Bachelorarbeit

Marcus Wenzel

Aspekte des Web-Based Enterprise Management (WBEM) im Energiemanagement

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Marcus Wenzel

Aspekte des Web-Based Enterprise Management (WBEM) im Energiemanagement

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

Betreuender Prüfer : Prof. Dr. rer. nat. Bernd Kahlbrandt
Zweitgutachter : Prof. Dr.-Ing. Franz Korf

Abgegeben am 30.03.2011

Marcus Wenzel

Thema der Bachelorarbeit

Aspekte des Web-Based Enterprise Management (WBEM) im Energiemanagement

Stichworte

WBEM, CIM, Gebäudemanagement, Energiemanagement, Smart Metering, iFlat, Green-IT

Kurzzusammenfassung

Das „Web-Based Enterprise Management“ (WBEM) ist ein Modell zur Verwaltung großer IT-Infrastrukturen. Diese Arbeit analysiert die Anwendbarkeit des WBEM auf das "erweiterte Gebäudemanagement". Dies geschieht mit Blick auf die Verbrauchsoptimierung von Gebäuden. Es wird der allgemeine Aufbau des WBEM/CIM analysiert und nach Anknüpfungspunkten an das Gebäudemanagement gesucht. Dies erfolgt in Form von Modellen. Des weiteren werden existierende WBEM-Projekte vorgestellt und diese auf ihre Verwendbarkeit untersucht. Es werden die „WBEM Services – Java Web-Based Enterprise Management“ vorgestellt und für eine Minimal-Implementierung verwendet.

Marcus Wenzel

Title of the paper

Aspects of Web-Based Enterprise Management (WBEM) in energy management

Keywords

WBEM, CIM, Buildung Management, Energy Management, Smart Metering, iFlat, Green IT

Abstract

The “Web-Based Enterprise Management” (WBEM) is a model for managing large IT infrastructures. This paper analyzes the applicability of the WBEM on the “Advanced Building Management”. This is done with a view of the optimized energy consumption of buildings. It analyzes the general structure of the WBEM/CIM and searched for intersections between WBEM and the building management. Furthermore, it presents existing WBEM-Projects and analyzes their usability. There will be the “WBEM Services – Java Web-Based Enterprise Management“ presented and used for a minimum implementation.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1. Einleitung

Die EDV hält immer weiter Einzug in unser alltägliches Leben. Dabei sind nicht nur die offensichtlich als solche erkennbaren, seit den 80'gern etablierten Heimcomputer/PCs, sonder auch immer mehr die Eingebetteten (embedded) Systeme zu finden. Sie sind in nahezu allen, heute zu findenden, Geräten vorhanden. Dort ersetzen sie die, vormals mechanisch realisierten, Programmwahl-Automaten (z.B. in der Waschmaschine) und ermöglichen zusätzliche Dienste, die auf althergebrachte weise nicht zu realisieren sind.

Durch die Verwendung von Bussystemen innerhalb der Geräte, lässt sich zudem der Verkabelungsaufwand erheblich verringern. Es muss nicht mehr jede Komponente mit mehreren Leitungen an das Steuergerät angeschlossen werden. Dicke fehleranfällige Kabelbäume gehören damit der Vergangenheit an. Auch die Fehlerdiagnose wird erheblich vereinfacht, denn es kann anhand von Systemprotokollen, auf fehlerhafte Bauteile geschlossen werden.

Was liegt näher, als sich die Vorteile von Embedded-Systems und Bussystemen, auch für das Gebäudemanagement nutzbar zu machen. Dadurch müsste nicht mehr jeder Schalter, über einen Schaltkreis, mit Lampen, Türöffnern, etc. verbunden sein. In jeder Installationskomponente (z.B. Taster, Lampe, Steckdose) steckte einfach ein Embedded-System.

Diese Komponenten bräuchte man nur noch mit einer Stromversorgung und dem Bus zu verbinden. Ein Steuergerät würde durch Events (z.B. das Betätigen des Tasters) über Veränderungen an den Komponenten informiert. Als Reaktion darauf, könnte das Steuergerät einer Lampe den Befehl geben, sich einzuschalten. Das Verhalten (Ursache → Wirkung) lässt sich frei programmieren. Dadurch lassen sich z.B. auch exotische Anwendungen, wie das Schalten der Kellerbeleuchtung mit einem Taster im Wohnzimmer, realisieren und nach belieben ändern (siehe [Saßmann2010], [Scherg2008]).

1.1 Motivation

Im Bereich der Gebäudeinstallation hat sich bereits seit Ende 90'er Jahre [G+H2010] eine derartige Vernetzung der Installationstechnik durchgesetzt. Die Auswahl an verfügbaren Installationskomponenten ist groß und reicht von einfachen Tastern bis hin zu komplexen Heizungssteuerungen (siehe Abbildung 1). Die Vernetzung der einzelnen Komponenten geschieht mittels eines oder mehrerer Bussysteme. Die drei großen Systeme die sich den Markt weitestgehend teilen sind der „Europäischen Installationsbus“ (EIB/KNX), das „Local Control Network“ (LCN) und das „Local Operating Network“ (LON). Eine genaue Statistik über die Installationszahlen gibt es nicht, so kann der jeweilige Marktanteil nur geschätzt werden und liegt jeweils zwischen 20-30% ([G+H2010], [Nieß2010], [Saßmann2010]). Außerdem gibt es eine Vielzahl proprietärer Systeme.

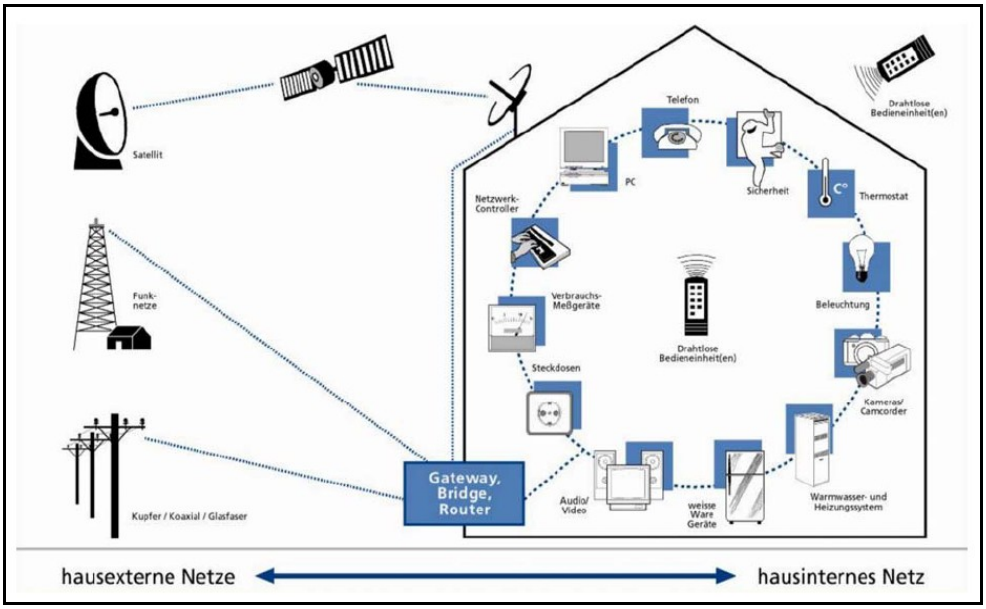


Abbildung 1: Systemumfang im inHaus Projekt [Altgeld2001]

Das Hauptproblem bei der Verwendung von Installationsbussen ist, dass Sie sich schon in der frühen Planungsphase auf eines dieser Systeme festlegen müssen. Die Entscheidung für ein bestimmtes System hat direkte Auswirkungen auf die Wahl der Verkabelung, denn die Art der Datenübertragung geschieht auf sehr unterschiedlichen Wegen mit noch unterschiedlicheren Protokollen! Das hat zur Folge, dass die verschieden Systeme zueinander inkompatibel sind. Mit der Festlegung auf ein Bussystem verringert sich jedoch die Auswahl an verfügbarer Installationstechnik. Die Entscheidung für ein „falsches“ Bussystem, kann nur mit erheblichem Kosten und Arbeitsaufwand geändert werden.

Gibt es auf dem Markt ein neues innovatives Produkt, kann es sein, dass Sie davon ausgeschlossen sind, weil Sie einen zum Produkt inkompatiblen Bus verwenden. Hier könnten Sie z.B. zwei Wege beschreiten:

- Sie installieren parallel zum vorhandenen System ein weiteren Bus
- Sie verwenden einen Adapter/Übersetzer

Wenn Sie die erste Lösung bevorzugen, haben Sie wieder einen hohen Kosten/Arbeitsaufwand für die Installation und ein neues Problem: Sie müssen ein Gateway verwenden, damit Geräte an verschiedenen Bussen, miteinander kommunizieren können (siehe Abbildung 2). Wenn sie auf diese Art mehr als zwei verschiedene Busse miteinander verbinden wollen, wird es schnell sehr unübersichtlich. Denn Sie benötigen ein Gateway pro Bus-Paar, das sind bei drei Bussen schon drei Gateways. Ein nicht unerheblicher Aufwand!

Die zweite Lösung hat den Nachteil, dass Sie für jedes „Fremdgerät“ einen Adapter benötigen. Das kann mit steigender Anzahl an „Fremdgeräten“ eine sehr teure Angelegenheit werden und unter Umständen reicht der, in den Installationsdosen vorhandene, Platz nicht für die Komponente und den Adapter aus.

Beide Lösungen haben den Nachteil, dass die Betreuer dieser Busse sich mit allen verwendeten Systemen und Komponenten auskennen müssen. Bei der Vielzahl an verschiedenen Verkabelungen, Protokollen und Adressierungen eine schwierige Aufgabe mit hohem Schulungsaufwand.

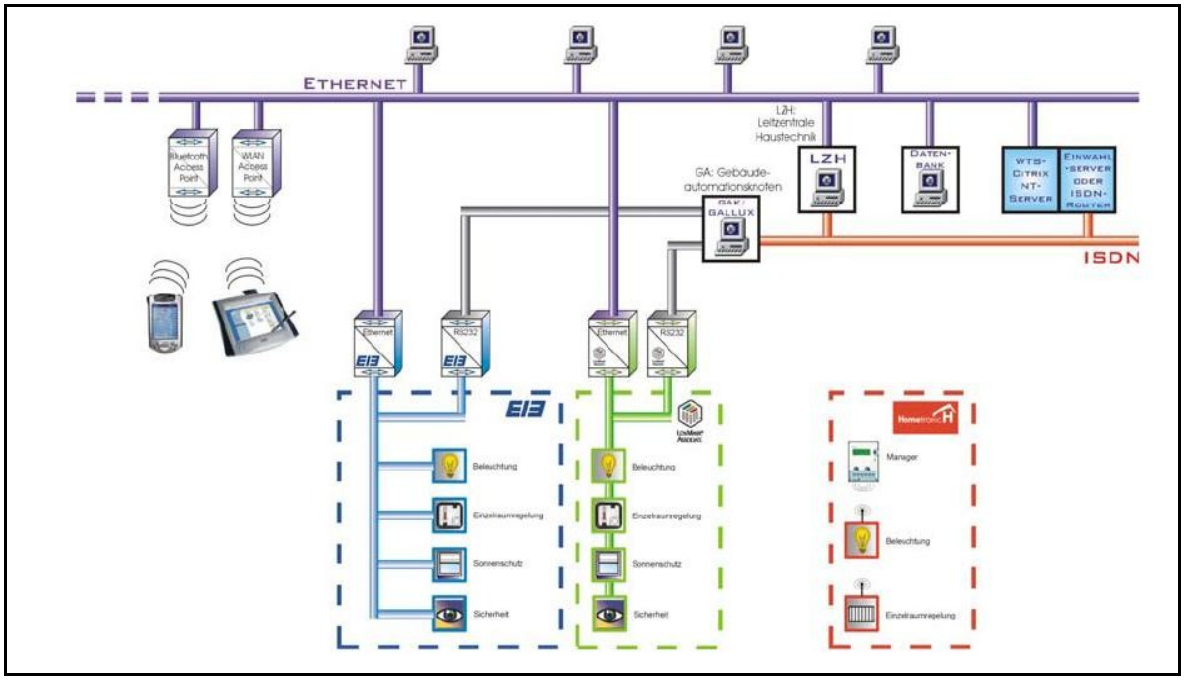


Abbildung 2: Beispiel Gebäudeinstallation [Altgeld2001]

Ein besserer Ansatz könnte es sein, den Weg mit den parallelen Bussen zu gehen, dabei aber die Gateways gegen einen zentralen Rechner auszutauschen. Dieser bräuchte dann nur einen Adapter pro Bus und eine geeignete Steuer- und Management-Software. Dieses Rechnersystem kann dann die Vermittlung zwischen den Bussen und den angebundenen Komponenten übernehmen. Zum Beispiel kann die kommerzielle Software „Elvis“ diese Aufgabe übernehmen und zusätzlich die Komponenten im Gebäude visualisieren [IT-GmbH2008]. Damit ist es möglich Geräte an verschiedenen Bussen, über das Rechnersystem, miteinander kommunizieren zu lassen.

Eine Software gibt Ihnen viel mehr Möglichkeiten in die Hand als eine reine Hardware-Lösung und Sie sind frei in Ihrer Entscheidung, von welchem Hersteller Sie welche Komponenten beschaffen.

Einen Schritt weiter gehen derzeit Hochschulen und Forschungseinrichtungen, sie verschmelzen die Installationstechnik mit Audio-, Video-, Kommunikationstechnik und Haushaltsgeräten, sowie zusätzlicher neuartiger Hardware (z.B. neuartige Benutzerschnittstellen). Diese Installationen werden dann, durch „intelligente Software“, zum „Leben“ erweckt. Diese Systeme sind in der Lage die Einsatzumgebung zu beobachten und auf vordefinierte Ereignisse zu reagieren.

Ziel dieser Entwicklungen ist es den Komfort der Benutzer zu erhöhen und auf dieser Grundlage neue Dienstleistungen anbieten zu können. Die dabei gewonnen Erkenntnisse werden genutzt, um neue innovative Produkte für den Endverbraucher zu fertigen.

Dieser Zugewinn an Komfort hat aber auch Nachteile, denn mehr Technik bedeutet auch mehr Stromverbrauch! Um den zu erwartenden Mehrverbrauch zu kompensieren gibt es mehrere Möglichkeiten, z.B.:

- Sie können neue energieeffizientere Technik einsetzen
- Sie können die vorhandene Technik sparsamer einsetzen

Das beste wäre eine Kombination aus beiden Punkten. Denn selbst das effizienteste Gerät verschwendet Energie, wenn es eingeschaltet ist, obwohl es nicht gebraucht wird.

Da das intelligente System seinen Einflussbereich schon überwacht und steuert, könnte es möglicherweise auch das Ressourcen-Management übernehmen! Zum Beispiel durch eine Erweiterung, der Software um einen „Energiespardienst“, der die Umgebung beobachtet und dann anhand von Mustern eine „Energieverschwendung“ erkennt und gegensteuert. Natürlich darf darunter der gewonnene Komfort nicht leiden. Gleichzeitig ist es sinnvoll den „Energiespardienst“ mit einem „Informationsdienst“ zu verbinden.

Das hat zwei Vorteile:

1. Der Benutzer hat weiterhin das Gefühl die Kontrolle über das System zu haben.
2. Es wäre so möglich den Anwender auf Fehlverhalten hinzuweisen.

Zu Punkt 2: Ähnliches kennen Sie vom Auto:

- Das Anschnallzeichen
- Der Lichtwarner

Dieser „Informationsdienst“ verfolgt das Ziel ihr Verhalten so zu beeinflussen, dass sie längerfristig auch außerhalb der eigenen Umgebung, Energiebewusst und damit Kostenbewusst handeln. Das Informationssystem kann nach Bedarf erweitert werden. Denkbar wäre eine Anzeige, die nicht die geleistete Arbeit (KWh) oder die aktuelle Leistungsaufnahme in Watt (W), sondern den Verbrauch in €/h ausweist. Dadurch würde

unmittelbar verdeutlicht, welchen Gegenwert die Energieeinsparung hat.

Einige Energieversorger, darunter RWE, arbeiten daran Ihnen ein ähnliches Auskunftssystem zur Verfügung zu stellen und haben erste Produkte auf den Markt gebracht. Stichwort: „Smart Metering“. Dazu ermitteln intelligente Stromzähler den Verbrauch und übermitteln diesen über ein Datennetz zum Versorger (siehe Abbildung 3). Der Energieversorger bereitet für sie die Daten auf und erstellt daraus Grafiken und Tabellen, die ihren Energieverbrauch visualisieren.

Leider sind die Systeme noch nicht ganz ausgereift, so das die „live Verfolgung“ der Messwerterfassung derzeit nicht möglich ist. Sie geschieht im Nachhinein und kann im „Kundenportal des Versorgers“ betrachtet werden. *„Das Standardintervall der Ablesung beim Smart Meter ist derzeit ein Monat.“* [RWE2010] Auch die Leistungsaufnahme der einzelnen Verbraucher können sie daraus nur schwer ermitteln, da nur der Gesamtverbrauch gemessen wird. Wollen Sie die Messwerte pro Gerät auswerten, so ist die Messung direkt an den einzelnen Verbrauchern erforderlich. Dies kann von den Energieversorgern derzeit nicht geleistet werden. Diese Daten sind jedoch für eine sinnvolle Steuerung der Energie-Ressourcen unverzichtbar, so das Sie selbst ein solches Messsystem betreiben müssen.

Abbildung 3: Smart Meter [RWE2010]

Die Herausforderung ist es jetzt alle vorbenannten Systeme zu einem Gesamtsystem zu vereinen. Dazu bedarf es eines Management-Systems, das auf der einen Seite mit allen Teil-Systemen kommunizieren kann und auf der anderen Seite ein einfaches Interface, für die Interaktion und Programmierung, des Gesamtsystems zur Verfügung stellt. Dies wird in Kapitel 1.2 weiter beleuchtet.

1.2 Problemstellung

Wie in Kapitel 1.1 behandelt, sind die im Gebäudemanagement anzutreffenden Schaltungen, Geräte und Bussysteme sehr heterogen. Die in Gebäuden vorzufindenden Kommunikations-, Unterhaltungs-, Haushalts- und sonstige Systeme verfügen, über keine einheitlichen Schnittstellen. Die Herausforderung ist es, über ein solches Gesamtsystem eine Management-Ebene zu legen. Dies wird durch die uneinheitlichen Schnittstellen der Einzelsysteme behindert. Man spricht hier von der sogenannten „Management Gap“ [Hobbs2004 Kapitel 4] (siehe Abbildung 4).

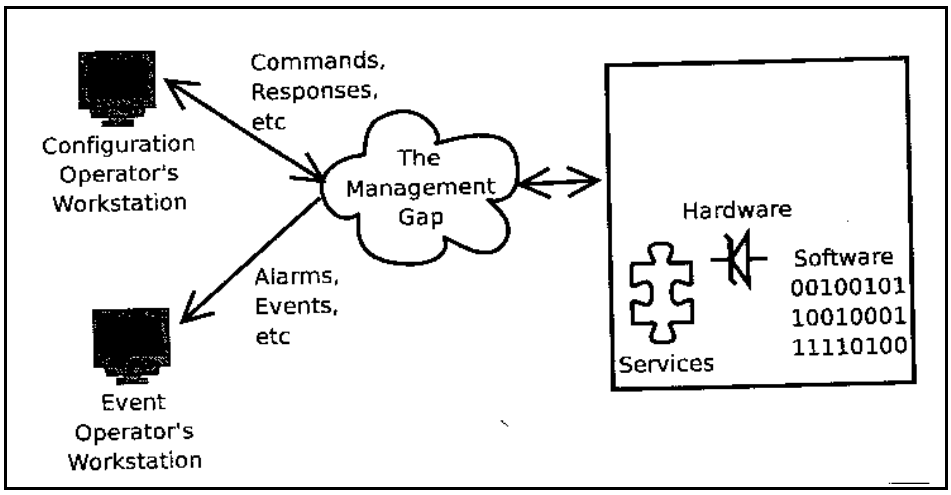


Abbildung 4: The Management Gap [Hobbs2004 S.34]

Beispiel:

Um ein Energiesparsystem für das Haus in Abbildung 5 zu schaffen, benötigen Sie Zugriff auf sämtliche Daten des Hauses, wie zum Beispiel:

- Umgebung des Hauses
  - Temperatur
  - Windgeschwindigkeit
  - Windrichtung
  - Luftfeuchtigkeit
  - Helligkeit
  - ...
- Zustand des Hauses
  - Verlassen
  - Bewohnt
  - ...
- Anzahl der Räume
- Art des Raumes
  - Küche
  - Bad
  - Schlafzimmer
  - Esszimmer

    - Wohnzimmer
    - Kellerräume
    - Garage
    - ...
- Zustand des Raumes
    - Temperatur
    - Luftfeuchtigkeit
    - Helligkeit
    - Beleuchtung
    - Personen
    - Fenster (auf/zu)
    - Türen (auf/zu)
    - ...
- Ausstattung
    - Waschmaschine
    - Hifi-Anlage
    - Herd
    - Lüftung
    - Heizkörper
    - …

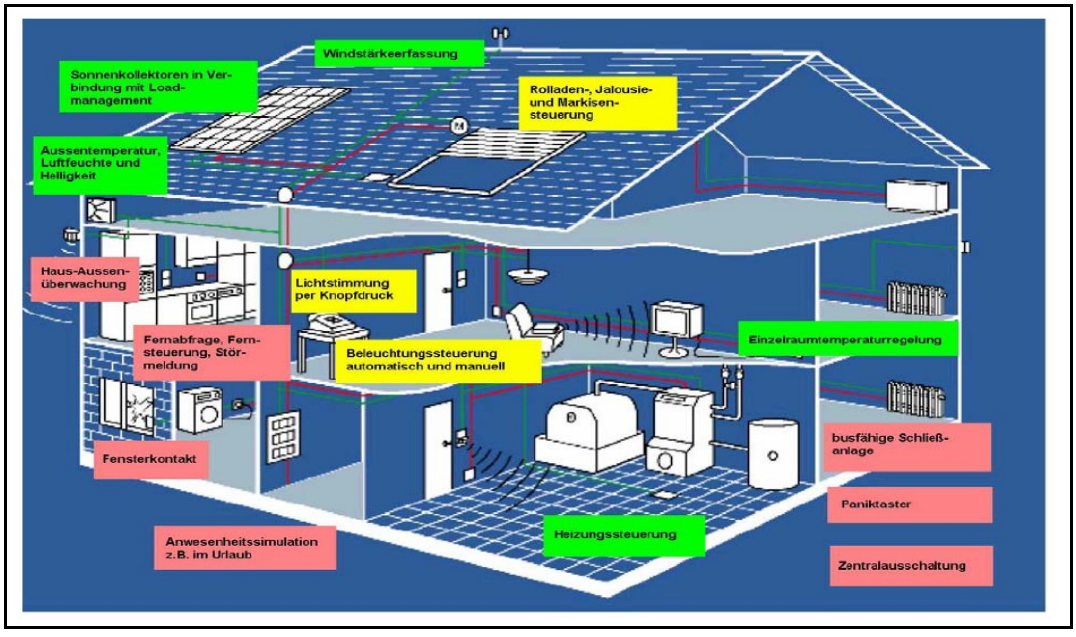


Abbildung 5: Modell eines intelligenten Hauses [Altgeld2001]

Diese unvollständige Aufzählung soll zeigen wie vielschichtig die anfallenden Informationen sind. Auch deren Repräsentationen sind sehr unterschiedlich, so das die anfallenden Informationen in eine einheitliche Form gebracht und ausgewertet werden müssen. Erst dann können die Informationen als sinnvolle Datenbasis dienen.

Die Informationen lassen wie folgt einordnen:

- Strukturinformationen (seltene Änderungen) z.B.
  - Räume
  - installierte Systeme
  - Heizkörper
  - Fenster und Türen
  - …
- Eigenschaftsinformationen (seltene Änderungen) z.B.
  - Abmessung und Lage des Heizkörpers
  - Lage der Fenster und Türen
  - Raummaße (H * B * T)
  - ...
- Zustandsinformationen (häufige Änderungen) z.B.
  - Raumtemperatur
  - Leistungsaufnahme
  - Windgeschwindigkeit
  - Fenster offen/geschlossen
  - ...

Die Strukturinformationen lassen sich hervorragend in einem UML Klassendiagramm, die Zustandsinformationen im Zustandsdiagramm und Aktivitäten im Aktivitätsdiagramm abbilden. Dies wird in Kapitel 4 behandelt und Lösungsmöglichkeiten diskutiert. Das Ausgangsproblem, wie die Daten aus der Anlage in die Management- Ebene kommen, ist noch nicht gelöst. Hier ist zu prüfen, ob mittels WBEM/CIM (siehe Kapitel 2 & 3) diese Lücke geschlossen werden kann.

1.3 Ziel

Im Rahmen dieser Arbeit soll geprüft werden, ob das WBEM/CIM für das „erweiterte Gebäudemanagement“ verwendet werden kann und ob es möglich ist auf dieser Grundlage ein Energiemanagement zu realisieren.

1.4 Aufbau der Arbeit

Die Kapitel 2 und 3 geben einen Einblick in die Funktionsweise vom WBEM und CIM und legen den Grundstein zum Verständnis der nachfolgenden Kapitel.

In Kapitel 4 wird die Einsatzumgebung analysiert und in Klassenmodellen abgebildet.

Kapitel 5 befasst sich mit den Interaktionen zwischen Einsatzumgebung, Benutzern und Sensoren. Diese werden in Form von „Szenarien“ dargestellt.

In Kapitel 6 werden die in den Kapiteln 4 und 5 entwickelten Modelle (exemplarisch) in Software implementiert. Dabei wird nach verwendbarer Software gesucht und diese in Anschluss auf ihre Eignung geprüft.

Kapitel 7 zieht ein Resümee und gibt einen Ausblick auf mögliche Fortführungen und Verbesserungsmöglichkeiten.

Kapitel 8 beherbergt die Anhänge, wie Glossar, Bibliographie und verwendete Software.

Innerhalb der Kapitel wird eine möglichst einheitliche Darstellung, von ähnlichem Inhalt, angestrebt.

Für die Darstellung von Quelltexten findet ein grauer Kasten Verwendung, in dem der Quelltext in der Schriftart `Courier` abgedruckt ist. In der Beschreibung (am Fuß des Kastens) wird die verwendete Sprache, der Klassenname und optional die Quelle genannt.

```
Quelltext
```

Listing x: <Sprache>: <Klassenname> [Quelle]

Für die Abbildung von Befehlseingaben auf einer Shell wird ebenfalls ein grauer Kasten verwendet. Darin wird ebenfalls die Schriftart `Courier` verwendet. Eine Befehlseingabe ist an dem „Shell-Prompt“ (z.B. user@rechner>), gefolgt vom eigentlichen Befehl zu erkennen. Beispiel:

Der Benutzer „user“ ist am Rechner „rechner“ angemeldet. Er gibt den Befehl „ls -l“, der den Inhalt des aktuellen Verzeichnisses (im Listenformat) anzeigt.

```
user@rechner> ls -l
```

Quellenangaben sind an eckigen Klammern, unter Angabe des Autors und dem Erscheinungsjahr des Dokuments (z.B. [Autor2011]), zu erkennen. Mit Hilfe dieser Verweise lassen sich, über die Bibliographie (Anhang B), die Original-Dokumente auffinden.

1.5 Einordnung

Bisher wird das „Web-Based Enterprise Management“ primär für die Verwaltung, Überwachung und Steuerung (Management) von Serverfarmen eingesetzt. Eine Verwendung von WBEM, im Gebäudemanagement, im Zusammenhang mit einem Energiemanagement, hat es nach meinen bisherigen Recherchen noch nicht gegeben.

1.6 Abgrenzung

Es soll untersucht werden, ob und welche Anknüpfungspunkte es zwischen dem „erweiterten Gebäudemanagement“ und dem WBEM/CIM gibt. Dabei soll die Untersuchung, Planung und Modellierung im Vordergrund stehen. Eine Implementierung erfolgt, sofern möglich, nur Exemplarisch. Für eine vollständige Implementierung aller gegebenen Beispiele, reicht die Bearbeitungszeit dieser Arbeit nicht aus.

2. Einführung in WBEM

Dieses Kapitel soll einen Überblick über die Struktur des „Web-Based Enterprise Management“ (WBEM) geben. WBEM ist eine Entwicklung der „Distributed Management Task Force“ (DMTF). Die DMTF ist einen Standardisierungsgremium, dem viele namhafte Firmen der TK- und IT-Branche (darunter: Mircosoft, Intel, AMD, HP, DELL, IBM, Oracle, …) angehören.

Das WBEM ist ein Paket, bestehend aus mehreren Internet- und Management-Techniken, zur Verwaltung großer IT-Infrastrukturen wie Netzwerken und Serverfarmen sowie Telekommunikationsinfrastrukturen. Zusammengefasst stellt WBEM ein Modell dar, das es ermöglicht, die Kommunikation und das Management verschiedener Teilbereiche eines Systems zu vereinheitlichen. Die Vereinheitlichung erstreckt sich auf:

- die Systemstruktur
- das Datenmodell
- die verwendeten Protokolle
- die Datenrepräsentation

Die DMTF hat dabei nicht das Rad neu erfunden, sonder auf etablierte WWW-Techniken und eigene Standards zurück gegriffen. Das DMTF eigene „Common Information Model“ (CIM) (siehe Kapitel 3) wird als Datenmodell eingesetzt.

Für die Kommunikation wird das „Hypertext Transfer Protocol“ (HTTP), bzw. dessen verschlüsselte Variante „Hypertext Transfer Protocol Secure“ (HTTPS), der „Object Management Group“ (OMG) verwendet. Die Datenrepräsentation erfolgt mittels „Extensible Markup Language„ (XML), welche ebenfalls eine Entwicklung der „OMG“ ist.

Neben den Protokollen, dem Datenformat und dem Datenmodell, legt WBEM auch die Aufteilung des Systems in folgende Bestandteile nahe:

- WBEM-Server(n)
- WBEM-Providern
- WBEM-Clients
- WBEM-Listener

Diese Komponenten (siehe Abbildung 6) werden in den folgenden Unterkapiteln näher beschrieben.

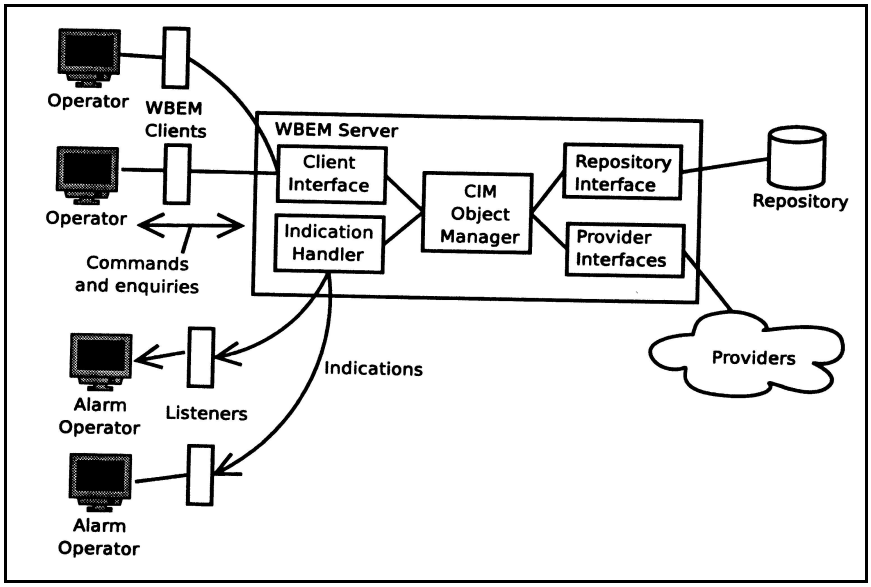


Abbildung 6: WBEM-Aufbau (vereinfacht) [Hobbs2004 S.124]

Wichtig! WBEM ist nur ein Modell ohne jegliche Implementierung!

Es gibt bereits eine überschaubare Anzahl von Projekten (z.B. WBEM Services / openPegasus [OG2005]), die das WBEM/CIM in Software implementiert haben. Die Projekte setzen dabei auf unterschiedliche Programmiersprachen (z.B. C/C++, Java), verfolgen aber ähnliche Ziele.

Auch die Software-Lizenzierung der Projekte ist sehr unterschiedlich und reicht von „Kommerziell“ bis „Open Source“. Die Auswahl einer geeigneten WBEM-Software, ist sehr stark von den Zielen des eigene Projektes abhängig. Die Auswahl, der für dieses Projekt am besten geeigneten Software, wird in Kapitel 6 diskutiert.

2.1 WBEM-Server

Der WBEM-Server dient als Bindeglied und Vermittler zwischen den Clients und Listener auf der einen Seite und den Providern auf der Anderen. Er kennt die Struktur des zu verwaltenden Systems und überwacht die Zugriffsrechte der Benutzer und Systeme.

Der WBEM-Server besteht aus mehreren Komponenten (siehe Abbildung 7), die Hauptkomponenten sind der „CIM Objekt Manager“ (CIMOM), das „Repository“ und eine Vielzahl von „Adaptern“. Der CIMOM verwaltet die sogenannten „Managed Objekts“. „Managed Objekts“ sind z.B. Instanzen von Klassen, aber auch die Klassen selbst. Außerdem werden Informationen über deren Eigenschaften (Protertys) und Beziehungen (Associations) gespeichert.

Die Klassen und Assoziationen müssen im „Managed Object Format“ (siehe Kapitel 3.4) bereitgestellt werden. Diese werden mit Hilfe eines „MOF- Compiler“ (siehe Kapitel 6.2) ins Repository eingespielt.

Der WBEM-Server hält auch Informationen über die Provider (siehe Kapitel 2.2), welche die Klassen / Instanzen betreuen.

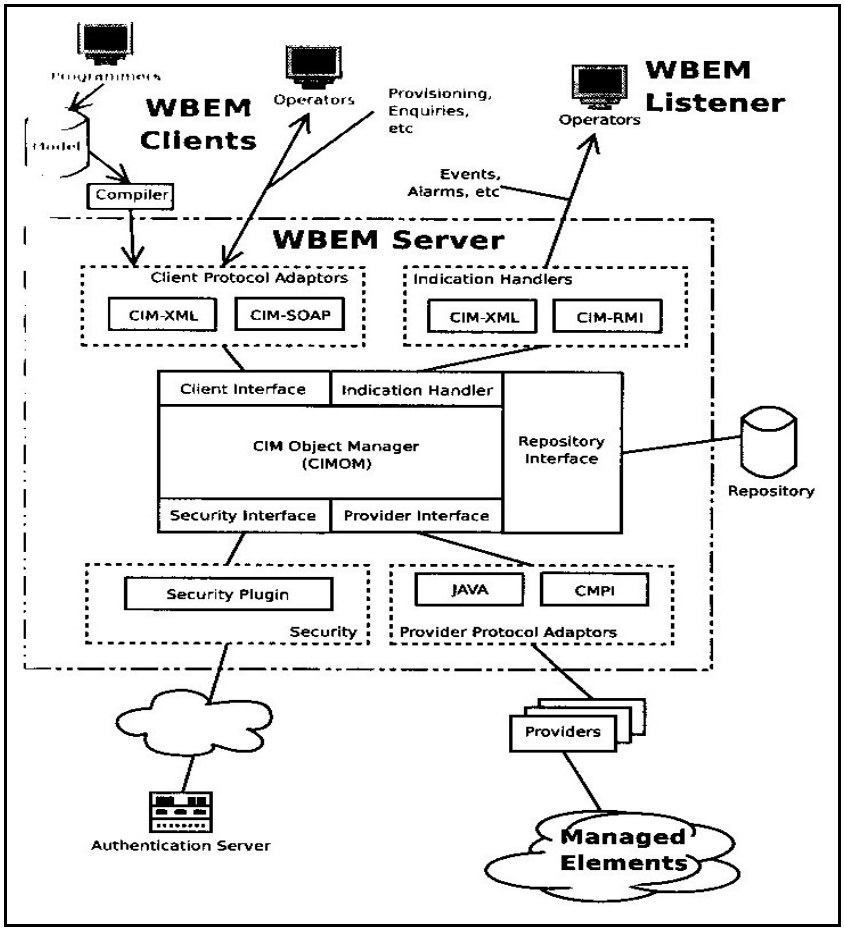


Abbildung 7: WBEM-Server Architektur [Hobbs2004 S.39]

Mit Hilfe von Instanz-Providern können Instanzen der Klassen erzeugt werden. Die

Erzeugung statischer Instanzen ist auch per MOF-Datei möglich. Das Repository stellt somit die umfassende Datenbasis für das WBEM dar.

Der Aufbau des Repository ist von der jeweiligen Implementierung abhängig. Es können SQL-Datenbanken, Konfigurationsdateien, Verzeichnisstrukturen oder jede andere denkbare Art der Datenspeicherung zu Einsatz kommen.

Um mit seiner Umwelt kommunizieren zu können und die interne Datenrepräsentation des Repository in andere Formate zu exportieren, verfügt der WBEM-Server über Adapter. Die Adapter stellen Schnittstellen für die Clients und Provider dar, dabei werden mehrere Protokolle angeboten. Die wichtigsten von ihnen sind:

Protokoll:	***Verwendung:***
CIM-XML	Ist das Protokoll für die Kommunikation des WBEM-Server mit: • Clients • Listenern • anderen WBEM-Servern • MOF-Compiler
CMPI	Ist ein Standard-Protokoll für die Kommunikation zwischen WBEM-Server und Providern.
Java-RMI	Die „Remote Method Invocation“ wird in Java-Implementierungen, als Alternative / Ersatz für CMPI, zur Kommunikation zwischen Server und Provider verwendet.

Tabelle 1: WBEM- Protokolle

Wie Tabelle 1 (CIM-XML) indirekt zu entnehmen ist, besteht die Möglichkeit mehrere WBEM-Server hierarchisch anzuordnen. Dabei tritt der abfragende WBEM-Server, gegenüber dem befragten Server, als Client auf. Für nähere Informationen zu CIM-XML wird auf die Dokumentation der DMTF verwiesen (siehe „CIM Operations over HTTP“ [DMTF2009-2] & „Representation of CIM in XML“ [DMTF2009-3]).

CMPI steht für „Common Manageability Programming Interface“ und ist ein offener Standard von „The Open Group“. Es handelt sie dabei um eine standardisierte Provider-Server-Schnittstelle. Bis zur Schaffung dieses Standards, gab es keine einheitliche Schnittstelle zur Kommunikation zwischen Provider und Server. Erst durch diese Vereinheitlichung ist es möglich einen Provider zu entwickeln und mit nur geringem Aufwand, mit verschiedenen Server-Implementationen zu betreiben. Leider wird diese Schnittstelle nicht von jedem WBEM-Projekt unterstützt.

Weitere Informationen zum WBEM-Server finden sie im Kapitel 6.2, [Hobbs2004] und [SUN2005]. Im nächsten Unterkapitel werden Provider behandelt.

2.2 Provider

Ein Provider stellt die Schnittstelle zwischen dem WBEM-Modell und dem „Rest der Welt“ dar. Provider kapseln WBEM fremde Elemente, wie Hardware, Software oder Dienste (siehe Abbildung 8) und machen sie für das WBEM-Modell erreichbar. Damit ein Provider die Betreuung eines dieser Elemente übernehmen kann, muss er manuell an das zu bereuende Element angepasst werden. Diese Anpassung muss für jedes weitere Element erneut durchgeführt werden. In Richtung des WBEM-Server, stellen Provider eine standardisierte Schnittstelle (z.B. JAVA-RMI oder CMPI (Kapitel 2.1)) zur Verführung.

Das hat die positive Wirkung, dass zwei unterschiedliche Provider auf die gleiche Art befragt werden können und sie die Ergebnisse im selben Format zurückliefern. Die Repräsentation der erfragten Daten, kann innerhalb der Betreuten Elemente (z.B. Hardware), sehr verschieden sein. Die Provider müssen die geforderten Daten aus dem betreuten Element auslesen und die Daten in das einheitliche Format konvertieren.

Die betreuten Elemente werden durch sogenannte MOF-Dateien (siehe Kapitel 3.4) beschrieben. In diesen Dateien werden auch die, für die Klasse zuständigen, Provider definiert. Ein Provider stellen in diesem Zusammenhang ein Programm dar, das für die ihm zugeordnete Klasse zuständig ist. Dieses Programm muss in kompilierter Form in einem Verzeichnis des WBEM-Server (CIMOM) liegen. Der CIMOM startet, bei einem Zugriff auf die betreute Klasse (bzw. dem Erzeugen einer Instanz der Klasse), automatisch die zuständige Provider-Anwendung.

Es gibt verschiedene Arten von Providern, hier ein Überblick:

Typ:	***Beschreibung:***
Method Provider	Nimmt Methoden-Aufrufe aus dem WBEM entgegen und leidet diese an die von ihm betreuten Elemente weiter.
Instance Provider	Verwaltet das Erzeugen, Löschen und Aufzählen von Instanzen.
Property Provider	Verwaltet den Aufruf von Setter- und Getter- Methoden und beobachtet Veränderungen.
Association (Associator) Provider	Verwaltet Assoziationen zwischen Klassen / Instanzen sowie deren Erzeugung und Löschung.
Indication Provider	Verwaltet im System aufkommende Events und leitet diese, über den WBEM-Server, an angemeldete Clients weiter.
Query Provider	Verwaltet Anfragen, die in einer Art Datenbank-Abfragesprache gestellt werden.
Autorizable	Hierbei handelt es sich um ein Marker-Interface. Durch die Implementierung dieses Interfaces erkennt der CIMOM (siehe Abbildung 7), dass der Provider die Zugangskontrolle selbst durchführt.

Tabelle 2: Provider-Typen

Ein Provider kann auch mehrere der in Tabelle 2 aufgelisteten Provider-Typen implementieren. So kann ein Provider z.B. Instanz- und Methoden-Provider zugleich sein.

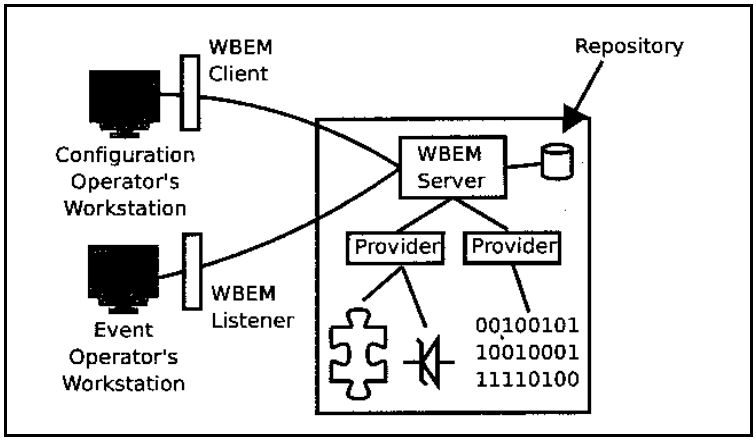


Abbildung 8: The WBEM Components [Hobbs2004 S.35]

Zur Behandlung von Events, werden sogenannte Indication-Provider verwendet. Events der betreuten Elemente werden an diese Indication-Provider abgegeben. Diese Provider lösen ihrerseits einen Event innerhalb des WBEM-Servers aus. Der WBEM-Server gibt diese an die angemeldeten Clients / Listener weiter.

Weitere Informationen sind [Hobbs2004 S.201-202] und [SUN2005 S.99-100] zu entnehmen.

Im nachfolgenden Unterkapitel werden Erläuterungen zum Client gegeben.

2.3 Client

Ein Client stellt die Benutzerschnittstelle des WBEM-Modell dar. Dabei stellt WBEM keine GUI oder ähnliches zur Verfügung, sonder nur die Möglichkeit auf standardisierte Weise mit dem WBEM-Server zu kommunizieren (siehe Abbildung 9).

Ein Client ist dadurch in der Lage, im Rahmen seiner Befugnisse, den WBEM-Server zu befragen und Anweisungen zu erteilen. Anfragen an den WBEM-Server werden, soweit möglich, direkt vom Server beantwortet. Ist eine direkte Antwort nicht möglich, z.B. weil die notwendigen Informationen im Repository fehlen, wird die Anfrage an den zuständigen Provider weitergeleitet.

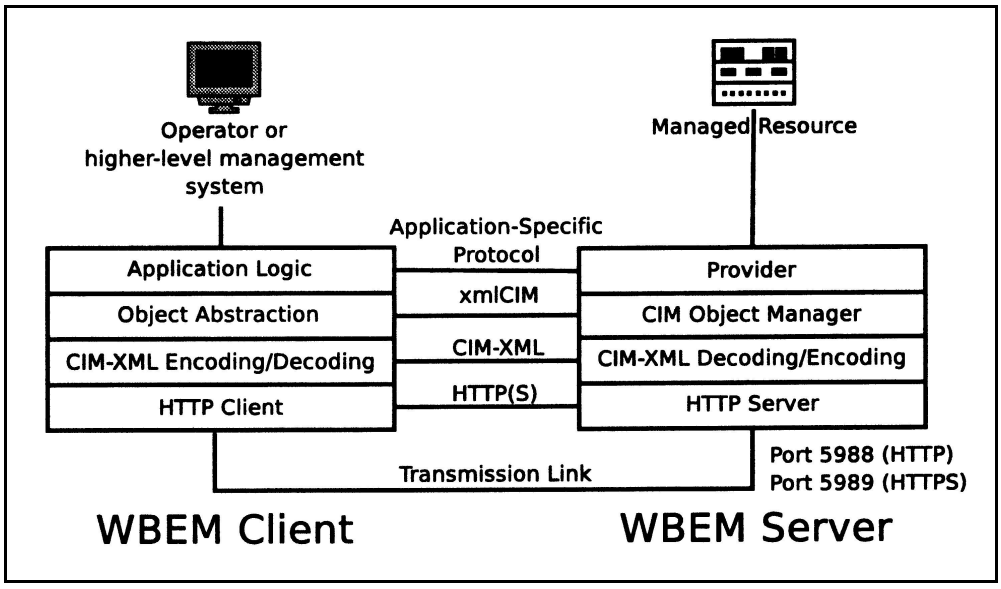


Abbildung 9: WBEM Client / WBEM Server Interaction [Hobbs2004 S.126]

Die Antwort des Providers wird ausschließlich über den Server an den Client weitergeleitet. Ein direkte Kommunikation zwischen Client und Provider findet nicht statt.

Für die Kommunikation zwischen Client und Server wird CIM-XML verwendet. (siehe Ende Kapitel 2.1) Für weitere Informationen zum Client siehe Kapitel 6, [Hobbs2004 Kapitel 7] und [SUN2005 S.45ff]

Kapitel 2.4 gibt eine kurze Erläuterung zu Listenern.

2.4 Listener

Ein Listener ist eine eingeschränkter Client. Aus WEBM- Sicht ist er passiv und bezieht ausschließlich Informationen vom WBEM-Server. Die Informationen erhält der Listener durch „Indications“ (siehe Abbildung 10). Eine Indikation ist eine, durch einen Event erzeugte, Nachricht. Um diese Nachrichten zu erhalten muss sich der Listener beim Server (für ein bestimmtes Event) registrieren. Befehle an den WEBEM- Server kann ein Listener nicht abgeben.

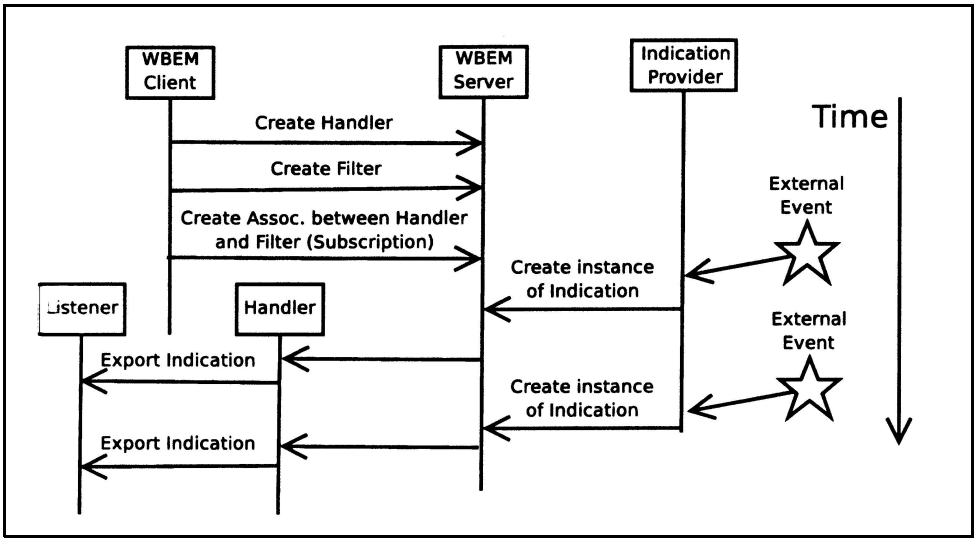


Abbildung 10: UML: Sequenzdiagramm: Indication [Hobbs2004 S.153]

Für die Kommunikation zwischen Listener und Server wird CIM-XML verwendet. (siehe Ende Kapitel 2.1)

Im Nachfolgenden Kapitel 3 wird eine Einführung in das „Common Information Model“ (CIM) gegeben.

3. Einführung in CIM

CIM steht für „Common Information Model“ und ist ein von der DMFT entwickelter Standard. Ziel dieses Standards ist es, ein einheitliches und systemunabhängiges Datenmodell zu schaffen, das an keine Implementierung gebunden ist. Es soll den Datenaustausch in heterogenen und verteilten Systemen vereinfachen.

Abbildung 11 veranschaulicht die Einsatzgebiete von CIM. Zu beachten ist, dass die DMTF für dieses Diagramm eine eigene Notation verwendet (im Gegensatz zu den später verwendeten UML-Diagrammen).

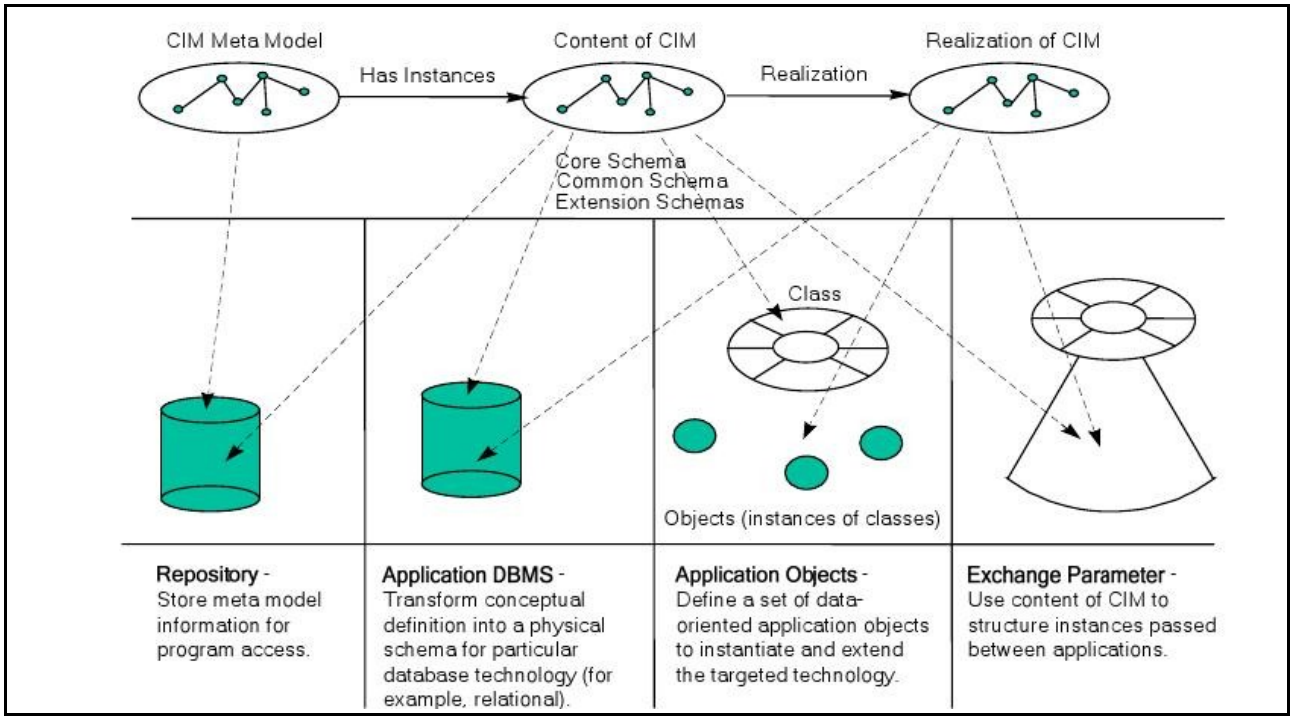


Abbildung 11: CIM- Schema - Anwendungen [DMTF2009]

Das CIM-Schema lässt sich in drei Bereiche unterteilen:

- Core Schema
- Common Schemata
- Extension Schemata

Tabelle 3 beinhaltet die Schemen-Beschreibung der DMTF.

Schemata:	***Beschreibung:***
Core Model	An information model that captures notions applicable to all domains of management.
Common Models	Information models that capture notions common to particular management domains but independent of a particular technology or implementation. The common domains include Systems, Applications, Devices, Users, Networks, Policies and Databases.
Extension Models	Represent technology-specific extensions of the Common Models. These models are specific to environments, such as operating systems, or to vendors.

Tabelle 3: CIM-Schemata [DMTF2009]

Abbildung 12 gibt einen Überblick wie die einzelnen Schemata aufeinander aufbauen. Im Kern liegt das „Core Schema“ es stellt die Basis des CIM-Modells dar. Um das „Core Schema“ herum sind die „Common Schemata“ angeordnet. Sie sind eine Erweiterung des „Core Schema“. Um die „Common Schemata“ liegen die „Extension Schemata“ sie sind herstellerspezifische Erweiterungen der „Common Schemata“.

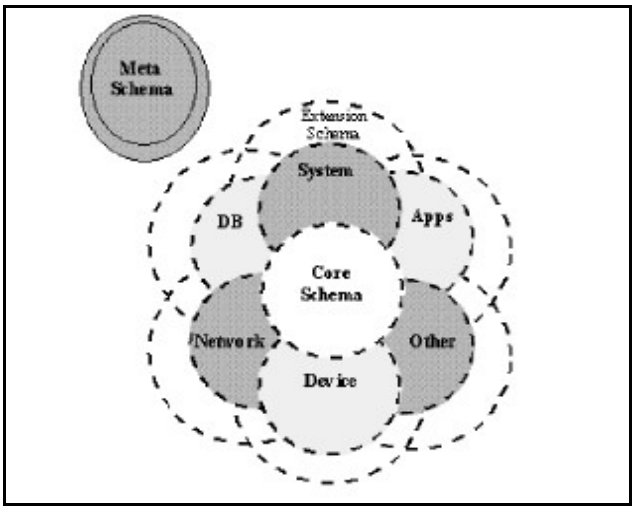


Abbildung 12: CIM- Schemata [Süß1998]

Die drei Arten vom Schemata werden in den nachfolgenden Unterkapiteln detaillierter beschrieben.

3.1 Core Model

Das „Core Model“ der DMTF stellt den kleinsten gemeinsamen Nenner aller denkbaren Einsatzbereichen dar. Es ist die Basis für die „Common Models“, die sich mit der plattformunabhängigen Behandlung von Geräten, Netzwerken, Benutzern und anderem befassen.

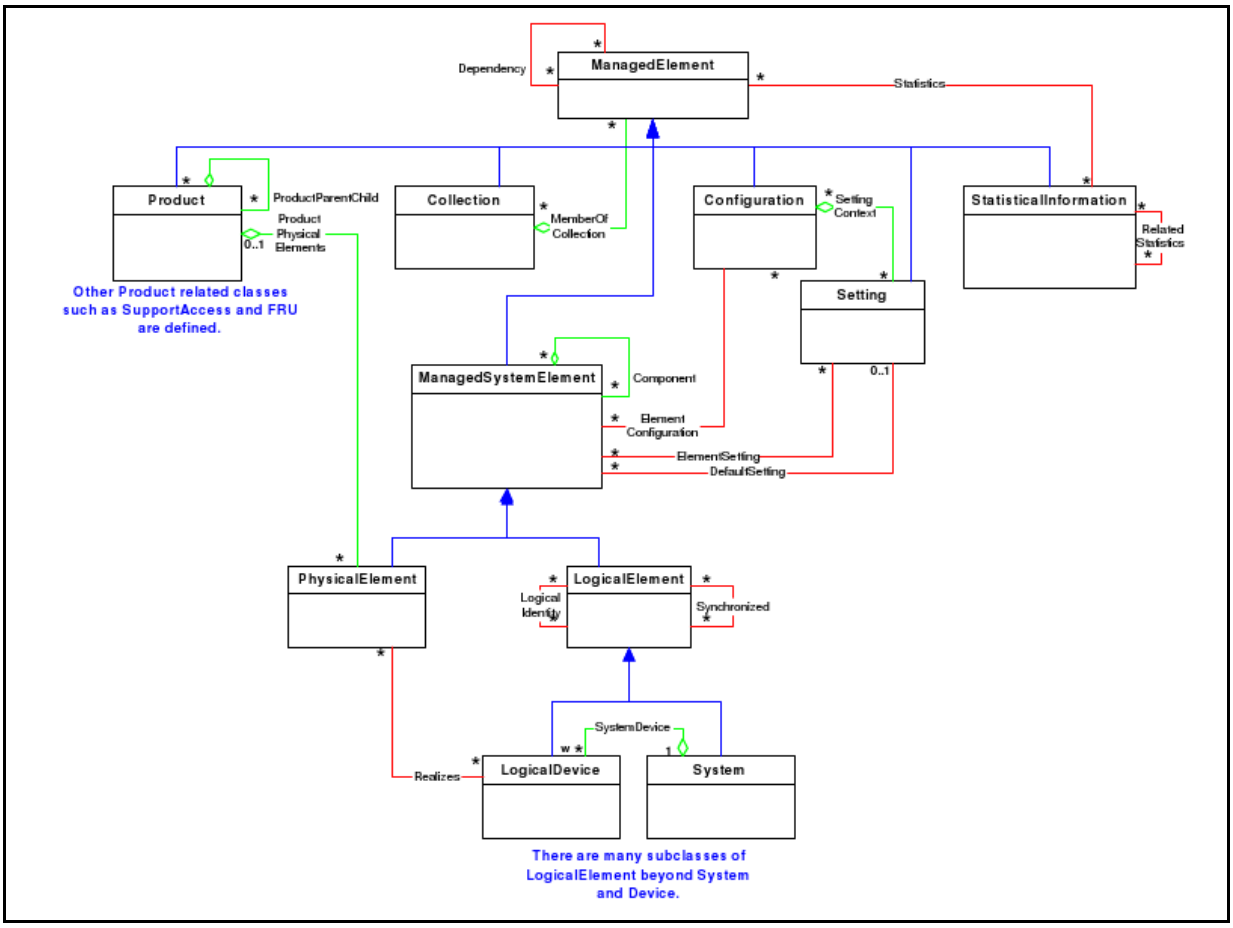


Abbildung 13: UML: CIM-Core Modell [DMTF2000]

Um das eigene System zu einem Bestandteil des CIM-Modells zu machen, muss entweder eine Klasse des CIM Schema implementiert werden oder das eigene Modell erbt mindestens von einer Klasse des Core Schema. So wird eine Minimal-Kompatibilität zu anderen „CIM-Systemen“ geschaffen. Dies ist nur möglich, da alle Klassen von der Superklasse „CIM_ManagedElement“ abgeleitet sind (siehe Abbildung 13). Das „Core Model“ ist beschrieben in DMTF-Dokument DSP0111 [DMTF2000].

In Kapitel 4 kommen die in Tabelle 4 aufgeführten Klassen des Core Schema zum Einsatz:

Klasse:	***Beschreibung:***
CIM_LogicalDevice	An abstraction or emulation of a hardware entity, that may or may not be Realized in physical hardware. Any characteristics of a LogicalDevice that are used to manage its operation or configuration are contained in, or associated with, the LogicalDevice object. Examples of the operational properties of a Printer would be paper sizes supported, or detected errors. Examples of the configuration properties of a Sensor Device would be threshold settings. Various configurations could exist for a LogicalDevice. These configurations could be contained in Setting objects and associated with the LogicalDevice.
CIM_ManagedSystemElement	CIM_ManagedSystemElement is the base class for the System Element hierarchy. Any distinguishable component of a System is a candidate for inclusion in this class. Examples of system components include: - software components such as application servers, databases, and applications - operating system components such as files, processes, and threads - device components such as disk drives, controllers, processors, and printers - physical components such as chips.
CIM_System	CIM_System represents an entity made up of component parts (defined by the SystemComponent relationship), that operates as a 'functional whole'. Systems are top-level objects in the CIM hierarchy, requiring no scoping or weak relationships in order to exist and have context. It should be reasonable to uniquely name and manage a System at an enterprise level. For example, a ComputerSystem is a kind of System that can be uniquely named and independently managed in an enterprise. However, these qualities are not true for the power supply (or the power supply sub-'system') within the computer. Although a System can be viewed as a Collection, this view is not the correct model. A Collection is simply a 'bag' that 'holds' its members. A System is a higher-level abstraction, built out of its individual components. It is more than the sum of its parts. Note that System is a subclass of EnabledLogicalElement which allows the entire abstraction to be functionally enabled or disabled at a higher level than enabling or disabling its component part.

Tabelle 4: Klassenbeschreibung [DMTF2010]

Eine Erläuterung, warum genau diese Klassen zu Einsatz kommen, wird ebenfalls in Kapitel 4 gegeben. Im nachfolgenden Unterkapitel wird eine kurze Erläuterung zu einem Common Schema gegeben.

3.2 Common Modelle

Die Common Schemata stellen die Erweiterungen des Core Schema dar. Sie sind sehr anwendungsspezifisch, jedoch mit gebotenem Abstand zu einer bestimmten Implementierung. Es gibt viele Common Schemata, darunter z.B.:

- Database
- Device
- Event
- Physical
- User

Aus dem Common-Schema Devices (CIM::Devices::Sensors) wird in Kapitel 4 die Klasse „CIM_NumericSensor“ Verwendung finden. (siehe Abbildung 14)

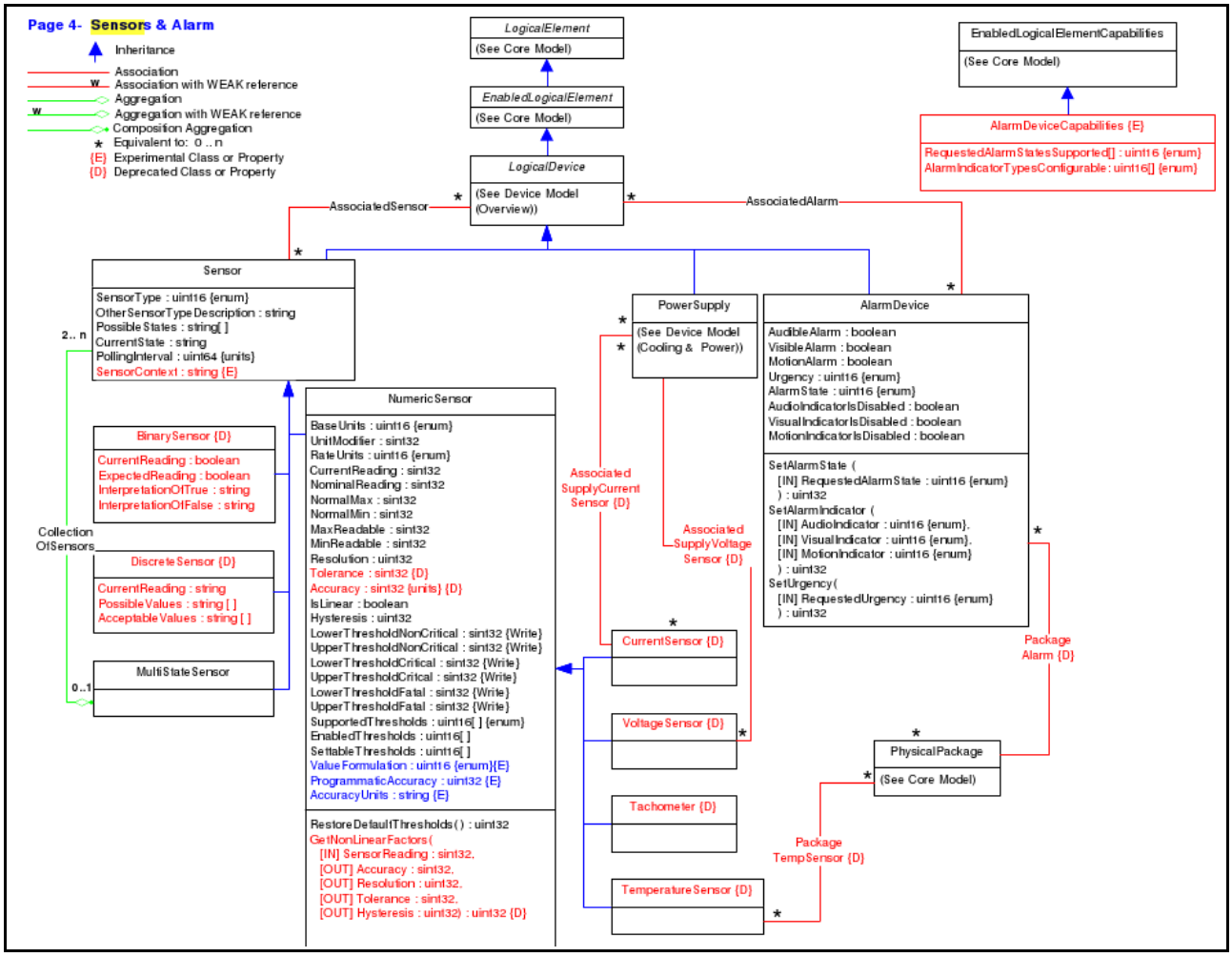


Abbildung 14: UML: CIM::DEVICES::SENSORS [DMTF2010]

Eine Begründung, warum diese Klasse verwendet wird, folgt in Kapitel 4. Für weiterführende Informationen siehe [DMTF2003], [DMTF2010] und [Hobbs2004 S.87-120].

Kapitel 3.3 behandelt die „Extension Schemata“.

3.3 Extension Modelle

Die Extension Schemata sind Erweiterungen der Common-Schema die sehr Herstellerspezifisch sind. Sie bauen gezielt auf ein bestimmtes System oder eine Klasse von Systemen auf, sind also nicht wie die Core- und Common-Schemata, von der Implementierung unabhängig. Durch das Erben von Basisklassen, sind jedoch auch diese Systeme, von jedem Client, mindestens auf Grundlage der Basisklassen zu managen.

Die in Kapitel 4 entworfenen Modelle zur Gebäudeautomation, sind Erweiterungen des „Core-“ und eines „Common Schema“ und stellen damit ebenfalls ein Extension Schema dar.

Kapitel 3.4 gibt eine Beschreibung des „Managed Object Format“ (MOF).

3.4 Managed Object Format (MOF)

MOF steht für „Managed Objekt Format“ und ist eine textuelle Darstellung von UML-Klassendiagrammen (nicht zu verwechseln mit „Meta Object Facility" der „OMG“). Die Sprache „MOF“ ist ebenfalls eine Entwicklung der DMTF und im Dokument DSP0004 [DMTF2009] beschrieben.

MOF lässt sich wie eine Programmiersprache einsetzen und ist dadurch sehr flexibel. Um z.B. eine neue Assoziation einzufügen, werden nur 3 Zeilen Code eingefügt (siehe Listing 4). Für Testzwecke lassen sich beliebige Teile auskommentieren (siehe Tabelle 5). Um gleiches in einem UML-Gaphik-Tool (z.B. Umbrello) zu erledigen, ist eine Vielzahl von Mausklicks erforderlich.

Interessant und arbeitserleichternd wäre eine automatische Umwandlung zwischen MOF und UML. Internetrecherchen nach derartigen Tools sind, aufgrund der Abkürzungsgleichheit zu „OMG – MOF“, schwierig und bislang erfolglos geblieben. Die DMFT selbst, stellt ihren Mitglieder verschiedene Tools zur Verfügung. Ob sich darunter ein Konverter „UML ↔ MOF“ befindet, konnte aufgrund der fehlenden Mitgliedschaft, nicht geprüft werden.

Systeme die CIM implementieren, müssen den Import und Export von MOF-Dateien beherrschen. Die meisten CIM-Realisierungen stellen dafür einen MOF-Compiler (siehe Kapitel 6.2) zur Verfügung. Dieser nimmt eine Übersetzung in die interne Repräsentation des WBEM-Server vor. Das Resultat wird in dessen Repository (siehe Kap.2.1) gespeichert.

Im folgenden wird eine minimale Einführung in die MOF-Sprache gegeben. Es werden nur die, im Rahmen dieser Arbeit, verwendeten Sprachelemente beschrieben. Für weitere Informationen siehe [DMFT2009], [Hobbs2004].

Verwendete Elemente der Sprache in der Zusammenfassung:

MOF:	***Beschreibung:***	***Vergleich in JAVA:***

`#pragma include(“xxx.mof“);`
`/* Import des Datei- Inhaltes an diese stelle */`
 	
`#pragma namespace(“root\cimv2“);`
`/* Festlegen des Namespace */`
 	
`#pragma locale(“de_de“);`
`/* Festlegen des Landes und der verwendeten Sprache */`
	Compiler Direktive – Implementierungen außerhalb der MOF-Sprache	`keine Entsprechung`
		`(ähnlich „#import xxx.h“ in c)`

`class Test_Class{ };`	Deklaration einer Klasse	`Class TestClass{ };`
`class Test_Class : Super_Class { };`	Eigenschaften einer Klasse erben	`class TestClass extends SuperClass { };`
`// Dies ist einen Kommentar` `/* Dies ist einen Kommentar */`	Kommentare	`// Dies ist einen Kommentar` `/* Dies ist einen Kommentar */`

`Description(“Beschreibender Text“)`	Klassen- / Methoden-Beschreibung	`ähnlich JavaDoc:`
		`/**`
		`* Beschreibender Text`
		`*/`

Tabelle 5: MOF- Sprachelemente

Die Tabelle 5 listet einige MOF-Sparachelemente, die im folgenden Verwendung finden, auf und verweist (soweit möglich) auf eine Java Entsprechung. Zu beachten ist, dass die MOF- Sprache nicht „case-sensitive“ ist. „Class“, „class“ und „CLASS“ meinen das selbe!

Syntax:

```
[<Qualifiers>]
class <class name> : <superclass name>
       {
       <property or method>;
       <property or method>;
       <property or method>;
       .
       .
       .
       <property or method>;
       };
```

Listing 1: MOF: Syntax – Class [Hobbs2004 S.74]

Listing 1 veranschaulicht den prinzipiellen Aufbau einer MOF-Datei. Die Datei beginnt mit der sogenannten Qualifiers-Section. In ihr werden „Qualifiers“ eingetragen, mit denen die nachfolgende Klasse um zusätzliche Informationen angereichert (z.B. Klassenbeschreibung) bzw. eingeschränkt (z.B. Klasse als Abstrakt markiert) werden kann. Eine Auswahl an möglichen „Qualifiers“ kann Tabelle 6 entnommen werden. Die Verwendung von Qualifiers ist nicht obligatorisch, kann also weggelassen werden.

Die 2. Zeile stellt eine Klassendefinition dar. Dem Signalwort „class“ folgt der Klassenname, sowie optional eine Superklasse von der geerbt wird, mit vorangestellten Doppelpunkt. Innerhalb der Klassendefinition erfolgt die Deklaration der Variablen (Listing 2) und Methoden (Listing 3).

```
[<qualifires>]
<datatype> <property name> { = default };
```

Listing 2: MOF: Syntax – Property [vergl. Hobbs2004 S.74]

Auch bei der Variablendeklaration (Listing 2) besteht die Möglichkeit, mit Hilfe von „Qualifiers“ z.B. eine Variablenbeschreibung oder einen Maximalwert festzulegen. Im Anschluss folgt die Deklaration des Datentyps (siehe Tabelle 7) gefolgt vom Variablennamen. Optional lässt sich ein initialer Wert für die Variable festlegen.

```
[<qualifires>]
<return datatype> <method name> (<parameter>, … <parameter>);
```

Listing 3: MOF: Syntax – Method [vergl. Hobbs2004 S.75]

Auch bei der Methodendeklaration (Listing 3) können „Qualifiers“ zum Einsatz kommen. Im übrigen setzt sich die Methodendeklaration aus den Rückgabewert (siehe Tabelle 7), dem Methodennamen sowie einer Parameterliste zusammen. Ein Parameter setzt sich wiederum aus einem Datentyp und einen Parameternamen zusammen.

In der nachfolgenden Tabelle werden die verwendbaren „Qualifiers“ aufgelistet. Dabei wird jeweils ein Anwendungsbeispiel, eine Beschreibung und Anmerkungen, zu Besonderheiten bei der Verwendung, gegeben.

Qualifiers:	***Beispiel:***	***Beschreibung:***	***Anmerkung:***
`Abstract`	`[Abstract]`	Kennzeichnet, dass von der Klasse keine Instanzen erzeugt werden können. Es kann von der Klasse nur geerbt werden.	„Terminal“ und „Abstract“ (s.u.) schließen sich gegenseitig aus.
`Association`	`[Association]`	Kennzeichnet eine Klasse als Assoziation.	Innerhalb der Assoziations-Klasse werden die verbundenen Klassen mit REF gekennzeichnet.
`Deprecated`	`[Deprecated{“New_Class“}]`	Kennzeichnet eine Klasse als veraltet und verweist auf die alternative Klasse.	Klassen die als „Deprecatet“ gekennzeichnet sind, sollten in neuen Projekten nicht verwendet werden.
`Description`	`[Description (“Beschreibung“)]`	Einleitender Kommentar einer Komponente.	Kann z.B. zur Beschreibung von Klassen, Methoden und Propertys eingesetzt werden.
`Key`	`[Key]`	Kennzeichnet ein Property als Schlüssel. Ein Key muss Systemweit eindeutig sein.	Es kann in einer Klasse mehrere Key-Propertys geben. In diesem Fall stellt die Gesamtheit der Key's den eindeutigen Schlüssel dar.

`Max`	`[Max(6)]`
	`HAW_Room antecedent;`
	`HAW_Flat dependent;`
		Dient zur Begrenzung der Kardinalität zweier Elemente.	Im Beispiel wird festgelegt, das eine Wohnung maximal 6 Räume haben darf.

`Maxvalue`	`[Maxvalue(15)]`	Legt fest, welchen Maximalwert ein Property annehmen darf.	Es ist nicht sichergestellt, das dieser Wert nicht überschritten wird. Eine Überprüfung ist abhängig von der WBEM/CIM Implementierung.

`Min`	`[Min(2)]`
	`HAW_Room antecedent;`
	`HAW_Flat dependent;`
		Dient zur Begrenzung der Kardinalität zweier Elemente.	Im Beispiel wird festgelegt, das eine Wohnung mindestens 2 Räume haben muss.

`Propagated`	`[Propagated]`	Erlaubt eine Assoziation des nach genannten Property mit einem Property einer anderen	Durch diese Assoziation reduziert sich die Anzahl der

		Instanz.	eindeutigen Key's um dieses Property.
`Provider`	`[Provider("java:haw.iflat.providers.myprovider")]`	Legt fest welcher Provider für die Verwaltung der nachfolgenden Klassen zuständig ist.	Für einen in JAVA geschriebenen Provider ist die Angabe des Signalwortes „java:" gefolgt von der Java-Klasse erforderlich.
`Read`	`[Read]`	Kennzeichnet ein Property als lesbar.	Ist nur ein Marker und bewirkt nicht zwingend einen Schreibschutz.
`Required`	`[Required]`	Legt fest, das ein Property einen gültigen Wert haben muss.	Das Property darf nicht „null" sein.
`Terminal`	`[Terminal]`	Legt fest, dass von dieser Klasse nicht geerbt werden darf.	„Terminal" und „Abstract" (s.o.) schließen sich gegenseitig aus.
`Version`	`[Version("1.5.8")]`	Kennzeichnet die Versionsnummer.	Darf pro Klasse nur einmal Verwendung finden.
`Write`	`[Write]`	Kennzeichnet ein Property als beschreibbar.	Ist ein Marker wie „Read".

Tabelle 6: Auflistung Qualifiers

Zu beachten ist, dass Tabelle 6 nur einen Auswahl der möglichen Qualifiers darstellt. Für weitere Informationen siehe [DMTF2009] und [Hobbs2004 Kapitel 5].

Qualifiers können kombiniert werden, dazu werden sie innerhalb der eckigen Klammern, durch Komma getrennt, aufgelistet. Um z.B. die Kardinalität einer Assoziation festzulegen, können die Qualifiers „Min" und „Max" kombiniert werden (siehe Listing 4).

```
[Min(4), Max(6)]
HAW_Room antecedent;
HAW_Flat dependent;
```

Listing 4: MOF: Beispiel: mehrfache Qualifiers [vergl. Hobbs2004]

Listing 4 ist ein Auszug einer Assoziationsklasse. In der „Qualifiers"-Sektion wird festgelegt, dass die Klasse „HAW_Flat" genau 4, 5 oder 6 Instanzen der Klasse „HAW_Room" halten darf.

Datentyp:	***Beschreibung:***
`boolean`	Boolean
`uint8`	Unsigned 8-bit integer
`sint8`	Signed 8-bit integer
`uint16`	Unsigned 16-bit integer
`sint16`	Signed 16-bit integer
`uint32`	Unsigned 32-bit integer
`sint32`	Signed 32-bit integer
`uint64`	Unsigned 64-bit integer
`sint64`	Signed 64-bit integer
`string`	UCS-2 string
`real32`	4-byte floating-point value compatible with IEEE-754® Single format
`real64`	8-byte floating-point compatible with IEEE-754® Double format
`datetime`	A string containing a date-time
`<classname> ref`	Strongly typed reference
`char16`	16-bit UCS-2 character
`void`	Methode hat keinen Rückgabewert

Tabelle 7: Auflistung Datentypen [vergl. DMTF2009 S.22]

Listing 5 ist ein Auszug der in Kapitel 3.2 erwähnten Klasse „CIM_NumericSensor“. Die Qualifiers-Section für die Variable (Property) „BaseUnits“ ist besonders lang. Sie beginn mit der „Description“ einer Beschreibung welche Bedeutung diese Variable hat. Danach folgt mit „ValueMap“ eine Aufzählung aller Werte die die Variable annehmen kann. Gefolgt von „Values“, einer Map mit Bezeichnern, die jedem Wert der „ValueMap“ eine Bezeichnung zuordnet. Zum Schluss folgt der Datentyp (uint16) gefolgt vom Variablennamen (BaseUnits).

```
.
.
.

class CIM_NumericSensor : CIM_Sensor {

      [Description (
          "The base unit of the values returned by this Sensor. All "
          "the values returned by this Sensor are represented in "
          "the units obtained by (BaseUnits * 10 raised to the "
          "power of the UnitModifier). For example, if BaseUnits is "
          "Volts and the UnitModifier is -6, then the units of the "
          "values returned are MicroVolts. However, if the "
          "RateUnits property is set to a value other than \"None\", "
          "then the units are further qualified as rate units. In "
          "the above example, if RateUnits is set to \"Per Second\", "
          "then the values returned by the Sensor are in "
          "MicroVolts/Second. The units apply to all numeric "
          "properties of the Sensor, unless explicitly overridden "
          "by the Units qualifier." ),
       ValueMap { "0", "1", "2", "3", "4", "5", "6", "7", "8", "9",
          "10", "11", "12", "13", "14", "15", "16", "17", "18",
          "19", "20", "21", "22", "23", "24", "25", "26", "27",
          "28", "29", "30", "31", "32", "33", "34", "35", "36",
          "37", "38", "39", "40", "41", "42", "43", "44", "45",
          "46", "47", "48", "49", "50", "51", "52", "53", "54",
          "55", "56", "57", "58", "59", "60", "61", "62", "63",
          "64", "65", "66" },
       Values { "Unknown", "Other", "Degrees C", "Degrees F",
          "Degrees K", "Volts", "Amps", "Watts", "Joules",
          "Coulombs", //10
          "VA", "Nits", "Lumens", "Lux",
          "Candelas", "kPa", "PSI", "Newtons", "CFM", "RPM",
          //20
          "Hertz", "Seconds", "Minutes", "Hours",
          "Days", "Weeks", "Mils", "Inches", "Feet", "Cubic Inches",
          //30
          "Cubic Feet", "Meters",
          "Cubic Centimeters", "Cubic Meters", "Liters",
          "Fluid Ounces", "Radians", "Steradians", "Revolutions",
          "Cycles", //40
          "Gravities", "Ounces", "Pounds",
          "Foot-Pounds", "Ounce-Inches", "Gauss", "Gilberts",
          "Henries", "Farads", "Ohms", //50
          "Siemens",
          "Moles", "Becquerels", "PPM (parts/million)", "Decibels",
          "DbA", "DbC", "Grays", "Sieverts",
          "Color Temperature Degrees K", //60
          "Bits",
          "Bytes", "Words (data)", "DoubleWords", "QuadWords",
          "Percentage", "Pascals" },
       ModelCorrespondence { "CIM_NumericSensor.UnitModifier",
          "CIM_NumericSensor.RateUnits" }]
   uint16 BaseUnits;

.
.
.

};
```

Listing 5: MOF: CIM_NumericSensor (Auszug) [DMTF2010]

4. Modelle

In diesem Kapitel geht es um die Entwicklung eigener Modelle und deren Integration in das CIM-Modell.

Wie schon in Kapitel 2 beschrieben ist die Fülle an vorhandenen oder denkbaren Umgebungen, Geräten und Anwendungen unüberschaubar. Um einen Überblick zu bekommen (und zu behalten) ist es erforderlich, die Komponenten zu klassifizieren. Danach ist es, aus Management-Sicht, nicht mehr erforderlich sich mit der einzelnen Komponente auseinander zu setzen. Es wird nur noch von einer „Komponente der Klasse x“ gesprochen. (Diese Aussage stimmt nur bis zur Implementierung der Provider. Der Programmierer des Providers muss natürlich das zu behandelnde Objekt, bzw. dessen Klasse, detailliert kennen.)

Im folgenden wird die Struktur des zu verwaltenden Gebäudes analysiert. Mit Hilfe der gewonnenen Erkenntnisse werden die Modelle entworfen.

Dabei wird wie folgt vorgegangen:

- Analyse des behandelten Objekts
- Diskussion der Anknüpfungspunkte an CIM
- UML- Diagramm(e) entwickelt
- UML- Diagramm(e) in MOF übersetzen

Um das eigene Modell an das CIM Schema anbinden zu können, muss eine Klasse innerhalb des CIM Schema gefunden werden, die als Superklasse für das eigene Modell dient. Die Suche nach einer geeigneten Klasse kann sehr langwierig sein. Es gibt viele mögliche Wege der Anknüpfung. Zum Beispiel könnte dies durch direktes Erben von „CIM_ManagedElement“ (Core Schema) geschehen. Ob es eine gute Designentscheidung ist, dass eigene Modell an der Wurzel des „Core Schema“ (siehe Kapitel 3.1) zu platzieren, sollte in Frage gestellt werden. Mit Sicherheit gibt es bessere Anknüpfungspunkte im „Core-Schema“ oder einem „Common-Schema“ (siehe Kapitel 3.2). Erfolglos kann die Suche letztlich nicht verlaufen, denn es lässt sich im Extremfall eben doch von der Wurzel des Core Schema („CIM_ManagedElement“) erben.

Bei der Suche nach einer geeigneten Superklasse ist es hilfreich, dass die Common Schemata in Einsatzgebiete (Device, User, .Security, …) unterteilt sind. Leider macht eine große Anzahl an Querverweisen, zwischen unterschiedlichen Dokumenten der DMTF, das Modell sehr unübersichtlich.

Um diesen Anknüpfungspunkt zu finden, ist es erforderlich das zu Verwaltende Objekt auf seine Eigenschaften und Besonderheiten zu untersuchen. Die dabei gewonnenen Informationen, müssen mit den Beschreibungen der CIM-Klassen [DMTF2010] verglichen werden. Die Suche kann, auf Grund der Komplexität des CIM-Schemas, mit unzähligen Querverweisen und seitenlangen Beschreibungen, recht mühsam sein.

Sobald eine Übereinstimmung gefunden wurde, hat sich die Mühe gelohnt. Die Klasse mit den meisten Übereinstimmungen, ist ein Kandidat für die Anknüpfung unseres Objekts. Es ist durchaus möglich, dass das eigene Objekt zu mehreren CIM-Klassen passt. In diesem Fall ist die Auswahl der Klasse eine reine Designentscheidung.

Beim erstellen der Modelle, müssen die Namenskonventionen der DMTF eingehalten werden. Alle durch die DMTF entwickelten Klassen, sind durch ein dem Klassennamen vorangestelltes „CIM_“ gekennzeichnet. Um Verwechslungen mit anderen, bestehenden und zukünftigen Klassen zu vermeiden, werden bei Eigenentwicklungen jedem Klassennamen ein eigenes Kürzel vorangestellt. Im Rahmen dieser Arbeit findet das Kürzel der „Hochschule für Angewandte Wissenschaften“ (HAW), gefolgt von einem Unterstrich („HAW_“) Verwendung. In den Modellen werden vorzugsweise englische Bezeichnungen verwendet (also HAW_Building statt HAW_Haus).

Im folgenden wird die oben geforderte Strukturanalyse des Hauses durchgeführt. Diese wird „Bottom-up“, vom inneren des Hauses bis zur Draufsicht, durchgeführt.

Betrachtet wird das Gebäude in Abbildung 20. Die Grundrisse (Abbildung 17 & 18) lassen mehrere Gruppierungen von Möbeln (z.B. Essbereich / Wohnbereich) innerhalb von Räumen erkennen. Diese Bereiche stellen in dieser Betrachtung den Bottom dar. Für diese Bereiche wird der Name „Funktionsbereich“ (Zone) definiert.

Legt man ein Augenmerk auf die „Räume“ in Abbildung 17 & 18, so ist zu erkennen, dass sich diese zum Teil in ihrer Ausstattung unterscheiden. Die „Räume“ dienen verschiedenen Aktivitäten der Bewohner und können anhand ihrer Eigenschaften klassifiziert werden. Die „Räume“ (Rooms) werden als die nächste Ebene, oberhalb der „Funktionsbereiche“, definiert.

Die Gesamtheit der „Räume“ wird hier als „Wohnung“ (Flat) definiert. Diese Sichtweise ist in dieser Betrachtung eigentlich überflüssig, denn es gibt keine weitere Wohnung. Um das System auch auf Mehrfamilienhäuser anwenden zu können, wird diese Sichtweise aber beibehalten.

Das „Haus“ (Building) ist als oberste Ebene zu nennen. Es stellt die Hülle dar, die alle Komponenten zu einem Ganzen zusammenfasst.

Zusammengefasst, lassen sich folgende Strukturen erkennen:

- Funktionsbereich (Zone)
- Raum (Room)
- Wohnung (Flat)
- Haus (Building)

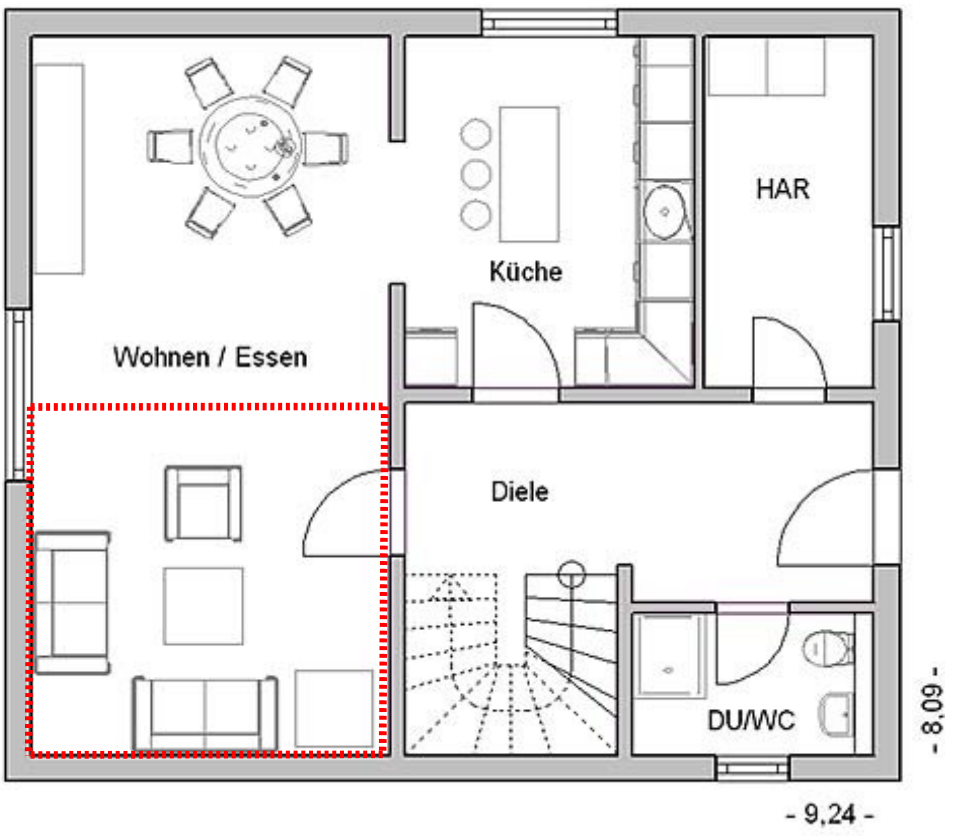
Anhaltspunkte zur Entwicklung eigener Modelle gibt [Hobbs2004 – Kapitel 9], hier wird exemplarisch die Implementierung einer Telefonanlage beschrieben. Auch [Beckmann2006] gibt interessante Anhaltspunkte. Hier wird die Einbindung einer LDAP-Kaffeemaschine in WBEM beschrieben. Leider sind hier die Quellen (Wikipedia und co.) zweifelhafter Natur. Dennoch beinhaltet der Kern diese Ausarbeitung interessante Ansätze.

In den nachfolgenden Unterkapiteln, werden die gefundenen Komponenten näher betrachtet. Dabei wird mit der untersten Ebene begonnen. Danach wird die jeweils darüber liegende Ebene betrachtet.

4.1 Funktionsbereich (Zone)

Unter einem Funktionsbereich (Zone) versteht man einen logisch abgegrenzten Bereich innerhalb eines Raumes, der auf Grund seiner Ausstattung einem bestimmten Zweck dient.

Räume sind häufig in noch kleinere Bereiche unterteilt. Zum Beispiel hat das „Wohn- und Esszimmer“ einen Bereich zum Essen, eine Wohnlandschaft mit Sofa, Sessel und



Zeichnung 1: Beispiel für eine Zone

Couchtisch und möglicherweise einen Hifi- und TV- Bereich. Diese logischen Unterteilungsebenen, die hier Zonen genannt werden, stellen die unterste Ebene dar. Der Wohnbereich in Zeichnung 1 ist, als Beispiel für eine Zone, gekennzeichnet.

Für die Anbindung der Zone an das CIM Schema, muss eine passende Klasse des CIM Schema gefunden werden, von der geerbt werden kann. Wie das nachfolgende Unterkapitel zeigt, ist die Zone ein Bestandteil eines Raumes. Damit ist die Zone, mit ihren Komponenten, ein Subsystem innerhalb eines größeren Systems. So ist es möglich, die Zone über die Klasse „CIM_ManagedSystemElement“ (siehe Tabelle 4 auf Seite 28), an das Modell anzubinden.

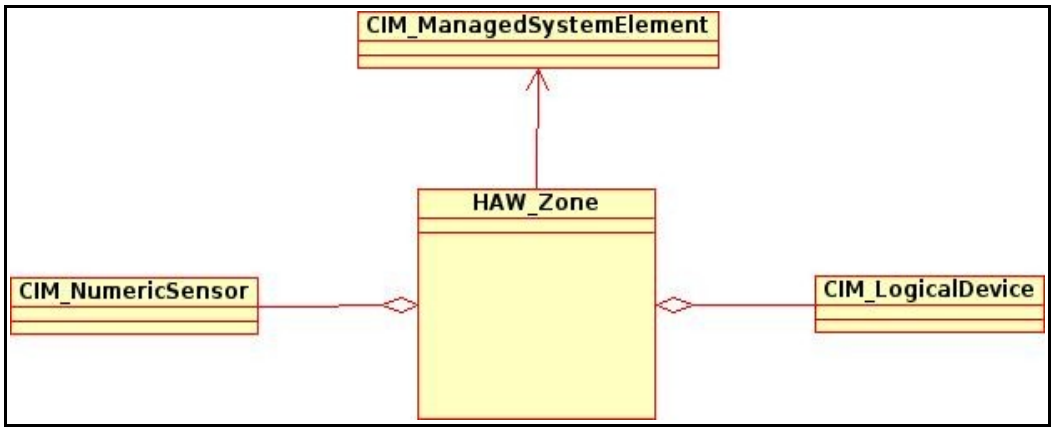


Abbildung 15: UML: HAW_Zone

Ein Funktionsbereich kann eine Vielzahl an Geräten und Sensoren beherbergen, aber auch gar keine! Für Sensoren gibt es eine entsprechende Klasse, innerhalb des CIM Schema („CIM_NumericSensor“), so dass die Modellierung einer eigenen Klasse entfallen kann. Die Verwendung von „CIM_NumericSensor“ wird im Unterkapitel 4.2 (Raum) erneut behandelt.

Für die Anbindung der Aktoren steht leider keine vorgefertigten Klassen zur Verfügung. Hier gibt es die Möglichkeit die Aktoren über die Klasse „CIM_LogicalDevice“ anzubinden. Dies wird ebenfalls in Unterkapitel 4.2 erneut aufgegriffen.

```
[Version("1.0.0"), Description("Bildet die Eigenschaften eines Raum-Teilbereiches ab, dient "
                               "als Hülle für die Komponenten eines Funktionsbereiches.")]
class HAW_Zone : CIM_ManagedSystemElement {

   [Key, Description("Zonen-Key")]
   string ZoneID;
};

//-------------------------------------------------------------------------------------

[Association, Description("Assoziation der Klasse HAW_Zone mit CIM_NumericSensor.")]
class HAW_ZoneSensor{

   [Key, Description("Key")]
   string ZoneSensorID;

   CIM_NumericSensor REF Sensor;
   HAW_Zone REF Zone;
};

//-------------------------------------------------------------------------------------

[Association, Description("Assoziation der Klasse HAW_Zone mit CIM_LogicalDevice.")]
class HAW_ZoneDevice{

   [Key, Description("Key")]
   string ZoneDeviceID;

   CIM_LogicalDevice REF Device;
   HAW_Zone REF Zone;
};
```

Listing 6: MOF: HAW_Zone

Listing 6 bildet das UML-Diagramm in Abbildung 15 als MOF-Datei ab. Das MOF-Listing enthält die Klasse „HAW_Zone“, sowie die beiden Assoziationen „HAW_ZoneSensor“ und „HAW_ZoneDevice“.

Kapitel 4.2 behandelt die nächsthöhere Ebene, den Raum.

4.2 Raum (Room)

Unter einem Raum versteht man einen durch Wände abgegrenzten Bereich, einer Wohnung (Flat). Ein Raum kann einem bestimmten Zweck dienen oder universell Verwendung finden.

Der Raum ist die nächste Ebene, oberhalb der Zone. Es gibt Räume mit unterschiedlichen Ausstattungen, die je nach Ihrem Einsatzgebiet unterschiedlich genutzt werden. In unseren Beispiel finden sich:

- Wohn- / Esszimmer
- Küche
- Hausarbeitsraum (HAR)
- Dusche/WC
- Diele (Flur EG)
- Schlafzimmer
- Kinderzimmer 1
- Kinderzimmer 2
- Badezimmer
- Galerie (Flur OG)

Alle Räume, unabhängig von ihrem Einsatzgebiet, müssen mit einer Grundausstattung an Sensoren und Aktoren bestückt werden, damit eine minimale Überwachung und Steuerung möglich wird.

Für den Anfang werden in jedem Raum nur die folgenden Sensoren angenommen:

Sensor-Typ:	***Beschreibung:***
Anwesenheits-Sensor	Erfasst, ob sich eine Person im Raum befindet.
Temperatur-Sensor	Erfasst die aktuelle Raumtemperatur.
Luftfeuchtigkeits-Sensor	Erfasst die aktuelle Raumluft-Feuchte.
Helligkeits-Sensor	Erfasst den aktuellen Lichteinfall des Raumes.

Tabelle 8: Auflistung der Sensoren

Bei der Suche nach einer Anknüpfungsmöglichkeit der Sensoren an das CIM-Schema, stellt das Common-Schema „Devices“ eine geeignete Klasse zur Verfügung. Die Klasse „CIM_NumericSensor“ kann dazu genutzt werden die Sensoren im Modell abzubilden. Dafür stellt die Klasse eine große Auswahl an Basiseinheiten (BaseUnits) (z.B. „Degrees C“ - für die Temperatur) bereit (siehe Listing 5 Seite 37).

Die Klasse CIM_NumericSensor in Abbildung 16 hat vier Unterklassen:

- HAW_LightSensor
- HAW_HygroSensor
- HAW_PresenceSensor
- HAW_TempSensor

Die Designentscheidung für diese vier von „CIM_NumericSensor“ abgeleiteten Klassen ist aus dem Wunsch nach eindeutiger Zuordnung, der Klassen zu Ihren entsprechenden Sensoren, entsprungen. Laut „DMTF – Device Model White Paper“ [DMFT2003, Kapitel 2.20] sollte auf die Verwendung von Unterklassen, der Klasse „CIM_NumericSensor“, zugunsten der Kompatibilität verzichtet werden. In „CIM_NumericSensor“ wird der Sensor-Typ durch setzen eines Attributs (uint16 BaseUnits) [Datentypen siehe Kapitel 3.4] festgelegt. Den Sensortyp nur anhand eines Attributes erkennen zu können, macht die Darstellung in einem Modell intransparent. Um die Lesbarkeit der Modelle zu erhalten, werden die eingeführten Unterklassen von „CIM_NumericSensor“ vorerst beibehalten. (Um der Empfehlung der DMTF zu folgen, müssten den Instanzen, der Klasse „CIM_NumericSensor“, prägnante Namen zugewiesen werden.)

Als Aktoren stehen zur Verfügung:

Aktor-Typ:	***Beschreibung:***
Heizkörper	Ermöglicht das Aufheizen der Raumluft. (z.B. durch Fußbodenheizung, Warmwasserradiator oder elektrischer Heizlüfter)
Belüftung	Ermöglicht das Steuern der Fischluftzufuhr. (z.B. durch Fensteröffner, Lüfter-Steuerung oder Klimaanlage)
Leuchtmittel	Beleuchtung des Zimmers. (z.B. durch Wand-/Decken-/Stehlampen, LED-Tapete oder eingeleitetes Tageslicht)
schaltbare Steckdosen	Ermöglichen das gezielte Ein-/Ausschalten von Geräten.

Tabelle 9: Auflistung der Aktoren

Die Anbindung der Aktoren ist leider nicht so einfach wie bei den Sensoren. Für die in Tabelle 9 aufgelisteten Aktoren gibt es keine direkte Entsprechung im CIM Schema. Das liegt daran, dass diese Komponenten keine typischen Bauteile eines IT-Systems sind. Ein Ansatz um diese Komponenten dennoch anzubinden, ist die Klasse „CIM_LogicalDevice“. Die Klassenbeschreibung kann Tabelle 4 auf Seite 28 entnommen werden. Prinzipiell kann diese Klasse sowohl für physikalisch existierende, als auch für virtuelle Geräte eingesetzt werden. Die von ihr abgeleitete Klasse, des jeweiligen Gerätes, beherbergt dessen Eigenschaften und Konfigurationseinstellungen.

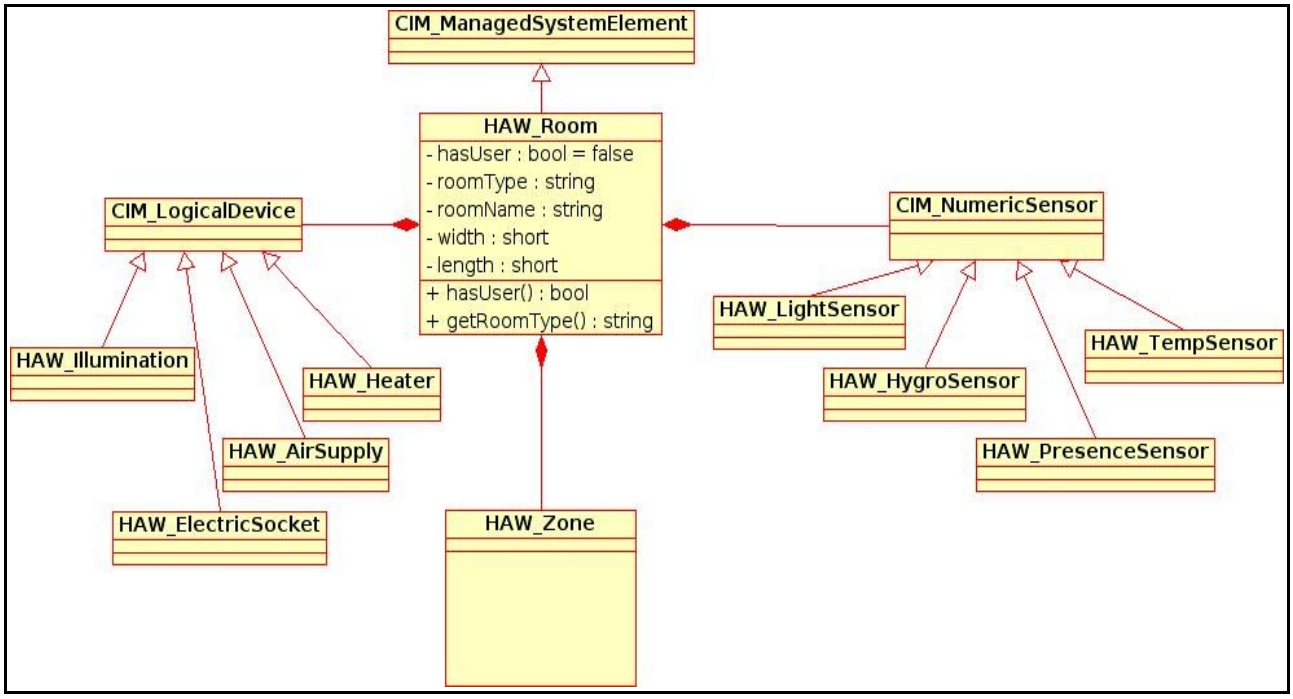


Abbildung 16: UML: HAW_Room

Listing 7 stellt die im UML-Diagramm (Abbildung 16) dargestellten Klassen als MOF dar. Das MOF-File beinhaltet die Klasse „HAW_Room“ als Kind der Klasse „CIM_ManagedSystemElement“. Darin enthalten sind die Propertys und Methoden des UML-Diagramms. Darunter befinden sich die Assoziationen zwischen der Klasse „HAW_Room“ und „CIM_LogicalDevice“, „CIM_NumericSensor“ bzw. „HAW_Zone“. Die von „CIM_LogicalDevice“ und „CIM_NumericSensor“ abgeleiteten Klassen sind im MOF-File nicht dargestellt. Sie müssten bei Bedarf ergänzt werden.

```
[Description("Bildet die Eigenschaften eines Raumes ab, dient "
             "als Hülle für die Komponenten eines Raumes")]

class HAW_Room : CIM_ManagedSystemElement {

   [Key]
   string roomID;

   string roomName;
   string roomType;

   [Description("Raumbreite")]
   uint16 width;

   [Description("Raumlänge")]
   uint16 length;

   boolean userPresent;

   boolean hasUser();

   string getRoomType();
};

//--------------------------------------------------------------------------

[Association, Description("Assoziation von HAW_Room mit CIM_NumericSensor")]
class HAW_RoomSensor{

   [Key]
   string roomSensorID;

   CIM_NumericSensor REF sensor;
   HAW_Room REF room;

};

//--------------------------------------------------------------------------

[Association, Description("Assoziation von HAW_Room mit CIM_LogicalDevice")]
class HAW_RoomDevice{

   [Key]
   string roomDeviceID;

   CIM_LogicalDevice REF device;
   HAW_Room REF room;

};

//--------------------------------------------------------------------------

[Association, Description("Assoziation von HAW_Room mit HAW_Zone")]
class HAW_RoomZone{

   [Key]
   string roomZoneID;

   HAW_Zone REF zone;
   HAW_Room REF room;

};
```

Listing 7: MOF: HAW_Room

Das nächste Unterkapitel befasst sich mit der nächsthöheren Ebene, der Wohnung.

4.3 Wohnung (Flat)

Eine Wohnung ist ein Raumverbund. Es werden Räume mit unterschiedlichen Funktionen und Aufgaben, zu einem Ganzen zusammengefasst. Eine Wohnung verfügt über mindestens einen Eingang und besteht aus mindestens einem Raum. Die hier betrachtete Wohnung hat 10 Räume, die auf 2 Ebenen verteilt sind (siehe Abbildung 17 + 18). Eine Auflistung der Räume wurde in Kapitel 4.2 gegeben.

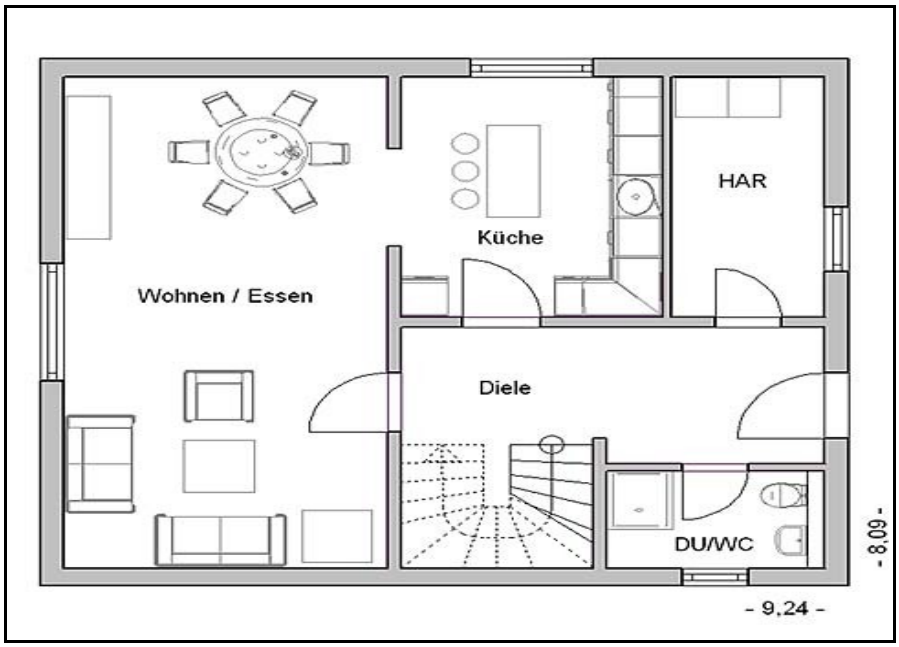


Abbildung 17: Haus: Grundriss Erdgeschoss [PRO2010]

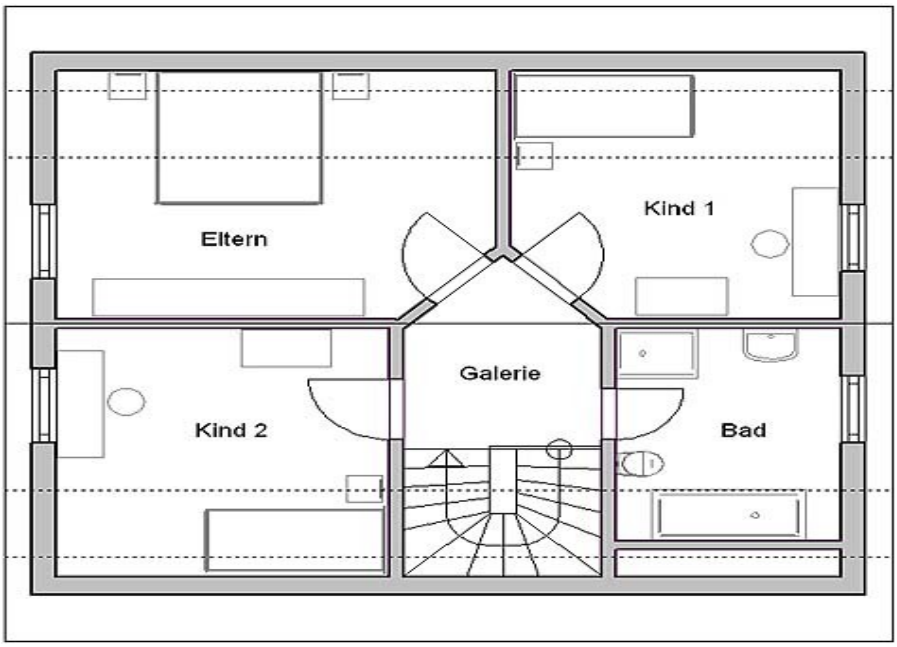


Abbildung 18: Haus: Grundriss Dachgeschoss [PRO2010]

Die Wohnung stellt die „Verpackung“ für die Räume dar. In der Klasse „HAW_Flat“ könnten die Räume selbst, deren Position und die Durchgänge zwischen zwei Räumen gespeichert werden. Diese Informationen werden vorerst ausgelassen, können zu einem späteren Zeitpunkt jedoch wichtig werden. Zum Beispiel dann, wenn sich „Personen“ oder „Agenten“ zwischen Räumen bewegen. Für diesen Fall müssen diese Informationen nachgepflegt werden.

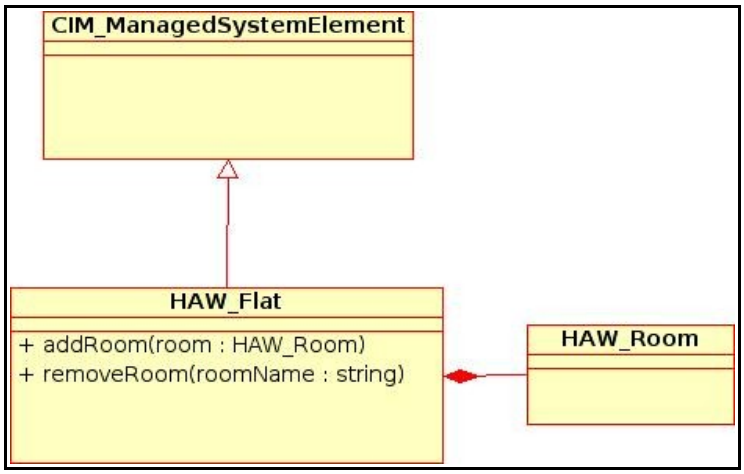


Abbildung 19: UML: HAW_Flat

```
[Description("Bildet die Eigenschaften einer Wohnung ab, dient "
             "als Hülle für die Komponenten eine Wohnung")]

class HAW_Flat : CIM_ManagedSystemElement {

   [Key]
   string flatID;

   string flatName;

   void addRoom(HAW_Room room);

   void removeRoom(string roomname);

};

//---------------------------------------------------------------------------

[Association, Description("Assoziation von HAW_Flat mit HAW_Room")]
class HAW_FlatRoom{

   [Key]
   string flatRoomID;

   HAW_Room REF room;
   HAW_Flat REF flat;

};
```

Listing 8: MOF: HAW_Flat

Listing 8 bildet die in Abbildung 19 gegebene Klasse und deren Assoziationen ab.

Kapitel 4.4 befasst sich mir der obersten Ebene, dem Haus.

4.4 Haus (Building)

Mit den Haus sind wir auf der obersten Ebene unserer „Bottom-up“-Betrachtung angelangt. Unter einen Haus versteht man eine Gebäude, das aus einen Fundament, 4 Wänden und einem Dach besteht. Es hat mindestens einen Eingang und in der Regel mehrere Fenster oder andere Belüftungssysteme.

Das Gebäude steht auf einem Grundstück, in einer Gemeinde und wurde von dieser Erschlossen. Seit der Erschließung stehen die folgenden Versorgungseinrichtungen zur Verfügung:

- Trinkwasser
- Abwasser
- Strom
- Telefon
- Internet
- Gas

Das Haus hat zwei Bereiche die getrennt betrachtet werden können:

- das Umfeld
- das Hausinnere

Das Umfeld ist alles was das Gebäude umgibt wie z.B:

- Luftdruck
- Temperatur
- Luftfeuchtigkeit
- Windgeschwindigkeit
- Geräusche
- Licht
- Regen
- ...

Im Inneren beherbergt das Haus die vorbenannten Objekte (siehe Kapitel 4.1 bis 4.3), welche ihrerseits Objekte halten. Das Haus selbst ist ein System, das aus einer Vielzahl von Subsystemen besteht. Die Summe der Systeme, innerhalb des Hauses, ermöglichen in ihrer Gesamtheit mehr Dienste, als es die Einzelsysteme anbieten könnten. Damit ist das Haus ein Kandidat für die Superklasse „CIM_System“. Die Klassenbeschreibung finden sie in Tabelle 4 auf Seite 28.

Abbildung 20: Haus: Außenansicht [PRO2010]

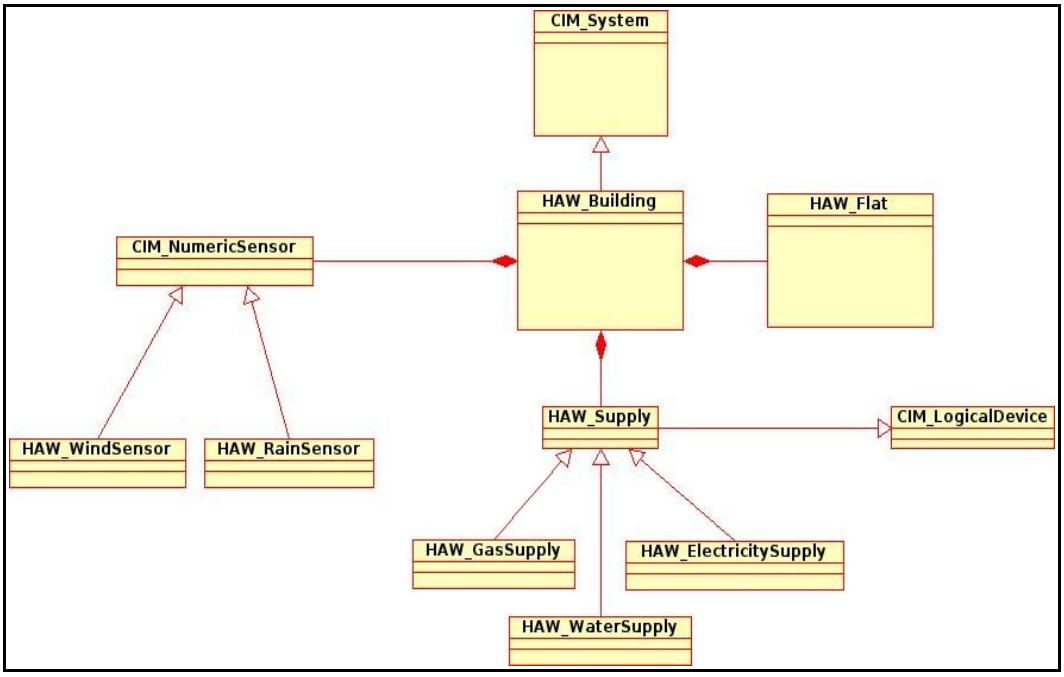


Abbildung 21: UML: Building

```
[Description("Bildet die Eigenschaften eines Gebäudes ab, dient "
             "als Hülle für die Komponenten eine Hauses")]

class HAW_Building : CIM_System {

string buildingID;

string buildingName;

};

//-------------------------------------------------------------------------

class HAW_Supply : CIM_LogialDevice{

   [Key]
   string supplyID;
}

//-------------------------------------------------------------------------

class HAW_GasSupply : HAW_Supply{

   [Key]
   string gasSupplyID;
}
//-------------------------------------------------------------------------

class HAW_WaterSupply : HAW_Supply{

   [Key]
   string waterSupplyID;
}

//-------------------------------------------------------------------------

class HAW_ElectricitySupply : HAW_Supply{

   [Key]
   string electricitiSupplyID;
}

//-------------------------------------------------------------------------

[Association, Description("Assoziation von HAW_Building mit HAW_Flat")]
class HAW_BuildingFlat{

   [Key]
   string buildingFlatID;

   HAW_Flat REF flat;
   HAW_Building REF building;

};

//-------------------------------------------------------------------------

[Association, Description("Assoziation von HAW_Building mit HAW_Supply")]
class HAW_BuildingSupply{

   [Key]
   string buildingSupplyID;

   HAW_Supply REF supply;
   HAW_Building REF building;

};

//-------------------------------------------------------------------------

[Association, Description("Assoziation von HAW_Building mit CIM_NumericSensor")]
class HAW_BuildingSensor{
```

```
   [Key]
   string buildingSensorID;

   CIM_NumericSensor REF sensor;
   HAW_Building REF building;

};
```

Listing 9: MOF: HAW_Building

Abbildung 21 zeigt das Haus („HAW_Building“) mit seiner Superklasse „CIM_System“. Das Haus hat jeweils mindestens eine(n):

- Sensor „CIM_NumericSensor“
- Ressourcenversorgung „HAW_Supply“
- Wohnung „HAW_Flat“

Das MOF-File (Listing 9) definiert die Klasse „HAW_Building“ als Kind der Klasse „CIM_System“. Gefolgt von den Assoziationen zwischen der Klasse „HAW_Building“ und den in Abbildung 21 dargestellten Klassen.

Im nächsten Unterkapitel folgt eine zusammenfassende Übersicht des Modells.

4.5 Übersicht

In diesem Kapitel werden die in vorangegangenen Modelle zu einem Gesamtmodell zusammengefügt.

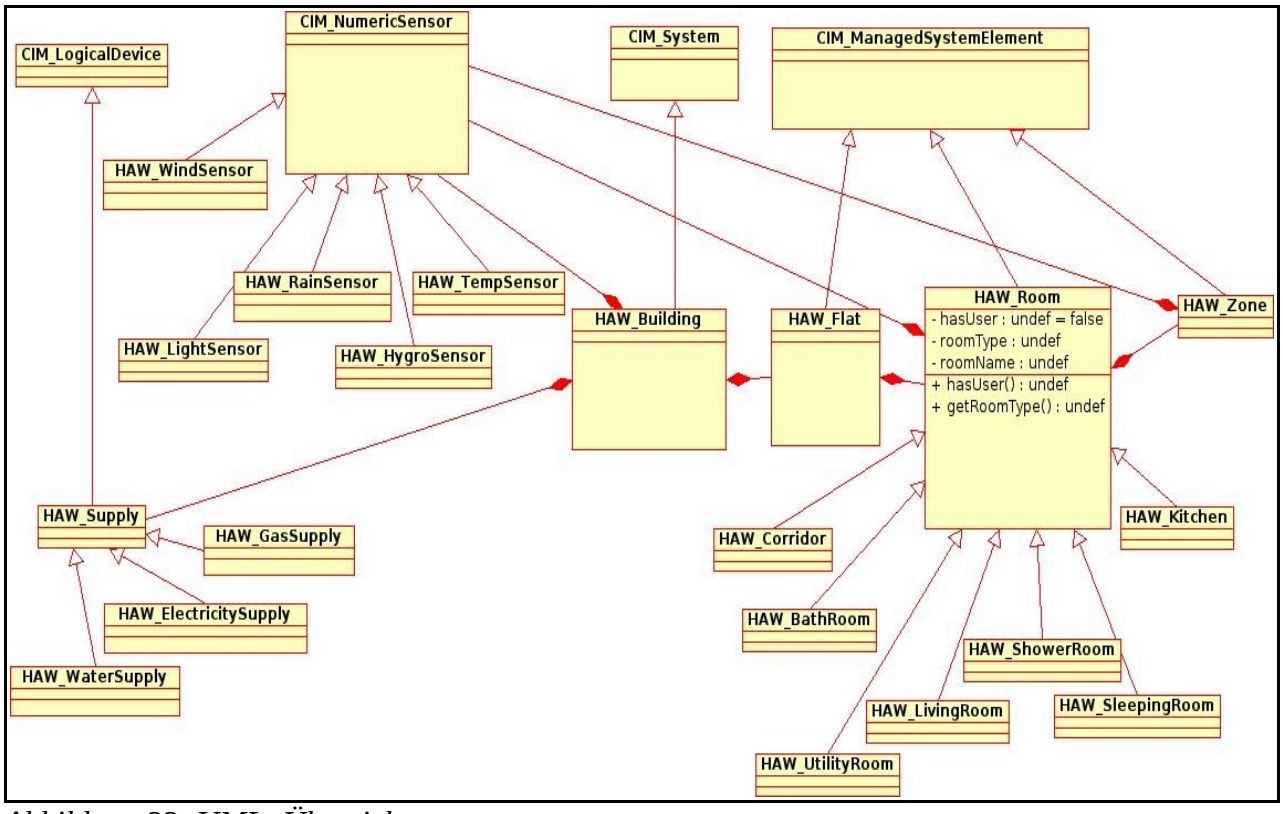


Abbildung 22: UML: Übersicht

Abbildung 22 gibt einen Überblick, über die Beziehungen der in Kapitel 4.1 bis 4.4 vorgestellten Strukturen. Um die Anknüpfungspunkte an das CIM Schema leichter erkennen zu können, wurden die dem CIM Schema „entnommenen“ Superklassen an den oberen Rand der Diagramms geschoben.

Das Modell ist damit vorläufig fertiggestellt.

Das Kapitel 5. befasst mit den Interaktion zwischen Sensoren, Aktoren und Systemen.

5. Szenarien

In diesem Kapitel wird nach Ansätzen gesucht, um mit Hilfe der in Kapitel 4 gefundenen Modelle, eine Energieeinsparung zu erreichen.

Um Ansätze zum einsparen von Energie zu ermitteln, werden Szenarien verwenden. Szenarien sind Abläufe, die in einem wohl definierten Bereich stattfinden und die durch Aktoren (Personen, Systeme) beeinflusst werden. Als Beispiel-Szenarium wird hier ein Tagesablauf in einem Privathaushalt (im Gebäude aus Kapitel 4) gewählt!

Die in den Szenarien verwendete Variablen, werden durch spitze (>Variable<) Klammern gekennzeichnet. Die Werte dieser Variablen können sich Kontextabhängig ändern (z.B. durch: Werktag/Wochenende, Arbeitstag/Urlaub, Jahreszeit, Tageszeit, Witterung, benutzerdefinierte Festlegung, ...). Durch die Veränderung der Variablen ändern sich z.B. die Umschaltzeitpunkte zwischen Szenarien oder die Höhe der Temperatur die ein Raum zu einem bestimmten Zeitpunkt annimmt. Die Tatsache, dass Mess- und Regelungsaufgaben vollzogen werden, bleibt davon unberührt.

Sensor-Events werden fett und kursiv dargestellt (***Sensor***). Aktoren werden kursiv und unterstrichen abgebildet (*Aktor*).

Aktivitäten auf gleicher Ebene erfolgen parallel.

Beispiel:

- Event
  - Aktivität 1
    - Unteraktivität 1.1
  - Aktivität 2
    - Unteraktivität 2.1
  - Aktivität 3
    - Unteraktivität 3.1

Im Beispiel werden, nach Auslösung eines Events, die Aktivitäten 1-3 parallel ausgeführt. Kommt es innerhalb eines Szenario zu widersprüchlichen Aktivitäten (z.B. Heizen und Lüften ohne Heizen), so ist das deaktivieren eines Aktor dem aktivieren vorrangig.

Die Szenarien berücksichtigen, dass das richtige Lüften zu einer Verbesserung des Raumklimas führt. Durch eine Verminderung der Luftfeuchtigkeit kann das Kälteempfinden der Bewohner gemindert werden. Dieser Umstand führt dazu, dass die Raum-Temperatur abgesenkt werden kann. Das Absenken der Raum-Temperatur wiederum führt zu einer Energieeinsparung.

Der zweite positive Effekt des Lüftens, ist das Vermindern von Feuchtigkeitsniederschlägen, durch Taupunkt-Unterschreitung, an Wände und Fenstern. Damit kann Schimmelpilz-Bildung vorgebeugt werden. Für weitere Informationen siehe [VZBu2008] und [UBA2003].

Kapitel 5.1 befasst sich mit dem Tagesanbruch.

5.1 Tagesanbruch

In diesem Szenarium geht es um den Tagesbeginn, der zu einem benutzerdefinierten Zeitpunkt >Tagesanbruch< oder einem Event (z.B. erreichen ***Außenhelligkeit***) beginnt. Dabei ist es unerheblich welches Event zuerst eintritt.

- Zeitevent ***Timer*** == >Tagesanbruch< (z.B. 5:00h) oder Sensor-Event ***Außenhelligkeit*** == >Schwellenwert Außenhelligkeit<.
  - *Heizung*
    - Brauchwasser auf <Tagtemperatur> + <Grundtemperatur> erwärmen
    - Heizungskreislauf auf <Tagtemperatur> + <Grundtemperatur> aufheizen
  - *Belüftung*
    - Ist die ***Außentemperatur*** und die ***Außenluftfeuchtigkeit*** größer als die ***Innentemperatur*** und ***Innenluftfeuchtigkeit*** ?
      - ja → keine Belüftung durchführen
      - nein → Raum belüften
        - *Belüftung* einschalten bis ***Innentemperatur*** == >min Temperatur< oder ***Innenluftfeuchtigkeit*** zwischen >max Innenfeuchtigkeit< und >min Innenfeuchtigkeit<
  - Wenn *Heizung* und *Belüftung* abgeschlossen → zu „Tagbetrieb“ wechseln.

5.2 Tagbetrieb

Der „Tagbetrieb“ soll der Standardzustand bei Anwesenheit und Aktivität mindestens einer Person abbilden. Dabei wird zum einen überwacht, ob weiterhin eine Person aktiv und anwesend ist. Zum anderen wird die Luftfeuchtigkeit und die Raumtemperatur auf vordefinierten Werten gehalten.

- Alle >Intervall< Minuten prüfen:
  - ***Innentemperatur*** == >Normaltemperatur<
    - nein → *Heizkörper* einschalten bis ***Innentemperatur*** == >Normaltemperatur<
    - ja → *Heizkörper* abschalten
  - Ist die ***Innenluftfeuchtigkeit*** kleiner als >max Innenfeuchtigkeit<?
    - ja → keine Belüftung durchführen
    - nein
      - *Heizkörper* abschalten
      - *Belüftung* einschalten bis ***Innentemperatur*** == >min Temperatur< oder ***Innenluftfeuchtigkeit*** zwischen >max Innenfeuchtigkeit< und >min Innenfeuchtigkeit<
- ***Anwesenheitssensor*** → Aktivitäten in der Wohnung
  - Person in Wohnung?
    - ja → „Tagbetrieb“ fortsetzen
    - nein → zu „Sparbetrieb“ wechseln

5.3 Tagesende

Das „Tagesende“ setzt ein, wenn ein vordefinierter Zeitpunkt >Tagesende< erreicht wurde. Es wird geprüft, ob noch Personen aktiv sind. Unter aktiv wird verstanden, dass sich mindestens eine Person in der Wohnung befindet und diese Aktivitäten (Bewegungen) nachgehen. Werden keine Aktivitäten erkannt, wird davon ausgegangen, dass die anwesenden Personen schlafen.

- Zeitevent ***Timer*** == >Tagesende<
  - ***Anwesenheitssensor*** → Personen aktiv?
    - nein
      - auf „Sparbetrieb“ schalten
    - ja
      - aktuellen Zustand beibehalten, alle >Intervall< Minuten erneut prüfen

5.4 Sparbetrieb

Der Sparbetrieb setzt ein, sobald sich keine Person mehr in der Wohnung befindet. Er sorgt dafür, dass in allen Räumen die Beleuchtung, Belüftung, Heizung und alle Geräte die nicht einer Dauerversorgung bedürfen abgeschaltet werden. Auch die Temperatur der Heizung wird abgesenkt, um Ressourcen zu schonen. Während des Sparbetriebs wird aus Energiespargründen auf das Lüften verzichtet.

- *Leuchtmittel* abschalten
- *Belüftung* abschalten
- alle *schaltbare Steckdose* ausschalten
- *Heizkörper* abschalten solange ***Innentemperatur*** größer >Spartemperatur<
- *Heizung*
  - Brauchwasser → aktive Erwärmung einstellen
  - Heizungskreislauf → auf >Spartemperatur< + >Grundtemperatur< absenken
- ***Anwesenheitssensor*** → Person aktiv?
  - ja
    - Ist **Timer** zwischen >Tagesanbruch< und >Tagesende< → zu „Tagbetrieb“ wechseln
    - Ist **Timer** zwischen >Tagesende< und >Tagesanbruch< → zu „Tagesende“ wechseln
  - nein
    - Sparbetrieb fortsetzen

Das Kapitel 6 befasst sich mit der Auswahl eines geeigneten WBEM-Projektes.

6. WBEM-System implementieren

Nachdem sich die vorausgehenden Kapitel sehr theoretisch mit dem Thema WBEM/CIM befasst haben, wird dieses Kapitel einen praktischen Einblick in die Implementierung eines WBEM-Systems geben.

Wie bereits in Kapitel 2 hingewiesen, stellt das „Web-Based Enterprise Management“ nur ein Modell ohne konkrete Implementierung dar. Um ein WBEM konformes System zu erstellen gibt es zwei Möglichkeiten:

- Die WBEM/CIM-Modelle von Grund auf selber in Software umsetzen.
- Sich einer „fertigen“ WBEM/CIM-Software zu bedienen.

Der erste Punkt dürfte ein sehr langwieriger Prozess sein, der die Ressourcen eines normalen Projektes sprengen dürfte.

Punkt zwei stellt den Entwickler vor die Herausforderung, eine für das eigene Projekt geeignete, Software zu finden. Es gibt eine Vielzahl von Projekten, die sich mit der Implementierung von WBEM/CIM befassen. Tabelle 10 gibt einen Überblick der bekanntesten Projekte.

Software:	***Herausgeber:***	***Sprache:***	***Homepage:***	***Version:***	***Lizenz:***
OpenPegasus	The Open Group	C++	http://www.openpegasus.org/	2.9.2 vom 13.10.2010	open-source
OpenWBEM	OpenWBEM Project	C++	http://openwbem.sourceforge.net/	3.2.2 vom 19.10.2006	open-source
WBEM Services - Java™ Web Based Enterprise Management	Sun Microsystems	Java	http://wbemservices.sourceforge.net/	1.0.2 vom 01.11.2004	open-source
Windows Management Instrumentation (WMI)	Microsoft	VBS	http://msdn.microsoft.com/de-de/library/bb742445.aspx	1.5 vom 02.11.2000	kommerziell

Tabelle 10: WBEM-Projekte

Durch Recherchen im Internet lassen sich noch andere Projekte finden, die jedoch bereits eingestellt wurden. Daher sind diese Projekte in Tabelle 10 nicht aufgeführt.

OpenPegasus ist das Projekt, welches von allen aufgelisteten am aktuellsten ist. Die letzte Release (2.9.2) wurde am 13.10.2010 herausgegeben. OpenPegasus ist eine C++ Implementierung, der Quellcode kann von der Projekt-Homepage bezogen werden. Der Quellcode lässt sich, einen C-Compiler vorausgesetzt, unter Windows und Linux ohne Probleme kompilieren. OpenPegasus ist auch das Projekt, dass für den Praxisteil von [Hobbs2004] verwendet wird.

OpenWBEM ist wie OpenPegasus ebenfalls eine C++ Implementierung. Leider wurde sie

schon ein paar Jahre nicht mehr aktualisiert. OpenWBEM kommt in einer Reihe von kommerziellen Software-Systemen zum Einsatz. Darunter z.B. Novells SuseLinux Enterprise Edition. Interessant ist, dass Chris Hobbs in seinem Buch [Hobbs2004], für die Realisierung seiner Beispiele OpenPegasus verwendet, obwohl er laut Homepage Mitentwickler von OpenWBEM ist.

Das dritte Projekt ist das „WBEM Services“. Es ist eine Entwicklung von SUN (Oracle) und eine Java Implementierung. Leider ist „WBEM Services“ noch älter als OpenWBEM, es wurde zuletzt 2004 aktualisiert. SUN setzt dieses Projekt für seine Solaris-Systeme ein. Der große Vorteil dieser Implementation liegt in der Programmiersprache Java. Anwendungen, die auf Basis dieses Projektes erstellt werden, sind in der Regel ohne jede Anpassung auf allen Systemen, die eine Java-VM ausführen, lauffähig. Das gilt sowohl für den Server (CIMOM), die Clients und (mit wenigen Ausnahmen) auch für die Provider. Die Provider sind in erster Linie normale Java-Anwendungen und damit prinzipiell auf allen VMs lauffähig. Eine Ausnahme besteht in der Ausführung von Hardware-Zugriffen. Diese sind nur auf den Plattformen lauffähig, auf denen es einen Treiber für das „Java-Nativ-Interface“ (JNI) gibt.

Das letzte Projekt in der Tabelle ist das „Windows Management Instrumentation“ (WMI). Es ist eine Implementierung von Mircosoft. Sie dienst Mircosoft zur Verwaltung und Fernwartung ihrer Windows Betriebssysteme. Dieses System ist eine sehr spezielle Implementierung des WBEM, die ausschließlich auf Windows zugeschnitten ist. Eine Verwendung mit anderen Betriebssystemen ist nicht möglich. Lediglich Abfragen können auch durch nicht Windows-Systeme ausgeführt werden. Erweiterungen lassen sich mittels VisualBasic-Skript einpflegen. Für weiterführende Informationen siehe [Microsoft1999], [Microsoft2004] und [Süß1998].

Im Anschluss geht es um die Auswahl des am besten geeigneten Projekts.

Die Maßstäbe für die Projektauswahl sind:

- Portabilität
- Dokumentation
- Aktualität
- Zeitspanne bis zur Einsatzbereitschaft

Im Punkt „Portabilität“ kann schon das erste Projekt „WMI“ aussortiert werden. Da WMI ausschließlich auf Windows-System läuft, ist es überhaupt nicht portabel und wird deshalb hier nicht weiter betrachtet.

Bei OpenPegasus handelt es sich um ein ISO C++ konformes Projekt. Es lässt sich mit Hilfe eines plattformspezifischen Compiler auf jede Plattform portieren. Damit ist OpenPegasus weiter im Rennen. Gleiches gilt für OpenWBEM.

Die WBEM Services sind in Java geschrieben. Der erstellte Bytecode lässt sich, ohne weitere Anpassung, auf jeder Java-VM ausführen. Portabler geht es im Zusammenhang mit diesem Projekten nicht. Damit sind die WBEM Services in dieser Kategorie Sieger.

Das zweite Ausschlusskriterium ist die Projekt-Dokumentation. Diese ist bei allen drei

verbliebenen Projekten gut. OpenPegasus stellt das „*OpenPegasus Administrator's Guide“* [OG2005] zur Verfügung. OpenWBEM hält den *„OpenWBEM Getting Started Guide“* [OWBEM2004] zum download bereit. SUN stellt für die WBEM Services den *„Solaris WBEM Developer’s Guide“* online.

Der Punkt für „Aktualität“ geht, wie bei der Vorstellung der Projekte bereits zu erkennen war, an OpenPegasus. Die anderen Kandidaten hängen mit ihren Releases doch sehr lange zurück.

Kommt noch der letzte Punkt „Zeitspanne bis zur Einsatzbereitschaft“. OpenPegasus und OpenWBEM stehen als Quellcode zum download bereit. Einen geeigneten Compiler vorausgesetzt, lassen sich die Projekte innerhalb einer halben bis dreiviertel Stunde kompilieren. Die WBEM Services stehen als Zip-Archiv zur Verfügung darin enthalten sind der Quellcode, der Bytecode und einige Skripte zur Steuerung des Systems. WBEM Services sind damit, eine Java-VM vorausgesetzt, sofort einsatzbereit.

Damit ist die Entscheidung knapp für die WBEM Services gefallen.

Das Kapitel 6.1 gibt eine Einführung in die WBEM Services.

6.1 WBEM Services

Dieses Kapitel soll eine Einführung in die WBEM Services geben. Die WBEM Services sind eine im Ursprung von „SUN Mircosystems“ stammende, in Java geschriebene, WBEM Implementierung. Das Projekt findet auch, nach der Übernahme durch Oracle (2010), noch im Solaris-Systemen Verwendung. Das Projekt steht als ZIP-Archiv auf der Projekt-Homepage (http://wbemservices.sourceforge.net/) zum download bereit. Dabei stehen zwei unterschiedliche Archive zur Auswahl (wbemservices-1.0.2.src.zip / wbemservices-1.0.2.bin.zip). Es enthält das Archiv mit „src“ den Source-Code, während das Archiv mit „bin“ den Bytecode enthält. Das Projekt besteht aus der API, dem Server (CIMOM), einer Reihe von Beispiel-Clients und Providern, der Anwendung „CIMWorkshop“ und einer Installationsanleitung im HTML-Format.

Nach dem Entpacken des BIN-Archives findet man einen Ordner „wbemservices-bin“ vor. Darin enthalten sind die folgenden Unterordner:

Verzeichnis:	***Inhalt:***
bin	Enthält eine Reihe von Skripten, darunter die Skripte für dem MOF-Compiler und den CIMWorkshop. Auch die Anwendungen „cimworkshop.jar“ und „mofcomp.jar“ selbst sind darin enthalten.
cimom	Dieser Ordner enthält einen Unterordner „bin“, der die Skripte zum Starten und Stoppen des CIMOM beinhaltet. Im Unterordner „lib“ befinden sich die „.jar-Archive“ „cimom.jar“, „cimrepository.jar“ und „wbemstartup.jar“.
doc	Der Ordner doc enthält Anweisungen zur Installation und dem Betrieb der WBEM Services. Es werden Anleitungen für Unix- und Windows-Systeme gegeben. Die Einstiegsseite heißt „Instructions.html“.
examples	Enthält Beispiele für Clients und Provider sowohl im Quellcode als auch im Bytecode.
lib	Dieses Ordner enthält das jar-Archiv mir der WBEM-API.
mof	In diesem Ordner sind die MOF-Dateien des CIM Schema in verschiedenen Versionen enthalten.

Die in den Ordner vorhandenen Skripte (bat-Dateien für Windows / sh-Dateien für Unix) setzen die erforderlichen System-Variablen und starten die zugehörigen Anwendungen.

Weitere Informationen zu den WBEM Services werden in den folgenden Unterkapiteln und dem „Soraris WBEM Developer's Guide“ [SUN2005] gegeben. Zu beachten ist, dass der „Developer's Guide“ speziell auf die Verwendung mit dem Oracle Solaris-System zugeschnitten ist. Die im „Guide“ genannten Pfade und Verzeichnisse stimmen nicht mit denen im Download überein. Hier ist im Zweifelsfall die Installationsanleitung im „doc“-Verzeichnis zu rate zu ziehen. Die Anleitungen „Writing a Client Program“ und „Writing a Provider Program“ können ohne Änderungen übernommen werden.

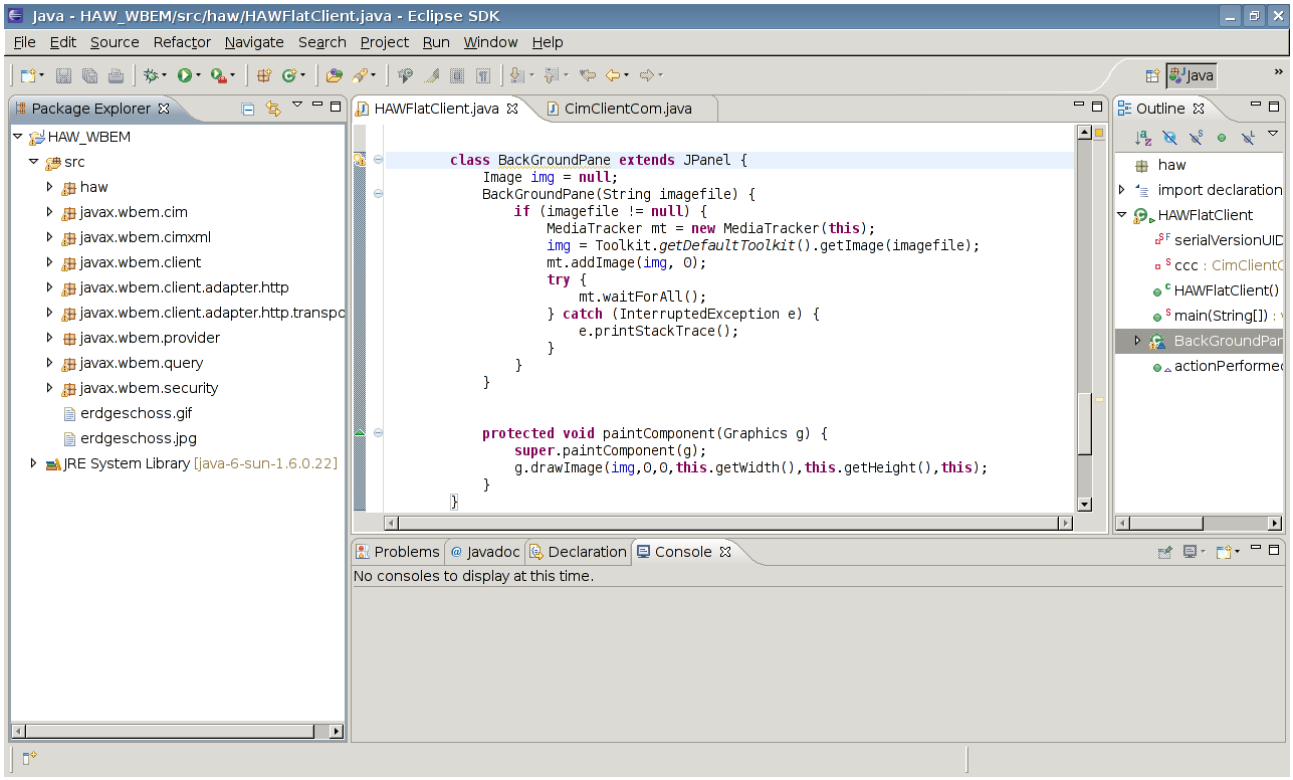


Abbildung 23: Screenshot: Eclipse: Projektfenster mit WBEM-API

Die Programmierung für dieses Projekt erfolgt auf einem PC mit Debian 5.0.8, unter Eclipse 3.6.1 (Helios). Um die API der WBEM Services verwenden zu können, müssen die Packages (Ordner „/open/java/wbem/javax“) in das Eclipse-Java-Projekt kopiert werden (siehe Abbildung 23 - Package Explorer). Die Packages sind Bestandteil des ZIP-Archives „wbemservices-1.0.2.src.zip“. Weitere Informationen zu den Packages werden zu gegebener Zeit in den jeweiligen Unterkapiteln (ab Kapitel 6.5, Seite 67) gegeben.

Kapitel 6.2 gibt eine Einführung in die Verwendung des Servers (CIMOM).

6.2 Server

Der Server (CIMOM) ist in den WBEM Services bereits fertig kompiliert und liegt als Bytecode vor. Ihm beigelegt sind eine Reihe von Skripten, die dem Anwender das Starten und Stoppen des CIMOM erleichtern sollen. Im nachfolgenden wird erläutert, wie der CIMOM der WBEM Services gestartet wird und wie er beeinflusst / gesteuert werden kann.

Um den CIMOM starten zu können, muss die Umgebungsvariable „JAVA_HOME“ gesetzt werden, die den Pfad der Java-VM angibt. Das geschieht auf der Konsole mit:

```
superuser@debian> export JAVA_HOME=/usr
```

Die Skripte für den CIMOM liegen im Verzeichnis:

/wbemservices-bin/cimom/bin/

Bevor der CIMOM gestartet werden kann ist es erforderlich sich als Superuser (root) am Rechner anzumelden. Als Superuser lässt sich der CIMOM wie folgt starten und stoppen:

CIMOM starten:

```
superuser@debian> /wbemservices/cimom/bin/start_cimom.sh
```

Der CIMOM wird gestartet und eine Konsolen-Ausgabe gibt Auskunft über die, dem CIMOM zugewiesene, Prozess-ID.

Ab jetzt „lauscht“ der CIMOM auf die folgenden Ports:

- 5987 – RMI
- 5988 – CIM-XML über HTTP
- 5989 – CIM-XML über HTTPS

Zu beachten ist, dass diese Ports nicht durch eine Firewall blockiert werden dürfen.

CIMOM stoppen:

```
superuser@debian> /wbemservices/cimom/bin/stop_cimom.sh
```

Beim Ausführen dieses Skriptes wird lediglich die Prozess-ID des CIMOM bekannt gegeben. Eine Beendigung des CIMOM findet dadurch nicht statt! Um die CIMOM endgültig zu beenden ist der Befehl:

```
superuser@debian> kill <prozess-id>
```

einzugeben.

Kapitel 6.3 stellt ein sehr hilfreiches Tool, den CIMWorkshop, vor.

6.3 CIMWorkshop

Der CIMWorkshop ist in erster Linie ein WBEM-Client. Nach dem Start des CIMOM besteht die Möglichkeit, die den WBEM Services mitgelieferte Anwendung „CIMWorkshop“ zu nutzen, um sich die Klassen, Instanzen und Events des CIMOM anzeigen zu lassen.

Das Skript für den Cimworkshop liegen im Verzeichnis:

/wbemservices/bin/

Um den CIMWorkshop zu starten, ist die folgende Eingabe auf der Konsole erforderlich:

```
user@debian> /webservices/bin/cimworkshop.sh
```

Das starten des CIMWorkshop kann mit normalen User-Rechten erfolgen. Damit unter Debian die GUI erscheint, muss der gleiche Benutzer den CIMWorkschop starten, der auch an der KDE/GNOME angemeldet ist. Anderenfalls versagt der X-Server das Anzeigen der GUI. Die Anwendung kann auch auf einem anderen Rechner, als dem der den CIMOM ausführt, gestartet werden.

Es erscheint die Anmeldemaske aus Abbildung 24.

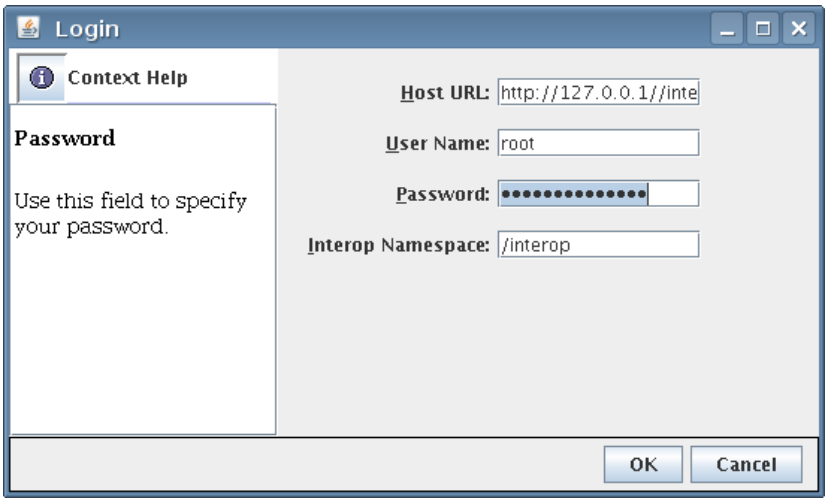


Abbildung 24: Screeshot: cimworkshop - Login

Der CIMWorkshop ist ein WBEM-Client, der mit den Privilegien des angegebenen Benutzers, zugriff auf den WBEM-Server (CIMOM) erhält. Im Beispiel wurde der CIMWorkshop auf dem selben Rechner wie der CIMOM gestartet. Als „Host URL“ wurde die IP-Adresse des Loopback-Interfaces (127.0.0.1) angegeben. Die Angabe der IPv4 Adresse des Loopback-Interfaces ist dem Namen „localhost“ vorzuziehen. Unter Debian 5.0.8 wurde „localhost“ automatisch gegen die IPv6-Adresse des Rechners ersetzt. Aus unerklärlichem Grund kam es dadurch zu Performance-Problemen, die eine Verwendung des CIMWorkshop nahezu unmöglich machte.

Der User „root“ ist der Superuser dieses Rechners und hat damit alle Rechte zum erstellen, löschen oder ändern der Klassen und Instanzen. Die Zeile in der Anmeldemaske

„Interop Namespace“ ermöglicht es, bereits bei Verbindungsaufbau einen Namespace anzugeben. Es kann getrost der Standard-Wert beibehalten werden.

Es gibt dabei zwei Möglichkeiten der Verbindungs-Aufnahme. Die eine ist Java-RMI, die andere CIM-XML über HTTP(s). Der CIMWorkshop verwendet, in diesem Beispiel, zur Kommunikation mit dem CIMOM das CIM-XML über HTTP. Der CIMWorkshop ist ein CIM-Client der, durch die Unterstützung von CIM-XML, auch mit anderen Server-Implementierungen verwendet werden kann.

Nachdem die Verbindung zum Server aufgebaut wurde, erscheint das Fenster in Abbildung 25. Der Linke „Verzeichnisbaum“ stellt die Klassen-Hierarchie des CIM Schema dar. Nach der Auswahl einer Klasse, lassen sich auf der rechten Seite die Eigenschaften und Methoden der Klasse anzeigen. Unter dem Reiter „Events“ ist es möglich, sich für den Empfang von Events anzumelden. Diese erscheinen im unteren Feld „Event Output“.

Über das Menü „Action“ lassen sich neue Klassen erstellen oder bestehende Klassen löschen. Beim Löschen einer Klasse ist Vorsicht geboten, da ein versehentliches Löschen einer Klasse das System unbrauchbar machen kann.

Beim erstellen einer neuen Klasse kann die Superklasse, die Eigenschaften, Methoden und Qualifiers gesetzt werden. Im Anschluss wird die neue Klasse ebenfalls links in der Klassen-Hierarchie angezeigt. Dieser Weg ist, aufgrund der unzähligen Mausklicks, nicht sehr Komfortabel.

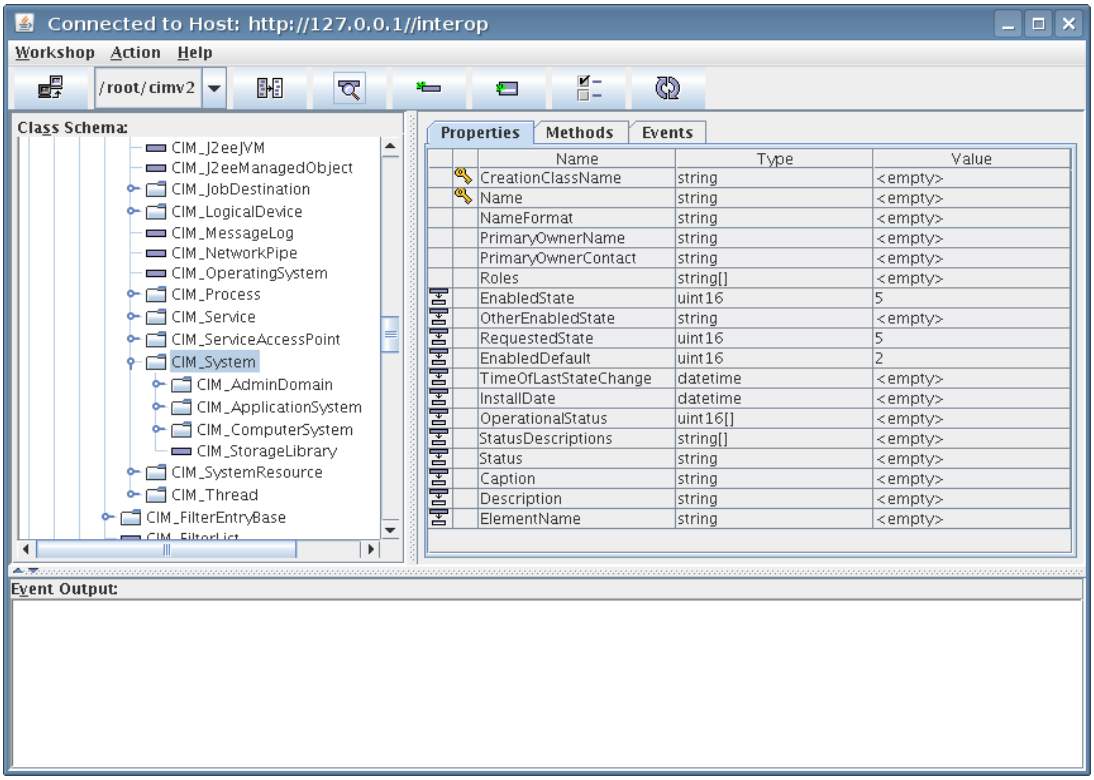


Abbildung 25: Screeshot: cimworkshop - GUI

Kapitel 6.4 stellt die Verwendung des MOF-Compiler vor.

6.4 MOF-Compiler

Der zweite Weg um neue Klassen in das Repository des CIMOM zu bekommen, ist das Kompilieren von MOF-Dateien. Die MOF-Dateien wurden in Kapitel 3.4 vorgestellt und fanden in Kapitel 4 Verwendung.

Die in Kapitel 4 erstellten MOF-Files lassen sich mit Hilfe des folgenden Skriptes kompilieren und in das Repository einspielen.

MOF-Compilieren:

```
user@debian> /webservices/bin/mofcomp.sh <user> <passwd> <mof_file>
```

Um prüfen zu können, ob das Kompilieren erfolgreich gewesen ist, sollte auf die Ausgaben des Compilers auf der Konsole geachtet werden. Alternativ kann dies, mit Hilfe des CIMWorkshop, überprüft werden (siehe Abbildung 25).

Der CIMOM der WBEM Services übersetzt die CIM-Klassen, sowie die per MOF-Compiler oder CIMWorkshop hinzugefügten Klassen, in Java-Klassen. Das heißt die interne Repräsentation des CIMOM besteht ausschließlich aus Java-Klassen.

Für weitere Informationen zum Server der WBEM Services siehe [SUN2005 S.27ff].

Kapitel 6.5 zeigt die Funktionsweise eines Providers.

6.5 Provider

Was Provider sind wurde bereits in Kapitel 2.2 behandelt. Hier geht es um die konkrete Implementierung eines Beispiel-Provider. Provider werden mit Hilfe der „WBEM- Services for JAVA“ realisiert.

Allgemein:

Jeder Provider muss das Interface „javax.wbem.provider.CIMProvider“ implementieren, welches die beiden Methoden:

- initialize(CIMOMHandle cimom);
- cleanup();

fordert.

Die Methode „initialize“ wird bei jedem Start des CIMOM aufgerufen. Der Provider bekommt dabei das Handle des CIMOM übergeben.

Die Methode „cleanup“ dient in der JAVA-Implementierung nur als Platzhalter und hat derzeit keine Funktionalität. [SUN2005 – S.101]

Zu beachten ist, dass der Provider auf dem selben Rechner wie der CIMOM ausgeführt werden muss! [SUN2005 S.98] Dies gilt speziell für diese WBEM-Implementierung. In anderen WBEM-Projekten (siehe Kapitel 6.) ist es durchaus möglich, Provider und CIMOM über ein Netzwerk kommunizieren zu lassen. Diese Einschränkung stellt, im Rahmen dieses Projektes, aber kein Problem dar.

Sollte dieses System in komplexeren Umgebungen zum Einsatz kommen, könnte dies aber sehr wohl zu einem Problem werden. Zum Beispiel, wenn es erforderlich wäre, die Last der Provider auf mehrere Rechner zu verteilen. Dies könnte nur durch den Einsatz mehrerer WBEM-Server in einer Hierarchie realisiert werden. Dabei würde jeder Server nur eine geringe Anzahl an Providern beherbergen. Es könnte einen obersten WBEM-Server geben, der sich in der Rolle eines Clients, mit ihm untergeordneten Servern verbindet. Dies könnte wiederum einen unnötigen Kommunikations-Overhead zwischen den Servern bedeuten. Möglicherweise würde das Gesamtsystem dadurch sehr träge.

Ein weiteres Problem stellen Zugriffe auf die System-Hardware dar. Um Zugriffe auf die Hardware zu erhalten, gibt es in Java nur wenig Möglichkeiten. Es ist z.B. möglich per Java-Klassen mit Peripherien an einer Seriellen- bzw. Parallelen-Schnittstelle zu kommunizieren. Diese Möglichkeit steht z.B. bei einer USB-Schnittstelle nicht ohne weiteres zur Verfügung. Das stellt ein Problem dar, denn die Parallele-/Serielle-Schnittstelle ist im PC und Serverbereich am aussterben. Hier gibt es nur noch die Möglichkeit, über das „Java Native Interface“ (JNI), C- oder Assemblercode einzubinden. Dieser Code muss für jede verwendete Plattform erneut geschrieben werden. Das geht auf Kosten der Portabilität der JAVA-Anwendung.

Für den Fall das Hardware-Zugriffe ausschließlich auf einer bestimmten Plattform erforderlich sind, ist es vielleicht die bessere Entscheidung den Provider gleich vollständig in C/C++ zu implementieren. In wieweit die Einbindung eines OpenPegasus-Providers in die WBEM Service möglich ist, konnte im Rahmen dieser Arbeit nicht geklärt werden. Dazu wäre es erforderlich, dass die WBEM Services das CMPI-Interface implementieren. Ob dieses Interface Bestandteil der WBEM Services ist, konnte im Rahmen der

Recherchen nicht in Erfahrung gebracht werden. Sollte CMPI nicht Bestandteil der WBEM Services sein, so könnte dies im Extremfall zu einem Wechsel des verwendeten WBEM-Projektes (z.B. zu OpenPegasus) führen. Dies würde eine, von Grund auf neue Einarbeitung, mit erheblicher Einarbeitungszeit in das neue Projekt bedeuten. Die Modelle und MOF-Dateien aus Kapitel 4 wären, aufgrund deren genormten Aufbau, davon aber nicht betroffen und könnten weiterverwendet werden.

Auch die parallele Verwendung zweier Projekt wäre denkbar. Dabei würde es zwei WBEM-Server geben, z.B. OpenPegasus und WBEM Services, die miteinander über das Client-Interface (CIM-XML über HTTP) kommunizieren. Dabei würde OpenPegasus die Provider für die Hardware-Zugriffe verwalten, während sich die WBEM Services z.B. um die Software-Provider kümmern würde.

Listing 10 zeigt einen Instanz-Provider, der Bestandteil des WBEM Services ist. Dieser Provider implementiert nur eine minimale Unterstützung, der Klasse Ex_SimpleInstanceProvider (siehe Listing 11). Dabei wird unter anderem gezeigt, wie die Enumeration von Instanzen geschehen kann. Das Erzeugen und Zerstören von Instanzen wird an dieser Stell nicht gezeigt, sonder ist nur als Methodenrumpf implementiert. Beim Aufruf einer konstruktiven oder destruktiven Methode wird eine „CIM_ERR_NOT_SUPPORTED“-Exception geworfen. Diese Exception kann, auch in eigenen Implementierungen verwendet werden, wenn durch eine Superklasse eine Methode gefordert wird, die das eigene Objekt nicht unterstützt.

```
import javax.wbem.cim.*;
import javax.wbem.client.*;
import javax.wbem.provider.*;

import java.util.*;


public class SimpleInstanceProvider implements CIMInstanceProvider {

    private static int loop = 0;

    public void initialize(CIMOMHandle cimom) throws CIMException {}

    public void cleanup() throws CIMException {}

    public CIMInstance[] enumerateInstances(CIMObjectPath op, boolean localOnly,boolean includeQualifiers,
                                            boolean includeClassOrigin, String[] propList, CIMClass cc)
                                            throws CIMException {
       return null;}

    /*
     * enumInstances:
     * The instance names are returned.
     * Deep or shallow enumeration is possible, however
     * currently the CIMOM only asks for shallow enumeration.
     */
    public CIMObjectPath[] enumerateInstanceNames(CIMObjectPath op, CIMClass cc)
    throws CIMException {
        if (op.getObjectName().equalsIgnoreCase("Ex_SimpleInstanceProvider")) {

            Vector instances = new Vector();
            CIMObjectPath cop = new CIMObjectPath(op.getObjectName(),
                                                  op.getNameSpace());
            if (loop == 0) {
                cop.addKey("First", new CIMValue("red"));
                cop.addKey("Last", new CIMValue("apple"));
                instances.addElement(cop);
                loop += 1;
            } else {
                cop.addKey("First", new CIMValue("red"));
                cop.addKey("Last", new CIMValue("apple"));
                instances.addElement(cop);

                cop = new CIMObjectPath(op.getObjectName(),
                                                  op.getNameSpace());
                cop.addKey("First", new CIMValue("green"));
                cop.addKey("Last", new CIMValue("apple"));
                instances.addElement(cop);
            }

            CIMObjectPath[] copArray = new CIMObjectPath[instances.size()];
            instances.toArray(copArray);
            return copArray;
        }
        return null;
    }

    public CIMInstance getInstance(CIMObjectPath op,
                                   boolean localOnly,
                                   boolean includeQualifiers,
                                   boolean includeClassOrigin,
                                   String[] propertyList,
                                   CIMClass cc)
    throws CIMException {
        if (op.getObjectName().equalsIgnoreCase("Ex_SimpleInstanceProvider")) {
            CIMInstance ci = cc.newInstance();
            ci.setProperty("First", new CIMValue("yellow"));
            ci.setProperty("Last", new CIMValue("apple"));
            if (localOnly) {
                ci = ci.localElements();
            }
            return ci.filterProperties(propertyList,
                                       includeQualifiers,
                                       includeClassOrigin);
        }
        return new CIMInstance();
    }

    public CIMInstance[] execQuery(CIMObjectPath op, String query,
                                   String ql, CIMClass cc)
    throws CIMException {
```

```
        throw(new CIMException(CIMException.CIM_ERR_NOT_SUPPORTED));
    }

    public void setInstance(CIMObjectPath op, CIMInstance ci,
        boolean includeQualifiers, String[] propList)
                throws CIMException {
        throw(new CIMException(CIMException.CIM_ERR_NOT_SUPPORTED));
    }

    public CIMObjectPath createInstance(CIMObjectPath op, CIMInstance ci)
                throws CIMException {
        throw(new CIMException(CIMException.CIM_ERR_NOT_SUPPORTED));
    }

    public void deleteInstance(CIMObjectPath cp)
                throws CIMException {
        throw(new CIMException(CIMException.CIM_ERR_NOT_SUPPORTED));
    }
}
```

Listing 10: JAVA: SimpleInstanceProvider.java [SUN]

```
/*
Title            SimpleInstanceProvider
Description      SimpleInstance Provider class
Date             12/05/1999
Version          1.0
(c) Copyright 2001, Sun Microsystems, Inc.
*/


#pragma Locale ("en-US")


    [Provider("java:SimpleInstanceProvider")
    ]

class Ex_SimpleInstanceProvider
{
    // Properties
    [Key, Description("First Name of the User")
    ]
    string First;

    [Description("Last Name of the User")
    ]
    string Last;

};
```

Listing 11: MOF: Ex_SimpleInstanceProvider [SUN]

Für weitere Informationen zu WBEM-Providern siehe [SUN2005 – S.97ff] und [Hobbs2004 S. 201-245]

Kapitel 6.6 zeigt die Verwendung von Clients.

6.6 Client

Was ein Client ist wurde in Kapitel behandelt. Hier geht es um die Implementierung eines minimalen WBEM-Clients, der mit Hilfe der „WBEM Services for Java“ realisiert wird.

Die Implementierung des Energiespardienstes und des Informationsdienstes erfolgt hier, aus Zeitmangel, nicht. Dennoch wird an dieser Stelle über die beiden „Dienste“ diskutiert. Bei einer Projekt-Fortführung könnte der Informationsdienst als WBEM-Listener implementiert werden. Dieser könnte sich z.B. für den Empfang von „Standby-Indications“ registrieren. Diese Indications würden z.B. durch einen Messprovider ausgelöst, der den Wechsel eines Gerätes in den Leerlauf-Modus als Event erkennt. Diese Indications könnten, auf dem Benutzerinterface des Listeners, zu den in der Motivation beschriebenen hinweisen führen.

Auch eine Veränderung der Leistungsaufnahme könnte als Event behandelt werden. Dies könnte genutzt werden, um die ebenfalls in der Motivation erdachte, Anzeige des Verbrauches in Euro pro Stunde (€/h), zu realisieren. Dies könnte z.B. so erfolgen:

- Der Messprovider misst den Strom (I) in Ampere (A) und die Spannung (U) in Volt (V). Daraus lässt sich mit $P=U*I$ die Leistung (P) in Watt (W) ermitteln.
- Um den Energieverbrauch in €/h ausweisen zu können, muss der Preis pro Kilowattstunde (ct/kWh) bekannt sein.
- Die ermittelte Leistung in Watt (W) wird in Kilowatt (kW) umgerechnet. Eine Kilowatt (1 kW) entsprechen 1000 Watt (1000 W). $P=x(\frac{W}{1000})=x\,kW$
- Der Verbrauch an Geldmitteln (Q), in Euro pro Stunde (€/h), lässt sich aus $Q=(x\,kW*y(\frac{ct}{kWh}))/100=z\,€/h$ errechnen. Dabei ist x die gemessene Leistung in Kilowatt und y der Arbeits-Preis in Cent pro Kilowattstunde. Das Ergebnis z ist der errechnete Verzehr von Geldmittel pro Stunde, der dem Anwender angezeigt wird.
- Beispiel: Ein Durchlauferhitzer im Bad nimmt eine Leistung von 5 kW auf. Der Preis pro Kilowattstunde beträgt 30 ct/kWh. $Q=(5\,kW*30(\frac{ct}{kWh}))/100=1{,}50\,€/h$ Daraus folgt: Sollte eine Person eine Stunde duschen, so entstehen die angezeigten Kosten (1,50€). Dies könnte der Ansporn sein, die Zeit fürs Duschen zu verkürzen. Positiver Nebeneffekt: Es wird zusätzlich Wasser gespart.

Die Implementation des Energiespardienstes könnte auf folgendem Weg erfolgen. Die Abbildung der Systemumgebung in WBEM/CIM hat zur Folge, dass diese Plattform als Grundlage für jede denkbare Art von Dienstleistung genutzt werden kann. So kann der Energiespardienst z.B. als Client realisiert werden, der das WBEM-System selbstständig nach „Energieverschwendern“ durchsucht (Enumeration der Instanzen) und diese nach Möglichkeit abschaltet, in einen Energiesparmodus versetzt oder zumindest den Nutzer auf Optimierungsmöglichkeiten hinweist. Dieser Client kann, die nötigen Rechte vorausgesetzt, über sämtliche Informationen des Systems verfügen und diese beeinflussen.

Damit ist es möglich diesen Client mit „Künstlicher Intelligenz“ (KI) auszustatten und diesen mit der bedarfsgerechten Steuerung von Diensten und Geräten zu betrauen. Der

Vorteil dieser Lösung liegt darin, dass dieser Client mit jedem WBEM-System verwendet werden kann. Und das unabhängig von der konkreten Implementierung des WBEM-Systems, denn der Client kommuniziert über das Protokoll CIM-XML über HTTP(s). So wäre es möglich den Energiesparclient, als eigenständige Software, zu veräußern. Nachteilig ist es, dass diesem Client weitreichende Rechte verliehen werden müssten. Dies kann für das WBEM-System ein Sicherheitsrisiko darstellen. Auch sind negative Wechselwirkungen, zwischen verschiedenen Clients, nicht auszuschließen.

Nachfolgend wird gezeigt, wie ein minimaler Client die Verbindung zu einem WBEM-Server herstellt. Ursprüngliche Idee war es, einen Client mit einer GUI auszustatten (siehe Abbildung 26), die die Zustände der einzelnen Räume im Grundriss der Wohnung visualisiert. Dabei sollte folgendes, in jedem Raum, angezeigt werden:

- Anwesenheit von Personen
- Temperatur
- Luftfeuchtigkeit
- Helligkeit
- Zustand Heizkörper
- Zustand Belüftung

Buttons am unteren Bildrand sollten den Wechsel zwischen Szenarien ermöglichen. Diese Dienste ließen sich, aufgrund der fehlenden Provider, bislang nicht realisieren.

Listing 12 zeigt die Java-Klasse CimClientCom. Sie wird für den Verbindungs- Aufbau und Abbau verwendet. Die Methode ConnectCIMOM erzeugt:

- NameSpace-Objekt
- UserPricipal-Objekt (Benutzerkennung)
- PasswordCredential (Passwort)

Stellt die Verbindung zum CIMOM her, in dem ein CIMClient-Objekt erzeugt wird.

```
package haw;

import javax.wbem.cim.CIMException;
import javax.wbem.cim.CIMNameSpace;
import javax.wbem.client.CIMClient;
import javax.wbem.client.PasswordCredential;
import javax.wbem.client.UserPrincipal;

public class CimClientCom {

	private CIMNameSpace nameSpace;
	private CIMClient client;

//-----------------------------------------------------------------------------------------------------------------
	public void connectCIMOM(String host, String namespace, String user, String passwd){
		nameSpace = new CIMNameSpace(host, namespace); // default NameSpace (root\cimv2) auf Localhost
		UserPrincipal up = new UserPrincipal(user); // Benutzer, der die Verbindung aufbaut
		PasswordCredential pc = new PasswordCredential(passwd); // Passwort des Benutzers

		System.out.println("Verbindung zu \"" + nameSpace.getHost() + "\" mit Benutzer \""  +
							up.getUserName() + "\" wird aufgebaut.");
		try {
			client = new CIMClient(nameSpace, up, pc); // Verbindung zum CIMOM aufbauen
		} catch (CIMException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}

//-----------------------------------------------------------------------------------------------------------------
	public void disconnectCIMOM(){

		if(client != null){
			System.out.println("Verbindung zu \"" + nameSpace.getHost() + "\" wird getrennt.");
			try {
				client.close(); // Verbindung zum CIMOM trennen
			} catch (CIMException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		} else System.out.println("Es besteht keine Verbindung!");
	}

//-----------------------------------------------------------------------------------------------------------------

	public boolean isConnected(){
		if(client != null){
			return true;
		} else return false;
	}



}
```

Listing 12: JAVA: CimClientCom.java

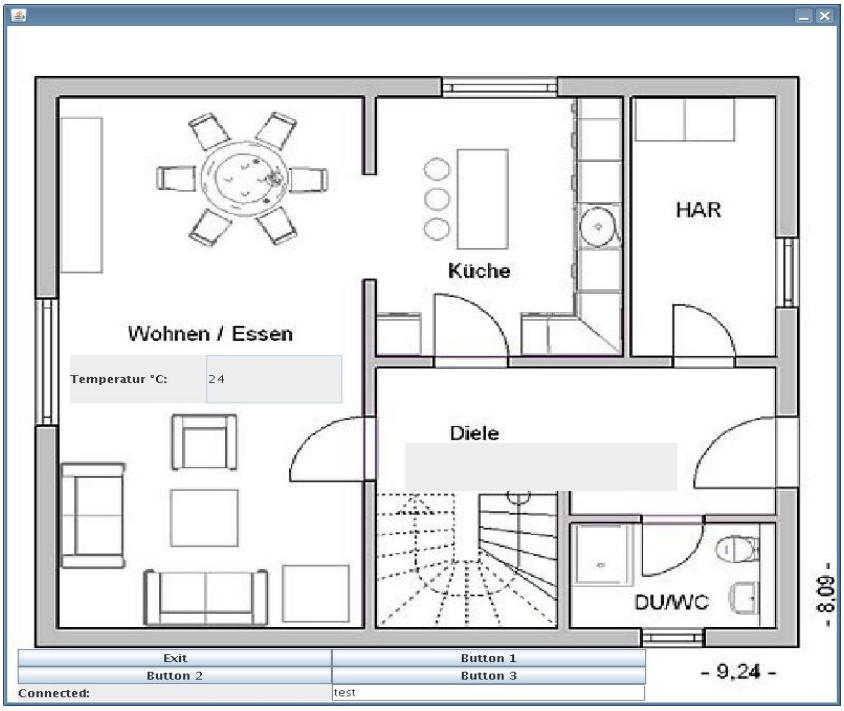


Abbildung 26: Screenshot: Client-GUI

Der Code zu Abbildung 26 befindet sich im „workspace“ auf der DVD im Anhang.

Für weitere Informationen zu WBEM Services-Clients siehe [SUN2005 – S.45ff].

7. Schlussbetrachtung

Dieses Kapitel gibt einen kritischen Rückblick auf das durchgeführte Projekt. Dabei wird ein Fazit gezogen und ein Ausblick gegeben.

7.1 Zusammenfassung / Fazit

Es wurde die Anwendbarkeit des „Web-Based Enterprise Management“, auf das erweiterte Gebäudemanagement untersucht. Dabei stand die Suche nach Anknüpfungspunkten, zwischen dem erweiterten Gebäudemanagement und dem WBEM/CIM, im Vordergrund. Festgestellt wurde, dass es prinzipiell möglich ist, die Komponenten eines Gebäudes in CIM abzubilden. Auch die im Gebäude befindlichen Komponenten lassen sich darstellen.

WBEM/CIM ist primär für den Einsatz in großen IT-/TK-Systemen ausgelegt. Für nahezu alle Standard-Komponenten, eines Rechenzentrums, stehen entsprechende Modelle zu Verfügung. Dies sieht für das Gebäudemanagement leider ganz anders aus. Es existieren hier keinerlei Modelle zur Anbindung von Installationsbussen, Multimedia- oder Haushaltsgeräten. Diese Modelle müssen bei Bedarf vollständig selbst modelliert werden. Das erstellen eigener Modelle, und die damit verbundenen Einbindung neuer Systeme, ist immer möglich. Damit ist das WBEM/CIM ein mächtiges Modell, mit dessen Hilfe die Verwaltung großer Strukturen erheblich vereinfacht werden kann.

Es wurden existierende WBEM-Projekte drauf hin untersucht, ob diese dazu verwendet werden können, um ein zentrales Rechnersystem aufzusetzen. Dabei wurden die WBEM Services näher betrachtet. Die Betrachtung kommt zum Ergebnis, dass dieses WBEM-Projekt als Plattform, für die Verwaltung und Steuerung eines zugrundeliegenden Gesamtsystems, verwendet werden kann. Dabei wird das Gesamtsystem in Teilsysteme zerlegt, die ebenfalls eine weitere Unterteilung erfahren können. Dies erfolgt nach dem Prinzip „teile und herrsche“.

Am Ende der Unterteilung liegen die, durch Provider betreuten, elementaren Systeme. Diese Systeme erledigen die eigentlichen Aufgaben des Gesamtsystems. Daraus wird die Erkenntnis gewonnen, dass der Leistungsumfang des WBEM-Systems erheblich vom Implementierungsumfang der Provider abhängt. Je besser ein Provider das betreute Objekt abbildet, desto Leistungsfähiger ist das Gesamtsystem. Auf Basis von Provider lassen sich alle denkbaren Dienste, wie z.B. Messeinrichtungen oder das Smart Meter des Energieversorgers, in WBEM einbinden.

Ein Energiespardienst lässt sich prinzipiell auf Basis eines WBEM-Client erstellen. Die Möglichkeiten der Energieeinsparung sind auf dieser Grundlage vielfältig, jedoch durch den oben beschriebenen Leistungsumfang der Provider beschränkt.

Auch ein Informationsdienst könnte auf Basis eines WBEM-Listener erstellt werden.

Letztendlich stellt WBEM eine einfache Schnittstelle zur Verfügung, die den System-Anwender in die Lage versetzt, ein komplexes System auf einfache und standardisierte Weise zu verwalten. Das ist das große Plus von WBEM/CIM.

Nachteilig ist die langwierige und komplexe Einarbeitungsphase in den Aufbau von WBEM/CIM. Der Aufwand für die Einarbeitung in WBEM/CIM und der Auswahl einer geeigneten WBEM/CIM Implementierung wurde, im Rahmen dieser Arbeit, unterschätzt.

Der für die Recherchen zu WBEM/CIM eingeplante Zeitabschnitt war zu knapp bemessen. Auch wurde erst spät erkannt, dass sich diese Recherchen beinahe vollständig auf dem kritischen Pfad befanden und sich so bis zum Projektende hinzogen. Das führte zu erheblicher Zeitknappheit zum Projektende, so das die in Kapitel 6.5 (Provider) und 6.6 (Client) geplanten Implementierungen zu kurz gekommen sind. Für künftige Projekte werden effizientere Methoden der Projektplanung / Projektsteuerung und mehr Pufferzeiten zum Einsatz kommen.

Bei einer Projektwiederholung würde die Auswahl eines WBEM-Projektes mit an vorderster Stelle stehen. Dieses Projekt würde verstärkt genutzt, um die erarbeiteten Themen unmittelbar in der Praxis zu testen. Dies hätte zu einem schnelleren Verständnis beigetragen und zudem früher zu verwertbarem Code geführt.

Hinderlich bei der Einarbeitung in das Thema war, das spärliche Angebot an Literatur zu WBEM/CIM. Außerhalb der DMTF-Dokumentation, ist zu diesen Themen nur ein Buch erschienen. Weitere gedruckte Literatur zu WBEM/CIM konnte, im Rahmen der Recherchen, nicht gefunden werden. Die DMTF-Dokumentation wird dafür sehr aktuell gehalten, das lässt darauf schließen, dass WBEM/CIM rege Verwendung findet.

Rückblickend lässt sich sagen, dass WBEM/CIM dafür geeigneten ist, die in der Motivation beschriebenen Dienste und Anwendung abzubilden. Aufgrund seiner Komplexität ist es aber nicht, für die Implementierung eines Einzelsystems, zu empfehlen.

7.2 Ausblick

Ein möglicher Ansatz zu Fortführung, wäre die Überprüfung, in wieweit ein Provider zum Energiesparen beitragen kann. Es könnte untersucht werden, ob ein Provider durch konsequentes Anwenden von Energiesparfunktionen (des betreuten Objekts), zur Ressourcenschonung beitragen kann.

Ein weiterer Ansatz ist die Verwendung von Protokollen zur Ermittlung des Nutzerverhaltens. Diese könnte zur automatischen Generierung von Szenarien verwendet werden. Die erhobenen Daten könnten auch genutzt werden, um z.B. bei Abwesenheit der Bewohner, Tagesabläufe zur Abschreckung von Einbrechern zu wiederholen.

Interessant ist auch die Fragestellung, in wieweit es möglich ist, mit Hilfe der „Künstlichen Intelligenz“ (KI), einen Energiesparclient zu entwickeln. Weiterführend könnte analysiert werden wie groß die Energieeinsparungen, gegenüber einer Implementation ohne KI-Techniken, sind.

Ein anderer Aspekt wäre die Untersuchung, wie sich widersprüchliche Befehle unterschiedlicher Clients, auf das WBEM-System auswirken. Die Wechselwirkungen könnten, am Beispiel eines Energiesparclient und eines Test-Client, behandelt werden.

Ebenfalls interessant ist die Frage, wie und welche Informationen von Haus, Wohnungen, Räumen und Zonen erfasst werden müssen, um sie zur Navigation autonomer Systeme nutzten zu können. Auch muss in diesem Kontext die Frage, nach der Anbindung des autonomen Systems an WBEM, beantwortet werden.

8. Anhänge

A Glossar

Begriff:	***Bedeutung:***
Administrator	Benutzer / Bediener eines Systems mit privilegierten Rechten.
Anwender	Benutzer / Bediener eines Systems mit eingeschränkten Rechten.
CIM	Steht für „Common Information Model“ und ist eine durch die DMTF entwickelter Standard zur Abbildung von Datenstrukturen.
CIMOM	Steht für „CIM Object Manager“. Verwaltet Klassen und Instanzen eines WBEM/CIM-Modell und ist eine der Kernkomponenten eines WBEM-Server.
CMPI	Steht für „Common Manageability Programming Interface“, stellt die Schnittstelle zwischen WBEM-Server und Provider dar.
Client	Rechnersystem, welches die angebotenen Dienste eines anderen (Rechner-) Systems nutzt.
DMTF	Die „Distributed Management Task Force“ ist eine, durch eine Vielzahl namhafte Firmen der IT-Branche gegründete, Organisation mit dem Zweck einheitliche Standards zu schaffen.
EIB	„Europäischer Installationsbus“ ist (wie KNX, LCN, LON) ein System zur Automation von Gebäuden.
erweitertes Gebäudemanagement	Ein Gebäudemanagement, welches um die Verwaltung weiterer, für die Gebäudeinstallation untypischen, Komponenten erweitert wurde.
Facility Management	Bewirtschaftung und Verwaltung von Gebäuden und deren technischen Anlagen
Gebäudeinstallation	Gesamtheit der in einem Gebäude montierten / installierten Komponenten, (Versorgungs-)Leitungen und Bussystemen.
Gebäudemanagement	Oberbegriff für das Verwalten, Überwachen und Steuern eines Gebäudes.
Gesamtsystem	Zusammenfassung mehrere Einzelsysteme zu einem Ganzen.
heterogen	vielschichtig / uneinheitlich
Installationsbus	Ist der Oberbegriff für ein System zur Automation von Gebäuden. (z.B. EIB/KNX, LON, LCN)
Kind	Klasse, die die Eigenschaften einer anderen Klasse geerbt hat.
KNX	Weiterentwicklung des EIB, wird synonym für EIB verwendet.
KWh	Ist das Einheitenzeichen für die verrichtete (elektr.) Arbeit mit der Einheit Kilowattstunden.

LCN	„Local Control Network“ ist (wie EIB/KNX, LON) ein System zur Automation von Gebäuden.
LON	„Local Operating Network“ ist (wie EIB/KNX, LCN) ein System zur Automation von Gebäuden.
OMG	Steht für „Object Management Group“ und ist ein Konsortium namhafter IT-Firmen (IBM, Oracle, Mircosoft,...). Einwickelt offenen Standards (z.B. UML, XML, …).
proprietär	Eigenentwicklung / nicht frei
Repository	Datenspeicher / Datenstruktur eines WBEM-Server.
Server	Rechnersystem das Dienste zur Verfügung stellt.
Smart Metering	intelligentes Messen / intelligentes Messgerät
Szenario	Beschreibung eines möglichen Anwendungsfalles
User	siehe Anwender
Verbraucher	Meint ein System oder Gerät das Ressourcen verzehrt.
Versorger	Dienstleister, der Ressourcen (Wasser, Stom, Gas,...) gegen Bezahlung zur Verfügung stellt / liefert.
W	Einheitenzeichen für (elektr.) Leistung mit der Einheit „Watt“.
WBEM	Steht für Web-Based Enterprise Management. Es handelt sich hierbei um einen durch die DMTF entwickelter Standard.
WMI	Steht für „Windows® Management Instrumentation“ und ist eine von Mircosoft entwickelte WBEM-Implementierung, die speziell auf Windows-System abgestimmt ist.

B Bibliographie

Altgeld2001 Altgeld, Horst: *Gebäudeintelligenz als Zukunftssystem*. Saarbrücken: Hochschule für Technik und Wirtschaft des Saarlandes (HTW) – Institut für ZukunftsEnergieSysteme, 2001 – Forschungsbericht

Beckmann2006 Beckmann, Kai: *Die Instrumentierung einer Kaffeemaschine mit CIM/WBEM*. Stand: 06.02.2006 http://wwwvs.cs.hs-rm.de/lehre/material/extern/vs05ws/projekte/P_08_Dokumentation.pdf – Abruf: 23.01.2011 – Hochschule RheinMain – FB Design Informatik Medien

Böhm2008 Böhm, Oliver: *C++ mit Eclipse programmieren: Programmieren lernen leicht gemacht: Studienausgabe.* Poing: Franzis, 2008 – ISBN: 978-3-7723-6749-6

Bünting2000 Bünting, Karl-Dieter; Bitterlich, Axel; Pospiech, Ulrike: *Schreiben im Studium: mit Erfolg: Ein Leitfaden.* Berlin: Cornelsen Scriptor, 2000 – ISBN: 3-589-21417-1

Dannen2008 Dannenberg, Detlev (Hrsg.): *Fit für die Bachelorarbeit: GW-Informatik WiSe 2008/09. Hamburg: HAW-Hamburg - Department Informatik,* 2008 – Skript

Dawson2003 Dawson,Christian W.: *Computerprojekte: im Klartext.* Paderborn: Pearson Studium, 2003 – ISBN: 3-8273-7067-1

DMFT2000 Distributed Management Task Force (Hrsg.): *Document Number: DSP0111: Common Information Model (CIM) Core Model. Stand: 30.08.2000* *http://www.dmtf.org/sites/default/files/standards/documents/DSP0111.pdf* – Abruf: 09.11.2010

DMFT2003 Distributed Management Task Force (Hrsg.): *Document Number: DSP0144:*CIM Device Model White Paper . *Stand: 19.06.*2003 http://www.dmtf.org/sites/default/files/standards/documents/DSP0144.pdf – Abruf: 28.01.2011

DMFT2009 Distributed Management Task Force (Hrsg.): *Document Number: DSP0004: Common Information Model (CIM) Infrastructure. Stand: 01.05.2009* *http://www.dmtf.org/sites/default/files/standards/documents/DSP0004_2.5.0.pdf* – Abruf: 09.11.2010

DMFT2009-2 Distributed Management Task Force (Hrsg.): *Document Number: DSP0200: CIM Operations over HTTP. Stand: 29.07.2009* *http://www.dmtf.org/sites/default/files/standards/documents/DSP0200_1.3.1.pdf* – Abruf: 28.02.2011

DMFT2009-3 Distributed Management Task Force (Hrsg.): *Document Number: DSP0201: Representation of CIM in XML. Stand: 29.07.2009* http://www.dmtf.org/sites/default/files/standards/documents/DSP0201_2.3.1.pdf – Abruf: 28.02.2011

DMFT2010 Distributed Management Task Force (Hrsg.): *CIM Schema: Version 2.26.0. Stand: 21.06.2010* http://dmtf.org/standards/cim/cim_schema_v2260 – Abruf: 09.11.2010

Forbrig2002 Forbrig, Peter: *Objektorientierte Softwareentwicklung mit UML*. Leipzig: Fachbuchverlag Leipzig, 2002 – ISBN: 3-446-21975-7

Heidiger2003 Heidinger, Emanuel: *Ein CIM-basiertes Modell der Rechnernetzepraktikum-Infrastruktur.* München: TU-München – Institut für Informatik, 2003 - http://www.nm.ifi.lmu.de/pub/Fopras/heid04/PDF-Version/heid04.pdf – *Abruf: 13.02.2011*

Hobbs2004 Hobbs, Chris: *A Practical Approach to WBEM/CIM Management.* New York/London: Auerbach Publications, 2004 – ISBN 0-8493-2306-1

IT-GmbH2008
http://www.it-gmbh.de/de/products/elvis/elvis27.htm . Stand: 28.03.2008 – Abruf: 09.11.2010

Kahlbrandt2001 Kahlbrandt, Bernd: *Software-Engineering mit der Unified Modeling Language. 2. Auflage* Berlin/Heidelberg: Springer-Verlag, 2002 – ISBN: 3-540-41600-5

Kirchner2007 Kirchner, Herbert (Hrsg.): *IT: Technologien: Lösungen: Innovationen.* (S. 199 – 204) Berlin/Heidelberg: Springer-Verlag, 2007 – ISBN:978-3-540-46164-7

Kofler2011 Kofler, Michael: *Linux 2011: Debian, Fedora, openSUSE, Ubuntu: 10., überarbeitete und erweiterte Auflage.* München: Addison-Wesley Verlag, 2011. - ISBN: 978-3-8273-3025-3

Krüger2007 Krüger, Guido: *Handbuch der JAVA-Programmierung: Studentenausgabe.* München: Addison-Wesley, 2007 – ISBN 978-3-8273-2447-4

OG2005 The Open Group (Hrsg.): *OpenPegasus Administrator's Guide.* Reading (UK): The Open Group, 2005.
http://www.openpegasus.org/uploads/40/21822/pegasus-2.10.0.zip – Abruf: 12.10.2010

OWBEM2004 Nuffer, Dan: OpenWBEM Getting Started Guide. Stand: 12.09.2004 http://openwbem.sourceforge.net/OpenWBEM_Getting_Started_Guide.pdf – Abruf: 18.03.2011

Saßmann2010 Saßmannshausen, Ralph; Meyer, Johannes: *KNX: Grundlagenwissen zum KNX Standard.*
http://www.knx.org/fileadmin/template/images/news_and_press_menu/lb2010/flyers/KNX_de_Sonderdruck_Screen_DE.pdf - Abruf: 26.09.2010

Scherg2008 Scherg, Rainer: *EIB/KNX-Anlagen: planen, installieren und visualisieren.* Würzburg: Vogel Industrie Medien, 2008. - ISBN: 978-3-8343-3125-0

Sommer2001 Sommerville, Ian: *Software-Engineering.* München: Addison-Wesley, 2001 – ISBN 3-8273-2001-9

SUN2005 Sun Microsystems, Inc: *Solaris WBEM Developer's Guide: Part No: 817–0366–10. Stand: 01.2005* *http://download.oracle.com/docs/cd/E19253-01/817-0366/817-0366.pdf* – Abruf: 01.02.2011

Süß1998 Süß, Bernd; Kacalek, Martin: *Prototypischer Einsatz von WBEM für das Management von Windows NT.* München: Ludwig- Maximilans- Universität - Institut für Informatik, 1998 - *http://www.mnm-team.org/pub/Fopras/kasu98/PDF-Version/kasu98.pdf* – Abruf: 23.01.2011

UBA2003 Umwelt Bundesamt (Hrsg.): *Energiemanagement in Wohnungsunternehmen: Chancen nutzen – Klimaschutz und Wirtschaftlichkeit verbinden.* Hamburg: Hammonia Verlag, 2003 – ISBN 3-87292-157-6

PRO2010 Pro Massivhaus GmbH: *Einfamilienhaus PRO 75.* *http://www.xn--massive-wohnhuser-2qb.de/einfamilienhaus/pro-linie/pro-massivhaus-pro75/details/* - Abruf: 10.11.2010

RWE2010 RWE AG: *Intelligente Stromzähler kommunizieren mit dem Kunden.* Pressemitteilung 10.02.2010. *http://www.rwe.com/web/cms/de/37110/rwe/presse-news/pressemitteilung/?pmid=4004478* – Abruf: 26.09.2010

Mircosoft1999 Maston, Michael: Managing Windows with WMI. Stand: 01.11.1999 http://msdn.microsoft.com/de-de/library/bb742445.aspx – Abruf: 08.03.2011

Mircosoft2004 Stemp, Greg: *WMI als Hilfe zur Selbsthilfe.* Stand: 29.06.2004 http://msdn.microsoft.com/de-de/library/ms974554.aspx#ID0E1C – Abruf: 08.03.2011

Nieß2010 Nieß, Jens-Uwe: Gebäudeautomation: Installationsbusse (EIB, LON, LCN). *http://www.svb-tga.de/gebaeudeautomation_bus.html* – Abruf: 26.09.2010

Fraunhofer2010 Fraunhofer-Institut für Integrierte Schaltungen IIS: *MotionSENS: Bewegungssensor mit Sturzerkennung. Stand: 01.2010* *http://www.iis.fraunhofer.de/bf/med/sensorik/sturz.jsp*. – Abruf: 09.11.2010

G+H2010
http://www.guh-elektro.de/index.cfm?pid=1398&pk=78554 – Abruf: 26.09.2010

VZBu2008 Verbraucherzentrale Bundesverband: *Richtiges Heizen und Lüften: 4. Auflage.* Stand: September 2008 - http://www.verbraucherzentrale-energieberatung.de/web/fileadmin/user_upload/downloads/VZE_Broschuere_HeizenLueften.pdf – Abruf: 01.03.2011

C Software

Debian 5.0.8 (Lenny) i386

http://cdimage.debian.org/debian-cd/5.0.8/i386/iso-cd/debian-508-i386-netinst.iso

Eclipse – Eclipse SDK 3.6.1

http://www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops/R-3.6.1-201009090800/eclipse-SDK-3.6.1-linux-gtk.tar.gz

Java-JDK6 – Oracle Java SE 6 Update 23

http://www.oracle.com/technetwork/java/javase/downloads/index.html

OpenOffice – OpenOffice.org 3.2

http://download.openoffice.org/

OpenPegasus – The Open Group OpenPegasus 2.10.0

http://www.openpegasus.org/uploads/40/21822/pegasus-2.10.0.zip

Umbrello – Umbrello UML Modeller 1.5.8

http://prdownloads.sourceforge.net/uml/umbrello-1.5.8.tar.bz2?download

VirtualBox – Oracle VirtualBox 4 for Linux

http://download.virtualbox.org/virtualbox/4.0.2/virtualbox-4.0_4.0.2-69518~Debian~lenny_i386.deb

WBEM Services - Java™ Web Based Enterprise Management

http://wbemservices.sourceforge.net/

D VirtualBox

Das Programm VirtualBox ist eine Virtualisierungslösung, die mit Unterstützung von Oracle entwickelt wird. Die offizielle Homepage ist „www.virtualbox.org“, von dort können kompilierte Versionen für Linux/Windows/Mac geladen werden.

Um separate Systeme für die Entwicklungsumgebung, Clients und Server zu haben, ohne mehrere PC's aufstellen zu müssen, bietet sich die Nutzung der Virtualisierung an. Mit Hilfe von Virtuellen Maschinen (VM), lassen sich mehrere, von einander unabhängige, Betriebssysteme auf der selben Hardware ausführen.

Dieses Kapitel gibt eine Hilfestellung zur Installation von VirtualBox.

Öffnen sie eine Konsole und melden sie sich als „root“ an:

```
user@debian> su root
user@debian> Password: root
```

Jetzt wechseln sie in das Verzeichnis /tmp

```
user@debian> cd /tmp
```

und laden sie die VirtualBox- Version für debian herunter:

```
user@debian> wget http://dlc.sun.com.edgesuite.net/virtualbox/3.2.10/virtualbox-3.2_3.2.10\
-66523~Debian~lenny_i386.deb
```

Jetzt können sie die Installation vornehmen:

```
user@debian> dpkg -i virtualbox-3.2_3.2.10-66523~Debian~lenny_i386.deb
```

Im Anschluss an die Installation befindet sich im „Startmenü“ von KDE/GNOME ein Button für VirtualBox.

E Linux (Debian)

Um Software- und Lizenzkosten zu vermeiden, werden in dieser Arbeit vorrangig Softwarepakete aus Freeware/OpenSource Quellen verwendet! Als Betriebssystem bietet sich hier Linux an, es ist in den verschiedensten Distributionen und Versionen verfügbar und die Beschaffung (download) sehr leicht. Es gilt als sehr zuverlässig und die Administration (auch aus der Ferne) ist leicht erlernbar. Ein weiteres Plus ist die breite Unterstützung durch Foren, Wiki's und Howtos.

Linux kann sowohl als Wirt, als auch als Gastsystem für die in Kapitel D genannten VM's dienen.

Wie in der Einführung angesprochen, wird als Betriebssystem Linux zum Einsatz kommen. Hier gibt es eine Vielzahl an Distributionen. Die Frage welche Distribution für dieses Projekt die richtige ist, wird nicht untersucht. Hier wird „Debian“ verwendet.

Debian lässt sich als DVD (4,7GB) mit einer Vielzahl von Paketen oder als kleine CD (180MB) zur Internet- Installation herunterladen. Die offizielle Homepage von Debian ist „www.debian.org“, dort können sie die .iso- Dateien unter dem Punkt „Debian besorgen“ herunterladen.

Die verwendete Version ist „Debian 5.0.8 (Lenny)“, die hier als i386-Netz-Installation-Version (URL: http://hammurabi.acc.umu.se/debian-cd/5.0.6/i386/iso-cd/debian-508-i386-netinst.iso) verwendet wird.

Die geladene Datei muss für die Installation des Host- Systems auf eine CD gebrannt werden. Unter Linux kann dies mit:

```
user@debian> wodim -v dev='<device>' debian-508-i386-netinst.iso
```

geschehen.

<device> steht hier für den zu verwendenden CD-Brenner. Welche Geräte zur Verfügung stehen, können sie durch Eingabe von:

```
user@debian> wodim --devices
```

herausfinden.

Die Debian-Installations CD wird in den neuen Host-PC eingelegt und das System gestartet.

Es wird vorausgesetzt, dass der PC über die Ethernet-Schnittstelle mit einem Netzwerk verbunden ist, welches über einen DHCP-Server und einem Internetzugang verfügt. Des weiteren wird angenommen, dass im BIOS des Host-PC das Booten von CD aktiviert wurde.

Im Bootmenü der CD wird die Option „Install“ mit >Return< bestätigt.

Die Installation wird sie durch einen Assistenten führen, dessen Vorschläge in der Regel durch drücken von >Return< übernommen werden kann. Für das Root-Kennwort verwenden sie, der Einfachheit halber, „root“. (Dies ist bei einem reellen Einsatz des System ein Sicherheitsrisiko!). Für den Benutzernamen und dessen Passwort geben sie „user“ ein. Nach Abschluss der Installation wird das frisch installierte System gebootet und die Anmelde-Maske erscheint.

Hier kann eine Anmeldung mit dem Benutzer „user“ geschehen.

Hilfestellungen zu Debian sind unter http://wiki.debian.org erhältlich.

F OpenPegasus

OpenPegasus ist eine OpenSource Implementierung des CIM-Modells. Dieses Kapitel befasst sich mit der Installation von OpenPegasus. Diese Software kann alternativ für die in Kapitel 6.1 vorgestellten WBEM Services verwendet werden.

Installationsanleitung:

Melden sie sich an dem Linux-Rechner, als root an, auf dem OpenPegasus installiert werden soll:

```
cim-server> Login: root
cim-server> Password: root
```

Wechseln sie in den Ordner /tmp:

```
root@cim-server> cd /tmp
```

und laden den Source-Code von openPegasus herunter:

```
root@cim-server> wget http://www.openpegasus.org/uploads/40/21822/pegasus-2.10.0.tar.gz
```

Entpacken sie das Paket mit:

```
root@cim-server> tar xvfz pegasus-2.10.0.tar.gz
```

und erhalten einen Ordner „/tmp/pegasus“.

Verschieben sie den Ordner „/tmp/pegasus“ nach „/usr/src/pegasus“

```
root@cim-server> mv -R /tmp/pegasus /urs/src/
```

In der Datei „/usr/src/pegasus/README.html findet sie Hinweise zur Installation. Es gibt mehrere Abhängigkeiten, die vor dem Kompilieren/Installieren gelöst werden müssen:

```
root@cim-server> apt-get install make g++ flex bison doxygen libicu38 openssl gzip sqlite
```

Zum Kompilieren von openPegasus sind drei Umgebungsvariable in „/etc/profile“ zu setzen, dafür rufen sie die Datei im Editor „nano“ auf.

```
root@cim-server> nano /etc/profile
```

Am Ende der Datei fügen sie folgende Zeilen ein:

```
PEGASUS_ROOT="/usr/src/pegasus"
export PEGASUS_ROOT
PEGASUS_HOME="/usr/bin/pegasus"
```

```
export PEGASUS_HOME
PEGASUS_PLATFORM=LINUX_IX86_GNU
export PEGASUS_PLATFORM
```

und speichern die Datei. Die Variable PEGASUS_PLATFORM ist für ihr System anzupassen, der hier verwendete Wert steht für Intel 32-Bit- Kompatible Systeme. Hilfe hierzu finden sie in der README.html

Um sicherzustellen, dass die Variablen zu Verfügung stehen, wird ein Neustart durchgeführt:

```
root@cim-server> shutdown -r now
```

Nach dem Neustart können sie mit dem Kompilieren von openPegasus beginnen. Melden sie sich wieder als „root“ an und wechseln sie nach /usr/src/pegasus.

```
root@cim-server> cd /usr/src/pegasus
```

Starten sie das Kompilieren und Installieren von openPegasus mit:

```
root@cim-server> make
```

Sie sehen eine Vielzahl von Meldungen die über den Bildschirm laufen. Dieser Vorgang dauert, je nach Hardware- und Software-Konfiguration, etwa 45 Minuten. Nach Abschluss der Installation kann der Server- Daemon mit:

```
root@cim-server> /usr/bin/pegasus/bin/cimserver
```

gestartet werden.
Zum Beenden des Server-Daemon verwenden sie:

```
root@cim-server> /usr/bin/pegasus/bin/cimserver -s
```

Um openPegasus ohne Eingabe des Pfades zu starten, editieren sie erneut die Datei „/etc/profile“ und fügen an die Variable „PATH“ den Pfad „/usr/bin/pegasus/bin/“ an.

Damit ist die Grundinstallation abgeschlossen!

G Datenträger (DVD)

An dieser Stelle befindet sich der Datenträger (DVD).

Die DVD enthält:

- Diese Arbeit im PDF-Format
- Die verwendete und unter Kapitel C aufgelistete Software
- Die mit Umbrello erstellten UML-Diagramme
- Den Eclipse-Workspace
- Im PDF-Format vorliegende Quellen

H Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24 (5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 29.03.2011 _____

Ort, Datum Unterschrift