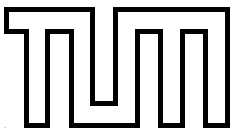


Technische Universität München

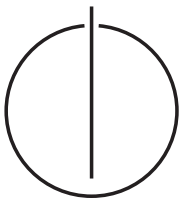
Fakultät für Informatik

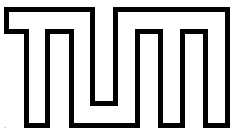
SEP in Informatik

Anlagensteuerung und -überwachung über VoIP

Aufbau eines Testbeds bei der SKYTEC AG

Oliver Rahner





Technische Universität München

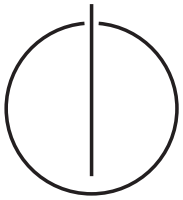
Fakultät für Informatik

SEP in Informatik

Anlagensteuerung und -überwachung über VoIP

Aufbau eines Testbeds bei der SKYTEC AG

Bearbeiter:	Oliver Rahner
Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Feng Liu Dr. Bernhard Lorenz (SKYTEC AG) Martin Roeder (SKYTEC AG)
Abgabedatum:	7. Juli 2008



Abstract

Supervisory Control And Data Acquisition Systeme, kurz SCADA, sind seit geraumer Zeit im Einsatz. Seit einigen Jahren gehen die Trends von SCADA-Systemen analog zu anderen Bereichen in Richtung der Verwendung von IP-Netzen. Ein SCADA-System, das von Beginn der Entwicklung an auf IP-Technologie setzt, ist das Produkt SCADA X-SIGHT.

Dieses ist gegenwärtig vor allem bei Betreibern von Verkehrsbetrieben im Einsatz. Bei diesen Installationen handelt sich um Systeme, die räumlich über große Gebiete verteilt sind.

Ein Problem dieser Systeme besteht darin, dass vor Ort an den zu steuernden Anlagen vollwertige PCs im Einsatz sind, um die Kommunikation zwischen Anlage und SCADA-System zu erledigen und als lokale Steuerkonsole. Bestehende Pläne sehen vor, diese PCs durch spezialisierte Embedded-Komponenten zu ersetzen. Um trotzdem über eine lokale Kontroll- und Steuermöglichkeit zu verfügen, sollen Voice-over-IP-Telefone als Konsole zum Einsatz kommen.

Diese Arbeit untersucht die generelle Machbarkeit des gewünschten Einsatzes von VoIP-Technologie und beschreibt die entstandene Implementierung in Form eines Moduls für das oben genannte SCADA X-SIGHT.

Inhaltsverzeichnis

1. Einführung **1**
1.1. Anwendungsszenario . . . 1
1.2. Problemstellung . . . 1

2. technische Grundlagen **3**
2.1. SCADA-Grundlagen . . . 3
2.2. FCML . . . 5
2.3. SCADA X-SIGHT . . . 5
2.3.1. Schichtmodell . . . 5
2.3.2. Module . . . 6
2.4. Beschreibung der Cisco-Geräte . . . 8
2.4.1. Tisch-Telefon 7971G-GE . . . 8
2.4.2. Tisch-Telefon 7961G-GE . . . 9
2.4.3. WLAN-Handset 7921G . . . 9
2.4.4. Cisco 2811 Integrated Services Router . . . 9
2.4.5. Cisco Aironet 1240G Access Point . . . 10
2.4.6. CallManager Simulator . . . 10

3. Aufbau des Testbeds - Hardware **11**
3.1. Physikalischer Aufbau . . . 11
3.2. Konfiguration der Telefone . . . 12
3.2.1. DHCP-Betrieb . . . 12
3.2.2. Betrieb mit manueller IP-Konfiguration . . . 12
3.3. Konfiguration des Routers . . . 13
3.3.1. Grundkonfiguration mit Hilfe des Assistenten . . . 13
3.3.2. Grundkonfiguration über den Assistenten hinaus . . . 16
3.3.3. Grundkonfiguration des CallManager Express . . . 16
3.3.4. Einstellen der CallManager-IP-Adresse . . . 17
3.3.5. Aktivieren der Konfigurationsdateien . . . 17
3.3.6. Einstellen der maximalen Anzahl von Verzeichniseinträgen und Telefonen 17
3.3.7. Konfiguration eines Telefons . . . 18
3.3.8. Konfiguration der Dienst-URLs . . . 18
3.4. Konfiguration des Access Points . . . 19
3.4.1. Cisco IOS installieren . . . 19
3.4.2. Zuweisen einer IP-Adresse . . . 19
3.4.3. Konfiguration der WLAN-Einstellungen . . . 20
3.5. Konfiguration des CallManager Simulator . . . 21

4. Software-Implementierung des Testbeds **23**

4.1. Anbindung an den X-SIGHT-Proxy . . . 23
4.1.1. Konzept . . . 23
4.1.2. Umsetzung . . . 24
4.1.3. Vor- und Nachteile . . . 25
4.2. Anbindung als X-SIGHT-Modul . . . 25
4.2.1. Konzept . . . 25
4.2.2. Umsetzung . . . 26

5. Ergebnis und Fazit **33**
5.1. Tests . . . 33
5.1.1. Testumgebung . . . 33
5.1.2. Unit-Tests . . . 33
5.2. Zusammenfassung . . . 34
5.3. Fazit . . . 35

6. Ausblick **37**
6.1. weiterführende Arbeiten . . . 37
6.2. andere Anwendungsgebiete . . . 37

A. Cisco XML **39**

Abbildungsverzeichnis **41**

Literaturverzeichnis **43**

1. Einführung

1.1. Anwendungsszenario

Zum besseren Verständnis wird das komplette Konzept dieser Arbeit mit einem praktischen Anwendungsfalles untermauert.

Als Beispiel dient die Essener Verkehrs-AG (EVAG), da dort zur Zeit ein FCML[1]-basiertes SCADA[2]-System installiert wird. Von den 38 U-Bahn- und Betriebshöfen sind bisher 25 mit FCML-Proxies ausgestattet, zwei weitere folgen noch.

Dadurch, dass gerade bei Verkehrsbetrieben diverse Anlagen wie z.B. Aufzüge, Fahrtreppen, Brandschutzeinrichtungen etc. über einen großen Bereich verteilt sind, eignen sich diese gut zur Visualisierung der Problemstellung.

Ohne die Unterstützung eines zentralen Überwachungssystems wäre es nahezu unmöglich, den Zustand jeder Anlage zu jeder Zeit im Blick zu haben und beurteilen zu können, wo Handlungsbedarf in Form von Reparaturen, Wartung etc. besteht.

1.2. Problemstellung

Um ein Überwachungssystem in dieser Größenordnung zu ermöglichen, bedarf es einer großen technischen Infrastruktur, die Anfälligkeiten mit sich bringt.

Diese Arbeit setzt an solch einem Punkt großer Anfälligkeit an. Es geht hierbei um die sogenannten FA-PCs, die sich momentan an jeder Fahrtreppe und jedem Aufzug befinden (Abbildung 1.1). Diese Geräte sind vollwertige PCs in Industrieausführung, die für die lokale Steuerung der Anlage sowie die Umsetzung der proprietären Formate in FCML verantwortlich sind. An ihren Standorten sind sie vielen schädlichen Faktoren ausgesetzt, vor allem einer großen Bandbreite von Temperaturen sowie viel Staub. Die Vergangenheit hat gezeigt, dass diese PCs alle 3 - 5 Jahre ausgetauscht werden müssen.

Um die daraus entstehende finanzielle Belastung zu eliminieren, verfolgt diese Arbeit einen anderen Gedanken.

Die FA-PCs werden durch zwei andere Komponenten ersetzt. Die Aufgabe der Formatkonvertierung übernimmt dann ein Embedded Device. Diese Geräte haben den Vorteil, dass sie einen wesentlich niedrigeren Energiebedarf haben, nur einen Bruchteil des Platzes benötigen und sie auf äußere Faktoren wesentlich robuster reagieren als ihre großen Kollegen.

[1] Facility Markup Language, siehe Abschnitt 2.2
[2] Supervisory Control and Data Acquisition, siehe Abschnitt 2.1

Abbildung 1.1: Steuerschrank mit eingebautem FA-PC

Da aber trotzdem eine lokale Diagnose und Steuerung der Anlage möglich sein muss, werden hierfür Cisco IP-Telefone eingesetzt. Dies erhöht die Komplexität des Gesamtsystems nur unwesentlich, da durch diese IP-Telefone natürlich die bisherige Telefon-Infrastruktur ersetzt werden kann. Die EVAG hat bisher noch konventionelle Telefonsysteme im Einsatz, was neben dem Ethernet/LWL-Netz ein zusätzliches Kabelnetz bedingt. Durch einen Umstieg auf IP-Telefone kann hier eine Konvergenz geschaffen werden, die dazu führt, dass nur noch ein Netz gewartet werden muß, und so zusätzliche Kosten einspart.

Im Rahmen dieser Arbeit wird untersucht, auf welche Art und Weise eine solche Einbindung von IP-Telefonen in das Überwachungssystem stattfinden kann. In einem ersten Schritt wird dafür zunächst die Hardware-Infrastruktur aufgebaut und rudimentär konfiguriert. Anschließend wird getestet, ob die IP-Telefone überhaupt die technischen Voraussetzungen mitbringen, die notwendigen Informationen in gewünschter Art und Weise darzustellen. Sollte dies gelingen, wird ein Konzept erarbeitet, wie dieses neue Frontend am sinnvollsten in das bestehende Softwareprodukt integriert werden kann, um möglichst große Flexibilität zu gewährleisten.

2. technische Grundlagen

2.1. SCADA-Grundlagen

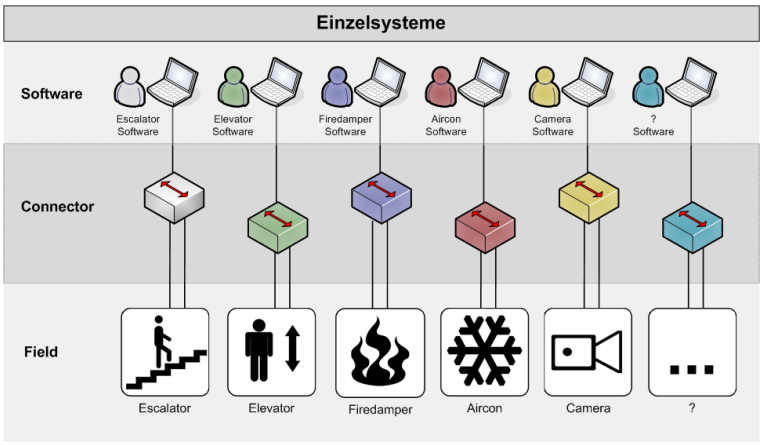


Abbildung 2.1: Steuerung von Einzelsystemen

Die PC-gestützte Steuerung von Anlagen auf herkömmliche Art und Weise funktioniert intuitiv. Die Anlage wird über ein bestimmtes Medium, z.B. RS-232, RS-485, LON, mit einem PC verbunden und von dort mit einer proprietären Software gesteuert und überwacht (Abbildung 2.1).

Dieses System macht jedoch nicht nur durch oftmals große Vielfalt von Anlagentypen und -herstellern Probleme, sondern auch wegen der räumlichen Trennung einer Anlage und des Leitstandes.

SCADA-Systeme wurden geschaffen, um das Frontend für alle zu steuernden Anlagen zu vereinheitlichen (Abbildung 2.2). SCADA steht als Abkürzung für „Supervisory Control and Data Acquisition“.

SCADA-Systeme sind üblicherweise hierarchisch aufgebaut. Auf unterster Ebene stehen hierbei die zu steuernden bzw. zu kontrollierenden Anlagen. Direkt über ihnen befinden sich die sogenannten RTUs (Remote Terminal Units). Diese nehmen die Ist-Werte von den Anlagen auf. Sollten diese Ist-Werte zu einem bestimmten Maß von den Soll-Werten abweichen, können RTUs ggf. selbstständig einschreiten, ohne dass die dritte Ebene berührt wird. Diese dritte Ebene stellt das zentrale System dar, über das auch das HMI (Human-Machine-Interface bzw. Mensch-Maschine-Schnittstelle) angebunden ist. Von hier aus kann z.B. der Status der Anlagen visualisiert oder Soll-Werte angepasst werden.[Wik08c]

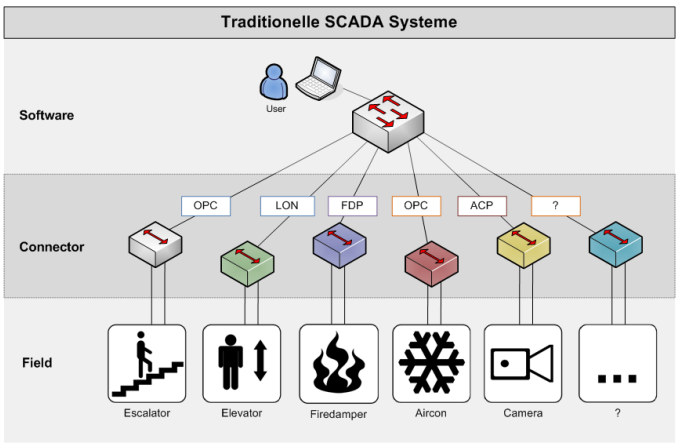


Abbildung 2.2: Steuerung von Anlagen über ein SCADA-System

Trotz der Vereinheitlichung des Frontends muss jedes SCADA-System allerdings immer noch jedes einzelne Hardware- und Softwareprotokoll der beteiligten Anlagen beherrschen. Dadurch wird nicht nur ein Erweitern des Systems um neue Anlagen, sondern sogar der Einsatz der gleichen Produkttypen anderer Hersteller stark erschwert.

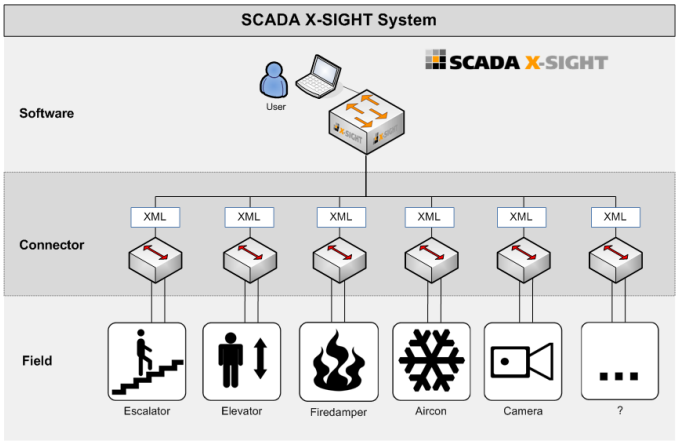


Abbildung 2.3: Steuerung von Anlagen über ein FCML-SCADA-System

Um diese Abhängigkeit aus dem Weg zu räumen, wurde SCADA X-SIGHT entwickelt. Hier geschieht die Kommunikation vom SCADA-System zum jeweiligen Connector über Ethernet, IP und XML (Abbildung 2.3).

Der Einsatz von IP-Technologie ermöglicht eine Konvergenz mit bestehenden IP-Netzen und damit eine beliebig große räumliche Trennung von Connector und SCADA-System.

Der verwendete XML-Dialekt ist mit FCML (Facility Control Markup Language) ein offener Standard, der über alle Gerätetypen und -hersteller derselbe ist.

2.2. FCML

Die FCML (Facility Control Markup Language) wurde entwickelt, um eine einheitliche Sprache zur Überwachung und Steuerung beliebiger Anlagentypen bereitzustellen. Es handelt sich bei FCML um eine Familie von XML-Dialekten.

Dabei ist die Grundspezifikation, der „core“, geräteunabhängig gehalten und schreibt nicht vor, welche Art von Daten zu übertragen ist oder welche Befehle unterstüzt werden. Diese Spezialisierung erfolgt in eigenen Unterformaten für jeden Gerätetyp.

Die FCML wird von der FCML-Group verabschiedet und weiterentwickelt, die sich aus IT-Dienstleistern, Anlagenherstellern, Anwendern sowie Teilnehmern aus Forschung und Lehre zusammensetzt.

Geräte, die FCML-konform sind, haben ein HTTP-Interface, über das mit GET- und POST-Requests Daten abgefragt sowie Kommandos gegeben werden können. Ist eine Anlage selbst nicht FCML-konform, so muß ein FCML-Interface zwischengeschaltet werden, dass die proprietären Formate in FCML überträgt und umgekehrt.

Weiterführende Informationen zur FCML finden sich in [BLO+08] und in der FCML-Spezifikation [RB07].

2.3. SCADA X-SIGHT

X-SIGHT ist ein webbasiertes SCADA-Produkt, das in der Kommunikation mit den angeschlossenen Anlagen ausschließlich auf FCML und damit auf offenen Protokollstandards aufsetzt.

Es basiert auf dem Spring MVC Framework[Spr08a]. Dieses Framework zeichnet sich durch seine Modularität und sein breites Spektrum an Funktionalität aus. Spring bietet auch zahlreiche Schnittstellen und Hilfestellungen an, andere Frameworks in die Spring-Entwicklung einzubeziehen. Im Falle von SCADA X-SIGHT sind dies vor allem *Apache Tapestry*[Apa08a] und *Apache Velocity*[Apa08b] als MVC-Frameworks für die Frontend-Schicht und *Hibernate*[Red08] als Persistenz-Framework zwischen DAO-Schicht und Datenbank.

2.3.1. Schichtmodell

X-SIGHT benutzt das Model-View-Controller-Muster. In diesem Architekturmuster werden die Programmfunktionen in drei Bereiche geteilt (Abbildung 2.4).[1]

[1] [Wik08b]

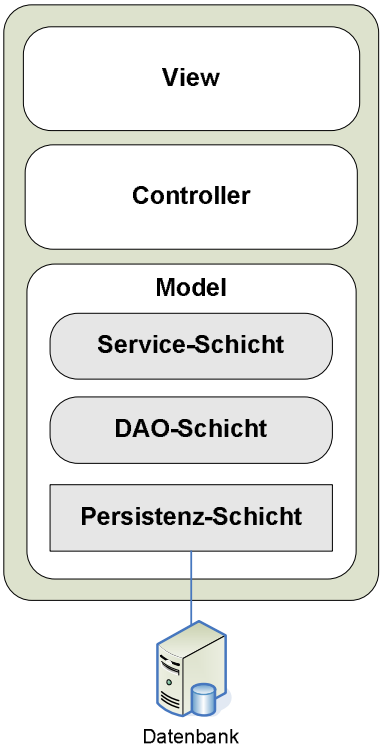


Abbildung 2.4: Schichten von X-SIGHT

View (= Präsentation): Diese Schicht ist verantwortlich für die Darstellung der Daten aus dem Modell und die Entgegennahme von Benutzeraktionen. Diese verarbeitet sie allerdings nicht selbst, sondern reicht sie weiter an den Controller. Im Modul Webcenter wird als View *Tapestry* eingesetzt, die Module Proxy und Cisco benutzen *Velocity*.

Controller (= Steuerung): Der Controller verwaltet die Views, nimmt Benutzeraktionen von ihnen entgegen, wertet sie aus und veranlasst evtl. Änderungen im Model.

Model (= Modell): Diese Schicht repräsentiert die Daten in der Anwendung. In X-SIGHT ist diese Schicht noch einmal aufgeteilt in Service-, DAO- und Persistenz-Schicht. Die **Service-Schicht** stellt die Geschäftslogik der Anwendung bereit. In der **Data Access Objects (DAO)-Schicht** wird der Zugriff auf die Persistenzschicht gekapselt. Die **Persistenz-Schicht** ist für die eigentlichen Anfragen an die Datenbank verantwortlich. Diese Aufgabe übernimmt in X-SIGHT das Framework *Hibernate*.

2.3.2. Module

X-SIGHT ist als Projekt modular aufgebaut. Es gibt verschiedene Arten von Modulen:

- unterstützende Module, die keinen Java-Code enthalten
- Module, die Bibliotheken als Dependencies zur Verfügung stellen
- (Web-)Anwendungsmodule, die definieren, welche Funktion das Kompilat übernimmt Jedes Kompilat enthält immer nur einer Anwendungsmodul

Abbildung 2.5 zeigt schematisch den funktionalen Zusammenhang zwischen den einzelnen Modulen.

Modul-Name	Typ	Funktion
xsight_build	unterstützend	Stellt das Standard-Ant-Skript zur Verfügung, mit dessen Hilfe alle Module kompiliert werden.
xsight_common	dependency	Hier sind nahezu alle Klassen der DAO- und Serviceschicht definiert

xsight_connector		Für jeden Anlagentyp wird im Proxy ein Connector instanziiert, die Kommunikation mit Anlagen dieses Typs übernimmt
xsight_connector_api	dependency	Stellt die API bereit, über die der Proxy auf die Connector-Klassen zugreift
xsight_core	Anwendung	Das Herz von X-SIGHT. Der Core bekommt die FCML-Dokumente bei Änderungen vom Proxy geliefert, verarbeitet diese und legt sie in der Datenbank ab. Zudem nimmt er manuelle Steuerbefehle vom Webcenter entgegen. Über den Core laufen auch zeitgesteuerte Jobs und die Überwachung der Proxy-Verfügbarkeit. Er läuft zwar als Webanwendung, hat aber kein Frontend, sondern stellt nur die HTTP-Remoting-Schnittstelle zur Verfügung.
xsight_core_api	dependency	Stellt die API bereit, in der die Remoting-Schnittstellen definiert sind, die der Proxy und das Webcenter zum Zugriff auf den Core benötigen.
xsight_database	unterstützend	Stellt Skripte zur Verfügung, die eine vollständige Demo-Datenbank aufbauen
xsight_documentation	unterstützend	beinhaltet die automatisch generierte JavaDoc-HTML-Dokumentation des Projektes
xsight_proxy	Web-Anwendung	Der Proxy sitzt zwischen den Anlagen bzw. den Konnectoren und dem Core. Er prüft regelmäßig, ob sich der Zustand seiner untergeordneten Anlagen geändert hat und schickt in diesem Fall das neue Dokument an den Core. In der Gegenrichtung leitet er Steuerbefehle vom Core an die Anlagen weiter.
xsight_proxy_api	dependency	Stellt die API bereit, in der die Remoting-Schnittstellen definiert sind, die der Core zum Zugriff auf den Proxy benötigt.
xsight_webcenter	Web-Anwendung	Das Webcenter stellt das Interface/Frontend zum Benutzer dar. Hierüber wird das System konfiguriert, Informationen abgerufen, Befehle an die Anlagen versandt etc.

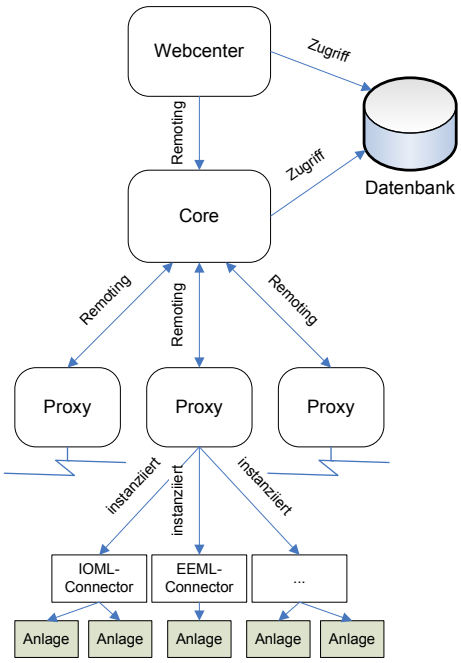


Abbildung 2.5: Zusammenhang zwischen den X-SIGHT-Modulen

2.4. Beschreibung der Cisco-Geräte

Im Folgenden sind die wichtigsten technischen Daten der einzelnen eingesetzten Cisco-Geräte aufgeführt.

2.4.1. Tisch-Telefon 7971G-GE

Betriebstemperatur	0 - 40° C
Luftfeuchtigkeit	10 - 95%
Maße (BxHxT)	~ 27,5 x 23 x 9 cm (mit Wandhalterung)
Display-Auflösung	320 x 234 Pixel, 12 bit Farbdisplay mit 4096 Farben
belegbare Softkeys	5
darstellbarer Zeichensatz	laut Spezifikation Latin-1, in Tests nur 7-bit ASCII

Abbildung 2.6: Cisco 7971G-GE

2.4.2. Tisch-Telefon 7961G-GE

Betriebstemperatur	0 - 40° C
Luftfeuchtigkeit	10 - 95%
Maße (BxHxT)	~ 20 x 27 x 15 cm
Display-Auflösung	320 x 222 Pixel, 4 bit Graustufen
belegbare Softkeys	4
darstellbarer Zeichensatz	laut Spezifikation Latin-1, in Tests nur 7-bit ASCII

Abbildung 2.7: Cisco 7961G-GE

2.4.3. WLAN-Handset 7921G

Betriebstemperatur	0 - 40° C
Luftfeuchtigkeit	10 - 95%
Maße (BxHxT)	~ 5,3 x 12,7 x 2,5 cm
Display-Auflösung	220 x 176 Pixel, Farbe
belegbare Softkeys	2
darstellbarer Zeichensatz	Latin-1
WLAN-Standard	802.11a/b/g
Verschlüsselung	WEP / TKIP / AES
Authentifizierung	LEAP / IEEE 802.1X / EAP-FAST / WPA 1/2 Personal, Enterprise / CCKM

Abbildung 2.8: Cisco 7921G

2.4.4. Cisco 2811 Integrated Services Router

- durch den integrierten Call Manager Express unterstützt der Router bis zu 36 IP-Telefone
- 2 Ethernet-Ports
- 4 Slots für WAN-Interface Module

Abbildung 2.9: Cisco 2811

2.4.5. Cisco Aironet 1240G Access Point

Betriebstemperatur	-20 - 55° C
Luftfeuchtigkeit	10 - 95%
Maße (BxHxT)	˜ 2,8 x 16,8 x 21,6 cm
WLAN-Standard	802.11 b/g
Verschlüsselung	AES CCMP (WPA2) / TKIP (WPA) / Cisco TKIP / WPA TKIP / WEP
Authentifizierung	WPA 1/2 / TKIP / MIC / WEP / EAP

Abbildung 2.10: Cisco Aironet AP1240G

2.4.6. CallManager Simulator

Im Cisco-SDK ist ein Programm namens „CallManager Simulator“ enthalten.

Dieses stellt einen TFTP-Server und einen CallManager mit gerade genug Funktionalität bereit, um die Telefone lauffähig zu machen sowie die diversen URLs zu konfigurieren. Damit kann man eine schlanke Testumgebung aufbauen, ohne sich um die Feinheiten eins „echten“ CallManager bzw. CallManager Express kümmern zu müssen. Telefonie ist dann zwar nicht möglich, dies wird jedoch für den Testaufbau auch nicht benötigt.

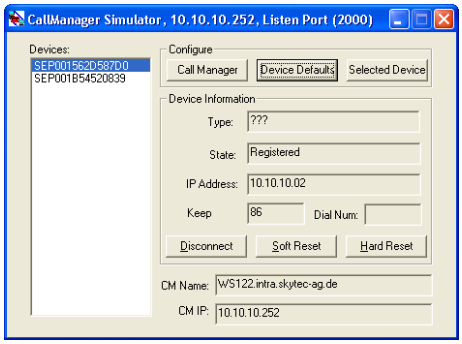


Abbildung 2.11: Der CallManager Simulator

3. Aufbau des Testbeds - Hardware

3.1. Physikalischer Aufbau

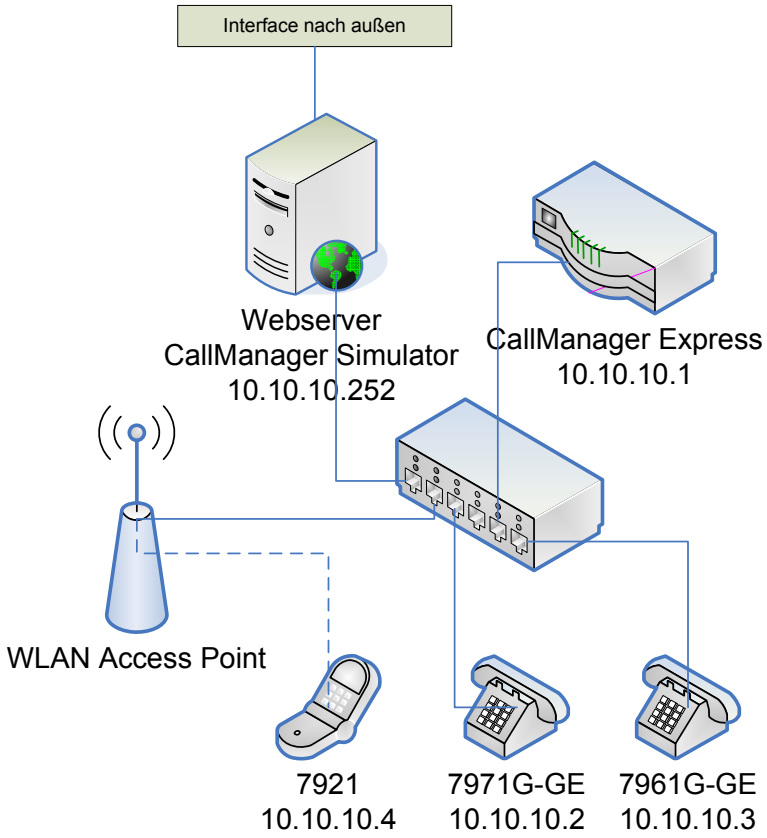


Abbildung 3.1: Physikalischer Aufbau der Gerätelandschaft

Das VoIP-Netz wurde getrennt vom Hausnetz der SKYTEC AG aufgebaut. Über einen Switch sind alle beteiligten Komponenten (Telefone 7971G-GE und 7961G-GE, W-LAN-Access-Point Aironet 1240G, Router 2811 ISR, „telefonnetzseitige“ NIC des Servers) miteinander verbunden.

Das Handset 7921 ist über den Access-Point, der als Bridge konfiguriert ist, in das VoIP-Netz eingebunden.

Die „externe“ NIC des Servers ist mit dem Hausnetz verbunden, über das auch die Verbindung zum X-SIGHT-Testsystem besteht.

Alle Geräte im Telefonnetz liegen im IP-Subnetz 10.10.10.0/24. Da die Telefone lediglich Zugriff auf den Webserver benötigen, ist auf diesem keinerlei Routing ins Hausnetz erforderlich. Damit bleibt eine Trennung der Netze gewährleistet.

Als Webserver läuft ein Apache Tomcat 5.5, der als Servlet-Container für die zu entwickelnde Anwendung dient. Daneben steht hier für den Betrieb ohne Hardware-Router auch der CallManager-Simulator bereit.

3.2. Konfiguration der Telefone

Die Telefone werden standardmäßig per DHCP und TFTP mit allen notwendigen Konfigurationsdaten versorgt. Alternativ kann aber die IP-Konfiguration auch per Hand vorgenommen werden. Ein TFTP-Server wird allerdings in jedem Fall benötigt.

Sämtliche andere Betriebsparameter erhalten die Telefone über die Konfigurationsdatei, die vom TFTP-Server ausgeliefert wird. Dort sind z.B. die IP-Adresse des Callmanagers, die Konfiguration bestimmter URLs, Locales etc. zu finden.

3.2.1. DHCP-Betrieb

Sollte der DHCP-Betrieb nicht aktiviert sein, müssen folgende Schritte ausgeführt werden.

Zuerst wird das „Settings“-Menü aufgerufen. Hier kann man nun durch Drücken der Tastenfolge **# die Einstellungen zum Verändern freigeben.

Im Folgenden wählt man nun Punkt 2 („Network Configuration“). Punkt 22 („DHCP Enabled“) wird markiert und durch den Softkey „Yes“ aktiviert.

Durch den Softkey „Save“ wird die Konfiguration übernommen und das Telefon beginnt mit der Suche des DHCP-Servers.

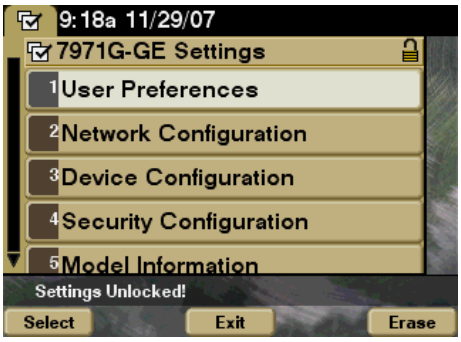


Abbildung 3.2: Das Settings-Menü

Wird statt des Cisco-Routers ein anderer DHCP-Server verwendet, so ist unbedingt darauf zu achten, dass dieser im Option-Feld 150 die IP-Adresse des TFTP-Servers mitteilt **oder** Punkt 24 in der Netzwerkkonfiguration („Alternate TFTP“) muß aktiviert werden, so dass über Punkt 8 („TFTP Server 1“) die TFTP-IP manuell eingestellt werden kann.

3.2.2. Betrieb mit manueller IP-Konfiguration

Sollen die IP-Parameter per Hand zugewiesen werden, müssen mindestens die folgendene Parameter konfiguriert werden:

- Punkt 6: IP Address
- Punkt 7: Subnet Mask

- Punkt 8: TFTP Server 1
- Punkt 10: Default Router 1

Achtung! Ohne Angabe des Default Routers ist das Speichern der Einstellungen zwar möglich, jedoch verwirft das Telefon die IP-Adresse und die Subnetzmaske sofort wieder. Dies scheint in der vorliegenden Version ein Bug zu sein.

3.3. Konfiguration des Routers

Die Konfiguration kann man in mehrere Schritte teilen:

- Grundkonfiguration mit Hilfe des Assistenten
- Grundkonfiguration über den Assistenten hinaus
- Grundkonfiguration des CallManager Express
- Einrichtung der einzelnen Voice-Endgeräte (ephones)

3.3.1. Grundkonfiguration mit Hilfe des Assistenten

Nach dem Einschalten des Routers erscheinen auf der Console die Meldungen, die das Gerät beim Hochfahren ausgibt. Nach einiger Zeit ist der Boot-Vorgang beendet und das Gerät ist bereit, in den Setup-Modus zu wechseln.

```
--- System Configuration Dialog ---

Would you like to enter the initial configuration dialog? [yes/no]:
```

Nach beantworten dieses Frage mit „yes“ beginnt ein geführter Setup-Prozess, der im wesentlichen aus vielen einzelnen Fragen besteht.

```
At any point you may enter a question mark '?' for help.
Use ctrl-c to abort configuration dialog at any prompt.
Default settings are in square brackets '[]'.

Basic management setup configures only enough connectivity
for management of the system, extended setup will ask you
to configure each interface on the system

Would you like to enter basic management setup? [yes/no]: no

First, would you like to see the current interface summary? [yes]:


Any interface listed with OK? value "NO" does not have a valid configuration

Interface                  IP-Address      OK? Method Status                Prot
ocol
FastEthernet0/0            unassigned      NO  unset  up                    up

FastEthernet0/1            unassigned      NO  unset  up                    down

BRI0/0/0                   unassigned      YES unset  down                  down

BRI0/0/0:1                 unassigned      YES unset  down                  down
```

```
BRI0/0/0:2                 unassigned      YES unset  down                    down

BRI0/0/1                   unassigned      YES unset  down                    down

BRI0/0/1:1                 unassigned      YES unset  down                    down

BRI0/0/1:2                 unassigned      YES unset  down                    down


Configuring global parameters:

  Enter host name [Router]:
```

Setzt den Hostname des Gerätes auf „Router“.

```
  The enable secret is a password used to protect access to
  privileged EXEC and configuration modes. This password, after
  entered, becomes encrypted in the configuration.
  Enter enable secret: router
```

Setzt das Passwort für den „Enable“-Mode (Konfigurationsmodus) auf „router“.

```
  The enable password is used when you do not specify an
  enable secret password, with some older software versions, and
  some boot images.
  Enter enable password: routerpassword
  The virtual terminal password is used to protect
  access to the router over a network interface.
  Enter virtual terminal password: routervirtual
```

Setzt das Passwort, das z.B. für den Telnet-Zugang benötigt wird.

```
  Configure SNMP Network Management? [yes]:
    Community string [public]:
Configure IP? [yes]:
    Configure RIP routing? [yes]: no
Configure CLNS? [no]:
  Configure bridging? [no]:

  BRI interface needs isdn switch-type to be configured
  Valid switch types are :
               [0]  none.....Only if you don't want to configure BRI.
               [1]  basic-1tr6....1TR6 switch type for Germany
               [2]  basic-5ess....AT&T 5ESS switch type for the US/Canada
               [3]  basic-dms100..Northern DMS-100 switch type for US/Canada
               [4]  basic-net3....NET3 switch type for UK and Europe
               [5]  basic-ni......National ISDN switch type
               [6]  basic-qsig....QSIG switch type
               [7]  basic-ts013...TS013 switch type for Australia
               [8]  ntt...........NTT switch type for Japan
               [9]  vn3...........VN3 and VN4 switch types for France
  Choose ISDN BRI Switch Type [2]: 0
```

BRI (Basic Rate Interface) ist der ISDN-Basisanschluß. Da in der Testkonfiguration keine externer Anschluß benötigt wird, wird hier auch kein BRI konfiguriert.

```
Configuring interface parameters:

Do you want to configure FastEthernet0/0  interface? [yes]:
  Use the 100 Base-TX (RJ-45) connector? [yes]:
  Operate in full-duplex mode? [no]: yes
```

```
  Configure IP on this interface? [yes]:
    IP address for this interface: 10.10.10.1
    Subnet mask for this interface [255.0.0.0] : 255.255.255.0
    Class A network is 10.0.0.0, 24 subnet bits; mask is /24
```

Die IP-Adresse des Routers wird auf 10.10.10.1 in einem /24er Subnetz festgelegt.

```
Do you want to configure FastEthernet0/1  interface? [yes]: no

Would you like to go through AutoSecure configuration? [yes]: no
```

Um den Test zu vereinfachen, wird AutoSecure nicht ausgeführt. Im Zuge dieser Konfiguration würden sonst eine große Anzahl (vermeintlich) nicht benötigter Services abgeschaltet, um die Router-Konfiguration sicherer zu machen.

```
AutoSecure dialog can be started later using "auto secure" CLI

The following configuration command script was created:

hostname Router
enable secret 5 $1$zzeI\$JtGfYEcKYIBiUMOl4I2QD.
enable password routerpassword
line vty 0 4
password routervirtual
snmp-server community public
!
ip routing
no clns routing
no bridge 1
isdn switch-type  none
!
interface FastEthernet0/0
media-type 100BaseX
full-duplex
ip address 10.10.10.1 255.255.255.0
no mop enabled
!
interface FastEthernet0/1
shutdown
no ip address
dialer-list 1 protocol ip permit
!
end


[0] Go to the IOS command prompt without saving this config.
[1] Return back to the setup without saving this config.
[2] Save this configuration to nvram and exit.

Enter your selection [2]: 2
media-type 100BaseX
  ^
% Invalid input detected at '^' marker.
```

Interessanterweise scheint der Router das Script, das vom Konfigurationstool erstellt wird, nicht 100%ig zu verstehen. Da es sich hier aber nur um das Festlegen des Ethernet-Medientypen handelt, ist dies nicht von kritischer Bedeutung.

```
Building configuration...
[OK] Use the enabled mode 'configure' command to modify this configuration.
```

```
Press RETURN to get started!
```

Hiermit ist die Grundkonfiguration abgeschlossen.

3.3.2. Grundkonfiguration über den Assistenten hinaus

Um eine Konfiguration am Gerät vornehmen zu können, muss zuerst in den priveligierten „EXEC“-Modus gewechselt werden.

```
Router>enable
Password:
Router#
```

Der „EXEC“-Modus ist an der Raute # nach dem Hostnamen zu erkennen. Nun kann der Konfigurationsmodus aufgerufen werden:

```
Router#configure terminal
Router(config)#
```

Der Parameter „terminal“ bedeutet dabei, dass die Konfiguration direkt über die Konsole vorgenommen wird. Alternativen wären „memory“ (liest die Konfiguration aus einer Datei im Speicher) oder „network“ (liest die Datei von einem Host im Netzwerk).

Nun kann mit der eigentlichen Konfiguration begonnen werden.

Der einzige Punkt, der am Router noch zu erledigen ist, ist die Konfiguration des DHCP-Servers. Dies geschieht in folgenden Schritten:

```
Router(config)#ip dhcp pool dhcppool
Router(dhcp-config)#network 10.10.10.0 255.255.255.0
Router(dhcp-config)#option 150 ip 10.10.10.1
Router(dhcp-config)#default-router 10.10.10.1
Router(dhcp-config)#end
Router#
*Oct 25 15:06:27.595: %SYS-5-CONFIG_I: Configured from console by console
```

Über den Befehl „option 150 ip 10.10.10.1“ wird dem DHCP-Server mitgeteilt, dass er das nicht-standardisierte Option-Feld 150 an alle anfragenden Geräte übermitteln soll. In diesem Feld wird Cisco-Geräten der TFTP-Server mitgeteilt, auf dem sich Konfigurationsdateien, Firmware-Images u.ä. befinden.

Hier kann statt der IP des Routers auch eine andere Maschine angegeben werden. Dort muß ein TFTP-Server laufen, der Dateien mit Namen `SEP<MAC>.cnf.xml` bereitstellt, also z.B. SEP001562D587D0.cnf.xml.

3.3.3. Grundkonfiguration des CallManager Express

Alle Konfigurationen in diesem Abschnitt können auch in einer einzigen Konfigurationssitzung durchgeführt werden.

3.3.4. Einstellen der CallManager-IP-Adresse

Es ist notwendig, die IP-Adresse des von den Telefonen zu kontaktierenden CallManagers zu konfigurieren. Diese IP-Adresse wird in die Konfigurationsdateien geschrieben, die beim Booten von den Telefonen verwendet werden.

```
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#telephony-service
Router(config-telephony)#ip source-address 10.10.10.1
Router(config-telephony)#end
Router#
*Oct 25 15:08:40.527: %SYS-5-CONFIG_I: Configured from console by console
```

3.3.5. Aktivieren der Konfigurationsdateien

Nun muß das Schreiben von Konfigurationsdateien aktiviert werden. Ohne diese Dateien wären die Endgeräte nicht in der Lage, den CallManager zu kontaktieren.

```
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#telephony-service
Router(config-telephony)#create cnf-files
Creating CNF files
CNF-FILES: Clock is not set or synchronized,
                retaining old versionStamps

Router(config-telephony)#end
Router#
*Oct 25 15:08:40.527: %SYS-5-CONFIG_I: Configured from console by console
```

3.3.6. Einstellen der maximalen Anzahl von Verzeichniseinträgen und Telefonen

Dem CallManager muß mitgeteilt werden, wieviele Telefone (ephones) und Verzeichniseinträge (dns) im System existieren können sollen.

```
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#telephony-service
Router(config-telephony)#max-ephones ?
  <1-42>  Maximum phones to support

Router(config-telephony)#max-ephones 42
Router(config-telephony)#max-dn ?
  <1-144>  Maximum directory numbers supported
Router(config-telephony)#max-dn 144
Router(config-telephony)#end
Router#
```

Über die Kontext-Hilfe zeigt der CallManager an, wie viele Telefone und Einträge er unterstüzt.

3.3.7. Konfiguration eines Telefons

Zuerst wird ein Verzeichniseintrag angelegt. In diesem wird immer ein Durchwahl-Namen-Paar gespeichert.

```
Router(config)#ephone-dn <dn-id> dual-line
Router(config-ephone-dn)#number <durchwahl>
Router(config-ephone-dn)#name <name>
Router(config-ephone-dn)#exit
Router(config)#
```

Der Parameter „dual-line“ bestimmt, dass diesem Verzeichniseintrag zwei „Leitungen“ zugeordnet werden. Erst damit werden Leistungsmerkmale wie z.B. Anklopfen oder Makeln möglich.

Anschließend muß ein Telefon angelegt und diesem Telefon ein Verzeichniseintrag zugeordnet werden.

```
Router(config)#ephone <ephone-id>
Router(config-ephone)#mac-address <mac-adresse>
Router(config-ephone)#type <telefon-typ>
Router(config-ephone)#button <button>:<dn-id>
Router(config-ephone)#exit
Router(config)#
```

Die Angabe der Mac-Adresse ist notwendig, um die eindeutige Zuordnung zwischen dem physischen und dem logischen Gerät herzustellen. Dabei ist darauf zu achten, dass der CallManager die Eingabe in der Form xxxx.yyyy.zzzz erwartet, aus der Mac-Adresse 001562D587D0 wird also 0015.62D5.87D0.

Der Telefontyp muß angegeben werden, damit der CallManager die Fähigkeiten des Telefons kennt. Durch Eingabe von „type ?“ erhält man eine Liste aller unterstützten Modelle.

Über den Befehl „button“ wird einem „Leitungsknopf“ am Gerät ein Verzeichniseintrag zugeordnet. Erst wenn einem Gerät mindestens ein Eintrag zugeordnet wurde, kann es telefonieren bzw. angerufen werden.

3.3.8. Konfiguration der Dienst-URLs

Die dienst URLs lassen sich folgendermaßen konfigurieren:

```
Router>enable
Password:
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#telephony-service
Router(config-telephony)#url ?
  authentication  authentication url
  directories     directories url
  idle            idle url
  information     information url
  messages        messages url
  proxy-server    proxy-server url
  services        services url
```

Diese bedeuten im Einzelnen:

- **Authentication URL** Mithilfe dieser URL überprüft das Telefon Benutzernamen und Passwort bei Anfragen über das integrierte HTTP-Interface, die einer Authentifizierung bedürfen. Dies sind die HTTP-Push-Schnittstelle sowie das Auslesen eines Screenshots.
- **Directories URL** Stellt zusätzliche Verzeichnisse bereit, die über den „Directories“-Button abgerufen werden können.
- **Messages URL** Stellt verschiedene Messaging-Dienste bereit, z.B. auch VoiceMail
- **Idle URL** Definiert, welche URL nach einer bestimmten Leerlaufzeit, die als erster Parameter übergeben wird, automatisch auf dem Bildschirm des Telefons angezeigt wird.
- **Information URL** Stellt eine Kontexthilfe über den ?-Button bereit
- **Proxy Server URL** Stellt einen HTTP-Proxy bereit.
- **Services URL** Stellt die Übersichtsseite bereit, die beim Starten des „Services“-Moduls geladen wird

3.4. Konfiguration des Access Points

3.4.1. Cisco IOS installieren

Der ausgelieferte Access Point beherbergte zuerst eine sogenannte „LWAPP“-Firmware. LWAPP steht für „Lightweight Access Point Protocol“. Diese Firmware ermöglicht eine Konfiguration vieler Access Points über einen zentralen Access Point Controller.

Da uns jedoch kein solcher Controller zur Verfügung steht, mußte die Firmware gegen eine normale IOS (Internetwork Operation System) Firmware getauscht werden.

Um den Access Point in den korrekten Modus für das Firmware-Upgrade zu versetzen, muß nach dem Einstecken der Stromversorgung die „Mode“-Taste an der Rückseite solange gedrückt gehalten werden, bis sich die „Status“-LED violett verfärbt. Anschließend versucht der Access Point mit der Quell-IP-Adresse 10.0.0.1, von einem TFTP-Server in seinem Subnetz eine Datei namens „c1240-k9w7-tar.default“ herunterzuladen. Gelingt ihm dies, entpackt er sie, überprüft, ob es sich um eine kompatible Firmware handelt und installiert sie. Der ganze Vorgang dauert in etwa sechs Minuten.

3.4.2. Zuweisen einer IP-Adresse

Die Vergabe einer IP-Adresse kann entweder per serieller Console oder per DHCP-Server erfolgen.

Ich habe mich für den Weg per Console entschieden.

Wie beim Router auch, muß zuerst der „privileged exec mode“ aktiviert werden.

```
ap>enable
Password:
ap#
```

Das Standard-Passwort ist hier „Cisco“.

Anschließend kann über den Configure-Modus eine IP-Adresse gesetzt werden.

```
ap#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ap(config)#interface bvi1
ap(config-if)#ip address 10.10.10.240 255.255.255.0
ap(config-if)#^Z
ap#
```

3.4.3. Konfiguration der WLAN-Einstellungen

Die weitere Konfiguration des AP kann über die Console oder per Webinterface erfolgen. Das Webinterface scheint mit Mozilla Firefox nicht vollständig kompatibel zu sein, da an verschiedenen Stellen Buttons fehlen, JavaScript-Fehler auftreten, etc. Durch Benutzung des Microsoft Internet Explorers sind die Fehler nicht mehr aufgetreten.

Hier wird die Konfiguration wiederum per Console vorgenommen.

Als erster Schritt wird eine SSID festgelegt.

```
ap(config)#dot11 ssid cisco
ap(config-ssid)#authentication open
ap(config-ssid)#guest-mode
ap(config-ssid)#exit
```

Diese Befehle legen eine SSID mit Namen „cisco“ an, definieren „offene“ Authentifizerung und aktivieren den „guest mode“, der bewirkt, dass die SSID vom Access Point per Broadcast versendet wird.

Nun kann die Verschlüsselung konfiguriert werden.

```
ap(config)#interface Dot11Radio0
ap(config-if)#encryption key 1 size 40bit 0 636973636f transmit-key
ap(config-if)#encryption mode wep mandatory
ap(config-if)#ssid cisco
ap(config-if)#no shutdown
ap(config-if)#exit
```

Obwohl es möglich ist, mehrere SSIDs pro Interface festzulegen, müssen alle SSIDs, die auf demselben Interface liegen, dieselbe Verschlüsselung nutzen.

Aus diesem Grund geschieht die Konfiguration der Sicherheitseinstellungen im „interface“-Menü des WLAN-Adapters.

Zeile 2 legt den WEP-Schlüssel Nr. 1 als 40bit-Schlüssel fest. Der Zusatz „transmit-key“ bewirkt, dass dieser Schlüssel zum Senden verwendet wird. Zeile 3 schaltet den Verschlüsselungs-Modus auf WEP und markiert ihn als „notwendig“.

In Zeile 4 wird die oben konfigurierte SSID dem Interface zugewiesen.

Zeile 5 aktiviert das Interface.

3.5. Konfiguration des CallManager Simulator

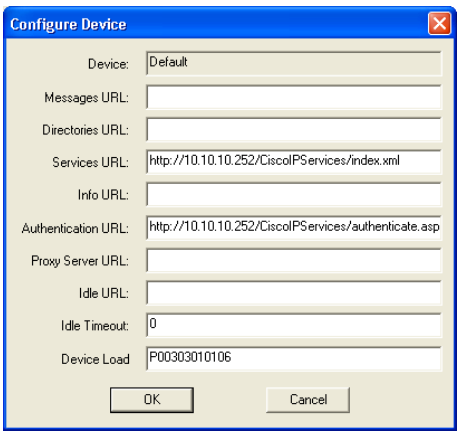


Abbildung 3.3: URL-Konfiguration

Der CallManager Simulator bedarf außer der Konfiguration der Dienst-URLs keiner weiteren Einstellungen (Abbildung 3.3).

4. Software-Implementierung des Testbeds

4.1. Anbindung an den X-SIGHT-Proxy

4.1.1. Konzept

Im ersten Schritt sollte eine einfache Lösung gefunden werden, die visualisiert, dass eine Überwachung und Steuerung der Anlagen mithilfe eines IP-Telefones überhaupt machbar ist.

Folgendes Konzept wurde hierfür geschaffen:

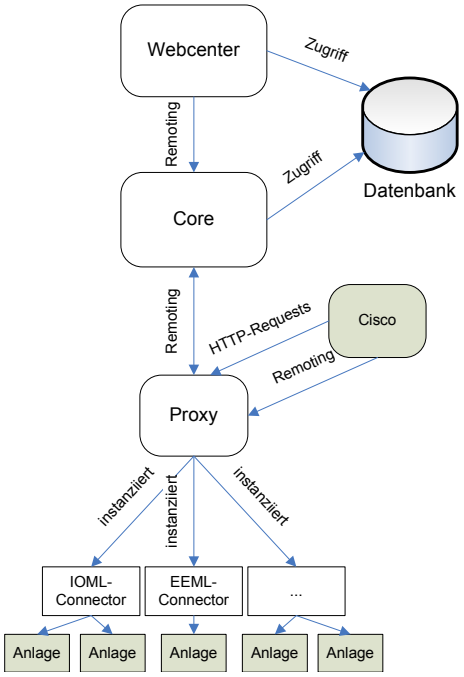


Abbildung 4.1: Anbindung an den X-SIGHT-Proxy

Die Applikation läuft auf einem Apache Tomcat direkt am Standort des Proxys, möglicherweise sogar auf derselben physischen Maschine oder in derselben Tomcat-Instanz. Sie kommuniziert ausschließlich mit dem Proxy.

Informationen werden abgerufen, indem vom Proxy über einfache HTTP-GET-Requests das FCML-Dokument der jeweiligen Anlage abgerufen und ausgewertet wird. Steuerbefehle werden über das Spring-Remoting an den Proxy gesendet, der diese wiederum an die Anlage weiterleitet.

4.1.2. Umsetzung

Da diese Applikation in erster Linie als Proof of Concept dient, sollte eine möglichst schnelle, schlanke Entwicklung ermöglicht werden, weshalb auf umfangreiche Frameworks verzichtet wurde. Im ersten Schritt wurde diese Applikation mithilfe einfacher Java-HTTP-Servlets und JSPs implementiert.

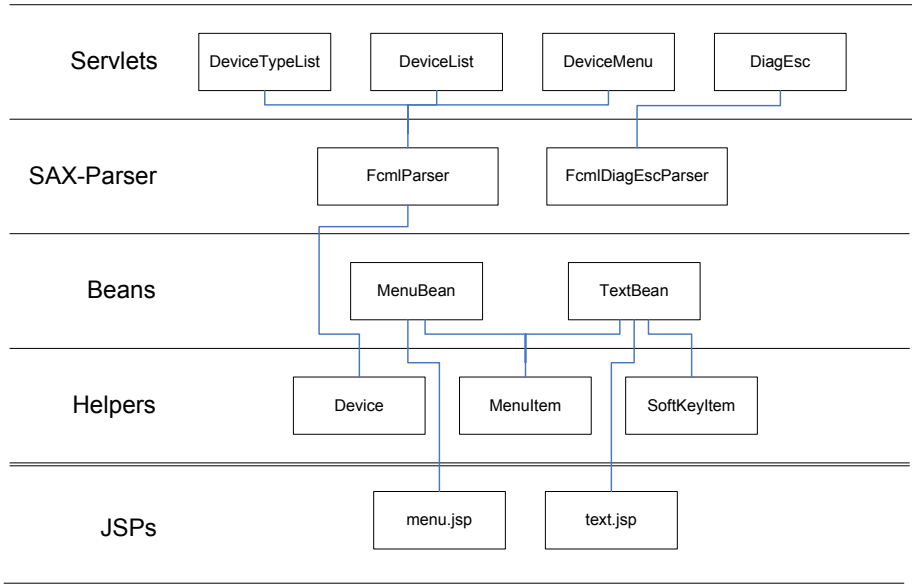


Abbildung 4.2: Verwendete Klassen/JSPs

Das Herzstück der Applikation sind die beiden SAX-Parser-Klassen, die von der Klasse `org.xml.sax.helpers.DefaultHandler` erben.

Die Klasse `FcmlParser` parst die Übersichtsseite des Proxys und stellt über die Methoden `HashMap<Integer, Device> getDevices()` und `HashMap<String, String> getDeviceTypes()` eine Liste der Anlagen und Anlagentypen zur Verfügung, die in dieser Übersichtsseite aufgelistet sind.

Die Klasse `FcmlDiagEscParser` erstellt eine textuelle Zusammenfassung der Diagnoseseite eines Aufzuges und stellt sie über die Methode `String getStatusText()` bereit. Diese Klasse steht exemplarisch für viele andere Klassen, da für jedes FCML-Schema und jede Dokumentart (Diagnose-, Stamm- oder Parameterdaten) eine eigene Klasse implementiert werden muß.

Die Resultate dieser beiden Klassen werden von vier verschiedenen HTTP-Servlet-Klassen verwendet, die alle von `javax.servlet.http.HttpServlet` erben sowie das Interface `javax.servlet.Servlet` implementieren. Die Klassen `DeviceTypeList`, `DeviceList` und `DeviceMenu` stellen jeweils eine Liste aller Anlagentypen, eine Liste aller Anlagen eines bestimmtes Typs sowie das Auswahlmenü für eine einzelne Anlage bereit. Sie verwenden die SAX-Klasse `FcmlParser`. Das Servlet `DiagEsc` zeigt die Diagnosedaten eines Aufzugs an und bedient sich dazu der Daten aus der Klasse `FcmlDiagEscParser`.

Wäre die Lösung der Proxy-Anbindung weiter verfolgt worden, wäre es zweckmäßig, abstrake Basisklassen `AbstractFcmlDiagParser`, `AbstractFcmlMasterParser` und `AbstractFcmlConfigParser` zwischen die Parserimplementierung und das Servlet einzuziehen. Da der Aufbau der Seite in diesem Fall immer gleich ist, nämlich lediglich die Darstellung von Text, kann ein Servlet für alle Basisklassen zuständig sein.

Das Remoting zum Proxy wurde in dieser Applikation nicht implementiert, da schon vorher feststand, dass das endgültige Produkt nicht an dieser Stelle ansetzen wird (siehe nächstes Kapitel).

4.1.3. Vor- und Nachteile

Die Anbindung an den Proxy hat eine Reihe von Vor- und Nachteilen:

Vorteile:

- Die Verwaltung des Systems ist verhältnismäßig einfach. Da der Proxy nur die lokal angeschlossenen Systeme „kennt“, ist es nicht erforderlich, für eine Beschränkung der auf den Telefonen sichtbaren Anlagen zu sorgen.
- Eine Steuerung der Anlagen ist auch dann möglich, wenn die Verbindung zwischen Proxy und Core unterbrochen ist.

Nachteile:

- Es wird viel Funktionalität neu implementiert, die bereits an anderer Stelle passiert. Die Auswertung der FCML-Dokumente zum Beispiel passiert eigentlich im Core, der diese Daten dann in einer leicht abrufbaren Form in der Datenbank ablegt.
- Da die Applikation nicht auf den Rest des System, wie z.B. Core oder Datenbank, zugreifen kann, ist eine spätere zentrale Konfigurierbarkeit dieser Applikation von vornherein ausgeschlossen.

4.2. Anbindung als X-SIGHT-Modul

4.2.1. Konzept

Nachdem das Konzept der Anbindung an den Proxy als nicht ausreichend verworfen wurde, sollte nun ein Konzept gefunden werden, dass den Beschränkungen nicht unterliegt und sich enger in das X-SIGHT-System integriert.

Es entstand ein Konzept, bei dem die Cisco-Applikation als vollwertiges Modul auf gleicher Ebene wie das Webcenter von X-SIGHT steht, und somit prinzipiell auch alle Möglichkeiten dessen hat.

Das Modul kann auf die komplette Datenhaltung des Systems zugreifen, hat also jederzeit den Zugriff auf vollständige Informationen, was diese Lösung sehr flexibel macht. Das Konzept beinhaltet auch die Möglichkeit der Steuerung der Anlagen mit Hilfe von Spring-Remoting-Aufrufen an das Core-Modul der Applikation. Diese sind allerdings aus sicherheitstechnischen

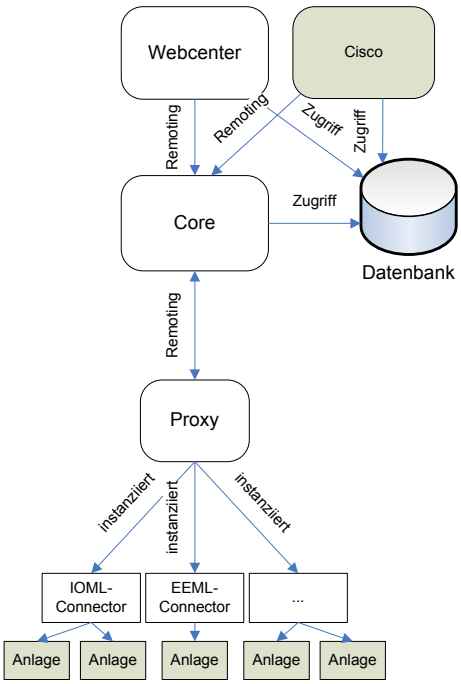


Abbildung 4.3: Anbindung als X-SIGHT-Modul

Gründen (siehe Abschnitt 5.3) in dieser Phase der Entwicklung noch nicht implementiert worden.

4.2.2. Umsetzung

Da alle X-SIGHT-Module auf dem Spring-Framework aufbauen, sollte auch das neue Modul sich daran anpassen. Im Gegensatz zu den „fertigen“ Modulen, die Spring 2.0 einsetzen, setzt das Cisco-Modul jedoch schon auf die aktuelle Version Spring 2.5.

Als Framework für die View-Schicht verwendet das Modul *Apache Velocity* [Apa08b].

Den Kern des Moduls bilden zwei Klassen sowie ein Interface (Abb. 4.4).

CiscoController

Wie am Namen zu erkennen ist, steht diese Klasse im MVC-Muster an der Stelle des Controllers.

Auf jede der sichtbaren Methoden wird eine URI gemappt, unter der diese Methode ansprechbar sein soll. Da das Projekt Spring 2.5 einsetzt, kann dies per Java-Annotations nach folgendem Muster geschehen:

```
@RequestMapping("/deviceTypeList.xml")
public String deviceTypeList(@RequestParam(value = "LOCATION", required = false) Long
    location,
        ModelMap model,
        HttpServletRequest request)
```

Diese Methoden rufen jeweils die private Methode `setFullContextPath` auf, die die absolute URL auf die Webapplikation ermittelt und sie im TransformatorBean-Objekt setzt.

Anschließend wird die Methode gleichen Namens im Transformator-Objekt aufgerufen und dieser die benötigten Parameter übergeben. Das Ergebnis dieses Aufrufs wird der ModelMap hinzugefügt. Ein Aufruf der Methode `addHelperObjects(ModelMap)` setzt zwei Hilfsobjekte in die ModelMap, die vom View benötigt werden. Dies sind Instanzen der Klassen `EscapeTool` und `UmlautTranscriber`.

Der Rückgabewert der Methoden ist ein String, der den Namen des zu verwendenden Views angibt. Für die Auflösung dieses Namens ist die viewResolver-Bean zuständig, die in der Servlet-Konfiguration festgelegt wird.

Transformator

Dieses Interface beschreibt die Methoden, die der Controller aufruft, um Daten aus dem Model in geeigneter Form zurückzuerhalten.

Methode	Beschreibung
deviceEventSummary	Gibt einen Text zurück, der alle Meldungen darstellt, die für diese Anlage angefallen sind.
deviceList	Gibt ein Menü zurück, in dem alle Anlagen aufgelistet sind. Dies kann durch die Parameter `locationId` und `devTypeId` auf bestimmte Standorte oder Anlagentypen eingeschränkt werden. Stehen diese Parameter auf „null“, so werden sie nicht beachtet.
deviceMasterConfigData	Gibt einen Text zurück, in dem die Stammdaten der Anlage stehen.
deviceNotebook	Gibt ein Menü zurück, in dem alle „Threads“ des Notizbuchs der Anlage aufgeführt sind.
devicePage	Gibt ein Menü zurück, in dem alle möglichen Aktionen für eine bestimmte Anlage aufgeführt sind.
devicePendingMessage	Gibt ein Menü zurück, dass alle Prozessvariableninstanzen der Anlage aufführt, die nicht der Prioritätsklasse „OK“ zugeordnet sind.
deviceSummary	Gibt ein Icon-Menü zurück, in dem fest definierte Prozessvariablen der Anlage aufgeführt sind, wie z.B. bei Fahrtreppen der Fahrmodus und die Fahrtrichtung.
deviceTypeList	Gibt ein Menü zurück, in dem alle Anlagentypen aufgelistet sind. Der Parameter `locationId` kann die Liste auf einen bestimmten Standort einschränken.
locationList	Gibt ein Menü zurück, in dem alle Standorte aufgeführt sind, in denen Anlagen existieren.
notebookEntryGroup	Gibt einen Text zurück, der einen „Thread“ eines Anlagennnotizbuchs darstellt.

pviInformation	Gibt einen Text zurück, der detailierte Informationen zu einer bestimmten Prozessvariableninstanz anzeigt.

TransformatorBean

Diese Klasse implementiert alle Methoden des Interfaces `Transformator`. Zusätzlich definiert sie einige Hilfsmethoden. Sie liegt im Schichtmodell im Model, jedoch zwischen Controller und der Service-Schicht. Sie kommuniziert konsequenterweise nicht direkt mit der Datenbank bzw. der Persistenzschicht, sondern leitet alle Anfragen an die Service-Schicht weiter.

Methode	Beschreibung
generateURL	Erstellt aus dem Klassenattribut `fullContextPath` und den ihr übergebenen Parametern eine URL.
getBreadcrumb	Erzeugt den String, der in der Statusleiste der Telefone angezeigt wird.
getDeviceSummaryProperties	Hier werden die Prozessvariblen festgelegt, die für den übergebenen DeviceType in der Methode `deviceSummary` angezeigt werden sollen.

Für alle privaten Klassenattribute existieren setter-Methoden. Die Services werden nicht selbst instanziiert, sondern per Spring Dependency Injection in das Objekt injiziert. Diese Dependency-Injection wird im Spring Application Context konfiguriert.

Hilfsklassen

Die Hilfsklassen sind in zwei Packages eingeordnet. Im Package `com.skytecag.xsight.cisco.model` befinden sich Klassen, die ausschließlich der Datenhaltung dienen und die keinerlei Logik implementieren. Die Klassen im Package `com.skytecag.xsight.cisco` beinhalten Konstanten oder stellen Funktionalität bereit.

Package com.skytecag.xsight.cisco.model

Klasse	Beschreibung
CIPImage	Beinhaltet alle Attribute, die für ein CIP-Image notwendig sind (Breite, Höhe, Farbtiefe, Bilddaten)
CiscoIPPhoneIconMenu	Repräsentiert die Daten, die für ein IconMenu notwendig sind (Titel, Prompt, Liste von Menüpunkten, Liste von Softkeys, List von Icons)
CiscoIPPhoneMenu	Repräsentiert die Daten, die für ein Menu notwendig sind (Titel, Prompt, List von Menüpunkten, Liste von Softkeys)
CiscoIPPhoneText	Repräsentiert die Daten, die für Textdarstellung notwendig sind (Titel, Prompt, Text, Liste von Softkeys)
IconItem	Beinhaltet Attribute für ein einzelnes Icon (Index, Breite, Höhe, Farbtiefe, Bilddaten)

MenuItem	Beinhaltet Attribute für einen einzelnen Menüpunkt (Name/Text, URL, Index des Icons)
SoftKeyItem	Beinhaltet Attribute für einen einzelnen Softkey (Name/Text, URL, Position)

Alle Klassen enthalten neben den Default-Konstruktoren, die keine Parameter erwarten, auch noch weitere mit denen sich die wichtigsten bis alle Paramter bei der Instanziierung belegen lassen.

Package com.skytecag.xsight.cisco

Klasse / *Enum*	**Beschreibung**
CIPImages	Beinhaltet zwei Konstanten, die zentral definierte `CIPImage`-Instanzen darstellen
ProcessVariableInstancePriorityComparator	Implementiert `java.Util.Comparator<ProcessVariableInstance>`. Wird benötigt, um anstehende Meldungen nach ihrer Priorität zu sortieren.
RequestParameters	Hier werden die Namen der HTTP-GET-Parameter definiert. **Vorsicht!** Werden hier Änderungen vorgenommen, muss die Controller-Klasse ebenfalls geändert werden. Dies liegt daran, dass in Annotations keine Enum-Werte verwendet werden können.
UmlautTranscriber	Vom View verwendete Hilfsklasse, mit der Umlaute durch ihre 2-Buchstaben-Äquivalente ausgetauscht werden.

Servlet-Konfiguration

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/
    spring-beans.dtd">

<beans>
    <bean id="myController" class="com.skytecag.xsight.cisco.gui.CiscoController">
                <property name="transformator" ref="transformator"/>
    </bean>

    <bean id="urlMapping" class="org.springframework.web.servlet.handler.
        SimpleUrlHandlerMapping">
        <property name="mappings">
            <props>
                <prop key="*">myController</prop>
            </props>
        </property>
    </bean>

    <bean id="velocityConfig" class="org.springframework.web.servlet.view.velocity.
        VelocityConfigurer">
        <property name="resourceLoaderPath" value="/WEB-INF/velocity/"/>
```

```
            <property name="velocityProperties">
                    <props>
                            <!-- nice for debugging, should be turned off in
                                production code -->
                            <prop key="runtime.log.invalid.references">true</prop>
                    </props>
            </property>
    </bean>

    <bean id="viewResolver" class="org.springframework.web.servlet.view.velocity.
        VelocityViewResolver">
        <property name="contentType" value="text/xml"/>
        <property name="cache" value="true"/>
        <property name="prefix" value=""/>
        <property name="suffix" value=".vm"/>
    </bean>

</beans>
```

In der Servlet-Konfiguration werden vier Beans definiert.

Bean-Name	Beschreibung
myController	definiert den Controller der Anwendung
urlMapping	legt fest, dass alle URLs (`key="*"`) über den Controller `myController` verarbeitet werden
velocityConfig	konfiguriert Velocity und legt dessen Basis-Pfad fest
viewResolver	konfiguriert den View-Resolver von Velocity, der dafür verantwortlich ist, über einen View-Namen das richtige File zu finden

Application Context

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/
    spring-beans.dtd">

<beans>
        <bean id="transformator" class="com.skytecag.xsight.cisco.transformation.
            TransformatorBean">
                <property name="locationService" ref="locationService"/>
                <property name="deviceService" ref="deviceService"/>
                <property name="deviceTypeService" ref="deviceTypeService"/>
                <property name="processVariableService" ref="processVariableService
                    "/>
                <property name="priorityClassService" ref="priorityClassService"/>
                <property name="eventService" ref="eventService"/>
                <property name="notebookService" ref="notebookService"/>
        </bean>
</beans>
```

Der Application Context der Cisco-Moduls ist sehr überschaubar. Es wird nur eine einzige Bean definiert. Über das `property`-Tag werden mit dem Attribut `ref` Beans referenziert, die an anderer Stelle definiert sind. Alle Services, die hier aufgeführt sind, werden im Projekt `xsight_common` definiert und befinden sich, wie am Namen erkennbar, im Service-Layer des Projekts.

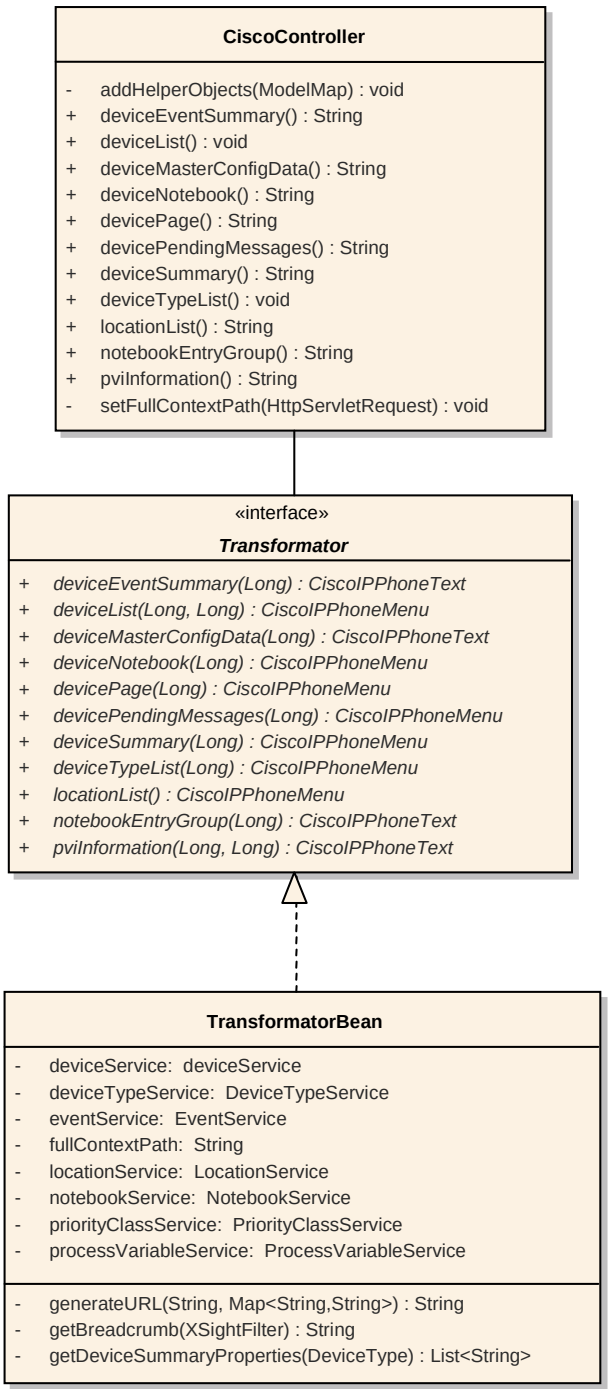


Abbildung 4.4: UML-Diagramm der wichtigsten Klassen des Moduls

5. Ergebnis und Fazit

5.1. Tests

5.1.1. Testumgebung

Es wurde eine Kopie der Produktivdatenbank der Essener Verkehrsbetriebe eingesetzt, um ein möglichst realitätsnahe Testumgebung zu schaffen. In dieser Datenbank sind alle ca. 50 Anlagen des Systems konfiguriert und die Zustände und Event-Logs mit reellen Werten gefüllt, so dass mit einer große Vielfalt an Anlagentypen und Situationen getestet werden konnte.

Anhand dieser Testdatenbank konnte dann die Darstellung einzelner Geräteinformationen zwischen Webcenter und dem Cisco-Modul verglichen werden, um Defizite bzw. Fehler erkennen und beheben zu können.

5.1.2. Unit-Tests

Zum entwicklungsseiten Test der Applikation wurden Unit-Tests auf Basis von JUnit eingesetzt. Das Spring-Framework bringt diesbezüglich einige Möglichkeiten mit, um z.B. den ApplicationContext speziell für einen Test gestalten zu können, um einzelne Elemente des Programms herauszulösen und jedes für sich testen zu können.[Spr08b]

```
package com.skytecag.xsight.cisco.transformation;

import junit.framework.TestCase;

public class TransformatorBeanTest extends TestCase {

        public void testGenerateURL() throws Exception {
                TransformatorBean transformator = new TransformatorBean();
                Map<String, String> parameters = new HashMap<String, String>;

                parameters.add("param1", "value1");
                parameters.add("param2", "value2");

                transformator.setFullContext("http://testhost.domain.tld/app/");

                String testString = transformator.generatorURL("uri", parameters);

                assertEquals(testString, "http://testhost.domain.tld/app/gui?param1=
                    value1&param2=value2");
        }

}
```

Dieses Listing zeigt das Beispiel eines Unit-Tests. Wichtig bei Unit-Tests ist, dass möglichst immer nur eine gekapselte Funktionalität für sich getestet wird. In diesem Fall wird die Funktion `generateURL(String)` der Klasse `TransformatorBean` getestet. Dieser Test bezieht sich neben der zu testenden Funktion ausschließlich Java Basis-Funktionalität, sodass diese Voraussetzung erfüllt ist.

Die zentrale Zeile dieses Tests ist der Aufruf an die Funktion `assertEquals()`. Dieser Aufruf teilt dem Test-Framework mit, dass es die beiden übergebenen Werte auf Gleichheit prüfen soll. Wird diese Bedingung nicht erfüllt, gilt der Test als fehlgeschlagen.

Wenn nun für alle (wichtigen) Methoden einer Applikation auf diese Weise Tests implementiert werden, kann einfach sichergestellt werden, dass die Applikation sich zu jedem Zeitpunkt, z.B. nach größeren Änderungen, in einem funktionalen Zustand befindet.

5.2. Zusammenfassung

Im ersten Schritt dieses Projektes wurde gezeigt, dass eine Darstellung von SCADA-Daten in sinnvoller Art und Weise auf Voice-over-IP-Hardware möglich ist. Selbst mit dieser sehr schnellen, einfachen Lösung war bereits eine Darstellung aller relevanten Daten möglich.

Im zweiten Schritt wurde im Gespräch mit den Entwicklern der SKYTEC AG erörtert, wie eine Anbindung an das SCADA-System am sinnvollsten gestaltet werden kann. Anschließen wurde die Lösung auf dem besprochenen Weg implementiert. Bei der Implementierung wurde besonders darauf geachtet, dass die Benutzerführung trotz vollkommen verschiedener Möglichkeiten auf den Telefonen analog zu der im X-SIGHT-Webcenter gehalten wird, sodass Benutzer, die mit beiden Systemen arbeiten, sich möglichst schnell zurechtfinden können.

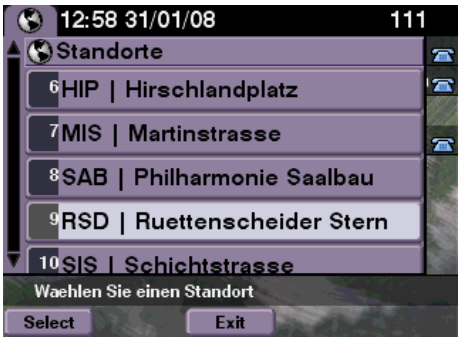


Abbildung 5.1: Liste der Gerätestandorte

Abbildung 5.2: Aktionsauswahl

Abbildung 5.3: Kurzüberblick über den Gerätestatus

Diese drei Screenshots zeigen die Kernfunktionalitäten des Cisco-Modul-Frontends. In Abbildung 5.1 ist die Auswahl des Standortes zu sehen. Im Echtsystem ist denkbar, dass dieser Schritt übersprungen wird und nur der Standort sichtbar ist, an dem sich das Gerät und der Bediener befinden.

Abbildung 5.2 zeigt das Menü, in dem für eine ausgewählte Anlage ausgewählt wird, welche Informationen angezeigt werden sollen. Abbildung 5.3 beinhaltet eine solche Informationsanzeige.

In den letzten beiden Abbildungen ist die sog. „Brotkrümel“-Navigation zu sehen, die die hierarchische Einordnung der aktuellen Anlage auf Anlagentyp und Standort visualisiert.[Wik08a]

5.3. Fazit

Die Anwendung von VoIP-Hardware und entsprechender Interface-Software kann im Bereich der verteilten SCADA-Systeme durchaus Vorteile gegenüber den herkömmlich eingesetzten PCs aufweisen.

Als wichtigste Merkmale stechen die Robustheit und der geringe Platz- und Energiebedarf der Telefone hervor. Auch die Wartungsintensität der Telefone ist wesentlich geringer als die von PCs, da bei den Telefonen kein komplexes Patch-Management erforderlich ist. Zu guter Letzt ist durch das Solid-State-Design der Telefone im Gegensatz zu PCs mit ihren beweglichen Lüftern und Laufwerken und einer höheren Empfindlichkeit gegenüber Temperaturschwankungen eine längere Lebensdauer zu erreichen.

Durch die Entwicklung des vorliegenden Interfaces steht dem Produkt SCADA X-SIGHT nun die Möglichkeit offen, Cisco VoIP-Hardware in das SCADA-System zu integrieren.

Durch die strengen Vorgaben bezüglich der Sicherheit des Systems seitens der Verkehrsbetriebe konnte jedoch nicht die volle Palette der technischen Möglichkeiten ausgeschöpft werden. Ein steuernder Zugriff auf das SCADA-System ist von der technischen Seite keine große Herausforderung. Dem System die notwendige Sicherheit zu verschaffen, wirft jedoch einige Probleme auf (siehe auch Kapitel 6).

6. Ausblick

6.1. weiterführende Arbeiten

Mit dem jetzigen Stand der Implementierung ist ein rein lesender, passiver Zugriff auf das System gegeben. Um auch den aktiven, steuernden Zugriff zu ermöglichen, muss erst eine geeignete Authentifizierung implementiert werden.

Da sich hier verschiedenste Möglichkeiten, aber auch Schwierigkeiten bieten, wird dieses Thema in einer Folgearbeit tiefer beleuchtet.

Probleme, die bei der Authentifizierung mit Hilfe der vorhandenen Telefone auftreten können, teilen sich in die Kategorien Sicherheit und Benutzerfreundlichkeit. Da die Telefone ausschließlich über Zifferntasten verfügen, ist die Eingabe von üblichen alpha-numerischen Benutzername-Passwort-Kombinationen äußerst mühsam und unkomfortabel. Die Telefone unterstützen darüberhinaus das https-Protokoll nicht und erlauben aufgrund der fehlenden Möglichkeit, Daten auf ihnen verarbeiten zu können, auch keine anderweitige Verschlüsselung. Dadurch ist eine Authentifizierung nur im Klartext und somit potenziell unsicher möglich.

Um diese Probleme zu umgehen, werden unter anderem folgende Authentifizierungs-Mechanismen betrachtet:

- One Time Pads
- PIN/TAN
- Smart Cards

Eine weitere Aufgabe, die zu bearbeiten ist, stellt die möglichst nahtlose Integration der Administration der VoIP-Infrastruktur in SCADA X-SIGHT dar. Damit soll zum Beispiel erreicht werden, dass jedes Telefon an seinem Standort in der grafischen Übersicht zu sehen ist und dort direkt konfiguriert werden kann.

6.2. andere Anwendungsgebiete

Neben dem in dieser Arbeit beschriebenen Szenario zum Einsatz bei Verkehrsbetrieben gibt es natürlich eine Reihe weiterer Anwendungsmöglichkeiten für die Anlagen- und Gerätesteuerung mit VoIP-Telefonen.

So ist zum Beispiel denkbar, ein Bürogebäude, das durch SCADA X-SIGHT oder ein anderes SCADA-Produkt geregelt und überwacht wird, um diese Steuerungsmöglichkeit zu erweitern.

Dies erspart Bedienterminals in einzelnen Büros, um bestimmte Funktionen wie z.B. Öffnen oder Schließen von Sonnenschutzvorrichtungen oder Regelung einer Klimaanlage durchzuführen. Neben der Möglichkeit, diese Funktionen lokal ohne Authentifizierung zu bedienen, kann mit dem im Rahmen der Diplomarbeit evaluierten Konzept auch eine entfernte bzw. globale Steuerung des Gebäudes erfolgen.

A. Cisco XML

Der Browser der Cisco-IP-Telefone versteht ausschließlich Cisco-eigene XML-Formate. Alle anderen Mime-Types als „text/xml“ werden als Plain-Text angezeigt.

Folgende XML-Typen stehen zur Verfügung:

- **CiscoIPPhoneMenu** Ein Menü der gleichen Art wie z.B. das Settings-Menü. Jedem Menü-Eintrag ist eine URL zugeordnet. Zusätzlich können die Softkeys ebenfalls mit URLs belegt werden.
- **CiscoIPPhoneText** Stellt einfachen Plain-Text dar, allerdings mit der Möglichkeit, die Softkeys zu belegen.
- **CiscoIPPhoneInput** Ein Formular mit einer beliebigen Anzahl von Eingabefeldern mit bestimmten Eigenschaften, z.B. Numerisch, Uppercase, Lowercase, Password field.
- **CiscoIPPhoneDirectory** Ein Telefonverzeichnis. Jeder Eintrag besteht aus Name und Telefonnummer. Dieser Typ ist hauptsächlich für die Erweiterung der Directory-Services über den Telefonparameter „Directories URL“ gedacht.
- **CiscoIPPhoneImage** Ein Bild mit 1-2 Bit pro Pixel. Die Bitmap ist im XML-Dokument enthalten. Eine weitere Navigation ist über die Softkeys möglich.
- **CiscoIPPhoneImageFile** Zeigt ein Bild im PNG-Format von einer externen URL an. Auch hier ist die Angabe von Softkeys möglich.
- **CiscoIPPhoneGraphicMenu** Zeigt ein Bild mit 1-2 Bit pro Pixel an. Die Bitmap ist im XML-Dokument erhalten. Navigation erfolgt über die DTMF-Zahlentasten oder über die Softkeys.
- **CiscoIPPhoneGraphicFileMenu** Zeigt ein Bild im PNG-Format von einer externen URL an. Navigation erfolgt über die DTMF-Zahlentasten, über Softkeys oder über definierte TouchAreas bei einem Gerät mit Touchscreen.
- **CiscoIPPhoneIconMenu** Wie ein „normales“ Menü, jedoch mit einem Icon von maximal 16x10 Pixel Größe vor jedem Menüeintrag.
- **CiscoIPPhoneStatus** Eine spezielle Art der Darstellung. Stellt eine kleine Statusbox da, die über dem eigentlichen Display „schwebt“. Besteht aus einer CIP-Grafik und (optional) aus einer Beschreibungszeile und einem Timer.
- **CiscoIPPhoneExecute** Objekte dieser Art werden per HTTP POST-Request an das HTTP-Push-Interface des Telefons gesendet. Innerhalb dieses Objekts können URIs/URLs angegeben werden, die vom Telefon sofort ausgeführt bzw. angezeigt werden sollen.

- **CiscoIPPhoneResponse** Wird vom Telefon als Antwort auf ein CiscoIPPhoneExecute-Objekt zurückgegeben.
- **CiscoIPPhoneError** Wenn beim Verarbeiten eines CiscoIPPhoneExecute-Objektes ein Fehler auftritt, wird dieses Objekt zusätzlich zur CiscoIPPhoneResponse zurückgegeben.

Die XML-Schema-Definition (xsd) findet sich im Cisco IP-Services SDK.

Abbildungsverzeichnis

1.1. Steuerschrank mit eingebautem FA-PC 2

2.1. Steuerung von Einzelsystemen 3
2.2. Steuerung von Anlagen über ein SCADA-System 4
2.3. Steuerung von Anlagen über ein FCML-SCADA-System 4
2.4. Schichten von X-SIGHT 6
2.5. Zusammenhang zwischen den X-SIGHT-Modulen 8
2.6. Cisco 7971G-GE 8
2.7. Cisco 7961G-GE 9
2.8. Cisco 7921G 9
2.9. Cisco 2811 9
2.10. Cisco Aironet AP1240G 10
2.11. Der CallManager Simulator 10

3.1. Physikalischer Aufbau der Gerätelandschaft 11
3.2. Das Settings-Menü 12
3.3. URL-Konfiguration 21

4.1. Anbindung an den X-SIGHT-Proxy 23
4.2. Verwendete Klassen/JSPs 24
4.3. Anbindung als X-SIGHT-Modul 26
4.4. UML-Diagramm der wichtigsten Klassen des Moduls 31

5.1. Liste der Gerätestandorte 34
5.2. Aktionsauswahl 34
5.3. Kurzüberblick über den Gerätestatus 35

Literaturverzeichnis

[Apa08a] Apache Software Foundation: *Homepage des Tapestry-Framworks*, 2008. http://tapestry.apache.org, Zugriff am 16.04.2008.

[Apa08b] Apache Software Foundation: *Homepage des Velocity-Frameworks*, 2008. http://velocity.apache.org, Zugriff am 16.04.2008.

[BLO+08] Bry, François, Bernhard Lorenz, Hans Jürgen Ohlbach, Martin Roeder und Marc Weinberger: *The Facility Control Markup Language FCML*. In: *Proc. of the The Second International Conference on the Digital Society (ICDS), (to appear)*. IEEE, 2008.

[RB07] Roeder, M. und O. Blume: *FCML Spezifikation, Version 1.1*. FCML-Group, 2007. http://www.fcml-group.org/fileadmin/Downloads/FCML/Anlagen/Core/Version_1.1/FCML-v1.1.pdf, Zugriff am 11.01.2008.

[Red08] RedHat, Inc.: *Homepage des Hibernate-Framworks*, 2008. http://www.hibernate.org, Zugriff am 16.04.2008.

[Spr08a] SpringSource: *Homepage des Spring-Framworks*, 2008. http://www.springframework.org, Zugriff am 16.04.2008.

[Spr08b] SpringSource: *SpringFramework Documentation*, Kapitel 8: „Testing“. 2008. http://static.springframework.org/spring/docs/2.5.x/reference/testing.html, Zugriff am 20.04.2008.

[Wik08a] Wikipedia: *Brotkrümelnavigation*, 2008. http://de.wikipedia.org/wiki/Brotkr%C3%BCmelnavigation, Zugriff am 15.04.2008.

[Wik08b] Wikipedia: *Model View Controller*, 2008. http://de.wikipedia.org/wiki/Model_View_Controller, Zugriff am 11.01.2008.

[Wik08c] Wikipedia: *SCADA*, 2008. http://en.wikipedia.org/wiki/SCADA, Zugriff am 15.04.2008.