Fachbereich Elektrotechnik und Informationstechnik
Institut für Datentechnik
Fachgebiet Rechnersysteme
Prof. Dr.-Ing. Hans Eveking

Technische Universität Darmstadt

Diplomarbeit

Entwicklung eines Softwaresystems
zur Planung und Inbetriebnahme
von Gebäudeautomationssystemen

von

Oliver Alt
Matrikel-Nr. 1024577
Juli 2003

Betreuer: Prof. Dr. Wolfgang Kastner (TU-Wien)
Prof. Dr. Hans Eveking (TU-Darmstadt)

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Dreieich, den 24. Juli 2003 Oliver Alt

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Gebäudeautomation spielt im Zweck-, aber auch im Wohnbau eine zunehmend wichtigere Rolle. Durch die Vernetzung der Gebäudefunktionen (Licht, Klima, Alarm und Jalousien) können Gebäude individuell und flexibel auf die jeweiligen Bedürfnisse der Bewohner und Nutzer angepaßt werden.

Zur Umsetzung der Gebäudeautomation stehen dem Bauherren und Installateur bereits heute eine Reihe von verschiedenen Systemen zur Verfügung. Neben dem in Europa am weitesten etablierten Europäischen Installationsbus, seien auch noch LONWorks und LCN genannt. Die Auswahl an Datennetzen für die Gebäudeautomation wird sich in Zukunft sicherlich noch erhöhen, zumal die genannten Netze alle mit für die heutigen Verhältnisse recht geringen Datenraten arbeiten. Es gibt auch bereits erste Ansätze Ethernet, CAN oder FireWire als Netz im Gebäude zu verwenden. Voraussichtlich werden diese Netze aber eher im Backbone Bereich eingesetzt werden, da für die Schalt- und Steuerungsaufgaben in einem Gebäude die niedrigen Datenraten der etablierten Gebäudebusse durchaus ausreichend sind.

Bei allen Systemen der Gebäudeautomation müssen die Installationskomponenten (Sensoren und Aktoren) programmiert und konfiguriert werden. Erst danach hat der Sensor die Informationen, welche Aktoren er wie ansteuern muß. Gleiches gilt natürlich auch für die Aktoren. Fast alle Systeme nutzen zur Konfiguration eine spezielle PC Software. Der PC wird dann mit Hilfe von Buskopplern mit dem Gebäudebus verbunden und die Programmierung der Geräte kann erfolgen.

Jedes Bussystem bringt seine eigene proprietäre Konfigurations- und Installationssoftware mit. Die Programme und Datenformate sind zueinander inkompatibel. Dies macht netzwerkübergreifende Installationen schwierig, wenn nicht sogar unmöglich. Die meisten Programme sind bereits seit ca. 10 Jahren auf dem Markt. Neuerungen in der Software- und (PC-)Netzwerktechnik sind daher gar nicht oder nur sehr begrenzt darin enthalten. Zudem sind

alle Programme an ein Betriebssystem gebunden (DOS oder Windows). Freie Systeme wie Linux oder auch Apple-Macintosh können daher zur Zeit noch nicht für die Planung und Inbetriebnahme von Gebäudeautomationssystemen verwendet werden.

Aufgabenstellung

Ziel dieser Arbeit ist es daher, ein plattformunabhängiges und nach Möglichkeit auch automationssystemübergreifendes Werkzeug zur Planung und Konfiguration von Gebäudebussystemen zu entwerfen und zu implementieren. Beim Entwurf und der Implementation sollen die Techniken und Erkenntnisse der Softwaretechnik des aktuellen Standes eingesetzt werden. Die Datenformate für die Speicherung der Projektierungsdaten sollen so entworfen werden, daß verschiedene Bussysteme die gleichen Grunddaten haben. Beim Entwurf des Systems soll darauf geachtet werden, daß es möglichst modular und leicht erweiterbar ist, da es sicherlich im Zeitrahmen dieser Arbeit nicht möglich sein wird, mehr als ein Bussystem komplett zu unterstützen.

Ausblick auf den Rest der Arbeit

- Kapitel 2 gibt eine Einführung und einen Überblick über Gebäudeautomationstechniken und die am Markt befindlichen Konfigurationswerkzeuge. Es werden außerdem die Gemeinsamkeiten der verschiedenen Systeme zusammengestellt.

- Kapitel 3 erläutert die Anforderungen an das in dieser Arbeit zu entwerfende Softwaresystem, die aufgrund der Sichtung bestehender Systeme und einer durchgeführten Anforderungsanalyse gefunden wurden.

- Kapitel 4 beschreibt den Lösungsansatz, der aufgrund der Anforderungen und der Aufgabenstellung gewählt wurde. Hier werden neben den funktionalen Aspekten des Systems auch die eingesetzten Softwaretechnologien beschrieben.

- Kapitel 5 stellt die wichtigsten Aspekte der Implementierung des Softwaresystems vor. Es wird aufgezeigt, auf welche Weise der Lösungsansatz realisiert wurde.

- Kapitel 6 enthält das Fazit der Arbeit und einen Ausblick über die Weiterentwicklungsmöglichkeiten der hier gemachten Implementierung.

Kapitel 2

Gebäudeautomation

In diesem Kapitel erfolgt eine Übersicht über Gebäudeautomation und Gebäudeautomationssysteme und deren heutigen Einsatz.

2.1 Grundlagen

Moderne Gebäude besitzen eine Reihe von elektrisch betriebenen oder kontrollierten Geräten bzw. Funktionen, die bedient und/oder gesteuert werden müssen. Solche Geräte sind z.B. Lampen, Jalousien oder Wärmestellventile. Die moderne Gebäudeautomationstechnik versucht nun möglichst alle Geräte und Funktionen des Gebäudes miteinander zu vernetzen, so daß daraus ein Komfort- und Sicherheitsgewinn resultiert. Man unterscheidet Sensoren (z.B. Schalter) und Aktoren (z.B. zum Schalten von Lampen). In der konventionellen Elektroinstallation werden die Sensoren direkt elektrisch mit den Aktoren verbunden. Verschiedene Gebäudefunktionen existieren nebeneinander (Licht, Jalousien, Heizung) ohne Querverbindung untereinander. Sollen zusätzliche Funktionen (z.B. neue Schalter) realisiert werden, sind meist erhebliche und kostspielige bauliche Maßnahmen erforderlich.

In einem Gebäude mit Automationssystem sind die einzelnen Geräte miteinander vernetzt und tauschen Daten aus. Die Geräte sind nicht mehr zwangsläufig elektrisch miteinander gekoppelt (230 V Netz), sondern die Aktoren bekommen einen digitalen Schaltbefehl des Sensors. Neue Funktionalitäten sind durch einfache Umprogrammierung der Geräte möglich, ohne daß bauliche Maßnahmen erforderlich sind. Alle Geräte sind nur noch über ein serielles Bussystem miteinander verbunden, so daß sich auch noch der Verkabelungsaufwand im Gebäude reduziert.

Auch Zentralfunktionen lassen sich mit Hilfe von Gebäudeautomation viel einfacher realisieren, da die Geräte so programmiert werden können, daß

sie zusätzlich zu den normalen Tastbefehlen auch noch auf zentrale Befehle reagieren.

Die bisher realisierten Gebäudeinstallationen mit Automationssystemen ähneln fast ausschließlich einer konventionellen Elektroinstallation, die mit Hilfe des Automationssystems nachgebildet wird. Eine umfassende Vernetzung und Integration *sämtlicher* Geräte eines Gebäudes scheitert heute meist noch an den Kosten und den Inkompatibilitäten zwischen verschiedenen Geräteklassen (weiße Ware, braune Ware, Elektroinstallation).

Zukünftige automatisierte Gebäude werden voraussichtlich aber um eine Reihe weiterer Sensoren und Querverbindungen verfügen. Die Bedienung wird sich dahingehend ändern, wie es in [DKS00, Seite 4] beschrieben wird:

> *Daß die heutigen Lichtschalter bald nur noch als Notschalter anzusehen sind, wird verständlich, denn der Raum weiß, wann die sich im Raum aufhaltende Person Licht benötigt und wann nicht. Die intelligente Steuerung kann dann auch entziffern, welche Beleuchtung notwendig ist, denn wenn etwas von einem Monitor abgelesen werden muß, sind andere Lichtverhältnisse notwendig als für das Lesen in einer Zeitschrift. Zudem wird die zukünftige Person verbal äußern, wenn sie andere Lichtverhältnisse haben möchte.*

2.2 Bussysteme der Gebäudeautomation

Es gibt heutzutage eine Vielzahl von verschiedenen Bussystemen, mit denen sich eine Gebäudeautomation realisieren läßt. Die Unterschiede zwischen den verschiedenen Systemen liegen hauptsächlich auf der physikalischen Schicht der Netze. Alle Netze haben Feldbus-Charakter und bieten eine große Störunempfindlichkeit und eine möglichst einfache Installation.

2.2.1 Der Europäische Installationsbus (EIB)

Der EIB ist das in Europa bekannteste und verbreitetste Netzwerk für die Gebäudeautomation. Der Bus wurde ursprünglich von Siemens entwickelt, wird heute aber von eine übergeordneten Organisation standardisiert und weiterentwickelt (*EIB Bus Association, EIBA*). Um die Zusammenarbeit von EIB Geräten zu garantieren, werden die Geräte bevor sie auf den Markt kommen einer (kostenpflichtigen) Konformitätsprüfung durch die EIBA unterzogen. Bestehen die Geräte diese Tests, können sie als garantiert kompatibles EIB Gerät verkauft werden und bekommen das EIB Logo.

EIB Implementierungen gibt es für verschiedenen Medien. Am weitesten verbreitet sind Twisted Pair Leitungen unterster Kategorie (EIB-TP). Weiterhin sind Powerline (EIB-PL) und Funkübertragungssysteme am Markt erhältlich. Die Protokolle der oberen Schichten sind jedoch für allen Medien gleich. Das EIB System besitzt einen ISO/OSI-konformen Protokollstapel mit 7 Schichten, jedoch sind die Sitzungs- und Präsentationsschicht leer.

Die große Verbreitung des EIB beruht auf geschicktem Marketing und der Einbeziehung des Elektrohandwerks von Beginn an. Inzwischen gibt es mehrere tausend EIB fähige Geräte am Markt.

Seit 3 Jahren gibt es nun auch den TP-UART Chip, der zum ersten Mal auf einfache und kostengünstige Weise eine Verbindung zwischen dem EIB und einer RS-232 UART Schnittstelle ermöglicht. Von den etablierten Elektrofirmen wird von dieser Technologie, zumindest bisher, noch kein oder wenig Gebrauch gemacht. Kleinere Firmen (z.B. F. Schlaps und Partner GmbH) und Forschungseinrichtungen sind im Moment die Hauptnutzer. Meist verzichten sie auf die doch sehr kostenintensive Zertifizierung ihrer Entwicklungen durch die EIBA und vertreiben diese ohne das offizielle EIB Logo.

Eine ausführliche Beschreibung des EIB findet sich in [DKS00] und [EIB99]. Auf einzelne technische Details wird in dieser Arbeit an den Stellen, wo es erforderlich ist, näher eingegangen.

2.2.2 LONWorks

LONWorks ist ein von der Fa. Echelon entwickeltes Feldbussystem. Neben der Anwendung in der Gebäudeautomation wird es auch in industriellen Anwendungen vielfach eingesetzt. Ein LON Netzwerk besteht aus einer Reihe von Knoten. Diese Knoten haben im Gegensatz zum EIB bereits ab Werk eine eindeutige Adresse, mit der sich eine Kommunikation aufbauen läßt. Dadurch erleichtert sich die Konfiguration und auch die Realisierung von *Plug and Play* ähnlichen Diensten.

Wie auch beim EIB können für die LON Kommunikation verschiedene Übertragungsmedien zum Einsatz kommen. Am Markt erhältlich sind z.B. Niederspannung und Powerline.

Das Protokoll, mit dem LONWorks arbeitet, wird *LONTalk* genannt. Eine Reihe von bereits verfügbaren Mikrocontrollern, sogenannte Neuron Chips, unterstützen den LONTalk Standard auf Hardwareebene.

Zusätzlich zu den LON Knoten gibt es von der Fa. Echelon eine Reihe von Gateway Lösungen zur Anbindung eines PC an das LON Netzwerk. Benutzte Schnittstellen sind hierbei RS-232, USB oder auch PCI. Weiterführende Informationen zu LONWorks finden sich in [DLS99].

2.2.3 LCN

Das LCN ist ein reines Netzwerk zur Gebäudeautomation. Die Daten werden hier über eine freie Ader eines 5-adrigen 230V Installationskabels übertragen. Da im modernen Wohnungsbau schon heute fast überall 5-adrig verkabelt wird, eignet sich LCN auch hier als einfach nachrüstbares Automationssystem, wenn z.B. beim Bau keine EIB Leitungen vorgesehen wurden.

LCN benutzt ein eigenes Protokoll und versendet Telegramme variabler Länge. Die Telegrammstruktur ist recht flexibel. Dadurch kann einem Dimmaktor z.B. zur Laufzeit des Systems mitgeteilt werden, in welcher Zeit er einen bestimmten Helligkeitswert erreichen soll. Bei anderen Bussystemen müssen solche Daten bei der Projektierung fest in die Aktoren einprogrammiert werden.

Einen Überblick über LCN findet sich in Anhang A.

2.3 Gemeinsamkeiten der Gebäudebussysteme

Alle Bussysteme, die in der Gebäudeautomation eingesetzt werden, benutzen das Prinzip von Sensor und Aktor. Sensoren nehmen hierbei Eingaben von Benutzern oder auch bestimmte Meßdaten (z.B. Wetterdaten oder Helligkeitswerte) auf. Die Sensoren sind über die Programmierung mit bestimmten Aktoren logisch verbunden und melden die aufgenommenen Daten an diese weiter.

Die Aktoren verarbeiten die von den Sensoren gelieferten Daten und agieren in der ihrer Programmierung und Funktion entsprechenden Weise, d.h., sie schalten, dimmen oder stellen eine Größe ein.

Neben den Sensoren und Aktoren existieren noch Busgeräte die Schnittstellen zu anderen Systemen bieten. Solche Systeme sind z.B. RS-232, Ethernet, etc.

2.3.1 Sensoren in der Gebäudeautomation

In der Gebäudeautomationstechnik gibt es folgende Sensortypen:

Tastsensoren. Mit Tastsensoren werden Tastereingaben der Benutzer entgegengenommen. Tastsensoren werden meist für binäre Eingaben verwendet (an/aus). Längeres Drücken kann aber auch dazu verwendet werden, um stetige Wertänderungen zu aktivieren (z.B. dimmen). Die

meisten Tastsensoren enthalten auch noch Status LEDs, die programmierbar sind und oft dazu verwendet werden, den Status des durch den Sensor gesteuerten Aktors anzuzeigen.

Bewegungssensoren. Bewegungssensoren nehmen Bewegung von Personen wahr und geben wie Tastsensoren binäre Signale aus (an/aus). Stetige Wertänderungen sind mit Bewegungssensoren nicht möglich.

Raumtemperaturregler. Raumtemperaturregler nehmen die momentane Raumtemperatur auf und bieten dem Benutzer zusätzlich noch einige Bedienelemente, um auf die vorprogrammierten Raumtemperaturen manuell Einfluß zu nehmen. Viele besitzen noch einige Statusanzeigen, die Aktor- und Betriebszustände des geregelten Systems anzeigen.

Bedientableaus. Bedientableaus sind universell programmierbare Anzeigeelemente. Diese sind je nach Typ entweder text- oder graphikbasiert. Über Eingabeelemente (Tasten oder Touchscreen) kann mit der Anzeige interagiert werden und es können Aktionen ausgelöst werden. Bedientableaus bieten durch ihre Programmierbarkeit die universellste Schnittstelle zwischen dem Benutzer und dem Gebäudeautomationssystem. Sie können daher auch als Ergänzung bzw. Ersatz der oben genannten Sensortypen eingesetzt werden.

Binäreingänge. Binäreingänge nehmen binäre Daten auf und geben diese entsprechend ihrer Programmierung an die Aktoren weiter. Typische Anwendungsbeispiele sind die Anbindung von *normalen* Schaltern oder Alarmkontakten.

Analogeingänge. Analogeingänge verarbeiten die Daten von analogen Sensoren und steuern entsprechend ihrer Programmierung Aktoren an. Mit Hilfe von Analogeingängen können alle denkbaren Informationen, die durch analog arbeitende Sensoren erfaßbar sind, dem Gebäudeautomationssystem zugänglich gemacht werden.

Schaltuhren. Schaltuhren dienen dazu, Vorgänge zeitgesteuert zu aktivieren. Es gibt verschiedene Typen die Quarz oder auch DCF-77 (Funkuhr) gesteuert arbeiten.

Infrarot-Empfänger. Infrarotempfänger nehmen Befehle von IR Fernbedienungen entgegen und senden dann passende Kommandos zu den Aktoren.

Spezielle Wettersensoren. Wettersensoren integrieren meist mehrere verschiedene Sensoren, um Wetterdaten zu erfassen. Dazu gehören Temperatur, Luftdruck und Regen. Die Daten können dann für verschiedenste Steueraufgaben verwendet oder auch über ein Tableau angezeigt werden.

2.3.2 Aktoren in der Gebäudeautomation

Die Zahl der unterschiedlichen Aktoren in der Gebäudeautomation ist sehr viel geringer als die Zahl der unterschiedlichen Sensoren. Man unterscheidet folgende Typen:

Schaltaktoren. Schaltaktoren reagieren auf binäre Schaltbefehle und kennen die Zustände *ein* und *aus*. Schaltaktoren werden für binäre Steuerungsaufgaben eingesetzt. Typische Beispiele sind Lampen, Türöffner und auch Heizgeräte.

Dimmaktoren. Dimmaktoren können in ihrem Spannungsbereich beliebige Werte am Ausgang ausgeben. Die Spannungswerte sind variabel zwischen 0 und 100 Prozent der angelegten Spannung. Dimmaktoren steuern meist Lampen oder Motoren an. Für bestimmte Lasten sind auch spezielle Dimmaktoren nötig (z.B. Leuchtstoff- oder Halogenlampen).

(Thermoelektrische) Stellantriebe. Stellantriebe werden verwendet, um Ventile zu steuern. Der Haupteinsatzbereich liegt hier in Steuerungsaufgaben im Heizungsbereich (Mischer, Heizkörper). Stellantriebe können, wie Dimmaktoren auch, einen Bereich von 0 bis 100 Prozent der zu steuernden Größe abdecken. In der Ansteuerung sind sie daher identisch mit den Dimmaktoren.

Für die oben genannten Aktoren existieren auch Kombigeräte, die beispielsweise Schalt- und Dimmaktoren enthalten.

2.3.3 Applikationsmodule

In einigen Bussystemen gibt es spezielle Applikationsmodule, die Daten über den Bus empfangen und gemäß der in ihnen laufenden Applikation Verknüpfungen herstellen und Aktoren ansteuern. Applikationsmodule haben nur eine Busschnittstelle und keinerlei Sensor- und Aktorfunktionen. Die Funktion ist vergleichbar mit einem Makro in einer Programmiersprache, das durch ein Bustelegramm aktiviert wird.

2.4 Datentypen der Gebäudeautomationstechnik

In der Gebäudeautomationstechnik gibt es einige grundlegende Datentypen, die ausreichen um sämtliche für eine Gebäudeautomation benötigten Funktionen auszuführen. Nachfolgend werden die Datentypen aufgelistet und erläutert.

Binärsignale (0/1). Binärsignale haben die Größe 1 Bit und werden für sämtliche binären Schalt- und Steuerungsaufgaben eingesetzt.

Ganzzahlige Wertsignale. Ganzzahlige Wertsignale unterschiedlicher Breite (8, 16, ... Bit) werden benötigt, um Aktionen mit Wertvorgaben zu erledigen. Hierzu zählen z.B. die Vorgabe von Helligkeitswerten, Timerwerte oder die Übertragung von Sensorwerten mit ganzzahligem Charakter.

Fließkommawerte. Fließkommawerte werden benötigt, um nichtganzzahlige Werte wie z.B. von Temperatursensoren zu übertragen. In den Bussystemen werden hierfür aufgrund verschiedener Genauigkeitsanforderungen und Speichergrößen unterschiedliche Codierungen eingesetzt.

Datums- und Uhrzeitformate. Datumsformate dienen der Übertragung von Datums- und Zeitinformationen, wie sie z.B. für zeitgesteuerte Schaltaufgaben benötigt werden.

Zeichen und Strings. Zeichen- und Stringformate entsprechen meist dem ASCII Standard und werden benötigt, um textuelle Meldungen über das Bussystem zu übertragen. Einsatz findet dies vor allem in der Ansteuerung von LCD Anzeigeelementen.

2.5 Softwarewerkzeuge zur Gebäudeautomation

2.5.1 Die EIB Tool Software (ETS)

Die ETS ist *die* Standardsoftware zur Konfiguration und Inbetriebnahme von EIB Installationen. Sie ist in C++ implementiert und verwendet Komponenten- bzw. Modultechnik. Dadurch läßt sich die Software um weitere Funktionen erweitern. In der Praxis wird davon aber kaum Gebrauch gemacht.

In der Standardausstattung besteht die ETS Version 2 aus folgenden Komponenten:

Projektierung. In der Projektierung werden alle Komponenten der geplanten EIB Installation eingegeben. Die Geräte werden per Drag and Drop Verfahren miteinander verknüpft (Sensoren-Aktoren). Es werden dadurch die Adressen der Geräte festgelegt und die zur Programmierung notwendigen Daten bestimmt. Die Parameter der Geräte lassen sich an die jeweilige Anwendungssituation anpassen.

Inbetriebnahme. In der Inbetriebnahme erfolgt das Programmieren und Parametrieren der realen EIB Geräte mit den in der Projektierung festgelegten Daten. Hierzu gehören die Programmierung der physikalischen Adresse, der Applikation und der Parameter der EIB Geräte. Im Inbetriebnahmeteil der ETS stehen desweiteren noch Funktionen zur Fehlersuche (Telegrammtracer) und erweiterten Kommunikation (u.a. Auslesen von Geräteinformationen) zur Verfügung.

Projektverwaltung. Alle Daten der ETS werden in einer zentralen Datenbank gehalten. Diese umfaßt sowohl die Daten der EIB Geräte, als auch die Daten der Installationsprojekte. Die Projektverwaltung dient zum Export und Import von einem oder mehreren Projekten. Da die Inbetriebnahme häufig auf der Baustelle stattfindet und der dort verwendete PC meistens nicht mit jenem identisch ist, mit dem die Projektierung erfolgte, müssen die Daten zu diesem extrahiert und übertragen werden können.

Produktverwaltung. Dieser ETS Programmteil dient zum Importieren und Verwalten der Produktdaten für die EIB Geräte. Die Gerätedaten werden von den Herstellern in einem Standardformat zur Verfügung gestellt und können mit der Produktverwaltung ganz oder teilweise zur ETS Datenbank hinzugefügt werden.

Konvertierung. In diesem Programmteil lassen sich Daten aus älteren ETS Versionen (1.x) in das Datenformat der Version 2 konvertieren.

Neben diesen Standardfunktionen sind noch Module erhältlich, die die ETS um weitere Funktionen erweitern. Das bekannteste Zusatzmodul ermöglicht die Eingabe von Gebäudeplänen und den Export im DXF Format [DKS00, S. 257]. Dieses Format ist das Standardformat, um im CAD Bereich Daten auszutauschen. Das CAD Modul wird jedoch kaum eingesetzt, da die Projektierung mit der sonst in der ETS üblichen strukturellen Ansicht schneller von der Hand geht und die Gebäudepläne oft schon mit anderen CAD

Werkzeugen gezeichnet werden (z.B. AutoCAD). Mit diesen professionellen CAD Werkzeugen lassen sich mittlerweile auch schon EIB Symbole einbinden, so daß die (neuerliche) Eingabe der Daten in die ETS nur doppelte Arbeit mit sich bringt.

Die ETS ist ausschließlich in einer Version für Windows erhältlich. Die Verbindung zum EIB geschieht über die erste oder zweite serielle Schnittstelle des PC und einen EIB-RS232 Buskoppler. Seit einiger Zeit ist eine Internetfähige Version der ETS auf dem Markt, die iETS genannt wird. Mit der iETS und einem am Installationsort des EIB Systems laufenden Applikationsserver ist es möglich, die Programmierung, mit Ausnahme der physikalischen Adressen[1] , auch über das Internet durchzuführen. Der Installateur braucht somit nicht immer vor Ort zu sein, um (kleinere) Umkonfigurationen vorzunehmen.

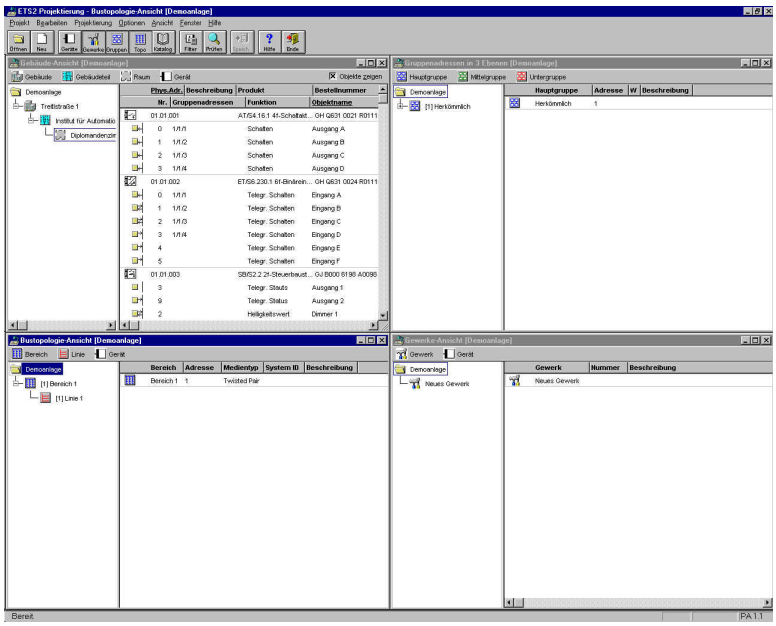

Abbildung 2.1: Bedienoberfläche der EIB Tool Software

Die ETS ist die einzige von der EIBA zertifizierte Software für den EIB. Die EIBA selbst finanziert sich ausschließlich durch Mitgliedsbeiträge und Lizenzgebühren für die ETS. Durch dieses Konzept wird zwar einerseits eine Kompatibilität zu allen zertifizierten EIB Produkten garantiert, andererseits führt ein mangelnder Wettbewerb aber auch dazu, daß Neuerungen erst sehr langsam Einzug in das Produkt halten.

[1]Hierfür muß vor Ort ein Programmiertaster betätigt werden.

2.5.2 Konfigurationssoftware für LONWorks

Im Gegensatz zum EIB bringt LONWorks keine komplette eigene Planungs- und Projektierungssoftware mit, sondern erweitert die Visualisierungssoftware MS Visio durch eigene Softwarekomponenten. Die Geräte einer LON Installation werden auf einem Arbeitsblatt platziert und durch virtuelle Verbindungen verknüpft. Die so entstandenen Daten lassen sich dann über ein PC-LON-Gateway in die LON Geräte programmieren.

Durch den Einsatz von MS Visio konnte der Entwicklungsaufwand auf die Entwicklung passender Softwareadapter (*plug ins*) beschränkt werden. Weiterhin kommen Anwender, die schon einmal mit MS Visio gearbeitet haben, schon nach kurzer Zeit mit dem System zu Recht.

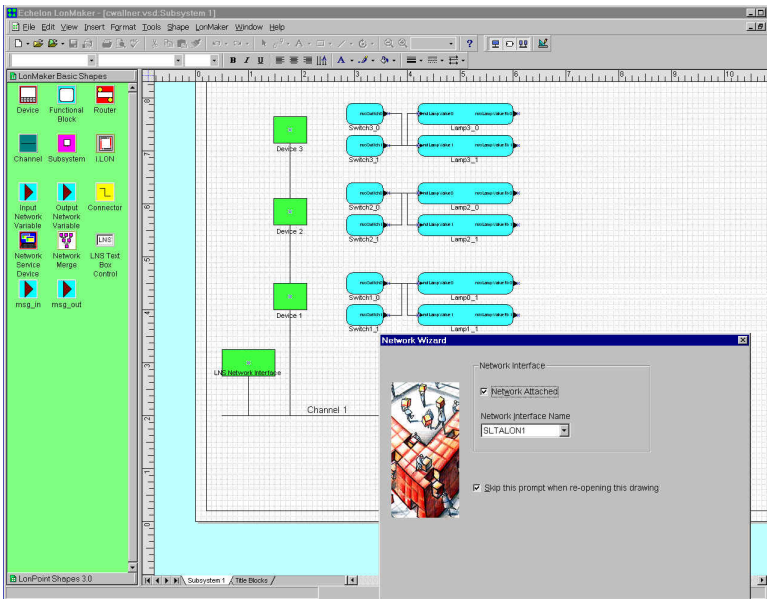

Abbildung 2.2: LON Konfigurationssoftware basierend auf MS Visio

Ein Nachteil des Ansatzes besteht jedoch darin, daß die Software nicht nur an Windows, sondern auch an den Visio Hersteller Microsoft gebunden ist. Mit neuen Versionen von Visio müssen eventuell auch jedesmal die LON Komponenten angepaßt werden.

Kapitel 3

Anforderungen

Dieses Kapitel beschreibt die Anforderungen an das neue Softwaresystem zur Gebäudeautomation.

3.1 Grundsätzliche Anforderungen

Die Anforderungen sind die Resultate aus einer Anforderungsanalyse bestehender Systeme und Schlußfolgerungen aus Gesprächen mit Planern und Installateuren aus der Elektroindustrie. Hierzu wurden Gespräche mit Mitarbeitern der Firmen Scholl in Neuhof-Dorfborn (www.schollgroup.de) und Berger in Wien (www.eib.at) geführt, um positive und negative Erfahrungen mit den bestehenden Systemen (hauptsächlich ETS) zu ergründen. Die beiden genannten Firmen führen beide Großprojekte der Gebäudeautomation/EIB in Deutschland und Österreich aus.

Aus der Anforderungsanalyse ergeben sich folgende notwendige und/oder gewünschte Punkte:

Plattformunabhängigkeit. Bisherige Software für Gebäudeautomation ist meist nur für Windows Plattformen erhältlich. Das in dieser Arbeit zu realisierende System soll allerdings auf möglichst vielen Zielplattformen laufen.

Offenes, XML basiertes Datenformat. Im Gegensatz zu bestehenden System sollen alle Projekt- und Gerätedaten im XML Format gespeichert werden. Dies ermöglicht offene Standards, einfachen Datenaustausch und einfache Weiterverarbeitung.

Unterstützung verschiedener Bussysteme. Die Software soll mehrere Bussysteme zur Gebäudeautomation unterstützten, oder zumindest so

entworfen werden, daß sich ein bisher noch nicht unterstütztes Bussystem in Zukunft leicht anbinden läßt.

Zentrale Datenhaltung mit Versionskontrolle. Ein oft bemängelter Schwachpunkt bei bestehenden Systemen ist eine mangelhafte oder fehlende Versionsverwaltung für die Projektdaten. Daten werden oft vom stationären PC im Büro auf den mobilen Computer für die Baustelle transferiert. Änderungen der Projektdaten auf der Baustelle führen dann leicht zu Inkonsistenzen in den Datenbeständen.

Unterstützung von Mehrsprachigkeit. Bestehende Software steht meist nur in einer Sprachversion zur Verfügung. In Zeiten des Internet und der zunehmenden Internationalisierung sollte die Software daher so gestaltet werden, daß mit geringem Aufwand mehrere Sprachen unterstützt werden. Englisch wird in der Informationstechnologie zwar allgemein vorausgesetzt. Eine Vielzahl der Anwender von Software für Gebäudeautomationssysteme sind aber keine Informatiker sondern Elektriker und Elektromeister, die mit einer muttersprachlichen Software sicher erheblich besser zurecht kommen.

Anbindung von Gebäudeplänen (CAD-Daten). Die durchgehende Einbindung von Gebäudeplänen (Grundrissen, etc.) in allen Projektierungs- und Inbetriebnahmefunktionen einer Gebäudeautomationssoftware ist wünschenswert, da auf diese Art und Weise die Geräte der Gebäudeautomation auch räumlich und nicht nur strukturell den Gebäudeteilen zugeordnet werden können. Die Daten sollten allerdings mit geringem Aufwand aus einem professionellen CAD System importiert werden können, da eine doppelte Eingabe in zwei Systeme wenig sinnvoll erscheint. Mit Hilfe der CAD-Daten ist es dann auch möglich, aufgrund der restlichen Planungs- und Installationsdaten automatisch Bedienoberflächen (z.B. für Bedientableaus) zu generieren. Außerdem kann der Test und die Fehlerdiagnose durch das schnellere Auffinden von Geräten vereinfacht und verbessert werden.

3.1.1 Ähnlichkeit zu bestehenden Systemen

Viele Anwender bestehender Systeme, gerade auch der ETS, kennen ausschließlich diese mit ihrer Bedienung per Drag and Drop. Auch die Ansichten der Geräte und Querverweise untereinander sind ihnen vertraut. Bedienung und Menüstrukturen, die ein Außenstehender (Informationstechniker) vielleicht als Schwachpunkte empfindet, werden von Anwendern gar nicht so

gesehen, da sie durch die tägliche Arbeit mit der Software auch an etwas umständliche Bedienung gewöhnt sind.

Für ein neues Softwaresystem muß daher geklärt werden, in wieweit bestehende Konzepte übernommen oder neue Wege beschritten werden sollen. Nur so stößt die Software dann auch auf Akzeptanz bei den Benutzern. Hier gilt wie überall: *Das beste Produkt ist nichts wert, wenn es vom Kunden nicht angenommen wird!*

3.2 Aktoren

Zur Dokumentation der Anforderungen wird das aus dem *Unified Process* bekannte Verfahren der Aktoren und Anwendungsfälle eingesetzt. Zur textuellen Beschreibung werden daher auch UML Diagramme angefügt.

Für das Softwaresystem wurden folgende Rollen der Aktoren gefunden. [1]

Bewohner. Der Bewohner kann über Bedienelemente die ihn betreffenden Gebäudefunktionen beeinflussen und ändern.

Datenadministrator. Der Datenadministrator ist für alle Daten im System verantwortlich. Hierzu zählen das Generieren, Konvertieren und Archivieren von System- und Projektdaten.

Gebäudemanager. Der Gebäudemanager kontrolliert und überwacht die Gebäudefunktionen im laufenden Betrieb. Er kann über Bedienelemente Einfluß auf den Zustand des Gebäudes nehmen und diesen ändern.

Inbetriebnahmetechniker. Der Inbetriebnahmetechniker ist für die Inbetriebnahme der Gebäudeautomation verantwortlich. Er nutzt hierfür die vom Planer erzeugten Planungsdaten, um die geplante Automation auf ein reales Bussystem zu übertragen. Er programmiert die Geräte mit Adressen und Applikationen, die in der Planungsphase festgelegt wurden, und paßt nötigenfalls die Parametrierung an.

Planer. Der Planer plant die gesamte Gebäudeautomation. Zu seinen Aufgaben gehören die Eingabe der Gebäudedaten und das anschließende Platzieren der Sensoren und Aktoren. Er stellt auch die Verbindungen zwischen den Sensoren und Aktoren her und parametriert sie. Als Arbeitsgrundlage dienen Pläne oder auch Pflichtenhefte mit den Anforderungen des Bauherren.

[1] Selbstverständlich können mehrere Rollen von ein und derselben Person ausgefüllt werden.

Systemtester. Der Systemtester überprüft nach der Programmierung und Inbetriebnahme der Gebäudefunktionen das korrekte Funktionieren der Installation. Hierzu gehören Tests der Zusammenarbeit der Aktoren und Sensoren gemäß der spezifizierten und geplanten Anforderungen.

3.3 Funktionale Anforderungen

Die funktionalen Anforderungen an eine Software zur Gebäudeautomation lassen sich in die Bereiche Planung, Inbetriebnahme, Test, Betrieb und Datenverwaltung einteilen. Die einzelnen Bereiche werden nun detailliert beschrieben.

3.3.1 Planung

Hauptaufgabe der Planung ist es, eine Installation für ein Gebäude aufgrund der Anforderungen durch den Architekten und Bauherren durchzuführen. Hier werden die Komponenten des Automationssystems festgelegt und ausgewählt. Die Gebäudestrukturen und Daten werden in das System eingegeben und die Geräte des Installationssystems ihren Standorten zugeordnet.

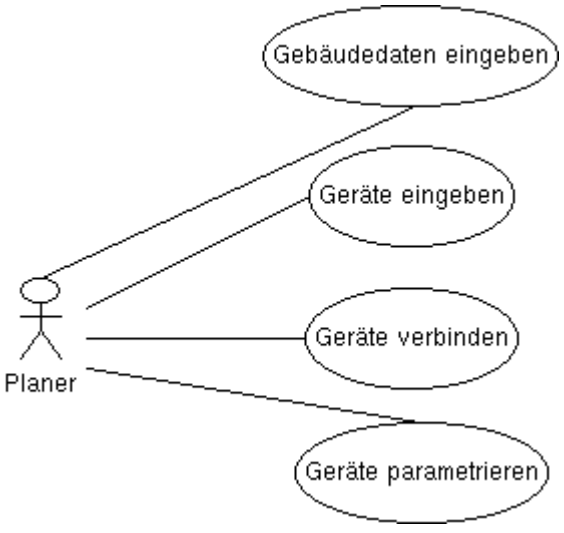

Abbildung 3.1: Planung

Nach der Auswahl der Geräte werden die Geräte untereinander (virtuell) verbunden. Geräte bekommen Adressierungen und Parameter zugeordnet, so daß die nötigen Sensor/Aktor-Verknüpfungen entstehen.

Für die Planung sind eine Reihe von Benutzerschnittstellen denkbar. Ein Gebäude und die Geräte können strukturell und/oder auch graphisch-architektonisch angezeigt werden. Die Querverbindungen zwischen den Geräten sollten möglichst intuitiv herstellbar sein. Bestehende Software nutzt hierbei Drag and Drop (ETS) oder auch das Zeichnen virtueller Verbindungen (LONWorks).

3.3.2 Inbetriebnahme

Die Inbetriebnahme besteht aus der Programmierung der Geräte. Hierbei werden die in der Planung festgelegten Daten in die Speicher der Geräte übertragen. Nach der Inbetriebnahme aller Geräte sollte bei einer korrekt durchgeführten Planung das Gebäudeautomation gemäß der spezifizierten Vorgaben funktionieren und benutzbar sein.

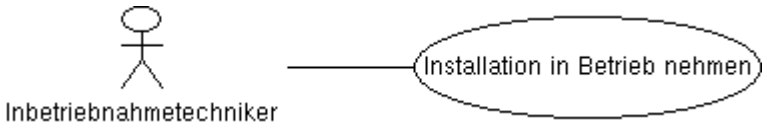

Abbildung 3.2: Inbetriebnahme

Die Software sollte die Inbetriebnahme nach einer korrekt und vollständig durchgeführten Planung weitestgehend automatisch durchführen können. Probleme bei der Programmierung sollten dem Benutzer in aussagekräftigen Fehlermeldungen, nach Möglichkeit mit Vorschlag zur Behebung, mitgeteilt werden. Protokolle, welche die durch das Softwaresystem durchgeführten Aktionen nachvollziehbar machen, sollten automatisch erstellt werden können.

Um ein Gebäudebussystem zu programmieren, muß eine Verbindung zwischen dem Softwaresystem und dem jeweiligen Bussystem hergestellt werden. Heute werden dazu die seriellen Schnittstellen der PC's verwendet. Aufgrund der geforderten Plattformunabhängigkeit des neuen Softwaresystems und der zunehmenden Zahl von PC's die keine seriellen (RS-232) Schnittstellen mehr besitzen, sollte nach Möglichkeiten gesucht werden, über welche (weitere) Schnittstellen der Zugriff auf die Bussysteme möglich ist.

3.3.3 Test

Der Test hängt eng mit der Inbetriebnahme zusammen. Die in der Inbetriebnahme durchgeführte Programmierung muß geprüft werden. Hierbei werden im Idealfall alle denkbaren Zustände der Installation herbeigeführt und dann

geprüft, ob der Zustand den spezifizierten Vorgaben entspricht. Im Falle von nicht erfolgreichen Tests kann eine Umprogrammierung oder eine Parameteränderung an einem oder mehreren Geräten notwendig werden.

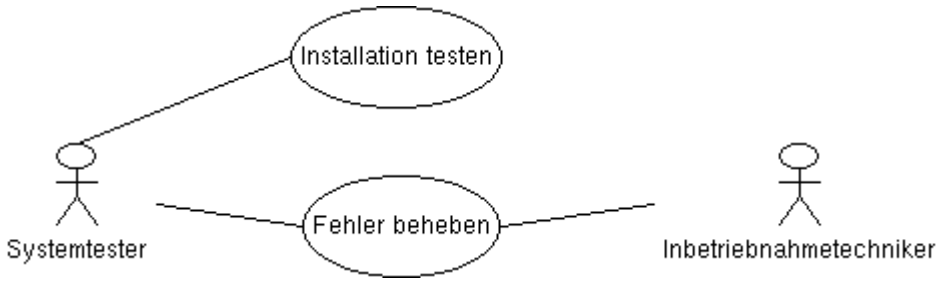

Abbildung 3.3: Test

Die Software sollte Unterstützung beim Testen der Installation bieten. Denkbar sind hier aus den Planungsdaten generierte Tableauübersichten, die es dem Systemtester ermöglichen, die Aktoren zunächst einmal vom PC aus anzusprechen und dadurch die Programmierung zu testen. Auch könnte die Software *virtuelle* Aktoren generieren, die auf Daten, welche von einem realen Sensor kommen, reagieren.

3.3.4 Betrieb

Nach erfolgreicher Inbetriebnahme und Test geht die Gebäudeautomation in den Betrieb über. Die Nutzer des Gebäudes sollten dann die Gebäudefunktionen über das Softwaresystem beeinflussen können. Natürlich muß sichergestellt sein, daß ein Nutzer nur auf die in seinem Nutzungsbereich liegenden Funktionen zugreifen darf.

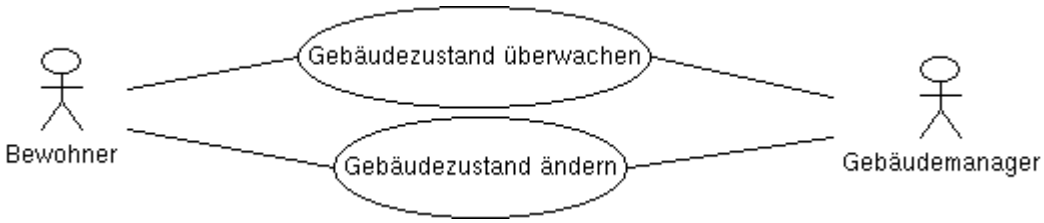

Abbildung 3.4: Betrieb

3.3.5 Datenverwaltung

Daten für eine Gebäudeautomation bestehen aus folgenden Teilen:

Daten der Geräte des Installationssystems. Diese Daten werden meist von den Herstellern der Geräte zur Verfügung gestellt und enthalten Informationen über die Fähigkeiten und Parameter der Geräte. Bei Bussystemen, bei denen die Geräte erst durch die Programmierung ihre Intelligenz erhalten (z.B. EIB), sind auch die kompletten Applikationen im Binärformat enthalten.

Projektdaten. Diese Daten enthalten die Informationen eines Projektes, wie die Gebäudedaten und die darin eingesetzten Geräte, d.h., alle Daten die bei der Planung entstehen, aber auch Informationen, in wieweit ein Projekt bereits in Betrieb genommen wurde.

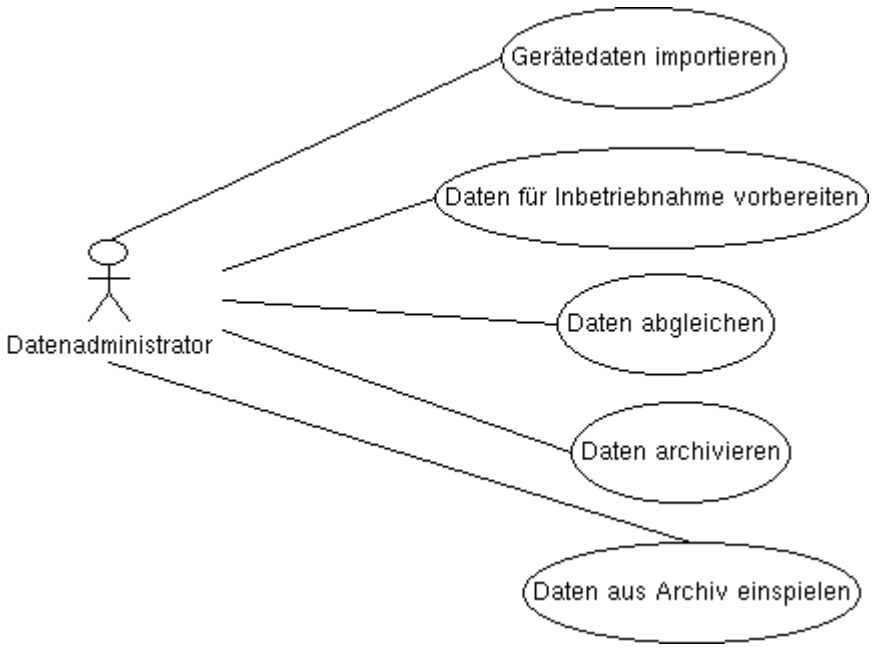

Abbildung 3.5: Datenverwaltung

Die Software muß nun verschiedene Funktionen zur Administration dieser Daten zur Verfügung stellen. Gerätedaten müssen in eine Datenbasis importiert und bei Bedarf auch modifiziert werden können. Die Planungsdaten sollten, da die Inbetriebnahme in vielen Fällen mit einem anderen Rechner als die Planung durchgeführt wird, entweder zentral verwaltet werden

oder aber vom Planungsrechner exportierbar sein. Eine Versionskontrolle der Projektdaten sollte vorhanden sein, um Inkonsistenzen zu vermeiden und Projektänderungen leichter nachvollziehen zu können.

Kapitel 4

Lösungsansatz

Dieses Kapitel beschreibt den Lösungsansatz, der gewählt wird, um die in Kapitel 3 beschriebenen Anforderungen umzusetzen.

4.1 Softwareentwicklungsprozeß

Zur Realisierung dieser Arbeit werden moderne Verfahren und Methoden der Softwaretechnik (*Software Engineering*) eingesetzt. Es wird jedoch nicht ausschließlich nur ein Prozeß wie der *Unified Process (RUP)* oder das *eXtreme Programming* verwendet, sondern es werden Techniken aus beiden Verfahren kombiniert.

Aus RUP werden die Technik der Anwendungsfälle mit den jeweiligen Diagrammen übernommen. Das Software Design hingegen geschieht eher mit Methoden aus dem eXtreme Programming Paradigma. Die Software wird nicht komplett vorher entworfen, sondern das Design entsteht vorwiegend während der Programmierung. Wobei auch bei diesem Ansatz einige Vorüberlegungen zum Einsatz von passenden Entwurfsmustern und Strukturen im Vorfeld gemacht werden müssen.

Der Ansatz bietet sich an, da weitere Anforderungen teilweise erst mit der Implementierung sichtbar werden und das eXtreme Programming hierbei das flexiblere Konzept darstellt. Die gesamte Software wird auch nur durch den Autor der vorliegenden Diplomarbeit entwickelt und kann daher auch ohne große UML Design Modelle überblickt werden.[1]

[1] Ein in Teamarbeit entstehendes System bedarf aufgrund des erhöhten Kommunikationsbedarfes sicherlich mehr Modellierung und Material in Form von UML Diagrammen.

4.2 Programmiersprache

Als Programmiersprache zur Implementierung des Softwarewerkzeuges wird Java [Krü97] gewählt. Java bietet alle Technologien, um die gewünschten Anforderungen umzusetzen. Neben der ohnehin durch die virtuelle Maschine vorhandenen Plattformunabhängigkeit, bietet die Java Klassenbibliothek standardmäßig (fast) alle Funktionen, um ein solches Softwareprojekt durchzuführen. Dazu gehören mit *Swing* eine moderne Bibliothek, um plattformunabhängige graphische Benutzeroberflächen zu implementieren. XML Werkzeuge [McL01] und Internationalisierung sind ebenfalls bereits in der Standard Edition integriert. Durch die Java 2 Enterprise Edition läßt sich außerdem die Klassenbibliothek so erweitern, daß sich auch Web und Client-Server basierte Dienste mit relativ geringem Aufwand umsetzen lassen.

Neben den Standardbibliotheken gibt es auch eine Vielzahl von frei erhältlichen weiteren Komponenten, die beispielsweise den Umgang mit XML erheblich vereinfachen. Bei der gesamten Umsetzung und Konzeption der Lösung werden ausschließlich frei erhältliche Werkzeuge und Bibliotheken eingesetzt.

Die Möglichkeit C# anstelle von Java zu verwenden, scheidet aus dem Grunde aus, da es für C# momentan noch keine plattformübergreifene Bibliothek zur Erstellung von graphischen Oberflächen gibt.

4.3 Werkzeuge

Die Entwicklung des Systems selbst findet auf einem Linux Rechner statt. Bis auf ein einziges eingesetztes Werkzeug sind jedoch alle Entwicklungswerkzeuge in Java implementiert, wodurch auch im Entwicklungsprozeß Plattformunabhängigkeit gewährleistet ist.

Folgende Werkzeuge werden zur Umsetzung der Arbeit verwendet:

Umbrello. Umbrello ist ein UML Editor und das einzige nicht plattformunabhängige Werkzeug. Der Editor läuft unter Linux/KDE 3 und steht unter der GPL. Er bietet Funktionen zum Erstellen der meisten UML Diagramme und steht zur Zeit noch im Entwicklungsstadium. Als Dateiformat benutzt er ein XML Format.

Eclipse. Eclipse wurde ursprünglich von IBM initiiert und ist eine freie Entwicklungsumgebung für verschiedene Programmiersprachen. Standardmäßig wird die Entwicklung von Java unterstützt. Die Leistungsfähigkeit steht der von kommerziellen System (z.B. Borland JBuilder) in nichts nach. Insbesonders ist JUnit (s.u.) fest in Eclipse integriert und

ermöglicht es, auf einfache Art und Weise Komponententests durchführen zu lassen. Eclipse bietet auch erweiterte Refakturierungsfunktionen zum Umstrukturieren der Programmsystem-Struktur an.

JUnit. JUnit ist das von Kent Beck und Erich Gamma entwickelte Test *Framework* für Unit Tests in Java. Mit JUnit lassen sich Testsuiten aufbauen, die automatisch ablaufen können, um die Korrektheit der Implementation von Methoden oder ganzen Klassen gegenüber der durch Testklassen repräsentierten Spezifikation zu testen.

Log4j. Log4j ist eine vom Apache Jakarta Projekt entwickelte Java Bibliothek, die es ermöglicht, komplexere und strukturiertere Ausgaben zur Diagnose und Fehlersuche auf die Konsole auszugeben. Log4j stellt daher einen Ersatz für die sonst für diese Zwecke eingesetzte Anweisung `System.out.println()` dar.

Weiterführende Informationen sind über die Internetseiten, die am Ende der Arbeit aufgeführt sind, erhältlich.

4.4 Einsatz von Entwurfsmustern

Beim Entwurf und der Implementierung des Softwaresystems werden die in [GH+95] beschriebenen Entwurfsmuster, wo immer möglich verwendet. Durch den Einsatz der Muster bekommt das System eine bessere Struktur und Wartbarkeit. Die einzelnen Teile der Software werden so gestaltet, daß sie so wenig wie möglich untereinander verknüpft sind. Dies erleichtert Erweiterungen und den Einsatz von hier entwickelten Softwareteilen/Softwarekomponenten in anderen Anwendungen.

4.5 Grundprinzipien des Softwaresystems

In naher und mittelfristiger Zukunft wird die Vernetzung innerhalb von Gebäuden zunehmen. Die vorhandenen Netze wie Ethernet, Telefon, Kabelfernsehen und Gebäudeautomationsnetz werden gekoppelt werden. Durch diese Migration ergeben sich neue Möglichkeiten für die Steuerung der Gebäudefunktionen. Denkbar sind hierbei PDA oder auch TV basierte Steuerungen. Das den bisherigen Tastsensoren dann wahrscheinlich eine untergeordnetere Rolle als heute zukommen wird, wurde bereits in 2.1 angedeutet.

Wie auch immer sich die Sensorik in der Zukunft verändern wird, die Aktoren werden auch in Zukunft noch benötigt werden, um die Lampen, Ja-

lousien, etc. zu steuern. Die zu realisierende Software setzt deshalb, im Gegensatz zu bestehenden Systemen, bei den zu aktivierenden Elektrogeräten an. Dazu wird folgende Abgrenzung eingeführt:

Aktive Komponenten. Zu den aktiven Komponenten gehören alle Geräte, die über das Installationsbussystem miteinander kommunizieren, also Sensoren und Aktoren des Installationsbusses. Ein EIB Netzteil hingegen zählt nicht zu den aktiven Komponenten, da es selbst keine Buskommunikation betreibt.

Passive Komponenten. Zu den passiven Komponenten gehören alle an der Installation beteiligten Geräte, die nicht direkt über den Bus angesprochen werden. Dies sind Lampen, Jalousien und Ventile. Diese Komponenten müssen elektrisch mit Aktoren verbunden werden, da sie keine eigenen Busankoppler besitzen. Weitere unter diese Kategorie fallenden Dinge sind z.B. Leitungen und Netzgeräte.

In der Planungsphase sollen daher zunächst die passiven Komponenten platziert werden (Lampen, Jalousien, Ventile). Das System soll danach in der Lage sein passende Aktoren aufgrund der spezifischen Eigenschaften der passiven Geräte zu suchen und zu parametrieren.

Nach der Zuordnung der Aktoren, können dann auch passende *reale* Sensoren einfach gefunden und hinzugefügt oder auch webbasierte Steuerungen generiert werden. Die Festlegung der Sensoren vereinfacht sich spürbar, da die Aktoren schon im Arbeitsschritt vorher mit festgelegten Adressen und Applikationen versorgt sind. Auch ein Test der Aktoren ist einfach realisierbar, da das Softwaresystem Schnittstellen für die Aktivierung der passiven Komponenten bereitstellen könnte und man die Funktion dieser damit einfach testen kann.

Bei bestehenden Systemen, wie z.B. der ETS tauchen die Endgeräte (passive Komponenten) bei der Planung überhaupt nicht auf. Es werden dort ausschließlich Sensoren und Aktoren des Bussystems eingegeben und per Hand querverbunden. So wird es auch möglich, z.B. versehentlich eine Lampe mit einem Raumtemperaturregler zu verbinden. Das neue System vermeidet dies aufgrund des vorliegenden Wissens über die an den Aktoren befindlichen Endgeräte.

4.6 Systemdaten

Als eine Hauptanforderung an das neue Softwaresystem gilt der Entwurf eines erweiterbaren und zukunftsfähigen Datenformates für die zu speichern-

den Daten. Die Daten sollen im XML Format aufbereitet werden, da dies ein strukturiertes, textbasierendes Datenformat ist und außerdem leicht mit (fast) jeder Programmiersprache bearbeitet werden kann.

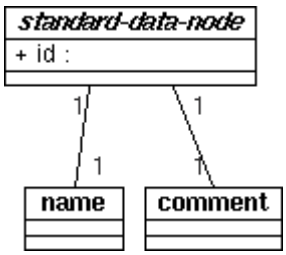

Abbildung 4.1: Standard-XML-Datenknoten

Um den Umgang mit den XML Daten zu erleichtern, wird in dieser Arbeit zur Speicherung von Information ein Standard-Datenknoten definiert. Jeder dieser Knoten hat eine ID als Attribut, die aus dem Knotennamen und einer Nummer besteht. Über diese ID lassen sich eindeutige Querverbindungen herstellen. Jeder der Knoten enthält außerdem standardmäßig zwei Untertags für Namen und Kommentar. Weitere Unterknoten hängen vom jeweiligen Verwendungszweck ab und müssen explizit implementiert werden. Abbildung 4.1 verdeutlicht den Aufbau des Standard-Datenknoten.

4.6.1 Anfallende Daten bei der Gebäudeautomation

Bei einem Softwaresystem zur Planung und Inbetriebnahme von Gebäudeautomationen fallen sowohl projektübergreifende (globale) als auch projektbezogene Daten an. Die projektübergreifenden Daten enthalten Informationen über die Geräte der Automationssysteme, die bei der Planung der Gebäudeautomation Verwendung finden. Die projektbezogenen Daten beinhalten alle Informationen des mit dem Softwaresystem realisierten Projektes. Dies sind Daten mit Informationen über die Struktur des Gebäudes und der Elektroinstallation.

4.6.2 Aufspaltung der projektbezogenen Daten

Aufgrund der oben gemachten Feststellung zur Notwendigkeit von projektbezogenen und projektübergreifenden Daten bietet sich zunächst als Lösungsansatz an, pro Projekt eine XML Datei mit allen nötigen Informationen des Projektes zu definieren. Desweiteren sollte es pro Bussystem eine Datei mit

den Informationen zu den einzelnen Busgeräten geben, die die projektübergreifenden Daten beinhalten.

Bei dem in dieser Arbeit entworfenen Softwaresystem werden die Projektdaten allerdings in zwei XML Dateien aufgeteilt. Eine Datei enthält ausschließlich Daten über die Gebäudestruktur, also Gebäude, Stockwerke, Räume und Schaltschränke. Die zweite Datei enthält alle installationsspezifischen Daten, d.h., Informationen zu den elektrischen Geräten in den Gebäuden. Um elektrische Geräte im Gebäude zuordnen zu können, wird die Querverbindung durch eine Verlinkung über IDs herangezogen.

Die Aufspaltung der Daten erfolgt aufgrund der Anforderung CAD-Daten in das System einzubinden. Diese gehören eindeutig zu den Daten der Gebäudestruktur. Daher wird es möglich, bei einem neuen Projekt zuerst die Gebäudestrukturdatei anzulegen (z.B. mit einem Werkzeug eines Drittanbieters). Die Datei kann dann in das neue Softwaresystem zur Gebäudeautomation importiert. Im letzten Schritt kann die Planung der Installation erfolgen.

4.6.3 Gebäudestrukturdaten

Die Daten der Gebäudestruktur werden hierarchisch in einer XML Datei gespeichert. Zu jedem Projekt gehört genau eine Gebäudestrukturdatei.

Abbildung 4.2 zeigt den Aufbau der Gebäudestrukturdaten. Jeder Knoten der Datenhierarchie (project, building, floor, room und junctionbox) ist ein oben beschriebener Standard-Datenknoten. In der Arbeit wird bisher nur Gebrauch von dem Tag für die Speicherung des Namens gemacht. In zukünftigen Weiterentwicklungen sollten in der Strukturdatei aber auch semantische Informationen über das Gebäude gespeichert werden (CAD-Daten). Denkbar sind hier z.B. Abmessungen und Lage der Räume, aber auch Kapazitäten der Verteilerkästen.

In der Strukturdatei wird außer der Gebäudestruktur noch der Verzeichnisname des Projektes, sowie das beim Anlegen des Projektes festgelegte (und bevorzugte) Bussystem gespeichert. Weitere projektbezogene Daten könnten zukünftig an dieser Stelle abgelegt werden.

Die Verbindung zu den Geräten der Elektroinstallation geschieht über Querverweise der eindeutigen IDs. Diese IDs zeigen auf Einträge in den Installationsdaten. Auf diese Weise können die elektrischen Geräte den Räumen und Verteilerkästen zugeordnet werden.

4.6.4 Installationsdaten

Neben der Gebäudestrukturdatei gibt es bei jedem Projekt auch noch eine Datei mit den Installationsdaten der Elektrogeräte. Beide Dateien bilden

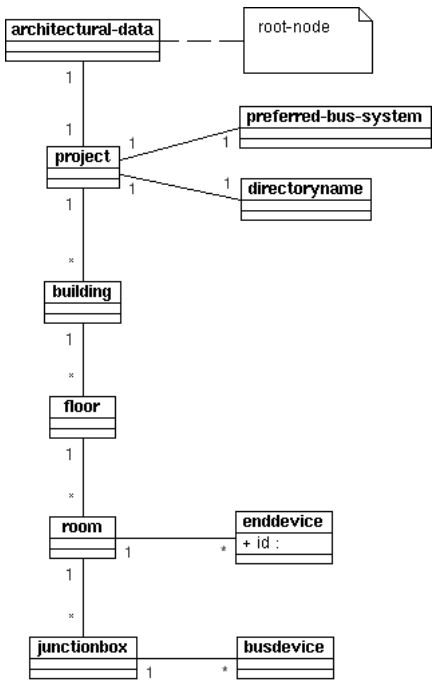

Abbildung 4.2: XML Gebäudestruktur

zusammen die komplette Projektbeschreibung und liegen in einem gemeinsamen Verzeichnis, das den Namen des Projektes trägt.

Abbildung 4.3 zeigt den Aufbau der XML Datei zur Aufnahme der installationsspezifischen Daten. Wie man sieht, wird zwischen aktiven und passiven Komponenten unterschieden.

Weiter ist ersichtlich, daß die Sensoren, Aktoren und passiven Komponenten eine Reihe von Eigenschaften (*properties*) zugewiesen bekommen. Diese enthalten einen Namen und einen Wert. Auf diese Weise lassen sich Geräten der Installation Daten zuweisen, ohne daß immer neue XML-Tags definiert werden müssen.

Über die *connection*-Tags können Verbindungen an Hand der Knoten IDs festgelegt werden. Auf diese Weise kann ein Ausgang eines Aktors mit einer passiven Komponente (z.B. einer Lampe) verlinkt werden. Je nach Typ der Geräte besteht dann im Gebäude eine elektrische oder logische (über den Bus) Verbindung.

Auf die Funktionsgruppen (*function groups*) wird weiter unten detailliert eingegangen.

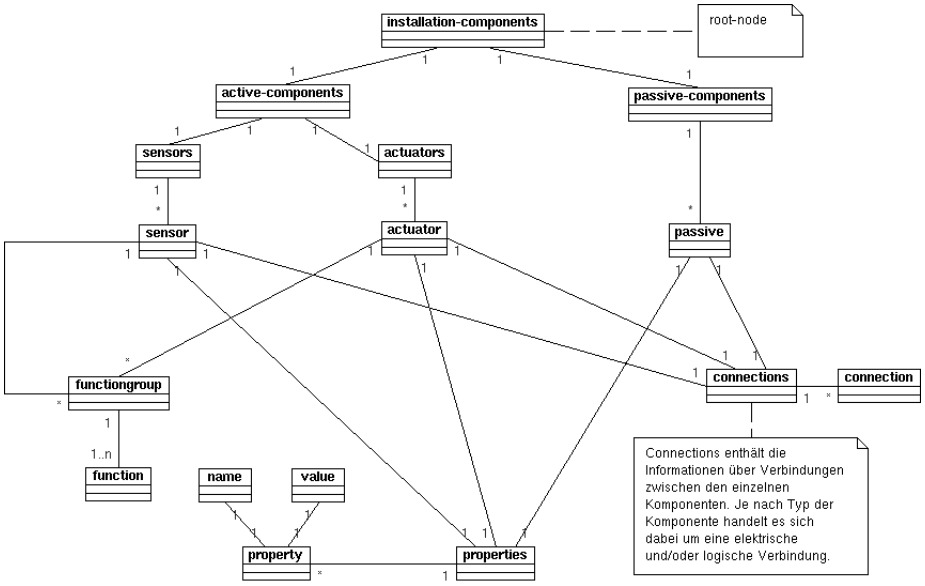

Abbildung 4.3: XML Installationsstruktur

Standardisierte Geräteeigenschaften

Den aktiven und passiven Komponenten der Installation werden diverse Eigenschaften zugewiesen. Die passiven Komponenten verwenden:

bussystem. Der Wert dieses Tags bezeichnet das vom Aktor der Komponenten verwendete Bussystem. Da bisher nur der EIB unterstützt wird, tritt bisher hier auch nur EIB auf.

actor-installation-location. Dieser Wert bezeichnet den Installationsort des Aktors der passiven Komponente. Zulässige Werte sind REG (Reiheneinbaugerät), UP (Unterputz-Gerät), CI (Kanaleinbau) und device (Aktor im Gerät).

type. Der Tag bezeichnet den numerisch kodierten Typ der Komponente. Zulässige Werte sind Nummern, die folgende Bedeutung haben: Schaltbare Lampe (1), dimmbare Lampe (2), Ventil (3), Jalousie (4) und Sensor (100). Sensor bezeichnet in diesem Zusammenhang keinen EIB Sensor sondern einen Taster, der eine Sensorfunktion erfüllt.

Die Eigenschaften der aktiven Komponenten sind:

device-name. Der Tag enthält den Gerätenamen des Busgerätes, also z.B. AT/S4.16.1 4f-Schaltaktor,16A,REG.

device-id. Die device-id ist die ID des Gerätes in der Busgeräte Datei. Über diese ID werden die für die Programmierung nötigen Daten aus der Busgeräte Datei gesucht.

device-state. Dieser Tag speichert den aktuellen Programmierungszustand des Gerätes. Mögliche Werte sind unprogrammed, addressed und ready.

manufacturer. Der Wert enthält den Namen des Geräteherstellers.

bussystem. Bussystem enthält analog zum Eintrag der passiven Komponenten das Bussystem des Gerätes.

installation-location. Der Tag bezeichnet den Installationsort der aktiven Komponenten analog zum Eintrag *actor-installation-location* der passiven Komponenten. Es ist durchaus denkbar, daß eine aktive Komponente ein Reiheneinbaugerät ist, die passive Komponente aber unter Putz eingebaut ist (z.B. Binäreingang mit passiven Tastsensoren).

eib-physical-address. Dieser Wert enthält die physikalische EIB Geräteadresse. Dieser Eintrag existiert nur, wenn es sich um ein EIB Gerät handelt.

In zukünftigen Versionen der Software wird es sicherlich noch eine Reihe weitere Einträge geben, die z.B. dann hinzukommen, wenn weitere Bussysteme unterstützt werden.

4.7 Anbindung der Bussysteme

Eine der Hauptanforderungen an das neue Softwaresystem stellt sicherlich die hardwaremäßige Anbindung verschiedener Bussysteme dar. Da die Umsetzung der Planung und Projektierung, sowie der Anbindung eines Bussystems eine der arbeitsintensivsten Tätigkeiten bei der Umsetzung des Systems ist, beschränkt sich die Arbeit auf die Unterstützung eines Bussystems. Das Softwaresystem wird jedoch so entworfen und umgesetzt, daß Schnittstellen in den Systemdaten und der Benutzerführung so gestaltet werden, daß weitere Bussysteme mit relativ geringem Aufwand in Zukunft unterstützt werden können.

4.7.1 Unterstützung des EIB

Als erstes Bussystem, das durch das neue Softwaresystem unterstützt wird, wird der EIB ausgewählt. Der EIB ist das in Europa verbreitetste System.

Durch seine Unterstützung können daher schon in der ersten Version der Software eine große Zahl an Geräten und Installationen geplant und in Betrieb genommen werden.

Die elektrische Anbindung des EIB geschieht über ein EIB-Ethernet Gateway [Alt02]. Die kommerzielle EIB Software ETS benutzt hingegen eine serielle Schnittstelle des PC. Die Schnittstelle zum EIB wird daher so gestaltet, daß es möglich wird verschiedene Anbindungsmöglichkeitem (z.B. USB, RS-232) zu verwenden.

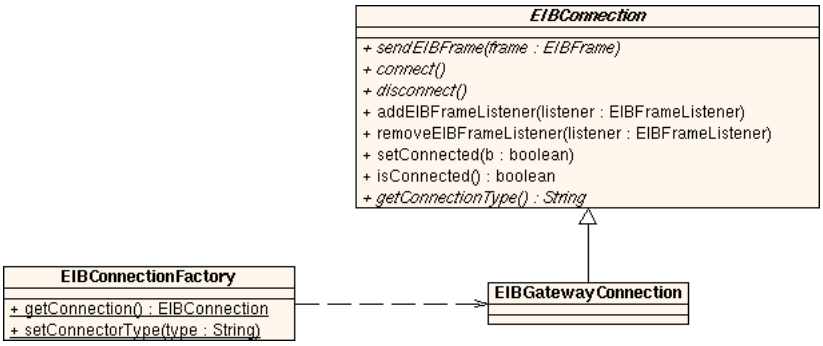

Abbildung 4.4: Anbindung des EIB durch die Klasse EIBConnection

Die softwaremäßige Anbindung an den EIB geschieht durch einen Ereignisorientierten Ansatz. Die EIB Daten werden, wie im Java Event Modell üblich, mit Hilfe von sogenannten Listenern (Methode `addEIBFrameListener`) an alle registrierten Komponenten verteilt. Zum Senden eines EIB Rahmens wird das Datenpaket einer speziellen Methode (`sendEIBframe`) übergeben.

Alle Klassen, die Logik enthalten, um mit dem EIB zu kommunizieren, werden von der abstrakten Klasse `EIBConnection` abgeleitet. Diese Klasse enthält Methoden zum Aufbau einer EIB Verbindung nach dem oben beschriebenen Muster. Eine Softwarekomponente, die mit dem EIB kommuniziert, benutzt ein Objekt der Klasse `EIBConnection` und implementiert die `EIBFrameListener` Schnittstelle. Die Softwarekomponenten erhalten durch die Object-Factory `EIBConnectionFactory` ein Objekt einer Realisierung der abstrakten Klasse `EIBConnection`, abhängig davon über welche Hardware die Verbindung zum EIB hergestellt wird. Durch das hier benutzte Entwurfsmuster ist es für die mit dem EIB kommunizierenden Komponenten unerheblich, auf welche Weise der EIB elektrisch angebunden wurde.

Die Zuordnung zur konkreten Realisierung der Busanbindung geschieht über eine Reihe von String-Bezeichnern. Das Softwaresystem gibt durch die Factory standardmäßig ein `EIBGatewayConnection`-Objekt zurück, das die

Anbindung durch das EIB-Ethernet-Gateway ermöglicht. In zukünftigen Erweiterungen der Software kann die Factory daher so gestaltet werden, daß sie weitere von `EIBConnection` abgeleitete und realisierte Objekte retourniert. Diese Objekte stellen dann z.B. über USB oder eine serielle Schnittstelle die Verbindung mit dem EIB her.

Wird durch die Klasse `EIBConnectionFactory` ein Objekt mit unbekanntem Typ angefordert, so wird eine `EIBConnectionNotAvailableException` ausgelöst.

Um eine Verbindung mit dem EIB aufzubauen, muß nach dem erfolgreichen Erzeugen des Connection-Objektes die Methode `connect()` aufgerufen werden. Ist keine Verbindung möglich, wird eine `EIBConnectionNotPossible-Exception` ausgelöst. Abbildung 4.4 verdeutlicht die Zusammenarbeit der an der EIB Kommunikation beteiligten Klassen.

4.8 Einbindung der EIB Gerätedaten

Zur Planung und Inbetriebnahme einer Gebäudeautomation mit Geräten des EIB braucht die dafür eingesetzte Software Informationen über die zur Verfügung stehenden EIB Geräte. In diesen Informationen müssen die Beschreibung und alle zur Programmierung notwendigen Daten des EIB Gerätes enthalten sein.

Für die ETS werden von den Herstellern der EIB Geräte sogenannte Produktdaten zur Verfügung gestellt. Diese Produktdaten können in die ETS importiert werden. Danach stehen die Geräte für die Planung zur Verfügung.

Die Gerätedaten der Hersteller sind in einem im ZIP-Format gespeicherten Archiv enthalten. Leider ist das Archiv durch ein Passwort vor normalem (unbefugten) Zugriff geschützt. Während des Einlesevorganges in die ETS wird das Archiv jedoch temporär auf die Festplatte in den Installationsordner der ETS entpackt. Die mit der Dateiendung `VD_` versehene Datei läßt sich in diesem Stadium kopieren und damit sichern. Dieser kleine Trick funktioniert übrigens auch mit der ETS Demo Version, die kostenlos aus dem Internet geladen werden kann.

Die Gerätedaten liegen in einem ASCII Datenformat vor, das sich besonders einfach für den Import in eine Datenbank eignet. Da das hier realisierte Softwaresystem bei der Planung anders vorgeht als bestehende Systeme müssen allerdings noch in den Gerätedaten nicht enthaltene Informationen erzeugt werden. Dies betrifft vor allem die Funktionsart des Gerätes, damit das Softwaresystem automatisch passende Geräte zu einer passiven Installationskomponente (Lampe, Ventil, ...) zuordnen kann.

4.8.1 Umwandlung in XML

Um die Gerätedaten besser handhaben zu können, empfiehlt sich die Umwandlung der ASCII Daten in ein XML Format. Die ASCII Datei liegt in einem Format aus beschreibenden Köpfen mit anschließender Folge der Daten in der Beschreibungsreihenfolge vor.

```
T 3 manufacturer
C1 T3 1 4 N MANUFACTURER_ID
C2 T3 3 50 Y MANUFACTURER_NAME
C3 T3 1 4 Y ADDRESS_ID
R 1 T 3 manufacturer
1
Siemens

R 2 T 3 manufacturer
4
Albrecht Jung
```

```
<manufacturer>
  <MANUFACTURER_ID>1</MANUFACTURER_ID>
  <MANUFACTURER_NAME>Siemens</MANUFACTURER_NAME>
</manufacturer>
<manufacturer>
  <MANUFACTURER_ID>4</MANUFACTURER_ID>
  <MANUFACTURER_NAME>Albrecht Jung</MANUFACTURER_NAME>
</manufacturer>
```

Abbildung 4.5: Auszug aus der ASCII- und der nach der Umwandlung entstandenen XML-Datei

Eine Zeile enthält damit genau ein Datum. Sind Daten leer gelassen, bleibt auch die entsprechende Zeile leer. Abbildung 4.5 zeigt einen Auszug aus der `VD_`-Datei und der XML Datei nach der Umwandlung.

Das XML Format erhöht die Lesbarkeit erheblich. Die Umwandlung ermöglicht zudem, leere Zeilen nicht in die XML Datei zu übernehmen.

Das Format der XML Datei orientiert sich vollständig am Format der ETS Gerätedaten. Die Datenbezeichner werden als XML Tagnamen verwendet. Daher sind die übergeordneten Tags klein und die Daten umgebenden Tags groß geschrieben.

4.8.2 Relationen in den Daten

Da die Gerätedaten in der ETS in einer relationalen Datenbank gespeichert werden, enthalten sie eine Reihe von internen Verweisen. Die Querverbindung der Daten untereinander geschieht mit Hilfe von numerischen IDs.

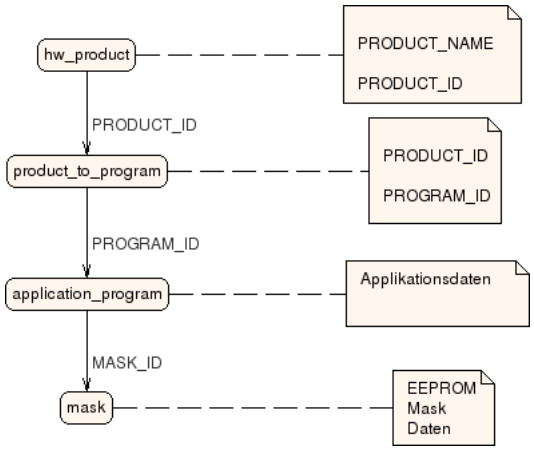

Abbildung 4.6: Relationen der benötigten Daten

Abbildung 4.6 zeigt die Relationen, die vom realisierten Softwaresystem benötigt werden, um die zur Programmierung der EIB Geräte nötigen Daten zu erhalten. Durch mehrfaches Einlesen der XML Dateien gelingt es, die jeweiligen IDs zu ermitteln und damit an die querverbundenen Daten zu kommen.

Die benötigten Gerätedaten können vom Datenadministrator ausgewählt und zur Datenbasis des Systems hinzugefügt werden. Die für das automatische Zuweisen von Geräten notwendigen Zusatzinformationen müssen danach noch manuell eingegeben werden.

4.9 Auswahl passender Busgeräte

Der hier gemachte Ansatz der automatischen Zuordnung passender Aktoren und Sensoren zu den passiven Komponenten eines Gebäudes zieht zusätzliche Information in der Datei der Busgeräte nach sich. Bisherige Systeme brauchen solche Zusatzinformationen nicht, da sie dem *Anwender* die richtige Zuordnung der Geräte überlassen. Daher ist Information über die an die Aktoren anschließbaren Geräte auch höchstens in Prosa- und nicht in maschinenlesbarer Form in den Gerätebeschreibungen enthalten.

4.9.1 Funktionsgruppen

Eine der Hauptaufgaben besteht daher darin zu überlegen, welche Zusatzinformationen man benötigt, um die gewünschte Funktionalität zu realisieren. Da sich die Arbeit mit dem EIB im besonderen beschäftigt, wird hieraus der Ansatz entwickelt.

Jedes EIB Gerät arbeitet mit sogenannten Kommunikationsobjekten. Diese werden bei der klassischen EIB Projektierung durch den Anwender mit bestimmten Gruppenadressen verbunden. Nach der Programmierung der Geräte löst ein Bustelegramm an die jeweilige Gruppenadresse die dem Kommunikationsobjekt zugeordnete Aktion aus.

Die Aktionen und Anzahl der Kommunikationsobjekte unterscheiden sich meist mit dem Wechsel der Applikation des EIB Gerätes. Kommunikationsobjekte können Daten empfangen oder senden. Aktionen von Kommunikationsobjekten sind z.B. Schalten, Dimmen, Status melden, Wert empfangen, Wert senden usw. Ein einfacher Schaltaktor benötigt für seine Aufgabe oft nur ein Kommunikationsobjekt (Schalten), während ein Dimmaktor schon mehrere verwendet (Schalten, Dimmen, Wert). Dies bedeutet, daß mehrere Kommunikationsobjekte einer Gruppe bzw. einer Funktion angehören können. Auch wenn es sich z.B. um einen mehrfach Aktor/Sensor handelt, sind jedem Ausgang/Eingang bestimmte Kommunikationsobjekte zugeordnet, die zusammen gehören.

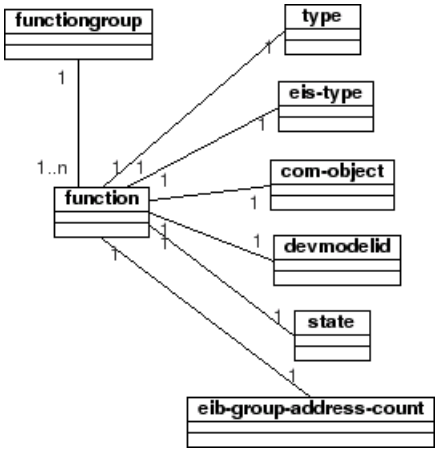

Abbildung 4.7: Aufbau der Funktionsgruppen

Um diese nur in textueller Form zugänglichen Informationen für einen Automatismus verarbeitbar zu machen, werden sogenannte Funktionsgruppen definiert.

Abbildung 4.7 zeigt den Aufbau der XML Funktionsgruppenstruktur und der in ihr enthaltenen Funktionen. Das Tag *functiongroup* und *function* gehören zur Gruppe der oben beschriebenen Standard-Datenknoten, enthalten also einen Tag zur Aufnahme von einem Namen und Kommentar.

Eine Funktionsgruppe faßt mehrere zusammenhängende Funktionen zusammen. Eine Funktion besteht im Falle des EIB aus der Nummer des Kommunikationsobjektes, dessen EIS Typ [EIB99, Volume 3/Part 7] , einem Verweis auf die ETS Gerätedatenbank und einer eindeutigen Nummer, die zu einer Gruppenadresse aufgelöst werden kann.

Die genannten Informationen können aus den ETS Produktdaten gewonnen werden.

Der noch nicht genannte Tag *type* nimmt die Funktion des Kommunikationsobjektes auf und muß durch den Datenadministrator festgelegt werden. Das *state*-Tag enthält den Status der Funktion und kann die Zustände *unused* oder *addressed* aufnehmen. Dadurch ist die Software in der Lage zu entscheiden, welche Funktionen eines (mehrfach-)Aktors noch zur Verfügung stehen.

Mit Hilfe der Funktionsgruppen kann die Software daher für die elektrischen Geräte des Gebäudes die passenden Aktoren und Sensoren suchen. So hätte ein 4-fach Dimmaktor dann beispielsweise 4 Funktionsgruppen (pro Ausgang eine) mit jeweils 3 Funktionen (Schalten, Dimmen, Wert). Ein 4-fach Schaltaktor hätte auch 4 Funktionsgruppen aber mit nur jeweils einer Funktion (Schalten). Folgender Auszug aus der EIB Gerätedatei des Softwaresystems verdeutlicht die Zusammenhänge:

```
<eibdevice id="eibdevice-2">
    <name>AT/S4.16.1 4f-Schaltaktor,16A,REG</name>
    <comment/>
    <functions>
      <functiongroup id="functiongroup-5">
        <name/>
        <comment/>
        <function id="function-5">
          <name>Ausgang A</name>
          <comment/>
          <type>switching</type>
          <com-object>0</com-object>
          <function-name>Schalten/function-name>
          <eis-type>0</eis-type>
        </function>
      </functiongroup>
```

```
  <functiongroup id="functiongroup-6">
    <name/>
    <comment/>
    <function id="function-6">
      <name>Ausgang B</name>
      <comment/>
      <type>switching</type>
      <com-object>1</com-object>
      <function-name>Schalten</function-name>
      <eis-type>0</eis-type>
    </function>
  </functiongroup>
  <functiongroup id="functiongroup-7">
    <name/>
    <comment/>
    <function id="function-7">
      <name>Ausgang C</name>
      <comment/>
      <type>switching</type>
      <com-object>2</com-object>
      <function-name>Schalten</function-name>
      <eis-type>0</eis-type>
    </function>
  </functiongroup>
  <functiongroup id="functiongroup-8">
    <name/>
    <comment/>
    <function id="function-8">
      <name>Ausgang D</name>
      <comment/>
      <type>switching</type>
      <com-object>3</com-object>
      <function-name>Schalten</function-name>
      <eis-type>0</eis-type>
    </function>
  </functiongroup>
</functions>
...
```

4.10 Programmierung der EIB Geräte

Neben dem Entwurf der Datenstrukturen und dem Zusammenstellen der benötigten EIB Gerätedaten gehört die Umsetzung des Programmierens der EIB Geräte zu den weiteren wichtigen Aufgaben der Arbeit.

Im Gegensatz zum Programmieren der physikalischen Adresse eines EIB Gerätes, das sowohl in [EIB99] als auch in [DKS00] beschrieben wird, sind über das für die Inbetriebnahme der Installation dringend erforderliche Programmieren der Applikation bislang nur sehr spärlich Informationen veröffentlicht worden.

Aus diesem Grund wurden eine Reihe von Programmiervorgängen, die mit der ETS durchgeführt wurden mitgeschnitten und analysiert (*reverse engineering*).

4.10.1 Schreiben der Applikationsdaten

Die Programmierung der EIB Applikationen geschieht durch Übertragen der binären Applikationsdaten direkt in den Speicher des Buskopplers. Hierzu wird vorweg eine Verbindung mit dem Buskoppler hergestellt. Danach wird durch Schreiben der Daten der Programmiervorgang durchgeführt und durch anschließendes Wiederauslesen überprüft, ob die Daten im Gerät richtig hinterlegt wurden.

Abbildung 4.8 zeigt die Schritte zur Programmierung einer Applikation. Die Überprüfung der Schreibvorgänge ist nicht explizit aufgeführt. Vor der Übertragung der Applikation wird die Maskenversion und der Herstellercode ausgelesen. Wenn die gewählte Applikation zu den Werten paßt, wird die Applikation geschrieben, ansonsten wird der Vorgang abgebrochen.

Die Applikationsdaten werden in einen speziellen EEPROM Bereich der Buskoppler geschrieben, der für den Benutzer für Schreiboperationen freigegeben ist (*User-ROM*). Die ROM-Adressen unterscheiden sich dabei je nach Maskenversion des eingesetzten Buskopplers.

Nach dem Übertragen der Applikationsdaten wird der Benutzer-RAM Bereich durch Schreiben von Nullen gelöscht und der Buskoppler zurückgesetzt. Nach einem erfolgreichen Neustart arbeitet der Buskoppler mit der neu programmierten Software.

4.10.2 Bestandteile einer EIB Applikation

Eine EIB Applikation besteht aus folgenden Daten: Systemparameter, Adreßtabelle, Assoziationstabelle, Kommunikationsobjekttabelle, Applikationsparameter und das Applikationsprogramm (inklusive Start- und Stop-Routinen).

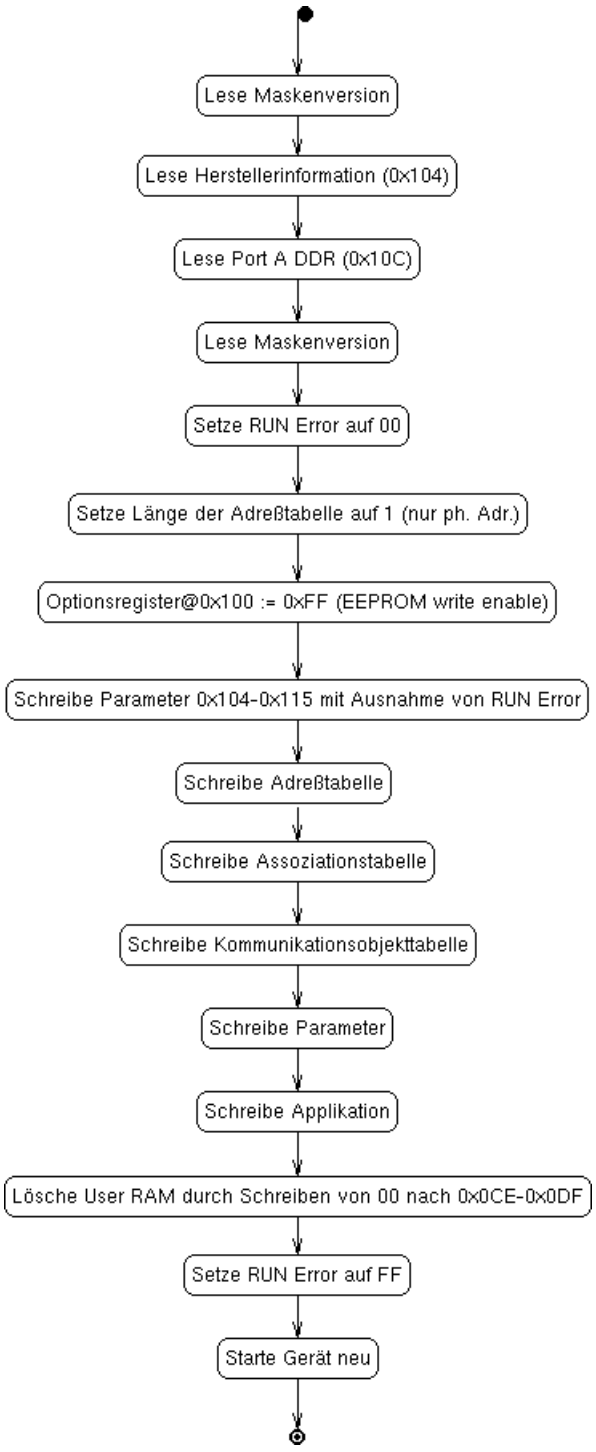

Abbildung 4.8: Programmieren einer EIB Applikation

Die einzelnen Komponenten werden nun kurz genauer erklärt. Auch die Adressen dieser ROM-Bereiche unterscheiden sich je nach Hersteller. Die Adreßangaben können jedoch aus der vom Hersteller für die ETS zur Verfügung gestellten Gerätedatei herausgefunden werden.

Systemparameter

Die Systemparameter stehen am Beginn des Applikationsspeichers und enthalten Informationen über den Buskoppler, das angeschlossene Gerät sowie über die Applikation. Unter anderem sind hier Informationen über den Gerätehersteller, den Gerätetyp, sowie Zeiger auf die Speicherbereiche der Tabellen und die Start- und Stop-Routine der Applikation enthalten. Durch die Benutzung von Zeigern können die Tabellen von Applikation zu Applikation unterschiedliche Längen besitzen.

Adreßtabelle

Die Adreßtabelle enthält die Geräteadresse und die Gruppenadressen des Gerätes. Die physikalische Adresse wird jedoch nicht durch direkten Speicherzugriff programmiert, sondern erfolgt vorher durch ein Systemservice zum Schreiben der physikalischen Adresse (`A_SET_PHYSICAL_ADDRESS`-Service). Da das Programmieren der Applikation verbindungsorientiert, unter Benutzung der physikalischen Adresse geschieht, bleibt diese unangetastet und die Länge der Adreßtabelle wird zunächst auf 1 gesetzt (nur physikalische Adresse).

Die Gruppenadressen sind sortiert in der Tabelle eingetragen (kleinste zuerst). Dies ist notwendig, da die Systemsoftware als Optimierung beim Durchforsten der Tabelle die Suche vorzeitig abbricht, wenn zwischen der letzten (kleineren) und der aktuellen (größeren) die gesuchte Adresse nicht zu finden war.

Unbenutzte Tabellenplätze werden mit dem Wert `0xFF` beschrieben.

Assoziationstabelle

Die Assoziationstabelle stellt Querverbindungen zwischen den Gruppenadressen und den zugehörigen Kommunikationsobjekten her. Einer ein Byte großen Längenangabe, die die Anzahl der Assoziationen angibt, folgen die Assoziationen bestehend aus 2 Byte. Das erste Byte enthält die Nummer der Gruppenadresse. Das zweite Byte die Nummer des Kommunikationsobjektes. Die erste Gruppenadresse hat die Nummer 1. Unbenutzte Assoziationen enthalten statt der Adreßnummer im ersten Byte den Wert `0xFE`.

Kommunikationsobjekttabelle

Die Kommunikationsobjekttabelle enthält zunächst im ersten Byte die Anzahl der Kommunikationsobjekte, gefolgt von 3 Byte langen Einträgen für die jeweiligen verwendeten Kommunikationsobjekte.

Eine ausführliche Beschreibung der Tabellen findet sich in [DKS00, S.149ff., S.211].

Applikation und Applikationsparameter

Die Applikation eines EIB Buskoppler besteht aus ausführbarem Binärcode für den jeweils verwendeten Mikrocontroller. Diese Daten werden von den Geräteherstellern bereits in den Produktdaten in binärer Form (ASCII Hex-Werte) geliefert. Zu einer Applikation gehören auch noch Start- und Stop-Routinen, die beim Starten bzw. Stoppen (z.B. Stromausfall) des Buskopplers notwendige Operationen ausführen (Datensicherung).

Jeder Buskoppler enthält auch noch eine Reihe von Applikationsparametern, die das Verhalten der Applikation und damit die Funktionalität des Gerätes beeinflussen. In den von den Herstellern vorgegebenen Daten sind diese Parameter jeweils mit den Standardwerten belegt. Die Modifizierung der Parameter ist Sache der Konfigurationssoftware.

Kapitel 5

Implementierung

Dieses Kapitel beschreibt Details der Implementierung des Softwaresystems aufgrund des gemachten Lösungsansatzes.

5.1 Systempaketstruktur

Durch die Verwendung von Java als Implementierungssprache wird der Quellcode auch in den bei Java üblichen Paketen organisiert. Bei der hier gemachten Implementierung werden jedoch zunächst keine Domainnamen für die Benennung der Pakete benutzt, da der endgültige Verbleib des Quellcodes noch nicht auf eine bestimmte Domain festgelegt ist.

5.1.1 Systemname

Statt umgekehrten Domainnamen (z.B. at.ac.tuwien.auto) wird das Oberpaket mit dem Namen des Softwaresystems bezeichnet. Das System bekommt den Namen BASys (*Building Automation System Software*).

5.1.2 Unterpakete

Unter dem Basispaket `basys` befinden sich mehrere Unterpakete, die Quellcode für bestimmte Aufgaben enthalten:

`basys.client` Dieses Paket enthält den Quellcode für die Client-Applikation des Softwaresystems und besteht aus den Funktionen zum Aufbauen und Verwalten der graphischen Bedienoberfläche des Programmes.

`basys.server` Dieses Paket enthält die Serverkomponenten des Softwaresystems. Durch die Trennung von Client und Server Paketen wird das

Erstellen von Distributionen, die nur den Server- oder Client-Teil enthalten ermöglicht. Eine Klasse im Server-Paket sollte niemals Gebrauch von Klassen aus dem Client-Paket machen und umgekehrt.

`basys.datamodels` Dieses Paket enthält die Klassen, die den Zugriff auf die XML Daten ermöglichen (Installation, Gebäudestruktur, EIB Gerätedaten). Die Klassen dieses Paketes werden sowohl vom Server-, als auch vom Client-Teil des Systems verwendet.

`basys.eib` Dieses Paket enthält alles, was nötig ist, um auf EIB Geräte physikalisch und logisch zuzugreifen. Im Unterpaket `basys.eib.dataaccess` befindet sich der Quellcode, der für das Einlesen der ETS Gerätedaten zuständig ist. Die Klassen dieses Paketes werden sowohl vom Server-, als auch vom Clientteil benötigt.

Die nachfolgenden Abschnitte beschreiben einige Details der Pakete und des in ihnen befindlichen Quellcodes.

5.2 Dateistruktur des Systems

Das Softwaresystem BASys verwendet eine bestimmte Verzeichnisstruktur, um seine Dateien abzulegen. Zunächst befindet sich im Arbeitsverzeichnis des Benutzers (z.B. `/home/JonDoe/`) ein Verzeichnis mit Namen `basys`, das alle weiteren Daten aufnimmt. Zur besseren Strukturierung finden sich in `basys` zwei Unterverzeichnisse, nämlich `global-data` und `projects`. In `projects` werden alle projektrelevanten Daten abgespeichert. Jedes Projekt besteht aus zwei XML-Dateien: `installation.xml` mit den Elektroinstallationsdaten und `structure.xml` mit den strukturellen Gebäudedaten. Die beiden Projektdateien liegen jeweils in einem Verzeichnis, das den Namen des zugehörigen Projektes trägt.

In `global-data` befinden sich die Busgerätedaten und andere projektübergreifende Dateien. Die umgewandelten ETS Produktdaten liegen im Verzeichnis `basys/global-data/eib/devicedata`. Die EIB Datenbank von BASys befindet sich in `basys/global-data/eib/eib-devices.xml`.

Das Verzeichnis `global-data/help` enthält HTML Hilfedateien des Softwaresystems und die zugehörigen Graphikdateien.

In zukünftigen Versionen sollten die Bussystemdaten anderer Bussysteme auch in eigenen Verzeichnissen (z.B. `basys/global-data/LON` oder `basys/global-data/LCN`) untergebracht werden.

5.3 XML Systemdaten

Im Paket `basys.datamodels` befinden sich sämtliche Klassen, die den Zugriff auf XML Daten ermöglichen. Um XML Dateien mit der Struktur des in Abschnitt 4.6 beschriebenen Standard-Datenknotens zu erzeugen, gibt es die abstrakte Klasse `XMLDataModel`. Diese Klasse enthält Methoden, die das Anlegen und Modifizieren solcher Standard-Datenknoten unterstützen.

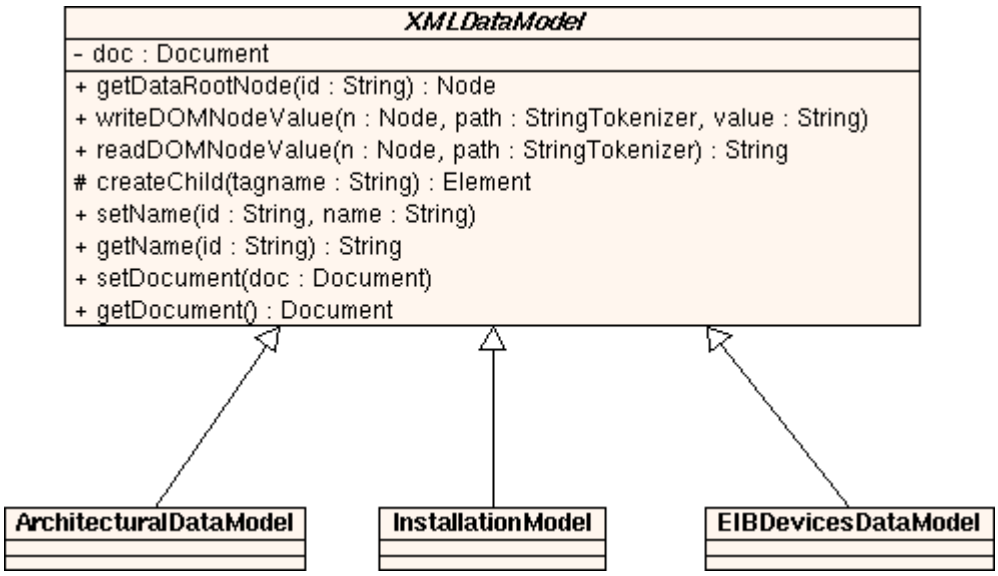

Abbildung 5.1: Die Klasse XMLDataModel und daraus abgeleitete konkrete Klassen

Das UML Diagramm in Abbildung 5.1 zeigt die Klasse `XMLDataModel` mit ihren wichtigsten Methoden. Die Speicherung der Daten basiert auf dem Document Object Model (DOM) [McL01, Seite 184ff.] zur Bearbeitung von XML Daten. Die Klasse `XMLDataModel` bietet Methoden zum Auslesen und Schreiben von XML Werten. Diese Methoden kapseln den direkten Zugriff mit Hilfe des DOM, so daß Anwender einfacher Daten bearbeiten können und sich um Interna des DOM nicht zu kümmern brauchen.

Die Methoden `writeDOMNodeValue()` und `readDOMNodeValue` benutzen einen `StringTokenizer`, um den Pfad im DOM Baum ausgehend von einem Knoten zu bestimmen. Sämtliche beim direkten Zugriff nötige Fallunterscheidungen werden innerhalb der Methoden gemacht. Insbesondere werden automatisch neue Knoten angelegt, wenn es sie auf dem spezifizierten Pfad bei einer Schreiboperation noch nicht gibt. Durch Verwenden der Methode `getDataRootNode(String id)` bekommt man den DOM Knoten des

der ID entsprechenden Standard-Datenknotens zurück. Dieser kann dann als Parameter den Schreibe- und Lesemethoden übergeben werden. Sämtliche Zugriffsoperationen lassen sich daher über die ID-Strings durchführen.

Die Methoden `setName()` und `getName()` setzen den Wert des name-Tag des Standard-Datenknotens. Die konkreten Klassen, die ein XML Datenmodell repräsentieren, bieten solche Methoden auch für andere Werte an. Der Anwender eines solchen Datenmodell Objektes benutzt nur diese Methoden und braucht damit nicht mehr direkt mit dem DOM zu hantieren, d.h., er braucht sich insbesondere nicht mit der Traversierung von DOM Knoten explizit zu befassen. Durch die Kapselung ist es sogar möglich, die Daten in einem anderen Format als XML zu speichern.

Im Softwaresystem gibt es drei Konkretisierungen der `XMLDataModel`-Klasse (`ArchitecturalDataModel`, `InstallationModel` und `EIBDevices-Data Model`). Diese repräsentieren die XML Dateien der Projekt- und EIB Gerätedaten. Die Kindklassen befinden sich jeweils in Unterpaketen des Paketes `basys.datamodels`.

5.4 Anbindung des EIB

Die Anbindung des EIB erfolgt durch die Umsetzung des in Abschnitt 4.7.1 vorgestellten Ansatzes. Das Konzept des Ereignis-orientierten Empfanges von EIB Rahmen wird beibehalten. Um den Vorgaben des Java-Event-Konzeptes vollständig zu entsprechen, wird jedoch nun ein von `EventObject` abgeleitetes `EIBFrameEvent` Objekt an die zugehörigen Listener geschickt. Das Senden der EIB Daten geschieht auch weiterhin durch Aufruf der Methode `sendEIBFrame(EIBFrame frame)`.

Die schon aus [Alt02] bekannte Klasse `EIBGatewayConnection` wird verbessert und nun Tochterklasse von `EIBConnection`. Sie wird so modifiziert, daß nur noch ein Objekt aus ihr erzeugt werden kann (*Singelton*), da sonst der UDP Socket mehrfach initialisiert werden könnte, was zu einer Exception führt.

Sämtliche Objekte von den von `EIBConnection` abgeleiteten Klassen werden im System durch die Factory-Klasse `EIBConnectionFactory` erzeugt. Damit wird die Erweiterbarkeit sichergestellt und das System kann dem Benutzer in unterschiedlichen Ausbaustufen die jeweils verfügbaren Verbindungsmöglichkeiten zum EIB anbieten. In der aktuellen Version liefert die Factory nur Objekte zurück, die eine Verbindung über das Ethernet-Gateway ermöglichen. Der hierbei benutzte String zur Identifikation lautet `Gateway TU-Darmstadt`.

5.5 Verarbeitung der EIB Gerätedaten

Die gesamte Verarbeitung der ETS Gerätedaten übernehmen Klassen im Paket `basys.eib.dataaccess`.

5.5.1 Konvertierung der ETS Daten nach XML

Die Konvertierung der ETS VD_ Herstellerdateien geschieht durch die Klasse `VDConverter`. Dem Konstruktor werden einfach zwei Dateinamen für die Eingabe- und Ausgabedatei übergeben und die Konvertierung beginnt.

5.5.2 Verarbeitung der konvertierten XML Daten

Die Weiterverarbeitung der konvertierten XML Daten übernimmt die Klasse `DataExtractor` im Paket `basys.eib.dataaccess`. Die Daten werden dabei durch den SAX-Parser [McL01, S. 49ff.] ausgelesen und bearbeitet. Die Benutzung von DOM scheidet aufgrund der Größe der XML Datei aus (mehrere 10 Megabyte). Durch aufeinanderfolgende Aufrufe der Methoden können die in Abschnitt 4.8 beschriebenen Querverbindungen benutzt und hergestellt werden.

Die ausgelesenen Daten können dann mit den nötigen Zusatzinformationen (Funktionsgruppen, Typ) versehen in der EIB Gerätedatei des Systems abgespeichert werden.

Abbildung 5.2 zeigt den Systemdialog zum Erstellen der EIB Gerätedatenbank. Der Dialog benutzt die oben beschriebene Klasse zum Auslesen und Anzeigen der Daten. In den Listen lassen sich die gewünschten Geräte und Applikationen auswählen. Nach dem Parsen der Daten erscheint die Auflistung der gefundenen Kommunikationsobjekte in der oberen Tabelle. Durch Auswahl von zusammenhängenden Kommunikationsobjekten (Funktionen) lassen sich mit der entsprechenden Schaltfläche Funktionsgruppen definieren. Der Zusammenhang zwischen den Kommunikationsobjekten muß dafür durch eine menschliche Person (Datenadministrator) aufgrund der angezeigten Prosabeschreibung festgestellt werden.

Die untere Tabelle zeigt die bereits in der Gerätedatei befindlichen Geräte an. Die EIB Gerätedatenbank wird beim Start des Systems automatisch geladen und beim Beenden gespeichert. Da das Gerätedatenmodell von der Klasse `XMLDataModel` erbt, stehen neu erstellte Daten aufgrund des im RAM persistenten DOM Datenspeichers dem Systembenutzer sofort nach dem Hinzufügen zur Verfügung.

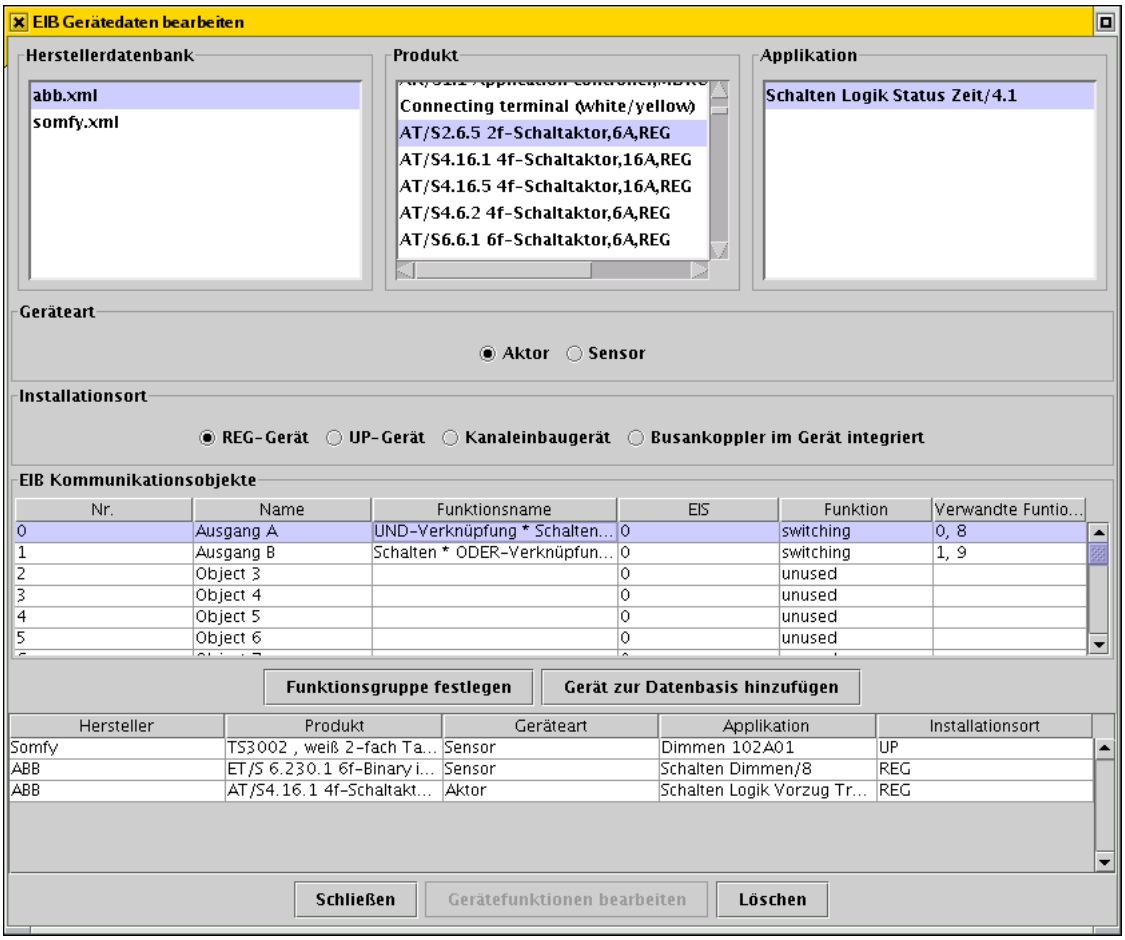

Abbildung 5.2: Dialog zum Erstellen der Gerätedatenbank

5.6 Gerätesuche

Das Suchen und Zuordnen von passenden Busgeräten erfolgt durch die Benutzung der Funktionsgruppeneinträge. Nach Auswahl des gewünschten Installationsgerätes (Lampe, dimmbare Lampe, Jalousie, Ventil) werden dazu passende Funktionsgruppen in der EIB Gerätedatei des Systems und in der Installationsdatei gesucht. Für den Fall, das passende Geräte mit freien Ausgängen vorhanden sind, wird der Benutzer gefragt, ob er diese verwenden möchte. Ansonsten bietet das System eine Liste passender Busgeräte an.

Im Falle mehrerer Ausgänge wird der Benutzer noch gefragt, welchen Ausgang er verwenden möchte.

Die Logik zum Auffinden der EIB Geräte befindet sich in der Klasse `EIBDeviceConfigurator`. Hierin sind vor allem Suchmethoden untergebracht, die die Kriterien der Busgeräte und Funktionsgruppen bei der Suche berücksichtigen. Die Klasse benutzt intensiv Methoden des Installationsmodells (Klasse `InstallationModel`), um ihre Aufgaben durchzuführen. Sowohl die Suche der Aktoren, als auch die der Sensoren erfolgt durch sie.

5.7 EIB Applikationen

Um die EIB Geräte programmieren zu können, müssen aufgrund der durch die Planung erzeugten Daten passende EIB Applikationen generiert werden, die dann beim Programmieren in die Gerätespeicher geschrieben werden.

Diese Aufgabe obliegt der Klasse `EIBProgramConfigurator`. Sie verwendet die Planungs- und Gerätedaten, um die Applikationen zu erstellen. Dabei werden passende Adreß-, Assoziations- und Kommunikationsobjekttabellen aufgebaut und mit den Programmdaten zur kompletten Applikation vervollständigt.

In der Klasse `EIBProgramConfigurator` befindet sich außerdem die Methode `getMemoryString()`, die die erstellte Applikation für Debugging-Zwecke als hexadezimale Werte mit den passenden Speicheradressen ausgibt.

5.8 Programmierung der EIB Geräte

Um EIB Geräte so zu programmieren, daß sie danach betriebsbereit sind, muß die physikalische Adresse und die Applikation durch das Softwaresystem einstellbar sein. Zum Programmieren einer Applikation muß außerdem eine verbindungsorientierte Kommunikation zwischen dem Softwaresystem und dem EIB Gerät aufgebaut werden.

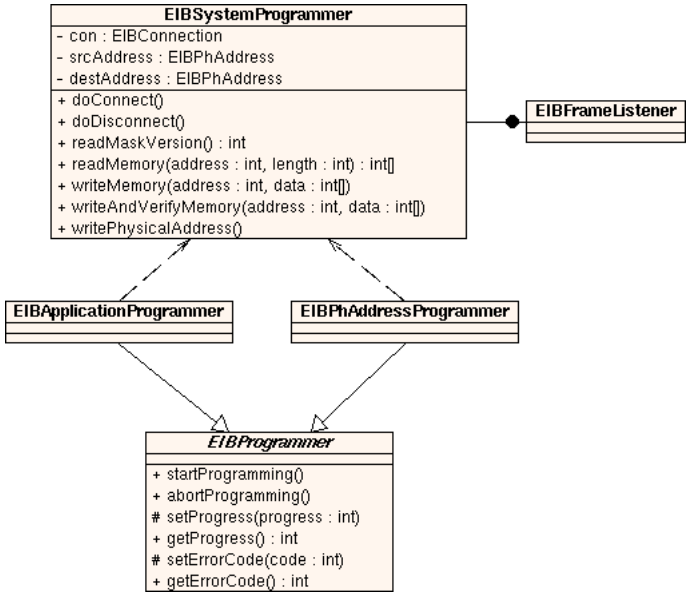

Abbildung 5.3: Softwarekomponenten zur Programmierung von EIB Geräten

Abbildung 5.3 zeigt die Komponenten zur Programmierung von EIB Geräten. Die Klasse `EIBSystemProgrammer` implementiert die Systemfunktionen des EIB. Anwender der Klasse brauchen sich nicht mehr um Details der EIB Transportschicht zu kümmern, sondern können durch einfaches Aufrufen der Methoden `doConnect()` und `doDisconnect()` eine verbindungsorientierte Kommunikation herstellen. Die Klasse `EIBSystemProgrammer` stellt damit die Applikationsschicht im EIB Protokollstapel dar.

Die einzelnen Methoden können eine Reihe von Exceptions auslösen (z.B. Feststellen eines Verbindungsabbruches). Der Anwender der Klasse wird somit gezwungen, die bei der EIB Kommunikation auftretenden Fehlerursachen auch zu behandeln. Die zur EIB Kommunikation gehörenden Exception-Klassen sind im Paket `basys.eib.exceptions` untergebracht.

5.8.1 Eigene Threads zur Programmierung

Benutzer der `EIBSystemProgrammer` Klasse sind die von `EIBProgrammer` abgeleiteten Klassen `EIBApplicationProgrammer` und `EIBPhAddressProgrammer`. Die beiden Klassen enthalten jeweils einen separaten, internen Thread, der die Programmieraufgabe ausführt und durch Aufruf der Methode `startProgramming()` gestartet wird. Die Kommunikation mit anderen Threads geschieht über gemeinsame Variablen. Während der Programmierung wird

beispielsweise ein Fortschrittszähler im Wertebereich von 0 bis 100 entsprechend gesetzt.

Die Aufspaltung in eigene Threads ist nötig, damit z.B. eine graphische Oberfläche einen Fortschrittsbalken aktualisieren und der Programmiervorgang währenddessen weiterlaufen kann.

Erfolg oder ein Fehler der Programmierung wird durch Fehlercodes signalisiert, die in `EIBProgrammer` definiert sind. Die Benutzerschnittstelle kann damit eine entsprechende Meldung ausgeben. Die Logik ist so gestaltet, daß im Fehlerfall der Programmiervorgang automatisch gestoppt wird.

5.8.2 Modifikation des Ethernet-Gateway

Da die Dienste der Systemprogrammierung beim EIB verbindungsorientiert mit der physikalischen Adresse als SAP arbeiten, muß das zur Herstellung der Verbindung benutzte Ethernet-Gateway leicht modifiziert werden. Das in [Alt02] realisierte Gateway bestätigt nämlich keinerlei Telegramme, sondern verhält sich wie ein rein passiver Empfänger. Damit eine verbindungsorientierte Kommunikation möglich ist, wird die Firmware so modifiziert, daß alle an eine bestimmte Adresse gerichteten Telegramme durch den für den Empfang zuständigen TP-UART-Chip bestätigt werden. Hierzu ist nur eine geringfügige Modifikation an der für den Telegrammempfang zuständigen Interrupt-Routine nötig.

5.9 Projektglobale Klassen

Im Paket `basys` befinden sich eine Reihe von Klassen, deren Funktionen sowohl vom Server-, als auch vom Client-Teil des Systems verwendet werden.

5.9.1 Laden und Speichern von XML Daten

Die Klassen `XMLLoader` und `XMLSaver` beinhalten Programmcode zum Laden und Speichern einer XML Datei in ein DOM Objekt. Sie bilden damit die Schnittstellen zwischen den im Dateisystem und den im Arbeitsspeicher befindlichen XML Daten.

5.9.2 Die Klasse `basys.Util`

Die `Util` Klasse beinhaltet statische Methoden für oft benötigte Funktionalitäten. Darunter befinden sich unter anderem Methoden zum Laden von Bilddaten (Icons, Graphiken). Bei Java ändert sich der Pfad der Bilddaten,

je nachdem ob das Programm aus einem JAR-Archiv heraus oder direkt gestartet wird. Die entsprechende Umsetzung der Pfade übernehmen daher die Methoden `public static ImageIcon getIcon(String filename)` und `public static Image getImage(String filename)`. Der Wurzelpfad für Bilddateien ist im System fest auf `basys/images/` festgelegt. Soll ein Bild aus diesem Verzeichnis geladen werden, genügt daher die Angabe des Dateinamens (z.B. *open.gif*).

Bilddaten werden hauptsächlich für Icons der Menüleisten und Graphiken in Dialogfeldern gebraucht.

5.9.3 Internationalisierung

Die Anforderung das Programm mit mehrsprachiger Benutzerführung auszustatten wird durch die Klasse `LocaleResourceBundle` erfüllt. Sie benutzt das von Java unterstützte Konzept der Internationalisierung. Objekte dieser Klasse lassen sich nicht direkt erzeugen, sondern werden durch die Methode `public static ResourceBundle getLocale()` zurückgegeben.

Jede Klasse des Systems, die Benutzerausgaben macht, muß ein entsprechendes Objekt und einen Referenznamen verwenden um die lokalen Texte anzufordern (z.B. `locale.getString("l.actuator")`). Im System müssen dann mit passenden Einträgen gefüllte *Ressource*-Daten/Dateien (z.B. `Locale_de_DE.properties`) vorhanden sein.

5.10 Die Java Client Applikation

Den für den Benutzer sichtbaren Programmteil des BASys Systems bildet eine Java-Client Applikation (*Java standalone application*). Die Applikation integriert die Funktionen zum Planen, zur Inbetriebnahme, zur Datenverwaltung, sowie zur Fehlersuche für eine Gebäudeautomation.

Abbildung 5.4 zeigt die Applikation nach dem Start. Sämtliche graphischen Komponenten sind modular, nach dem Vorbild der Java Beans Technologie aufgebaut. Die Komponenten zur Benutzerinteraktion verwenden somit Event-basierte Kommunikation und werden von schon vorhandenen Swing Klassen abgeleitet. In der momentanen Implementierung sind die Klassen jedoch nicht in einzelne Archive gepackt, wie es *echte* Java Beans sein sollten. Die Weiterentwicklung zu reinen Java Beans ist aufgrund des Programmieransatzes jedoch möglich.

Alle Programmteile, die an der graphischen Benutzeroberfläche mitwirken, sind im Paket `basys.client.ui` (*user interface*) enthalten. Unterpakete

Abbildung 5.4: BASys Applikation nach dem Start

beinhalten logisch und funktional zusammengehörende Klassen. Dialoge sind beispielsweise in `basys.client.ui.dialogs` zu finden.

5.10.1 Aufbau der graphischen Oberfläche

Das in Abbildung 5.4 zu sehende Hauptfenster mit seinem Menü und seiner Werkzeugleiste wird durch die Klasse `Mainframe` erzeugt. Diese Klasse implementiert auch einen `ActionListener`, um die Menü- und Knopfereignisse zu verarbeiten.

Die Logik der Applikation steckt jedoch in der Klasse `Application-EventHandler`. Die Methoden dieser Klasse werden von den Benutzerkomponenten bei eintreffen entsprechender Ereignisse (Befehle) aufgerufen, um die gewünschten Aktionen auszuführen. Die dadurch erreichte Trennung von Oberfläche (*view*) und Logik (*controller*) wird im gesamten Softwaresystem durchweg verwendet. Einzige Ausnahme bilden einige kleinere eigenständige Dialoge, die Ansicht- und Kontrollogik in einer Klasse vereinen.

Auf Daten wird nur über Datenmodellklassen zugegriffen. Dieser Ansatz der Trennung von Datenhaltung und Datenanzeige bzw. -bearbeitung wird auch in der Swing-Bibliothek gemacht, wo es zu fast jedem Dialogelement ein entsprechendes Datenmodell gibt.

Neben dem Aufbau des Fensters im Konstruktor der Klasse besitzt `Mainframe` noch Methoden zur Aktivierung der Menüeinträge und Knöpfe sowie Methoden zum Austausch der Bedienkomponente in der Mitte des Fensters (Hauptarbeitsbereich des Fensters).

5.10.2 Projektbezogenes Arbeiten

Die Datenhaltung und auch die Arbeit mit der Applikation geschieht projektbezogen. Wenn ein Projekt geöffnet ist, erscheint die Projektansicht in der Mitte des Hauptfensters (Abbildung 5.5). Programmtechnisch spiegelt sich das in der Klasse `Project` wider, die alle projektbezogenen Datenobjekte enthält. Momentan sind dies die beiden Datenmodelle (Gebäude- und Installationsdaten). Die Klasse implementiert eine `Observer`-Schnittstelle um Änderungen an den Projektdaten zu erfassen. Dadurch kann das Programm beim Beenden überprüfen, ob eine Sicherung der Daten erforderlich ist oder nicht.

Die Kapselung der Projektdaten in eine explizite Klasse ist bei Bearbeitung jeweils eines einzigen Projektes eigentlich nicht erforderlich, jedoch können durch diesen bereits heute gemachten Ansatz zukünftige Programmversionen auch mehrere Projekte gleichzeitig offen halten und bearbeiten.

Abbildung 5.5: BASys nach dem Öffnen eines Projektes

Die Projektansicht besteht aus einer Baumansicht der Gebäudestruktur und rechts davon der Datenanzeige des jeweils im Baum ausgewählten Gebäudeteils. Wie man sieht besteht die Datenansicht aus einer Karteikartenstruktur (*TabbedPane*), bei der es momentan nur eine Karteikarte (Strukturelle Ansicht in Tabellenform) gibt. In zukünftigen Versionen sollen hier auch eine CAD und andere Ansichten der Gebäudeteile auswählbar sein. Die gesamte Oberfläche der Projektansicht, bestehend aus Baum und Karteikarte, wird durch die Klasse `ProjectviewPane` realisiert.

Auf die Implementierung von CAD Projektansichten wird in dieser Arbeit aus zeitlichen Gründen bewußt verzichtet. Sowohl die Implementierung zum Erstellen, als auch zur Darstellung von CAD Ansichten hätte den Rahmen dieser Arbeit gesprengt. Die Anbindung von CAD Daten gehört aber auch weiterhin zu den sehr wünschenswerten Funktionen des Systems, da sich mit ihnen auch Gebäudeplan-bezogene Tableauübersichten automatisch generieren lassen.

Alle Klasse, die den Gebnäudestrukturbaum betreffen, sind im Paket `basys.client.ui.tree` enthalten. Der Baum kommuniziert über Events mit den Daten-anzeigenden Komponenten, indem er einen `ShowViewEvent` aus dem Paket `basys.client.ui.event` auslöst.

5.10.3 Drag und Drop

Angelehnt an das Bedienkonzept der EIB Tool Software verwendet auch BASys durchwegs Drag und Drop zur Planung der Gebäudeautomation. In der strukturellen Ansicht können Gebäudeteile, passive Komponenten (Lampen, Jalousien, Ventile) sowie Sensoren per Drag und Drop in die Baum- und Tabellenansicht gezogen werden. Um das Drag und Drop zu realisieren, werden von den Klassen `JLabel`, `JTree` und `JTable` um entsprechende Listener erweiterte Klassen abgeleitet. Die in der strukturellen Projektansicht sichtbare Werkzeugleiste beispielsweise verwendet die Klasse `DnDLabel`, um Drag-Quellen zu realisieren. Die gesamte strukturelle Ansicht ist in der Klasse `StructuralViewPane` enthalten.

5.10.4 Tabellenansichten der Projektstruktur

Die Projektstrukturdaten der einzelnen Gebäudeteile werden durch Tabellen dargestellt. In der hier implementierten Version wird bei den meisten Gebäudeteilen bisher nur der Name angezeigt. Allen Tabellen liegen entsprechende Tabellenmodellklassen zugrunde, die die Daten aus den Datenmodellen heraussuchen. Die von `AbstractTableModel` abgeleiteten Klassen liegen im Paket `basys.client.ui.tablemodels`. Damit zukünftige Daten in den

Tabellen angezeigt werden, müssen nur die Modelle entsprechend modifiziert werden.

5.10.5 Einsatz des `Command` Entwurfsmusters

Um bereits bearbeitete Daten wiederherstellen zu können, wird das Entwurfsmuster *Command* eingesetzt. Alle Aktionen, die rückgängig machbar sein sollen, müssen von der im Paket `basys.client.commands` liegenden Klasse `Command` abgeleitet sein.

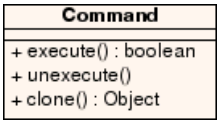

Abbildung 5.6: Command-Klasse

Abbildung 5.6 zeigt den Aufbau der Klasse `Command`. Durch den Aufruf der entsprechenden Methoden lassen sich Aktionen rückgängig machen und wiederherstellen. Eine genaue Beschreibung des Entwurfsmusters findet sich in [GH+95].

Alle Kommandos, die das Bearbeiten der Gebäudestruktur und Installation betreffen, sind durch solche Kommandoklassen realisiert.

5.10.6 Inbetriebnahme der EIB Geräte

Die Inbetriebnahme der EIB Geräte erfolgt mit dem in Abbildung 5.7 dargestellten Dialog. Alle unprogrammierten EIB Geräte befinden sich in der Tabelle und können einzeln oder komplett programmiert werden. Ist noch keine physikalische Adresse programmiert, wird diese zunächst geschrieben. Der Anwender wird dazu gegebenenfalls aufgefordert die entsprechende Programmiertaste zu drücken.

Der Dialog wird durch die Klasse `EIBProgrammingDialog` realisiert. Diese Klasse enthält auch die Logik zur Durchführung der Programmierung und nutzt dazu die in Abschnitt 5.8 beschriebenen Klassen. Um die beiden Fortschrittsbalken in Echtzeit zu aktualisieren, geschieht der Programmiervorgang in einem eigenen klasseninternen Thread.

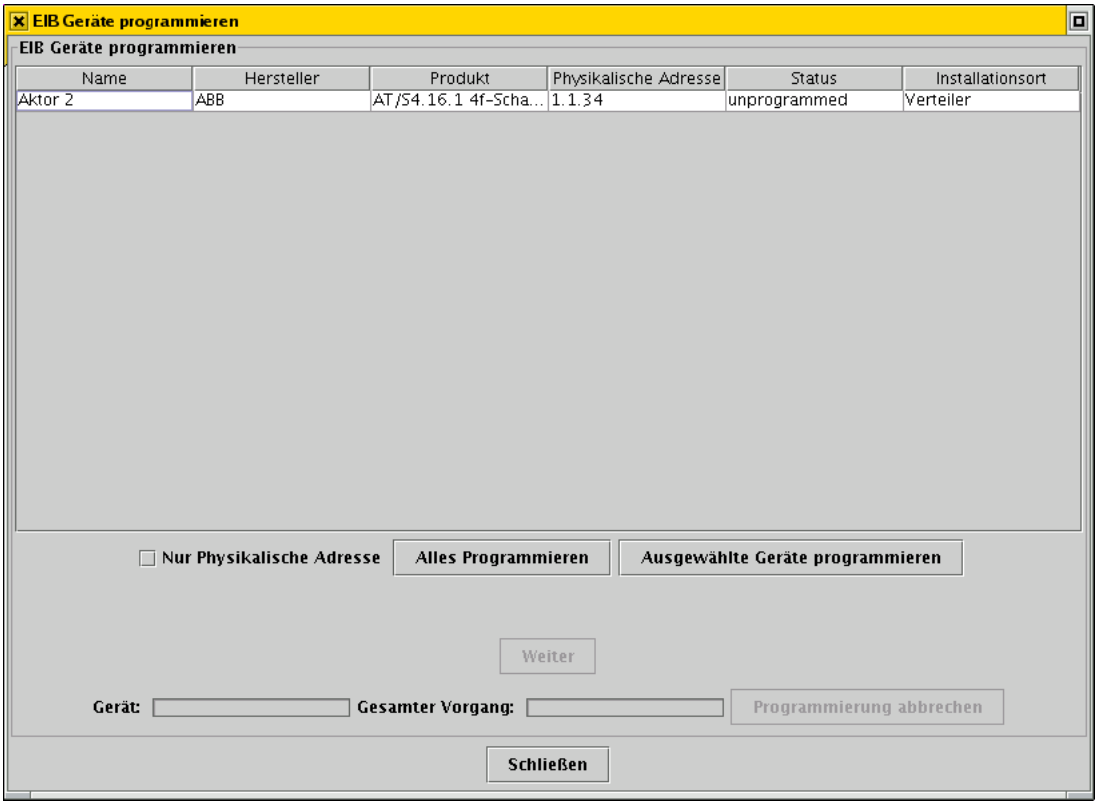

Abbildung 5.7: Dialog zum Programmieren der EIB Geräte

5.11 Web-Schnittstelle: Ein Servlet steuert EIB Geräte

Um die Leistungsfähigkeit der Datenmodelle und die simple EIB Kommunikation zu demonstrieren, existiert neben dem Applikations-Clienten auch noch eine Server-Komponente in Form eines Java-Servlets.

Das Servlet benutzt die Datenmodelle, um daraus eine webbasierte strukturelle Gebäudeansicht zu generieren. Desweiteren wird eine EIB Verbindung hergestellt, und im betriebsbereiten Zustand befindliche Aktoren lassen sich schalten. Die Zustände der Geräte werden Servlet-intern in einer Liste gespeichert und zwar so, daß auch Zustandsänderungen, die beispielsweise durch konventionelle Schalter herbeigeführt werden, registriert werden. Beim erneuten Aktualisieren der HTML Ansicht werden die Änderungen dann auch sichtbar.

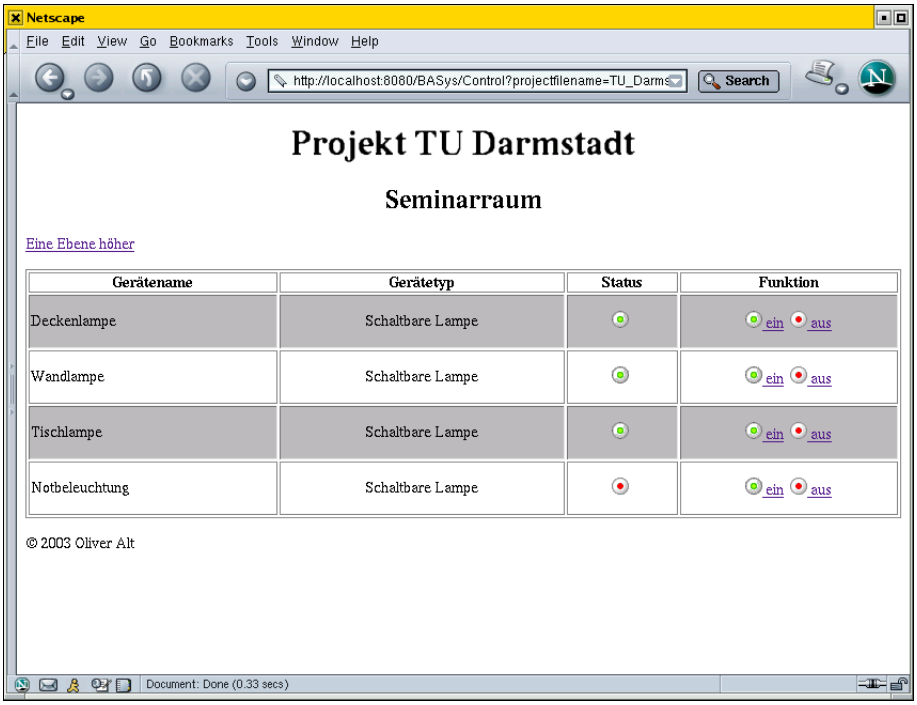

Abbildung 5.8: Ansicht des Servlets

Um eine Echtzeitanzeige der Daten zu erreichen, kann es sinnvoll sein, die anzeigenden Komponenten (Status) als Applet zu realisieren[1] .

[1]Servlets verlieren nach Übertragung der HTML Seite die Verbindung zum Client

Das `InstallationDeviceControlServlet` ist daher eher als Technologie Demonstration und im momentanen Stadium weniger als einsetzbare Komponente zu sehen.

5.12 Testklassen

Zu einigen Programmteilen, insbesondere der Datenmodellklassen, existieren Testklassen, die nach den Vorgaben des *JUnit-Framework* geschrieben sind. Sie testen Methoden und Konstruktoren der zugehörigen Klassen und geben dem Programmierer zudem Hinweise auf die korrekte Verwendung der Methoden und Konstruktoren.

Alle Testklassen liegen in Unterpaketen `tests` des Paketes der zu testenden Klasse. Nicht alle Programmteile können jedoch mit JUnit getestet werden. Das korrekte Funktionieren der graphischen Oberfläche, sowie die EIB Programmierung kann nur durch in Augenscheinnahme bzw. Netzwerk-Protokolle und (semantische) Analyse sichergestellt werden.

Kapitel 6

Fazit und Ausblick

Das in dieser Arbeit entworfene und realisierte Softwaresystem BASys ermöglicht die Planung und die Inbetriebnahme von Gebäudeautomationssystemen. Die in Kapitel 3 angeführten Anwendungsfälle konnten weitgehend realisiert werden. Die entstandenen Datenformate ermöglichen es, Bussysteme übergreifend zu planen. Der Planer einer solchen Gebäudeautomation bekommt durch das System eine Reihe von Hilfestellungen beim Auswählen passender Aktoren und Sensoren, die bei bisherigen Softwaresystemen von Hand ausgewählt werden mußten.

Sämtliche Daten liegen im XML Format vor und können daher leicht verarbeitet und erweitert werden. Die hier vorgestellte Implementierung ermöglicht bereits die Planung und Inbetriebnahme von Gebäudeautomationssystemen mit dem Europäischen Installationsbus (EIB). Die Software stellt außerdem das erste EIB Inbetriebnahmewerkzeug dar, das im Rahmen einer nicht kommerziellen Hochschularbeit entstanden ist. Damit könnte die Verbreitung des EIB weiter zunehmen, da das neue System die Anschaffung der kommerziellen (und teuren) EIB Tool Software nicht mehr zwingend erforderlich macht.

BASys ist kein auf den EIB beschränktes System. Es sollte leicht möglich sein, auch andere Bussysteme wie LON oder LCN in die Software zu integrieren. Durch die durchgehende Verwendung von Java als Programmiersprache ist die Software plattformunabhängig und objektorientiert. Der Einsatz von Entwurfsmustern strukturiert den Quellcode. Weiterentwicklungen und Ergänzungen sollten daher ohne aufwendige Einarbeitung nach kurzer Zeit erfolgen können.

Die Client-Applikation befindet sich noch im Prototyp-Stadium. Es sollte jedoch mit absehbarem Aufwand möglich sein, sie zu einem marktreifen Produkt weiter zu entwickeln. Dazu sind umfangreiche Tests notwendig. Besondere Beachtung sollte hier auf die Akzeptanz der Benutzerführung gelegt

werden, damit das fertige Produkt dann auch von den Anwendern angenommen wird.

Weiterhin muß sichergestellt sein, daß sich alle am Markt befindlichen Busgeräte damit in Betrieb nehmen lassen können. Ein potentieller Anwender wird nämlich das System nur dann verwenden, wenn er all seine Geräte auch programmieren kann.

Mit dem implementierten Servlet zur Kontrolle der EIB Geräte wurde außerdem gezeigt, daß es mit geringem Aufwand möglich ist, aus den Planungs- und Installationsdaten Bedienelemente zu erzeugen. Die Datenformate des Systems bilden daher auch eine Grundlage für das Gebäude im Betriebszustand. Die Datenklassen des Systems sind nicht an die Client-Applikation gebunden, sondern können auch als Basis weiterer Gebäudesoftware (Stichwort: *Facility Management, ...*) herangezogen werden.

6.1 Integration von CAD Daten

Die Anbindung von CAD Daten (Import oder eigener Editor) sollte in einer der nächsten Versionen implementiert werden. Mit Hilfe der Daten können nicht nur aussagekräftige Bedienoberflächen für den Gebäudebetrieb generiert, sondern die elektrische Installation der Busgeräte erleichtert werden. Der Installateur kann vom System erstellte Pläne verwenden, um die passiven Komponenten, Aktoren und Sensoren zu installieren.

Auch die Inbetriebnahme der Geräte (Programmierung) wird vereinfacht, da der Standort der Busgeräte vom System in Planansichten angezeigt werden kann. Mit einem PDA könnte der Installateur beispielsweise seinen Standort angeben. Anschließend wäre es möglich, daß das System eine Meldung ausgibt, an welchem der Geräte, vor denen er sich zu diesem Zeitpunkt befindet, der Programmiertaster zu drücken ist.

6.2 Unterstützung mehrerer Bussysteme

Der EIB bildet das einzige momentan unterstützte Bussystem. Außerdem ist das EIB-Ethernet-Gateway zur Zeit die einzige Möglichkeit zur Kommunikation des Systems mit dem EIB. Ziel zukünftiger Versionen sollte daher die Anbindung des EIB über andere Schnittstellen sowie die Integration weiterer Bussysteme sein. Um Systeme wie LON und LCN anzubinden, gilt es vorweg eine Reihe von Fragen zu beantworten:

- Welche Daten werden benötigt, um Busgeräte passiven Komponenten (Lampen, etc.) zuzuordnen?

- Welche Daten werden benötigt, um die Busgeräte zu programmieren?
- Wie geschieht die Buskopplung an das System?
- Wie werden Geräte lokalisiert (Programmiertaster, fest eingebrannte Adresse)?
- Wie können Übergänge zwischen den verschiedenen Bussystemen realisiert werden?

Nach Beantwortung dieser Fragen müssen die XML Datenstrukturen entsprechend erweitert werden und die Funktionen zur Planung und Inbetriebnahme implementiert werden. Die bestehenden Paketstrukturen könnten dann z.B. um ein Paket `basys.lcn` ergänzt werden.

6.3 Versionsverwaltung

In der bisherigen Implementierung werden alle Daten lokal im Arbeitsverzeichnis des aktuellen Benutzers gespeichert. In Zukunft sollten die Daten jedoch zentral mit Versionsinformationen gehalten werden. Das bekannte Versionsverwaltungssystem CVS (www.cvshome.org) bietet sich zu diesem Zweck an. CVS ist für alle wichtigen Plattformen erhältlich und unterstützt zudem Datenübertragung über Netzwerke. Eine Java-Anbindung dürfte mit vertretbarem Aufwand machbar sein. Durch den Einsatz des etablierten Systems braucht kein neuer, eigener Ansatz getätigt zu werden und aufwendige Funktionstests können entfallen.

6.4 Sprachsteuerung

In der nahen und mittelfristigen Zukunft wird es voraussichtlich eine Reihe neuer Sensorelemente für die Steuerung und den Zugriff auf Gebäudefunktionen geben. Sprachsteuerung könnte hierbei eine bedeutende Rolle zukommen. Für Java gibt es bereits heute Bibliotheken, um Sprachsteuerungssysteme anzubinden. Daher sollte auch die Möglichkeit ins Auge gefaßt werden, BASys mit Spracheingabesystemen
(z.B. IBMs *Via Voice*, www.ibm.com/software/speech/) zu verbinden.

Die Plattformunabhängigkeit sollte aber auch beim Einsatz von Spracheingabe erhalten werden.

6.5 Serverkomponenten

Mit der Java Enterprise Edition (J2EE) wird es erleichtert, Applikationslogik als Server Komponente zu realisieren. Bei der Weiterentwicklung des BASys Systems sollte daher überlegt werden, für welche Funktionen es sinnvoll sein kann, Komponenten serverseitig auszuführen. Beispielsweise könnte die gesammte Buskommunikation durch einen zentralen Server geschehen, auf den die Clients zugreifen. Auch die Integration von Facility Management Funktionalität könnte mit Hilfe von Serverkomponenten realisiert werden.

6.6 Integration aller elektrischen Geräte im Gebäude

Neben der klassischen Gebäudeautomation existieren noch eine Reihe weiterer elektrische Geräte in einem modernen Gebäude (Hifi, PC, Küchengeräte). Denkbar wäre, BASys so zu erweitern, daß es Schnittstellen zu solchen Geräten anspricht und bei der Automationsplanung einbezieht (z.B Steuerung der Gebäudefunktionen per Settop-Box am Fernseher).

Das System könnte daher den Grundstein für eine Software legen, die plattformunabhängig die elektrischen Geräte eines Gebäudes verbindet und eine umfassende Automation ermöglicht. Daraus sollte sich eine Steigerung der Lebensqualität und des Komforts für die Nutzer der Gebäude ergeben. Solche unter Schlagworten wie *Haus der Zukunft* oder *Smart Home* gehandelten Wünsche werden heute oft noch als Zukunftsvision abgehandelt.

Mit etwas Entwicklungsaufwand sollte die Software so erweitert werden können, daß sie kostengünstig die oben genannten Funktionen bieten und erfüllen kann. Schon in naher Zukunft sollten daher erste Gebäude entstehen können, die mit umfassender und kostengünstiger Funktionsvernetzung ausgestattet sind.

Vorher muß natürlich geprüft werden, welche Funktionen sinnvoll sind und welche nicht. Viele Ingenieure neigen dazu, ihrem Spieltrieb nachzugeben. Daher gibt es heute schon E-Mail versendende Kaffeemaschinen und Toaster. Der praktische Nutzen für den normal technisch interessierten Anwender ist hierbei eher in Frage zu stellen. Eine gründliche Anforderungsanalyse und einfache Bedienbarkeit sollten aber auch im Gebäudeautomationsbereich sinnvolle Produkte und Lösungen entstehen lassen, die weitgehend akzeptiert werden und daher auch Eingang in BASys finden können.

Literaturverzeichnis

[Alt02] Oliver Alt. *Entwurf und Realisierung einer EIB zu Ethernet Brücke (EIB-Gateway) in Hard- und Software.* TU Darmstadt, Studienarbeit KOM-S-137, 2002.

[D+01] Harvey M. Deitel et al. *Advanced Java 2 Platform: How to program.* Prentice Hall, 2001.

[DKS00] Dietmar Dietrich, Wolfgang Kastner, and Thilo Sauter. *EIB Gebäudebussystem.* Hüthig Verlag, 2000.

[DLS99] Dietmar Dietrich, Dietmar Loy, and Hans-Jörg Schweinzer. *LON-Technologie. Verteilte Systeme in der Anwendung.* Hüthig Verlag, 1999.

[EIB99] EIBA. *EIB Handbook for development, Volume 3.* EIBA Brüssel, 1999.

[Fra97] Karlheinz Frank. *EIB Ein neues Geschäftsfeld für den Elektroinstallateur.* Verlag Technik Berlin, 2. Auflage 1997.

[GH+95] Erich Gamma, Richard Helm, et al. *Design Patterns, Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1995.

[GR00] Michael Gossens and Sebastian Ratz. *Mit LaTeX ins Web.* Addisson-Wesley, 2000.

[HC01] Jason Hunter and William Crawford. *Java Servlet Programming.* O'Reilly, 2001.

[Krü97] Guido Krüger. *Java 1.1 lernen Anfangen, anwenden, verstehen.* Addison-Wesley, 1997.

[McL01] Brett McLaughlin. *Java und XML.* O'Reilly, 2001.

Internet-Links

Die nachfolgenden Internetseiten enthalten Informationen zu in dieser Arbeit verwendeten Werkzeugen und auf manche Projekte, auf die im Text Bezug genommen wurde.

www.eclipse.org Hier finden sich alle Programme und Informationen zur Eclipse Entwicklungsumgebung.

www.junit.org Informationen und Downloads rund um das Java Unit Testing Framework JUnit.

www.javasoft.com Homepage des Java Herstellers Sun. Hier finden sich die Entwicklungskits und sämtliche Dokumentation rund um Java.

www.sourceforge.net/uml Homepage des UML Editors Umbrello.

jakarta.apache.org Homepage des Apache/Jakarta Projektes. Hier findet sich unter anderem auch Log4j.

www.hto.fh-deggendorf.de/komm/englisch/eprojekte.html Projektseite der EIB Projekte an der FH Deggendorf. Unter anderem finden sich hier Informationen über eine Palm PDA-EIB Anbindung.

www.w3.org/DOM/ Informationen rund um das XML Document Object Model.

www.rational.com Informationen zum Rational Unified Process - einem Software Engineering Verfahren.

www.schlaps-automation.de Homepage der Firma F. Schlaps und Partner GmbH. Produkte des Unternehmens sind EIB-Gateways für den Anschluß an Ethernet und USB. Einige werden ohne Zertifizierung durch die EIBA vertrieben.

Anhang A

LCN

Die nachfolgenden Informationen über das LCN entstammen der Homepage des Herstellers (Fa. Issendorf).

Grundlagen

Das LCN integriert die gesamte Gebäudeinstallation in ein umfassendes Bussystem: Dort, wo bisher Schalter angebracht waren, werden jetzt kleine Computermodule eingebaut. Alle Module werden über eine zusätzliche Ader der gewöhnlichen Installationsleitung miteinander verbunden. Über diese Ader tauschen die LCN-Module untereinander Nachrichten aus. So kann ein Modul einem beliebigen anderen mitteilen: *Schalte Deinen 2. Ausgang ein!*

Die Module arbeiten absolut selbständig. Sie brauchen weder eine getrennte Stromversorgung, noch eine besondere Zuleitung. Sie bieten immer gleich mehrere Funktionen: Zwei Schaltausgänge und zwei/drei unabhängige Eingänge ermöglichen es, mit weniger Modulen und weniger Verdrahtungsaufwand auszukommen, als das bisher nötig war.

Alle LCN-Module beinhalten neben der Sensorik und Aktorik auch mehrere Zeitgeber und Verknüpfungen sowie eine Zähl- und Rechenfunktion, so daß automatische Steuerungen direkt vor Ort realisierbar sind.

Auch für das Erfassen und Verarbeiten analoger Messwerte sind LCN Module ausgerüstet. Die Meßwerte können fernabgefragt werden.

Multi-Master-Bus

Als modernes System benötigt das LCN keine Zentrale: Alle Module sind intelligent genug, den Datenverkehr untereinander selbst zu regeln. Jedes einzelne Modul kann als *Master* den Bus steuern. Der Bauherr kann ganz

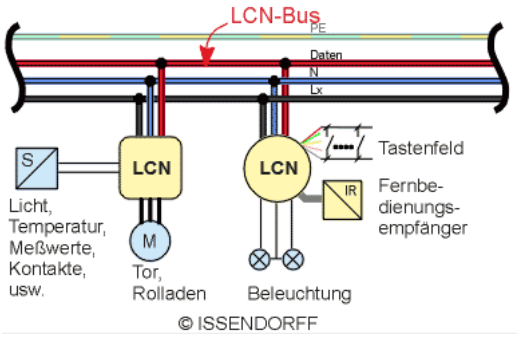

Abbildung A.1: LCN Verdrahtung

klein anfangen. Schon zwei Module bilden ohne weitere Hilfsmittel einen funktionierenden Bus. Tatsächlich können Sie die Module sogar einzeln einsetzen, z.B. mit IR-Empfänger als fernbedienbaren Doppeldimmer oder als Zutrittskontrolle.

Stück für Stück kann das LCN-Bussystem bis zu großen, alles umfassenden Gebäudeleitsystemen ausgebaut werden. Die Grenze liegt bei 30000 Modulen. Damit können 10000 bis 60000 Räume pro Objekt ausgestattet werden, je nach Anspruch des Bauherren.

LCN Bus-Module sind als Unterputz- oder Hutschienenmodule konzipiert, die gleichzeitig Sensorik und Aktorik vereinen. Sie verfügen alle über zwei dimmbare Ausgänge mit unterschiedlichen Ausgangsleistungen, je nach Typ 300W und 2kW. An diesen Ausgängen werden die Verbraucher angeschlossen, wie z.B. Glühlampen, Leuchtstofflampen, Antriebsmotoren usw. Über integrierte Schnittstellen können direkt Sensoren angeschlossen werden, mit denen standardmäßig Licht, Temperatur, Windgeschwindigkeit, Regen und Druck gemessen werden können.

Aufbau

Ein LCN-Bus ist einfach zu verdrahten. Bis zu 250 Module (siehe Abbildung) werden direkt miteinander verbunden über nur drei Anschlüsse: Phase, Neutralleiter und die Datenader. Bis zu mittleren Objektgrößen braucht die untere Busebene nicht verlassen zu werden.

Die Datenader ist eine der zwei freien Leitungen eines 5-adrigen NYM-Kabels. Sie kann genau wie die anderen Adern behandelt werden, Trennstege, etc. sind nicht erforderlich.

Bei grossen Gebäudekomplexen können bis zu 120 dieser Segmente über

einen Segmentkoppler (SK) miteinander verbunden werden. Diese Segmentierung kann zum Beispiel auch genutzt werden, um in einem Mehrfamilienhaus die einzelnen Wohneinheiten gegeneinander abzugrenzen und trotzdem die Kommunikation untereinander zu ermöglichen, z.B. für die Außenlichtsteuerung, Gefahrenmeldungen, usw..

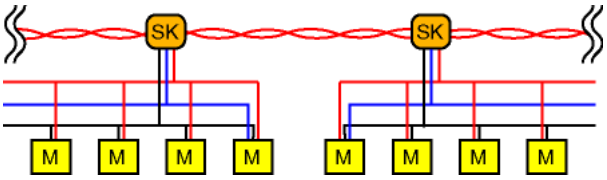

Abbildung A.2: LCN Topologie

Adressen

Damit jedes LCN-Modul ansprechbar ist, braucht es eine Adresse, eine Zahl zwischen 5 und 254, die per Installationssoftware LCN-P in Sekundenschnelle vergeben wird. Ein Zugang zum Modul ist dafür nicht erforderlich: Der PC wird an beliebiger Stelle an den Bus angeschlossen.

Wenn mehrere Netzsegmente über den Segmentbus gekoppelt werden, bekommen die Segmente jeweils eine Nummer zwischen 5 und 125.

Von beliebiger Stelle im Bus kann ein Datentelegramm an diese Adresse abgeschickt werden. So können alle installierbaren LCN-Module (max. 30000 Stk.) angesprochen werden.

Wenn mehrere Teilnehmer mit einem Telegramm angesprochen werden sollen, geschieht dies über die Gruppenadressierung. Pro Bussegment können 250 Gruppennummern im Bereich von 5....254 frei vergeben werden. Die Anzahl der Mitglieder pro Gruppe ist unbegrenzt.

Im LCN-System werden Gruppen nur dann gebildet, wenn auch tatsächlich mehrere Verbraucher gleichzeitig angesprochen werden sollen. Bei einer Punkt-zu-Punkt-Übertragung dagegen, wird direkt adressiert (an Modul 55: EG WC Licht), wie man es intuitiv auch erwarten würde.

Neben der Anschaulichkeit gibt es noch einen wichtigen weiteren Grund für die direkte Adressierung: Module können einander Informationen zur Weiterverarbeitung direkt zusenden und bilden so Bestandteile eines neuronales Netzwerkes, das mit steigender Modulzahl immer intelligenter und leistungsstärker wird. Das ist für zukünftige Automatisierungsaufgaben wichtig.

Datenübertragung

Im LCN-System werden durchschnittlich 100 Telegramme pro Sekunde übertragen. Das entspricht einer Datenübertragungsrate von 9600 Bd. Die Ausnutzung eines schon vorhandenen Leiters spart Kupfer, vereinfacht die Installation und macht das System unabhängig von der Phasenlage der einzelnen Module. Das LCN arbeitet im Basisband, die bei Trägerfrequenzübertragung üblichen Hilfsmittel (Phasentrenner/-koppler, Sperren, etc.) sind nicht erforderlich.

Zur Vereinfachung der Installation nach VDE ist die Datenader per Definition eine Netzader, obwohl im regulären Betrieb Spitzenspannungen von nur +-30V auftreten. Gegen höhere Spannungen am Datenanschluss, z.B. bei einem Installationsfehler, sind die Module bis 2 kV geschützt.

Die Datenader darf im Gebäude beliebig verdrahtet werden, eine spezielle Topologie wie z.B. die Sternform braucht nicht eingehalten zu werden. Die maximale Gesamtlänge beträgt 1 km und kann mit Zwischenverstärkern verlängert werden. Außerdem sind Lichtleiterkopplungen - z.B. zwischen Unterverteilungen - möglich. Bei Kunststoff-Lichtleitern, die mit einfachen Mitteln auf der Baustelle angeschlossen werden können, beträgt die Reichweite 100m / Strecke. Mit Glasfaserkopplern werden pro Strecke 2 km (optional 5 km) erreicht.

Telegramme

LCN-Datentelegramme haben eine flexible Struktur und können unterschiedlich lang sein.

Grundkomponenten:

```
Absender Ziel Prüfsumme Info Daten ... ...
```

Im Durchschnitt können etwa 100 Telegramme pro Sekunde übertragen werden. Ein mehrstufiges Verfahren zur Vermeidung von Datenkollisionen stellt sicher, daß auch bei hoher Buslast die Buskapazität voll ausgenutzt wird.

Die Überprüfung der Aussendung ist mehrstufig aufgebaut, so daß das LCN auch in solchen Umgebungen, die weit stärker gestört sind als geltende Normen zulassen, trotzdem perfekt arbeitet.

LCN Datentelegramme enthalten in kompakter Form sehr viel mehr Informationen, als dies bisher möglich war. Sie beschreiben die Funktion eines Sensors oder Aktors vollständig. So enthält ein Kommando an eine Leuchte zum Beispiel nicht nur die gewünschte Helligkeit, sondern auch die Geschwindigkeit, mit der diese erreicht werden soll. Zeitgeber brauchen nicht

im Aktor programmiert zu werden, denn jedes Telegramm kann die Zeitinformation enthalten. So kann der gleiche Aktor beliebig viele unterschiedliche Zeitschaltungen ausführen. Jede Taste kann ihm einen anderen Befehl senden.

Die Wartung des Busses wird wesentlich vereinfacht.

So kann im Betrieb die Busfunktion beobachtet werden. Jedes Telegramm enthält alle erforderlichen Informationen, die in Klarschrift angezeigt werden:

Wer sendet an wen? Wie lautet der Befehl?

Zukunftssicherheit

Wegen der hohen Innovationsgeschwindigkeit in der Mikroelektronik hatten bisherige Anlagen nur eine konzeptionelle Lebensdauer von ca. 10 Jahren. Bussysteme waren danach technisch überholt. Dies ist für Bauten eine kurze Zeit. Beim LCN wurde erstmals das folgende, zweistufige Konzept entwickelt, um diese Zeitgrenze zu durchbrechen:

1. Telegrammstruktur (flexibles Datenformat). Über das LCN-Telegramm können die Module einander beliebig strukturierte Informationen übermitteln, auch solche, die heute noch nicht absehbar sind. Das Datenformat kann für jede Funktion neu definiert werden, ohne die Kompatibilität zu älteren Baugruppen zu verlieren.

2. Intelligenz. Die Gebäudesystemtechnik steht erst am Anfang. Der Trend führt zu immer komplexeren Steuerungen, Meldungen und Regelungen. Das LCN ist darauf vorbereitet, bzw. bietet bereits heute viele Möglichkeiten in jedem Modul. Sollte es einmal ganz neue Entwicklungen geben, die heute noch nicht abzusehen sind, gibt es jetzt schon Bits, die später die Übertragung ganz andersartiger Telegramme auf dem LCN-Bus ermöglichen. Dazu verfügen alle Module unter anderem über Verknüpfungen, verschiedene Zeitgeber von 10ms bis zu 45 Tagen und eine Zähl- und Rechenfunktion. So könnte ein einziges Modul die Steuerung eines Parkhauses übernehmen und dabei sogar zwischen Fahrzeugen unterschiedlicher Grösse unterscheiden. In Abhängigkeit von der Belegung könnte es Ampeln steuern, die wiederum mit einfachen LCN-Modulen realisiert werden. An alle Module können IR-Empfänger und Sensoren für die Meßwerterfassung/-Verarbeitung angeschlossen werden. LCN wird erfolgreich für die Betriebsdatenerfassung eingesetzt, dafür gibt es Strichcode-Scanner, die direkt an jedem Arbeitsplatz eingesetzt werden können. Module aus der PLUS-Serie verfügen zusätzlich

über Funktionen wie Registersätze mit Szenenspeichern, Kopplungen zur Bühnenlichttechnik, usw.

LCN ist ein offenes System: Kopplungen mit anderen Bussystemen wie EIB oder DMX-512 (ein Bussystem, dass im Bühnenbereich eingesetzt wird) sind realisiert worden.

Programmierung

Jedes LCN-Modul *kennt* die angeschlossenen Verbraucher und steuert sie entsprechend an. Glühlampen z. B. werden gedimmt, Gasentladungslampen per Powerswitch gefahren. Bei Rolladenantrieben wird sichergestellt, dass nicht beide Ausgänge gleichzeitig eingeschaltet sind, denn das würde den Motor schädigen.

Diese wichtigen Angaben über den Verbraucher gibt der Installateur bei der Programmierung ein. Die Programmierung der LCN-Module ist derart einfach, dass sie direkt auf der Baustelle mit dem Laptop des Elektrikers durchgeführt werden kann.

Tastatur

An LCN-Module können herkömmliche Taster aller Fabrikate angeschlossen werden. Weil in einem grossen Gebäude sehr unterschiedliche Aufgaben auftreten können, sind LCN-Tastenfelder frei programmierbar. Es liegt beim Planer / Installateur, die Charakteristiken der Tastenfelder vorzugeben: Von einer einfachen Licht- oder Rolladensteuerung bis hin zur komplexen Regie von Lichtszenen, stehen beispiellos viele Möglichkeiten zur Verfügung, alle Bauherrenwünsche zu erfüllen.

Das LCN unterscheidet grundsätzlich zwischen kurzem Tippen und einem langen Tastendruck, sowie dem anschließenden Los lassen. In allen drei Fällen wird ein frei programmierbarer Befehl ausgesandt.

Beim LCN werden die Tastenfunktionen dort eingestellt, wo sie hingehören: In der Tastatur. Das macht die Programmierung und Parametrierung einfach und übersichtlich.

Fernsteuersystem

LCN-Module verfügen standardmäßig über eine Impuls-Schnittstelle, an welcher ein Infrarotempfänger angeschlossen werden kann. Sämtliche Befehle

können dann auch mittels Infrarot-Fernsteuerung an das entsprechende Modul übermittelt werden.

Über eine am Sender eingebbare Codierung können Schließsysteme realisiert werden. Die Empfänger-Module werten die Codes direkt aus. Weitere Installationskosten entstehen nicht.

Die Codierung kann auch zur Realisierung privilegierter Funktionen genutzt werden, der Chef könnte z.B. berechtigt sein, die Sollwerte der Lichtregelung zu ändern.

In einem Raum können bis zu 250 Sender unabhängig voneinander betrieben werden.

Schließlich kann ein komplexes Zugangskontrollsystem mit Zeiterfassung, etc. eingerichtet werden. Ein am LCN-Bus angeschlossener PC stellt dies dann über die LCN-Visualisierungssoftware dar.

Folgende Handsender zum LCN-Fernsteuersystem sind lieferbar:

1. Ein grosser Handsender mit 8 Tasten und 50-100m Reichweite.

2. Ein kompakter Handsender der an den Schlüsselbund paßt. Er kann mit 4 Tasten 8 bis 16 Verbraucher oder Verbrauchergruppen individuell dimmen, schalten, bzw. Steuerungen auslösen. Schon im Fernsteuerbetrieb ist er damit anderen Systemen überlegen.

Visualisierung

Der Buszustand kann mit Hilfe beliebiger Tableaus visualisiert werden. So kann zum Beispiel das Leuchten des Lichts im Badezimmer oder im Keller durch Leuchten eines Lämpchens auf dem Tableau angezeigt werden. Alle LCN-Tableaumodule zeigen beliebige Zustände im Bus als Echtzeit-Meldungen an. Die Programmierung ist einfach: Man *sagt* jedem Lämpchen, welchen Sensor oder Aktor es darstellen soll. Da neben den Anzeigen EIN, AUS auch BLINKEN und FLACKERN möglich ist, können LCN-Tableaumodule Erstwert- und Letztwertmeldungen nach DIN darstellen. Sowohl die Tastenbedienung als auch die Störmeldeverarbeitung kann hierarchisch über beliebig viele Tableaus erfolgen. Störmeldungen können im Bus beliebig weiterverarbeitet werden.

Für die Visualisierung auf dem Bildschirm bietet LCN preiswerte Lösungen für den Einstieg wie für Großanlagen:

Für Windows ist eine Software erhältlich, mit der die gesamte Elektroinstallation zentral gesteuert und visualisiert werden kann. Das Programmsystem ist modular aufgebaut und kann jederzeit erweitert werden. LCN ist das einzige System, das über ein frei programmierbares Visualisierungs- und

Steuersystem mit eigener Programmiersprache unter Windows verfügt. Die Funktion des Visualisierungs- und Steuersystems geht weit über die Anzeige und Steuerung hinaus.

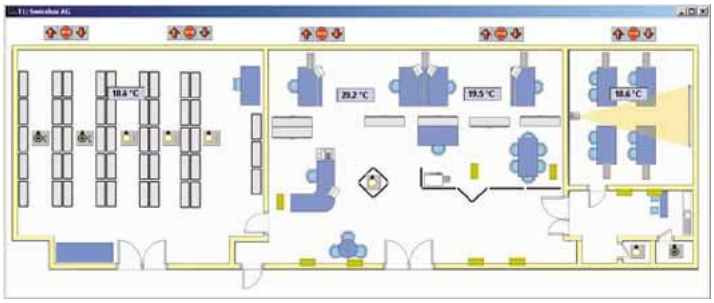

Abbildung A.3: LCN Visualisierung