an Energiekosten von (bis zu) $\approx 80\%$.

Dies ist jedoch nur ein theoretischer Wert, da der Energieverbrauch eines PCs nicht konstant ist, sondern je nach Last des Rechners schwankt. Da neuere Rechner, die nach der ATX-Norm (*Advanced Technology Extended*) aufgebaut sind, keinen Netzschalter mehr im physikalischen Sinne aufweisen, sondern das Mainboard des Rechners selbst im abgeschalteten Zustand mit Energie versorgt wird, relativiert sich die obige Rechnung weiter. Die kontinuierliche Spannungsversorgung des PCs ist notwendig, damit er bei Druck des Netzschalters — im physikalischen Sinne ein Taster — oder einem anderen Ereignis, wie z.B. einem Signal über ein angeschlossenes Netzwerk oder Modem, gestartet werden kann.

Um die möglichen Einsparungen realistischer einschätzen zu können, wurde der Verbrauch mehrerer PC-Systeme gemessen, die im Rechnernetz der Mathematik im Einsatz sind. Im einzelnen sind dies Rechner der Firma Dell der Marke Optiplex aus den Serien GX260,

GX280 und GX620. Die Leistungsaufnahme dieser Rechner unterscheidet sich sowohl im ausgeschalteten Zustand als auch im Leerlaufbetrieb stark. Die gemessenen Werte im Leerlauf- (P_L) bzw. Standbybetrieb (P_S) sind in Tabelle 1.1 aufgeführt.

PC-Serie	P_S	P_L
Optiplex GX260	6 W	73 W
Optiplex GX280	13 W	86 W
Optiplex GX620	13 W	86 W

Tabelle 1.1: Leistungsaufnahme verschiedener Rechnerserien im Überblick

Aus (1.1) und (1.2) geht hervor, dass sich ein Rechner innerhalb eines Jahres bis zu 7000 Stunden im Leerlauf befindet ($t_J - t_A$). Es wäre wünschenswert, den Rechner ausserhalb der Arbeitszeiten auszuschalten, sodass er sich während dieser Zeit im Standbymodus befindet. Die erforderliche Leerlauf- und Standbyenergie (E_L bzw. E_S) in einem Jahr kann mittels der folgenden Formeln berechnet werden.

$$E_L = (t_J - t_A) * P_L \tag{1.3}$$

$$E_S = (t_J - t_A) * P_S \tag{1.4}$$

Berechnet man den Energieverbrauch der in Tabelle 1.1 aufgeführten PCs pro Rechner pro Jahr, so ergeben sich die in Tabelle 1.2 aufgeführten Werte.

PC-Serie	E_S	E_L
Optiplex GX260	42 kWh	551 kWh
Optiplex GX280/GX220	91 kWh	602 kWh

Tabelle 1.2: Energieverbrauch verschiedener Rechnerserien pro Jahr

Ausgehend von den minimalsten gemessenen Verbrauchswerten (Optiplex GX260), ergibt sich hochgerechnet auf alle Mitarbeiter- und Poolrechner — insgesamt 165 — der Mathematik eine Summe von (bis zu)

$$(551\,\mathrm{kWh} - 42\,\mathrm{kWh}) * 165 = 83.985\,\mathrm{kWh}$$

pro Jahr, die allein am Institut für Mathematik eingespart werden könnte. Andere Bereiche der Universität, etwa das IMT oder das Institut für Informatik, verfügen ebenfalls über eine große Anzahl von PCs, die teilweise ungenutzt laufen.

Zur Berechnung eines Ersparnisfaktors ϵ_W ist die Kenntnis des Energieverbrauchs während der Arbeitszeit eines Jahres (E_A) erforderlich. Dieser Wert ist jedoch nicht bekannt. Als Grundlage einer Schätzung wird das arithmetische Mittel aller gemessenen Leistungswerte im Leerlaufbetrieb herangezogen und um 15 W erhöht (P_A). Dies dürfte einen realistischen Mittelwert über alle Arbeitsplatz- und Poolrechner darstellen, da aufgrund mehrjähriger Erfahrung als Administrator am Institut für Mathematik überwiegend ein Gleichgewicht zwischen ungenutzten PCs — meist Poolrechnern — und durch Rechnungen ausgelasteten Arbeitsplatzrechnern besteht.

Es ergibt sich ein jährlicher Energieverbrauch pro Rechner

$$\begin{aligned} E_A &= t_A * P_A \\ &= 1760\,\mathrm{h} * (81 + 15)\,\mathrm{W} \\ &\approx 169\,\mathrm{kWh} \end{aligned}$$

Aus (1.3) und (1.4) ergibt sich die folgende Formel zu Berechnung des möglichen Energieersparnisfaktors ϵ_W:

$$\epsilon_W = \frac{(E_A + E_L) - (E_A + E_S)}{E_A + E_L} = \frac{E_L - E_S}{E_A + E_L} \tag{1.5}$$

Es ergibt sich ein Energieeinsparpotential von (bis zu) 66% - 71%, je nach Rechnerserie. Der Wert könnte exakter berechnet werden, wenn Informationen über die Verbrauchswerte und Anzahl der PCs aller aufgestellter Serien verfügbar wären. Diese Informationen könnten jedoch nur durch Besuchen alle Pool- und Büroräume ermittelt werden. Eine absolut exakte Berechnung wäre jedoch selbst bei Kenntnis dieser Zahlen nicht möglich, da aufgrund von üblichen Verschleißerscheinungen Komponenten der Rechner, insbesondere Netzteile, getauscht wurden und somit nur eine Messung jedes einzelnen Gerätes ein exaktes Ergebnis garantieren könnte.

Aus dieser einfachen Rechnung entsteht der Wunsch, Rechner bedarfsorientiert starten und herunterfahren zu können, um einerseits weiterhin flexibel arbeiten zu können, andererseits aber auch einen schonenden Umgang mit Ressourcen zu gewährleisten.

Diese Arbeit stellt ein System vor, das nach den Vorgaben des Institutes für diesen Zweck entwickelt wurde und eine Lösung bietet, die sich optimal in die lokale heterogene IT-Landschaft integriert.

Mit diesem System — dem WakeUpManager — ist es möglich, den rechnerinduzierten Energieverbrauch des Institutes für Mathematik deutlich zu verringern und somit Kosten zu sparen, ohne zu sehr in den Arbeitsalltag der Studierenden und Mitarbeiter eingreifen zu müssen.

Die Anforderungen an dieses System werden in Kapitel 2 vorgestellt. Kapitel 3 beschreibt die Architektur des WakeUpManagers, gefolgt von einer Beschreibung des Aufbaus, der Möglichkeiten und der Technik der Benutzerschnittstellen in Kapitel 4. Kapitel 5 widmet sich dem verwendeten Datenmodell. Es folgt eine umfassenden Erläuterung der verwendeten Techniken der Implementation. Die Arbeit schließt mit einem Kapitel zur organisatorischen Einbindung des Systems in das Netzwerk der Mathematik, sowie einem Resumé.

2 Anforderungen

Das Hauptaugenmerk bei der Konzeption des WakeUpManagers liegt darauf, dass sich das System nahtlos in den Arbeitsalltag der Benutzer einfügt und keine künstlichen Hürden geschaffen werden. Nur so kann gewährleistet werden, dass das System von der Masse der Benutzer angenommen und nicht umgangen wird. Aus diesem Grund haben die funktionalen Anforderungen und die Bequemlichkeit der Benutzer oberste Priorität.

2.1 Funktionale Anforderungen

Der primäre Einsatzzweck des WakeUpManager besteht darin, die Arbeitsplatz- und Poolrechner zeitgesteuert starten und herunterfahren zu können. Ein besonderer Komfort für die Benutzer wäre es, wenn der Rechner bereits gestartet ist, wenn der Benutzer sein Büro bzw. den Poolraum betritt. Auf keinen Fall darf es jedoch passieren, dass ein Rechner, an dem ein Benutzer arbeitet — sei es lokal oder aus der Ferne — heruntergefahren wird.
Da die Arbeitszeiten der einzelnen Benutzer variieren, ist es notwendig, dass jeder Benutzer die Start- und Herunterfahrzeiten seines Arbeitsplatzrechners individuell konfigurieren kann. Eine Auswahlmöglichkeit aus vordefinierten Zeitenschemata ist vorgesehen, soll aber nicht paradigmatisch sein. Die Zeitpläne der Rechner eines Poolraum sollen von Studierenden nicht geändert werden können.

2.1.1 Fernzugriff

Die durchgehende Verfügbarkeit der Mitarbeiter- und Poolrechner erlaubt den Benutzern im Moment jederzeitigen Zugriff auf einen Rechner. So können sie aus der Ferne Berechnungen starten oder auf möglicherweise lokal abgelegte Daten, wie z.B. Zwischenergebnisse von Berechnungen, zugreifen. Diese Möglichkeit muss nach der Einführung des Systems weiterhin vorhanden sein.
Daher ist es notwendig, dass ein Benutzer einen Rechner aus der Ferne starten kann, damit der Zugriff auf den eigenen Arbeitsplatzrechner bzw. einen Poolrechner jederzeit gewährleistet ist (*Remote-Boot-Funktion*). Ist der Nutzer außerhalb der konfigurierten Zeiten in seinem Büro bzw. einem Poolraum anwesend, so kann er jeden Rechner durch Druck auf den Powerknopf starten.
Diese Anforderungen ermöglichen es, dass jeder Rechner zu jeder Tages- und Nachtzeit verfügbar ist, so dies notwendig bzw. gewünscht ist.

2.1.2 Organisation

Zur einfachen Vergabe von Rechten (s. Kapitel 2.2.1) für eine Gruppe von Rechnern — z.B. alle Rechner eines Poolraums — muss der WakeUpManager es gestatten, angeschlossene Rechner zu Rechnergruppen zusammenzufassen. Es soll ebenfalls möglich sein, Gruppen wieder zu einer Gruppe zusammenzufassen und somit eine hierarchische Ordnung abbilden zu können, die dem universitären Organisationsschema entspricht. Diese Funktionalität trägt auch zur besseren Übersichtlichkeit für Benutzer und Administratoren bei und erleichtert z.B. das Auffinden des eigenen PCs im System. Abbildung 2.1 zeigt eine mögliche Abbildung der Organisationsstruktur der Universität Paderborn.

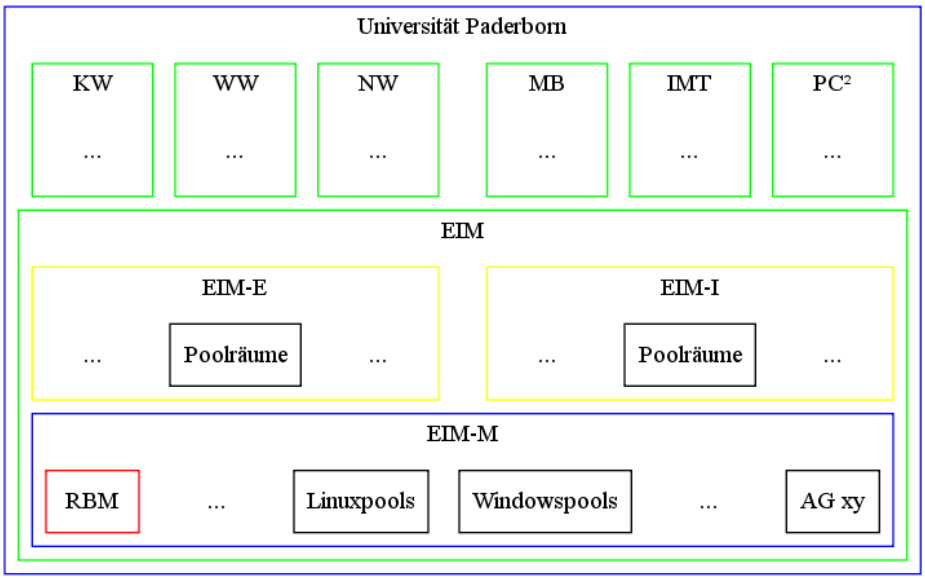

Abbildung 2.1: Beispiel einer Gruppenhierarchie an der Universität Paderborn

2.2 Allgemeine Anforderungen

Das System speichert Daten, die Rückschlüsse auf das Verhalten der Benutzer zulassen. Es ist deshalb unerlässlich, zum Schutz der Privatsphäre ein Berechtigungssystem zu konzipieren und zu implementieren.

2.2.1 Rollen und Rechte

Das Berechtigungssystem muss gewährleisten, dass explizit festgelegt werden kann, welche Benutzer welche Berechtigungen an welchem Rechner bzw. einer Gruppe von Rechner besitzen. Es soll konfigurierbar sein, ob ein Nutzer das Recht hat, einen Rechner manuell zu starten oder den Zeitplan eines Rechners lesen bzw. ändern zu können.
Das System muss beispielsweise erlauben, dass allen Mitgliedern einer Arbeitsgruppe das Recht eingeräumt werden kann, alle der AG zugeordneten Rechner über das Netzwerk zu starten, jedoch nur die Zeitsteuerung des jeweils eigenen Rechners einsehen bzw. ändern

zu können. Es soll alles verbieten, was nicht explizit erlaubt wurde und hat mögliche Hierarchien und vererbte Rechte zu beachten.

2.2.2 Datenschutz

Im Sinne der Datenvermeidung ist darauf zu achten, dass nur Daten in Logfiles geschrieben werden, die zum Betrieb des Systems bzw. zur Problemdiagnose und -behebung notwendig sind.

2.2.3 Lizenz

Um das fertige System unter einer OpenSource Lizenz wie z.B. der GNU General Public License (vgl. [Fre07]) veröffentlichen zu können, ist darauf zu achten, dass alle verwendeten Bibliotheken ebenfalls unter einer freien und kompatiblen Lizenz zur Verfügung stehen.

2.2.4 Mandantenfähigkeit

Es wäre wünschenswert, ein mandantenfähiges System zu entwickeln, das auch von anderen Fachbereichen und Instituten genutzt werden kann, ohne dort eine komplette eigene WakeUpManager-Infrastruktur bereitstellen zu müssen. Die Mandantenfähigkeit beinhaltet explizit die Möglichkeit, externen Benutzern die Administration bestimmter Rechner bzw. Gruppen von Rechner zu erlauben.

2.3 Technische Anforderungen

2.3.1 Softwareanforderungen

Damit das System in einer heterogenen Umgebung, wie dem Netzwerk der Mathematik, eingesetzt werden kann, muss darauf geachtet werden, dass alle verwendeten Softwarekomponenten für alle betriebenen Plattformen zur Verfügung stehen.

2.3.2 Konflikte mit administrativen Aufgaben

Die Erledigung administrativer Aufgaben macht es notwendig, alle Rechner bzw. alle Rechner einer Gruppe zu einem bestimmten — vom Nutzer nicht änderbaren — Zeitpunkt automatisch starten zu lassen. Es muss beispielsweise konfigurierbar sein, dass alle verwalteten Rechner einmal pro Nacht starten, damit Softwareaktualisierungen eingespielt werden können.

2.3.3 Hardwareanforderungen

Um die Clientrechner der Mathematik über ein solches System starten zu können, ist es erforderlich, dass sämtliche Rechner über eine Technologie verfügen, die es erlaubt, den Rechner über ein angeschlossenes Netzwerk zu aktivieren. Alle aktuellen Rechner verfügen bereits serienmäßig über die sogenannte *Wake-on-LAN*-Technologie (WoL), die es durch Absenden einer speziellen Netzwerknachricht erlaubt, einzelne Rechner im angeschlossenen Netzwerk gezielt zu booten. Sämtliche Poolrechner der Mathematik, insbesondere die in den letzten Jahren angeschafften Client-PCs der Firma Dell, unterstützen Wake-on-LAN.

3 Architektur

Aus den zuvor genannten Anforderungen entsteht die Notwendigkeit, den WakeUpManager als verteiltes System zu konstruieren. Es erscheint sinnvoll, die benötigten Funktionen auf einen zentralen Server, im Netzwerk verteilten Agenten und angeschlossenen Client-PCs aufzuteilen.

3.1 WakeUpManager Server

Der WakeUpManager Server wird als zentrale Verwaltungsinstanz konzipiert. Er stellt Schnittstellen für Benutzer und Client-PCs bereit; von ihm aus werden alle Aktivitäten des Systems gesteuert.

Die zum Betrieb notwendigen Daten über Netzwerke, Rechner, Agenten und Benutzerrechte sind in einer Datenbank gespeichert, auf die der Server bei Bedarf zugreift.
Der Aufbau der Datenbank wird in Kapitel 5 beschrieben.

3.1.1 Benutzerschnittstellen

Der Server wird so konzipiert, dass er zwei Schnittstellen für Benutzer bereitstellt. Über ein Web-Interface kann der Benutzer Rechner starten, Zeitpläne einsehen/ändern oder konfigurieren, ob ein bzw. "sein" Rechner automatisch gestartet und/oder heruntergefahren werden soll. Durch die Möglichkeit, den automatischen Start eines Rechner zu deaktivieren, ist es dem Benutzer möglich, längeren Abwesenheitsphasen, wie z.B. Urlaub oder

Forschungsreisen, Rechnung zu tragen. Diese Funktion darf weder den zeitgesteuerten administrativen Start eines Rechners deaktivieren, noch die Möglichkeit des Rechnerstarts bei Bedarf verhindern. Die Möglichkeit, das automatische Herunterfahren eines Rechner komplett zu deaktivieren wurde hauptsächlich der Vollständigkeit halber in das Konzept aufgenommen. Es könnte für administrative Aufgaben von Vorteil sein.

Alle genannten Funktionen sollen nur dann zur Verfügung stehen, wenn die jeweilige Berechtigung vorhanden ist. Die Authentifikation des Benutzers erfolgt automatisch durch den in der Mathematik eingesetzten Authentifikationsdienst *Kerberos* (vgl. [Ric07]), so der genutzte Browser dies unterstützt. Ist dies nicht der Fall, muss sich der Nutzer manuell durch seinen Benutzername und sein Passwort authentifizieren.

Neben dem Web-Interface soll auch ein Kommandozeilenprogramm namens `wum_boot` geschafffen werden, welches auf allen Rechnern der Mathematik installiert wird. Es erlaubt das Starten eines Rechner, ohne zuvor einen Webbrowser öffnen zu müssen. Der Name des zu startenden PCs wird als Parameter übergeben und durch `wum_boot` an den WakeUpManager-Server gesendet, der nach Überprüfung der Benutzerrechte den Start des Rechners in die Wege leitet. Die Authentifikation des Benutzers erfolgt ebenfalls automatisch über Kerberos.

Die Nutzung dieses Programms bietet sich z.B. an, wenn ein Nutzer per SSH im Netzwerk der Mathematik eingeloggt ist und auf seinen Arbeitsplatzrechner zugreifen möchte, obwohl dieser gemäß Zeitplan heruntergefahren ist. `wum_boot` soll insbesondere auf den SSH-Gateways der Mathematik installiert werden.

Die näheren Details der Benutzerschnittstellen werden in Kapitel 4 beschrieben, implementative Details sind in Kapitel 6.4 und 6.5 zu finden.

3.1.2 Zeitsteuerung

Der WakeUpManager Server ist auch dafür verantworlich, dass alle Rechner zu den vom Benutzer konfigurierten Startzeiten zur Verfügung stehen. Das Programm (`wakeup_cron`) übernimmt die Aufgabe, den zeitgesteuerten Start der Klienten zu koordinieren. Es wird in regelmäßigen Abständen ausgeführt und initiiert jeweils den Start aller Rechner, die innerhalb eines konfigurierbaren Zeitintervalls ab dem Ausführungszeitpunkt einen eingetragenen Startzeitpunkt aufweisen. In der Standardkonfiguration ist das Zeitintervall dreimal so lang gewählt, wie der zeitliche Abstand, in dem das `wakeup_cron` Programm ausgeführt wird. Dies geschieht in der Hoffnung, kurzzeitige Netzwerkausfälle kompensieren zu können, indem die Startaufforderung mit zeitlichen Abstand mehrfach versand wird. Sollte der Zielrechner bereits beim ersten Versuch starten, so werden die beiden darauf folgenden Startaufforderungen ignoriert und verändern den Rechnerstatus nicht.

3.2 WakeUpManager Boot-Agenten

Die WakeUpManager Boot-Agenten (`wakeup_agent`) übernehmen die Funktion, Clientrechner über das Netzwerk zu starten. Sie sind eigenständige Programme, die auf Anfrage des WakeUpManager Servers durch Aufruf des Programms `etherwake` sogenannte *Wake-on-LAN*-Pakete an den zu startenden Client senden.
Wake-on-LAN-Pakete (auch *Magic Packets* genannt) sind Teil einer Technologie zum Starten von Computern über das Netzwerk. Sie wurde hauptsächlich in Kooperation von AMD und Hewlett Packard entwickelt und stellt mittlerweile den Quasistandard in diesem Bereich dar, der von nahezu allen aktuellen Desktoprechnern unterstützt wird.

AMD schreibt in einem bereits 1995 veröffentlichen Whitepaper über die "Magic Packet Technologie" ([AMD95]):

> This collaboration resulted in a proposal for an industry standard mechanism that allows a networked PC to go completely asleep (Deep Green), yet still allows the network administrator or some kind of network management software to wake up the PC simply by sending it a specific Ethernet frame. This standard Ethernet frame contains a specific data pattern detected by the Ethernet controller on the receiving end. The Ethernet controller then alerts the system and the power mangement circuitry wakes it up.

Wake-on-LAN-Pakete können über das *Internetprotokoll* (*IP*, vgl. [Tan03a]), dem heute meist genutzten Netzwerkprotokoll, übertragen werden und sind somit *routingfähig* (vgl. [Tan03b]). Sie können daher ohne weiteres in netzwerktopologisch weit entfernte Netze übertragen werden. Dies bedeutet in der Praxis, dass die WoL-Pakete von einer zentralen Instanz, wie z.B. dem WakeUpManager Server, an alle angeschlossenen Rechner geschickt werden könnten, wenn alle Netzwerke, über die dieses Paket übertragen werden müsste, das Internetprotokoll verwenden.

Aufgrund der Tatsache, dass WoL-Pakete — notwendigerweise[1] — über das verbindungslose *User Data Protocol* (*UDP*, vgl. [Tan03c]) transportiert werden, ist es beim Einsatz einer zentralen Sendeinstanz jedoch nicht möglich, die Ankunft der Pakete zu garantieren. In einem überlasteten oder kurzzeitig unterbrochenen Netzwerk würden sie verloren gehen. Einige Router sind so konfiguriert, dass sie Pakete, die an eine Broadcast-Adresse eines angeschlossenen Netzwerkes gesendet werden sollen, verwerfen, um mögliche Angriffe zu verhindern.

Daher wurden die WakeUpManager Agenten so konzipiert, dass ein Agent jeweils direkt an ein Netzwerksegment bzw. *VLAN* (vgl. [Tan03d]) angeschlossen wird, in dem sich durch den WakeUpManager zu startende Clientrechner befinden. Die Agenten stellen einen RPC-Dienst für den WakeUpManager Server bereit, der das *Transmission Control Protocol* (*TCP*) als Transportprotokoll verwendet. Da TCP speziell für zuverlässige End-zu-End Kommunikation über ein unzuverlässiges Netzwerk entwickelt wurde (vgl. [Tan03e]), kann sichergestellt werden, dass Anfragen an den RPC-Dienst des Agenten korrekt empfangen werden bzw. ein Ausfall des Netzwerkes erkannt wird.
Somit wird ein Teil des Transferwegs als Fehlerquelle ausgeschlossen, die Auslieferung des WoL-Paketes im lokalen Netzwerk bleibt jedoch unwägbar. Die Wahrscheinlichkeit, dass das Paket sein Ziel erreicht, wird durch mehrfache Auslieferung erhöht.
Das Konzept der verteilten WakeUpManager Agenten erlaubt es auch Rechner in — möglicherweise netzwerktopologisch weit entfernten — Mandantennetzen zuverlässig zu starten.

Ein Agent erkennt angeschlossene und auf Systemebene konfigurierte Netzwerke automatisch, erlaubt aber zusätzlich die manuelle Konfiguration für Spezialfälle. Die Kenntnis über angeschlossene Netzwerke ist erforderlich, da das Programm `etherwake` die Angabe einer Schnittstelle erfordert, über die das Wake-on-LAN-Paket verschickt werden soll.

[1] vgl. das bereits referenziert *Magic Packet* Whitepaper [AMD95]

Pro Netzwerksegment ist mindestens ein Agent erforderlich. Es ist jedoch ohne weiteres möglich, mehrere Agenten für ein Netzwerk einzurichten, sowie mehrere Netzwerke durch einen Agenten versorgen zu lassen.

Weitere implementative Details werden in Kapitel 6.6.2 beschrieben.

3.3 WakeUpManager Klienten

Die durch den WakeUpManager zu steuernden Klienten — d.h. die Mitarbeiter- und Poolrechner am Institut für Mathematik — sind selbstverständlich ebenfalls Teil des Systems. Auf jedem Klienten wird ein weiteres Programm (`wum_shutdown`) installiert, welches regelmäßig gestartet wird und den Rechner herunterfährt, so niemand daran arbeitet und dies vorgesehen ist.

`wum_shutdown` wird periodisch aufgerufen und prüft, ob der Rechner bereits eine konfigurierte Zeitspanne läuft, Benutzer interaktiv angemeldet sind oder auf dem Rechner Programme laufen, die das Herunterfahren verbieten. Der konfigurierte Zeitplan wird ebenfalls berücksichtigt.
Alle getroffenen Entscheidungen werden unter Linux im Syslog und unter Windows in einer eigenen Logdatei protokolliert, ohne personenbezogene Daten zu speichern. Somit ist eine spätere Überprüfung im Fehlerfalle gewährleistet.

3.4 Architekturübersicht

Aus den oben genannten Diensten ergibt sich die in Abbildung 3.1 dargestellte Architektur.

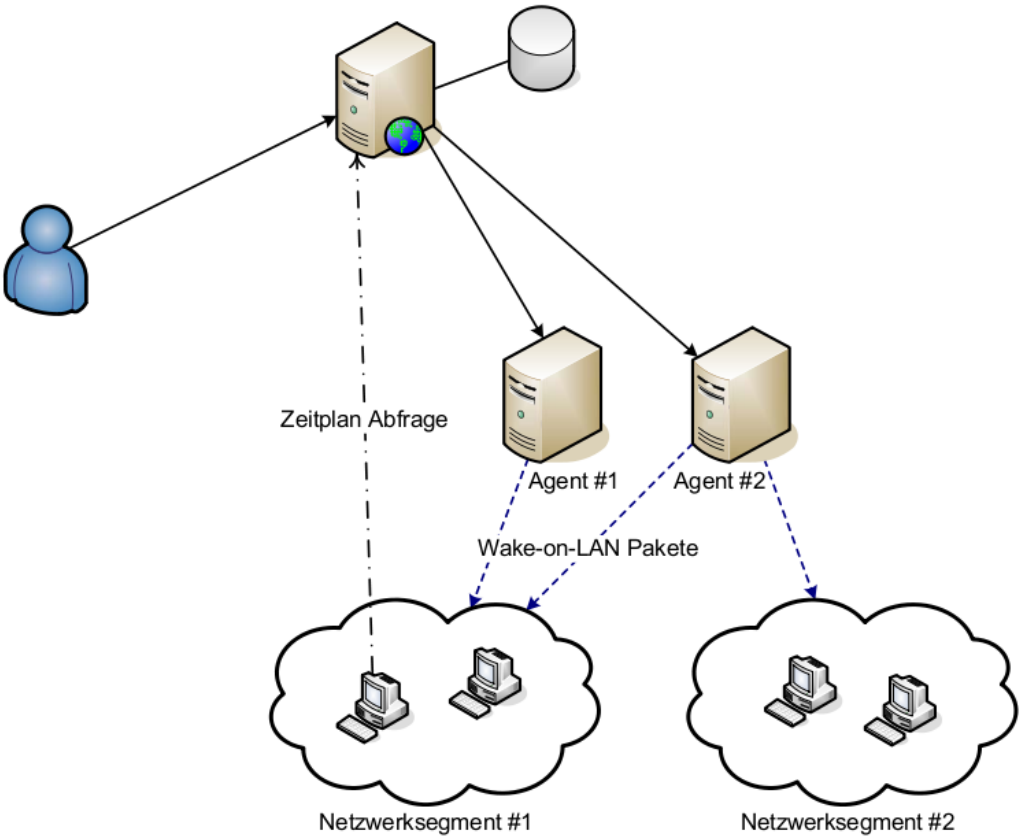

Abbildung 3.1: WakeUpManager Architekturübersicht

4 Benutzerschnittstellen

Dieses Kapitel beschreibt die Funktion und den Aufbau der Benutzerschnittstellen, die der WakeUpManager bereitstellt.

4.1 Web-Interface

Das Web-Interface stellt die zentrale Benutzerschnittstelle des WakeUpManagers dar. Hier stehen — je nach Berechtigung des Benutzers — alle Funktionen des WakeUpManagers zur Verfügung. Diese Schnittstelle wurde als Webapplikation konzipiert, da eine Webanwendung verhältnismäßig leicht zu realisieren ist und maximale Unabhängigkeit vom Betriebssystem des Clients garantiert. Freie Browser wie die "Mozilla"-Suite bzw. der "Firefox" stehen für alle gängigen Plattformen zur Verfügung.

Das WakeUpManager Web-Interface ist eine dynamische Webanwendung, d.h. dass die angezeigte Webseite bei jedem Aufruf mit den jeweils gültigen Parametern neu generiert wird. Dies ist erforderlich, da Benutzer Daten ändern oder Ereignisse auslösen können, wenn sie z.B. den Zeitplan ihres Rechners anpassen bzw. einen Rechner über den WakeUpManager starten. Das System muss auf diese Eingaben bzw. Ereignisse reagieren, Aktionen auslösen und eine Erfolgs- oder Fehlermeldung ausgeben. Aus diesem Grund wurde für den WakeUpManager eine eigene dynamische Webanwendung entwickelt, die an den Bedürfnissen des Systems ausgerichtet ist.

4.1.1 Konformität

Bei allen Teilen der Webanwendung wurde auf 100%ige Kompatibiliät zu den genutzten Standards geachtet. So halten sich sämtliche Webinhalte strikt an den *HTML 4.01* Standard (vgl. [Wor99]), wie er vom *World Wide Web Consortium (W3C)* veröffentlicht wurde.

Die Selbstbeschreibung des W3C aus seiner Webseite[1] lautet wie folgt:

> The World Wide Web Consortium (W3C) develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential.

Dort findet sich auch die Spezifikation von HTML 4.01 ([Wor99]):

> This specification defines the HyperText Markup Language (HTML), the publishing language of the World Wide Web. This specification defines HTML 4.01, which is a subversion of HTML 4. In addition to the text, multimedia, and hyperlink features of the previous versions of HTML (HTML 3.2 [HTML32] and HTML 2.0 [RFC1866]), HTML 4 supports more multimedia options, scripting languages, style sheets, better printing facilities, and documents that are more accessible to users with disabilities. HTML 4 also takes great strides towards the internationalization of documents, with the goal of making the Web truly World Wide.

Durch die konsequente Einhaltung dieses Standards kann maximale Kompatibilität zu allen verfügbaren Browsern gewährleistet werden. Die HTML-Standardkonformität kann mittels des vom W3C angebotenen Validator[2] überprüft werden.

Gleiches gilt für die *Cascading Style Sheets* (CSS) Spezifikation, die in der Version 2.1 vorliegt (vgl. [Wor07]). Mittels CSS wird das Layout (der Stil) einer Webseite, d.h. die

[1] http://www.w3.org
[2] http://validator.w3.org/

Anordnung und das Aussehen von Elementen, beschrieben. Der CSS-Standard wird ebenfalls vom W3C gepflegt und veröffentlicht. Die Spezifikation wird wie folgt beschrieben:

> This specification defines Cascading Style Sheets, level 2 revision 1 (CSS 2.1). CSS 2.1 is a style sheet language that allows authors and users to attach style (e.g., fonts and spacing) to structured documents (e.g., HTML documents and XML applications). By separating the presentation style of documents from the content of documents, CSS 2.1 simplifies Web authoring and site maintenance.

Der Einsatz von CSS ermöglicht eine einheitliche Anzeige einer Webseite in verschiedenen Browsern ("Mozilla", "Opera", "Internet Explorer", ...). Diese Technik wird beim WakeUpManager Web-Interface verwendet, um ein maximal kompatibles Layout zu gewährleisten. Die Standardkonformität der CSS-Elemente der WakeUpManager-Website kann über einen ebenfalls vom W3C angeboten Validator[3] erfolgen.

Neben Standardkonformität wurde auch Wert auf ein funktionales und schlichtes Design der Webanwendung gelegt, um die Benutzerführung einfach zu gestalten und den Nutzer nicht durch zuviele Möglichkeiten und Abläufe zu irritieren.

4.1.2 Templates

Eine dynamische Webanwendung arbeitet im Wesentlichen nach dem "Baukastenprinzip", d.h. sie setzt bestehende Text- und Layoutfragmente zu einer vollständigen Seite zusammen. Dieses Funktionsprinzip legt nahe, die einzelnen Fragmente unabhängig voneinander bzw. unabhängig vom Programmcode zu speichern. Dies kann die Komplexität der Anwendung verringern und erhöht die Wartbarkeit des Systems.

Solche Seitenfragmente bzw. Vorlagen für Webseiten werden auch *Templates* genannt.

[3] http://jigsaw.w3.org/css-validator/

Diese Technik wird im WakeUpManager Web-Interface genutzt, um statische und vordefinierte Inhalte von dynamischen Teilen zu trennen.

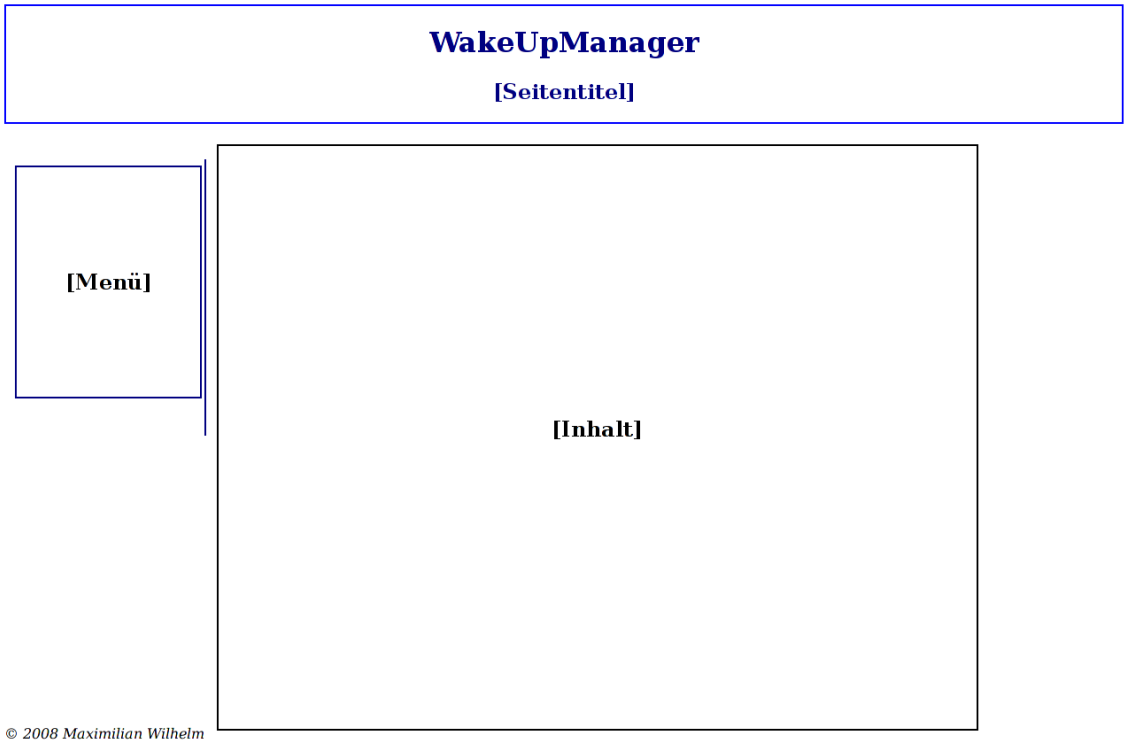

Abbildung 4.1: Globles WakeUpManager Seitenlayout

Die Nutzung erfolgt auf zwei Ebenen:

1. **Globales Seitenlayout**

   Ein Template bildet die Grundlage jeder Seite des WakeUpManager Web-Interfaces. Es enthält alle Metainformationen, die Überschrift "WakeUpManager" sowie drei Container für eine mögliche Unterüberschrift (Seitentitel), ein Menü sowie den eigentlichen Seiteninhalt (vgl. Abbildung 4.1). Die Webapplikation läd dieses Template und füllt die Container mit den entsprechenden Daten.

2. **Seiteninhalte**

   Jede Funktion des WakeUpManagers (Rechner starten, Zeitpläne einsehen bzw. ändern, ...) ist auf einer eigenen Seite dargestellt. Pro Seite existiert ein Template, das den Aufbau des Seiteninhalts sowie einen beschreibenden Text enthält. Dieses Template wird durch die Webapplikation geladen, ggf. mit Inhaltselementen gefüllt und in den Container für den Seiteninhalt des globalen Templates eingefügt.

Weitere implementative Details zur Umsetzung des Templating sind in Kapitel 6.5.2 beschrieben.

4.1.3 Einheitliches Design

Neben dem global einheitlichen Seitenaufbau halten sich alle Seiten des WakeUpManagers an ein einheitliches Konzept. Der Inhaltsteil der Seite startet stets mit einer Beschreibung der angebotenen Funktionalität sowie einer Anleitung für den Anwender, wie diese Funktion zu nutzen ist. Darauf folgt ein eingerahmtes Formular, in dem aus einer Liste von Rechnern derjenige gewählt werden kann, auf den die jeweilige Funktion angewand werden soll. Die Liste enthält nur Rechner, an denen der Benutzer das Recht besitzt, die Funktion auszuüben.

Sollte der Benutzer Berechtigungen für eine oder mehrere Gruppen von Rechnern haben, so wird oberhalb der Rechnerauswahlliste eine Auswahl auf Gruppenebene angezeigt. Durch Selektion einer Gruppe wird der Inhalt der Rechnerauswahlliste im Hintergrund neu geladen, sodass nur noch Rechner der gewählten Gruppe angezeigt werden. Dies erleichtert die Auswahl für Benutzer bzw. Administratoren, die Berechtigungen an einer großen Zahl von Rechnern besitzen.

Sollte ein Benutzer eine der angebotenen Funktionen (Rechner starten, Zeitplan/Rechnerkonfiguration einsehen bzw. ändern) für keinen im System bekannten Rechner ausführen dürfen, so wird der jeweilige Menüeintrag automatisch entfernt.

Abbildung 4.2: “Rechner starten“-Seite des WakeUpManager Web-Interfaces

Abbildung 4.2 zeigt die Seite “Rechner starten“ in deutscher Sprache, für einen Benutzer, der jede Funktion des System nutzen kann, nachdem der “Rechner starten“-Button angeklickt wurde.
Die untere “Ergebnis“-Box wird erst angezeigt, nachdem die jeweilige Funktion einer Seite ausgelöst wurde. So möglich, wird sie im Hintergrund geladen (s. Kapitel 4.1.5).

4.1.4 Mehrsprachigkeit

Alle Teile des WakeUpManager Web-Interfaces sind so konzipiert, dass die Inhalte in mehreren Sprache angeboten werden können. Die Entwicklung eines multilingualen Web-Interfaces erschien sinnvoll, da regelmäßig externe Dozenten zu Gast am Institut für

Mathematik sind, die nicht die deutsche Sprache beherrschen. Da diese häufig für längere Zeit in Paderborn sind, gehören sie ebenfalls zum Benutzerkreis des WakeUpManagers und ihren Bedürfnissen muss Rechnung getragen werden.

Das Konzept sieht vor, dass für jede zu unterstützende Sprache ein Satz von Templates, d.h. ein Template pro Seite, angelegt wird, sodass die Lokalisation soweit wie möglich unabhängig vom Programmcode geschehen kann. Die Webanwendung prüft selbstständig, ob die Sprachen, die ein Benutzer in seinem Webbrowser als Wunschsprache für Webseiten konfiguriert hat, unterstützt werden.[4] Werden mehrere vom Benutzer bevorzugte Sprachen unterstützt, so wählt das System die Sprache, die der Nutzer am stärksten priorisiert hat.

Hat der Benutzer keine Sprachpräferenzen in seinem Browser konfiguriert oder wird keine der Sprachen vom WakeUpManager unterstützt, so findet die Ausgabe automatisch in englischer Sprache statt.
Im Moment kann der WakeUpManager in deutscher und englischer Sprache genutzt werden.

4.1.5 AJAX

Seit einiger Zeit geht der Trend bei der Entwicklung von dynamischen Webanwendungen dahin, nicht mehr die vollständige Webseite neuzuladen, wenn der Benutzer Daten verändert oder Ereignisse ausgelöst hat. Neuerdings wird nur noch ein Teil der Webseite neu- bzw. dazugeladen. Dies wird durch das asynchrone Nachladen von Inhaltselementen erreicht, sodass sich eine Webanwendung für den Nutzer (nahezu) wie eine normale interaktive Anwendung verhält.
Diese Technik wird gemeinhin unter dem Begriff *Asynchronous JavaScript and XML* (*AJAX*) zusammengefasst. Sie wird innerhalb des WakeUpManager Web-Interfaces eingesetzt, um auf vom Benutzer ausgelöste Ereignisse zu reagieren.

[4] s. http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.4

Da diese Technik die Unterstützung der Scriptsprache *JavaScript* durch den genutzten Webbrowser voraussetzt, kann es zu Einschränkungen bei der Kompatibilität mit veralteten Browsern kommen. Daher wurden alle mittels der AJAX-Technologie implementierten Abläufe auch auf klassischem Wege implementiert. Die Webanwendung überprüft selbstständig, ob der genutzte Webbrowser JavaScript unterstützt und aktiviert die Nutzung der AJAX-Elemente nur, falls sie genutzt werden können.

4.2 wum_boot

Neben dem oben genannten Web-Interface, das sämtliche Funktionen des WakeUpManagers bereitstellt, existiert ein Kommandozeilenprogramm namens `wum_boot`. Es ermöglicht, einen Rechner über das Netzwerk zu starten, ohne zuvor einen Webbrowser öffnen zu müssen.

Das Programm erwartet den Namen des zu startenden Rechners als Parameter und sendet eine Anfrage an den WakeUpManager Server, dass der automatisch durch Kerberos authentfizierte Benutzer den angegebenen Rechner starten möchte. Besitzt der Benutzer das Recht für die Aktion, wird der Rechner gestartet und dem Benutzer wird eine Erfolgsmeldung ausgegeben. `wum_boot` überprüft vor der Ausgabe der Meldung die `$LANG` Umgebungsvariable und wählt die vom Benutzer präferierte Sprache, so diese unterstützt ist. Ist `$LANG` nicht definiert oder ist eine nicht unterstützte Sprache angegeben, so erfolgt die Ausgabe in Englisch.

Rechnernamen werden innerhalb des WakeUpManager System als voll qualifizierte Namen (FQDN[5], vgl. [PA01a]) gespeichert, d.h. so, wie sie im DNS[6] eingetragen sind. Daher ist es notwendig, dass `wum_boot` dem Server den Namen des zu startenden Rechners als FQDN übermittelt.

Damit der Benutzer einen Rechnernamen wie z.B. `regiomontanus.math.uni-paderborn.de` nicht vollständig angeben muss, unterstützt `wum_boot` die Autovervollständigung eines Rechnernamens durch nachschlagen im DNS-System (vgl. [PA01b]).
Der Aufruf

```
wum_boot regiomontanus
```

ist daher ausreichend.

[5]Fully Qualified Domain Name
[6]Domain Name System

Das Programm ist auf allen WakeUpMananager Klienten, sowie den SSH-Gateways installiert. Die SSH-Gateways stellen — neben einer VPN[7]-Verbindung — den alleinigen Einstiegspunkt in das Netzwerk der Mathematik von außen dar. Daher muss sich jeder Benutzer, der nicht via VPN mit dem Netzwerk der Mathematik verbunden ist, zuerst auf einem der Gateways anmelden, wenn er auf einen Arbeitsplatz- oder Poolrechner zugreifen möchte.

[7] Virtual Private Network

5 Datenmodell

Um die in Kapitel 2 geforderten bzw. in Kapitel 3 und 4 beschriebenen Dienste und Funktionen anbieten zu können, benötigt der WakeUpManager einige Daten über die angeschlossenen Netzwerke, darin enthaltene Klienten, deren Zeitpläne und Benutzerrechte. All diese Daten werden in einer zentralen Datenbank gespeichert, die in der Mathematik von einem PostgreSQL Datenbankserver in der Version 8.1 verwaltet wird.

Dieses Kapitel beschreibt den Aufbau der Datenbank und diskutiert die getroffenen Entwurfsentscheidungen.

5.1 Netzwerke und Agenten

Der WakeUpMananger soll Clientrechner in beliebig vielen Netzwerken starten bzw. verwalten können. Deshalb muss er zur Auswahl des zuständigen WakeUpManager Agenten wissen, welcher Agent mit welchem Netzwerk verbunden ist bzw. welche Klienten ein Agent starten kann.
Es ist daher notwendig, dass Informationen über die Agenten — insbesondere die IP-Adresse der Rechner, auf denen sie installiert sind — in der Datenbank gespeichert werden. Dies geschieht in der Tabelle *agent*, in der jeder Agent durch eine eindeutige ID repräsentiert wird. Hier sind die IP-Adresse, ein Name und eine optionale Beschreibung hinterlegt.

Das für den Versand von Wake-on-LAN Paketen verwendete Programm `etherwake` erfor-

det die Angabe der Netzwerkschnittstelle, die zum Versand des Paketes genutzt werden soll. Diese Information ermittelt der Agent, indem er die IP-Netzwerkadresse des Netzes, an das der zu startende Klient angeschlossen ist, mit den lokal konfigurierten Netzen an seinen Schnittstellen vergleicht. Daher ist es notwendig, die IP-Netzwerkadressen aller Netzwerke, an die durch den WakeUpManager verwaltete Klienten angeschlossen sind, ebenfalls in der Datenbank zu speichern.

In der Tabelle *network* sind alle Netzwerke samt ihrer IP-Netzwerkadresse im CIDR-Format (*Classless inter domain routing*, vgl. [Tan03f]), einem Namen sowie einer optionalen Beschreibung gespeichert. Jedes Netz wird analog zu den Agenten durch eine eindeutige ID (`net_id`) repräsentiert; die IDs der Agenten und Netze werden in der *agent_network*-Tabelle verwenden, um die "Agent ist angeschlossen an Netz"-Beziehung abzubilden. Durch die Modellierung dieser Beziehung als m:n Kardinalität ist es sowohl möglich, "mehrere Agenten pro Netz" als auch "mehrere Netze pro Agent" in der Datenbank abzubilden. Dieses Vorgehen ist naheliegend und muss nicht weiter begründet werden.

5.2 Clientrechner

Zentraler Bestandteil des WakeUpManagers sind die durch das System verwalteten Klienten. Es ist offensichtlich, dass einige Informationen über die Klienten zum Betrieb des Systems notwendig sind und daher in der Datenbank vorgehalten werden. Durch den WakeUpManager verwaltete Klienten erhalten einen Eintrag in der *host*-Tabelle. Dies beinhaltet

- eine eindeutige ID (`host_id`)
- den voll qualifizierten Rechnernamen (FQDN[1], vgl. [PA01a]), wie er im DNS[2] eingetragen ist
- das Netzwerk, an das der PC angeschlossen ist, in Form einer Referenz auf einen Eintrag in der *network*-Tabelle (*FOREIGN KEY*)
- sowie die Adresse der Netzwerkkarte, die diese Verbindung herstellt (*MAC-Adresse*)

Die Relation zwischen Rechnern und Netzwerken wird über die eindeutige `net_id` hergestellt.

Neben diesen infrastrukturbezogenen Informationen wird in der *host*-Tabelle auch gespeichert, welcher Zeitplan für diesen Rechner Anwendung findet, ob der PC automatisch gestartet bzw. heruntergefahren werden soll (`boot_host` bzw. `shutdown_host`), sowie zu welcher Gruppe von Rechnern er gehört. Die Zuordnung zu einer Rechnergruppe bzw. den für den Rechner zu nutzenden Zeitplan findet jeweils über die ID der Gruppe bzw. des Zeitplans statt.

Diese Modellierung der "Rechner ist Mitglied in Gruppe"-Beziehung besitzt offensichlich die Kardinalität *1:n*, d.h. ein Rechner kann nur Mitglied exakt einer Gruppe sein; ein

[1] Full qualified domain name
[2] Domain Name System

Datenbankconstraint erzwingt dies sogar, sodass er Mitglied einer Gruppe sein muss. Die Beziehung wurde mit *1:n* Kardinalität modelliert, da im Rechnerverbund der Mathematik kein Fall konstruierbar war, in dem ein Rechner Mitglied in mehreren Gruppen sein konnte, und es im Vergleich zu einer eigenständigen Beziehungstabelle das Datenbankschema vereinfacht. Sollte im Nachhinein die Notwendigkeit entstehen, die Kardinalität auf *n:m* zu erweitern, so kann dies durch Einfügen einer `hostgroup_host`-Tabelle und Änderung der WakeUpManager-Datenbank-API geschehen.

5.3 Rechnergruppen / Gruppenhierarchie

Aus der Anforderungsdefinition geht hervor, dass der WakeUpManager eine Möglichkeit bieten muss, Rechner zu Gruppen zusammenzufassen, die ihrerseits wieder zu Gruppen zusammengefasst werden können. Es entsteht so eine Abbildung einer hierarchischen Ordnung bzw. ein Gruppenbaum.

Rechnergruppen werden in der Tabele *hostgroup* gepflegt. Sie besitzen jeweils eine eindeutige ID (`hostgroup_id`), einen eindeutigen Namen, sowie optional einen administrativen Zeitplan bzw. eine Beschreibung.

Eine hierarchische Ordnung wird durch die *hostgroup_ tree*-Tabelle abgebildet, indem zwei Gruppen über die jeweilige `hostgroup_id` in eine `super group` $\supseteq$ `member group` Beziehung gebracht werden. Durch die Transitivität dieser Beziehung entsteht — bei konsequenter Definition — ein Gruppenbaum, d.h. eine vollständige Hierarchie.

Aus den an der Universität Paderborn vorhandenen Organisationsformen folgt, das eine Gruppe jeweils nur in einer anderen Gruppe enthalten sein kann. Durch ein zusätzliches Datenbankconstraint wird verhindert, dass einer Gruppe mehrere `super group`'s zugeordnet werden können, um eine eindeutige Zuordnung zu schaffen und Zyklen zu verhindern. Sollte diese Möglichkeit jedoch notwendig werden, so kann sie durch Entfernen des Constraints erlaubt werden.

Die WakeUpManager-Datenbank-API prüft die verwendeten Daten ihrerseits ebenfalls auf Zyklen und bricht sie ggf. auf. Dies wäre auch nach einem möglichen Entfernen des o.g. Constraints unproblematisch. Da der Gruppenbaum mehrfach genutzt wird und somit viele Abfragen auf diese Tabelle stattfinden, wurde auf beide Felder ein Index angelegt, um die Suchzeiten zu verringern.

Die Definition eines administrativen Zeitplans für eine Gruppe von Rechnern ist optional, da sich dieser Wert auf alle Gruppen, die in der Hierarchie unterhalb einer Gruppe stehen, vererbt, wenn für die Untergruppe kein administrativer Zeitplan eingetragen wurde. Abbildung 5.1 zeigt ein Beispiel einer Gruppenhierarchie inklusive vererbter administrativer Zeitpläne.

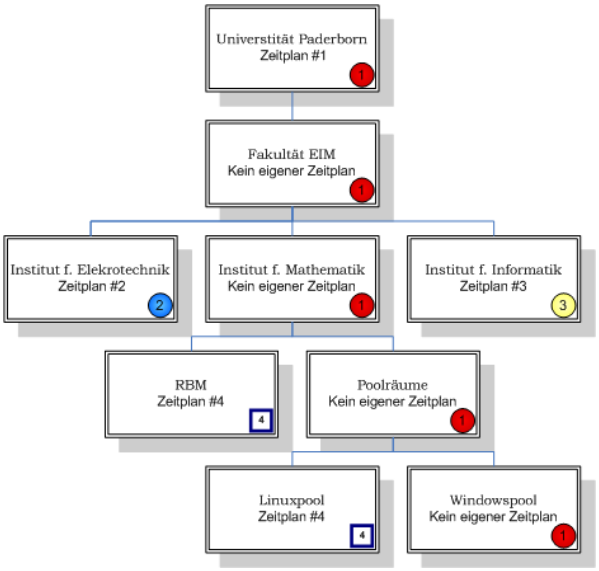

Abbildung 5.1: Beispiel einer Gruppenhierarchie mit vererbten Zeitplänen

Zur Vermeidung von überflüssigen Datenbankeinträgen werden beim Löschen einer Rechnergruppe sämtliche Einträge der *hostgroup_tree*-Tabelle mit der selben ID ebenfalls entfernt.

5.4 Zeitpläne

Ein weiterer fundamentaler Bestandteil des WakeUpManagers ist die Möglichkeit, Rechner zeitgesteuert starten und herunterfahren zu können. Diese Funktionaliät bedingt, dass das System Kenntnis über die gewünschten Zeitpunkte haben muss, an denen ein Rechner gestartet bzw. heruntergefahren werden soll.

Die Speicherung der Zeitpläne des WakeUpManagers erfolgt in einem zweistufigen Konzept. In der Tabelle *config_set* wird jeder im System bekannte Zeitplan mit einer eindeutigen ID (`csid`) und einem eindeutigen Namen repräsentiert. Die optionalen Parameter `preset` und `administrative` vom Typ boolean legen fest, ob ein Zeitplan von einem Administrator als Auswahlmöglichkeit für den Benutzer vorgegeben wurde bzw. ob es sich um einen administrativen Zeitplan handelt (vgl. Kapitel 2.3). Sollte kein Wert für diese Felder angegeben werden, wird ein neuer Zeitplan automatisch als "nicht vorgegeben" und "nicht administrativ" angelegt.
Der `preset`-Eigenschaft liegt die Überlegung zugrunde, dass es nach einer längeren Nutzungsdauer des System wahrscheinlich ist, dass einige ungenutzte Zeitpläne — und somit überflüssige Einträge in der Datenbank — existieren.

Folgende Ereignisse können einen Zeitplan überflüssig machen:

- Ein Benutzer verändert die Eintellungen eines Rechner und erzeugt somit einen neuen Zeitplan
- Es wird ein bereits existierender Zeitplan für einen Rechner konfiguriert. (Dies geschieht durch Änderung des `csid`-Feldes des entsprechenden Eintrages in der *host*-Tabelle.)
- Ein Rechner, für den ein individueller Zeitplan konfiguriert wurde, wird aus der Datenbank entfernt.

Wird bei einer Zeitplanänderung überprüft, ob der alte Zeitplan von keinem anderen Rechner mehr verwendet wird, so kann er aus der Datenbank gelöscht werden, wenn er nicht von einem Administrator vorgegeben wurde, d.h. das Feld `preset` einen wahren Wert enthält. Nach dem Entfernen eines Rechners aus der *host*-Tabelle sollte diese Überprüfung ebenfalls stattfinden.

Neben den oben genannten Metainformationen sind die gewünschten Zeitpunkte, an denen ein Rechner gestartet bzw. heruntergefahren werden soll von großer Bedeutung. Sie werden in der *times*-Tabelle als 4-Tupel gespeichert und sind mit jeweils einem Zeitplan verknüpft. Jedes Tupel besteht aus der Zeitplan ID, einem Tag in Form einer dreistelligen Angabe ("mon", "tue", ...), einer Uhrzeit und einer Aktion, die zu diesem Zeitpunkt ausgeführt werden soll ("boot" oder "shutdown"). Jeder Eintrag erhält zusätzlich eine eindeutige ID (`tid`).

Die Modellierung der Zeitpläne durch die Speicherung von Zeitpunkten erlaubt maximale Flexibilität bei der Definition der gewünschten Zeitspannen, in denen ein Rechner verfügbar sein soll. Es ist möglich, mehrere Intervalle für einen Tag anzulegen bzw. ein Intervall, dass mehrere Tage einschließt.
Durch ein Datenbankconstraint ("*ON DELETE CASCADE*") wird sichergestellt, dass bei der Löschung eines Zeitplans aus der *config_set*-Tabelle alle ihm zugeordneten Einträge in der *times*-Tabelle ebenfalls gelöscht werden. Dies ist ein weiterer Schutz vor ungenutzten Einträgen in der Datenbank.

5.5 Rechte

Das in den allgemeinen Anforderungen in Kapitel 2.2 beschriebene Berechtigungssystem benötigt seinerseits ebenfalls einen Speicher für Benutzerrechte.
Um eine einheitliche und zentrale Speicherung aller durch den WakeUpManager verwendeten Daten zu schaffen, werden die Berechtigungen ebenfalls in der zentralen Datenbank abgelegt. Es wäre auch möglich, diese Daten an die Einträge in der Benutzerdatenbank des Rechnerverbundes der Mathematik zu binden, die in einem zentralen LDAP-Verzeichnisdienst (vgl. [Ope08]) verwaltet werden. Dies würde jedoch zu einer Aufteilung zusammengehöriger Daten führen und zwangsläufig ein Synchronisationsproblem aufwerfen, da beim Entfernen eines Rechners aus der WakeUpManager-DB sämtliche Berechtigungen, die diesen Rechner betreffen, aus der Benutzerdatenbank gelöscht werden müssten.

Unter dem Gesichtspunkt der Mandantenfähigkeit wäre die Nutzung eines LDAP-Dienstes — bzw. einer beliebigen anderen externen Datenbank — als Speicherort für Berechtigungen innerhalb des WakeUpManagers ebenfalls nicht sinnvoll, da Mandanten höchst wahrscheinlich über keinen Zugriff auf die LDAP-Datenbank der Mathematik verfügen oder weitere externe Datenbanken vom System genutzt werden müssten.

Somit stellt die zentrale Speicherung sämtlicher Berechtigungen in der WakeUpManager Datenbank den — auch unter implementativen Gesichtspunkten — einfacheren Weg dar. Ein gewisses Synchronisationsproblem bleibt jedoch bestehen, da eingetragene Rechte beim Löschen eines Benutzerkontos ebenfalls gelöscht werden müssen. Dies muss manuell gelöst werden.

Aufgrund des globalen Namensraumes für Benutzernamen an der Universität Paderborn wurde der Namensraum in der WakeUpManager Datenbank ebenfalls flach modelliert. Durch die Einführung eines Wertes für die Domäne bzw. den Mandanten kann diese Struktur jedoch leicht erweitert werden.

Der WakeUpManager kennt drei Arten von Berechtigungen, die jeweils als Wahrheitswerte gespeichert werden. Wird kein Wert angegeben, so wird jedes nicht explizit angegebene Recht standardmäßig nicht gewährt.
Berechtigungen gelten stets nur für einen Benutzer, der über seinen Benutzernamen identifiziert wird. Sie können explizit an einen Rechner gebunden sein oder für eine Gruppe von Rechnern gelten. Die zuvor beschriebene Gruppenhierarchie gilt ebenfalls für Berechtigungen, d.h. dass ein Recht, das für eine `super group` gewährt wurde ebenfalls für alle `member groups` sowie deren `member groups` gilt.

Die drei verfügbaren Berechtigungen sind:

1. `allow_boot`

   Dieser Wert legt fest, ob ein Benutzer die Berechtigung hat, einen Rechner bzw. eine Gruppe von Rechnern mittels des WakeUpManagers über das Netzwerk zu starten. Dieses Recht wird in der Mathematik für Studierende für die Gruppe aller Poolrechner erteilt, sowie für Mitarbeiter für die Gruppe aller Rechner im Mitarbeiternetz.

2. `read_config`

   Der Wert `read_config` legt fest, ob ein Benutzer das Recht hat, den Zeitplan eines Rechners bzw. einer Gruppe von Rechnern einzusehen. Ein Benutzer, der diese Berechtigung besitzt, kann ebenfalls einsehen, ob ein Rechner bzw. eine Rechnergruppe automatisch gestartet oder heruntergefahren werden soll (*Aktivierungsstatus*). Das Recht wird meist in Verbindung mit dem folgenden Wert genutzt.

3. `write_config`

   Über das Feld `write_config` erhält ein Nutzer die Erlaubnis, den Zeitplan eines Rechners bzw. einer Gruppe von Rechnern zu verändern. Der Aktivierungsstatus darf ebenfalls modifiziert werden.

   Dieses Recht schließt die Berechtigungen, die durch `read_config` erteilt werden, **nicht** ein. Diese müssen gesondert vergeben werden.

Zur Vermeidung überflüssiger Datenbankeinträge werden beim Löschen eines Rechners bzw. einer Gruppe von Rechnern sämtliche daran gebundenen Berechtigungen ebenfalls gelöscht.

5.6 Datenbankschema

Abbildung 5.2 zeigt das Schema der WakeUpManager-Datenbank inklusive sämtlicher Beziehungen graphisch. Auf allen rot umrandeten Feldern ist ein Index angelegt, der nur das einmalige Vorkommen eines Wertes erlaubt ("*UNIQUE INDEX*").

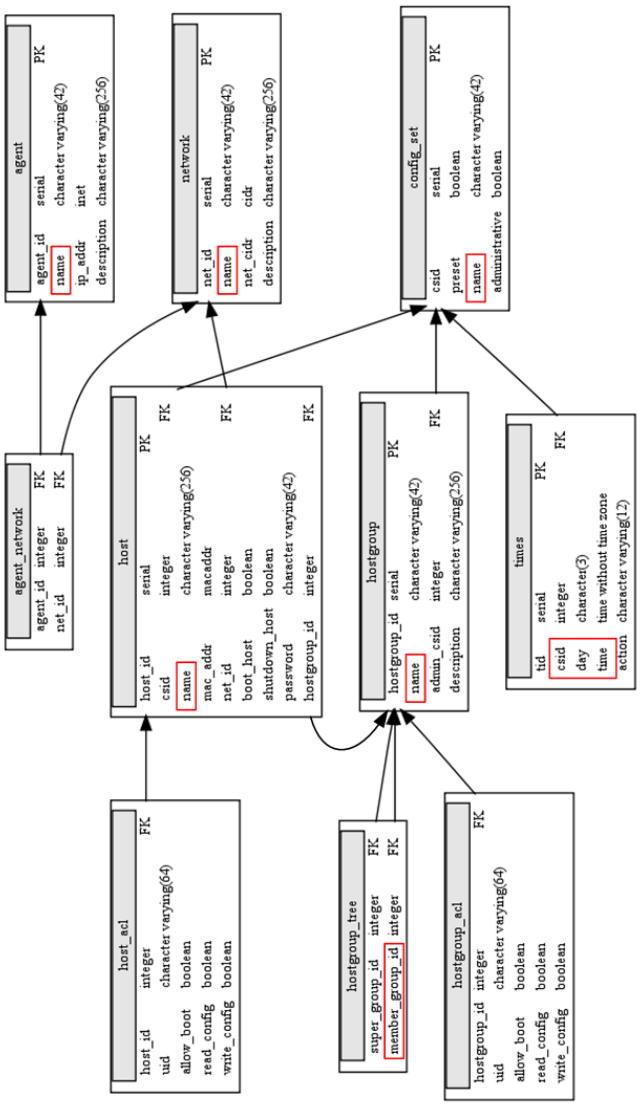

Abbildung 5.2: Schema der WakeUpManager Datenbank

6 Implementation

Dieses Kapitel behandelt implementative Details des WakeUpManagers.
Dies beinhaltet Erläuterungen zur Wahl der Programmiersprache, sowie die Vorstellung genutzter Techniken und Bibliotheken, sowie die Diskussion der Vor- und Nachteile der gewählten Umsetzung.

6.1 Perl

Die Wahl der Implementationssprache für den WakeUpManager fiel auf die Skriptsprache Perl. Sie wird durchgehend in allen selbst entwickelten Bestandteilen des Systems verwendet und ist unter einer dualen freien Lizenz verfügbar. Der Benutzer kann wählen, ob er Perl unter der Gnu General Public License oder der Perl-eigenen Artistic License verwenden möchte.

Es wurde Perl gewählt, da es einen ausgewogenen Mittelweg zwischen einfachen Skriptsprachen, wie z.B. der Unix Shell und Hochsprachen, wie Java oder C darstellt und für die in der Mathematik vorhandenen Plattformen Linux und Windows XP verfügbar ist.

Die Sprache erlaubt sowohl klassische prozedurale Programmierung, als auch die Nutzung von objektorientierten Programmiermodellen bzw. eine Mischung von diesen. Sie enthält eine sehr mächtige Regular-Expression-Semantik und leicht nutzbare Datenstrukturen, wie klassische bzw. assoziative Arrays (Listen bzw. Hashes).

Die Codestrukturierung erfolgt in Perl durch sogenannte *Module*. Diese kapseln meist eine Funktionalität und sind im Großen und Ganzen mit z.B. Java-Klassen vergleichbar.

Perl Erfinder Larry Wall über Version 1.000 der Sprache[1]:

> [...] It's also a good language for many system management tasks. The language is intended to be practical (easy to use, efficient, complete) rather than beautiful (tiny, elegant, minimal). It combines (in the author's opinion, anyway) some of the best features of C, sed, awk, and sh, so people familiar with those languages should have little difficulty with it. [...]

Aufgrund der langjährigen Entwicklung der Sprache und ihrer Nutzung für administrative Zwecke, dynamische Webauftritte und andere vernetzte Systeme steht eine Vielzahl von Bibliotheken für diese und andere Anwendungsfälle bereit.

Da Perl unter einer freien Lizenz entwickelt wird, stellen viele Entwickler selbst entwickelte Module ebenfalls als freie Software gemäß der Perl Lizenz zur Verfügung. Das *Comprehensive Perl Archive Network*[2] ist seit 1995 die zentrale Anlaufstelle für Perlinteressierte. Neben dem Perl Sourcecode findet man im CPAN eine ständig wachsende Anzahl an Perl-Modulen, die dort von ihren Autoren veröffentlicht wurden. Am 30.08.2008 umfasste es laut eigener Statistik 14.236 Module von 6.813 Autoren.

Die oben aufgeführten Spracheigenschaften böten selbstverständlich auch Sprachen wie Python, Ruby oder Pike. Die Wahl fiel auf Perl, da ich mit dem Umgang dieser Sprache und ihren Möglichkeiten bereits einige Erfahrung gesammelt habe. Dies trifft auf die genannten Alternativen nicht zu. Daher erschien es aufgrund der Mächtigkeit und meiner Erfahrung sinnvoll, auf Perl zurückzugreifen, da sich die Erfahrung sicherlich auch in der Qualität des Codes und der Fehleranfälligkeit des WakeUpManagers niederschlagen wird.

[1] http://www.isc.org/sources/devel/lang/perl.txt
[2] http://cpan.org/

Modulnamen bestehen in Perl aus einzelnen Namen oder Begriffen, die durch "`::`" voneinander getrennt sind. Alle als Perlmodul implementierten Komponenten des WakeUpManagers sind als `WakeUpManager::*` benannt, wobei "`*`" einen Platzhalter für beliebige weitere Namen oder Begriffe darstellt. Durch die Trennung einzelner Ausdrücke kann die Codestruktur im Namen der Module veranschaulicht werden. So ist im Modul `WakeUpManager::Agent::Daemon` der WakeUpManager Agent implementiert, der den RPC-Dienst für den zentralen Server bereitstellt und in `WakeUpManager::Agent::Client` eine Bibliothek, die als API zum Zugriff auf den Agenten dient.
Sämtliche Teile der Webschnittstelle sind als `WakeUpManager::WWW::*` benannt.

6.2 Datenzugriff via DBI

Das *Database independent interface for Perl* (DBI) ist die Standard-Datenbank-Schnittstelle für Perlprogramme. Es stellt eine für den Programmierer einheitliche Schnittstelle zu einer Vielzahl von relationalen Datenbanksystemen (RDBMS) da. Die Eigenheiten eines jeden unterstützten Datenbanksystems werden innerhalb der DBI-eigenen Treibermodule (*Database Driver*, DBD) abgedeckt und werden somit für den Anwender transparent. Der einzige Unterschied zwischen dem Zugriff auf eine MySQL- oder PostgreSQL-Datenbank — um nur zwei Beispiele zu nennen — ist die einmalige Angabe, welches DBI-Treibermodul genutzt werden soll.
Möchte man im Nachhinein zu einem anderen Datenbanksystem wechseln, so müssen nur die Parameter eines DBI-Aufrufs im Programmcode geändert werden; sämtliche Anfragen an die Datenbank können beibehalten werden.

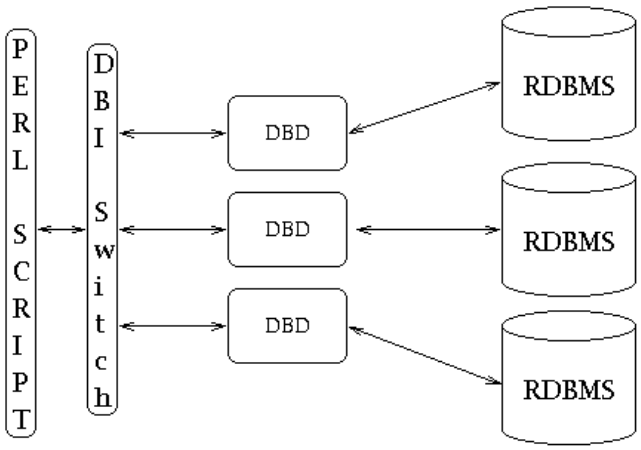

Abbildung 6.1: Perl DBI Architektur

Somit ist die Nutzung von PostgreSQL als Datenbankanwendung keinesfalls zwingend vorgegeben. Möchte man den WakeUpManager in Verbindung mit z.B. MySQL oder Oracle zur Datenhaltung nutzen, so muss nur das Datenbankschema für das gewählte RDBMS angepasst und eine Zeile in der WakeUpManager Konfigurationsdatei verändert werden.

Das Perl-Modul `WakeUpManager::DB::HostDB` stellt eine Programmierschnittstelle für alle Komponenten des WakeUpManagers zur in Kapitel 4 beschriebenen Datenbank bereit. Diese Schnittstelle bietet eine weitere Ebene der Abstraktion und erlaubt es, Änderungen am Datenmodell durch Anpassung nur dieses Moduls abzubilden.

6.3 Interprozesskommunikation

Aus der Verteiltheit des WakeUpManagers geht die Notwendigkeit hervor, einen Kanal zur Kommunikation zwischen den einzelnen Komponenten bereitzustellen, um Aktionen auf anderen Teilen des Systems auslösen zu können. Aufgrund der Tatsache, dass ein zentraler Server existiert, der alle Vorgänge des Systems steuert, ist ein klares Kommunikationsschema gegeben. Es reicht aus, Kommunikationskanäle dahingehend unidirektional zu gestalten, dass eine sendende Instanz eine Anfrage an einen Dienst schicken kann, es jedoch möglich ist, auf diesem Kanal eine Antwort auf diese Anfrage zurückzusenden.

Grundsätzlich kann ein IP-Netzwerk als Kommunikationsmedium als vorhanden angenommen werden. Es stellt sich jedoch die Frage, welches Übertragungprotokoll verwendet werden soll. Es ist möglich, ein eigenes Protokoll auf Basis von TCP/IP zu entwerfen und zu implementieren oder die Kommunikation über ein bestehendes Protokoll bzw. Framework abzubilden. Aus zeitlichen Gründen und zur Vermeidung unnötiger Fehlerquellen schied die Entwicklung eines eigenen Netzwerkprotokolls aus. Es musste ein adäquates Framework gefunden werden, für das bereits eine dokumentierte und funktionierende Implementation in der Sprache Perl existiert.

Die oben beschriebene Semantik der Interprozesskommunikation entspricht der klassischen Technologie des entfernten Prozeduraufrufs (*Remote Procedure Calls*, RPC). Daher erscheint es sinnvoll, ein Framework bzw. eine Bibliothek zu suchen, die diese Technologie in Perl implementiert.

6.3.1 XML-RPC

Zur Übertragung von Daten über ein Netzwerk ist es notwendig, dass sowohl Sender als auch Empfänger das Format der übertragenen Daten kennen und sich an dieses Proto-

koll halten. Der Großteil der neueren RPC-Implementation nutzt daher die *Extensible Markup Language* (XML) zur Darstellung der Aufrufe und Antworten. Durch die Nutzung eines standardisierten Formats wie XML kann sichergestellt werden, dass die sog. XML-RPC-Technologie auf möglichst vielen Systemplattformen genutzt werden kann, da XML-Bibliotheken für alle gängigen Betriebsysteme und Programmiersprachen zur Verfügung stehen.

Das wohl bekannteste Framework, welches eine XML-RPC-Schnittstelle implementiert, ist das *Simple Object Access Protocol*, (SOAP), für das eine umfassende Perl-Implementation bereitsteht. Es stellt jedoch eine unnötig große Abhängigkeit in Bezug auf Codeumfang und Ressourcenbedarf dar und ist für die Anforderungen des WakeUpManagers bereits zu mächtig. Aufgrund der Mächtigkeit wäre auch mit einer erheblichen Einarbeitungszeit zu rechnen, die für diese Arbeit nicht zur Verfügung stand. Eine "schlankere" Softwarekomponente für diesen Zweck wäre wünschenswert.

Neben SOAP, das bereits als Nachfolger der XML-RPC-Technologie betrachtet wird, existieren eine Reihe von kleineren Modulen, die sich auf die Implementation der Basisfunktionen der XML-RPC-Definition beschränken. Diese beinhaltet nur die "Generierung und Analyse der Aufruf- und Antwortpakete, (sowie) Übertragung und Empfang der selben und (die) Darstellung der Datentypen"(vgl. [XML08]).

6.3.2 Frontier::RPC

Eine solche Implementation der XML-RPC-Basisfunktionen ist die `Frontier::RPC` Distribution[3]. Da sie nur aus vier überschaubaren Modulen besteht und sehr leicht nutzbar ist, fiel die Wahl der RPC-Bibliothek auf diese Distribution.

Reine XML-RPC-Implementationen bieten den Nachteil, dass nur einfache Werte, wie

[3] http://search.cpan.org/~kmacleod/Frontier-RPC-0.07b4/

Ganz- und Fließkommazahlen, Strings oder Wahrheitswerte übertragen werden können. Die Definition der Technologie sieht nicht vor, dass sprachbezogene Daten, wie etwa ein Perl-Objekt (bei Nutzung der objektorientierten Ansätze), übertragen werden können. Durch die allgemein übliche Nutzung des *Hypertext Transfer Protocols* (HTTP) als zugrundeliegendes Netzwerkübertragungsprotokoll ist auch eine statusbehaftete Kommunikation ausgeschlossen, d.h. der Server kann sich keinen Status "merken". Das letztgenannte Problem wäre auch bei der Nutzung von SOAP gegeben.

Um größere Datenstrukturen wie etwas verschachtelte Perl-Hashes als Parameter oder Ergebniswerte von XML-RPC-Aufrufen übertragen zu können, wurde die Frontier::RPC Distribution selbstständig um ein Marshalling bzw. Demarshalling für diese Daten erweitert. Als Marshalling bezeichnet man den Prozess eine Menge von Daten in ein Format umzuwandeln, dass in einer Nachricht über das Netzwerk übertragen werden kann (vgl. [GC05]). Zusätzlich wurde das `Frontier::Daemon` Modul, das einen eigenen Webservice als Netzwerkdienst bereitstellt, dahingehend erweitert, dass über einen neuen Parameter eine Zugriffsteuerung auf Basis von IP-Adressen der anfragenden Klienten möglich ist, sowie alle (erlaubten und abgelehnten) Zugriffe im Syslog protokolliert werden.

Sämtliche Erweiterungen wurde so implementiert, dass alle neuen Funktionen explizit über einen Parameter aktiviert werden müssen, sodass die veränderte Version der Bibliothek zu 100% zur ursprünglichen Version kompatibel ist.

Das selbstentwickelte Perlmodul `WakeUpManager::RPC::Wrapper` stellt ein Schnittstelle zwischen Komponenten, die als RPC-Dienste verfügbar gemacht werden sollen und der Frontier::RPC-Distribution bereit. Eigene Module, die RPC-Dienste anbieten sollen sind somit nicht an implementative Details der verwendeten RPC-Transportschicht gebunden, sodass im Nachhinein mit wenig Aufwand auf eine andere Bibliothek, wie z.B. SOAP, zurückgegriffen werden kann.

Zur Vereinheitlichung und Vereinfachung der Fehlerdiagnose werden sämtliche via RPC übertragenen Anfrageergebnisse als Parameter eines `WakeUpManager::RPC::Result`-Objektes versand. Da die Erstellung des Objekts vom `WakeUpManager::RPC::Wrapper`-Modul übernommen wird, geschieht dies für die aufgerufene Methode auf Seite des RPC-Servers transparent. Der RPC-Client muss jedoch explizit so programmiert werden, dass ein `WakeUpManager::RPC::Result`-Objekt als Ergebnis erwartet wird. Diese Entscheidung ist darin begründet, dass RPC-Aufrufe aus Sicht des Clients stets anders behandelt werden müssen als lokale Aufrufe, da bei einem entfernten Aufruf andere Fehlerquellen beachtet werden müssen.

Durch den Einsatz der oben genannten Module sind alle Komponenten des WakeUp-Managers, die RPC-Dienste anbieten oder nutzen, unabhängig von der genutzten RPC-Implementation. Die Implementation der Module, die RPC-Dienste anbieten, ist bis auf die Definition der per RPC verfügbaren Methoden transparent.

6.4 Grundlagen der Webanwendungen

Alle Benutzerschnittstellen das WakeUpManagers werden als Webdienste realisiert.
Der Apache Webserver in Version 2.2.3, wie er in der stabilen Version der Debian/GNU Linux Distribution (Codename "Etch") enthalten ist, stellt sie bereit. Er ist als Freie Software verfügbar und bildet aufgrund seiner langen Entwicklunghistorie eine stabile und skalierfähige Grundlage für alle Webanwendungen des WakeUpManager Servers. Der Apache Webserver kann durch eine Vielzahl von Erweiterungsmodulen um zusätzlichen Funktionen erweitert werden. Für den WakeUpManager kommen Module zur Authentifikation von Benutzern (`auth_kerb`) und zur Beschleunigung von dynamischen Webanwendungen (`mod_perl`) zum Einsatz.

6.4.1 Benutzerauthentifikation

Zur Authentifikation der Benutzer gegenüber den angebotenen Webdiensten wird das ebenfalls frei verfügbare Apache Erweiterungsmodul `auth_kerb`[4] genutzt. Es authentifiziert die Benutzer gegen einen Kerberosdienst und stellt den Namen des Benutzers für dynamische Webanwendungen bereit. Neben der klassischen Anmeldung durch Benutzername und Passwort wird auch eine Identitätsprüfung über ein sog. *Kerberosticket* unterstützt. Aufgrund der durchgehenden Nutzung der Kerberosdienste im Rechnerverbund der Mathematik erhält jeder Benutzer automatisch ein solches Kerberosticket, wenn er sich erfolgreich an einem Clientrechner authentifiziert hat. Somit gliedert sich der WakeUpManager nahtlos in die bestehende Single-Sign-On-Umgebung ein. Die Funktionsweise von Kerberos, sowie die hier genutzten Begriffe werden in [Ric07] erklärt.

Unterstützt der verwendete Webbrowser ticketbasierte Authentifikation so erfolgt die Anmeldung automatisch und vollständig transparent für den Nutzer. Alle gängigen Browser

[4] http://modauthkerb.sourceforge.net/

wie z.B. “Mozilla“, “Firefox“ oder “Konqueror“ erlauben die automatische Anmeldung. Sie stehen auf allen Linuxarbeitsplätzen zur Verfügung; auf Windows-PCs ist nur der “Firefox“ installiert.

Die in den Webserver integrierte Kerberosauthentifikation steigert auch die Mandantenfähigkeit des Systems, da sämtliche großen Institute und Einrichtungen der Universität Paderborn Kerberos zur Authentifikation der Benutzer verwenden und zusätzlich jeder Benutzername global eindeutig ist.

Jede Kerberosinstallation wird durch einen eindeutigen Namen (den sog. *REALM*) beschrieben. Durch die Einrichtung von Vertrauensbeziehungen *REALM-Trusts* zwischen den verschiedenen REALMs (z.B. dem Kerberossystem des IMT und der Mathematik) und der global eindeutigen Benutzernamen ist es durch die Authentifikation mit Kerberostickets möglich, dass Benutzer des IMT den WakeUpManager-Dienst der Mathematik nutzen können (oder umgekehrt).

Alternativ zur Nutzung eines Apachemoduls wäre es möglich gewesen, die Authentifikation in das WakeUpManager Web-Interface zu integrieren. Dies hätte bedeutet, die Zugangsdaten des Benutzers über ein HTML-Formular zu erfragen und die eingegebenen Daten selbst gegen den Kerberosdienst zu validieren. Diese Methode würde jedoch einige Unterstützung innerhalb der Webapplikation, wie die Verwaltung von Sitzungen (*Sessions*), die Anbindung an die Kerberosdatenbank, sowie weitere Sicherheitsvorkehrungen erfordern und somit die Komplexität des Codes erhöhen. Eine transparente Authentifikation des Benutzers wäre ebenfalls nicht möglich, sodass bei jeder neuen Sitzung eine Anmeldung durch die manuelle Eingabe der Benutzerdaten erfolgen müsste.

Daher wurde zur besseren und transparenten Integration des Systems in das Netzwerk der Mathematik und zur Verringerung der Codekomplexität — und somit besseren Wartbarkeit — auf das oben genannte Apachemodul zurückgegriffen.

6.4.2 CGI

Zur Implementation der dynamischen Webanwendungen, wie dem WakeUpManager Web-Interface und den Webdiensten für `wum_boot` und `wum_shutdown`, kommt die *Common Gateway Interface* (*CGI*) Technologie zum Einsatz. Sie bildet die Schnittstelle zwischen Webserver und Anwendungen, über die Aufrufparameter und Rückgabewerte übertragen werden.
Die folgendenen Aufrufparameter werden z.B. als Umgebungsvariablen zur Verfügung gestellt:

- `$REMOTE_USER`

  Wenn bereits eine Benutzerauthentifikation auf Ebene des Webservers stattgefunden hat, enthält diese Variable den Namen des angemeldeten Anwenders.

- `$HTTP_ACCEPT_LANGUAGE`

  Diese Variable enthält eine nach Priorität geordnete Liste mit Sprachen, die der Benutzer in seinem Webbrowser als Präferenz konfiguriert hat.

- `$REQUEST_METHOD`

  Die `REQUEST_METHOD` Variable gibt an, mit welcher Methode die dynamische Webseite abgefragt wurde. In den Anwendungsfällen, die innerhalb des WakeUpManagers auftreten, ist dies entweder `GET` oder `POST`.

- `$QUERY_STRING`

  Der `$QUERY_STRING` einer Anfrage enthält die Parameter, die in einer URL[5] enthalten sind.

Das Perlmodul `CGI` (vgl. [LDS]) liefert ein API für die Nutzung der CGI-Technologie durch in der Sprache Perl implementierte Webanwendungen. Es stellt Funktionen[6] für den Zugriff auf die oben genannten Variablen bereit und verbindet den Ausgabekanal der

[5] Uniform Resource Locator
[6] Siehe [LDS], Abschnitt *FETCHING_ENVIRONMENT_VARIABLES*

Anwendungen mit dem Ausgabekanal des Webservers. Dies bedeutet praktisch, dass die Ausgabe, die ein `print`-Befehl innerhalb der Webanwendung erzeugt, im Webbrowser des Benutzers angezeigt wird.

Um innerhalb der WakeUpManager Webanwendungen eine einheitliche Schnittstelle für alle Ein- und Ausgaben zur Verfügung zu stellen, wurde das Modul `WakeUpManager::WWW` entwickelt. Es erzeugt ein `CGI`-Objekt, fragt alle relevanten Parameter von diesem ab, speichert die Werte in einer internen Repräsentation und stellt sie anderen Modulen durch entsprechende Funktionen zur Verfügung. Durch diese Abstraktionsschicht zwischen externer und interner Implementation ist es möglich, sämtliche interen Module unabhängig von der verwendeten CGI-Implementation zu programmieren, sodass die externe Anbindung an den Webserver mit geringfügigem Aufwand ausgetauscht werden kann.

6.4.3 Performance

Der zentrale WakeUpManager Server wird regelmäßig — standardmäßig alle 15 Minuten — von jedem laufenden Clienten, auf dem alle lokalen Überprüfungen abgeschlossen wurden (vgl. Kapitel 3.3), nach dem jeweiligen Zeitplan befragt. Damit dies auch bei einer großen Anzahl von Rechnern keine Auswirkungen auf die Antwortzeiten des WakeUpManager Servers, insbesondere auf das dort ebenfalls betriebene Web-Interface hat, ist es notwendig, dass der Webserver mehrere Anfragen schnell und zuverässig parallel bearbeiten kann. Dies ist beim Apache Webserver sicherlich der Fall.

Neben den Einstellungen des Webservers spielt die Ausführungszeit der Webanwendung und -dienste eine entscheidene Rolle bei der Performance des Gesamtsystems. Die CGI-Technologie startet standardmäßig für jede neue Anfrage eine eigene Interpreterinstanz. Dies kostet bei jeder Anfrage Zeit und bietet Optimierungspotential.

Das Apache Erweiterungsmodul `mod_perl` ist dazu konzipiert, diese aus dem Design des *Common Gateway Inteface* resultierende Problematik zu lösen. Es kommt auf dem zentralen WakeUpManager Server zum Einsatz und läd pro Instanz (Thread) des Webservers einen Perlinterpreter, kompiliert alle benötigten Module einmal und hält die kompilierte Version des Applikationscodes im Speicher. Durch diese Technik kann eine signifikante Verringerung des Antwortzeiten des Websystems erreicht werden.

Im "Why mod_perl?"-Artikel (vgl. [Bek02]), der auf der Perl.com Webseite veröffentlich wurde, heißt es:

> There are big savings in startup and compilation times. Having the Perl interpreter embedded in the server saves the very considerable overhead of starting an external interpreter for any HTTP request that needs to run Perl code. At least as important is code caching: the modules and scripts are loaded and compiled only once, when the server is first started. Then for the rest of the server's life the scripts are served from the cache, so the server only has to run the pre-compiled code. In many cases, this is as fast as running compiled C programs.

6.5 WakeUpManager Web-Interface

Die folgenden Abschnitte behandeln die Implementation des Web-Interfaces. Sie stellen die Softwarearchitektur vor und diskutieren implementative Details zu den bereits in Kapitel 4.1 vorgestellten Elementen des Benutzerinterfaces.

6.5.1 Architektur

Das Web-Interface des WakeUpManagers ist eine dynamische Webanwendung, die in der Sprache Perl entwickelt wurde und — wie bereits in Kapitel 6.4.2 beschrieben — die *Common Gateway Interface* Technologie nutzt, um die Anwendung in den Webserver zu integrieren. Nahezu alle Komponenten der Webapplikation sind in objektorientiertem Perl programmiert, sodass im folgenden die Begriffe Klasse und Modul synonym sind. Die Begriffe Klasse und Objekt sind insofern auch synonym, dass stets der Programmablauf einer Objektinstanz gemeint ist.

Den Einstiegspunkt in die WakeUpManager Webapplikation stellt das Perlscript `index.pl` dar, das innerhalb des Dokumentenverzeichnisses des Webservers liegt und bei einer Anfrage eines Webbrowsers durch diesen ausgeführt wird. Es erzeugt ein Objekt der Klasse `WakeUpManager::WWW` und ruft die Funktion `get_page()` dieses Objektes auf. Der Rückgabewert dieses Aufrufs, eine komplette HTML-Seite als String, wird direkt ausgegeben, sodass die Seite im Browser des Benutzers angezeigt wird.

Das Modul `WakeUpManager::WWW` (im folgenden `WWW`) stellt eine Abstraktionsebene des `CGI`-Moduls dar und bildet die Schnittstelle zu allen internen Komponenten der WakeUpManager Webapplikation. Wird ein Objekt der Klasse `WWW` instanziiert, so erzeugt es jeweils ein Objekt der Klassen `CGI` und `WakeUpManager::WWW::Params` (im folgenden `Params`). Sämtliche Umgebungsvariablen und Anfrageparameter werden über das `CGI`-Objekt abgefragt und im `Params`-Objekt gespeichert.

Das WWW-Modul erzeugt ebenfalls ein Objekt der Klasse WakeUpManager::WWW::Page (Page), das eine Referenz auf das Params-Objekt übergeben bekommt. Die get_page()-Methode des WWW-Moduls leitet den Aufruf an die gleichnamige Funktion der Klasse Page weiter und gibt das Ergebnis dieses Aufrufs unverändert an das index.pl-Skript zurück.

Das Page-Objekt liest die Konfigurationsdatei des WakeUpManagers ein, entnimmt daraus die Parameter für das DBI-Modul (vgl. Kapitel 6.2) und stellt eine Verbindung zur WakeUpManager Datenbank her. Zur Generierung der Webseite wird ebenfalls ein Objekt der Templating-Klasse HTML::Template erzeugt, welches das Grundlayout der WakeUpManager Webseite enthält (vgl. Kapitel 4.1.2 und Abbildung 4.1). Die Funktionsweise des Template-Moduls wird im folgenden Kapitel genauer beschrieben.

Nachdem überprüft wurde, welche Berechtigungen für den Benutzer in der Datenbank vermerkt sind, generiert das Page-Modul ein Menü, in dem nur die Funktionen enthalten sind, für die der Nutzer eine Berechtigung an mindestens einem Rechner besitzt. Es folgt die Verifikation, ob die angeforderte Seite existiert und eine entsprechende Fehlermeldung bei einer ungültigen Anfrage. Existiert die Seite, wird das zugehörige Logikmodul WakeUpManager::WWW::Page::[Seitenname] geladen und instanziiert, wobei die Referenz auf das Params-Objekt und die Datenbankverbindung übergeben werden.

Das Page-Objekt läd nun das Template für die angeforderte Seite in der gewünschten Sprache, fragt mögliche Headerelemente sowie die Elemente des Seiteninhalts vom Logikmodul ab und gibt diese an das Inhaltstemplate weiter. Das Inhatstemplate wird abgefragt und in den Inhaltsteil des globalen Templates eingefügt. Nun kann die get_page()-Funktion des Page-Moduls die erzeugte HTML-Seite zurückgeben, sodass sie im Browser des Benutzers angezeigt wird. Sämtliche auftretenden Fehlerfälle (Fehler im Logikmodul, Templatedatei existiert nicht, ...) werden abgefangen und durch eine entsprechende Meldung angezeigt.

Durch die Nutzung bereits existierender Komponenten, wie der Implementation des *Common Gateway Interfaces* oder des Templating-Moduls, war es möglich, den Fokus bei der Entwicklung der Webapplikation auf inhaltliche Teile, d.h. die Programmlogik, zu richten. Eine eigene Implementation dieser Komponenten hätte zuviel Zeit in Anspruch genommen und die Fehleranfälligkeit der Programms erhöht. Die beschriebene Kapselung der Funktionalitäten erlaubt den leichten Austausch einzelner Komponenten und trägt zu einer besseren Übersichtlichkeit des Programmcodes bei. Dies erhöht die Qualität und Sicherheit der Software und die Wartbarkeit der Anwendung.

Abbildung 6.2 zeigt ein UML-Klassendiagramm der beschriebenen Struktur inklusive aller relevanten Funktionen der Klassen. Funktionen, die in diesem Szenario nicht genutzt werden, wurden zur Übersichtlichkeit der Grafik nicht aufgenommen.

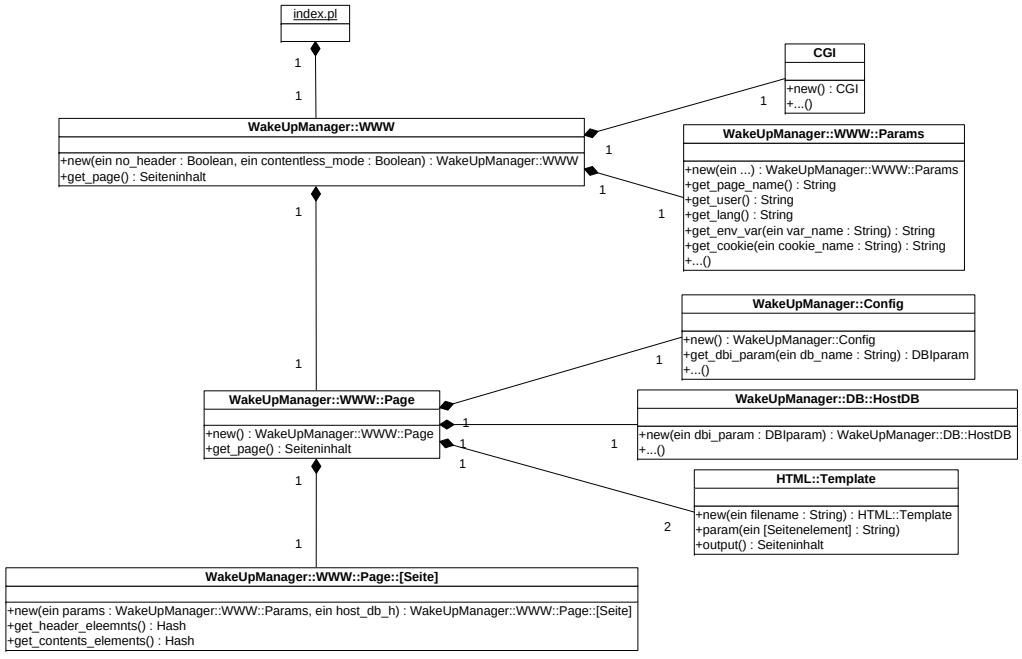

Abbildung 6.2: UML Klassendiagramm des WakeUpManager Web-Interfaces

6.5.2 HTML::Template

Die WakeUpManager Webanwendung greift zur Generierung des Seitenlayouts und der Inhalte auf eine Templating Bibliothek zurück. Dies ermöglicht eine strikte Trennung zwischen Applikationslogik und Präsentationsschicht und dient der Strukturierung der Programmcodes.

Als Templating-Bibliothek kommt das Perlmodul `HTML::Template` (vgl. [Tre]) zum Einsatz. Es ermöglicht das Ersetzen von Platzhaltern in entsprechend präparierten HTML-Seiten bzw. Seitenelementen. Ein Platzhalter kann entweder eine einfache Variable (`TMPL_VAR`) sein, die durch einen String ersetzt wird, oder eine `TMPL_LOOP`-Struktur, die für eine Liste von Argumenten wiederholt ausgeführt wird, um z.B. eine `<select>`-Liste mit Werten zu füllen. Es ist ebenfalls möglich, die Ausgabe von frei definierbaren Blöcken davon abhängig zu machen, ob eine frei wählbare Variable einen wahren Wert enthält. Somit können sehr leicht sämtliche Fehlerfälle mit Meldungen im Template versehen werden, der entsprechende Fehler wird jedoch nur angezeigt, wenn vom Programm eine bestimmte Variable gesetzt wurde.

Über die `TMPL_INCLUDE`-Anweisung innerhalb eines Templates, kann an dieser Stelle ein anderes Template hineingeladen werden, als ob der Inhalt des zu inkludierenden Templates an dieser Position stünde. Dies ermöglicht es, für Seitenfragmente, wie die mehrfach vorkommende Rechnerauswahl-Komponente, ein Template zu definieren und dieses innerhalb der einzelnen Seiten zu referenzieren. Dies vereinfacht spätere Änderungen an einer mehrfach genutzten Komponente.

Neben `HTML::Template` existieren weitere Templating-Bibliotheken, die nach einem ähnlichen Prinzip funktionieren, oder aber die Einbettung von Methodenaufrufen innerhalb der Templates vorsehen, um eine Webseite zu generieren. Da letzteres der Trennung zwischen Logik und Präsentationskomponenten widerspricht, wurden Module dieses Prinzips

ausgeschlossen. Das `HTML::Template`-Modul kommt zum Einsatz, da es eine einfache Syntax für die Einbettung von Variablen beseitzt, eine simple Schnittstelle zur Applikation bereitstellt und nur ein Modul umfasst.

Die Organisation der Templates auf der Festplatte wird im folgenden Abschnitt beschrieben.

6.5.3 Mehrsprachigkeit

Die Mehrsprachigkeit des WakeUpManager Web-Interfaces wird durch die Nutzung der Templating-Bibliothek unterstützt. Pro Sprache existiert ein Verzeichnis mit einem Satz von Templates, die einem definierten Namensschema folgen. Das Templateverzeichnis befindet sich zudem außerhalb der Verzeichnisstruktur, die durch den Webserver freigegeben ist. Dies verhindert den direkten Zugriff auf die Seitenbausteine.

Die Struktur des Templateverzeichnisse wird in Abbildung A.8 gezeigt. Der Pfad hält sich an den *Filesystem Hierarchy Standard* (vgl. [Fil04]), der in Kapitel 3.16 vorsieht, "site-specific data, which is served by this system" unter `/srv` abzulegen.

6.5.4 AJAX

Die Nutzung von AJAX innerhalb des WakeUpManager Web-Interfaces bedingt, dass auf dem Server eine Schnittstelle zur Verfügung steht, die die asychronen Anfragen bearbeiten kann. Dies geschieht beim WakeUpManager durch einige weitere Skripte ähnlich dem `index.pl`-Skript, das in Kapitel 6.5.1 beschrieben wurde. Diese Skripte erzeugen ein `WWW`-Objekt mit dem Parameter `ajax_mode`.

Durch die aktivierung des `ajax_mode` der `WWW`-Klasse wird anstatt des `Page`-Moduls ein Objekt der Klasse `WakeUpManager::WWW::AJAX` erzeugt. Dies erwartet die Angabe des

Namens einer Logikseite (`ajax_page_name`) sowie einer Funktion (`ajax_func_name`) und erstellt analog zur `Page`-Klasse eine Instanz der entsprechenden Seite.
Das `AJAX`-Modul implementiert ebenfalls eine `get_page()`-Funktion. Sie besteht aus den folgenden Schritten.

1. Erstellen eines `HTML::Template`-Objekts des `result_box` Templates in der entsprechenden Sprache
2. Aufruf von `ajax_call (ajax_func_name)` des Logikmoduls
3. Einbetten des Ergebnises dieses Aufrufs in das Template
4. Rückgabe des Templateinhalts

Die Funktion `ajax_call` wird von jedem Logikmodul implementiert und stellt den Zugriff auf interne Methoden des jeweiligen Moduls bereit, die die Bearbeitung von Benutzereingaben und Ereignissen übernehmen. Sämtliche Skripte zur Unterstützung der AJAX-Anfragen sind so konzipiert und umgesetzt.

6.6 RPC-Dienste

Sämtliche RPC-Dienste sind als sog. *Webservices* implementiert, d.h. sie werden über das HTTP(S)-Protokoll abgewickelt.

6.6.1 WakeUpManager Server

Die Webservices des zentralen WakeUpManager Servers nutzen analog zum Web-Interface die `WakeUpManager::WWW`-Bibliothek zur Realisation der jeweiligen Dienste. Sie stellt Funktionen bereit, mit denen der XML-Datenstrom der RPC-Anfrage an ein Modul aus der `Frontier::RPC`-Distribution weitergeleitet werden kann. Das Ergebnis eines Aufrufs wird wie der Inhalt einer Seite der Webanwendung über einen Kanal an den Webserver übergeben, sodass er zum RPC-Client zurückgesendet werden kann.

Die `content_less`-Option des `WakeUpManager::WWW`-Moduls erlaubt es zur Nutzung durch Webdienste, die automatische Instanziierung der `Page`-Klasse und aller folgenden Module zu unterbinden (im Gegensatz zur in Kapitel 6.5.1 beschrieben Architektur). Dadurch ist es möglich, die interne Schnittstelle und die Vorteile des `WWW`-Objektes nutzen können, jedoch direkten Einfluss auf die Ausgabe der jeweiligen Anwendung zu haben.

Diese Möglichkeit wird in den Webdiensten für die `wum_boot` und `wum_shutdown` Programme genutzt. Die Abbildungen A.4 und A.6 zeigen UML-Klassendiagramme der RPC-Dieste. Die Skripte `ui` bzw `client` stellen die Schnittstelle zwischen dem Webserver und der Anwendung dar.

6.6.2 WakeUpManager Agenten

Neben den beiden zuvorgenannten RPC-Diensten, bietet jeder Agent einen RPC-Dienst für den zentralen Server an. Dieser wird nicht über einen Apache Webserver realisiert,

da er für diesen Zweck eine zu große Abhängigkeit im Sinne der Größe der ausgeführten Anwendung darstellt.

Der `wakeup_agent` nutzt den Webservice des `Frontier::Daemon` Moduls, der den in Perl implementierten Webserver des `HTTP::Daemon`-Moduls benutzt. Abbildung A.7 zeigt ein UML-Klassendiagramm des WakeUpManager Agents.

Die Client-Schnittstelle zu den Agenten bilden die Module `WakeUpManager::Agent::Client` und `WakeUpManager::Agent::Connector`. Das `WakeUpManager::Agent::Client`-Modul stellt über ein Objekt der Klasse `Frontier::Client` eine RPC-Verbindung zu einem Agenten her und stellt über die Funktion `boot_pc (mac_addr, ip)` ein API zum Agenten bereit.

Damit nicht in jeder Komponenten des WakeUpManagers, die den Start von Clients bearbeitet, eine Abfrage aller benötigten Daten (zuständiger Agent, MAC, Netzwerk, ...) aus der Datenbank implementiert werden muss, wurde das `WakeUpManager::Agent::Connector`-Modul geschaffen, welches eine Funktion `boot_host (host_id)` bereitstellt. Diese fragt die benötigten Daten über eine bei der Instanziierung übergebene Datenbankverbindung ab und übergibt sie dem zuständigen Agenten.

6.7 Clientprogramme

6.7.1 wum_shutdown

Das Programm `wum_shutdown` ist auf jedem WakeUpManager Klienten installiert und wird regelmäßig durch den jeweiligen Zeitsteuerungsdienst des Betriebsystems aufgerufen. Dies geschieht in der Standardinstallation alle 15 Minuten.

Es prüft nach dem Start zuerst, ob der Rechner, auf dem es ausgeführt wird, bereits mindestens eine konfigurierbare Zeitspanne — standardmäßig 15 Minuten — läuft. Wurde der Rechner vor weniger als der eingestellten Zeit gestartet, so bricht das Programm ab, um dem Benutzer im Falle eines manuellen Starts aus der Ferne die Möglichkeit zu geben, sich mit dem PC verbinden.

Läuft der Client bereits länger als die konfigurierte Zeitspanne, so wird überprüft, ob ein Benutzer interaktiv eingeloggt ist oder ob auf dem Rechner Programme ausgeführt werden, die nicht in einer integrierten Liste aufgeführt sind. Ist ein Nutzer am Rechner angemeldet oder läuft ein unbekannter Prozess, bricht das Programm ab.
Die Liste der zu ignorierende Programme kann über eine Konfigurationsdatei erweitert oder überschrieben werden. Der `CLIENT` Abschnitt der in Abbildung A.2 gezeigten Konfigurationsdatei zeigt die in der Mathematik genutzten Einstellungen.

Wurde weder ein Benutzerlogin noch ein unbekannter Prozess gefunden, befragt `wum_shutdown` den WakeUpManager Server über eine XML-RPC-Schnittstelle, ob der Rechner heruntergefahren werden soll. Soll der Rechner durch den WakeUpManager heruntergefahren werden und sollte er laut Zeitplan inaktiv sein, wird der Client heruntergefahren. Anderenfalls wird das Programm beendet.

Bei Klienten, die mit dem Betriebsystem Windows XP betrieben werden, entfällt die

Überprüfung laufender Prozesse, da sich aufgrund der Einzelplatzlizenz der Client-PCs kein Benutzer aus der Ferne anmelden kann und daher ein Test auf eine lokale Sitzung ausreicht.

Abbildung A.5 zeigt ein UML-Klassendiagramm des Programms. In Abbildung A.6 ist der RPC-Dienst für `wum_shutdown` dargestellt.

6.7.2 wum_boot

Die implementativen Details des `wum_boot` Programms können aus der Beschreibung des Skripts in Kapitel 4.2 entnommen werden. Hier sei nur noch auf das UML-Klassendiagramm der Anwendung (Abbildung A.3) bzw. des RPC-Dienstes (Abbildung A.4) verwiesen.
Der `UI`-Abschnitt der in Abbildung A.2 gezeigten Konfigurationsdatei zeigt die in der Mathematik genutzten Einstellungen.

6.8 Sicherheit

Da der WakeUpManager personenbezogene Daten verarbeitet, ist es essentiell, bei der Implementation ein besonderes Augenmerk auf die Sicherheit der Software zu legen. Dies geschieht auf mehreren Ebenen.

Sichere Programme zeichnen sich auf oberster Ebene u.a. dadurch aus, dass sie übersichtlich strukturiert sind und auch von anderen Programmierern gelesen und verstanden werden können. Durch Vermeidung oder Nutzung bestimmter Konstrukte einer Sprache kann es passieren, dass der Compiler oder Interpreter der Sprache einige Fehler nicht bzw. leichter findet und somit ggf. präziser melden kann.

Zwei bekannte und wirkungsvolle Konstrukte der Sprache Perl sind die einfachen Anweisung `use strict` bzw. `use warnings`. Sie veranlassen den Perlinterpreter dazu, Definition und Nutzung von Varaiablen und Funktionen sowie deren Typen strikter zu prüfen und entsprechende Fehler auszugeben. Die Nutzung veralteter Schreibweisen wird ebenso angezeigt, wie der Zugriff auf undefinierte Variablen. Der Kommandozeilen-Parameter `-w` des Perlinterpreters ist nahezu äquivalent zur Nutzung von `use warnings` innerhalb des Programms, wirkt sich aber auf sämtlichen Code aus, während `use strict` und `use warnings` nur im jeweiligen Gültigkeitsbereich aktiv sind. Im Gegensatz zur Option `-w`, die vor Fehlern warnt, das Programm jedoch trotzdem ausführt, bricht die Ausführung bei Nutzung des Parameters `-W` nach Ausgabe der Fehlerliste ab. Ein fehlerfreies Programm wird ungehindert ausgeführt.

Eine weitere noch wirkungsvollere Maßnahme ist die Aktivierung des sogenannten "Taintmodes"[7]. Ist er aktiviert, so prüft Perl zur Compilezeit sämtliche Variablen darauf, ob sie Werte aus externen Quellen enthalten, die unzureichend bzw. gar nicht auf Validi-

[7] vgl. die über die Kommandozeile verfügbare Perleigene Dokumentation durch Aufruf von `perldoc perlsec`

tät geprüft wurden. Ein Programm, das beispielsweise einen Kommandozeilen-Parameter ungeprüft für einen Systemaufruf verwendet, kann sehr leicht für einen entsprechenden Angriff verwendet werden. Dies wird durch den Taintmode verhindert.
Ähnlich der Optionen für Warnungen zur Programmsyntax existieren zwei Perloptionen, die den Taintmode aktivieren. Mit `-t` gestartet meldet Perl Verstöße, führt das Programm aber aus, bei Angabe der Option `-T` wird die Ausführung im Fehlerfall verhindert.

Da Kommandozeilenparameter für Perl über den Shebang in der ersten Zeile eines jeden Perlmoduls angegeben werden können, bietet es sich an, Perl stets mit den Parametern `-WT` aufzurufen und auf dieser Sprachebene maximalen Schutz zu erreichen. Dies ist bei sämtlichen Komponenten des WakeUpManagers der Fall.

Das Testen der programmierten Anwendung im Allgemeinen und insbesondere die Angabe unerwarteter Eingabewerte ist ein weiterer Schritt auf dem Weg zu sicherer Software. Alle eigenständig erstellten bzw. erweiterten Komponenten wurden ausgiebig getestet. Das Web-Interface, das von allen für Benutzer zugänglichen Teilen des Systems die größte Angriffsfläche bieten dürfte, wurde besonders streng auf Sicherheitslücken untersucht. Die darin implementierten Sicherheitsmechanismen wurden ausgiebig getestet.

Neben den genannten Ebenen der Softwaresicherheit ist auch darauf zu achten, dass sämtliche Netzwerkkommunikation verschlüsselt wird. Zu diesem Zweck wurde das Module `mod_ssl` zum Apache Webserver hinzugefügt, sodass die Kommunikation über das *HTTPS*-Protokoll stattfinden kann. Alle Anfragen, die den Server über das unverschlüsselte *HTTP*-Protokoll erreichen, werden automatisch zum HTTPS-Dienst umgeleitet, sodass ein Benutzer unbewusst niemals Aktionen über eine unverschlüsselte Verbindung auslösen kann.

7 Organisatorische Einbindung

Dieses Kapitel beschreibt, welche Schritte notwendig waren bzw. sind, um den WakeUp-Manager im Netzwerk der Mathematik einzurichten und Pool- und Mitarbeiterrechner durch das System verwalten zu können.

7.1 Organistationseinheiten

Bevor Rechner in die WakeUpManager Datenbank aufgenommen werden können, müssen zuvor die entsprechenden Rechnergruppen sowie Netzwerke und Agenten in der Datenbank eingetragen werden. Dies muss durch die manuelle Eingabe von SQL-Kommandos geschehen, da diese Arbeit aus zeitlichen Gründen keine Schnittstelle zur Administration enthält.

Da das Datenbankschema in Kapitel 5 ausführlich beschrieben wurde und der Zusammenhang der Tabellen recht offensichtlich ist, sollte dies kein Hindernis darstellen.

Für das Netzwerk der Mathematik wurden initial eine “globale“ Gruppe “EIM-M“ sowie Gruppen für den Rechnerbetrieb und die Poolräume eingefügt. Im weiteren Verlauf folgte eine Gruppe pro angeschlossener Arbeitsgruppe, die unterhalb der “EIM-M“-Gruppe angesiedelt ist.

7.2 Netzwerktopologie

Um Rechner im Netzwerk der Mathematik durch den WakeUpManager starten zu können, müssen Agenten an strategischen Punkten im Netzwerk installiert werden, sodass mindestens ein Agent mit jedem Netzwerk verbunden ist, an das Clientrechners angeschlossen sind. Nach der Einrichtung von zwei Agenten in der Testphase des Systems im administrativen Netz der Mathematik wurden zwei weitere Agenten installiert.

Sie wurden als virtuelle Maschinen mittels der Xen-Technologie[1] realisiert und sind mit jeweils einer Netzwerkschnittstelle mit jedem Netzwerkstrang verbunden, der WakeUpManager Klienten enthält. Aus Sicherheitsgründen sind keine IP-Adressen auf diesen Schnittstellen konfiguriert, sodass die automatische Konfiguration des Agenten versagt. Die Zuordnung zwischen IP-Netz und ausgehender Schnittstelle (vgl. Kapitel 3.2) muss manuell über die Konfigurationsdatei erfolgen.

Die Schnittstellen besitzen keine IP-Adressen, da sonst das Risiko bestünde, dass ein Agent bei einer geringfügigen Fehlkonfiguration als Router zwischen den angeschlossenen Netzwerken fungieren könnte. Dies würde jedoch das Sicherheitskonzept der Mathematik negativ beeinflussen. Da zum Versand von Wake-on-LAN-Paketen keine IP-Konnektivität erforderlich ist, kann dies verhindert werden.

Neben der Konfiguration der Schnittstellen sollte eingestellt werden, welche Systeme dem Agenten Bootaufträge erteilen dürfen. Ein Beispiel einer Agent-Konfigurationsdatei mit manueller Angabe der angeschlossenen Netzwerk wird in Abbildung A.1 gezeigt.

[1] http://www.xen.org

7.3 Softwarepakete

Die Verteilung sämtlicher Software im Rechnerverbund der Mathematik erfolgt über die Bereitstellung und Installation von Softwarepaketen.

Im Linuxbereich werden die Pakete in einem zentralen Depot (*APT Repository*) gesammelt und über einen Webserver an die Klienten freigegeben. Ein in der Mathematik entwickeltes Softwareverteilungssystem gleicht regelmäßig die Liste der lokal installierten Pakete mit einer Datenbank ab und fügt der lokalen Softwareinstallation Pakete hinzu oder entfernt diese. Durch diese Technologie können die benötigten Komponenten des WakeUpManagers leicht auf allen Klienten installiert werden. Die WakeUpManager Distribution enthält alle benötigten Daten, um installationsfertige Debianpakete zu erstellen.

Auf den folgenden Systemkomponenten müssen die folgenden Pakete installiert werden:

- **WakeUpManager Server**

  Der Server benötigt die Pakete `wum-cron, wum-www`, sowie die Bibliotheken `wum-common` und `wum-db`. Sie stellen den Zeitsteuerungsdienst, sämtliche Komponenten des Web-Interfaces und der Webservices, sowie grundlegende Bibliotheken und die Datenbankschnittstelle bereit.

- **WakeUpManager Agent**

  Ein Agent benötigt das Paket `wum-agent` sowie die Bibliothek `wum-common` und das externe Paket `etherwake`.

- **WakeUpManager Klient**

  Auf jedem angeschlossenen Klienten muss das Paket `wum-utils` und die Bibliothek `wum-common` installiert werden. Es enthält die Programme `wum_shutdown`, das den jeweiligen Rechner herunterfährt, wenn der PC ungenutzt ist und der Zeitplan es vorsieht und `wum_boot`, welches es erlaubt, einen anderen PC über die Kommandozeile zu starten.

Neben den oben genannten existiert das Paket `wum-admin`, welches adminstrative Programme enthält, die es erlauben, einen Rechner ohne vorherige Prüfung der Berechtigungen zu starten, oder Berechtigungen für Benutzer in der Datenbank einzutragen. Diese Paket ist nur auf dem Administrationsserver der Mathematik installiert. Der für das Paket erforderliche Zugriff auf die WakeUpManager Datenbank ist nur von diesem System möglich, sodass die in diesem Paket enthaltenen Programme auf keinem anderen Rechner benutzerbar sind.

Alle Pakete können durch Aufruf des Programms `dpkg-buildpackage` aus dem Hauptverzeichnis der WakeUpManager Distribution erstellt werden, sofern alle benötigten Abhängigkeiten auf dem System installiert sind.

Im Windowsbereich existiert ein Paket namens "WakeUpManager für Windows", das mittels des Symantec LiveStateDelivery Systems auf Client-PCs verteilt wird. Es enthält eine für Windows angepasste Version des `wum_shutdown` Programms (vgl. Kapitel 6.7.1).
Da Windowsbenutzer erfahrungsgemäßg nicht zur Nutzung von Kommandozeilenprogramme neigen, wird `wum_boot` auf Windowsrechnern nicht installiert.

7.4 Konfiguration

Damit `wum_shutdown` feststellen kann, ob ein Rechner heruntergefahren werden darf, muss das Programm eine Liste mit Namen von Prozessen kennen, die auf dem Rechner ausgeführt werden dürfen, ohne diese Entscheidung zu beeinflussen. Diese Liste beinhaltet die Namen gängiger Hintergrunddienste, die auf nahezu jedem Linuxsystem zu finden sind, wie z.B. `cron, sshd, syslogd` sowie `udev` oder Displaymanager wie `gdm` oder `kdm`.

Da diese Prozesse zu jeder Zeit auf einem Linuxsystem gestartet sind, müssen sie ignoriert werden, wenn überprüft wird, ob ein Benutzer an einem Rechner angemeldet ist oder Anwendungen ausführt werden. Eine allgemeine Liste solcher Prozesse ist bereits in `wum_shutdown` enthalten, es ist jedoch möglich und teilweise notwendig, diese Liste durch Einträge in der WakeUpManager Konfigurationsdatei zu erweitern bzw. zu überschreiben.

Abbildung A.2 zeigt die in der Mathematik genutzte Konfigurationsdatei. Der Abschnitt `UI` wird von `wum_boot` ausgewerten, der Abschnitt `CLIENT` von `wum_shutdown`.

7.5 Clientrechner und Benutzerrechte

Die Aufnahme der Pool- und Mitarbeiterrechner in das WakeUpManager System erfolgt sukzessiv. Der Betatest des Systems begann im Rechnerbetrieb der Mathematik und wurde nach einigen Wochen auf die Poolräume der Mathematik ausgeweitet. Die PCs der RBM und der Pools mussten in die WakeUpManager Datenbank eingefügt werden, sodass sie korrekt in der Hierarchie enthalten sind.

Die Vergabe der Rechte für Administratoren erfolgt einmalig global für alle Rechner der Mathematik (s. Vererbung der Rechner, Kapitel 5.5). Für alle Studierenden wurden Remote-Boot-Berechtigungen an der Gruppe aller Poolrechner vergeben.

Die Aufnahme zweier Arbeitsgruppen geschah rechnerweise, da für jeden Benutzer neben der Bootberechtigung für alle Mitarbeiterrechner volle Berechtigungen für den eigenen Arbeitsplatz PC eingetragen werden müssen. Es ist ebenfalls notwendig, die korrekte Funktion der beteiligten Soft- und Hardware zu testen.

Es wurde ein Aufruf in die Benutzerverwaltungssoftware der Mathematik integriert, die Bootberechtigungen für Studierende und Mitarbeiter direkt beim Anlegen einer Benutzeraccounts in der WakeUpManager Datenbank einträgt, sodass nur bei Mitarbeitern die Berechtigung für einen Arbeitsplatzrechner manuell eingetragen werden muss.

8 Resumé

In dieser Arbeit wurde ein System vorgestellt und erfolgreich implementiert, das es erlaubt, den rechnerinduzierten Energieverbrauch des Institutes für Mathematik der Universität Paderborn signifikant zu senken.

Nach erfolgter einmaliger Grundkonfiguration der WakeUpManager Kompomenten (s. Kapitel 7.1 - 7.4) können Client-Rechner und Berechtigungen in das System eingefügt werden (s. Kapitel 7.5). Ab diesem Zeitpunkt können Benutzer die Zeitsteuerung ihres Arbeitsplatzrechners eigenständig, d.h. ohne Hilfe eines Administrators, konfigurieren.
Da der WakeUpManager in die Benutzerverwaltung der Mathematik integriert wurde (vgl. Kapitel 7.5), ist nur beim Anlegen eines neuen Mitarbeiters das manuelle Eintragen der Berechtigungen für dessen Arbeitsplatz-PC durch einen Administrator erforderlich. Abgesehen davon verhält sich der WakeUpManager für die Systemverwalter des Instituts transparent.

Der Standardzeitplan der Rechner sieht vor, dass die PCs an Werktagen von 8:00 - 18:00 Uhr verfügbar sind. Er kann vom Benutzer mit frei definierbaren Zeiten an die persönlichen Bedürfnisse angepasst werden.

Diese Praxis hat sich bereits im Rechnerbetrieb sowie in zwei Arbeitsgruppen der Mathematik bewährt. Für 15 der 42 an das System angeschlossenen Mitarbeiterrechner existiert ein eigener Zeitplan, der durch den jeweiligen Benutzer selbstständig angelegt wurde. Die

"Urlaubs"-Schaltung des WakeUpManagers kommt bei 7 PCs zum Einsatz (s. Kapitel 3.1.1).

Eine Beeinträchtigung des Lehrbetriebes kann aufgrund der ausgebliebenen Rückmeldung durch Studierende oder Tutoren auch ausgeschlossen werden. Daher kann das System als praxistauglich eingestuft werden und wird im Laufe der folgenden Wochen in den verbleibenden Arbeitsgruppen der Mathematik eingerichtet.

Es zeigt sich, dass das System von den Benutzern akzeptiert und seine Potentiale genutzt werden!

In den zwei Monaten, in denen das System im Rechnerbetrieb, den Poolräumen und zwei Arbeitsgruppen evaluiert wurde, konnte der Energieverbrauch durch ungenutzte Rechner gesenkt werden. Zum Zeitpunkt dieser Arbeit werden insgesamt bereits über 90 Rechner durch den WakeUpManager verwaltet. Dies ist der Großteil der am Institut für Mathematik aufgestellten Klienten.

Zur Erledigung administrativer Aufgaben wird jeder Rechner einmal pro Nacht gestartet (vgl. Kapitel 2.3.2). Dies vermindert das Einsparpotential geringfügig.

Neben der Einsparung von Energie ist auch ein Abfall des Geräuschpegels innerhalb der Büros zu verzeichnen, in denen alle Rechner durch den WakeUpManager verwaltet werden. Dies ist ein ungeplanter, wenn auch angenehmer Nebeneffekt, der möglicherweise die Konzentrationsfähigkeit der Mitarbeiter und Studierenden steigert.

Der WakeUpManager erfüllt somit sämtliche funktionalen Anforderungen bis ins Details. Das geschaffene Berechtigungssystem beinhaltet alle geforderten Funktionen und ist restriktiv; das Prinzip der Datenvermeidung wird eingehalten.

Eine Lizenz für den WakeUpManager steht noch nicht fest. Das System als Frei Software zu verbreiten ist möglich und wird sehr wahrscheinlich erfolgen. Der WakeUpManager darf unabhängig von der gewählten Lizenz vom Institut für Mathematik der Universität Paderborn genutzt werden.

Der Wunsch nach einem mandantenfähigen System wurde berücksichtigt und soweit möglich umgesetzt. Durch das entworfene Rechtemodell ist es möglich, dass einem Systemverwalter eines Bereichs sämtliche Rechte an alle Rechnern dieses Bereichs erteilt werden können.
Zum praktischen Betrieb des WakeUpManagers als mandantenfähiges System sind eigene Schnittstellen zur Administration erforderlich, um Rechner und Gruppe ein- bzw. austragen oder Rechte verwalten zu können. Die Implementation dieser Schnittstellen hätte jedoch den Rahmen dieser Arbeit gesprengt und ist daher nicht erfolgt. Es wäre sinnvoll, diese Schnittstellen als RPC-Dienste zu konzipieren, sodass administrative Aufgaben automatisiert erfolgen können.

Die technischen Anforderungen an das System wurden ebenfalls zu 100% erfüllt.

Somit wurden alle Ziele, die im Rahmen dieser Studienarbeit umsetzbar waren, erreicht!

A Anhang

A.1 Konfigurationsdateien

A.1.1 wakeup_agent

```
#!/usr/bin/perl -WT

package WakeUpManager::Main::Config;

$config = {
   Agent => {
       # You may specify a comma separated list of IP address or networks in
       # CIDR notation from which RPC calls should be allowed to your agent.
       # If unset, every host will be allowed to start RPC calls on this agent.
       allowed_clients => [ '127.0.0.1', '131.234.101.0/24', '131.234.111.23' ],

       # Define network_to_device_map manually
       #network_to_device_map => 'scan',
       network_to_device_map => {
          '131.234.104.0/24' => [ 'eth104' ],
          '131.234.105.0/24' => [ 'eth105' ],
       # [ ... ]
       },
   }
};
```

Abbildung A.1: Beispiel einer Agentkonfigurationsdatei `/etc/wum/agent.conf`

A.1.2 wum_boot und wum_shutdown

```
#!/usr/bin/perl -WT

package WakeUpManager::Main::Config;

$config = {
    UI => {
        # URL of RPC server, only specify protocol and host!
        RPC_URL => "https://wum.math.uni-paderborn.de",

        # Try to autocomplete hostname if no FQDN was given?
        hostname_autocompletion => 1,
    },

    CLIENT => {
        # URL of RPC server, only specify protocal and host!
        RPC_URL => "https://wum.math.uni-paderborn.de",

        # Processes which should be ignored while searching for user processes or any other
        # process which would prevent 'wum_shutdown' from shutting down the machine.
        #
        # Provide a comma separated list of process names WITHOUT arguments.
        #
        ignore_processes => [ 'setup-theme.sh', 'xterm', 'dhclient', 'nxssh' ],

        # If set to yes, all processes defined in 'wum_shutdown' will be droped and
        # superseeded by the list specified in ignore_processes list above.
        #
        drop_default_ignore_processes => 0,

        # Don't shutdown by default. This has to be activated by hand!
        do_shutdown => 1,
    },
};
```

Abbildung A.2: Konfigurationsoptionen für `wum_boot` und `wum_shutdown`

A.2 UML-Klassendiagramme

Die im folgenden gezeigten UML-Klassendiagramme zeigen das Zusammenspiel der einzelnen Module und Komponenten des WakeUpManagers. Zur besseren Übersichtlichkeit wurde im jeweiligen Szenario ungenutzte Methoden der Module ausgelassen. Dies ist durch einen "...()"- Eintrag in der Methodenliste kenntlich gemacht.

In Tabelle A.1 sind alle in dieser Arbeit enthaltenen UML-Klassendiagramma aufgeführt.

Systemkomponente	**Abb.**	**Seite**
WakeUpManager Web-Interface	6.2	59
`wum_boot`	A.3	82
Webservice für `wum_boot`	A.4	83
`wum_shutdown`	A.5	84
Webservice für `wum_shutdown`	A.6	85
WakeUpManager Agent	A.7	86

Tabelle A.1: Übersicht der UML-Klassendiagramme

A.2.1 wum_boot

Abbildungen A.3 und A.4 zeigen den Aufbau von Client- und Serverseite des `wum_boot` Programms (vgl. Kapitel 4.2).

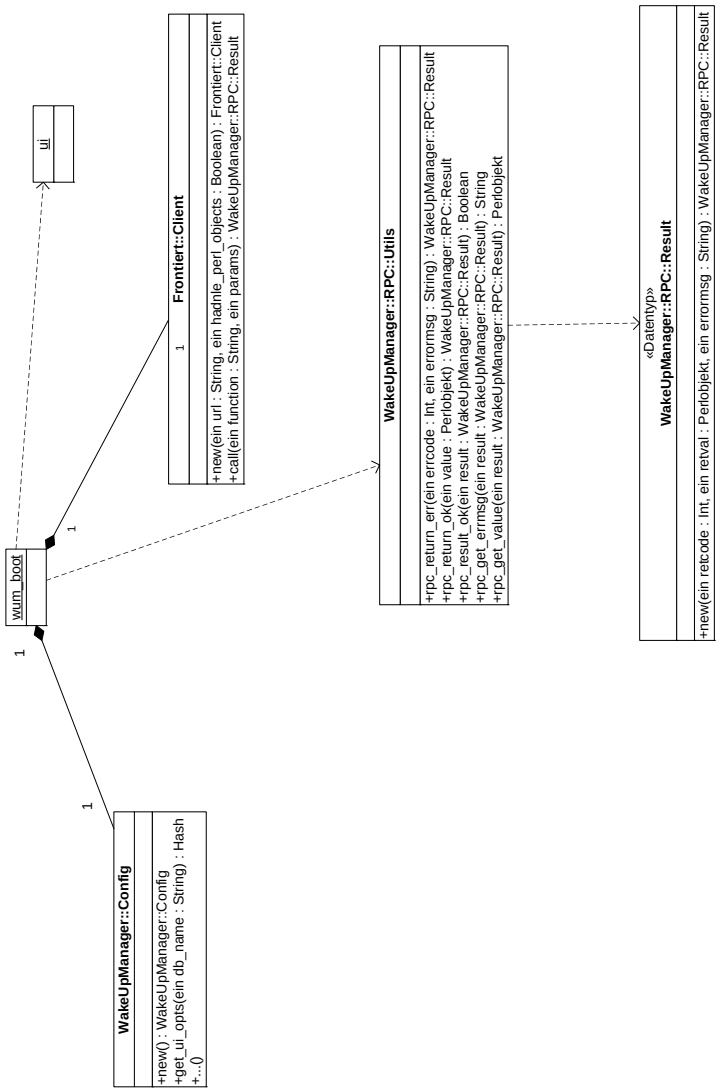

Abbildung A.3: UML-Klassendiagramm des `wum_boot`-Programms

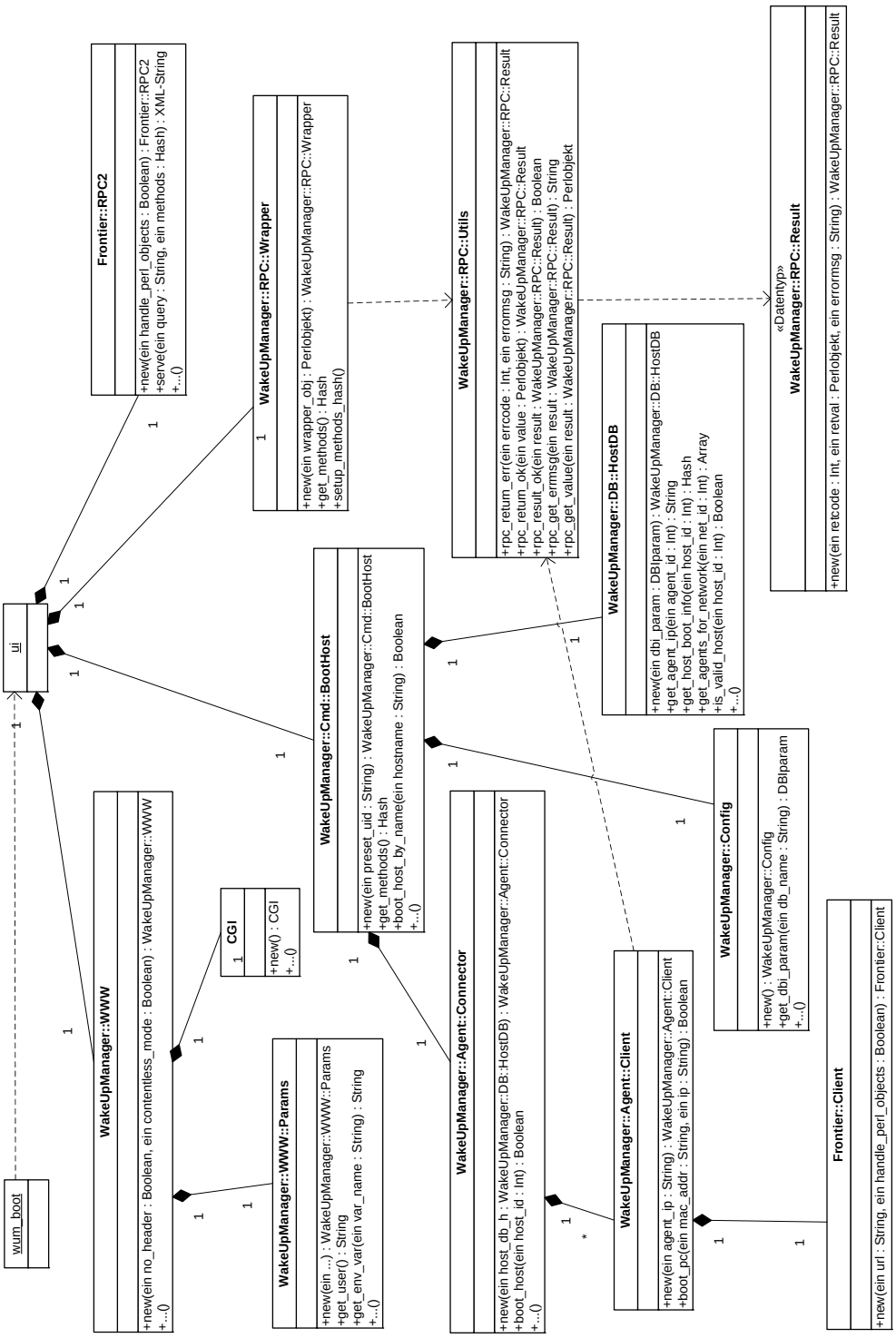

Abbildung A.4: UML-Klassendiagramm des Webdienstes für `wum_boot`

A.2.2 wum_shutdown

Die beiden Abbildungen A.5 und A.6 zeigen die Komponenten des `wum_shutdown` Programms, sowie des dazugehörigen Webservices (vgl. Kapitel 3.3, 6.7.1).

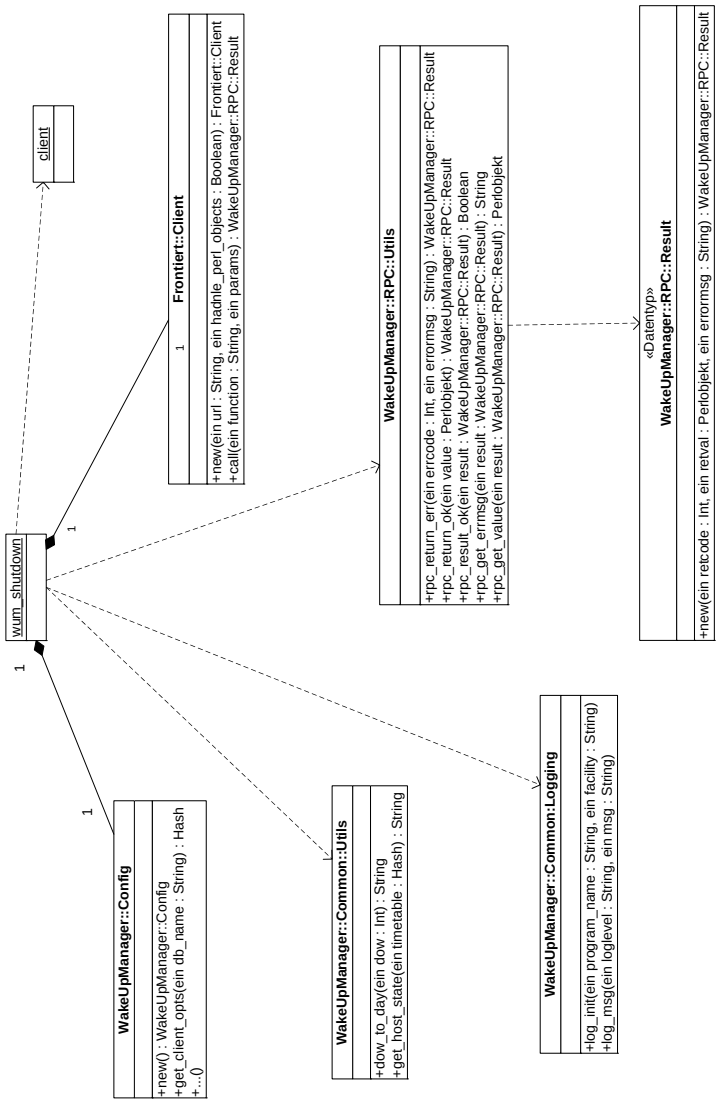

Abbildung A.5: UML-Klassendiagramm des `wum_shutdown`-Programms

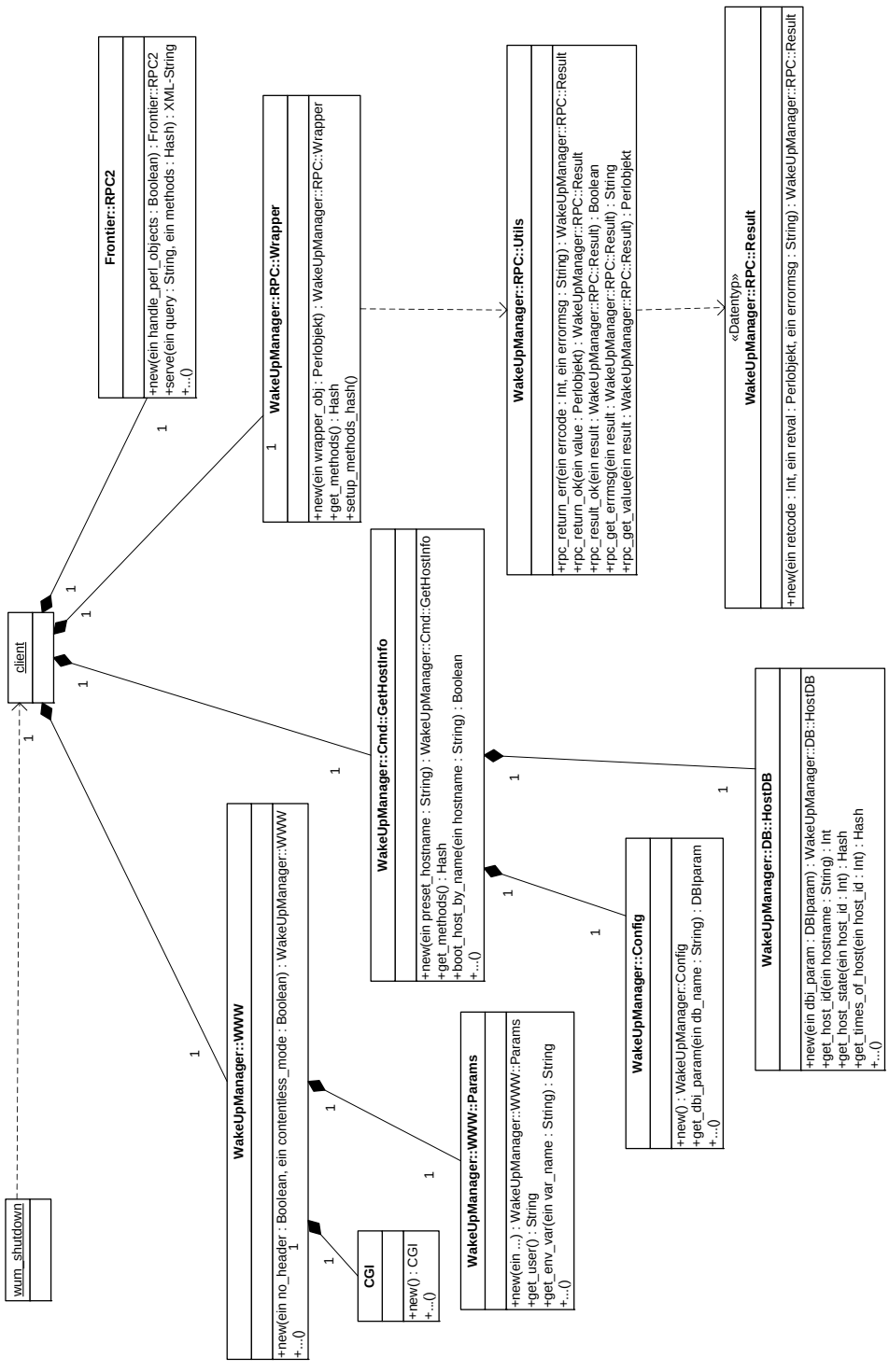

Abbildung A.6: UML-Klassendiagramm des Webdienstes für `wum_shutdown`

A.2.3 wakeup_agent

Abbildung A.7 zeigt den Aufbau des `wakeup_agent`-Dienstes (vgl. Kapitel 3.2).

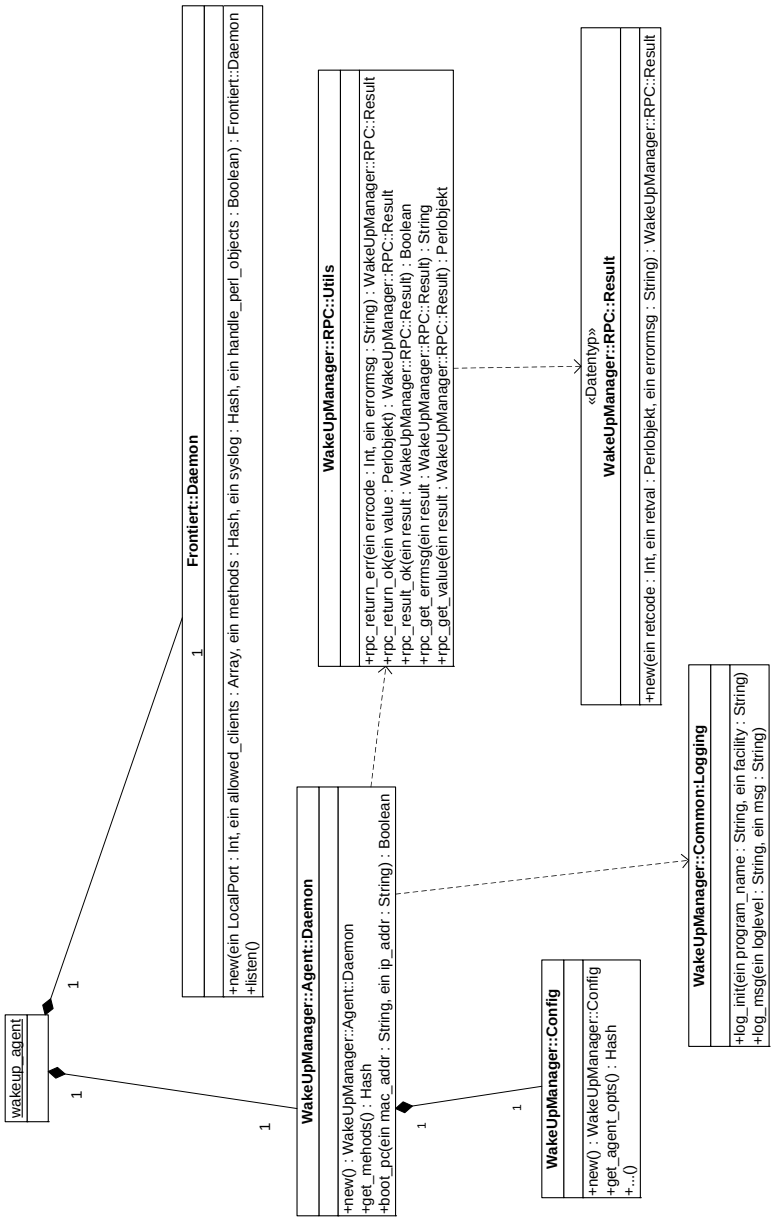

Abbildung A.7: UML-Klassendiagramm des `wakeup_agent`-Programms

A.3 Struktur des Templateverzeichnisses

```
/srv/wum/htdata/templates/
                          main.tmpl
                          en/
                             error.tmpl
                             host_select.tmpl
                             result_box.tmpl
                             BootHost.tmpl
                             ShowTimetable.tmpl
                             [...]
                          de/
                             error.tmpl
                             [...]
                             BootHost.tmpl
                             [...]
```

Abbildung A.8: Struktur des Templateverzeichnisses

Literaturverzeichnis

[AMD95] AMD: *Magic Packet Technology Whitepaper Rev. A*. http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/20213.pdf, 1995.

[Bek02] Bekman, Stas: *Why mod_perl?* . http://www.perl.com/pub/a/2002/02/26/whatismodperl.html, 2002.

[Bis08] Bishop, Tom: *Energieeffiziente Rechenzentren: Intelligente Software statt mehr Hardware*. LanLine (Mai/08), 2008.

[Fil04] Filesystem Hierarchy Standard Group: *Filesystem Hierarchy Standard*, 2004. http://www.pathname.com/fhs/pub/fhs-2.3.html.

[Fre07] Free Software Foundation: *GNU General Public License*, 2007. http://www.fsf.org/licensing/licenses/gpl.html.

[GC05] George Coulouris, Jean Dollimore, Tim Kindberg: *Distributed systems - Concepts and design*, Kapitel 4.3 External data representation and marshalling, Seite 144ff. Addison Wesley, Vierte Auflage, 2005.

[Gre08] Greiner, Dr. Wilhelm: *Green-IT-Gipfel von BMU, UBA und Bitkom: Klimaschutz als riesiger Markt*. LanLine (Mai/08), 2008.

[LDS] Lincoln D. Stein, et all: *CGI - Simple Common Gateway Interface Class*. http://search.cpan.org/~lds/CGI.pm-3.41/CGI.pm.

[Ope08] The OpenLDAP Project: *OpenLDAP Software 2.4 Administrator's Guide, Kapitel 1*, 2008. http://www.openldap.org/doc/admin24/guide.html.

[PA01a] Paul Albitz, Cricket Liu: *DNS and Bind*, Kapitel 1 How does DNS work?, Seite 12. O'Reilly, Vierte Auflage, 2001.

[PA01b] Paul Albitz, Cricket Liu: *DNS and Bind*, Kapitel 6 Configuring Hosts - The Search List, Seite 104. O'Reilly, Vierte Auflage, 2001.

[Ric07] Ricciardi, Fulvio: *Kerberos protocol Tutorial* . http://kerberos.org/software/tutorial.html, 2007.

[Sta07] Stansberry, Matt: *Rechenzentrum muss Strom sparen - Energie wird strategischer Kostenfaktor*. Searchdatacenter, 2007.

[Tan03a] Tanenbaum, Andrew S.: *Computer Networks*, Kapitel 5.6.1 The IP Protocol, Seite 433ff. Prentice Hall, Vierte Auflage, 2003.

[Tan03b] Tanenbaum, Andrew S.: *Computer Networks*, Kapitel 5.5.6 Internetwork Routing, Seite 426ff. Prentice Hall, Vierte Auflage, 2003.

[Tan03c] Tanenbaum, Andrew S.: *Computer Networks*, Kapitel 6.4.1 Introduction to UDP, Seite 525ff. Prentice Hall, Vierte Auflage, 2003.

[Tan03d] Tanenbaum, Andrew S.: *Computer Networks*, Kapitel 4.7.6 Virtual LANs, Seite 328ff. Prentice Hall, Vierte Auflage, 2003.

[Tan03e] Tanenbaum, Andrew S.: *Computer Networks*, Kapitel 6.5.1 Introduction to TCP, Seite 532ff. Prentice Hall, Vierte Auflage, 2003.

[Tan03f] Tanenbaum, Andrew S.: *Computer Networks*, Kapitel 5.6.2 IP-Adresses, Seite 441ff. Prentice Hall, Vierte Auflage, 2003.

[Tre] Tregar, Sam: *HTML::Template - Perl module to use HTML Templates from CGI scripts.* http://search.cpan.org/ samtregar/HTML-Template-2.9/Template.pm.

[Wor99] World Wide Web Consortium (W3C): *HTML 4.01 Specification*, 1999. http://www.w3.org/TR/html401/.

[Wor07] World Wide Web Consortium (W3C): *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*, 2007. http://www.w3.org/TR/CSS21/.

[XML08] *XML-RPC.* Wikipedia, die freie Enzyklopädie, 2008. http://de.wikipedia.org/w/index.php?title=XML-RPC&oldid=49440576.

Index

S

T

U

W

X