HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

SIEMENS

Mobile Commissioning Tool

for

Room Automation

Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2010

Autoren:	Kaspar Fenner
	Reto Schneebeli
Betreuer:	Prof. Hansjörg Huser
Projektpartner:	Siemens Schweiz AG Building Technologies, Zug
Experte:	Stefan Zettel, Ascentiv AG, Zürich
Gegenleser:	Prof. Dr.-Ing. Andreas Rinkel

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

Erklärung über die eigenständige Arbeit

Erklärung

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.

Ort, Datum:

Name, Unterschrift:
Kaspar Fenner

Reto Schneebeli

1	Aufgabenstellung
2	Management Summary
3	Technischer Bericht
4	Projektplan und Zeitplanung Zeiterfassung
5	Anforderungen
6	Domainanalyse
7	Architektur und Design
8	Testdokumentation
9	Erfahrungsberichte
10	Glossar Literaturverzeichnis

Mobile Commissioning Tool for Room Automation

Bachelorarbeit für Kaspar Fenner, Reto Schneebeli

Einführung

In der Siemens Raumautomation werden alle Raumgeräte durch einen Kontroller gesteuert. Die Raumkontroller werden in der nächsten Produkt Version (Einführung Ende 2011) untereinander über ein IP-Netz verbunden und besitzen einen integrierten Webserver für die Konfiguration des Kontrollers und den Datenpunkttest. Um einen Kontroller in Betrieb zu nehmen, muss der passende Raumkontroller im IP-Netz gesucht & identifiziert werden und es müssen Konfigurationsdateien in den Kontroller geladen werden, welche die Datenpunktzuweisungen festlegen. Nach der netzwerkseitigen Inbetriebnahme wird der Datenpunkttest durchgeführt, um zu überprüfen, ob die Verdrahtung zu den Feldgeräten im Raum und deren Funktion korrekt ist. Die Person welche den Datenpunkttest durchführt wird *Installer* genannt und verfügt nur über minimale PC Kenntnisse.

Problemstellungen

1. Unkonfigurierte Kontroller haben eine dynamisch vergebene IP und es ist nicht bekannt welcher Kontroller welche IP hat.

2. Der *Installer* muss alle IPs der konfigurierten Kontroller kennen, welche er dann einzeln im Browser eintippen muss, um auf deren Konfigurationswebseite zugreifen zu können. Zusätzlich sollte er wissen, welche IP zu welchem physischen Kontroller gehört, der sich irgendwo im Gebäude befindet.

3. Bei der Inbetriebnahme sind die Feldgeräte bereits im Gebäude installiert und mit dem Kontroller verkabelt. Beim Datenpunkttest muss die korrekte Verdrahtung der Kontroller überprüft werden, wobei diese Tests am Notebook über die Webseite des Kontrollers ausgeführt werden. Während dem Test muss der Installer im Raum selbst überprüfen, ob beim Verändern der Werte auch wirklich das Licht angeht, oder die Lüftung startet. Es ist umständlich, die ganze Zeit mit dem Notebook im Gebäude herumzulaufen, die Webseite zu bedienen und gleichzeitig Tests an den Feldgeräten durchführen (z.B. Temperaturfühler erwärmen).

4. Zurzeit ist geplant, dass der Datenpunkttest nur über eine für die Benutzung am PC optimierte HTML-Webseite durchgeführt werden kann, welche vom Kontroller direkt bereitgestellt wird. Die Bedienung einer solchen Webseite mit einem Smartphone o.ä. ist nicht Benutzerfreundlich.

Lösungsansatz

Der *Installer* verwendet als primäres Gerät nicht mehr ein Notebook zur Inbetriebnahme sondern ein Smartphone, das über Wireless mit dem IP-Netz verbunden ist, und mit dem er sich frei im Gebäude bewegen kann und so z.B. den Datenpunkttest durchführt. Das Herkömmliche Datenpunkttest-Tool verwendet Graphen, welche die Aktivitäten eines Datenpunktes in Realtime aufzeichnen. Dies ermöglicht das *One-Man-Commissioning*, da der *Installer* dadurch zuerst an den Feldgeräten seine Tests durchführen kann, und danach mit dem Datenpunkttest-Tool überprüfen kann, ob die Aufgezeichneten Änderungen korrekt sind.

Durch den Einsatz eines Smartphones würde die herkömmliche Graphaufzeichnung nicht mehr benötigt, da der Installer nun mit einer Hand die Tests an den Feldgeräten durführen und direkt auf dem Smartphone in der anderen Hand die Aktivitäten überprüfen kann.

Vorhandenes Umfeld/Entwicklungsstand während BA

- Hardware des neuen Raumkontrollers PXC3 ist nicht verfügbar
- Schnittstellen der Kontrollerfirmware sind/werden zurzeit definiert, sind während der BA z.T. vorhanden, aber ohne Garantie auf Vollständigkeit/Endgültigkeit
- Kontroller-Architektur basiert auf dem Thrift-Framework von Facebook
- „UDP Multicast Device Discovery“-Protokoll Spezifikation vorhanden, Lieferung der API möglicherweise im November
- Windows Phone 7 Entwicklungsplattform und Emulator vorhanden
- Windows Phone 7 Hardware vorhanden??? (voraussichtlicher Release: Nov/Dez)

⇨ **Fazit:** Da sich das ganze Kontroller-Umfeld zurzeit in Entwicklung befindet, müssen Abhängigkeiten möglichst vermieden werden. Es kann keine direkte Kommunikation zwischen Smartphone und Raumkontroller stattfinden. Deshalb soll als Zwischenstelle ein Notebook geschalten werden, das die Webservices anbietet und als Gateway für das Kontroller-IP-Netz fungiert. So kann die Kommunikation mit den Raumkontrollern einfach simuliert („faked“) werden.

Anforderungen

1. Smartphone-Plattform: Windows Phone 7 (Android, iPhone, etc. sollen in Zukunft auch unterstützt werden können)
   Desktop-Plattform: Windows XP/Vista/7

2. Raumkontroller sollen über das Smartphone gesucht werden können. Da es in einem Gebäude u.U. sehr viele Kontroller hat, werden Filter-/Sortier-/Struktur-Hilfen benötigt.

3. Wink/Service-Pin Event soll über das Smartphone gesendet/empfangen werden können.

4. Eine vereinfachte Version des Datenpunkttests soll über eine für Smartphones optimierte Oberfläche ausgeführt werden können.

Aufgabenstellung

Als neue Client/Server Anwendung soll eine Applikation für Windows Phone 7 entwickelt werden, welche mit einem ebenfalls zu entwickelnden Webservice (C#) kommuniziert, der auf einem Windows XP/Vista/7 Notebook läuft.

Die WP7 Applikation bietet folgende Funktionalität:
- Suchen, filtern, auflisten von Raumkontrollern in einem IP-Netz
- Senden von Wink an einen Raumkontroller (LED leuchtet auf)
- Empfangen von Service-Pin auf dem Smartphone
- Vereinfachter und an die Anforderung eines Smartphones angepasster Datenpunkttest

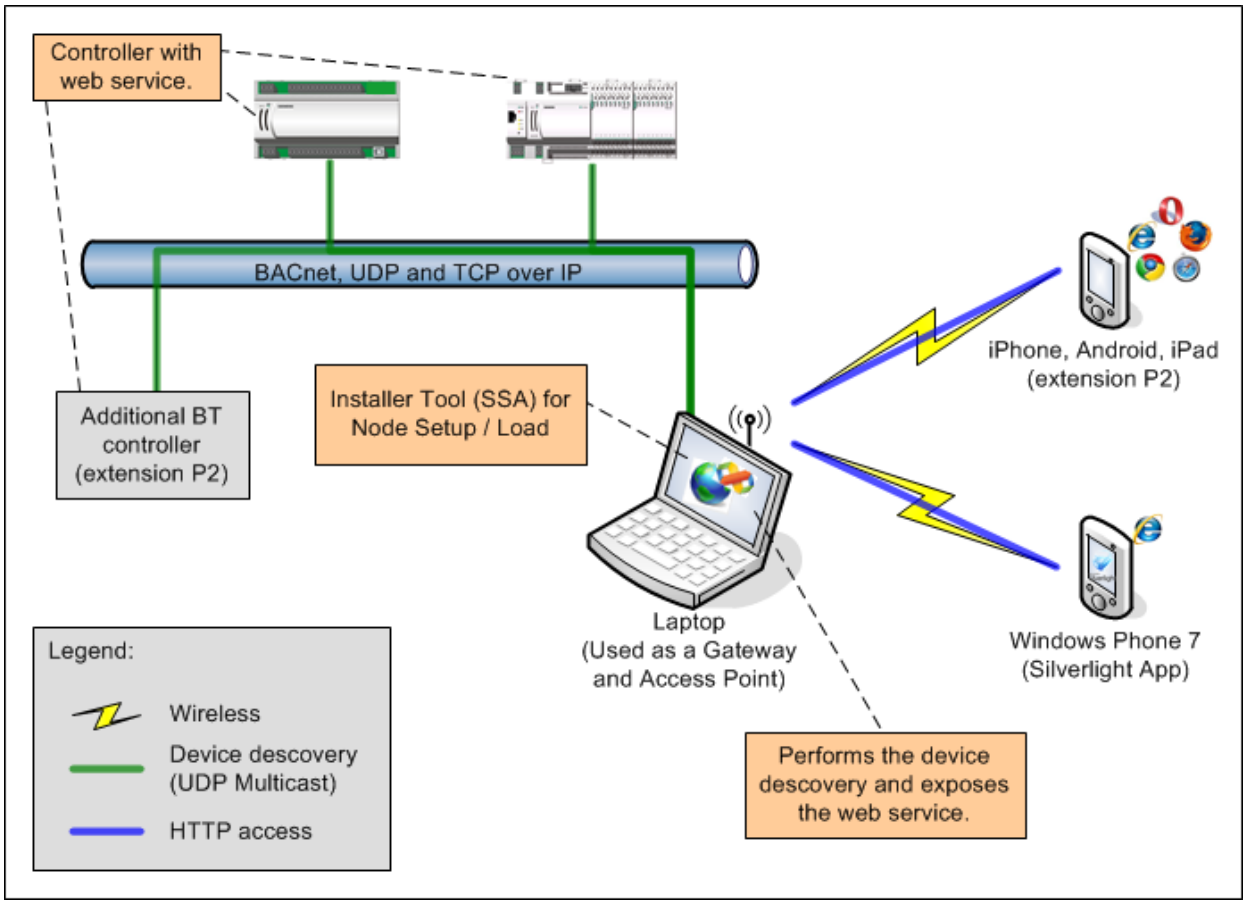
Der Webservice (C#) auf dem Notebook stellt die von der WP7 Applikation benötigten Funktionalitäten via REST zur Verfügung. Zur Abkoppelung bzw. Aufbau einer Simulations-Umgebung müssen darüber hinaus folgende Teile implementiert werden:

- Simulation der UDP Device Discovery API für das Suchen der Raumkontroller (allenfalls später ersetzt durch echte API sobald verfügbar)
- Simulation der Raumkontroller-Schnittstelle für den Datenpunkttest

Die wichtigsten Resultate der BA stellen folgende Artefakte dar:

- Requirements/UseCases
- Schnittstellenbeschreibung des Webservices für den Datenpunkttest bzw. Umsetzung als Prototyp
- GUI-Studie für den Datenpunkttest bzw. Prototyp für das WP7
- Proof of Concept für den Physikalischen Aufbau der Netzwerkkommunikation mit einem Laptop als Gateway
- (optional) Vergleich REST/SOAP/WCF/Thrift etc.
- (optional) Security-Konzept (Authentifizierung, Autorisierung, Vertraulichkeit)

Systemarchitektur-Übersicht



Projektpartner

Auftraggeber

René Föhn
Project Manager DESIGO V5 Tools

Ekaterina Panteleeva

Senior SW Architect
Project Manager LMS

Studenten

Reto Schneebeli

Kaspar Fenner

Betreuung HSR

Hansjörg Huser

Experte
Stefan Zettel

Projektabwicklung

Termine:

- Beginn der Arbeit: M**o., 20. Sept. 2010**
- Abgabetermin Kurzfassung zum Review Eingabe via Spezialtool: Fr. **16.12.10**
- Abgabetermin (ink. Poster): **Do. 23.12.10**, 12.00 Uhr
- Zwischenbesprechung/Review mit Auftraggeber nach Projektplan
- Mündliche Diplomprüfung: in der Zeit vom 3.1 bis 4.2.2010
- Diplomfeier: **04.03.11**

Arbeitsaufwand

Für die erfolgreich abgeschlossene Arbeit werden 12 ECTS angerechnet. Dies entspricht einer Arbeitsleistung von 360 Stunden pro Student.

Hinweise für die Gliederung und Abwicklung des Projektes:

Gliedern Sie Ihre Arbeit in 4 bis 5 Teilschritte. Schliessen Sie jeden Teilschritt mit einem Meilenstein ab. Definieren Sie für jeden Meilenstein, welche Resultate dann vorliegen müssen!

Folgende Teilschritte bzw. Meilensteine sollten Sie in Ihrer Planung vorsehen:
Schritt 1: Projektauftrag inkl. Projektplan (mit Meilensteinen),

- Meilenstein 1: Review des Projektauftrages abgeschlossen. Projektauftrag von Auftraggeber und Dozent genehmigt
  Letzter Meilenstein: Systemtest abgeschlossen
  Termin: ca. eine Woche vor Abgabe

- Entwickeln Sie Ihre SW in einem iterativen, inkrementellen Prozess: Planen Sie möglichst früh einen ersten lauffähigen Prototypen mit den wichtigsten und kritischsten Kernfunktionen. In die folgenden Phasen können Sie dieses Kernsystem schrittweise ausbauen und testen.
- Falls Sie in Ihrer Arbeit neue oder Ihnen unbekannte Technologien einsetzen, sollten Sie parallel zum Erarbeiten des Projektauftrages mit dem Technologiestudium beginnen.
- Setzen Sie konsequent Unit-Tests ein! Verwalten Sie ihre Software und Dokumente auf einem SVN-Repository. Stellen Sie sicher, dass der/die Betreuer jederzeit Zugriff auf das Repository haben und dass das Projekt anhand des Repositories jederzeit wiederhergestellt werden kann.
- Achten Sie auf die Einhaltung guter Programmier- und Designprinzipien
  - DRY, high cohesion, loose coupling, etc.
  - Clean Code!
- Halten Sie sich im Übrigen an die Vorgaben aus dem Modul SE-Projekt.

Projektadministration

- Führen Sie ein individuelles Projekttagebuch aus dem ersichtlich wird, welche Arbeiten Sie durchgeführt haben (inkl. Zeitaufwand). Diese Angaben sollten u.a. eine individuelle Beurteilung ermöglichen.
- Dokumentieren Sie Ihre Arbeiten laufend. Legen Sie Ihre Projektdokumentation mit der aktuellen Planung und den Beschreibungen der Arbeitsresultate elektronisch in einem Projektordner ab. Dieser Projektordner sollte jederzeit einsehbar sein (z.B svn-Server oder File-Share).

Inhalt der Dokumentation

Bei der Abgabe muss jede Arbeit folgende Inhalte haben:

- Dokumente gemäss Vorgabe: https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html
- Aufgabenstellung
- Technischer Bericht
- Projektdokumentation
- Die Abgabe ist so zu gliedern, dass die obigen Inhalte klar erkenntlich und auffindbar sind.
- Zitate sind zu kennzeichnen, die Quelle ist anzugeben.
- Verwendete Dokumente und Literatur sind in einem Literaturverzeichnis aufzuführen.
- Projekttagebuch, Dokumentation des Projektverlaufes, Planung etc.

Form der Dokumentation:

- Bericht (Struktur gemäss Beschreibung) in Ordner(1 Exemplar für HSR)
- Alle Dokumente und Quellen der erstellten SW auf CD, CD's sauber angeschrieben (2 Ex.).

Fortschrittsbesprechung:

Regelmässig findet zu einem fixen Zeitpunkt eine Fortschrittsbesprechung statt.
Teilnehmer: Dozent und Studenten, bei Bedarf auch Vertreter der Auftraggeber
Termin: jeweils xxx, Raum 6.010 (Abweichungen werden rechtzeitig kommuniziert)
Traktanden

- Was wurde erreicht, was ist geplant, welche Probleme stehen an
- Review von Code/Dokumentation (Abgabe jeweils einen Tag vor dem Meeting

Falls notwendig, können weitere Besprechungen / Diskussionen einberufen werden.
Sie erstellen zu jeder Besprechung ein Kurzprotokoll, welches Sie spätestens 2 Tage nach der Sitzung per e-mail an den Betreuer senden.

Rapperswil, 16. Sept.10
Hansjörg Huser

Management Summary

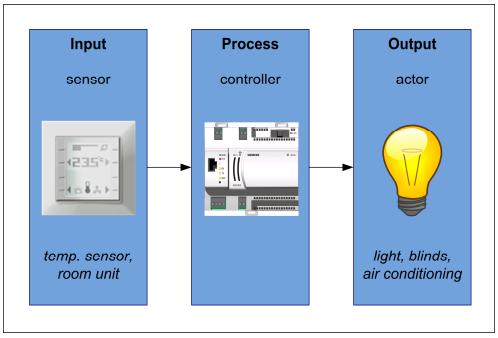
SMART COMMISSIONING

Windows Phone 7

Bachelorarbeit:	Mobile Commissioning Tool for Room Automation
Autoren:	Kaspar Fenner Reto Schneebeli
Betreuer:	Prof. Hansjörg Huser
Projektpartner:	Siemens Schweiz AG Building Technologies, Zug
Experte:	Stefan Zettel, Ascentiv AG, Zürich

Ausgangslage

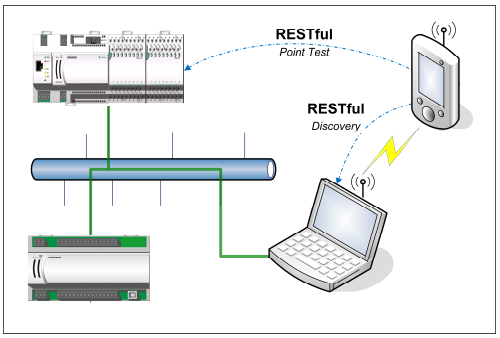
Bei Siemens Building Technologies wird zurzeit ein neues System für die Raumautomation entwickelt. Es ist das Ziel, durch die intelligente Steuerung von Aktoren (Heizung, Lüftung, Storen, etc.) ein optimales Raumklima zu schaffen. Ermöglicht wird dies durch programmierbare Steuergeräte, welche Signale von Sensoren (Temperatur, Helligkeit, Luftfeuchtigkeit, etc.) verarbeiten und die Aktoren steuern. Die Inbetriebnahme eines solchen Systems ist sehr komplex. So muss u.a. die Verdrahtung zwischen den Geräten und deren Funktionsfähigkeit überprüft sowie die Programmlogik eingespielt werden. Das Ziel dieser Bachelorarbeit ist es, den Inbetriebnahmeprozess softwareseitig zu vereinfachen.



Vorgehen/Technologien

Als wichtiger Teil der Aufgabe wurde das bestehende Umfeld und die Anforderungen der verschiedenen Stakeholders analysiert. Dieses Softwareprojekt wurde inkrementell iterativ nach den Vorgaben von RUP durchgeführt. Mit der Fertigstellung eines Architekturprototyps bei Projekthalbzeit konnten die Projektrisiken minimiert werden. Die folgenden Technologien wurden eingesetzt:

- C#
- Silverlight for Windows Phone 7
- .NET Framework 4.0 mit WPF, WCF
- REST-basierte Webservices



Ergebnis

Die Lösung umfasst mehrere Softwareteile in einer Client-Server-Architektur:

- Eine App für Windows Phone 7, die den Ingenieur bei der Inbetriebnahme optimal unterstützt.
- Ein Software-Gateway als PC-Anwendung, der die Kommunikation zwischen Smartphones und Steuergeräten vereinfacht.
- Ein Simulator als PC-Anwendung zur Simulation der Kommunikation mit den Steuergeräten, die sich zurzeit in Entwicklung befinden.
- Eine Spezifikation von RESTful Webservices für die zukünftige Umsetzung direkt in der Steuergeräte-Hardware.

Technischer Bericht

SMART COMMISSIONING

Windows Phone 7

Bachelorarbeit:	Mobile Commissioning Tool for Room Automation
Autoren:	Kaspar Fenner Reto Schneebeli
Betreuer:	Prof. Hansjörg Huser
Projektpartner:	Siemens Schweiz AG Building Technologies, Zug
Experte:	Stefan Zettel, Ascentiv AG, Zürich

0. Dokumentinformationen

0.1. Inhalt

0. Dokumentinformationen 2
0.1. Inhalt 2
0.2. Abbildungen 4
1. Einleitung 5
1.1. Umfeld 5
1.1.1. Unternehmen und Produkte 5
1.1.2. Raumautomation 5
1.1.3. Inbetriebnahme 5
1.2. Problemstellungen bei der Inbetriebnahme 6
1.3. Lösungsansatz 7
1.4. Abgrenzung 8
1.5. Aufgabenstellung 9
2. Analyse der bestehenden Lösungen 10
2.1. DESIGO Point Test (DPT) 10
2.2. Kontroller-Konfigurationswebseite (in Entwicklung) 11
2.3. Schlussfolgerungen der Analyse 11
3. Systemübersicht 12
3.1. Physikalische Systemarchitektur 12
3.2. Hardware-Sicht 13
3.3. Logische Sicht 14
3.4. Domainmodell 14
4. Anforderungen 16
4.1. Funktionale Anforderungen 16
4.2. Anwendungsfälle (Use Cases) 16
4.3. Nichtfunktionale Anforderungen 17
5. Architektur und Design 18
5.1. Komponentenübersicht 18
5.2. Kommunikation 19
5.2.1. Übersicht 19
5.2.2. Webservice-Schnittstelle (REST) 21
5.2.3. Benachrichtigung mit REST - Long Polling 22
5.2.4. Datenübertragungsformat (JSON) 22
5.2.5. Discovery-Protokoll (DICP) 22
5.3. Sicherheit 23
5.3.1. Basis-Authentifizierung für REST-Webservices 23
5.3.2. Verschlüsselte Verbindung durch WLAN Sicherheitsprotokolle 23
5.3.3. Kontroller sind ungeschützt im drahtgebundenen IP-Netz 23
5.4. Fehlerbehandlung und Fehlersuche 23
5.4.1. Fehlerbehandlung auf der SCT-App 23
5.4.2. Logging 24
6. Gateway (SCG) 26
6.1. Einleitung 26
6.1.1. Notwendigkeit 26
6.1.2. Funktionsumfang 26
6.1.3. Technologie 26
6.2. Architekturübersicht 26
6.3. Serviceschnittstelle (REST-Webservice) 27
6.3.1. Interfaces 27
6.3.2. Implementation mit WCF 28
6.3.3. Caching 29
6.3.4. Fehlerbehandlung / Fehlerrückgabe 29
6.5. Benutzerschnittstelle (UI) 30
7. Kontrollersimulator (CST) 31

7.1. Einleitung....31
7.1.1. Notwendigkeit....31
7.1.2. Funktionsumfang....31
7.2. Serviceschnittstelle (REST-Webservice)....31
7.3. Benutzerschnittstelle (UI)....32
8. Einführung in die Entwicklung für Windows Phone 7....34
8.1. Kurze Faktenübersicht....34
8.2. Entwicklungsumgebung....34
8.3. App auf echtem Gerät testen (App Hub Account)....35
8.4. App Hub Account für Studenten....35
8.5. Verteilung der Apps über Windows Phone Marketplace....35
8.6. Zusätzliche Tools und Libraries....35
9. Windows Phone 7 App (SCT)....37
9.1. Übersicht....37
9.2. Architektur- und Designentscheide....37
9.2.1. Architekturübersicht....37
9.2.2. MVVM-Pattern mit View-Model-Locator....38
9.2.3. Asynchrone Service-Architektur....38
9.2.4. Internationalisierung (I18N)....39
9.3. Leistungsfähigkeit (Performance)....39
9.4. Windows Phone Ausführungsmodell (Execution Model)....40
9.5. Benutzeroberfläche (UI)....41
9.5.1. Gebäudehierarchie und Gerätesuche....41
9.5.2. Kontrollerinformationen und Punktauswahl....42
9.5.3. Datenpunkttest....42
9.6. Lösungen für die Benutzeroberfläche aus technischer Sicht....43
9.6.1. Gebäudehierarchie-Ansicht....43
9.6.2. Application Bar....44
9.6.3. Kontextmenü....44
9.6.4. Popups für nicht wiederkehrende Seiten....45
9.6.5. Windows Phone Themes....46
9.6.6. List Picker....46
10. Schlussfolgerungen....47
10.1. Zusammenfassung....47
10.2. Beurteilung der Resultate....47
10.3. Ausblick....48

0.2. Abbildungen

Abbildung 1: Schematische Darstellung der Verdrahtung 5
Abbildung 2: DESIGO Point Test 10
Abbildung 3: Kontroller-Konfigurationswebseite 11
Abbildung 4: Physikalische Systemarchitektur 12
Abbildung 5: Domainmodellübersicht 14
Abbildung 6: Domainmodell 15
Abbildung 7: Logische Architektur 18
Abbildung 8: Ablaufdiagramm - Gebäudehierarchie 19
Abbildung 9: Ablaufdiagramm - Kontroller Suchen 20
Abbildung 10: Ablaufdiagramm - TXIO Module 20
Abbildung 11: Ablaufdiagramm - Service Pin 21
Abbildung 12: Ablaufdiagramm - Flash LED 21
Abbildung 13: SCT-App - Fehlermeldung 24
Abbildung 14: Gateway - Log-View 25
Abbildung 15: Gateway - Architekturübersicht 27
Abbildung 16: Gateway Schnittstelle - Authentifikation 27
Abbildung 17: Gateway Schnittstelle - Engineering-Daten 28
Abbildung 18: Gateway Schnittstelle - Node-Setup 28
Abbildung 19: Gateway UI 30
Abbildung 20: Gateway UI - Kontrollernetzwerk-Auswahl 30
Abbildung 21: Gateway UI - Kontextmenü in der Windows-Taskbar 30
Abbildung 22: Kontrollersimulator Schnittstelle - Authentifikation 31
Abbildung 23: Kontrollersimulator Schnittstelle - Kontrollerinformationen 32
Abbildung 24: Kontrollersimulator Schnittstelle - Datenpunkttest 32
Abbildung 25: Kontrollersimulator - Kontroller bearbeiten 32
Abbildung 26: Kontrollersimulator - Datenpunkte bearbeiten 33
Abbildung 27: Windows Phone Template-Auswahl 34
Abb. 28: Windows Phone Emulator 34
Abb. 29: Entwickler-Phone Registrierung 35
Abbildung 30: SCT-App Architekturübersicht 37
Abb. 31: Gebäude-hierarchie 40
Abb. 32: Gebäudehierarchie 41
Abb. 33: Gerätesuche 41
Abb. 34: Kontextmenü 42
Abb. 35: Kontrollerinformationen 42
Abb. 36: Punktauswahl 42
Abb. 37: Punktübersicht 43
Abb. 38: Punktbearbeitung 43
Abb. 39: Punktinformationen 43
Abbildung 40: Application Bar (zugeklappt) 44
Abbildung 41: Application Bar (aufgeklappt) 44
Abbildung 42: Login Dialog 45
Abb. 43: Schwarzer Hintergrund 46
Abb. 44: Weisser Hintergrund 46
Abb. 45: List Picker für Commissioning State 46

1. Einleitung

1.1. Umfeld

1.1.1. Unternehmen und Produkte

Die Siemens Building Technologies ist eine Siemens-Division mit Hauptsitz in Zug. Siemens Building Technologies bietet verschiedenste Produkte und Lösungen rund um das Thema Gebäude an. Die wichtigsten Bereiche sind Gebäudeautomatisierung, Sicherheit, Brandschutz sowie Energieverteilung.

Im Bereich der Gebäudeautomatisierung umfasst die Produktpalette verschiedene Systeme für die unterschiedlichen Gebäudegrössen und Einsatzgebiete. So wird für mittlere bis grosse Geschäftsanwendungen das Gebäudeautomatisierungssystem DESIGO angeboten. DESIGO besteht aus mehreren Komponenten für die einzelnen Systemebenen, welche das Management, die Automation von Primäranlagen (u.a. Heizung, Lüftung, Klima), die Raumautomation sowie Feldgeräte abdecken. Die nächste Generation der DESIGO-Produktlinie befindet sich zurzeit in Entwicklung und erscheint voraussichtlich Ende 2011.

Neben dem eigentlichen Produkt DESIGO werden auch Softwarelösungen für das Engineering dieser Systeme angeboten. Mit Engineering ist die Konfiguration, Programmierung und Inbetriebnahme der eigentlichen Produkte gemeint. Für das Engineering von Primäranlagen gibt es das Softwarepaket DESIGO XWORKS plus. Für das Engineering auf Raumebene wird zurzeit eine von Grund auf neue Lösung namens System ONE entwickelt.

1.1.2. Raumautomation

Die Raumautomation hat zum Ziel, ein optimales Klima in einzelnen Räumen oder geschlossenen Zonen zu schaffen. Bei der Steuerung und Automation auf Raumebene gibt es drei Komponenten: Bediengeräte/Sensoren, Steuergeräte (Raumkontroller) und Aktoren. Sensoren wie z.B. Temperatursensoren liefern Eingangswerte, die vom Raumkontroller verarbeitet werden. Der Raumkontroller sendet wiederum Ausgangssignale mit denen er die Aktoren, wie z.B. das Licht oder die Sonnenstoren steuert. Vereinfacht entspricht dies dem EVA-Prinzip (Eingabe, Verarbeitung, Ausgabe). Die Logik, die auf dem Raumkontroller für die Verarbeitung der Eingangswerte und die Steuerung der Aktoren zuständig ist, ist mit Hilfe der Engineering-Tools frei programmierbar. Die Raumkontroller werden in der nächsten Produktversion untereinander über ein IP-Netz verbunden und besitzen einen integrierten Webserver für die Konfiguration und Überwachung des Raumkontrollers via Webbrowser.

Beispiele für Sensoren: Temperatur, Feuchtigkeit, Helligkeit

Beispiele für Aktoren (auch Aktuatoren genannt): Licht, Sonnenstoren, Heizung, Klimaanlage

1.1.3. Inbetriebnahme

Die Inbetriebnahme der Systeme beim Endkunden wird durch Siemens-Ingenieure oder Ingenieure eines Systemhauses (VAP) durchgeführt. Die hardwareseitige Installation und Verdrahtung der Geräte im Gebäude erfolgt hingegen nicht durch Siemens-Spezialisten. Diese Arbeiten werden vorgängig von Elektrikern eines Drittunternehmens ausgeführt. Die Qualität dieser Arbeiten kann sehr unterschiedlich sein und führt unter Umständen zu beträchtlichen Mehrkosten. Deshalb ist es wichtig, vorhandene Fehler bei der Installation und Verdrahtung belegen zu können, damit sie dem Elektriker bzw. dem ausführenden Unternehmen nachträglich in Rechnung gestellt werden können. Neben fehlerhaften Verdrahtungen gilt es auch Defekte Sensoren, Kabel und Ein-/Ausgänge zu erkennen.

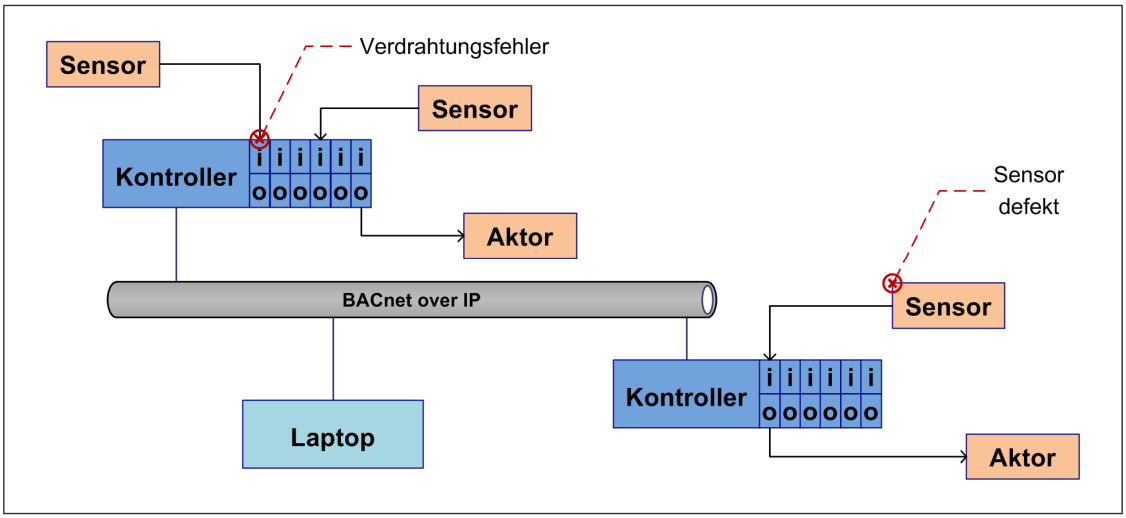


Abbildung 1: Schematische Darstellung der Verdrahtung

Um einen Kontroller in Betrieb zu nehmen, muss als erstes der entsprechende Kontroller im IP-Netz gesucht und identifiziert werden. Danach müssen Konfigurationsdateien in den Kontroller heruntergeladen werden, welche unter anderem die Netzwerkeinstellungen und Datenpunktzuweisungen festlegen. Mit der Datenpunktzuweisung ist die Verknüpfung eines Ein- oder Ausgangs am Kontroller mit dem logischen Datenpunkt in der Software gemeint. Nach dieser initialen Konfiguration des Kontrollers kann die Verdrahtung überprüft und ein Funktionstest gemacht werden. Für die Ein- und Ausgänge am Kontroller muss beim Testen unterschiedlich vorgegangen werden:

Bei den Eingängen wird der Draht auf der Seite des angeschlossenen Sensors kurz abgehängt, was beim Kontroller zu einer Veränderung des Eingangswertes führen sollte. Durch die Veränderung des Eingangswertes kann überprüft werden, ob der Sensor am richtigen Eingang angeschlossen ist. Zusätzlich muss der Eingangswert auch auf seine Plausibilität überprüft werden, um eine mögliche Fehlfunktion des Sensors zu erkennen.

Bei den Ausgängen wird getestet, ob der angeschlossene Aktor auf Kommandos bzw. Wertveränderungen reagiert. Dazu wird der entsprechende Ausgangswert so verändert, dass zum Beispiel das Licht angeht oder die Storen herunterfahren. So kann sehr einfach überprüft werden, ob der entsprechende Aktor funktionstüchtig und am richtigen Ausgang angeschlossen ist.

Zusätzlich wird auch überprüft, ob der richtige Signaltyp und die richtige Werteeinheit konfiguriert sind, da es Feldgeräte von unterschiedlichen Typen gibt. So liefern z.B. Temperatursensoren Analogwerte einer bestimmten Einheit wie °C oder °F. Eine Lampe wird aber mit Binärwerten (ein/aus) kommandiert. Andere Geräte arbeiten wiederum mit einer Reihe von vordefinierten Zuständen wie z.B. schwach, mittel und stark.

Die neue Produktegeneration von Raumkontrollern verfügt über einen integrierten Webserver. So kann die Überwachung und Kommandierung der Ein-/Ausgangswerte direkt über eine kontrollereigene Webseite erfolgen.

Der englische Begriff für die Inbetriebnahme: Commissioning

1.2. Problemstellungen bei der Inbetriebnahme

Generell gilt es bei der Inbetriebnahme die folgenden Aufgaben zu lösen:

- Datenpunktzuweisungen und Programmlogik in den Kontroller laden
- Überprüfung der Verdrahtung:
    - Richtiger Sensor mit richtigem Kontroller-Eingang verbunden
    - Richtiger Aktor mit richtigen Kontroller-Ausgang verbunden
- Defekte Komponenten ausfindig machen

Die aktuellen technischen Gegebenheiten sowie der Engineering-Prozessablauf bei den Siemens-Ländergesellschaften und VAPs verhindern eine völlig automatisierte Inbetriebnahme der Systeme. Es sind zurzeit zwingend manuelle Arbeitsschritte nötig, welche aber, wo immer möglich, durch Softwarelösungen unterstützt werden. Die wichtigsten Punkte des Arbeitsablaufs sind in Kapitel 1.1.3 bereits grob beschrieben. Trotz der vorhandenen oder sich in Entwicklung befindlichen Softwarelösungen, wird der Arbeitsablauf der Inbetriebnahme-Ingenieure noch von einigen Beschwerlichkeiten begleitet. Diese Problemstellungen werden in der untenstehenden Auflistung genauer beschrieben und bilden die Grundlage für diese Bachelorarbeit. Zuerst folgen einige wichtige Begriffsbestimmungen:

Installer
Nachfolgend werden die Inbetriebnahme-Ingenieure als Installer bezeichnet. Für gewöhnlich wird die Inbetriebnahme in Zweierteams durchgeführt. Es kommt aber auch vor, dass nur eine oder aber auch mehrere Personen vor Ort sind.

Datenpunkttest
Die Überprüfung der Verdrahtung zusammen mit dem Funktionstest wird im Folgenden als Datenpunkttest bezeichnet. Für den Datenpunkttest existiert bereits ein Hilfstool namens DPT (DESIGO Point Test) für das Engineering von Primäranlagen. Diese Lösung wird in Kapitel 2 genauer betrachtet. Für die Raumautomation steht die Kontroller-Webseite als Hilfsmittel für den Datenpunkttest zur Verfügung.

Problemstellungen:

- Zusätzliche Software für initiale Netzwerkkonfiguration
  Nicht konfigurierte Kontroller haben eine dynamisch vergebene IP-Adresse und es ist nicht bekannt welcher Kontroller welche IP-Adresse hat. Um die initiale Netzwerkkonfiguration auszuführen, wird ein zusätzliches Tool benötigt, genannt „Discovery & Node Setup Tool“ (DNT), mit welchem die Kontroller über ein UDP Multicast-Protokoll gesucht werden können.

- Umständliches Arbeiten mit IP-Adressen
  Der Installer ist gezwungen, alle IP-Adressen der konfigurierten Kontroller manuell im Browser einzutippen, um auf deren Konfigurationswebseite zugreifen zu können.
- Mühsame Lokalisierung des physikalischen Kontrollers
  Bei der Raumautomation sind die Kontroller im ganzen Gebäude verteilt. Ein Kontroller kann für einen oder mehrere Räume zuständig sein. Um den Datenpunkttest durchzuführen, muss der Installer wissen, welche IP-Adressen zu welchem physischen Kontroller gehört, bzw. für welche Räume der Kontroller zuständig ist.
- Laptop ist zu wenig mobil
  Bei der Inbetriebnahme sind die Feldgeräte bereits im Gebäude installiert und mit dem Kontroller verkabelt. Beim Datenpunkttest muss die korrekte Verdrahtung der Kontroller überprüft werden, wobei diese Tests am Laptop über die Webseite des Kontrollers ausgeführt werden. Während dem Test muss der Installer im Raum selbst überprüfen, ob beim Verändern der Werte auch wirklich das Licht angeht, oder die Lüftung startet. Es ist umständlich, stets mit dem Laptop im Gebäude umherzugehen, die Webseite zu bedienen und gleichzeitig die Verdrahtung an den Feldgeräten zu überprüfen (z.B. Draht von Sensor an der Raumdecke lösen).
- Nicht für mobile Geräte optimierte Kontroller-Webseite
  Zurzeit ist geplant, dass der Datenpunkttest nur über eine für die Benutzung am PC optimierte HTML-Webseite durchgeführt werden kann, welche vom Kontroller direkt bereitgestellt wird. Die Bedienung einer solchen Webseite mit einem Smartphone o.ä. ist nicht benutzerfreundlich.
- Fehlendes Reporting
  Beim Datenpunkttest wird für jeden Datenpunkt der zugehörige Zustand festgehalten. Dieser sogenannte *„Commissioning State“* zeigt, ob ein Datenpunkt bereits überprüft wurde, und falls ja, ob ein allfälliger Verdrahtungsfehler oder Hardwaredefekt vorliegt. Zurzeit existiert noch keine automatisierte Funktion, die einen Report über die Zustände aller Datenpunkte der Kontroller in einem Gebäude generiert.

1.3. Lösungsansatz

Die Problemstellungen lassen sich in zwei Bereiche aufteilen:

- Die Suche und Konfiguration von Kontrollern im Gebäude
- Die Durchführung des Datenpunkttests

Beide Problembereiche verlangen nach einer mobilen Lösung mit intuitiver Eingabe, da sich der Installer ständig von einem Raum zum nächsten durch das Gebäude bewegt. Bei der Siemens Regionalgesellschaft Schweiz wird zurzeit als semi-mobile Lösung ein Rollwagen verwendet, auf dem sich neben Werkzeug auch ein Laptop befindet. Mit diesem Rollwagen kann dann durch das ganze Gebäude gefahren werden. Dabei ist der Laptop ständig mit einem Wireless Access Point verbunden, der an das Kontroller-Netzwerk angeschlossen ist. So wird zwar mit Rollwagen und Laptop jeder Raum erreicht, aber die Sensoren und Kontroller befinden sich oft nicht auf Augenhöhe. So muss zum Teil auf eine Leiter gestiegen werden, wo der Laptop nur schlecht mitgenommen werden kann. Ebenfalls kann der Laptop nur schlecht in einer Hand gehalten werden und wird deshalb auf dem Rollwagen stehen gelassen. So ist der Laptop aber oft nicht im Blickfeld oder zumindest nicht nahe genug. Auch die Bedienung des Laptops mit Maus oder Touchpad ist etwas umständlich in Anbetracht der simplen Tätigkeiten. Aus genannten Gründen bieten sich für die Bedienung die folgenden zwei Gerätetypen an:

- Smartphone (WP7, iPhone, Android)
- Tablet-Computer (iPad, Samsung Galaxy Tab)

Der Hauptnachteil des Smartphones gegenüber einem Tablet-Computer ist der viel kleinere Bildschirm, was die Verwendung über längere Zeit anstrengend machen kann. Der Nachteil des Tablet-Computers ist hingegen die leicht eingeschränktere Portabilität. Ein Tablet-Computer lässt sich zum Beispiel nicht einfach in die Hosentasche stecken, wenn gerade beide Hände anderweitig benötigt werden.

Für den ersten Problembereich, der Suche und Konfiguration von Kontrollern, ist es wichtig, dass die Zugehörigkeit zwischen Räumen und Kontrollern einfach ersichtlich wird. Wenn man sich in einem Raum befindet, soll man möglichst einfach und schnell herausfinden können, welcher Kontroller dafür zuständig ist. Eine mögliche Lösung ist die Darstellung einer Gebäudehierarchie, in welcher die Kontroller den einzelnen Räumen oder Etagen zugewiesen sind. Eine zusätzliche Suchfunktion scheint aber unerlässlich. Dadurch währe die direkte Eingabe von IP-Adressen nicht mehr nötig.

Für den zweiten Problembereich, die Durchführung des Datenpunkttests, ist insbesondere eine intuitive Bedienung gefragt. Dabei sollte der Installer schnell und einfach zu den wichtigsten Informationen gelangen, die

er für den Test benötigt. Er sollte auch direkt die wichtigsten Werte verändern sowie den *Commissioning*-Status mit Kommentar erfassen können. Da man beim Datenpunkttest mit der realen Umgebung, also den Sensoren und Aktoren, interagiert, sollte die unterstützende Software helfen und keine Ablenkung erzeugen. Hier liegt auch ein grosser Vorteil in der Verwendung von mobilen Geräten mit Touch-Bedienung. Denn da lässt sich die Bedienung viel intuitiver gestalten, weil die Darstellung auf das Wesentliche reduziert ist. Der Benutzer sieht nur das, was er auch wirklich für seine Arbeit braucht. Auf einem vollwertigen Desktopsystem drängen sich immer wieder andere Anwendungen und Elemente in den Vordergrund, die nichts mit der aktuellen Tätigkeit zu tun haben.

1.4. Abgrenzung

Diese Bachelorarbeit hat zum Ziel, eine Software auf einem mobilen Gerät zu entwickeln, die den Inbetriebnahmeprozess für die neue Generation von Raumautomationssystemen unterstützt und verbessert. Dabei kann und soll der Inbetriebnahmeprozess nicht grundlegend umgestaltet werden. Vielmehr sollen bestehenden Lücken geschlossen und umständliche Lösungen vereinfacht werden. Die Software wird als Prototyp entwickelt werden und soll zeigen, wie der Einsatz eines mobilen Gerätes den Inbetriebnahmeprozess verbessern kann. Die Software wird bei Siemens zu Demonstrationszwecken eingesetzt werden und die Erfahrungen werden in das endgültige Produkt einfliessen.

Auswahl des mobilen Gerätes
In Bezug auf die Wahl des mobilen Gerätes für die Softwareumsetzung spielt die Plattform eine untergeordnete Rolle, da es sich um eine Prototypen-Software handelt und sich die Auswahl der zur Verfügung stehenden Plattformen schnell ändern kann. Die zu entwickelnde Gesamtlösung sollte möglichst Plattformneutral sein, damit sie später für beliebige Systeme umgesetzt werden kann. Da man bei Siemens gerne einige Erfahrungen mit der Entwicklung für Smartphones sammeln möchte und die vorhandenen Projekte mehrheitlich für die .NET-Plattform umgesetzt werden, ist das neue Windows Phone 7 (WP7) als Zielplattform ausgewählt worden. Zum Projektstart liegt zwar noch keine entsprechende Hardware vor, auf der WP7 läuft. Dafür existiert von Microsoft ein voll funktionsfähiger Emulator, welcher zusammen mit der Visual Studio Entwicklungsumgebung eingesetzt werden kann.

Simulation der Kontroller-Schnittstelle
Veränderungen an der Kontroller-Schnittstelle sind erwünscht, können im Zeitraum der Bachelorarbeit aber nicht direkt umgesetzt werden, da die entsprechende Hardware noch nicht vorhanden ist. Die neue Kontroller-Hardware mit dem Namen PXC3 befindet sich zurzeit in der Entwicklung und es ist ungewiss, wann entsprechende Geräte zur Verfügung stehen werden. Deshalb müssen Abhängigkeiten soweit als möglich vermieden werden. Durch die strikte Simulation jeglicher Kommunikation zu und von den Kontrollern wird die benötigte Unabhängigkeit geschaffen.

Laptop als Gateway
Die erste Version von Windows Phone 7 unterstützt keine Socket-Programmierung und erlaubt deshalb auch keine Kommunikation über das UPD Multicast-Protokoll, welches für die Suche und Konfiguration der Raumkontroller eingesetzt wird. Dadurch muss das sogenannte „Device Discovery“ zwingend auf einem anderen Gerät ausgeführt werden, welches dann die Resultate an das Smartphone weiterleitet. Als solches Gerät bietet sich der bereits vorhandene Engineering-Laptop an, auf dem auch die benötigten Engineering-Daten zur Konfiguration der Kontroller zur Verfügung stehen. Dieser Laptop kann bei Bedarf mit Hilfe eines „Ad Hoc“-Netzwerkes auch als Ersatz für den Wireless Access Point eingesetzt werden. Durch die Funktion als zentraler Einstiegspunkt wird die entsprechende Software auf dem Engineering-Laptop nachfolgend als Gateway bezeichnet.

RESTful Webservices
REST ist ein Softwarearchitekturstil, der sich zurzeit grosser Popularität erfreut, wenn es um die Entwicklung plattformunabhängiger Webservices geht. Gerade bei der Entwicklung für mobile Plattformen bietet REST enorme Vorteile, welche in Kapitel 5.2.2 genauer beschrieben sind. REST wird bei dieser Bachelorarbeit als zugrundeliegender Architekturstil für Design und Umsetzung der Webservices verwendet.

Herausforderungen
Die Herausforderungen dieser Bachelorarbeit finden sich in den folgenden Punkten:

- Analyse des Umfeldes und der bestehenden Lösungen
- Erarbeiten der Anforderungen unter Berücksichtigung der Interessen und Bedürfnisse der Stakeholder
- Erarbeitung eines Machbarkeitsnachweises für den physikalischen Systemaufbau
- Erfahrungen mit der Entwicklung auf Windows Phone 7 sammeln mit Fokus auf die Benutzerschnittstelle
- Design einer universellen Webservice-Schnittstelle für den Datenpunkttest
- Simulation der Kontroller-Kommunikation

1.5. Aufgabenstellung

Als neue Client/Server Anwendung soll eine Applikation für Windows Phone 7 entwickelt werden, welche mit einem ebenfalls zu entwickelnden Webservice (C#) kommuniziert, der auf einem Windows XP/Vista/7 Laptop läuft.

Die WP7 Applikation bietet mindestens folgende Funktionalität:

- Suchen, filtern, auflisten von Raumkontrollern in einem IP-Netz
- Senden von Wink an einen Raumkontroller (LED leuchtet auf)
- Empfangen von Service-Pins auf dem Smartphone
- Vereinfachter und an die Anforderung eines Smartphones angepasster Datenpunkttest

Der Webservice (C#) auf dem Laptop stellt die von der WP7 Applikation benötigten Funktionalitäten via REST zur Verfügung. Zur Abkoppelung bzw. Aufbau einer Simulationsumgebung müssen darüber hinaus folgende Teile implementiert werden:

- Simulation der UDP Device Discovery API für das Suchen der Raumkontroller (allenfalls später ersetzt durch echte API sobald verfügbar)
- Simulation der Raumkontroller-Schnittstelle für den Datenpunkttest

Die wichtigsten Resultate der Bachelorarbeit stellen folgende Artefakte dar:

- Anforderungen mit UseCases
- Schnittstellenbeschreibung des Webservices für den Datenpunkttest bzw. Umsetzung als Prototyp
- GUI-Studie für den Datenpunkttest bzw. Prototyp für Windows Phone 7
- Machbarkeitsnachweis für den physikalischen Aufbau der Netzwerkkommunikation mit einem Laptop als Gateway
- (optional) Vergleich von Webtechnologien: REST/SOAP/WCF/Thrift etc.
- (optional) Sicherheitskonzept (Authentifizierung, Autorisierung, Vertraulichkeit)

2. Analyse der bestehenden Lösungen

2.1. DESIGO Point Test (DPT)

Zum Softwarepacket DESIGO XWORKS plus für das Engineering von Primäranlagen gehört u.a. das Inbetriebnahme-Tool DESIGO Point Test (DPT). Das DPT bietet bereits einige Vorteile gegenüber der herkömmlichen Inbetriebnahme. Vor dem DPT waren immer zwei Personen nötig, um den Verdrahtungstest für Signaleingänge durchzuführen. Eine Person die am Laptop selbst den Wert eines Signaleinganges beobachtete und eine, die das Kabel am Sensor entfernte oder einen Schalter betätigte. Zusätzlich mussten sich beide Personen in Sichtweite befinden bzw. gut verständigen können, um koordiniert Punkt für Punkt zu testen.

Das DPT hat den Begriff *„One-Man-Commissioning“* eingeführt. *„One-Man-Commissioning“* bedeutet, dass neu eine Person alleine die Inbetriebnahme inklusive dem Verdrahtungstest durchführen kann. Ermöglicht wird dies durch eine neu eingeführte Trendfunktion. Der Installer kann für den zu testenden Signaleingang eine Trend-Aufzeichnung starten, das Kabel beim Sensor kurz entfernen und auf dem Trend überprüfen, ob die Wertänderung sichtbar ist. Möglich ist auch mehrere Signaleingänge gleichzeitig zu testen, jedoch nur wenn sie klar unterschieden werden können. Im Nachfolgenden Screenshot wurden zwei Kippschalter getestet, einer mit nur zwei Stufen und einer mit mehreren Stufen.

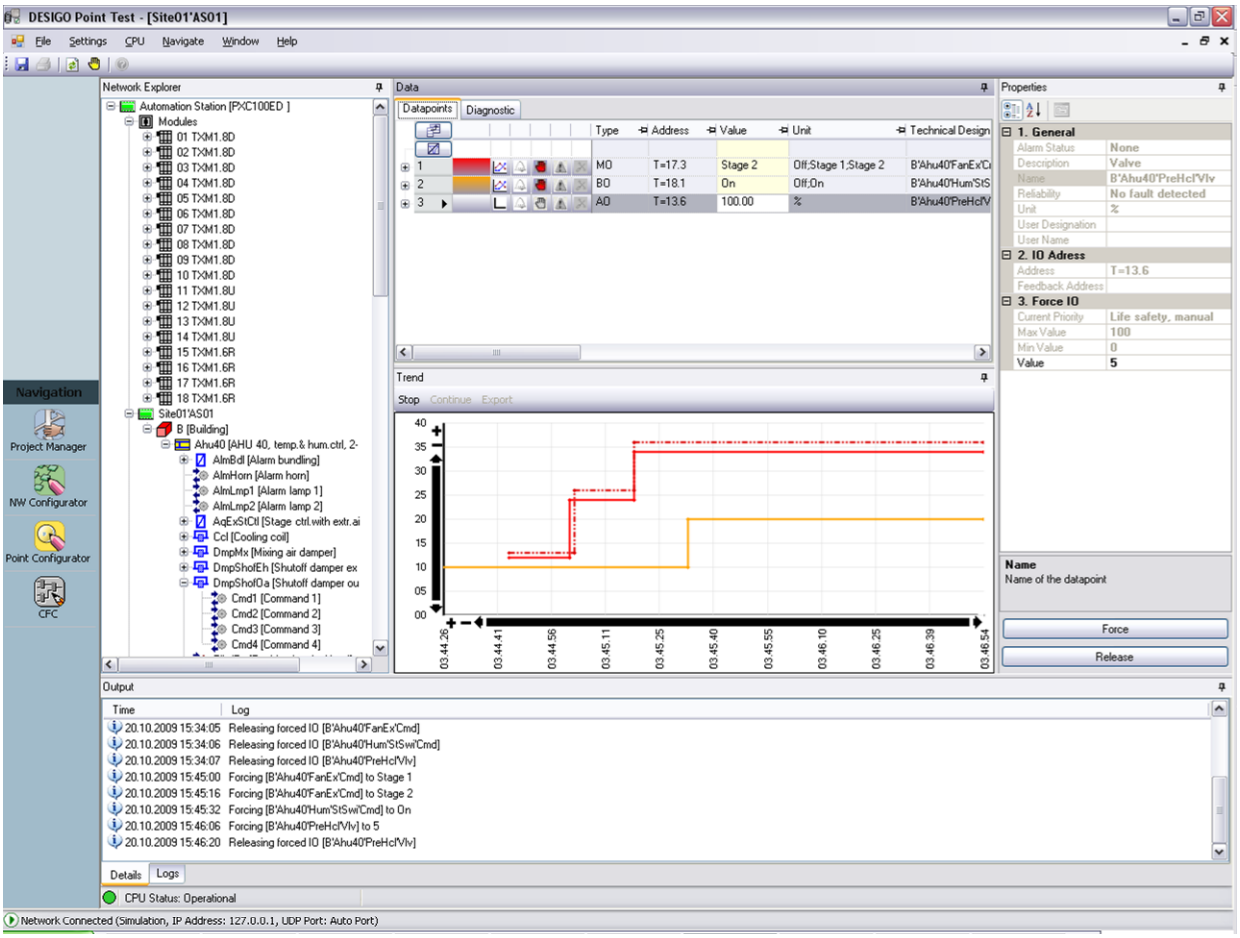


Abbildung 2: DESIGO Point Test

Vorteile:

- Kostenersparnis weil für die Inbetriebnahme nur noch ein Installer nötig ist.
- Keine Vollinstallation des Engineering-Tools XWORKS plus nötig.

Nachteile:

- Installer muss den Laptop weiterhin in Reichweite haben.

2.2. Kontroller-Konfigurationswebseite (in Entwicklung)

Das in Entwicklung stehende System für das Engineering auf Raumebene (System ONE) verfolgt einen anderen Ansatz als das XWORKS plus. Im System ONE werden die Kontroller dahingehend erweitert, dass jeder Kontroller selbst eine Webseite für die Konfiguration und die Inbetriebnahme zur Verfügung stellt. Über die Eingabe der IP-Adresse des Kontrollers im Webbrowser gelangt der Installer auf die Konfigurationswebseite und kann einen Verdrahtungstest durchführen. Falls der Kontroller netzwerkseitig noch nicht konfiguriert ist, oder die IP-Adresse unbekannt ist, steht das *„Discovery & Node Setup Tool“* (DNT) zur Verfügung. Das DNT wird für das Suchen der Kontroller und die Netzwerkkonfiguration benötigt.

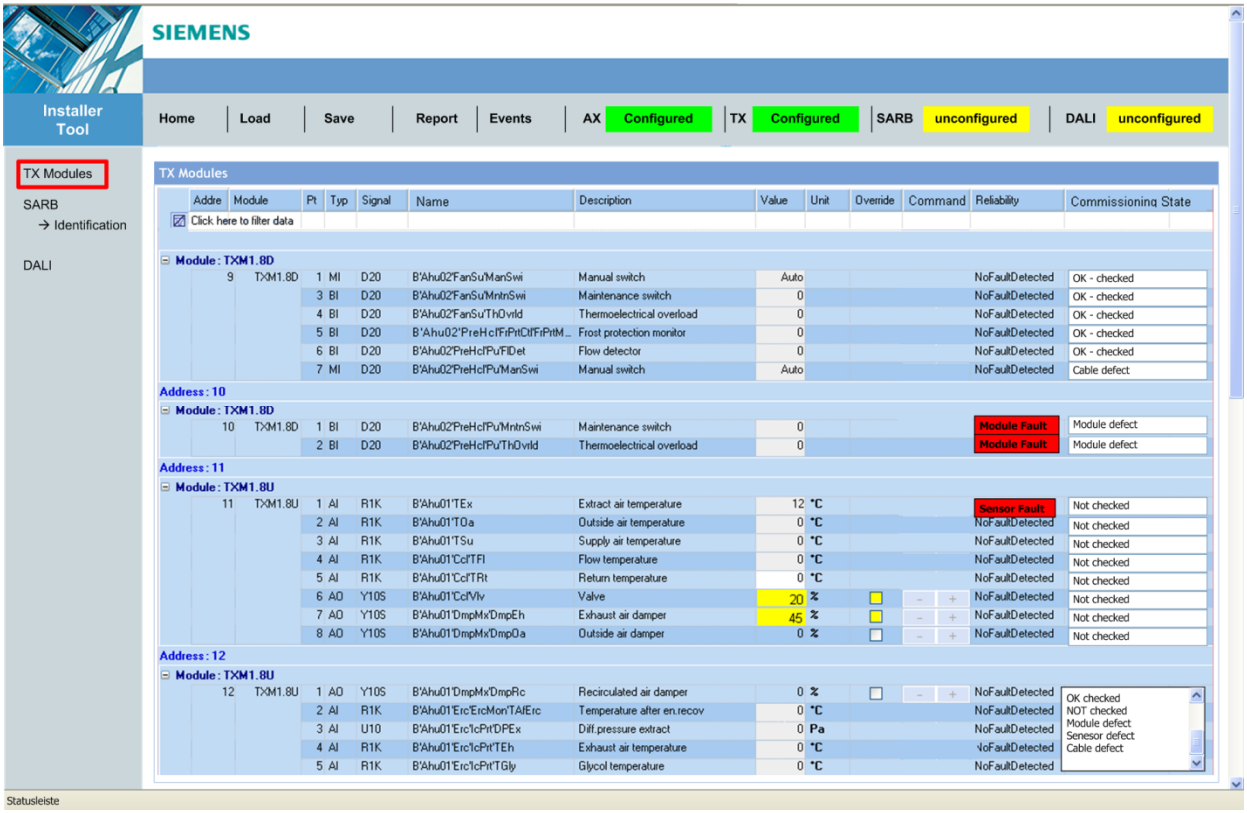


Abbildung 3: Kontroller-Konfigurationswebseite

Vorteile:

- Keine Installation nötig, der Installer kann direkt über den Browser auf die Konfigurationswebseite zugreifen.

Nachteile:

- Installer muss den Laptop weiterhin in Reichweite haben.
- IP-Adresse des Kontrollers muss bekannt sein, oder das DNT wird benötigt, um die Kontroller zu suchen.
- *„One-Man-Commissioning“* ist nur möglich, wenn zusätzlich eine Trendfunktion eingebaut wird.

2.3. Schlussfolgerungen der Analyse

Die Nachteile der bestehenden Lösungen sowie Gespräche mit Produktdesignern und Ländervertretern dienten als Input für eine neue Lösung: Die WP7 Applikation mit dem Namen *„Smart Commissioning Tool“* (SCT).

- Für den Verdrahtungstest kann neu ein Smartphone benutzt werden, welches der Installer im Gegensatz zum Laptop überallhin mitnehmen kann.
- Die neu gewonnene Mobilität erlaubt weiterhin ein *„One-Man-Commissioning“*.
- Der Laptop bleibt an einem zentralen Ort und fungiert nur als Gateway, bzw. Zugriffspunkt für Funktionen, welche auf dem Smartphone nicht möglich sind.

Mehr hierzu in den nachfolgenden Kapiteln.

3. Systemübersicht

3.1. Physikalische Systemarchitektur

Folgende physikalische Architektur ist als Teil der Bachelorarbeit erarbeiten worden.

Technische Einschränkungen und Gegebenheiten:

- Die Kontroller sind über ein IP-Netz miteinander verbunden.
- Mobile Geräte haben häufig keinen Netzwerkanschluss, die Kommunikation über ein IP-Funknetz ist jedoch möglich.
- Einige mobile Geräte schränken die erlaubten Kommunikationsprotokolle stark ein. Beim WP7 ist Beispielsweise nur die Kommunikation über das HTTP-Protokoll erlaubt.

Für das Auffinden von Kontrollern *(Device Discovery)* entwickelte die Siemens das DICP-Protokoll, welches weiter unten kurz erklärt wird. Durch die eingeschränkten Möglichkeiten auf dem WP7 ist es nicht möglich das DICP-Protokoll direkt auf dem mobilen Gerät zu benutzen. Eine Lösung ist einen Laptop (Gateway) ins IP-Netz mit den Kontrollern zu hängen, auf welchem das DICP-Protokoll läuft und die Informationen als Webservice den mobilen Geräten zur Verfügung stellt.

Das System besteht nun aus den drei Hauptteilen Kontroller, Engineering-Laptop und Smartphone welche über ein IP-Netz miteinander kommunizieren. Der Engineering-Laptop ist entweder über ein drahtgebundenes Netz oder über ein Funknetz mit den Kontrollern verbunden und das Smartphone kommuniziert ausschliesslich über ein Funknetz mit dem Engineering-Laptop und den Kontrollern.

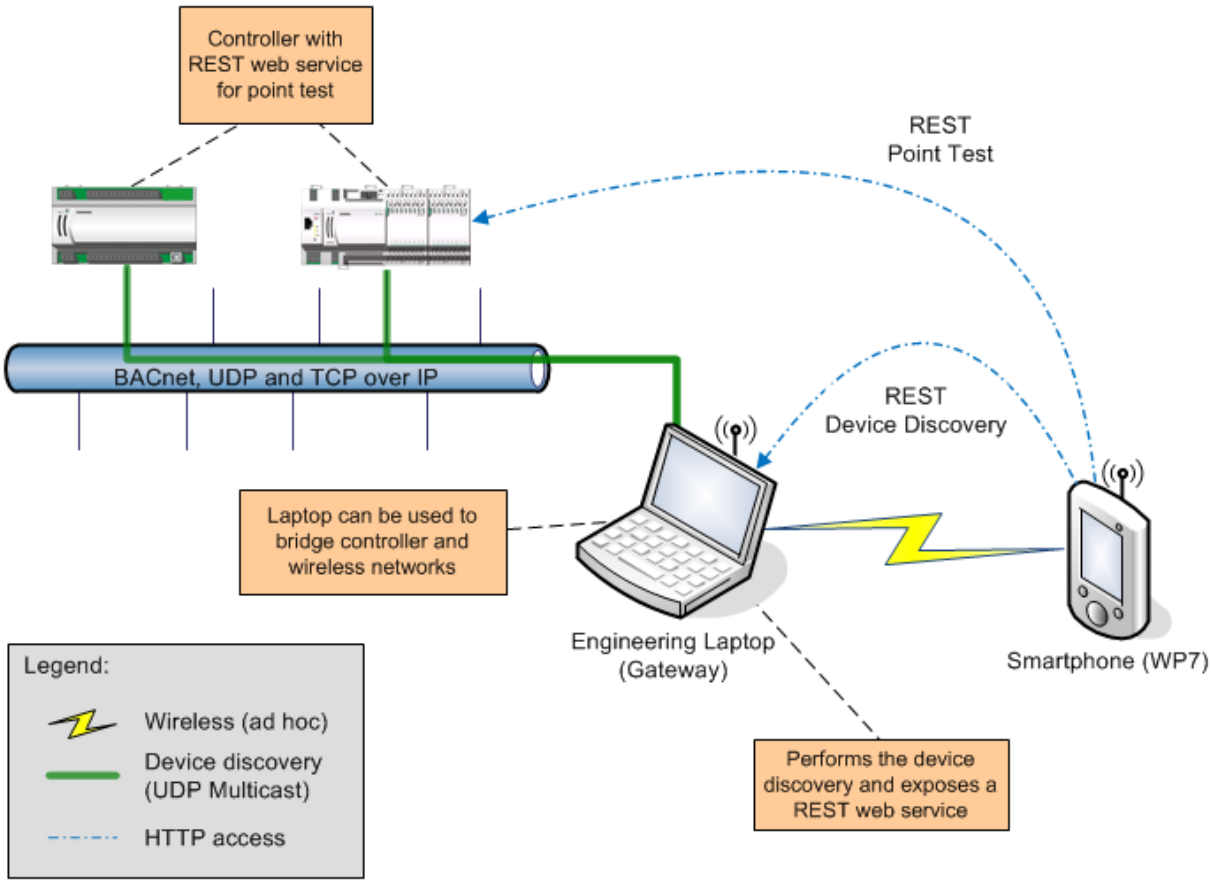


Abbildung 4: Physikalische Systemarchitektur

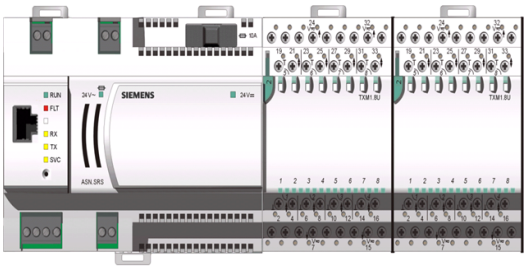
DICP - Device Identification and Configuration Protocol (UDP Multicast)
Für das Auffinden der Kontroller vom Engineering-Laptop aus wird das DICP Protokoll verwendet. DICP ist ein von der Siemens entwickeltes Protokoll für das Identifizieren und Konfigurieren von Kontrollern basierend auf UDP-Multicast-Anfragen. Mehr über DICP im Kapitel 5.2.5.

REST - Representational State Transfer (HTTP)
Das Smartphone kommuniziert ausschliesslich über HTTP mit den RESTful Webservices auf dem Engineering-Laptop (Gateway) für das Auffinden der Kontroller und mit den RESTful Webservice auf dem Kontroller selbst für den Datenpunkttest. Mehr über REST im Kapitel 5.2.2

BACnet - Building Automation and Control Networks
Die Kommunikation zwischen Engineering-Laptop und Kontroller oder weiteren Geräten von Drittherstellern geschieht während dem Engineering sowie im laufenden Betrieb über BACnet. BACnet ist ein herstellerunabhängiges und standardisiertes Netzwerkprotokoll für die Gebäudeautomation.

3.2. Hardware-Sicht

Die folgenden Informationen zur Hardware helfen die logische Sicht zu verstehen.



Kontroller mit zwei Modulen
Die neuen Kontroller (Raumsteuergeräte) haben einen Feldbus, an den mehrere TXIO Module (Erweiterungsmodule) angeschlossen werden können. Die Module haben häufig 8 oder 16 Signaleingänge oder Signalausgänge. Die Eingänge können mit den Bediengeräten oder Sensoren verbunden werden und die Ausgänge mit den Aktoren.

Aussentemperaturfühler
Ein Sensor wie zum Beispiel der Aussentemperaturfühler wird direkt mit einem Signaleingang an einem TXIO Modul eines Kontrollers verbunden. Ein Signal von einem Sensor ist entweder analog, binär oder mehrstufig. Ein mehrstufiges Signal ist ein Signal bei dem die Stufen definiert sind (z.B. Low, Medium, High) und keine Zwischenwerte wie bei einem analogen Signal möglich sind.

Raumbediengerät
Bediengeräte wie z.B. Schalter werden wie auch die Sensoren direkt mit einem Signaleingang am TXIO Modul verbunden. Sie liefern Eingangswerte, welche zusammen mit der Steuerungslogik im Kontroller die Ausgangswerte zu den Aktoren festlegen.

3.3. Logische Sicht

Es gibt die zwei Hauptbereiche Engineering und Onlinesystem. Das Engineering beinhaltet Konzepte wie die Gebäudehierarchie, welche nur im ABT, dem Engineering-Tool, existieren. Die Gebäudehierarchie ist später online im Kontroller nicht mehr verfügbar. Zum Onlinesystem gehört alles was in den physikalischen Kontroller geladen wird und somit auf dem Kontroller auch verfügbar ist.

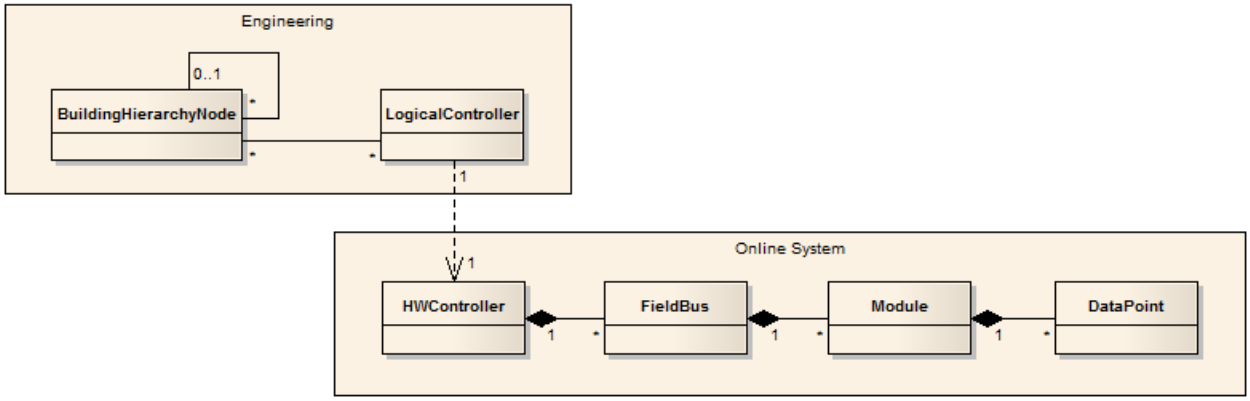


Abbildung 5: Domainmodellübersicht

Engineering:

- Die Gebäudehierarchie (*BuildingHierarchyNode*) ist die hierarchische Unterteilung eines Gebäudekomplexes in Gebäude, Stockwerke und Räume.
- Der logische Kontroller (*LogicalController*) ist die Information über einen Kontroller, welche bereits im Engineering-Tool eingegeben wird, bevor die zugehörige Hardware konfiguriert wurde. Nur über den logischen Kontroller ist ersichtlich, welcher Kontroller für welche Räume zuständig ist. Im Onlinesystem sind keine Informationen zur Gebäudehierarchie verfügbar.

Onlinesystem:

- Der Hardwarekontroller (*HWController*) repräsentiert den physikalischen Kontroller.
- Ein Hardwarekontroller hat mehrere Feldbusse (*FieldBus*), welche es ermöglichen den Kontroller mit Modulen wie z.B. einem TXIO Modul zu erweitern.
- Die Module (*Module*) werden in einer Reihe aneinandergehängt und erweitern so den Feldbus welcher die Verbindung zwischen den Modulen und dem Kontroller darstellt. Die Module sind zuständig für den Übergang von logischen Adressen zu physikalischen Ein- und Ausgängen.
- Ein Modul hat mehrere Ein- und Ausgänge, welche Datenpunkte (*DataPoint*) genannt werden. Ein Datenpunkt kann wie das darunter liegende Signal analog, binär oder mehrstufig sein. Bei einem mehrstufiger Datenpunkt sind wie beim mehrstufigen Signal die Stufen definiert und keine Zwischenwerte möglich.

3.4. Domainmodell

Folgendes Modell zeigt eine vereinfachte Darstellung der Objekte der realen Welt mit ihren Beziehungen. Es beinhaltet dieselben Konzepte wie die Domainmodellübersicht, jedoch mit zusätzlicher Komplexität und zusätzlichem Detailierungsgrad.

Das Modell ist als Teil der Bachelorarbeit erarbeitet worden. Die Informationen, die als Grundlage für das Modell dienen, sind während der Systemanalysephase aus verschiedenen Architekturdokumenten zusammengetragen worden. Zum Teil dienten auch XML-Dateien sowie die bestehenden Applikationen selbst als Informationsquelle. Es ist mehrmals mit den verantwortlichen Systemarchitekten besprochen und überarbeitet worden.

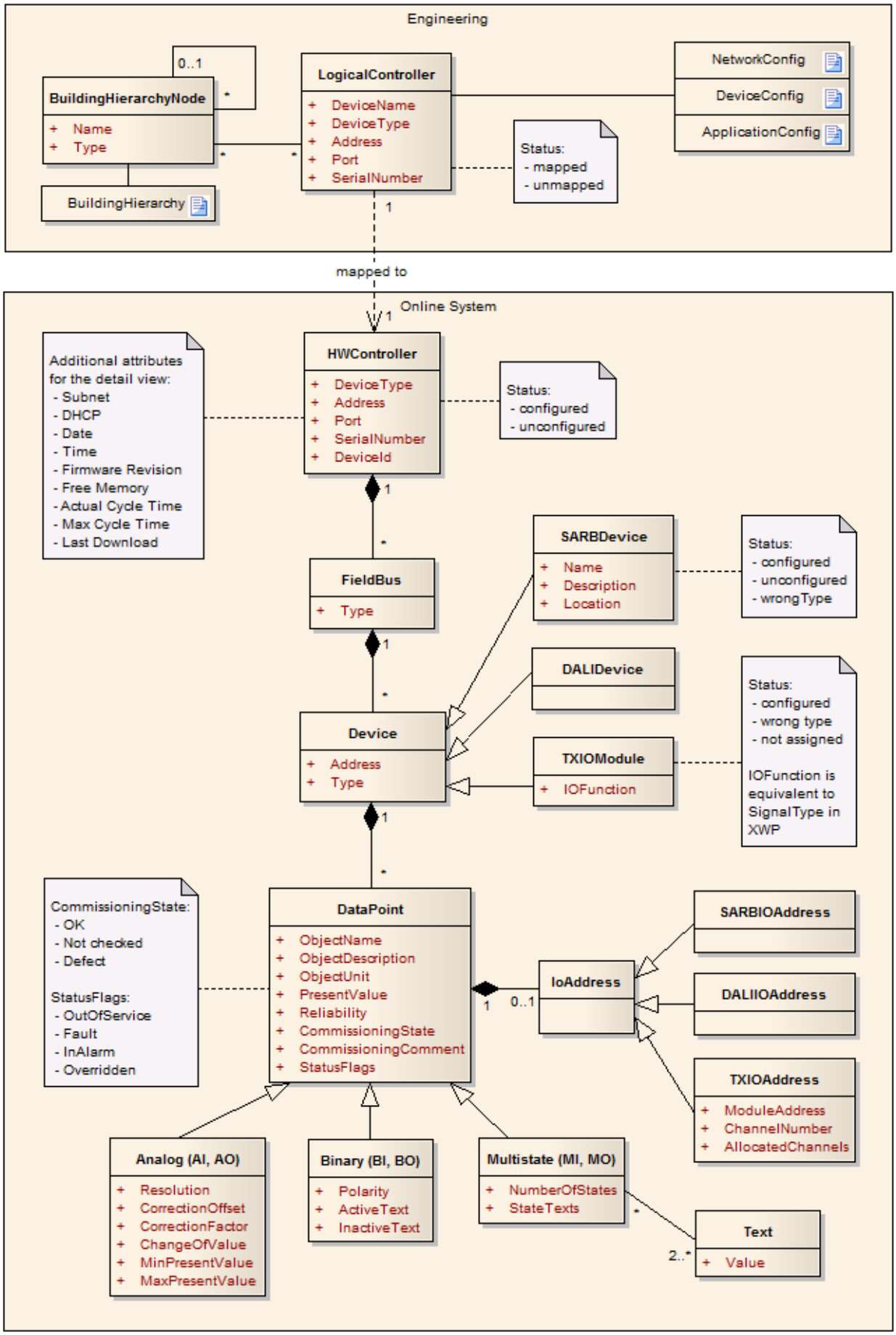


Abbildung 6: Domainmodell

4. Anforderungen

Vor den eigentlichen Anforderungen sind zuerst das Umfeld und die Arbeit beim Datenpunkttest selbst, sowie die nötigen Arbeitsschritte mit der bestehenden Software analysiert worden. Viele nützliche Informationen konnten direkt aus der Praxis von einer Kontaktperson der Schweizer-Ländervertretung eingeholt werden. Die Basis für die Anforderungen war somit bereits ermittelt. Detailinformationen konnten Siemens-Intern, durch Gespräche mit Produktmanagern, Systemarchitekten und Entwicklern der bestehenden Software eingeholt werden. Es wurde grossen Wert auf eine saubere Ermittlung der Anforderungen gelegt. Dieses Kapitel ist nur eine kurze Auflistung der wesentlichen Punkte, eine detaillierte Beschreibung und die kompletten Anwendungsfälle sind in der Entwicklungsdokumentation unter Anforderungen einsehbar.

4.1. Funktionale Anforderungen

Kurze Auflistung der Funktionalen Anforderungen:

- Durchsuchen der Gebäudehierarchie und der für die Räume zuständigen Kontroller.
- Suchen nach Kontrollern von einem definierten Kontroller-Typ.
- Konfigurieren eines Kontrollers und löschen der Konfiguration (optional).
- Das Drücken der *Service Pin* Taste am Kontroller auf dem Smartphone sichtbar machen.
- Ein *Flash LED* vom Smartphone aus an den Kontroller senden, damit auf dem Kontroller eine LED aufleuchtet.
- Durchsuchen der Module und Datenpunkte von einem Kontroller.
- Durchführen des Datenpunkttestes für einen ausgewählten Datenpunkt.
- Setzen des Inbetriebnahme-Status und Kommentar für einen Datenpunkt.
- Generieren eines Inbetriebnahme-Reports (optional).

4.2. Anwendungsfälle (Use Cases)

Folgende Auflistung zeigt welche Anwendungsfälle implementiert und erfolgreich getestet wurden. Die Dokumentation der Anwendungsfälle ist in Englisch geschrieben, weshalb auch die UseCase-Titel in Englisch sind.

Anwendungsfall (Use Case)	*Implementiert*	*Erfolgreich getestet*
UC01: Start web service on gateway	ja	ok
UC02: Connect gateway	ja	ok
UC03: Browse RA controllers	ja	ok
UC04: Map RA controller to corresponding hardware	ja	ok
UC05: Remove controller - hardware mapping	nein, optional	-
UC06: Search RA controllers with filter	ja	ok
UC07: Configure RA controller	nein, optional	-
UC08: Unconfigure RA controller	nein, optional	-
UC09: Send wink to RA controller	ja	ok
UC10: Receive service pin	ja	ok
UC11: Show controller details	ja	ok
UC12: Automatically search corresponding hardware	nein, optional	-
UC13: Download Device- and Application Configuration	nein, optional	-
UC14: Browse data points of an RA controller	ja	ok
UC15: Perform wiring test for a selection of points	ja	ok
UC16: Set commissioning state	ja	ok
UC17: Add commissioning comment	ja	ok
UC18: Release all commanded objects	nein, optional	-
UC19: Generate commissioning report	nein, optional	-
UC20: Show alarm	nein, optional	-
UC21: Confirm alarm	nein, optional	-

4.3. Nichtfunktionale Anforderungen

Allgemeine Überlegungen und daraus resultierende Anforderungen:

- Die Installer welche den Datenpunkttest durchführen verfügen unter Umständen über geringe PC-Kenntnisse.
  - ➔ Gute Benutzerführung und eine möglichst einfache Bedienung.
- Die Benutzerschnittstelle soll unterstützend sein und nicht von der eigentlichen Arbeit ablenken.
  - ➔ Jede Benutzeroberfläche ist schlicht und zeigt nur das Wesentliche.
- Die Installer müssen bei gewissen Sensoren auf eine Leiter steigen, um Zugriff zum Sensor zu erlangen.
  - ➔ Applikation muss mit nur einer Hand bedienbar sein.

Bedienbarkeit
Die Bedienung soll sich soweit möglich an der in Entwicklung stehenden Konfigurationswebseite orientieren. Trotzdem sollen die Designprinzipien für Windows Phone 7 Applikationen berücksichtig werden. Durch die reine Touch-Bedienung und die kleinere Auflösung, wird sich die Bedienung jedoch von der Konfigurationswebseite und vom DPT unterscheiden.

Leistung
Die Ansprechbarkeit der Benutzerschnittstelle soll ohne wahrnehmbare Verzögerung erfolgen und zu jedem Zeitpunkt gegeben sein. Falls eine Aufgabe mehr Zeit in Anspruch nimmt, muss dies z.B. durch einen Fortschrittsbalken klar ersichtlich sein.

Die Anzahl von Smartphones, welche gleichzeitig mit dem Gateway oder einem Kontroller verbunden sind, soll nicht auf eine exakte Zahl limitiert werden. Im Normalfall stellt jedoch nur ein einziges Smartphone eine Verbindung her. In seltenen Fällen vielleicht einmal zwei oder drei.

Datenschutz
Die Windows Phone 7 Applikationen soll durch eine Benutzername/Passwort-Kombination geschützt werden. Die Verbindung zwischen dem Smartphone und dem Gateway soll vor *Sniffing*-Attacken geschützt werden.

Datensicherheit
Datensicherheit oder Backupstrategien sind für dieses Projekt nicht relevant. Alle Informationen kommen entweder von den Engineering-Daten oder direkt vom Kontroller.

Wartbarkeit
Die Software soll einfach erweiterbar sein und das Hinzufügen von neuen Funktionen erlauben.

5. Architektur und Design

5.1. Komponentenübersicht

Das folgende Modell zeigt die vereinfachte logische Architektur der *Smart Commissioning* Softwarekomponenten. Es veranschaulicht welche Webservices in welcher Softwarekomponente verfügbar sind und wie die einzelnen Komponenten miteinander interagieren. Zusätzlich zeigt es welche XML Dateien von welchen Komponenten als Datenquellen benutzt werden.

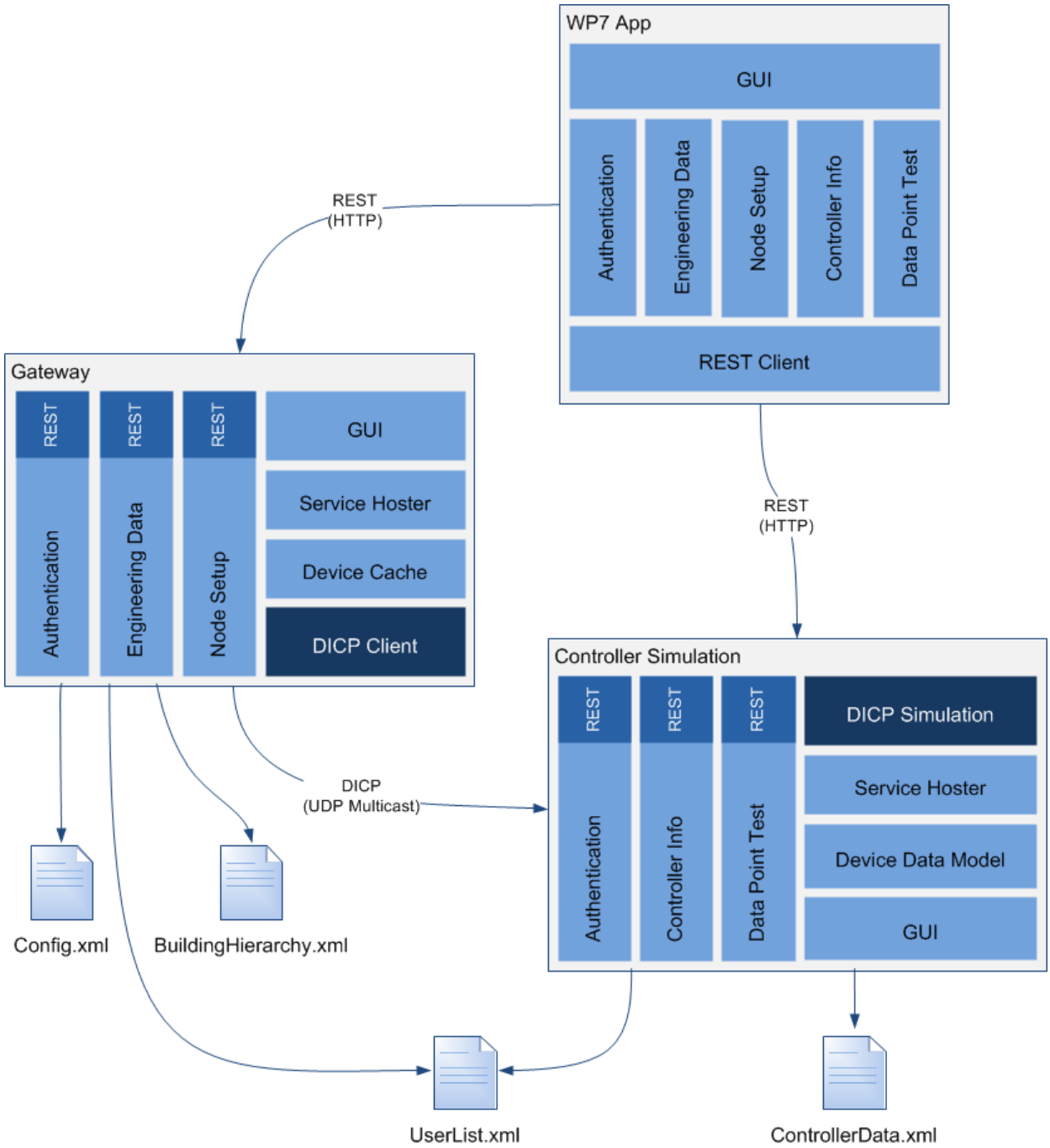


Abbildung 7: Logische Architektur

WP7 Application (SCT)
Die WP7 App ist das *Smart Commissioning Tool*, welches auf dem Windows Phone 7 läuft. Alle benötigten Informationen kommen direkt von den REST-Services des Gateways und den Kontrollen. Somit ist kein eigener Datenspeicher nötig. Die WP7 App stellt zuerst eine Verbindung mit dem Gateway her für eine Benutzerauthentifizierung, um die Gebäudehierarchie darzustellen und um nach Kontrollern zu suchen. Für

detaillierte Kontroller-Informationen oder um einen Datenpunkttest durchzuführen, ist eine Verbindung zum REST-Service des Kontrollers selbst nötig. Die Kontroller selbst werden vom Kontrollersimulator simuliert.

Gateway (SCG)
Der Gateway ist zuständig für die Benutzerauthentifizierung, das zur Verfügung stellen der Engineering-Daten (Gebäudehierarchie) und das Ausführen von *Node-Setup* Funktionen (Auffinden von Kontrollern, Konfigurieren, *Flash LED* und *Service Pin*). Der Gateway greift für die Benutzerauthentifizierung auf die Datei UserList.xml zu und für die Engineering-Daten auf die Datei BuildingHierarchy.xml. Um *Node-Setup* Funktionen durchzuführen wird der DICP-Client benutzt.

Controller Simulation (CST)
Der Kontrollersimulator *(Controller Simulation)* hat zwei Zuständigkeiten. Erstens ist er dafür zuständig den Kontrollerteil des DICP-Protokolls zu simulieren, damit die simulierten Kontroller beim Suchen von Kontrollern mit dem echten Protokoll auch erscheinen. Zweitens ist er dafür zuständig die REST-Services von jedem simulierten Kontroller zu simulieren, damit ein Punkttest möglich ist. Der Kontrollersimulator stellt eine Benutzerschnittstelle zur Verfügung, um Kontroller hinzuzufügen, zu editieren und zu löschen. Im Weiteren können die REST-Services gestartet und beendet werden. Für die Simulation des Datenpunkttestes können Module und Datenpunkte hinzugefügt, editiert und gelöscht werden.

Config.xml
Die Datei config.xml enthält einige Grundeinstellungen für den Gateway.

BuildingHierarchy.xml
Die Datei BuildingHierarchy.xml ist ein Export aus dem Engineering-Tool (ABT). Sie beinhaltet die komplette Gebäudehierarchie mit Gebäuden und Gebäudeelementen (Stockwerke, Räume). Zusätzlich beinhaltet sie eine Liste von logischen Kontrollern mit einigen Offline-Informationen zu den Kontrollern und ihrer Beziehung zu einem oder mehreren Gebäudeelementen.

UserList.xml
Die Datei UserList.xml beinhaltet Benutzerinformationen für eine einfache Authentifizierung (Benutzername, Vorname, Nachname und Passwort). Diese Informationen werden benutzt, um eine einfache Authentifizierung zu simulieren.

ControllerData.xml
Die Datei DeviceData.xml ist die interne Speicherdatei für alle online Informationen zu den Kontrollern. Sie beinhaltet alle Informationen über die Kontroller, die Module und die Datenpunkte, welche für die Simulation des DICP-Protokolls und den Datenpunkttest notwendig sind.

5.2. Kommunikation

5.2.1. Übersicht

Die folgenden Diagramme zeigen die Kommunikation auf Netzwerkebene zwischen der SCT-App (WP7 App), dem Gateway und dem Kontrollersimulator. Um die Multicast-Nachrichten zu veranschaulichen wird zwischen dem Gateway und den Kontrollern noch das Netz (*Network*) dargestellt. Das Netz steht für alle Router und Switches und verteilt die Multicast-Nachrichten. Damit die Diagramme übersichtlich bleiben, werden bei den Multicast-Antworten die Pfeile für Nachrichten nur zum richtigen Empfänger gezeichnet. Im Gateway sowie in den simulierten Kontrollern laufen jeweils ein REST-Service und ein DICP-Service. Der REST-Service ist für die Kommunikation mit der SCT-App und der DICP-Service für die *Node-Setup* Funktionen.

Das folgende Ablaufdiagramm zeigt die Anfrage für die Gebäudehierarchie von der SCT-App aus an den Gateway.

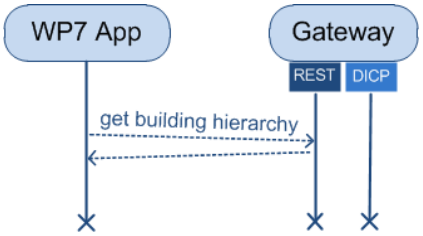


Abbildung 8: Ablaufdiagramm - Gebäudehierarchie

Die Anfrage für das Suchen nach Kontrollern wird ebenfalls von der SCT-App an den Gateway gesendet. Der Gateway startet daraufhin einen *„DICP device discovery request“* und alle Kontroller antworten nach einem zufälligen Warten (simuliert die Dynamik der Kontroller). Der Gateway wartet, nachdem die erste Antwort zurückkommt, noch einige Millisekunden und gibt danach der SCT-App die bis dahin gefundenen Kontroller zurück. Einige Millisekunden länger wird gewartet, damit in der ersten Antwort nicht immer nur ein Kontroller enthalten ist. Nachdem die SCT-App die Antwort bekommen hat, wartet sie zwei Zehntelsekunden und sendet danach eine Anfrage an den Gateway, um die zusätzlich gefundenen Kontroller zu bekommen. Der letzte Vorgang wird von der SCT-App einige Mal wiederholt.

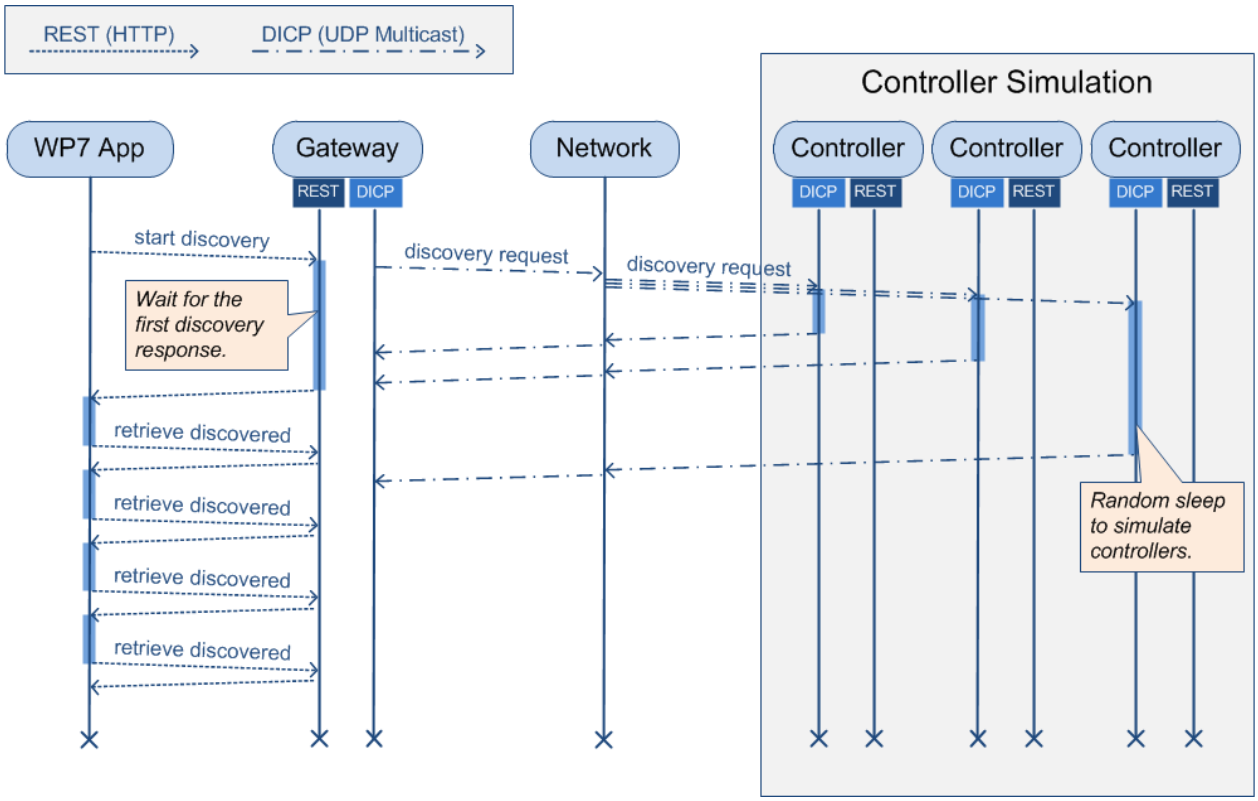


Abbildung 9: Ablaufdiagramm - Kontroller Suchen

Nach dem Suchen nach Kontrollern sind die IP-Adressen der Kontroller bekannt und die SCT-App kann direkt mit dem REST-Service eines Kontrollers kommunizieren. Im folgenden Ablaufdiagramm wird gezeigt wie die Modulliste direkt von einem Kontroller angefordert wird.

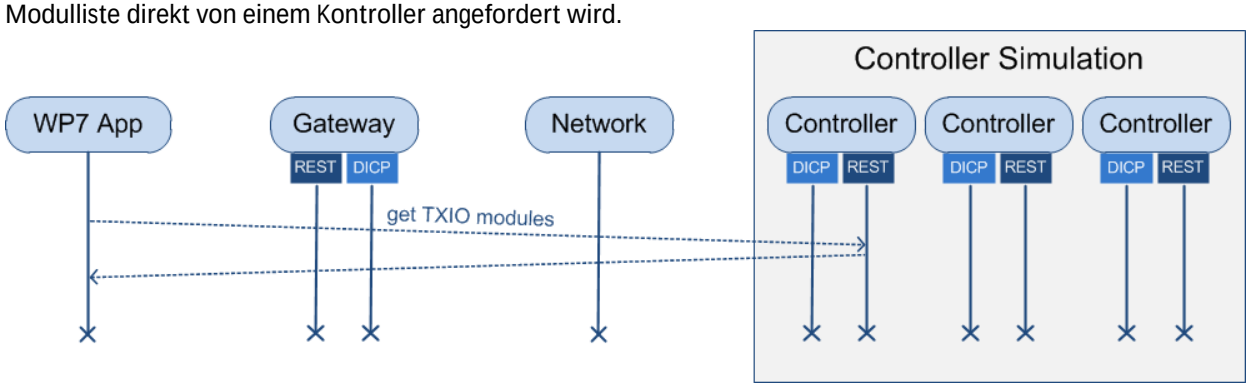


Abbildung 10: Ablaufdiagramm - TXIO Module

Das nachfolgende Ablaufdiagramm zeigt wie die SCT-App beim Drücken der *Service Pin* Taste am Kontroller benachrichtigt wird. Die Anfrage für das Warten auf *Service Pins* wird an den Gateway gesendet. Der Gateway lässt die Anfrage solange offen bis ein *Service Pin* ankommt oder das Timeout von z.B. 30 Sekunden abgelaufen ist. Mehr hierzu im Kapitel 5.2.3 Benachrichtigung mit REST - Long Poll.

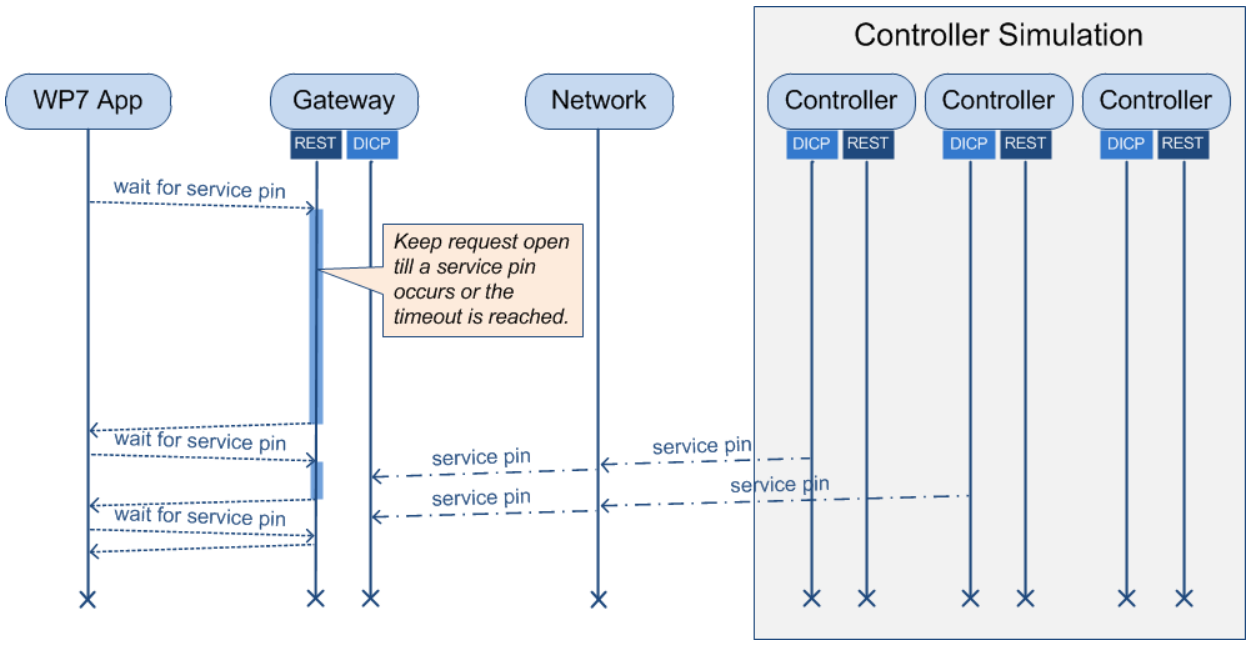


Abbildung 11: Ablaufdiagramm - Service Pin

Das Senden einer *flash LED* Anfrage geht von der SCT-App über den Gateway an alle Kontroller. Jedoch reagiert nur der Kontroller, mit welchem die Seriennummer übereinstimmt, indem er ein LED-Lämpchen leuchten lässt.

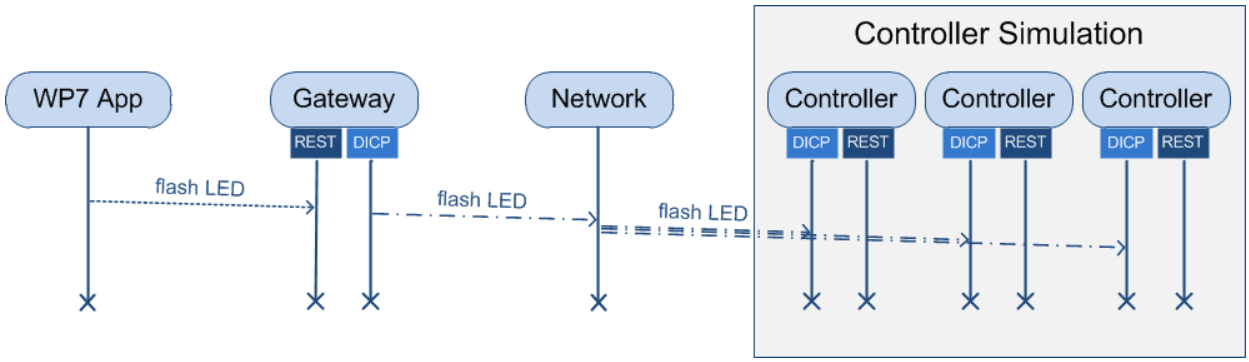


Abbildung 12: Ablaufdiagramm - Flash LED

5.2.2. Webservice-Schnittstelle (REST)

An eine Webservice Architektur für mobile Geräte stellen sich folgende Anforderungen:

- Plattformunabhängig: Applikationen für weitere mobile Plattformen wie Android oder iPhone sollen entwickelt werden können.
- Schlank und einfach: Auf mobilen Geräten ist es wichtig die eingeschränkten Ressourcen nicht zu verschwenden.
- Keine Bibliotheken notwendig: Sobald Bibliotheken benötigt werden, müssten diese für jede mobile Plattform zur Verfügung stehen.

REST erfüllt die erwähnten Anforderungen. REST ist nicht direkt eine Softwarearchitektur, es ist mehr ein Architekturstil. Wenn REST im Kontext von HTTP verwendet wird, beschreibt es einige Basisprinzipien wie Webservices aussehen sollen und wie Client und Server kommunizieren, basierend auf den Konzepten des „World Wide Web“ (WWW). Heute wird REST von vielen grossen Webapplikationen wie z.B. Facebook, Flickr oder Amazon für ihre Webservice-Schnittstelle nach aussen benutzt.

Der Unterschied zwischen REST und anderen Webservice Architekturen wie z.B. SOAP ist, dass REST mit dem bestehenden Vokabular und Konzepten des „World Wide Web“, wie z.B. URIs, Fehler-Codes (404, 500, ...), Methoden (GET, POST, PUT, ...) arbeitet, während in SOAP für jede Applikation eigene Konzepte (Methoden) definiert werden. Für die Abfrage einer Benutzerliste würde zum Beispiel in SOAP eine neue Methode

‚*getAllUsers()*' definiert werden, während in REST für dieselbe Aufgabe das existierende HTTP URI Konzept benutzen wird ‚*GET Users/All*'.

Damit ein Client fähig ist REST-Webservice zu benutzen, muss er nur in der Lage sein über HTTP zu kommunizieren. Weil eine HTTP-Kommunikation auf allen Mobile- und Desktop-Plattformen möglich ist, wird mit REST auf einfache Weise ein plattformunabhängiger Webservice bereitgestellt.

Die Entwicklung eines REST-Webservice auf Server Seite gestaltet sich z.B. mit .NET sehr einfach. Die WCF-Bibliothek in .NET bietet die Möglichkeit REST-Webservices bereitzustellen.

Auf Client Seite ist keine Bibliothek vorhanden, um REST-Webservice zu konsumieren. Dass ist aber nicht weiter ein Problem, weil gar keine Bibliothek benötigt wird. Die Basisfunktionen für eine HTTP-Kommunikation reichen völlig aus und existieren auf jeder Plattform.

5.2.3. Benachrichtigung mit REST - Long Polling

In einer reinen Client-Server-Architektur ist es nicht möglich als Client eine Benachrichtigung vom Server zu empfangen. Anfragen können immer nur vom Client her kommen und nicht umgekehrt, denn ein normaler Client wäre gar nicht in der Lage eine Anfrage eines Servers zu bearbeiten. Um trotzdem eine Benachrichtigung auf dem Client zu bekommen, wenn z.B. die Service Pin Taste am Kontroller gedrückt wird, gibt es folgende Möglichkeiten:

- *Polling*
  Mit *Polling* ist gemeint, dass der Client in definierten Zeitabständen (z.B. 2 Sekunden) eine Anfrage an den Server sendet. Wenn keine Benachrichtigung vorhanden ist, gibt der Server einfach eine leere Antwort zurück. Ansonsten enthält die Antwort den angefragten Inhalt. Ein Nachteil ist, dass damit viel unnötiger Netzwerkverkehr generiert wird.
- Long *Polling*
  Mit Long *Polling* ist eine spezielle Art des *Polling* gemeint. Der Unterschied zum *Polling* besteht darin, dass beim Long *Polling* die Verbindung serverseitig viel länger offen gelassen wird. Der Client sendet eine Anfrage an den Server, der Server gibt jedoch nicht direkt eine leere Antwort zurück, sondern lässt die Anfrage offen bis neue Informationen verfügbar sind oder ein Timeout (z.B. 30 Sekunden) abgelaufen ist. Wenn das Timeout abgelaufen ist, wird trotzdem eine leere Antwort an den Client gesendet. Der Client hingegen wartet nie und sendet, wenn er eine Antwort erhält, immer gleich wieder eine neue Anfrage. Ein Nachteil dieser Strategie ist, dass sie sehr schlecht skaliert. Auf einem viel benutzten Server wären dadurch stets sehr viele Verbindungen offen und dies stellt schnell ein grosses Problem dar.

Unser Entscheid fiel auf das *Long Polling*, weil die schlechte Skalierung in unserem Fall keine Rolle spielt. Es werden nie mehr als zwei, drei mobile Geräte gleichzeitig mit dem Gateway kommunizieren.

5.2.4. Datenübertragungsformat (JSON)

Als Datenübertragungsformat für die REST-Webservices wird JSON verwendet. JSON steht für JavaScript Object Notation und ist ein kompaktes, textbasiertes Dateiformat, welches einfach und schnell lesbar und interpretierbar ist. Es wird häufig für den Datenaustauch zwischen Applikation verwendet.

Eine Alternative wäre XML, doch XML hat strukturbedingt einen viel grösseren Overhead als JSON und das Interpretieren von XML ist um einiges langsamer.

5.2.5. Discovery-Protokoll (DICP)

Im Rahmen des neuen Raumautomationssystems System ONE hat die Siemens BT ein eigenes Discovery-Protokoll entwickelt. Das Protokoll heisst *„Device Identification and Configuration Protocol"* (DICP) und basiert auf selbst definierten UDP-Multicast Anfragen und Antworten.

Die folgenden Gegebenheiten zeigen die Notwendigkeit Raumkontroller suchen, identifizieren und konfigurieren zu können:

- Raumkontroller, welche frisch aus der Fabrik kommen, haben keine definierte IP-Adresse. Die IP-Adressen werden von den unkonfigurierten Raumkontrollern selbst im IP-Netz dynamisch ausgehandelt.
- Die IP-Adressen der unkonfigurierten Raumkontroller sind beim Engineering noch nicht bekannt.
- Jeder Raumkontroller hat eine eindeutige Seriennummer. Die Seriennummern der benutzten Kontroller sind jedoch beim Engineering noch nicht zwingend bekannt.
- Wo sich welcher Raumkontroller befindet ist beim Engineering nicht zwingend bekannt.

Das DICP-Protokoll bietet Abhilfe dafür und unterstützt folgende Funktionen:

- Suchen nach konfigurierten und unkonfigurierten Raumkontrollern.
- Initiale netzwerkseitige Konfiguration eines Raumkontrollers.
- Zurücksetzen eines Raumkontrollers (Konfiguration löschen).
- Senden einer *flash LED* Nachricht an den Kontroller (der Kontroller lässt beim Empfang einer solchen Nachricht eine LED leuchten).
- Empfangen von *Service Pin* Nachrichten (wird vom Kontroller beim Betätigen der *Service Pin* Taste gesendet).

Warum die Siemens ein eigenes Discovery-Protokoll entwickelt hat, ist nicht Teil dieser Arbeit und wird nicht weiter erläutert.

5.3. Sicherheit

5.3.1. Basis-Authentifizierung für REST-Webservices

Die Authentifikation am Gateway sowie am Kontrollersimulator basiert auf dem „*token based authentication*"-Prinzip. Das bedeutet, es ist zu Beginn immer eine Basis-Authentifikation mit Benutzername und Passwort nötig. Bei erfolgreicher Authentifizierung generiert der Server ein Sicherheits-Token und gibt es dem Client zurück. Für jeden weiteren Funktionsaufruf vom Client an den Server, muss dieses Sicherheits-Token übergeben werden. Der Server kann damit überprüfen, ob er dieses Sicherheits-Token kennt und wenn ja den Zugriff zulassen.

5.3.2. Verschlüsselte Verbindung durch WLAN Sicherheitsprotokolle

Das „*token based authentication*"-Prinzip ist stark anfällig auf *„man in the middle"*-Attacken. Ein MITM-Angriff, ist ein Angriff in Rechnernetzen, bei dem der Angreifer physikalisch oder logisch zwischen den beiden Kommunikationspartnern steht. Der Angreifer hat dadurch vollständige Kontrolle über den Datenverkehr und kann Informationen einsehen und sogar manipulieren. Bei „*token based authentication*" muss ein Angreifer nur eine Nachricht abfangen können und er hat das Sicherheits-Token und kann unter diesem Benutzer ab diesem Zeitpunkt selbst Anfragen starten. In unserem Fall wird dieses Problem stark relativiert, weil die Kommunikation immer nur über ein drahtloses Netz geht. Für drahtlose Netze existieren sehr gute Sicherheitsstandards wie z.B. WPA2, welche heute als sicher gelten.

5.3.3. Kontroller sind ungeschützt im drahtgebundenen IP-Netz

Ein ungelöstes Sicherheitsproblem stellt die Tatsache dar, dass die Kommunikation im drahtgebundenen IP-Netz nicht verschlüsselt ist. Das in der Gebäudetechnik benutzte BACnet-Protokoll implementiert keine starken Sicherheitsmechanismen und erlaubt unautorisierten Zugriff auf Kontroller.

Die Sicherheit von BACnet ist ein ungelöstes Problem in der Gebäudeautomation und wird in dieser Arbeit nicht weiter behandelt.

5.4. Fehlerbehandlung und Fehlersuche

Folgende Grundgedanken in der Fehlerbehandlung gelten für den Gateway, den Kontrollersimulator und die SCT-App:

- Der Benutzer soll möglichst wenig durch Fehlermeldungen behindert werden.
- Nur für den Benutzer relevante Fehler werden als Fehlermeldung angezeigt.
- Fehler in der Hintergrundverarbeitung, welche keinen Einfluss auf die Arbeit des Benutzers haben, werden nur geloggt.

5.4.1. Fehlerbehandlung auf der SCT-App

Die SCT-App ist nach einer Serviceorientierten Architektur (SOA) aufgebaut. Jeder Servicefunktion muss eine Callback-Methode übergeben werden. Nach dem asynchronen Abarbeiten der Aufgabe durch den Service-Layer wird die Callback-Methode aufgerufen und damit das Resultat zurückgegeben. Mehr hierzu im Kapitel 9.2.3.

Um auf der SCT-App eine einfache Fehlerbehandlung zu ermöglichen, werfen die Servicefunktionen bewusst keine Ausnahmen (*Exceptions*). Auftretende Ausnahmen in der Kommunikation oder in der Verarbeitung werden in einem Resultat-Objekt gekapselt und über die Callback-Methode zurückgegeben. Dadurch erhält der Aufrufer der Servicefunktion im Erfolgs- wie im Fehlerfall ein Resultat zurück. In der Callback-Methode ist der Kontext bekannt und somit eine spezifische Fehlerbehandlung möglich. Fehler bei denen eine manuelle Operation des Benutzers

fehlgeschlagen ist, werden als Fehlermeldung in der SCT-App angezeigt. Fehler von Hintergrundprozessen wie z.B. dem Nachfragen nach Wertänderungen im Punkttest werden nur geloggt und vorerst nicht angezeigt. Die Wertänderungen werden einmal pro Sekunde angefragt und es wäre störend, wenn bei jedem Fehlschlagen eine Fehlermeldung erscheint. Erst wenn die Kommunikation über mehrere Sekunden hinweg gestört ist, wird dies dem Benutzer mitgeteilt.

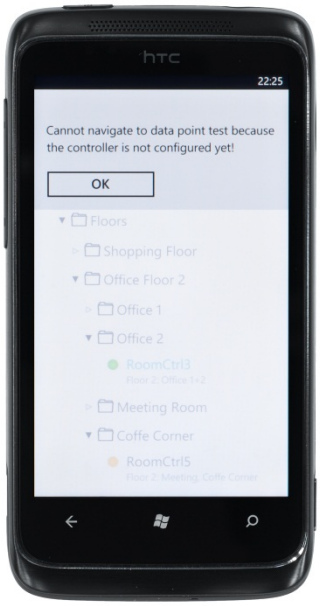


Abbildung 13: SCT-App - Fehlermeldung

5.4.2. Logging

Die *Smart Commissioning* Softwarekomponenten sind grundsätzlich asynchron aufgebaut und das ganze System basiert stark auf Remoteanfragen an den Gateway oder den Kontrollersimulator. Es ist auch ein gleichzeitiger Zugriff von mehreren Clients auf den Gateway oder den Simulator möglich und dadurch entstehen Nebenläufigkeiten in der Verarbeitung der Anfragen. Gerade bei verteilten Systemen ist ein gutes *Logging*-Konzept wichtig, um in einem Fehlerfall Hinweise auf die Ursache zu bekommen. Zusätzlich ist es nicht bei jedem Fehler möglich mit dem Debugger zu arbeiten, weil das Laufzeitverhalten dadurch zu stark verändert wird und Fehler unter Umständen nicht mehr auftreten. Die Log-Daten können entscheidend sein, um das Problem zu lokalisieren.

Der Gateway, der Simulator sowie die SCT-App benutzen dieselbe erarbeitete *Logging*-Infrastruktur, welche folgende Funktionen bietet:

- Hinzufügen und Entfernen von Log-Zielen (Datei-Logger, *View*-Logger, Service-Logger).
- Absetzen von Log-Meldungen basierend auf einem Beschreibungstext oder einer Ausnahme (*Exception*).
- Über *Reflection* werden den Log-Meldungen noch weitere Informationen wie z.B. Namespace, Klassenname und Funktionsname hinzugefügt.

Gateway und Simulator
Der Gateway sowie der Simulator haben als Log-Ziel jeweils einen Datei-Logger, welcher die Log-Meldungen in separate Log-Dateien schreibt, sowie einen *View*-Logger, welcher die Log-Meldungen in einer Log-*View* übersichtlich darstellt. Die Log-*View*s können ein- und ausgeblendet werden und bieten für jede Spalten eine Sortierfunktion.

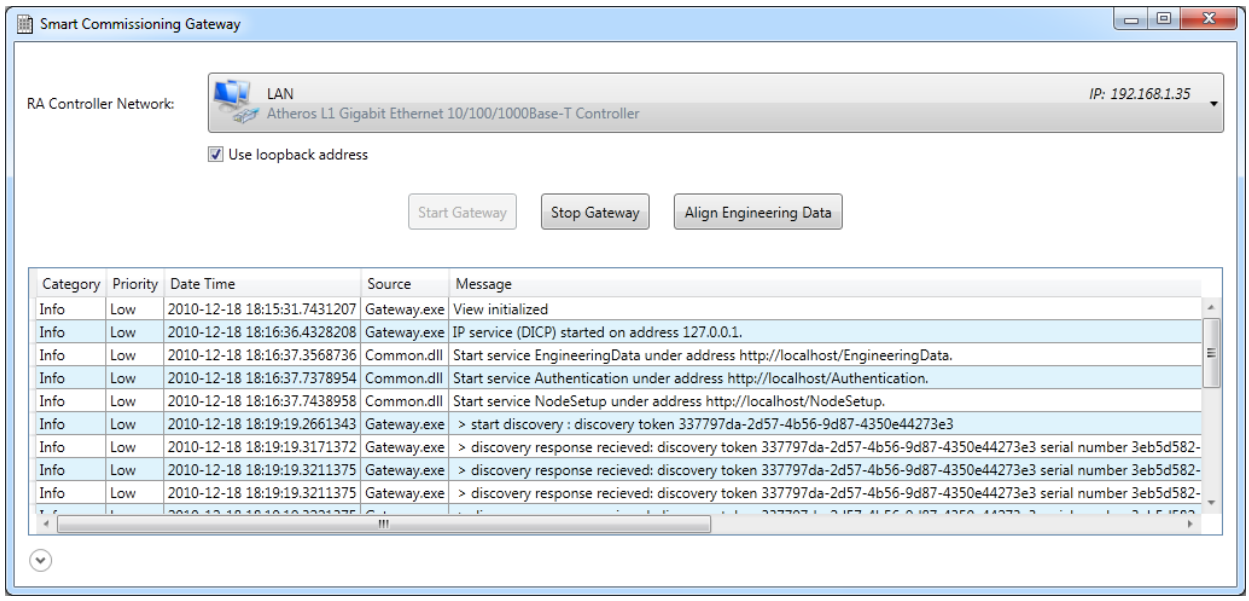


Abbildung 14: Gateway - Log-View

SCT-App

Eine Log-*View* auf der SCT-App ist nicht sinnvoll, denn auf dem kleinen Display wäre die Analyse der Log-Daten ziemlich schwierig. Das Schreiben in eine Log-Datei ist auch keine Option, weil es auf dem WP7 noch keine akzeptable Möglichkeit gibt, Dateien auf dem PC zu transferieren. In Anbetracht der erwähnten Einschränkungen werden die Log-Daten von der SCT-App über den Service-Logger an den Gateway gesendet. Alle Log-Meldungen des Gateways und der SCT-App können somit zusammen bequem auf dem Laptop analysiert werden.

6. Gateway (SCG)

6.1. Einleitung

6.1.1. Notwendigkeit

Während der Analyse-Phase hat sich herausgestellt, dass ein Gateway zum jetzigen Zeitpunkt aus folgenden Gründen zwingend notwendig ist:

- Das WP7 erlaubt keine Socket-Programmierung, dass bedeutet auch keine UDP-Multicast Anfragen. Somit ist es nicht möglich nach dem DICP-Protokoll nach Kontrollern zu suchen.
- Es gibt keinen einfachen Weg die Engineering-Daten auf das WP7 zu kopieren. Die einzige Möglichkeit besteht über einen lokalen Webserver, das Internet oder per Email. Über das Internet oder per Email stellen keine akzeptablen Lösungen dar und ein lokaler Webserver wäre bereits eine vereinfachte Art des Gateways.

Für die Zukunft wäre eine Lösung denkbar, in der gewisse Funktionen des Gateways durch einen Kontroller bereitgestellt werden.

6.1.2. Funktionsumfang

Der Gateway bietet gegen aussen folgende Funktionen:

- Basis-Authentifizierung
  Der Gateway stellt Funktionen über einen REST-Webservice zur Verfügung, um sich als Benutzer mit Benutzernamen und Password anzumelden. Bei der Authentifizierung wird ein Sicherheits-Token generiert, welches bei jedem weiteren Funktionsaufruf übergeben werden muss.
- Bereitstellen der Engineering-Daten
  Der Gateway stellt über einen REST-Webservice Funktionen zur Verfügung, um vom WP7 aus Informationen aus den Engineering-Daten zu bekommen. Zu diesen Informationen gehört die Gebäudehierarchie, eine Liste von logischen Kontrollern und welcher logische Kontroller für welche Elemente der Gebäudehierarchie zuständig ist.
- *Node-Setup* Funktionen (Funktionen des DICP-Protokolls über REST)
  Zusätzlich stellt der Gateway über einen REST-Webservice Funktionen zur Verfügung, um vom WP7 aus Kontroller zu suchen, zu identifizieren und zu konfigurieren. Einfach gesagt werden durch den Gateway die Funktionen des DICP-Protokolls als REST-Webservice allen nicht UDP-Multicast fähigen Geräten zur Verfügung gestellt.

6.1.3. Technologie

Der Gateway ist eine in C# geschriebene Applikation und basiert auf dem .Net Framework 4.0. Für die Benutzerschnittstelle wird WPF (*Windows Presentation Foundation*) verwendet und für die Kommunikation WCF (*Windows Communication Foundation*).

6.2. Architekturübersicht

Intern ist der Gateway nach einer Layer-Architektur aufgebaut mit *Presentation*-, *Domain*- und *Communication*-Layer. Zusätzlich sind gewisse Basisfunktionalitäten für den Gateway, den Simulator und die SCT-App in einer gemeinsamen Bibliothek implementiert.

Presentation-Layer
Die Entkopplung von Darstellung und Funktionalität wird mit dem MVVM-Pattern und *Databinding* erreicht. MVVM steht für *Model-View-ViewModel* und wird in Kapitel 9.2.2 ausführlicher behandelt. Um Operationen auszuführen, werden UI-Elemente wie z.B. Buttons an *Commands* gebunden. Ob ein Button aktiviert ist, wird über die *CanExecute*-Methode des *Commands* ermittelt und beim Klicken wird die Execute-Methode ausgeführt. Durch das MVVM-Pattern und die *Commands* wird eine strikte *„separation of concerns“* erreicht:

- *View*: Ist nur für die Darstellung zuständig und enthält keine Logik.
- *Model*: Ist nur für die Daten zuständig und weiss nichts über die Darstellung.
- *ViewModel*: Weiss wie die Daten dargestellt werden müssen und stellt die Verbindung zwischen den Daten und der Darstellung dar. Enthält jedoch keine Logik zu den Operationen.
- *Command*: Ist nur für eine Operation zuständig und enthält die Logik wann eine Operation ausgeführt werden darf und wie sie ausgeführt wird.

Domain-Layer
Der *Domain*-Layer ist zuständig für die Datenklassen (*Model*), die Businesslogik (*Logic*) und die Webservice-Implementation (*Services*). Die Webservice-Klassen beinhalten nicht viel Logik, sondern rufen nur Funktionen der Businesslogik auf.

Communication-Layer
Der *Communication*-Layer ist zuständig für die Webservice-Schnittstellen (*Interface Contracts*) gegen aussen.

Common (Bibliothek)
In der Common-Bibliothek befinden sich Infrastrukturklassen, Metadaten, Funktionalität für das *Logging* und die Fehlerbehandlung, sowie alle „*Data Contract*"-Klassen (Klassen für den Datenaustausch über REST).

Die folgende Grafik zeigt eine stark vereinfachte Darstellung der Klassen und Namespaces. Die Klassen *View, ViewModel* und *Command* stehen nur als Platzhalter für die spezifischen *View*- und *ViewModel*-Klassen.

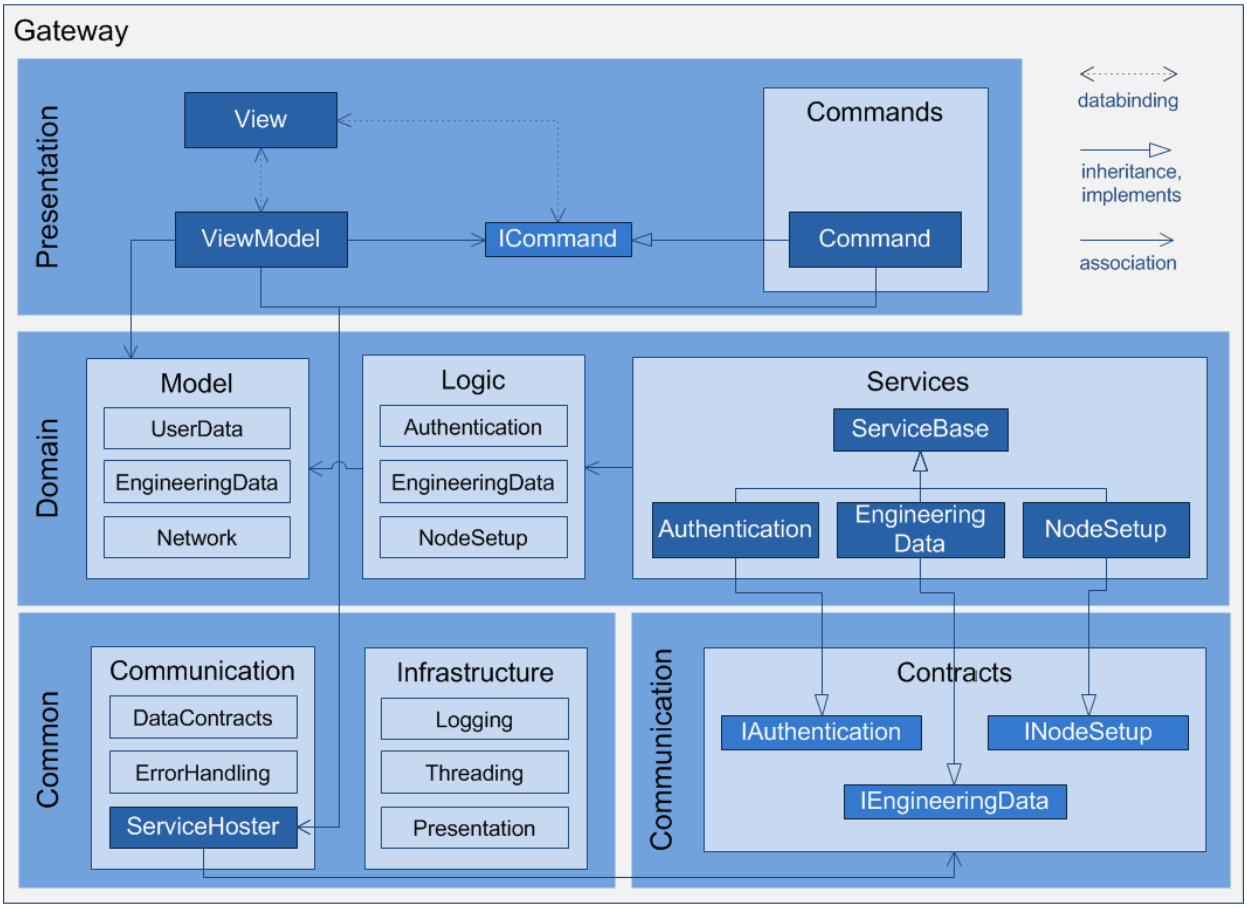


Abbildung 15: Gateway - Architekturübersicht

6.3. Serviceschnittstelle (REST-Webservice)

6.3.1. Interfaces

Im folgenden Abschnitt werden die Interfaces gezeigt, welche die Webservice-Schnittstelle definieren.

Basis-Authentifizierung
Beispiel: http://192.168.1.1/Authentication/Authenticate?user=Reto&pw=phone

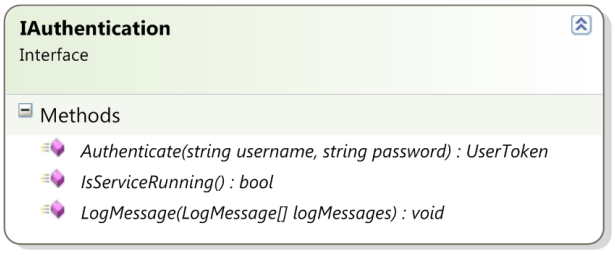


Abbildung 16: Gateway Schnittstelle - Authentifikation

Bereitstellen der Engineering-Daten
Beispiel: http://192.168.1.1/EngineeringData/BuildingHierarchy?auth=c65b2a15-5b1b-4d7b-a487-81a9f0168047

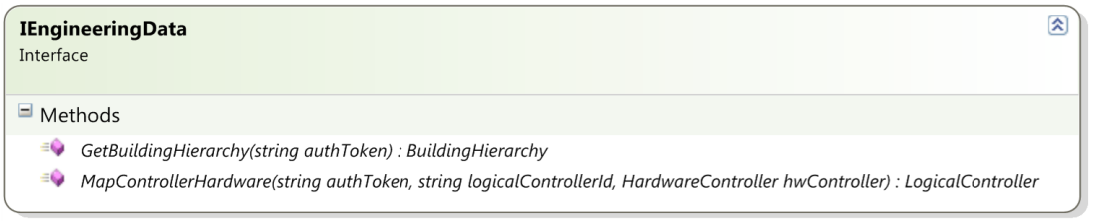


Abbildung 17: Gateway Schnittstelle - Engineering-Daten

Node-Setup Funktionen (Funktionen des DICP-Protokolls)
Beispiel: http://192.168.1.1/NodeSetup/StartDiscovery?auth=c65b2a15-5b1b-4d7b-a487-81a9f0168047&model=PXC3.E72&uconf=False&timeout=1000

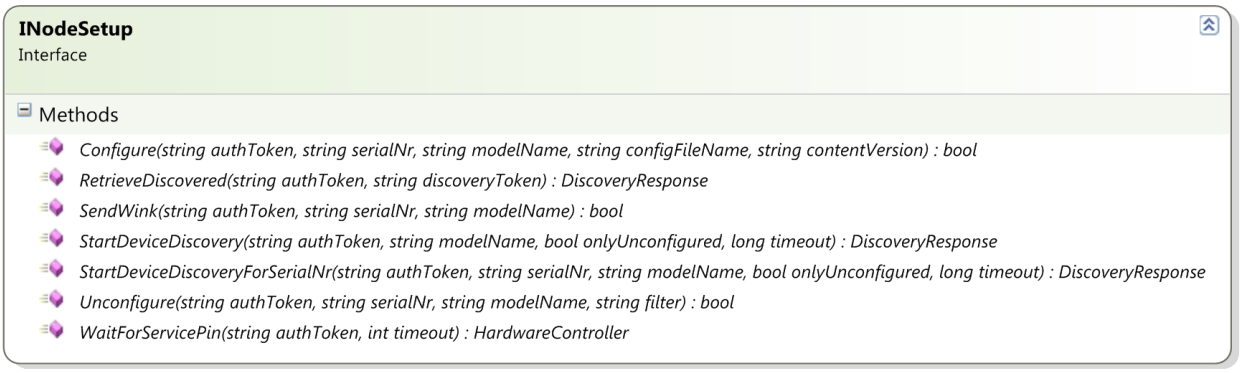


Abbildung 18: Gateway Schnittstelle - Node-Setup

6.3.2. Implementation mit WCF

Das Bereitstellen eines REST-Webservices mit WCF ist sehr einfach. Es wird nur ein Interface mit Attributen und eine Klasse welche das Interface implementiert benötigt.

Webservice-Interface mit Attributen
Das Interface selbst wird mit dem Attribut *ServiceContract* annotiert.

```
[ServiceContract]
public interface IAuthentication
```

Und die Interface-Operationen mit den Attributen *OperationContract* und *WebGet*.

```
[OperationContract]
[WebGet(ResponseFormat = WebMessageFormat.Json,
UriTemplate = "Authenticate?user={username}&pw={password}")]
UserToken Authenticate(string username, string password);
```

Über das Attribut *WebGet* kann mit dem *ResponseFormat* das Datenübertragungsformat (XML oder JSON) gewählt werden. Das Attribut *UriTemplate* definiert die URI und die Verknüpfung zwischen den Attributen in der URI und den Parametern der Operation.

Webservice-Klasse
Das Verhalten der Webservice-Klassen kann über das Attribut *ServiceBehavior* angegeben werden.

```
[ServiceBehavior(InstanceContextMode = InstanceContextMode.Single,
ConcurrencyMode = ConcurrencyMode.Multiple)]
public sealed class Authentication : IAuthentication
```

Der *InstanceContextMode* definiert, ob für jeden Aufruf eine neue Instanz kreiert werden soll oder ob jeder Aufruf an dieselbe Instanz gehen soll. Über den *ConcurrencyMode* wird definiert, ob Nebenläufigkeit erlaubt wird oder ob immer nur ein Aufrufer in einer Funktion sein darf. Diese Einstellungen haben einen grossen Einfluss auf die Geschwindigkeit des Webservices. Das Verbieten von Nebenläufigkeit/Parallelität kann den Webservice stark verlangsamen. Auf der anderen Seite vereinfacht es aber die Implementation der Service-Funktionen, weil nicht auf die Synchronisation geachtet werden muss. Es wird durch WCF bereits beim Funktionseintritt synchronisiert. Bei jedem Webservice im Gateway sowie im Simulator wird aus Performancegründen Nebenläufigkeit erlaubt. Dies bedeutet, dass auch alle Business-Klassen threadsicher sein müssen. Es wurde z.B. für alle gefundenen Kontroller bei der Kontroller Suche ein threadsicherer Zwischenspeicher (Concurrent-Cache) implementiert. Damit

der Zwischenspeicher nicht irgendwann überfüllt ist, müssen alte Einträge gelegentlich entfernt werden. Dafür wurde ein Hintergrundprozess implementiert, welcher in zeitlichen Abständen nach alten Einträgen sucht und diese entfernt.

Bereitstellen (Hosten) durch WCF
Folgende zwei Zeilen reichen aus, um den eigenen Rest-Webservice durch WCF bereitzustellen:

```
WebServiceHost serviceHost = new WebServiceHost(typeof(Authentication), serviceUri);
serviceHost.Open();
```

Die Webservice-Klassen werden in jedem Fall durch WCF instanziert und es gibt keine Möglichkeit über den Konstruktor etwas zu übergeben. Um trotzdem von den Webservice-Klassen auf die Business-Logik zugreifen zu können, ist ein Singleton nötig. Eine Klasse *ServiceContext* existiert als Singleton und bietet eine Instanz von jeder Businesslogik-Klasse als Property. Ein Singleton erhöht die Kopplung, da von überall auf das Singleton zugegriffen werden kann. Zusätzlich erschweren sie ein sauberes Unittesting, da ein Singleton nicht durch ein Mock-Objekt ersetzt werden kann. In diesem Fall ist das nicht weiter tragisch, denn die Webservice-Klassen enthalten keine Logik, sondern rufen nur Funktionen der Businesslogik auf.

6.3.3. Caching

Die *WebClient* Klasse auf dem WP7, um REST-Webservices zu konsumieren hat ein sehr starkes Caching (Zwischenspeichern) implementiert. Wenn die URI identisch ist, wird nicht einmal eine Anfrage an den Server gesendet, sondern direkt das alte Resultat aus dem Cache zurückgegeben. Um Wertänderungen von einem Datenpunkt auf der SCT-App darzustellen, wird in einem definierten Zeitabstand (z.B. 1 Sekunde) nach dem neuen Wert gefragt. Die URI dieser Abfragen ist natürlich immer dieselbe. Durch das Caching des WP7 wird somit gar keine neue Anfrage an den Kontroller gesendet, sondern immer der erste Wert zurückgegeben. In der SCT-App kann dieses Verhalten nicht beeinflusst werden, jedoch auf Serverseite (Gateway, Kontrollersimulator). Auf Serverseite kann im WCF über den HTTP-Header das Caching mit folgender Anweisung ausgeschaltet werden:

```
WebOperationContext.Current.OutgoingResponse.Headers.Add("Cache-Control", "no-cache");
WebOperationContext.Current.OutgoingResponse.Headers.Add("Pragma", "no-cache");
```

6.3.4. Fehlerbehandlung / Fehlerrückgabe

Bei REST-Webservices wird als Fehlerrückgabe das bestehende Konzept von HTTP mit den Fehlercodes (*Error-Code*) benutzt. Es gelten auch dieselben Fehlercodes wie z.B. *„not found“* (404) oder *„internal server error“* (500). Ein interner Serverfehler wird mit folgender Anweisung gesetzt:

```
WebOperationContext.Current.OutgoingResponse.StatusCode =
HttpStatusCode.InternalServerError;
```

Zusätzlich zum Fehlercode kann ein Beschreibungstext mit folgender Anweisung gesetzt werden:

```
WebOperationContext.Current.OutgoingResponse.StatusDescription = "Wrong authentication
token.";
```

Diese Fehlerbeschreibung ist natürlich sprachabhängig und kann von der SCT-App programmatisch nicht ausgewertet werden. Es wird ein eigener Fehlercode benötigt, welcher eine genauere Fehlerrückgabe ermöglicht, als die HTTP-Fehlercodes. Eine einfache Möglichkeit ist, in das Feld für die Fehlerbeschreibung nicht einfach den Fehlertext, sondern strukturierte Informationen mit Fehlercode, Text und Daten zu speichern. Als Datenaustauschformat bei den Webservices wird JSON verwendet. Damit nicht verschiedene Datenumwandler (*Serializer*) benutzt werden müssen, wird auch hier JSON verwendet. Das heisst wenn ein Fehler auftritt, ist in der Fehlerbeschreibung ein komplettes Fehlerobjekt in JSON gespeichert. Folgendes Code-Beispiel zeigt das Aufsetzen eines Fehlerobjekts, wenn z.B. die angefragten Daten nicht existieren.

```
Error error = new Error(ErrorCode.IdNotFound, ex.Message);
WebOperationContext.Current.OutgoingResponse.StatusDescription =
JsonSerializationHelper<Error>.SerializeJson(error);
```

Die SCT-App kann bequem den JSON-Text wieder in ein Fehlerobjekt umwandeln und den Fehler auswerten.

```
Error error =
JsonSerializationHelper<Error>.DeserializeJson(webResponse.StatusDescription);
if (error.ErrorCode == ErrorCode.IdNotFound)...
```

6.5. Benutzerschnittstelle (UI)

Folgende Abbildungen zeigen die Benutzerschnittstelle des Gateways. Sie ist sehr einfach gehalten, bietet aber alle benötigten Einstellungen und Funktionen.

Abbildung 19: Gateway UI

Der Gateway muss wissen in welchem Netz das DICP-Protokoll für die Kontrollersuche laufen soll. Damit dafür keine IP-Adresse eingegeben werden muss, kann das Netz mit den Kontrollern komfortabel in einer Liste ausgewählt werden.

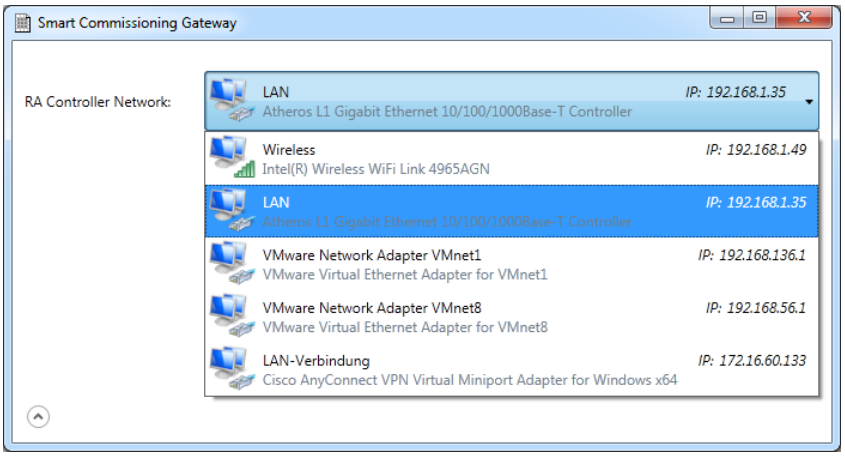


Abbildung 20: Gateway UI - Kontrollernetzwerk-Auswahl

Da der Gateway mit seinem Webservice mehr ein Dienst ist als eine Applikation, wird er beim Schliessen des Fensters nur minimiert und nicht beendet. Solange der Gateway läuft erscheint in der Windows-Taskbar ein Icon. Über dieses Icon ist es möglich das Fenster wieder zu öffnen oder den Gateway ganz zu beenden.

Abbildung 21: Gateway UI - Kontextmenü in der Windows-Taskbar

7. Kontrollersimulator (CST)

7.1. Einleitung

7.1.1. Notwendigkeit

Die Simulation der Kontroller ist aus folgenden Gründen notwendig:

- Die Kontroller-Hardware wurde erst vor kurzem für die Produktion freigegeben und ist nur sehr erschwert verfügbar.
- Die Firmware auf den Kontrollern befindet sich zurzeit noch in Entwicklung.
- Um Abhängigkeiten zu einem sich in Entwicklung befindlichen System zu vermeiden.
- Die Kontroller bieten bis anhin nur eine Konfigurationswebseite und keinen REST-Webservice.

7.1.2. Funktionsumfang

Wie bereits mehrfach erwähnt wird das Suchen nach Kontrollern mit dem DICP-Protokoll durchgeführt. Jeder Kontroller implementiert den DICP-Server und jeder Client wie z.B. der Gateway den DICP-Client.

Der Simulator bietet gegen aussen folgende Funktionen:

- DICP Server pro Kontroller (Kontroller Suchen)
  Der Simulator stellt einen DICP-Server für jeden simulierten Kontroller zur Verfügung. Jeder Kontroller gibt somit eine Multicast-Antwort auf Suchanfragen, wird selektiert wenn ein *flash LED* kommt und kann einen *Service Pin* absenden.
- REST Server pro Kontroller (Datenpunkttest)
  Der Simulator stellt für jeden simulierten Kontroller einen REST-Webservice für den Datenpunkttest zur Verfügung.
- Benutzerschnittstelle
  Der Simulator bietet eine Benutzerschnittstelle, um auf einfache Weise Kontroller, Module und Datenpunkte hinzuzufügen, zu bearbeiten oder zu löschen. Zusätzlich kann für jeden Kontroller ein *Service Pin* abgesendet werden.
- Simulation von Signaländerungen
  Der Simulator kann für einen Datenpunkt zufällige Signaländerungen simulieren, um auf der SCT-App die Trend-Funktion zu testen. Die Wertänderung beträgt jeweils zufällig bis zu 10 Prozent vom alten Wert und kann nach oben oder nach unten geschehen.

7.2. Serviceschnittstelle (REST-Webservice)

Da die REST-Webservices von der SCT-App benutzt werden und dort die Leistungsfähigkeit eingeschränkt ist, stellen sie zusätzliche Anforderungen an den Webservice. Der REST-Webservice soll möglichst schlank und einfach sein und die Daten in der Form liefern, wie sie von der SCT-App benötigt werden. Das bedeutet gut aufgearbeitet, damit auf der SCT-App keine grosse Umwandlung mehr notwendig ist.

Im folgenden Abschnitt werden die Interfaces gezeigt, welche die Webservice-Schnittstelle definieren. Detailliertere Informationen sind im Architektur- und Designdokument zu finden.

Basis-Authentifizierung
Beispiel: http://192.168.1.1:11001/Authentication/Authenticate?user=Reto&pw=phone

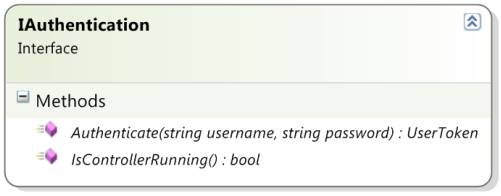


Abbildung 22: Kontrollersimulator Schnittstelle - Authentifikation

Kontroller-Information
Beispiel: http://192.168.1.1:11001/ControllerInfo/Details?auth=6f1fb301-99e3-418b-8cb1-311813191b29

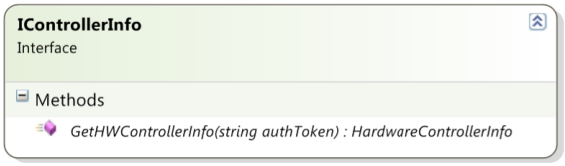


Abbildung 23: Kontrollersimulator Schnittstelle - Kontrollerinformationen

Datenpunkttest
Beispiel: http://192.168.1.1:11001/DataPointTest/TXIO/Modules?auth=6f1fb301-99e3-418b-8cb1-311813191b29

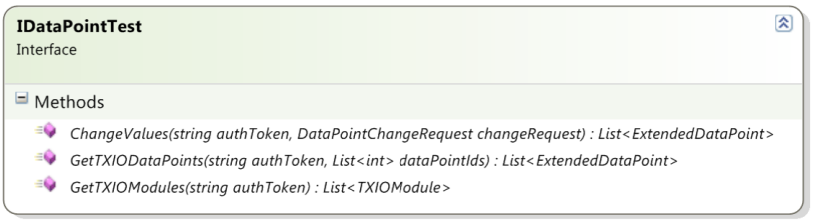


Abbildung 24: Kontrollersimulator Schnittstelle - Datenpunkttest

Um Signalausgänge zu testen, muss es auch möglich sein, von der SCT-App aus den Wert eines Datenpunktes zu ändern. Häufig wird für das Ändern eines Wertes von aussen das Wort „*kommandieren*“ verwendet, weil der berechnete Wert manuell überschrieben wird. Die Funktion *ChangeValues(...)* bietet die Möglichkeit von aussen mehrere beliebige schreibbare Attribute von mehreren Datenpunkten zu verändern. Ermöglicht wird dies dadurch, dass der Funktion eine Liste von *DataPointChangeRequests* übergeben werden muss. Ein *DataPointChangeRequests* steht für die Änderungen an einem Datenpunkt und hat immer eine Datenpunkt-Id sowie eine Liste von Attribut-Ids mit den zugehörigen neuen Werten.

7.3. Benutzerschnittstelle (UI)

Folgende Abbildungen zeigen die Benutzerschnittstelle des Simulators. Es gibt grundsätzlich zwei Sichten: Eine um die Kontroller für das DICP-Protokoll zu bearbeiten und eine um die Module und Datenpunkte pro Kontroller für die REST-Webservices zu bearbeiten. Für das DICP-Protokoll ist es wie beim Gateway notwendig das entsprechende IP-Netz auszuwählen. Dies geschieht auch hier komfortabel über eine Liste.

Kontrollersicht
Für jeden Kontroller können folgende Attribute bearbeitet werden: *Name, Description, Device Instance, Model Name, Serial Number, Version Net Mask, Gateway, If Mode, Location, Status, Address, Port, DHCP, Firmware, Free Memory, Actual Cycle Time, Max Cycle Time* und *Last Download*. Zusätzlich gibt es pro Kontroller einen Button um den *Service Pin* zu senden und Buttons um die Services zu starten und zu stoppen.

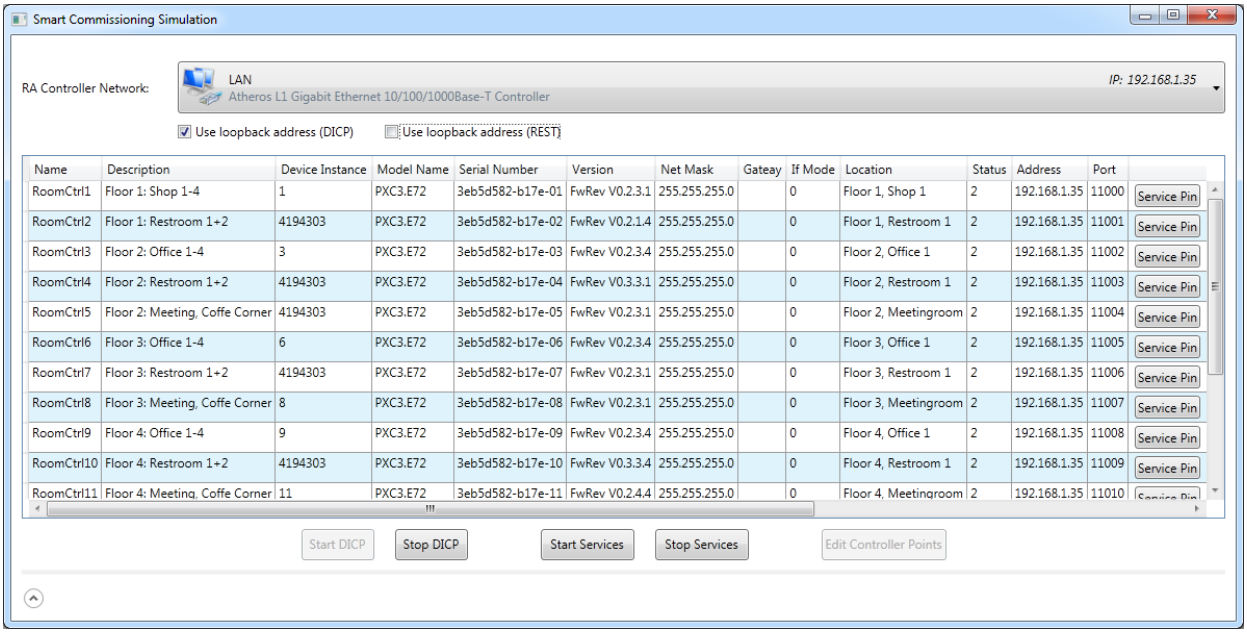
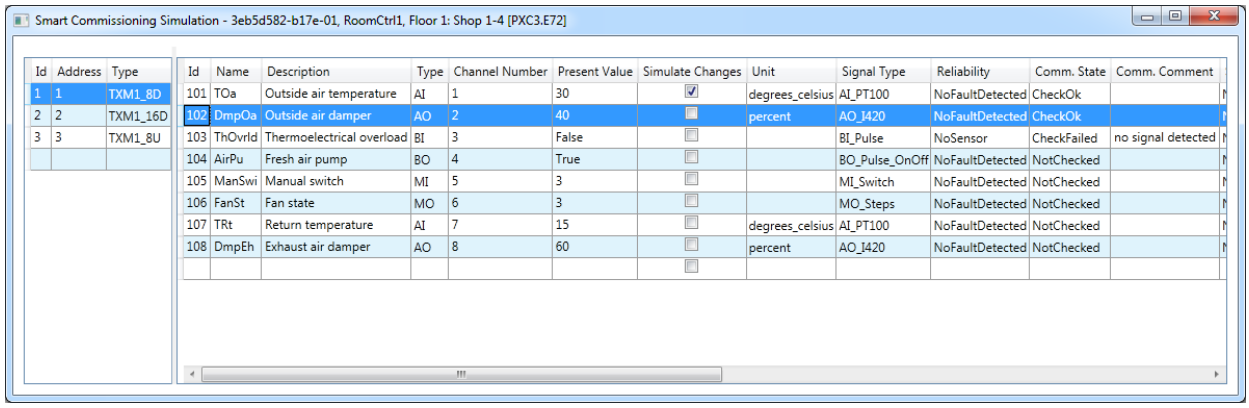


Abbildung 25: Kontrollersimulator - Kontroller bearbeiten

Datenpunktsicht

Jedes Modul hat eine Id, eine Addresse und einen Typ und jeder Datenpunkt hat folgende Attribute: *Id, Name, Description, Type, Channel Number, Present Value, Unit, Signal Type, Reliability, Commissioning State, Commissioning Comment, Status Flag, Out of Service, Event State, Resolution, Correction Offset, Correction Factor, Change of Value, Min, Max, Polarity, Inactive Text, Active Text* und *State Text*. Nicht jedes Attribut gibt es für jeden Datenpunkttyp. Gewisse werden nur für analoge Signale, andere nur für binäre Signale und wieder andere nur für mehrstufige Signale verwendet.



Id	Address	Type
1	1	TXM1_8D
2	2	TXM1_16D
3	3	TXM1_8U

Id	Name	Description	Type	Channel Number	Present Value	Simulate Changes	Unit	Signal Type	Reliability	Comm. State	Comm. Comment
101	TOa	Outside air temperature	AI	1	30	☑	degrees_celsius	AI_PT100	NoFaultDetected	CheckOk	
102	DmpOa	Outside air damper	AO	2	40	☐	percent	AO_I420	NoFaultDetected	CheckOk	
103	ThOvrld	Thermoelectrical overload	BI	3	False	☐		BI_Pulse	NoSensor	CheckFailed	no signal detected
104	AirPu	Fresh air pump	BO	4	True	☐		BO_Pulse_OnOff	NoFaultDetected	NotChecked	
105	ManSwi	Manual switch	MI	5	3	☐		MI_Switch	NoFaultDetected	NotChecked	
106	FanSt	Fan state	MO	6	3	☐		MO_Steps	NoFaultDetected	NotChecked	
107	TRt	Return temperature	AI	7	15	☐	degrees_celsius	AI_PT100	NoFaultDetected	NotChecked	
108	DmpEh	Exhaust air damper	AO	8	60	☐	percent	AO_I420	NoFaultDetected	NotChecked	

Abbildung 26: Kontrollersimulator - Datenpunkte bearbeiten

8. Einführung in die Entwicklung für Windows Phone 7

8.1. Kurze Faktenübersicht

- Entwicklungsumgebung ist Visual Studio 2010
- Entwicklungsframework ist ein Silverlight-Derivat genannt *„Silverlight for Windows Phone“*
- Phone-Emulator ist vorhanden
- Echte Phone-Geräte müssen für die Entwicklung registriert werden
- Probleme bei der Registrierung von Entwickler-Phones bei Studenten-Accounts
- App-Verteilung über Windows Phone Marketplace ist kostenpflichtig
- Keine private oder firmeninterne Verteilung von Apps möglich

8.2. Entwicklungsumgebung

Um mit der Entwicklung für Windows Phone 7 zu starten, genügt bereits die Installation der Windows Phone Developer Tools, welche von Microsoft als kostenloser Download angeboten werden (create.msdn.com). Im Installationspaket befinden sich unter anderem die folgenden Programme und Tools:

- Visual Studio 2010 Express
- Windows Phone Emulator
- XNA Game Studio 4.0
- Microsoft Expression Blend for Windows Phone

Falls bereits eine oder mehrere der enthaltenen Komponenten installiert sind, werden nur die zusätzlich benötigten Programme installiert. Die Windows Phone Developer Tools integrieren sich auch automatisch in eine bereits vorhandene Installation von Visual Studio 2010.

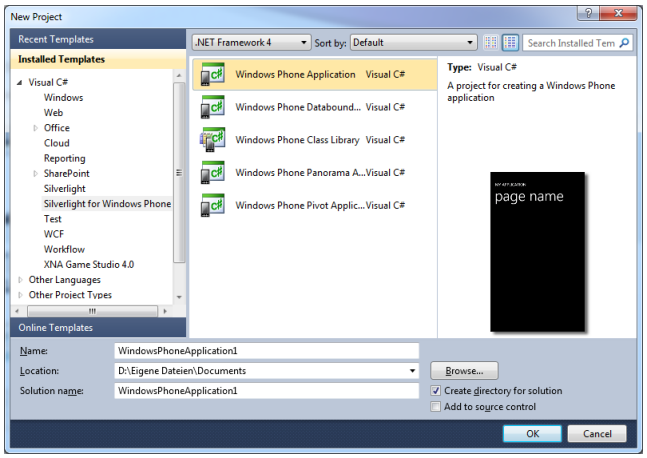


Abbildung 27: Windows Phone Template-Auswahl

Nach der Installation der Entwicklertools stehen die Windows Phone Projektvorlagen in Visual Studio 2010 zur Auswahl (sieh Abbildung links).

Das Framework für die Entwicklung für Windows Phone 7 nennt sich *Silverlight for Windows Phone*. Dieses basiert auf einer abgespeckten Version von Silverlight 3.0 mit spezifischen Erweiterungen für die Windows Phone Hardware. Die Laufzeitumgebung (CLR) ist aber nicht eine Silverlight-Portierung, sondern spezifisch für die Smartphone-Hardware optimiert, da diese ja ohne leistungsfähige PC-Prozessoren und Grafikkarten auskommen muss.

Im Visual Studio gibt es zwei auswählbare Ziele („Targets“) für die Ausführung bzw. das Debuggen von Applikationen. Davon ist eines der Windows Phone Emulator und das andere ein per USB angeschlossenes Windows Phone Gerät. Wenn der Emulator als Ziel ausgewählt ist, wird er automatisch gestartet, sobald man ein Windows Phone Projekt aus dem Visual Studio heraus ausführt (gilt für Release- und Debug-Version).

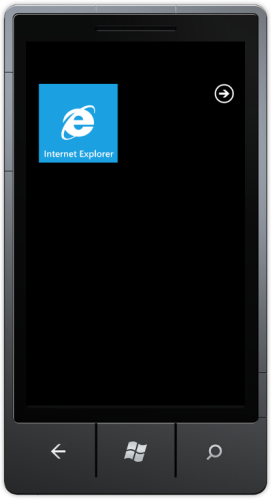


Abb. 28: Windows Phone Emulator

Der Emulator ist Microsoft wirklich sehr gut gelungen. Es ist eigentlich alles so umgesetzt, wie es sich auf einem richtigen Gerät verhält. Die Bedienung der Apps auf dem Emulator erfolgt mit Maus und Tastatur. Der Emulator unterstützt aber auch echtes Multi-Touch, wozu aber natürlich ein entsprechender Touch-Monitor vorhanden sein sollte. Es gibt zwar spezielle Tools, mit denen sogar Multi-Touch mit zwei angeschlossenen PC-Mäusen simuliert werden kann, aber die Möglichkeiten sind sehr beschränkt und auch etwas realitätsfern. Der Emulator erlaubt auch den Zugriff auf das Netzwerk und somit auch auf das Internet. Somit können Client-Server-Applikationen uneingeschränkt getestet werden. Hardwarespezifische Funktionen wie den Beschleunigungssensor oder die Kamera sowie Kommunikationsmöglichkeiten über Telefon, Email und SMS sind dagegen nur eingeschränkt oder gar nicht verfügbar.

Um eine App auf einem echten Smartphone zu testen, braucht es noch einige zusätzliche Schritte. Diese sind zusammen mit den Voraussetzungen im nächsten

Abschnitt beschrieben.

8.3. App auf echtem Gerät testen (App Hub Account)

Um eine App auf einem echten Smartphone zu testen, muss das Phone zuerst als Entwicklergerät bei Microsoft registriert werden. Dies ist u.a. eine Massnahme, um das Abspielen von „Raubkopien“ zu verhindern. Der Registrierungsprozess ist grundsätzlich sehr einfach gehalten. Dazu muss ein kleines Registrierungstool gestartet werden, das bei der Installation der Windows Phone Developer Tools mitinstalliert wurde. In diesem Registrierungstool muss nur das Login für den App Hub Account eingegeben werden. Danach wird das über USB angeschlossene Phone auf den angegebenen Account registriert und für die Entwicklung freigeschaltet.

Abb. 29: Entwickler-Phone Registrierung

App Hub ist Microsofts neue Web-Plattform für die Entwicklung mit Windows Phone 7 und XBOX 360, und ist unter create.msdn.com erreichbar. Um den für die Registrierung des Phones benötigten App Hub Entwickler-Account zu erstellen, ist leider etwas Geduld gefragt. Für diesen Account muss nämlich eine jährliche Gebühr von $99 USD bezahlt werden. Zusätzlich wird von Microsoft eine Identitätsprüfung verlangt, die von der von Microsoft beauftragten Firma GeoTrust durchgeführt wird. Für die Validierung muss normalerweise ein Formular zusammen mit einem Identitätsausweis oder Führerschein an GeoTrust geschickt werden. Der Validierungsprozess kann einige Tage in Anspruch nehmen, währenddessen keine Phones für die Entwicklung freigeschaltet werden können. Der App Hub Account wird erst freigeschaltet, wenn die Identitätsprüfung erfolgreich und die Gebühr von $99 USD bezahlt worden ist.

8.4. App Hub Account für Studenten

Studenten profitieren vom Microsoft DreamSpark Programm (www.dreamspark.com), wodurch keine Kosten für den App Hub Account anfallen. Dafür gibt es bei den Studenten-Accounts ein Problem bei der Registrierung von Entwickler-Phones. Die Ursache liegt darin, dass die Validierung von Studenten-Accounts anders abläuft, als bei den bezahlten Accounts. So wird der Identitätsprüfungsvorgang normalerweise gleich bei Erstellung des Accounts initiiert. Bei Studenten erfolgt dies aber erst mit dem Einsenden der ersten App für den Marketplace. Da bei der Erstellung des Accounts seitens Microsoft aber nicht genügend auf diesen Ablauf hingewiesen wird, verbleibt man in der Annahme, der Account würde automatisch freigeschaltet. Sobald man aber versucht, ein Phone als Entwicklergerät zu registrieren, scheitert dieser Vorgang mit einer kryptischen Fehlermeldung. Erst durch eine Anfrage beim Microsoft Support, konnte der Missstand aufgeklärt werden. Der Microsoft Support kann den Identitätsprüfungsvorgang aber nicht anstossen. Er rät dazu, eine Dummy-App zur Veröffentlichung auf dem Marketplace einzusenden, welche dann automatisch den Identitätsprüfungsvorgang initiiert. Dieses Vorgehen funktioniert tatsächlich. Die Dummy-App wird bei der Validierung auch umgehend zurückgewiesen und somit nicht im Marketplace veröffentlicht.

Dass zuerst eine App eingesendet werden muss, bevor auf einem Phone getestet werden kann, ist als Fehler im Prozess anzusehen. Hier sollte Microsoft unbedingt nachbessern und zumindest die Benutzer besser informieren.

8.5. Verteilung der Apps über Windows Phone Marketplace

Apps können nur auf für die Entwicklung registrierten Geräten direkt aufgespielt werden. Ansonsten wird die Verteilung der Apps ausschliesslich über den Windows Phone Marketplace abgewickelt. Das während dieser Bachelorarbeit entwickelte *Smart Commissioning Tool* wurde nicht im Marketplace veröffentlicht. Die Gründe dafür sind, dass es einerseits nur ein Prototyp ist und andererseits eine firmeninterne Anwendung, welche der Öffentlichkeit nicht zugänglich sein sollte. In diesem Punkt gibt es noch einen gewissen Aufholbedarf seitens Microsoft, da zurzeit keine Möglichkeit besteht, Apps nur firmenintern zu verteilen. Bei der Konkurrenz von Apple wird die firmeninterne Verteilung, als Teil eines grösseren Leistungspakets, gegen eine Gebühr von $299 USD im Jahr angeboten.

Apps, welche für die Veröffentlichung auf dem Marketplace eingesendet werden, durchlaufen einen Validierungs- und Zertifizierungsprozess. Microsoft hat dazu auf dem App Hub eine Anleitung veröffentlicht, welche den Prozess erklärt und alle Anforderungen an die einzusendenden Apps beschreibt. Die Apps werden erst nach erfolgreichem Bestehen aller unterzogenen Prüfungen im Marketplace veröffentlicht.

8.6. Zusätzliche Tools und Libraries

Da Silverlight for Windows Phone auf Silverlight 3.0 basiert, sind viele hilfreiche und dringend benötigte Funktionen, welche in Silverlight 4.0 zur Verfügung stehen, noch nicht vorhanden. Das bedeutet, dass der Einsatz

von zusätzlichen Libraries und Frameworks fast unumgänglich ist. Im Folgenden werden zwei wichtige Zusatz-Libraries kurz vorgestellt, die auch in dieser Bachelorarbeit verwendet wurden.

Silverlight for Windows Phone Toolkit
Das *Silverlight for Windows Phone Toolkit* ist von Microsoft selbst entwickelt worden. Es beinhaltet vor allem zusätzliche Controls sowie Komponenten für eine vereinfachte Handhabung von *Gestures*. Es werden regelmässig aktualisierte Versionen auf silverlight.codeplex.com veröffentlicht.

Prism - Windows Phone 7 Developer Guide
Prism für Windows Phone 7 ist als Teil des *Windows Phone 7 Developer Guide* verfügbar, welcher unter dem *Microsoft patterns & practices* Label veröffentlicht wurde. Prism besteht aus einer Library sowie Beispielprojekten und Hilfestellungen für die Entwicklung unter Windows Phone 7. Der *Windows Phone 7 Developer Guide* kann von wp7guide.codeplex.com heruntergeladen werden.

9. Windows Phone 7 App (SCT)

9.1. Übersicht

Wie bereits in Kapitel 4 erwähnt, ist das *Smart Commissioning Tool* (SCT) eine Windows Phone 7 App und bildet die Schnittstelle zwischen dem Inbetriebnahme-Ingenieur, genannt Installer, und dem Raumautomationssystem. Es ist Teil einer Client-Server-Architektur und kommuniziert ausschliesslich über das HTTP-Protokoll mit RESTful-Webservices, die sich auf dem Gateway sowie auf dem Kontrollersimulator befinden.

In diesem Kapitel werden zuerst die Architektur sowie die wichtigsten Designentscheide erläutert. Weiter werden einige generelle Erkenntnisse zur Softwareentwicklung auf Windows Phone 7 sowie spezifische Beobachtungen und Entscheide, die während der Entwicklung von SCT gemacht wurden, aufgezeigt. Als letzter Teil dieses Kapitels werden die Probleme, Entscheidungen und Lösungen zur Benutzeroberfläche präsentiert.

9.2. Architektur- und Designentscheide

9.2.1. Architekturübersicht

Die SCT-App ist als Layer-Architektur aufgebaut mit *Presentation*-, *Domain*- und *Infrastructure*-Layer. Zusätzlich sind die gemeinsamen Funktionalitäten, die mit Gateway und Simulator geteilt werden, in einer gemeinsamen Bibliothek (*Common*) ausgelagert. Die folgende Grafik zeigt eine sehr vereinfachte Darstellung der Klassen und Namespaces.

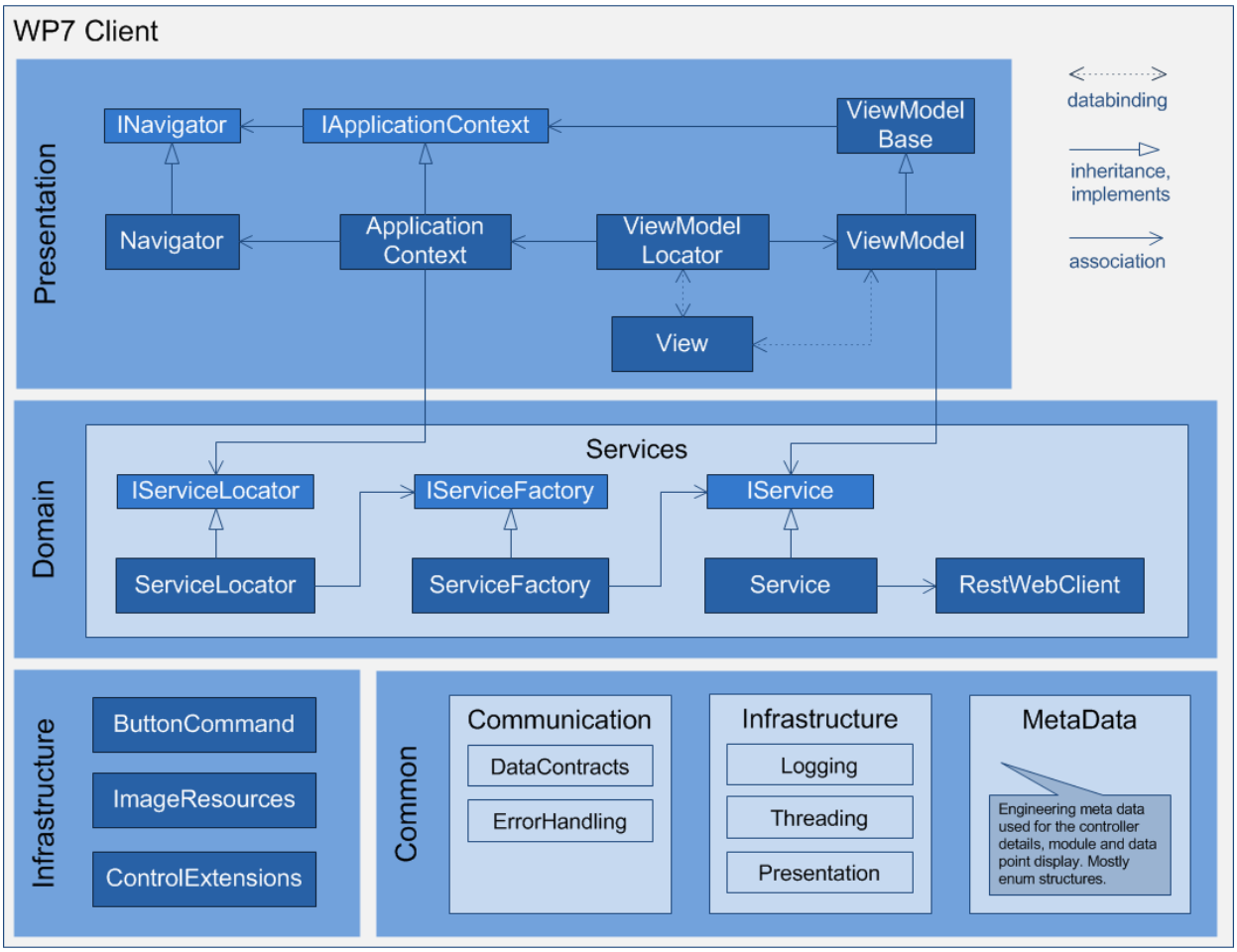


Abbildung 30: SCT-App Architekturübersicht

Presentation-Layer
Im Presentation-Layer wird die Trennung von Business-Logik und Darstellung mit Hilfe des MVVM-Patterns und *Databinding* erreicht. Der Einsatz des MVVM-Patterns und der detaillierte Aufbau des Presentation-Layers wird in Abschnitt 9.2.2 genauer beschrieben. In der obenstehenden Grafik stehen die Klassen *„View“* und *„ViewModel“* nur als Platzhalter für die vielen spezifischen *View*- und *ViewModel*-Klassen, damit die Grafik übersichtlich bleibt.

Domain-Layer
Der Domain-Layer enthält fast ausschliesslich die Service-Klassen. In der obenstehenden Grafik sind die

spezifischen Service-Klassen nicht enthalten und werden durch die Platzhalter *„IService“* und *„Service“* repräsentiert. Die Service-Klassen enthalten keine eigene Business-Logik. Sie kapseln nur die Aufrufe für die jeweiligen Webservice-Funktionen sowie die Fehlerbehandlung und Rückgabe der Resultate. Die eigentliche Kommunikation über das HTTP-Protokoll und die JSON-Serialisierung/Deserialisierung übernimmt die *„RestWebClient“*-Klasse. Aus den *ViewModel*-Klassen erfolgt der Zugriff auf die Service-Klassen immer zuerst über den *ServiceLocator*.

Infrastructure-Layer
Der Infrastructure-Layer enthält ein paar wenige Klassen mit Basisfunktionalität, wie den Zugriff auf Bild-Ressourcen oder Hilfsklassen, die fehlende Silverlight-Funktionen kompensieren.

Common-Bibliothek
In der Common-Bibliothek befinden sich Infrastrukturklassen, Funktionalität für das *Logging* und die Fehlerbehandlung, sowie alle *„Data Contract“*-Klassen (Klassen für den Datenaustausch über REST). Ebenfalls sind Engineering-Metadaten enthalten, welche für die Darstellung der Module, Datenpunkte und anderer Informationen, die auf den Kontrollern gespeichert sind, benötigt werden.

9.2.2. MVVM-Pattern mit View-Model-Locator

Das MVVM-Pattern (*Model-View-ViewModel*) ist ein Entwurfsmuster, welches bei der Entwicklung der Benutzeroberfläche hilft, Daten, Verhalten und Darstellung voneinander zu trennen. Grundsätzlich gesehen ist es nur eine Variante des jahrzehntealten *Model-View-Controller*-Patterns. MVVM wird vor allem bei Projekten eingesetzt, welche auf den Microsoft-Technologien *Windows Presentation Foundation* (WPF) und Silverlight aufbauen. Im Internet sind viele Ressourcen vorhanden, welche sich mit diesem sehr populären Entwurfsmuster eingehend beschäftigen.

Im SCT ist die gesamte Benutzeroberfläche nach dem MVVM-Pattern aufgebaut. Richtig mächtig wird das MVVM-Pattern aber erst, wenn zusätzlich *Databinding* eingesetzt wird. Dadurch wird die Kopplung zwischen den Klassen noch weiter verringert und viel wichtiger, es muss weniger Code geschrieben werden. Bei SCT wird, wo überall möglich, *Databinding* eingesetzt. Als Höhepunkt sind sogar die *ViewModel*-Klassen per *Databinding* an die jeweilige View-Klasse gebunden. Damit dies funktioniert, wird eine *ViewModel*-Locator-Klasse benötigt, welche für die Instanzierung der *ViewModel*-Klassen verantwortlich ist und auch den Zugriff auf diese *ViewModel*-Instanzen ermöglicht. In der View-Klasse, hier eine *PhoneApplicationPage*, kann so mit einer einzigen Zeile XAML-Code an das entsprechende *ViewModel* gebunden werden:

```
<phone:PhoneApplicationPage
    DataContext="{Binding Source={StaticResource ViewModelLocator},
    Path=DeviceBrowserViewModel}">
```

Damit das alles funktioniert, muss zuerst der *ViewModelLocator* als statische Ressource verfügbar gemacht werden. Dies geht am einfachsten in den XAML-Definitionen der App-Klasse, welche Teil eines jeden WP7-Projektes ist. Mit der folgenden Zeile wird eine neue Instanz der *ViewModelLocator*-Klasse erstellt und als applikationsweite Ressource verfügbar gemacht:

```
<Application.Resources>
    <presentation:ViewModelLocator x:Key="ViewModelLocator"/>
</Application.Resources>
```

9.2.3. Asynchrone Service-Architektur

Bei den meisten Client-Server-Architekturen sind Client und Server einfach zu kontrollieren, aber bei dem was dazwischen liegt, der Datenverbindung, gibt es nur wenig Einflussmöglichkeiten. So kann die Verbindung jederzeit gestört, kurzzeitig unterbrochen oder auch für immer unterbrochen werden. Um mit diesen Situationen möglichst vernünftig umzugehen, ohne dem Benutzer das Leben schwer zu machen, wird eine asynchrone Kommunikation verwendet. Damit das aus der Perspektive des Softwaredesigns auch handhabbar ist, wird eine asynchrone, serviceorientierte Architektur verwendet. Mit dieser werden die REST- bzw. HTTP-spezifischen Kommunikationseigenheiten vor den Service-Konsumenten vollständig versteckt. Die Benutzer eines Services entsprechen in der SCT-App den *ViewModel*-Klassen. In diesen wird die Benutzerinteraktion in Service-Anfragen umgewandelt, welche fast immer in einer Kommunikation mit dem Server enden. Damit die Benutzung der Services für die *ViewModel*-Klassen so einfach wie möglich ist, wird das bekannte Konzept eines Service-Locators eingesetzt. Um eine Instanz eines bestimmten Services zu erhalten, wird nur eine einzige Zeile Code benötigt:

```
IGatewayAuthentication authenticationService =
    Context.ServiceLocator.FindService<IGatewayAuthentication>();
```

Der Aufruf einer Servicemethode sieht dann z.B. folgendermassen aus:

```
authenticationService.Authenticate(UserName, Password, AuthenticateCallback);
```

Das Methodenargument "*AuthenticateCallback*" ist ein Delegate auf eine Methode, welche bei der Rückgabe des Resultates aufgerufen wird:

```
private void AuthenticateCallback(AuthenticateEventArgs e)
{
    if (e.ResultState.State == OperationState.Successful)
    {
        // Successfully logged in, do work...
    }
    else
    {
        // Error occurred:
        MessageBox.Show(e.ResultState.Error.ToString());
    }
}
```

Diese drei Schritte bleiben bei der Benutzung aller Services die gleichen:

1. Instanz des entsprechenden Services über den *ServiceLocator* holen
2. Servicemethode aufrufen und eine Callback-Methode angeben
3. Rückgabe des Resultates in der Callback-Methode verarbeiten

Dabei ist nie klar, wie viel Zeit zwischen Serviceaufruf und Rückgabe des Resultates vergeht. Dies muss bei der Gestaltung sowie dem Verhalten der Benutzeroberfläche berücksichtigt werden.

Der Vorteil an diesem Vorgehen ist, dass die Benutzeroberfläche nie „einfrieren" kann, wenn eine Serviceanfrage etwas länger dauert oder im Fehlerfall ein extrem langes Timeout auftritt. Auch können Fehler so viel schöner behandelt werden und müssen nicht überall mit *Try-Catch*-Blöcken abgefangen werden.

9.2.4. Internationalisierung (I18N)

Statische Texte in der Benutzeroberfläche sind direkt im XAML-Code geschrieben. Da die SCT-App als Prototyp entwickelt wurde, war der Fokus nicht auf einer sauberen Unterstützung einer Mehrsprachigen Benutzeroberfläche. Diese Texte in eine Ressourcen-Datei (*.resx) auszulagern, wäre aber gut umsetzbar.

Im Gegensatz dazu sind Texte, welche als dynamisch generierter Inhalt auf der Benutzeroberfläche dargestellt werden, bereits in einer Ressourcen-Datei abgelegt. Bei diesen Texten handelt es sich mehrheitlich um Übersetzungen für Enum-Werte der Datenpunkte. Bei diesen Enumerationen handelt es sich zum Beispiel um Einheiten wie C°, km/h, m², etc. oder auch um Signal- oder Fehlerzustände usw. Die Texte für diese Enumerationen werden ebenfalls in der Benutzeroberfläche des Kontrollersimulators benötigt. Daher ist die Ressourcen-Datei Teil der Common-Bibliothek.

9.3. Leistungsfähigkeit (Performance)

Microsoft stellt mit Windows Phone 7 relativ hohe Leistungsanforderungen an die Smartphone-Hersteller, um eine möglichst schnelle und verzögerungsfreie Bedienung zu garantieren. Man merkt tatsächlich schon bei der erstmaligen Verwendung eines Windows Phone 7-Gerätes, dass das Betriebssystem speziell auf Performance optimiert wurde. Alle Systemfunktionen sowie die von Microsoft erstellten Apps reagieren extrem schnell und es gibt weder Ruckeln noch Ladebildschirme oder Sanduhren. Bei Apps von Drittherstellern sieht dies hingegen ganz anders aus. Oft kommen Ladebildschirme mit beträchtlichen Wartezeiten vor oder beim Scrollen von längeren Listen stockt der Bildaufbau. Schon hier zeigt sich also, dass es nicht ganz einfach ist, eine App zu entwickeln, die über eine ansprechende Optik verfügt und zugleich eine störungsfreie Benutzung erlaubt.

Bei der Entwicklung der SCT-App stellt die Performance vor allem bei der Darstellung der Gebäudehierarchie ein grösseres Problem dar. Es hat sich herausgestellt, dass die baumähnliche Struktur schon bei relativ wenigen Elementen zum Leistungsfresser wird. Dies äussert sich dahingehend, dass es sehr lange dauert, bis die Gebäudehierarchie auf dem Bildschirm erscheint. Das Scrollen funktioniert hingegen recht gut und fast ruckelfrei.

Die eigentliche Ursache für die langen Ladezeiten liegt im Aufbau der Baumstruktur, welche aus vielen in sich geschachtelten Listen besteht. Das verwendete *ItemsControl*, welche die Darstellung einer Liste von beliebigen Elementen erlaubt, benötigt einiges an Rechenleistung. Bei vielen Hierarchiestufen und Elementen ergibt das schnell eine ganze Menge solcher *ItemsControls*, die gleichzeitig in den Speicher geladen werden müssen. Da alle diese visuellen Elemente in XAML-Code definiert werden, muss diese XAML-Struktur zuerst geparst und im Speicher, als sogenannter *Visual-Tree*, aufgebaut werden. Grundsätzlich kann festgehalten werden, je mehr Elemente sich in diesem *Visual-Tree* befinden, desto mehr Speicher und Rechenleistung wird benötigt. Wie sich gezeigt hat, ist hier die Smartphone-Hardware relativ schnell überfordert.

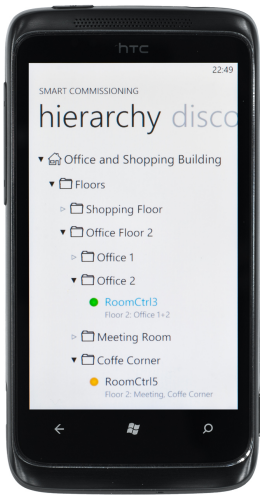


Abb. 31: Gebäude-hierarchie

Zur Behebung der Performance-Probleme bei der Darstellung der Gebäudehierarchie haben sich die folgenden zwei Lösungen angeboten:

- Flache Liste: Die hierarchische Struktur durch eine flache Liste mit unterschiedlichen linksseitigen Einrückungen für die jeweiligen Ebenen ersetzen. Dadurch wird der *Visual-Tree* schlanker.
- Lazy-Loading: Anfänglich nur die Elemente der obersten Ebene laden und die restlichen Elemente erst nachladen, sobald der Benutzer eine Ebene tiefer navigieren will. Dadurch wird die Ladezeit aufgeteilt und auf einen späteren Zeitpunkt verschoben. Der Nachteil ist, dass beim Navigieren bzw. Öffnen einer tieferen Ebene eine kurze Verzögerung entsteht.

Auf Grund der relativ einfachen Implementation und der garantierten Wirksamkeit ist bei der SCT-App der *Lazy-Loading*-Ansatz implementiert worden. Als endgültige Lösung müsste der erste Ansatz ausprobiert werden, oder zumindest versucht werden, durch intelligenteres Nachladen, die auftretenden Verzögerungen beim *Lazy-Loading* zu vermindern.

9.4. Windows Phone Ausführungsmodell (Execution Model)

Das Ausführungsmodell von Windows Phone 7 erlaubt generell kein Multitasking. Das bedeutet, dass nur eine Applikation gleichzeitig im Arbeitsspeicher geladen sein darf. Nur gewisse Systemprogramme und Microsoft Apps, wie z.B. die Musikwiedergabe, dürfen beim Starten einer anderen Applikation im Hintergrund weiterlaufen. Das Ausführungsmodell zusammen mit einem ausgeklügelten Bedienkonzept lassen den Benutzer aber im Glauben, dass trotzdem mehrere Applikationen gleichzeitig laufen.

So wird jeweils beim Navigieren durch die einzelnen Seiten einer App, jede Seite auf einen Stapel abgelegt. Durch Drücken der Starttaste kann eine weitere Applikation gestartet werden, wobei die erste App deaktiviert und aus dem Speicher entfernt wird. Wenn der Benutzer nun in der aktiven App weiter navigiert, werden wiederum alle besuchten Seiten auf den Stapel abgelegt. Das geht so weiter, bis der Stapel gefüllt ist und die zuerst geöffneten Seiten zuunterst wieder aus dem Stapel herausfallen. Durch Drücken der Zurücktaste wird nun die letzte Seite vom Stapel genommen und wiederhergestellt. Sobald der Benutzer durch wiederholtes Drücken der Zurücktaste zur letzten Seite der ersten App zurückkommt, wird die zweite App endgültig beendet und die erste App wieder in den Speicher geladen und aktiviert.

Der Vorgang, bei dem eine App deaktiviert und aus dem Speicher entfernt wird, wird als Tombstoning bezeichnet. Das „Back-Button"-Konzept, bei dem der Navigationspfad über mehrere Apps hinweg auf einem Stapel abgelegt wird und beim Drücken der Zurücktaste wiederhergestellt wird, lässt den Benutzer im Glauben, dass die vorhergehend aufgerufenen Apps die ganze Zeit im Hintergrund weitergelaufen sind. Dabei überlässt es Microsoft dem Anwendungsentwickler, bei der Deaktivierung der App, den Applikationszustand zu speichern und bei der erneuten Aktivierung, den vorhergehenden Zustand wiederherzustellen.

Das gesamte Konzept für die Speicherung und Wiederherstellung des Applikationszustandes ist relativ umfangreich und es sind einige Spezialfälle zu beachten. Microsoft stellt ein paar rudimentäre Hilfsmittel zur Verfügung, welche die Handhabung etwas erleichtern. Grundsätzlich muss zwischen der Speicherung des Applikations- und des Seitenzustandes unterschieden werden. Mit ersterem ist der globale Applikationskontext gemeint und mit letzterem der Zustand der einzelnen Controls auf der Benutzeroberfläche. Die Zustände der Controls sind z.B. der Inhalt einer Textbox, die Position einer Scrollbar oder das aktuell fokussierte Control.

Bei der SCT-App wird zurzeit nur der Applikationszustand gespeichert. Da der gesamte applikationsweite Zustand in der ApplicationContext-Klasse enthalten ist, gestaltet sich dessen Speicherung und Wiederherstellung sehr einfach. Das Speichern und Laden der Seitenzustände wurde aus Zeitgründen nicht implementiert. Dies ist auch nicht weiter kritisch, sondern eher als ein „Schönheits-Fehler" anzusehen.

Applikationszustand sowie Seitenzustand sind als flüchtige Daten zu bezeichnen, die nur wieder geladen werden, falls die Applikation reaktiviert wird. Diese Daten überleben eine endgültige Beendigung der Applikation nicht und werden gelöscht. Daten, welche die Beendigung einer Applikation überleben und bei einer frisch gestarteten Applikation wieder vorhanden sein sollten, müssen im *IsolatedStorage* gespeichert werden. Dies wird beim SCT z.B. mit den Login-Daten so gemacht.

9.5. Benutzeroberfläche (UI)

Kurze Erklärung der benutzten Gesten bei der Touch-Bedienung:

- Antippen (Tap)
  Die Geste „Antippen“ ist eine kurze Berührung des Bildschirms mit einem Finger in einem definierten Bereich.
- Flick-Bewegung (Flick)
  Die Flick-Bewegung ist eine Berührung mit einem Finger und eine schnelle Bewegung nach links oder rechts.
- Halten (Hold)
  Die Geste „Halten“ ist eine lange Berührung ohne Bewegung in einem definierten Bereich.

9.5.1. Gebäudehierarchie und Gerätesuche

Um einen Kontroller zu suchen gibt es zwei Möglichkeiten: Die Gebäudehierarchie und die Gerätesuche. Zwischen diesen beiden Sichten kann mit einer Flick-Bewegung nach links oder rechts gewechselt werden. Ermöglicht wird dies durch das *„Pivot Control“* von Microsoft. Zum *„Pivot Control“* gehört auch die Titelleiste, in der jeweils der Titel der nächsten Ansicht grau dargestellt wird. Ein antippen des grauen Textes bewirkt einen Wechsel zu dieser Ansicht.

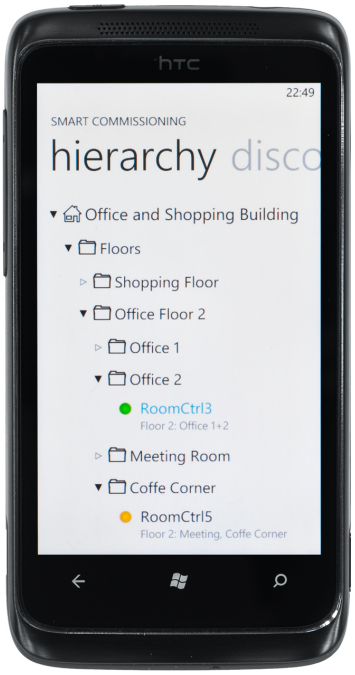


Abb. 32: Gebäudehierarchie

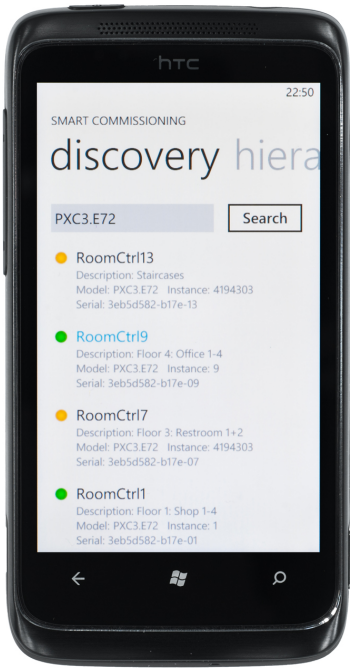


Abb. 33: Gerätesuche

Gebäudehierarchie
Die Gebäudehierarchie ist eine hierarchische Darstellung der Gebäude, Stockwerke und Räume. In den Räumen werden zusätzlich die Kontroller dargestellt, welche für diesen Raum zuständig sind.

Gerätesuche (device discovery)
Die Gerätesuche ist eine flache Darstellung von allen Geräten im Netzwerk. Grundsätzlich erscheinen dort nicht nur Raumkontroller, sondern auch Kontroller für Primäranlagen oder Router. Deshalb ist es notwendig nur nach einem bestimmten Gerätetyp wie z.B. PXC3.E72 (Raumkontroller) suchen zu können. Die Gerätesuche wird hauptsächlich benötigt, um Kontroller, die noch keinem Raum zugeordnet sind, zu suchen und zu identifizieren. Für die Identifizierung eines Kontrollers bestehen zwei Möglichkeiten: *Flash LED* und *Service Pin*. Auf jedem gefundenen Kontroller kann ein *„flash LED“* ausgeführt werden, was zum Aufleuchten einer LED beim Kontroller führt. Auf der anderen Seite kann die *„Service Pin“*-Taste beim Kontroller gedrückt werden, was zur Selektion des Kontrollers in der Gerätesuche führt. Zusätzlich zur grafischen Hervorhebung vibriert das WP7-Phone beim Empfangen eines *Service Pins,* um dem Benutzer eine bessere Rückmeldung zu geben.

9.5.2. Kontrollerinformationen und Punktauswahl

Um von der Gebäudehierarchie oder der Gerätesuche aus weiter zu navigieren, sind mehrere Möglichkeiten vorhanden. Durch das Halten auf einem Kontroller wird das Kontextmenü geöffnet und durch das Antippen eines Kontrollers gelangt man in die Punktauswahl.

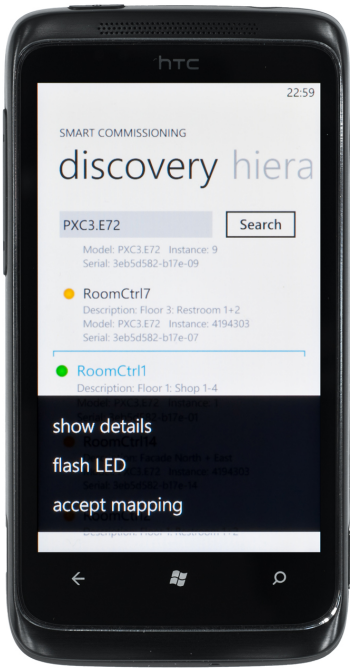


Abb. 34: Kontextmenü

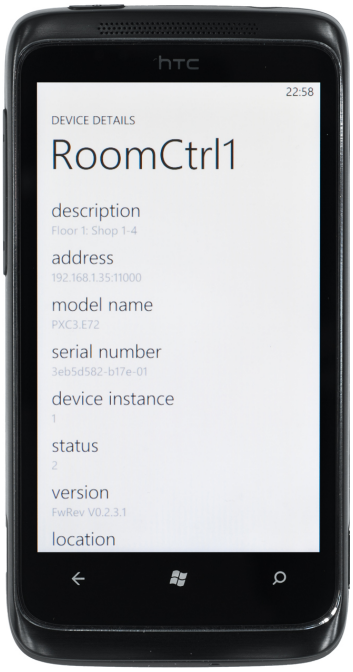


Abb. 35: Kontrollerinformationen

Abb. 36: Punktauswahl

Kontextmenü
Über das Kontextmenü sind folgende Funktionen möglich:

- Anzeigen von zusätzlichen Kontrollerinformationen.
- Senden eines „*flash LED*“ an den Kontroller.
- Mit einer Kontroller-Hardware verbinden (nur in der Gebäudehierarchie-Sicht).
- Verbindung zur Kontroller-Hardware akzeptieren (nur in der Gerätesuche-Sicht).

Kontrollerinformationen
Unter Kontrollerinformationen werden alle Daten dargestellt, welche über diesen Kontroller verfügbar sind. Häufig sind das zusätzliche Daten wie Ort, Adresse, Firmware-Version, freier Speicher, usw.

Punktauswahl
Der erste Schritt für den Datenpunkttest ist die Punktauswahlsicht. In der Punktauswahlsicht werden alle TXIO-Module eines Kontrollers mit ihren Datenpunkten aufgelistet. Bei jedem Datenpunkt werden die Zuverlässigkeit und der Inbetriebnahme-Status dargestellt. Damit ist einfach ersichtlich welche Datenpunkte bereits überprüft wurden. Das Attribut „Zuverlässigkeit“ kann unter anderem folgende Werte haben: Kein Sensorsignal, über dem Limit, unter dem Limit, falsche Adresse, Prozessfehler, Konfigurationsfehler, usw. Der Inbetriebnahme-Status ist entweder auf „nicht geprüft“, „geprüft und falsch“ oder „geprüft und ok“.

Aus der Punktauswahl-Sicht wird ein Datenpunkttest gestartet, indem ein Datenpunkt angetippt wird.

9.5.3. Datenpunkttest

Der Datenpunkttest hat die drei Ansichten: Punktübersicht, Punktbearbeitung und Punktinformationen. Zwischen diesen drei Ansichten kann durch Antippen der grauen Titel oder mit einer Flick-Bewegung gewechselt werden. Die Informationen eines Datenpunktes werden einmal pro Sekunde neu angefragt und dargestellt. Die Daten existieren im Hintergrund nur einmal, werden für die Darstellung einfach auf drei Sichten verteilt.

Abb. 37: Punktübersicht

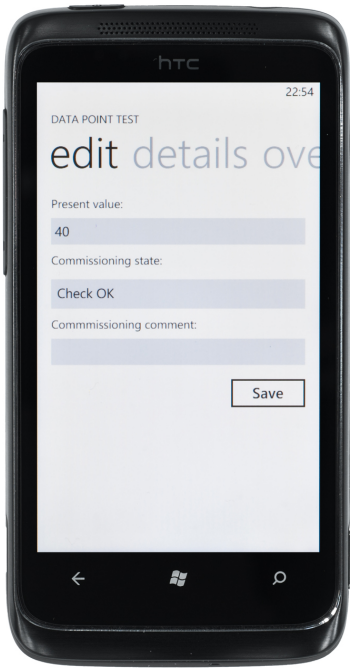


Abb. 38: Punktbearbeitung

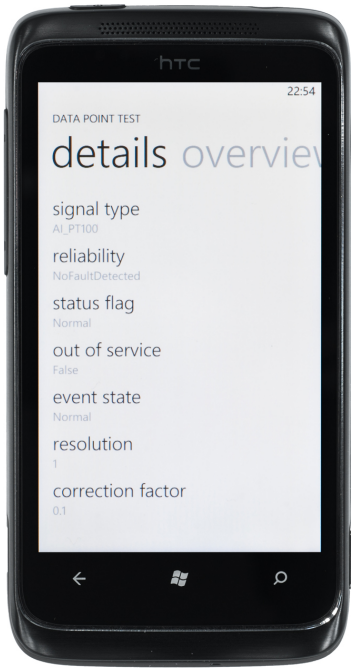


Abb. 39: Punktinformationen

Punktübersicht
Alle für den Datenpunkttest nötigen Informationen werden in der Punktübersicht zusammengefasst. Zusätzlich zeigt ein Trend-Graph die Wertänderungen über die Zeit. Der Trend-Graph ist einer der wichtigsten Hilfsmittel beim Datenpunkttest, denn damit wird die Wertänderung beim Entfernen des Kabels vom Sensor auch grafisch dargestellt. Falls der Wert nicht auf 0 geht, ist dadurch schnell ersichtlich, dass ein Verdrahtungsfehler vorliegt.

Der Trend-Graph wurde mit einem Liniengraph implementiert und skaliert sich bei genügend grosser Änderung des Maximum- und Minimum-Wertes neu.

Punktbearbeitung
Die Punktbearbeitungssicht dient dazu, den Inbetriebnahme-Status, sowie bei einem Fehler, den Inbetriebnahme-Kommentar, zu setzen. Beim Datenpunkttest von einem Signalausgang wird diese Sicht zusätzlich benötigt, um den Wert des Signalausganges manuell zu überschreiben. Nur durch das manuelle Überschreiben eines Signalausganges kann während dem Datenpunkttest z.B. das Licht betätigt werden. Denn im normalen Betrieb werden Signalausgänge durch die Logik im Kontroller berechnet und können nicht geändert werden.

Punktinformationen
Die Punktinformationssicht beinhaltet alle weiteren Informationen eines Datenpunktes. Dazu gehört unter anderem die Auflösung des Signals, Korrekturfaktoren, Signaltypen, Polarität, usw. Die zusätzlichen Informationen sind abhängig vom Datenpunktyp. Analoge Signale verfügen über andere Informationen als binäre oder mehrstufige Signale.

9.6. Lösungen für die Benutzeroberfläche aus technischer Sicht

9.6.1. Gebäudehierarchie-Ansicht

Die Gebäudehierarchie wird in einer Art Baumstruktur dargestellt. Für solche Strukturen wird auf den meisten Systemen ein sogenanntes *TreeView*-Bedienelement benutzt. Bei Silverlight für Windows Phone ist jedoch weder eine *TreeView* vorhanden noch wird eine andere einfache Lösung geboten, um eine hierarchische Struktur darzustellen. In einschlägigen Diskussionsforen wird zu diesem Thema meist darauf hingewiesen, dass sich komplexe hierarchische Strukturen grundsätzlich nicht für eine Bedienung auf einem kleinen Smartphone-Bildschirm eignen. Es wird aber auch darüber spekuliert, dass Microsoft das fehlende *TreeView*-Bedienelement in einer der nächsten Aktualisierungen mitliefern wird.

Als Lösung für die Darstellung der Gebäudehierarchie wurde eine eigene Baumansicht entwickelt (siehe Abbildung 32). Jedes Gebäudeelement kann dabei eine unbeschränkte Anzahl Unterelemente beinhalten. Die Unterelemente werden dazu einem *ItemsControl*, also einem Listen-Element, hinzugefügt. Beim Auf- und Zuklappen wird diese Liste ausgeblendet. Da alle Gebäudeelemente vom gleichen Typ sind und somit Unterelemente besitzen können, entsteht eine hierarchische Struktur von beliebiger Tiefe. Da der Smartphone-Bildschirm relativ klein ist, sind aber nicht mehr als vier bis fünf Stufen sinnvoll.

Der gewählte Ansatz wurde aus Zeitgründen nicht generisch umgesetzt und funktioniert somit nur für die Darstellung der Gebäudeelemente und für keine anderen hierarchischen Strukturen.

9.6.2. Application Bar

Die *Windows Phone Application Bar* ist sozusagen das Pendant zur Menüleiste oder Toolbar einer Desktopanwendung. Die *Application Bar* wird zuunterst auf dem Bildschirm angezeigt und wird vollständig von der aktiven App kontrolliert. Es finden bis zu vier Icons auf der *Application Bar* Platz. Zusätzlich existiert noch ein erweitertes, rein textuelles Menü, das von der *Application Bar* aus aufgerufen werden kann und bis zu 50 weitere Menüpunkte aufnimmt. Es besteht jederzeit die Möglichkeit, die *Application Bar* sowie einzelne Icons bzw. Menüpunkte ein- oder auszublenden. Die folgenden Screenshots stammen von der Windows Phone Internet Explorer App.

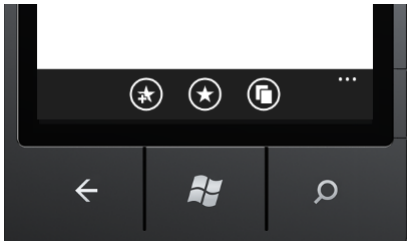


Abbildung 40: Application Bar (zugeklappt)

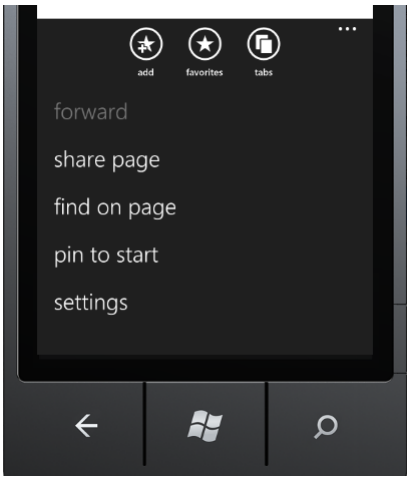


Abbildung 41: Application Bar (aufgeklappt)

Es empfiehlt sich, die *Application Bar* als möglichst statisches Element zu verwenden, da häufiges Ändern der Icons den Benutzer leicht verwirren könnte. Da im SCT-App die Funktionen meistens vom aktuell ausgewählten Kontroller abhängig sind, ändert sich die Auswahl der möglichen Funktionen sehr häufig. Des Weiteren ändern die Funktionen von Seite zu Seite. Diese Argumente haben dazu geführt, die *Application Bar* im SCT nicht zu verwenden, sondern Kontextmenüs zu implementieren.

Da die *Application Bar* ein Systemobjekt und nicht direkt Teil des XAML-*Visual-Trees* ist, gilt es ein paar Besonderheiten zu beachten. Obwohl die *Application Bar* im XAML-Code definiert werden kann, funktioniert weder *Databinding* noch Command-Binding. Anstatt dessen ist ein Workaround vorhanden, bei dem die *ApplicationBarButtonCommand*-Klasse der Prism-Library verwendet wird. So können die *Application Bar* Buttons an *Commands* gebunden werden.

9.6.3. Kontextmenü

Obwohl in den Microsoft-eigenen Apps ein Kontextmenü verwendet wird, ist es in den Standardbibliotheken von *Silverlight for Windows Phone* nicht enthalten. Dafür wird dieses wichtige Bedienelement im *Silverlight for Windows Phone Toolkit* nachgeliefert, welches Aussehen und Verhalten des Kontextmenüs aus den Microsoft-eigenen Apps imitiert.

Im SCT-App wird das Toolkit-Kontextmenü bei der Kontroller-Auswahl in der Gebäudehierarchie- sowie in der Gerätesuche-Ansicht eingesetzt (siehe Abbildung 34).

Bei der Benutzung des Kontextmenüs tritt ein Problem auf: Sobald ein Menüpunkt per Tippen angewählt wird, wird der Tipp-Befehl auch an das dahinterliegende Bedienelement gesendet. Im Fall der Gebäudehierarchie- und Gerätesuche-Ansichten führt dies unter Umständen dazu, dass zwei Aktionen gleichzeitig ausgeführt werden, wenn sich hinter dem gewählten Menüpunkt direkt ein Kontroller befindet. Die Ansichten reagieren also ganz normal auf Tipp-Befehle, wie wenn gar kein Kontextmenü geöffnet wäre. Dies scheint ein Fehler in der Implementation des Kontextmenüs oder des verwendeten *Toolkit-Gesture-Listeners* zu sein, welche auf dieselben Bewegungsmuster reagieren. Aber möglicherweise ist dies auch Auslegungssache, weil diese Bedienelemente vielleicht gar nicht für eine gemeinsame Benutzung ausgelegt sind. Da aber an keiner Stelle auf eine solche Limitation hingewiesen wird, kann nicht von einem Implementationsfehler seitens SCT ausgegangen werden.

Als Workaround wurde das Problem folgendermassen gelöst: Beim Öffnen des Kontextmenüs wird ein Flag gesetzt, welches, solange aktiv, die Ausführung aller Aktionen verhindert. Sobald das Kontextmenü geschlossen

wird, was z.B. bei der Auswahl eines Menüpunktes geschieht, wird das Flag nach einem Timeout von 150ms wieder gelöscht:

```
private void ControllerContextMenu_Opened(object sender, RoutedEventArgs e)
{
    IsContextMenuOpened = true;
}

private void ControllerContextMenu_Closed(object sender, RoutedEventArgs e)
{
    // Use a timeout of 150ms before resetting the flag to prevent other
    // action listeners from being executed after clicking a context menu item.
    new Thread(() =>
    {
        Thread.Sleep(150);
        IsContextMenuOpened = false;
    }).Start();
}
```

9.6.4. Popups für nicht wiederkehrende Seiten

Als Regel des Windows Phone Ausführungsmodells sollten Seiten, welche bei der Verwendung der Zurücktaste nicht wieder angezeigt werden dürfen, keine eigenständigen Seiten sein. D.h. sie dürfen nicht von der *PhoneApplicationPage*-Klasse ableiten. Das betrifft in der SCT-App die Login- und Fehlerseiten, welche anstatt als eigenständige Seiten, als bildschirmvolle Popups implementiert wurden. So können diese Popups jederzeit ein- und ausgeblendet werden. Dabei werden sie aber nie auf den Stapel für die „Back-Button"-Funktion abgelegt.

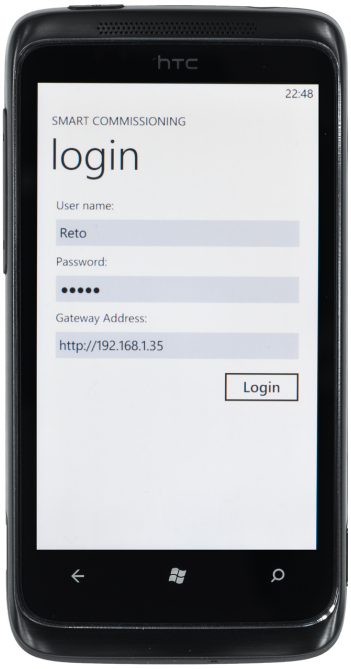


Abbildung 42: Login Dialog

9.6.5. Windows Phone Themes

Auf dem Windows Phone 7 gibt es Einstellungen, um die Farben für die Darstellung den eigenen Vorlieben anzupassen. Grundsätzlich sind zwei *Themes* vorhanden. Ein dunkles, bei dem die Grundfarbe Schwarz ist und ein helles, welches Weiss als Grundfarbe hat. Zusätzlich zur Grundfarbe kann eine Akzentfarbe definiert werden. Die Akzentfarbe wird für Bereiche/Texte benutzt, welche selektiert sind oder grafisch hervorgehoben werden sollen. Die SCT-App benutzt diese Akzentfarbe für selektierte Elemente und ist kompatibel mit dem dunklen sowie dem hellen *Theme*. Dafür bietet es zwei verschiedene Bildersätze, damit die Bilder auf den unterschiedlichen Hintergründen genügend Kontrast bieten.

Abb. 43: Schwarzer Hintergrund

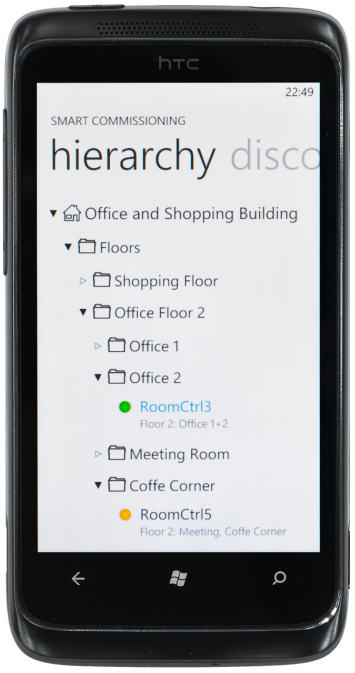


Abb. 44: Weisser Hintergrund

9.6.6. List Picker

Das Combobox-Bedienelement wird auf dem Windows Phone nicht angeboten, da es aus der Welt der Desktopanwendungen stammt und nicht für die Touch-Bedienung ausgelegt ist. Trotzdem bleibt natürlich die eigentliche Problemstellung auf dem Smartphone dieselbe: Wie kann der Benutzer aus einer gegebenen Liste von Elementen eines auswählen, ohne dass das Bedienelement viel Platz benötigt. Als Lösung bietet das *Silverlight for Windows Phone Toolkit* ein *List Picker*-Bedienelement an. Der *List Picker* zeigt normalerweise nur das ausgewählte Element an. Mit Antippen wir der *List Picker* zu einer vollwertigen Liste vergrössert, woraus ein anderes Element durch ein weiteres Antippen „herausgepickt“ werden kann. Danach wird die Liste wieder verkleinert und es wird nur noch das neu ausgewählte Element angezeigt. Der *List Picker* zeigt dieses Verhalten jedoch nur bei weniger als sechs Elementen. Bei mehr Elementen wird das Bedienelement nicht einfach vergrössert, sondern es erscheint ein bildschirmfüllendes Popup, das eine scrollbare Liste der Elemente zeigt. Sobald ein Element ausgewählt wurde, schliesst sich das Popup wieder.

Der *List Picker* wurde erst in der November-Aktualisierung des *Silverlight for Windows Phone Toolkits* veröffentlicht und scheint noch ein paar Bugs zu haben. So führt z.B. das *Databinding* in gewissen Situationen zu Fehlern, aber nur wenn der Visual Studio Debugger nicht angeheftet ist. Darum musste bei der Entwicklung der SCT-App beim *List Picker* vollständig auf *Databinding* verzichtet werden.

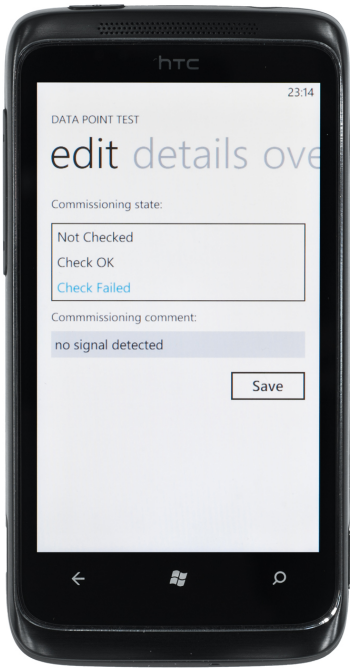


Abb. 45: List Picker für Commissioning State

10. Schlussfolgerungen

10.1. Zusammenfassung

Diese Bachelorarbeit hat zum Ziel, den Inbetriebnahmeprozess softwareseitig zu verbessern, um eine bessere Unterstützung des Inbetriebnahme-Ingenieurs zu erreichen. Dieses Ziel wurde während der gesamten Projektdauer konsequent verfolgt. Der Grundstein wurde mit einer sorgfältigen Analyse des bestehenden Umfeldes und dem Sammeln von technischen sowie anwenderspezifischen Anforderungen und Gegebenheiten gelegt. Dies erforderte die Einbeziehung und Befragung von verschiedenen Personen aus dem Problembereich. Die meisten dieser Personen arbeiten zurzeit bei der Siemens am neuen Engineering-Tool für die Raumautomation, wobei sie ganz unterschiedliche Funktionen einnehmen. Darunter sind Systemarchitekten, Entwickler, ein Produktedesigner sowie eine Kontaktperson der Schweizer-Ländervertretung. Durch Einbezug dieser Fachpersonen konnte die fachliche Qualität hochgehalten werden. Darüber hinaus wurde dadurch auch vermehrtes Interesse an dieser Arbeit geweckt.

Als erstes Resultat wurde ein Systemkonzept ausgearbeitet und die Machbarkeit durch Tests mit Prototypen wiederholt überprüft. Das Systemkonzept erlaubt die optimale Integration der neu entwickelten Software in den bestehenden Inbetriebnahmeprozess. Dem Systemkonzept liegt eine Client-Server-Architektur zugrunde, welche drei verschiedene Softwareteile umfasst, die alle als Teil dieser Arbeit konzipiert und umgesetzt wurden. So wurde das *Smart Commissioning Tool* (SCT) als App für Windows Phone 7 entwickelt, welche den Inbetriebnahme-Ingenieur bei seiner Arbeit bestmöglich unterstützt. Als Ergänzung wurde ein Software-Gateway als PC-Anwendung entwickelt, der die Kommunikation zwischen Smartphones und Kontrollern vereinfacht. Zusätzlich zur entwickelten Software wurde eine Webservice-Schnittstellen-Spezifikation erarbeitet, die zukünftig direkt in der Kontroller-Firmware umgesetzt werden könnte. Damit SCT-App und Gateway dennoch, trotz nicht vorhandener Kontroller-Hardware, getestet werden konnten, wurde ein Kontroller-Simulator als PC-Anwendung entwickelt. Dieser implementiert einen Grossteil der spezifizierten Webservice-Schnittstelle und simuliert so die vollständige Kommunikation mit der Kontroller-Hardware.

Aus Sicht des Designs konnten in der SCT-App sowie im Gateway eine saubere und stabile Architektur ausgearbeitet werden. Bei der SCT-App wurde in der Präsentationsschicht durchgängig das MVVM-Pattern mit *Databinding* und *ViewModel-Locator* verwendet. Der Zugriff auf die zugrundeliegenden Webservices wurde in modularen Service-Klassen gekapselt, welche eine asynchrone, jedoch nach wie vor einfache Verwendung erlauben. Beim Gateway wurde insbesondere auf die Implementation einer möglichst einfach zu verwendenden Schnittstelle geachtet. Der Gateway abstrahiert das dahinterliegende „Chaos“ des Kontroller-Netzwerkes und stellt dem SCT-App die benötigten Informationen massgeschneidert zur Verfügung. Ebenfalls wurden Themen wie Threadsicherheit und Caching grosse Beachtung geschenkt. Beim Simulator liegen die Schwerpunkte vor allem auf der möglichst leichten Eingabe von Testdaten sowie der zweckmässigen Umsetzung der definierten Kontroller-Webservice-Schnittstelle.

Die Benutzeroberfläche der SCT-App wurde anwendergerecht umgesetzt, wobei die neue Windows Phone 7 Plattform zahlreiche Knacknüsse geboten hat. Es mussten einige Workarounds gefunden, sowie neue Lösungen entwickelt werden, bis die Benutzeroberfläche den Erwartungen entsprach.

Zur Qualitätssicherung wurden Codereviews sowie Systemtests eingesetzt. Die Systemtests wurden nach jedem Release manuell durchgeführt. Dadurch konnte auch der Implementationsstand der einzelnen *UseCases* überprüft werden.

10.2. Beurteilung der Resultate

Alle in der Aufgabenstellung geforderten nicht-optionalen Anforderungen konnten während der Bachelorarbeit umgesetzt werden. Dieser Erfolg ist darauf zurückzuführen, dass zu Beginn der Arbeit alle weniger wichtigen Anforderungen als optional deklariert wurden. Dafür konnte die volle Energie für die Umsetzung der essentiellen Punkte aufgewendet werden. Dass sich diese Strategie ausbezahlt hat, zeigte sich bereits am durchwegs positiven Feedback der involvierten Schlüsselpersonen.

Die Umgesetzte Lösung bringt unter anderem die folgenden wichtigen Vorteile und Verbesserungen:

- Smartphone ermöglicht Bewegungsfreiheit
- «One man commissioning» wird unterstützt
- Ergebnisorientierter Workflow
- Gebäudehierarchie vereinfacht die Suche nach Kontrollern
- All-in-one Lösung mit einer einzigen Anwendung als Frontend
- Durchführung von Tests mit separatem Simulator

Das Generieren eines Inbetriebnahme-Reports ist als wichtigster, noch offener Punkt anzusehen. Diese Anforderung wurde zwar als optional deklariert, würde aber sicher den grössten zusätzlichen Nutzen bringen. Durch eine einfache Möglichkeit zur Generierung eines solchen Reports, würde dem Inbetriebnahme-Ingenieur bereits viel Arbeit abgenommen.

Das erarbeitete Systemkonzept erlaubt den einfachen Austausch des Simulators gegen die reale Umgebung. Einzige Voraussetzung dafür ist, dass die spezifizierte Webservice-Schnittstelle direkt in der Firmware des Hardware-Kontrollers implementiert wird. Die Funktion des Gateways für das Suchen und Auflisten der Kontroller über das DICP-Protokoll würde aber bereits schon mit den echten Kontrollern funktionieren. Dagegen ist der Export der Gebäudehierarchie aus dem ABT-Engineering-Tool noch nicht vorhanden. Die benötigten Daten könnten aber relativ einfach aus dem bestehenden DESIGO Insight-Export extrahiert werden.

Mit der Darstellung einer Gebäudehierarchie zum einfachen Auffinden von Kontrollern sowie der smartphonegerechten Umsetzung des Datenpunkttests bietet die SCT-App sicher interessante Innovationen. Nun können Arbeiten mit einer Smartphone-App erledigt werden, für die bis anhin die Verwendung von zwei separaten Desktopanwendungen erforderlich waren. Dadurch wird nicht nur Zeit, sondern auch Schulungs- und Einführungskosten gespart.

Mit dem neuen Windows Phone 7 ist Microsoft grundsätzlich ein guter Wurf gelungen. Das neue Bedienkonzept ist eingängig und der „Look“ schlicht aber ansprechend. Dadurch ist die Windows Phone 7 Plattform für praktisch alle Benutzergruppen eine ernstzunehmende Option. Aber erst nachdem einige fehlende Grundfunktionen wie *„Copy & Paste“* via Patch nachgereicht werden, kann von einer wirklich konkurrenzfähigen Plattform gesprochen werden. Die Entwicklung für Windows Phone 7 gestaltet sich mit Visual Studio 2010 und dem Phone Emulator sehr komfortabel. Einzig das Silverlight for Windows Phone-Framework, das noch auf Silverlight 3 basiert, entspricht nicht mehr ganz den heutigen Erwartungen. Hier sollten unbedingt die Neuerungen von Silverlight 4 sowie einige vielverwendete Controls hinzugefügt werden.

10.3. Ausblick

Ob sich Windows Phone 7 als die richtige Plattform für diese Art Anwendung erweisen wird, kann noch nicht abschliessend beurteilt werden. Es müssen in Zukunft sicher auch Alternativen, insbesondere mit Tablet-Computern wie dem iPad, geprüft werden. Ebenfalls zu beachten ist, dass die in dieser Bachelorarbeit präsentierte Lösung nur eine Softwarelösung ist, die den bestehenden Inbetriebnahmeprozess vereinfachen soll. Die der Inbetriebnahme zugrundeliegenden Gegebenheiten wurden dabei weder genauer analysiert, noch in Frage gestellt. Bei der nächsten Produktegeneration muss auf jeden Fall die Inbetriebnahme-Problematik noch einmal grundlegend analysiert werden. Die Ergebnisse sollten dann möglichst direkt in das Produktedesign miteinbezogen werden. Damit kann in Zukunft der Aufwand für die Inbetriebnahme sicher noch einmal weiter reduziert werden.

Bei einer Präsentation der Zwischenergebnisse dieser Arbeit wurde bereits das Interesse geweckt, die SCT-App auch für Tätigkeiten einzusetzen, die über die reine Inbetriebnahme hinausgehen. Die Funktionalität von SCT könnte auch während dem normalen Betrieb des Gebäudes zur Funktionsprüfung oder zur „Remote-Steuerung“ der Raumgeräte von Nutzen sein. Das Thema „Smartphone in der Gebäudeautomatisierung“ wird in nächster Zeit zweifelsohne immer wichtiger werden. Es wird interessant sein, mitzuverfolgen, welche neuen Möglichkeiten sich dadurch ergeben werden.

Projektplan

SMART COMMISSIONING

Windows Phone 7

Bachelorarbeit:	Mobile Commissioning Tool for Room Automation
Autoren:	Kaspar Fenner Reto Schneebeli
Betreuer:	Prof. Hansjörg Huser
Projektpartner:	Siemens Schweiz AG Building Technologies, Zug
Experte:	Stefan Zettel, Ascentiv AG, Zürich

0. Dokumentinformationen

0.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
24.09.2010	0.1	Meilensteine, Iterationen (Diagramm und Tabellen)	Reto Schneebeli
24.09.2010	0.2	Projektübersicht, Projektorganisation, Risikoanalyse, Layout überarbeitet	Kaspar Fenner
25.09.2010	0.3	Risikoanalyse erweitert, Kapitel 3, 4, 7, 8 überarbeitet	Kaspar Fenner
26.09.2010	0.4	Zeitschätzungen aus der Zeitplanung hinzugefügt	Reto Schneebeli
29.09.2010	0.5	Risikoanalyse + Iterationsplanung überarbeitet	RS, KF
02.10.2010	0.6	Kapitel 6: Verantwortlichkeiten der Arbeiten hinzugefügt	Reto Schneebeli
18.10.2010	0.7	Planung der Elaboration Phasen 2 und 3 überarbeitet	Reto Schneebeli
31.10.2010	0.8	Risiken überarbeitet , Aufteilung der Implementation und Zeitschätzungen für Construction Phasen 1 und 2	RS, KF
21.11.2010	0.9	MS4 Verschiebung begründet	Kaspar Fenner
22.12.2010	1.0	Überarbeitet für Endabgabe	RS, KF

0.2. Inhalt

0. Dokumentinformationen 2
0.1. Änderungsgeschichte 2
0.2. Inhalt 3
1. Einführung 4
1.1. Zweck 4
1.2. Gültigkeitsbereich 4
1.3. Definitionen und Abkürzungen 4
1.4. Referenzen 4
1.5. Übersicht 4
2. Projekt Übersicht 4
2.1. Zweck und Ziel 5
2.2. Annahmen und Einschränkungen 5
3. Projektorganisation 5
3.1. Organisationsstruktur 5
3.2. Externe Schnittstellen 5
4. Management Abläufe 6
4.1. Projekt Kostenvoranschlag 6
4.2. Wöchentliche Besprechungen 6
4.3. Projektplan 6
4.3.1. Zeitplan 6
4.3.2. Meilensteine 6
4.3.3. Meilenstein-Verschiebungen, Unerwartetes, Erklärungen 7
4.3.4. Iterationsplanung 7
4.3.5. Software Releases 7
5. Risiko Management 7
6. Zeitschätzungen der Arbeiten 9
6.1. Projekt Management 9
6.2. Requirements 9
6.3. Analyse 9
6.4. Design 9
6.5. Implementation 10
6.6. Test 11
6.7. Dokumentation 11
6.8. Technologie Studien (über ganze Projektdauer) 12
6.9. Sitzungen (über ganze Projektdauer) 12
6.10. Qualitätssicherung (über ganze Projektdauer) 12
7. Infrastruktur 12
7.1. Räumlichkeiten 12
7.2. Hardware 12
7.3. Software 13
7.3.1. Entwicklungstools 13
7.3.2. Dokumentation 13
7.3.3. Versionsverwaltung 13
7.3.4. Sonstiges 13
8. Qualitätsmassnahmen 13
8.1. Teamarbeit und Kommunikation 13
8.2. Dokumentation und Planung 13
8.3. Reviews 13
8.4. Backup/Sicherung 14
8.5. Tests 14
8.6. Programmierrichtlinien 14

1. Einführung

1.1. Zweck

Der Projektplan beschreibt das Projekt in seinen Grundzügen und gibt eine Übersicht über den gesamten Projektverlauf. Im Projektplan werden unter anderem die Projektorganisation sowie die Zeitplanung und die Meilensteine festgelegt. Als zentralen Ausgangspunkt werden die Arbeitspakete beschrieben, die dann auch im Zeitplan ersichtlich sind.

1.2. Gültigkeitsbereich

Die Gültigkeit dieses Dokumentes erstreckt sich über die gesamte Projektdauer von 14 Wochen. Änderungen werden laufend nachgetragen und in der Änderungshistorie vermerkt.

1.3. Definitionen und Abkürzungen

Begriff	Beschreibung
.NET	.NET ist eine von Microsoft entwickelte Softwareplattform. Mit dem .NET-Framework wird eine gut ausgestattete Library für die Entwicklung mit .NET bereitgestellt. Aktuelle Version ist 4.0.
OOA	Objektorientierte Analyse
OOD	Objektorientiertes Design
System ONE	System ONE (Engineering-System für die Raumautomation von Siemens BT)
WP7	Windows Phone 7 ist die neue Mobile-Plattform von Microsoft und ersetzt Windows Mobile (Release: Ende Oktober 2010).
SCT	Smart Commissioning Tool
RA	Room Automation (Raumautomation)
HW	Hardware (Physikalisches Gerät)
UI	User Interface (Benutzeroberfläche)

1.4. Referenzen

Projektplan_Zeitplanung.pdf

1.5. Übersicht

Im nächsten Kapitel wird das Projekt in groben Zügen beschrieben. Darauf folgt eine kurze Beschreibung der Projektorganisation. Im vierten Kapitel wird dann der Projektablauf mit einem Verweis auf den Zeitplan beschrieben. Weiter folgt eine Tabelle mit Risiken und Massnahmen sowie die Auflistung der Arbeitspakete. Zum Schluss wird noch die Infrastruktur sowie Massnahmen zur Qualitätssicherung festgelegt.

2. Projekt Übersicht

Im Hauptteil dieser Arbeit wird ein Commissioning-Tool für die Raumautomation entwickelt. Dieses *Smart Commissioning Tool* (SCT) besteht aus einem Webservice als Serverteil sowie aus einem App für Windows Phone 7 (WP7) als Clientapplikation. Der Webservice wird mit C# .NET 4.0 entwickelt und läuft auf einem Notebook, das als Gateway zwischen dem Smartphone und den Raumkontrollern fungiert. Dabei verbindet sich das Smartphone nur mit dem Gateway und nie direkt mit den Raumkontrollern. Die wichtigsten Funktionen der WP7 App sind das Suchen und Auflisten von Raumkontrollern sowie das Durchführen von Datenpunkttests. Die grössten Herausforderungen liegen im Design der Webservice Schnittstelle sowie der Entwicklung einer auf Smartphones angepassten Benutzeroberfläche für die WP7 App. Als optionale Herausforderung bietet sich die Ausarbeitung eines geeigneten Sicherheitskonzeptes.

2.1. Zweck und Ziel

Mit diesem Projekt wollen wir das bisher angeeignete Wissen in Softwareengineerin, objektorientierter Entwicklung sowie User Interface Design anwenden und vertiefen. Als zusätzliche Herausforderung müssen auch Netzwerktechnische Überlegungen gemacht werden sowie die Eigenheiten von Embedded- und Mobile-Systems beachtet werden. Wir wollen uns der Herausforderung stellen, ein Projekt mit einer für uns neuen Plattform (Windows Phone 7) durchzuführen. Des Weiteren möchten wir auch unsere Skills im Team- und Projektmanagement weiter ausbauen und verfeinern.

2.2. Annahmen und Einschränkungen

Pro Teammitglied wird ein Arbeitsaufwand von 25.7 Stunden pro Woche festgelegt. Falls Verzögerungen eintreten, die nicht mehr in sinnvollem Masse durch Mehrarbeit aufgeholt werden können, müssen einzelne weniger kritische Features oder die Bearbeitung von optionalen Themen weggelassen werden (Prioritäten beachten).

3. Projektorganisation

Das Projektteam besteht aus zwei Personen, die einander gleichgestellt sind. Jedes Teammitglied übernimmt die Verantwortung für mehrere Teilaufgaben, an denen dann selbstständig oder gemeinsam gearbeitet wird. Herr H. Huser übernimmt das Projektcontrolling und ist für die Bewertung zuständig. Industriepartner ist die Siemens Schweiz AG. Dort sind Herr R. Föhn und Frau K. Panteleeva die Ansprechpartner.

3.1. Organisationsstruktur

Person	Kürzel	Verantwortlichkeitsbereiche
Kaspar Fenner	KF	• Software: Clientteil (WP7 App), User-Interface, (Netzwerk-)Sicherheitskonzepte, Physikalische Architektur • Dokumentation
Reto Schneebeli	RS	• Software: Serverteil (Webservice, Gateway), Webservice-Interface, Web-Technologies, RA Controller Simulation • Testing (Unit/System) • Versionsmanagement/Backup/Build

3.2. Externe Schnittstellen

Person	Kürzel	Details und Funktion
Hansjörg Huser	HH	Prof. für Informatik, Dozent an der HSR, Leiter INS Beratung, Betreuung, Kontrolle (Reviews), Bewertung
Katja Panteleeva	KP	Project Manager LMS, Siemens Schweiz AG Fachliche Betreuung seitens Industriepartner
René Föhn	RF	Manager DESIGO V5.0 Tools, Siemens Schweiz AG Beratung und Administratorisches seitens Industriepartner

4. Management Abläufe

4.1. Projekt Kostenvoranschlag

Das Projekt startet am 20. September 2010 und dauert 14 Wochen. Abgabetermin ist der 23. Dezember 2010 um 12:00 Uhr. Der Projektumfang beträgt rund 720 Arbeitsstunden. Das entspricht ca. 25.7 Stunden pro Woche und Person.

4.2. Wöchentliche Besprechungen

Besprechung	Teilnehmer	Datum/Zeit	Ort
Mit Betreuer	FK, RS, HH	Mittwochs, 15:00 - 16:00	HSR, Raum 6.010
Mit Industriepartner	FK, RS, KP, RF (optional)	Donnerstags, 17:00 - 18:00	Siemens, ZW07, Raum 405

Bei den Wöchentlichen Besprechungen werden die wichtigsten Fragen und Entscheide jeweils in einem Sitzungsprotokoll festgehalten.

Neben den regulären Besprechungen werden auch zwei gemeinsame Treffen mit dem Betreuer und dem Industriepartner zusammen durchgeführt. Diese finden in den Projektwochen 7 und 11 statt.

4.3. Projektplan

4.3.1. Zeitplan

Siehe separates Dokument: Projektplan_Zeitplanung.pdf

4.3.2. Meilensteine

Meilenstein	Geplant	Erreicht	Artefakte
MS1: Projektplan	04.10.2010	04.10.2010	• Projektplan • Zeitplan
MS2: Anforderungen und Analyse	18.10.2010	18.10.2010	• Anforderungspezifikation (nichtfunktionale Anforderungen, UseCases) • Domain Modell
MS3: Prototyp	01.11.2010	01.11.2010	• Architekturprototyp „proof of concept“ • Architektur Entwurf
MS4: Varianten Analysen	15.11.2010	22.11.2010	• Alpha Release • Varianten Analysen • Webservice Interfaces • Design Entwurf
MS5: Design	29.11.2010	29.11.2010	• Beta Release • Software Architektur und Design Spezifikation • UI Design Studie
MS6: Final	13.12.2010	13.12.2010	• Final Release • Technischer Bericht Entwurf
	16.12.2010	16.12.2010	• Kurzbericht zum Review
MS7: Abgabe	23.12.2010	23.12.2010	• Poster • Testdokument • Technischer Bericht

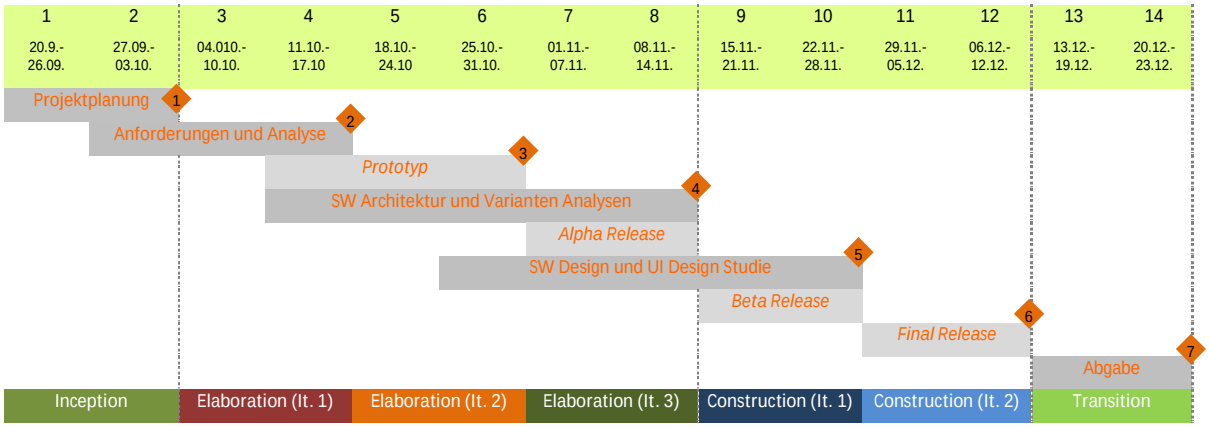
4.3.3. Meilenstein-Verschiebungen, Unerwartetes, Erklärungen

Wir mussten den Meilenstein 4 geplant am 15.11.2010 um eine Woche auf den 22.11.2010 verschieben. Dies hat die folgenden zwei Gründe: Einerseits sind wir etwas in Verzug mit der Implementation der „Browse RA Controllers“ Funktionalitäten. Andererseits wollten wir unbedingt auch schon den Alpha Release auf der echten Hardware testen. Unsere WP7 Smartphones konnten wir aber erst am Donnerstag, 18. November für die Entwicklung freischalten, da unsere Identität zuerst von Microsoft überprüft werden musste. Dies war eine relativ mühsame Angelegenheit, da Studenten-Accounts bei Microsoft nicht automatisch, sondern erst bei der ersten Übermittlung einer App zur Identitätsprüfung gelangen. Somit waren wir also gezwungen, eine Dummy-App einzuschicken, damit der Identitätsvalidierungsprozess angestossen werden konnte. Diese Validierung wird benötigt, um unsere Accounts freizuschalten, was uns wiederum erlaubt, unsere WP7 Phones als Entwicklergeräte freizuschalten.

4.3.4. Iterationsplanung

Jede Iteration dauert zwei Wochen. Aufgrund der vielen analyse- und designlastigen Aufgaben haben wir uns für drei Iterationen in der Elaborationsphase und nur zwei Iterationen in der Konstruktionsphase entschieden. Es werden aber auch bereits während der Elaborationsphase ein „Proof of Concept“ Prototyp erstellt sowie die ersten Grundfunktionen implementiert. Damit sollen möglichst alle Hindernisse so früh als möglich erkannt und die technische Machbarkeit bewiesen werden.

Folgendes Diagramm zeigt die Projektphasen, die jeweiligen Hauptaufgaben sowie die Meilensteine. Die grau markierten Bereiche (Aufgaben) dürfen nur als Schwerpunkte angesehen werden, die Bearbeitung beginnt jeweils schon früher, nur der Endtermin (Meilenstein) ist fix.



4.3.5. Software Releases

Release	Datum	Beschreibung
Prototyp	01.11.2010	Architekturprototyp über alle Layer, „proof of concept“
Alpha	22.11.2010	Grundfunktionalität: Kontrollerliste
Beta	29.11.2010	Grundfunktionalität: Datenpunkttest
Final	13.12.2010	Voller Funktionsumfang (alle Pflicht-Anforderungen) Systemtest erfolgreich durchlaufen

5. Risiko Management

ID	Risiko	Auswirkung	Massnahmen	S	W	G
R01	Ausfall eines Teammitglieds	Ein Teil der definierten Arbeit wird nicht geleistet	Zeitreserve einplanen	360	1%	3.6
R02	Datenverlust	Teil der Arbeit geht verloren	SVN einsetzen mit häufigem Commit	10	10%	1

R03	WP7 hat unbekannte Restriktionen (gegenüber normalem Silverlight)	Funktionen nicht umsetzbar, unschöne Lösungen	Technologiestudium bzw. Prototyp möglichst früh	20	30%	6
R04	Fehler in der Zeitplanung	Meilensteine können nicht gehalten werden	Zeitreserve einplanen, Funktionen streichen	80	40%	32
R05	Notebook als Wireless AP und Gateway technisch nicht umsetzbar	Idee mit Notebook als Gateway und Wireless AP nicht möglich	Allenfalls nach Alternativen suchen (separater Wireless AP?)	10	20%	2
R06	Ausfall Entwicklungslaptop	Laptop auf dem entwickelt wird, fällt aus	- Ausweichen auf Firmen-/HSR-Rechner - Ersatz/Reparatur beantragen	15	5%	0.8
R07	Windows Phone 7 Gerät nicht verfügbar	Keine Tests und Demos mit echter Hardware möglich	Tests und Demo mit Emulator	0	30%	0
R08	Polling mit REST => keine zufrieden-stellenden Ergebnisse oder unbekannte Seiteneffekte	- Zeitverzögerungen in der Anzeige - Stocken bei Navigation - Performance- und/oder Synchronisationsprobleme	- Unschönheiten kaschieren - Andere Technologie (Variantenanalyse)	30	30%	9
R09	Schwierigkeiten bei Simulation der RA Controller (Komplexität)	Tests können nicht realitätsnah durchgeführt werden	Vereinfachte Simulation; Prototyp muss nicht zwingend in realem Umfeld laufen	20	50%	10
R10	Keine Erfahrung mit Entwicklung für Smartphones	Schwierigkeiten bei Gestaltung komplexer GUIs für Smartphones	Zeitreserven für GUI Design einplanen	30	40%	12
R11	Aufgabenstellung unklar	Verzögerungen, Fehlleistungskosten	Möglichst frühe Abklärungen mit entsprechenden Personen	80	20%	16
R12	Verzögerung der Lieferung von Device Discovery API + Simulation	- Verzögerung bei Implementation von Device Discovery - Mehraufwand: Device Discovery selbst simulieren - Nicht genügend realitätsnah	- Status bei A. Schlumberger verfolgen - Device Discovery selbst simulieren (vereinfacht)	60	20%	12
						Reserve: 104.4 h

S: Max. Schaden in Stunden [h]
W: Wahrscheinlichkeit [%]
G: Gewichteter Schaden in Stunden (G=W*S)

Aus der Tabelle wird ersichtlich, dass auf jeden Fall eine Zeitreserve von 100 h (>10% des Projektumfangs) eingeplant werden muss. Da der Endtermin fix ist und unter keinen Umständen verschoben werden kann, müssen Verzögerungen durch Mehrarbeit aufgeholt werden. Falls das Ausmass der Mehrarbeit nicht mehr zumutbar ist, muss optionales weggelassen bzw. gewisse Funktionen gestrichen werden.

6. Zeitschätzungen der Arbeiten

6.1. Projekt Management

ID	Name	Beschreibung	Verant-wortlich	Aufwand [h]	Abhängigkeiten / Risiken
A101	Projektplanung (initial)	Projektplanungs-Dokument verfassen und übergeben.	RS, KF	35	
A102	Infrastruktur Aufbau	SVN-Zugang organisieren und Ablagestruktur definieren, Persönliche Infrastruktur.	RS, KF	17	
A103	Projektplanung (laufend)	Arbeitspakete und Zeitplanung weiterführen/verfeinern	RS	16	A101

6.2. Requirements

ID	Name	Beschreibung	Verant-wortlich	Aufwand [h]	Abhängigkeiten / Risiken
A201	Anforderungen nichtfunktional	Definition aller nichtfunktionalen Anforderungen.	KF	4	
A202	Dokument-erstellung + Use Cases ‚brief'	Dokumenterstellung + Definition aller funktionalen Anforderungen im ‚brief' Format.	KF	10	A201
A203	Use Case Diagramm	UseCase Diagramm erstellen.	RS	3	A202
A204	Use Cases ‚fully dressed'	Für die wichtigsten funktionalen Anforderungen die UseCases im ‚fully dressed' Format erstellen.	RS, KF	20	A202, A203
A205	Abklärungen	Abklärungen zu Anforderungen inklusive User Interface.	RS, KF	18	

6.3. Analyse

ID	Name	Beschreibung	Verant-wortlich	Aufwand [h]	Abhängigkeiten / Risiken
A301	Domain Analyse (OOA)	Domain Modell erarbeiten.	RS, KF	16	A202
A302	VA: Sicherheits-Konzept	Varianten Analyse für ein Sicherheitskonzept.	KF	8 (optional)	
A303	VA: SOA Web-Technologie	Varianten Analyse für SOA Web Technologien in Bezug auf Mobile Devices (REST, SOAP, WCF, Thrift).	RS	20 (optional)	

6.4. Design

ID	Name	Beschreibung	Verant-wortlich	Aufwand [h]	Abhängigkeiten / Risiken
A401	Software Architektur	Beschreiben der Software-Architektur.	RS, KF	40	A2*, A3*

A402	Design (OOD)	Beschreibung des Software-Designs (Package- Klassen- und Sequenzdiagramme).	RS, KF	30	A2*, A3*
A403	Web-Service Interface	Definition eines Web-Service Interfaces für die Inbetriebnahme über ein Mobile-Device.	RS	16	A2*
A404	UI Design Studie für Mobile-Devices	Studie für ein auf die Anforderungen von Mobile-Devices optimiertes User-Interface.	KF	25	

6.5. Implementation

ID	Name	Beschreibung	Verant-wortlich	Aufwand [h]	Abhängigkeiten / Risiken
A501	Architektur-prototyp	Implementation des Architektur-prototypen für ein ‚proof of concept'.	RS, KF	60	A2*, A301, A401, A402
A502	Implementa-tion	Umfasst die komplette Implementation. Wird am Ende der Elaborationsphase 2 genauer aufgeteilt.	RS, KF	163	A2*, A3*, A4*
	Refactoring Architektur-prototyp	Refactoring des Architekturprototypen.	RS	5	A501
	Gateway starten	Gateway Web-Service starten.	RS	4	
	Authentifizierung (Service)	- Login/Authentifizierung auf Gateway - Login/Authentifizierung bei Simulation - Sessionverwaltung	RS	6	
	Gebäude-hierarchie (Service)	- Gebäudehierarchie und RA-Kontroller zur Verfügung stellen - Logischen RA-Kontroller mit seine Hardwarekontroller verbinden	RS	6	
	Hardware-kontroller simulieren	- UI um Hardwarekontroller hinzuzufügen, zu bearbeiten, zu starten und zu stoppen - HW-Kontroller Simulation von A. Schlumberger anpassen.	RS	22	
	Node Setup (Service)	- HW-Kontroller suchen - *'Wink'* zu einem Kontroller senden - *'Service pin'* von einem Kontroller empfangen	RS	10	
	HW-Kontroller Details (Service)	Weiter Attribute von einem Hardwarekontroller zur Verfügung stellen.	RS	4	

	Punkttest (Service)	Web-Service für jeden HW-Kontroller bereitstellen und Funktionen simulieren: - Module und Datenpunkte - Funktionen für den Punkttest - Inbetriebnahme Status und Kommentar setzten	RS	16	
	Login (UI)	Loginbildschirm + Sessionverwaltung		8	
	Gebäude-hierarchie (UI)	Gebäudehierarchie darstellen und browsen ermöglichen.	KF	16	
	Node Setup (UI)	- RA-Kontroller auflisten - *'Wink'* zu einem Kontroller senden - *'Service pin'* von einem Kontroller empfangen	KF	12	
	RA-Kontroller verbinden (UI)	Logischen RA-Kontroller mit einem Hardwarekontroller verbinden. - RA-Kontroller auflisten und auswählen - Auf einem *'service pin'* warten	KF	8	
	HW-Kontroller Details (UI)	Detailansicht eines HW-Kontrollers.	KF	6	
	Punkttest (UI)	Punkttest für einen HW-Kontroller: - Module und Datenpunkte browsen - Punkttest für einen oder mehrere ausgewählte Datenpunkte - Inbetriebnahme Status und Kommentar setzten	KF	35	
A503	Fehlerkorrektur	Korrektur allfälliger Fehler.	RS	6	A601
A504	Codedokument ation	Überarbeiten der Code-Kommentare und generieren der Dokumentation.	RS	5	A5*

6.6. Test

ID	Name	Beschreibung	Verant-wortlich	Aufwand [h]	Abhängigkeiten / Risiken
A601	Systemtest	Beinhaltet den manuellen, auf die Auslieferung ausgerichteten Test der gesamten Applikation vor jedem Release.	RS	6	A2*, A5*, A602 Falls Fehler vorhanden Mehraufwand bei A503.
A602	Testdokumen-tation	Erstellen der Testdokumentation.	RS	14	A2*, A5*

6.7. Dokumentation

ID	Name	Beschreibung	Verant-wortlich	Aufwand [h]	Abhängigkeiten / Risiken
A701	Kurzbericht	Verfassen und online Abgabe des Kurzberichts.	KF	10	

A702	Technischer Bericht	Verfassen des Technischen Berichts.	RS, KF	40	
A703	Management Summary	Management Summary erstellen	KF	12	
A704	Poster	Erstellen des Posters.	RS	10	
A705	Dokument Abschluss	Abschluss und Bereinigung aller Dokumente.	KF	16	
A706	Schlussbericht	Schreiben des persönlichen Schlussberichtes	RS, KF	6	

6.8. Technologie Studien (über ganze Projektdauer)

ID	Name	Beschreibung	Verant-wortlich	Aufwand [h]	Abhängigkeiten / Risiken
A011	Einarbeiten in WP7	Wissen zur Entwicklung für das Windows Phone 7 aneignen.	RS, KF	20	
A012	Einarbeiten in Mobile UI Design	Wissen über UI-Design für Mobile-Devices aneignen.	RS, KF	10	

6.9. Sitzungen (über ganze Projektdauer)

ID	Name	Beschreibung	Verant-wortlich	Aufwand [h]	Abhängigkeiten / Risiken
A021	Teamsitzungen	Teamsitzungen zur Orientierung über den aktuellen Projektstand.	KF	26	
A022	Reviews	Reviews zur Überprüfung der Meilensteine.	KF	37	

6.10. Qualitätssicherung (über ganze Projektdauer)

ID	Name	Beschreibung	Verant-wortlich	Aufwand [h]	Abhängigkeiten / Risiken
A031	Code Review	Gegenseitiges Code-Review	RS	8	
A032	Reserve	Reserve 100h	RS	100	

7. Infrastruktur

7.1. Räumlichkeiten

Ein Tag pro Woche werden die zugeteilten Arbeitsplätze für Bachelorarbeiten an der HSR genutzt. Weitere zwei Tage pro Woche werden die Arbeitsplätze beim Industriepartner genutzt. Die restliche Arbeit wird zu Hause erledigt.

7.2. Hardware

- Private Notebooks der einzelnen Teammitglieder
- Rechner in den Computerräumen der HSR
- SVN-Server der HSR (svns.hsr.ch)
- Private Drucker

- Drucker und Kopierer der HSR

7.3. Software

7.3.1. Entwicklungstools

- Microsoft Visual Studio 2010 (Entwicklungsumgebung)
- Microsoft Expression Blend 4 (GUI Design)
- NUnit (Unit-Tests)
- Notepad2 (XML-Dateien)
- FxCop (Überprüfung auf Microsoft Coding-Guidelines)
- GhostDoc (Unterstützung für die Kommentare in .Net)
- MsBuild (Build-Automatisation)
- Adobe Photoshop CS4 (Icons/Grafiken)

7.3.2. Dokumentation

- Microsoft Word 2010 (Dokumentation)
- Microsoft Excel 2010 (Planung, Dokumentation)
- Adobe Acrobat 9 Professional (PDF-Generation)
- Enterprise Architekt 7.0 (UML-Modeling)
- Doc-O-Matic (Quellcode-Dokumentation)

7.3.3. Versionsverwaltung

- Subversion (SVN-Server der HSR)
- TortoiseSVN als Client

7.3.4. Sonstiges

- Skype (Telefonkonferenzen über Internet)
- Windows Live Messenger

8. Qualitätsmassnahmen

8.1. Teamarbeit und Kommunikation

Da das Team nur aus zwei Personen besteht, ist die Koordination relativ einfach. Durch eine klare, lösungsorientierte Kommunikation untereinander, wollen wir die Teamarbeit so effizient wie möglich gestalten. Alle wichtigen Entscheide werden im Team besprochen und gemeinsam gefällt. Ebenso werden Probleme frühzeitig im Team besprochen und falls nötig, gemeinsam nach Lösungen gesucht.

8.2. Dokumentation und Planung

Die Qualität der Dokumentation bestimmt massgeblich den Erfolg des Projektes. Daher werden alle Dokumente laufend oder zumindest vor jeder Teamsitzung aktualisiert, wobei die (Zeit-)Planung ein besonderes Augenmerk verlangt. Um das Zeitmanagement im Griff zu haben und Verzögerungen frühzeitig zu erkennen, ist eine gute Zeitplanung essentiell wichtig. Die Teammitglieder tragen die aufgewendeten Stunden laufend in den Zeitplan ein.

Da die Verständlichkeit und damit auch die Qualität des Quellcodes nicht zuletzt auch durch die Qualität der Quelltextkommentare mitbestimmt wird, werden alle Klassen sowie mindestens alle Public-Funktionen und Properties mit Quelltextkommentaren versehen. Quelltextkommentare werden in Englischer Sprache geschrieben. Als Hilfsmittel wird GhostDoc eingesetzt.

8.3. Reviews

Die Reviews tragen wesentlich zur Qualitätssicherung bei. Darum werden regelmässig Code- und Dokumentreviews mit dem Betreuer und/oder dem Industriepartner durchgeführt. Review-Kommentare werden als Notizen im Besprechungsprotokoll festgehalten.

Detailplan

Mobile Commissioning Tool for Room Automation

Ferien

			Woche 0				Woche 1				Woche 2			
Phasen			vor Projektstart				Inception							
Meilensteine							20.09				27.09			
Woche (von - bis)			--		--		20.09		26.09		27.09		03.10	
	Total		KF		RS		KF		RS		KF		RS	
Task	SOLL	IST	S	I	S	I	S	I	S	I	S	I	S	I
Projekt Managment	68.0	80.5	0.0	0.0	0.0	0.0	20.0	20.0	17.0	17.0	6.0	4.5	8.0	13.0
A101: Projektplanung (initial)	35.0	40.5					18.0	18.5	12.0	12.5	2.0	2.0	3.0	7.5
A102: Infrastruktur	17.0	27.5					2.0	1.5	5.0	4.5	3.0	1.5	3.0	3.0
A103: Projektplanung (laufend)	16.0	12.5									1.0	1.0	2.0	2.5
Requirements	55.0	64.5	0.0	0.0	0.0	0.0	0.0	0.0	2.0	2.0	17.0	14.0	15.0	10.0
A201: Nichtfunktionale Anforderungen	4.0	6.0									3.0		1.0	
A202: Dokumentenerstellung und Use Cases 'brief'	10.0	16.0							2.0	2.0	4.0	8.0	4.0	3.0
A203: Use Case Diagramm	3.0	2.0											2.0	1.5
A204: Use Cases 'fully dressed'	20.0	25.5									5.0		3.0	
A205: Abklärungen (Anforderungen und UI)	18.0	15.0									5.0	6.0	5.0	5.5
Analyse	24.0	24.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A301: Domain Analyse (OOA)	16.0	22.5												
A302: VA: Sicherheitskonzept	8.0	2.0												
A303: VA: SOA Web-Techn. (optional)	0.0	0.0												
Design	111.0	65.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A401: Software Architektur	40.0	25.5												
A402: Design (OOD)	30.0	13.0												
A403: Web-Service Interface Design	16.0	16.5												
A404: UI Design Studie für Mobile-Devices	25.0	10.0												
Implementation	229.0	299.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A501: Architekturprototyp	60.0	66.5												
A502: Implementation, wird am Ende der Elaborationphase 2 aufgeteilt.	0.0	0.0												
A502: Impl: Refactoring Architekturprototyp	5.0	6.5												
A502: Impl: Gateway starten	4.0	6.0												
A502: Impl: Authentifizierung (Service)	6.0	9.5												
A502: Impl: Gebäudehierarchie (Service)	6.0	10.0												
A502: Impl: Hardwarekontroller simulieren	22.0	25.0												
A502: Impl: Node Setup (Service)	10.0	22.5												
A502: Impl: HW-Kontroller Details (Service)	4.0	3.5												
A502: Impl: Punkttest (Service)	16.0	19.3												
A502: Impl: Authentifizierung (UI)	8.0	15.0												
A502: Impl: Gebäudehierarchie (UI)	16.0	32.0												
A502: Impl: Node Setup (UI)	12.0	16.5												
A502: Impl: RA-Kontroller verbinden (UI)	8.0	2.0												
A502: Impl: HW-Kontroller Details (UI)	6.0	5.0												
A502: Impl: Punkttest (UI)	35.0	56.0												
A503: Fehlerkorrektur	6.0	1.5												
A504: Codedokumentation	5.0	2.5												
Test	20.0	14.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A601: System Test	6.0	3.0												
A602: Testdokument	14.0	11.0												
Dokumentation	94.0	202.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A701: Kurzbericht	10.0	0.0												
A702: Technischer Bericht	40.0	147.0												
A703: Management Summary	12.0	19.0												
A704: Poster	10.0	13.0												
A705: Dokument Abschluss	16.0	18.5												
A706: Schlussbericht	6.0	5.0												
Tasks wiederkehrend														
Studium Technologien	30.0	28.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A011: Einarbeiten in WP7 Entwicklung	20.0	20.5												
A012: Einarbeiten in Mobile UI Design Konzepte	10.0	8.0												
Sitzungen	63.0	60.0	1.0	1.5	1.0	1.5	0.0	0.0	0.0	0.0	2.5	2.5	2.5	2.5
A021: Teamsitzungen HSR	26.0	17.0									1.0	1.0	1.0	1.0
A022: Teamsitzungen Siemens (Reviews)	37.0	43.0	1.0	1.5	1.0	1.5					1.5	1.5	1.5	1.5
_	0.0	0.0												
Qualitätssicherung	108.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A031: Code Review	8.0	4.0												
A032: Reserve (100h)	100.0	0.0												
Wochen Subtotal			**1.0**	**1.5**	**1.0**	**1.5**	**20.0**	**20.0**	**19.0**	**19.0**	**25.5**	**21.0**	**25.5**	**25.5**
Wochentotal (inkl. Reserve)	**802.0**	**842.8**			**2.0**	**3.0**			**39.0**	**39.0**			**51.0**	**46.5**

	Woche 3				Woche 4				Woche 5				Woche 6			
Phasen	Elaboration (1. Iteration)								Elaboration (2. Iteration)							
Meilensteine	04.10								18.10							
Woche (von - bis)	04.10		10.10		11.10		17.10		18.10		24.10		25.10		31.10	
	KF		RS		KF		RS		KF		RS		KF		RS	
Task	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I
Projekt Managment	2.0	0.0	2.0	0.0	1.0	0.0	2.0	0.0	0.0	0.5	0.0	1.0	1.0	2.0	2.0	2.0
A101: Projektplanung (initial)																
A102: Infrastruktur	2.0		2.0													
A103: Projektplanung (laufend)					1.0		2.0			0.5		1.0	1.0	2.0	2.0	2.0
Requirements	4.0	7.5	8.0	9.5	3.0	11.0	2.0	8.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	1.5
A201: Nichtfunktionale Anforderungen		3.0				3.0										
A202: Dokumentenerstellung und Use Cases 'brief'						2.0								1.0		
A203: Use Case Diagramm				0.5			1.0									
A204: Use Cases 'fully dressed'	3.0	3.5	7.0	8.0	2.0	6.0		8.0								
A205: Abklärungen (Anforderungen und UI)	1.0	1.0	1.0	1.0	1.0		1.0		1.0		1.0		1.0		1.0	1.5
Analyse	6.0	1.0	6.0	8.5	2.0	1.5	2.0	10.5	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0
A301: Domain Analyse (OOA)	6.0	1.0	6.0	8.5	2.0	1.5	2.0	10.5						1.0		
A302: VA: Sicherheitskonzept														2.0		
A303: VA: SOA Web-Techn. (optional)																
Design	0.0	0.0	0.0	0.0	7.0	0.0	5.0	1.5	1.0	0.0	2.0	1.0	10.0	11.0	11.0	7.0
A401: Software Architektur					7.0		5.0	1.5					7.0	10.0	7.0	2.5
A402: Design (OOD)																
A403: Web-Service Interface Design									1.0		2.0	1.0	1.0	1.0	4.0	4.5
A404: UI Design Studie für Mobile-Devices													2.0			
Implementation	0.0	0.0	0.0	0.0	4.0	4.5	0.0	0.0	16.0	20.0	20.0	18.0	10.0	9.0	10.0	15.0
A501: Architekturprototyp					4.0	4.5	0.0		16.0	20.0	20.0	18.0	10.0	9.0	10.0	15.0
A502: Implementation, wird am Ende der Elaborationphase 2 aufgeteilt.																
A502: Impl: Refactoring Architekturprototyp																
A502: Impl: Gateway starten																
A502: Impl: Authentifizierung (Service)																
A502: Impl: Gebäudehierarchie (Service)																
A502: Impl: Hardwarekontroller simulieren																
A502: Impl: Node Setup (Service)																
A502: Impl: HW-Kontroller Details (Service)																
A502: Impl: Punkttest (Service)																
A502: Impl: Authentifizierung (UI)																
A502: Impl: Gebäudehierarchie (UI)																
A502: Impl: Node Setup (UI)																
A502: Impl: RA-Kontroller verbinden (UI)																
A502: Impl: HW-Kontroller Details (UI)																
A502: Impl: Punkttest (UI)																
A503: Fehlerkorrektur																
A504: Codedokumentation																
Test	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A601: System Test																
A602: Testdokument																
Dokumentation	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A701: Kurzbericht																
A702: Technischer Bericht																
A703: Management Summary																
A704: Poster																
A705: Dokument Abschluss																
A706: Schlussbericht																
Studium Technologien	4.0	0.0	5.0	0.0	6.0	5.0	8.0	2.5	5.0	3.5	2.0	6.5	0.0	0.0	0.0	3.0
A011: Einarbeiten in WP7 Entwicklung	2.0		4.0		4.0	5.0	4.0	2.5	4.0	3.5	2.0	6.5				3.0
A012: Einarbeiten in Mobile UI Design Konzepte	2.0		1.0		2.0		4.0		1.0							
Sitzungen	2.5	2.0	2.5	2.0	2.5	3.5	2.5	3.5	2.5	1.0	2.5	1.0	2.5	2.5	2.5	2.5
A021: Teamsitzungen HSR	1.0		1.0		1.0	1.0	1.0	1.0	1.0		1.0		1.0	1.5	1.0	1.5
A022: Teamsitzungen Siemens (Reviews)	1.5	2.0	1.5	2.0	1.5	2.5	1.5	2.5	1.5	1.0	1.5	1.0	1.5	1.0	1.5	1.0
_																
Qualitätssicherung	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0
A031: Code Review													1.0		1.0	
A032: Reserve (100h)																
Wochen Subtotal	**18.5**	**10.5**	**23.5**	**20.0**	**25.5**	**25.5**	**21.5**	**26.0**	**25.5**	**25.0**	**27.5**	**27.5**	**25.5**	**28.5**	**27.5**	**31.0**
Wochentotal (inkl. Reserve)			**42.0**	**30.5**			**47.0**	**51.5**			**53.0**	**52.5**			**53.0**	**59.5**

	Woche 7				Woche 8				Woche 9				Woche 10			
Phasen	Elaboration (3. Iteration)								Construction (1. Iteration)							
Meilensteine	01.11								15.11							
Woche (von - bis)	01.11		07.11		08.11		14.11		15.11		21.11		22.11		28.11	
	KF		RS		KF		RS		KF		RS		KF		RS	
Task	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I
Projekt Managment	0.0	1.0	0.0	2.0	1.0	5.0	3.0	5.5	0.0	0.0	0.0	0.0	1.0	1.5	1.0	4.0
A101: Projektplanung (initial)																
A102: Infrastruktur		1.0		1.0		5.0		5.5								3.0
A103: Projektplanung (laufend)				1.0	1.0		3.0						1.0	1.5	1.0	1.0
Requirements	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A201: Nichtfunktionale Anforderungen																
A202: Dokumentenerstellung und Use Cases 'brief'																
A203: Use Case Diagramm																
A204: Use Cases 'fully dressed'																
A205: Abklärungen (Anforderungen und UI)																
Analyse	3.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A301: Domain Analyse (OOA)																
A302: VA: Sicherheitskonzept	3.0				5.0											
A303: VA: SOA Web-Techn. (optional)																
Design	11.0	10.0	10.0	4.0	8.0	3.0	9.0	4.5	7.0	5.0	5.0	9.5	11.0	1.0	6.0	0.0
A401: Software Architektur	3.0	4.0	3.0	1.0	2.0		2.0	2.5			2.0	4.0	2.0			
A402: Design (OOD)	1.0		2.0		1.0		2.0		3.0	3.5	3.0	3.5	4.0		6.0	
A403: Web-Service Interface Design			4.0	3.0			4.0	2.0		1.5		2.0				
A404: UI Design Studie für Mobile-Devices	7.0	6.0	1.0		5.0	3.0	1.0		4.0				5.0	1.0		
Implementation	16.0	17.5	19.0	27.0	14.0	19.0	13.0	20.0	20.0	24.0	16.0	15.3	11.0	30.0	12.0	26.5
A501: Architekturprototyp																
A502: Implementation, wird am Ende der Elaborationphase 2 aufgeteilt.																
A502: Impl: Refactoring Architekturprototyp			5.0	6.5												
A502: Impl: Gateway starten			2.0				2.0	6.0								
A502: Impl: Authentifizierung (Service)			5.0	9.5			1.0									
A502: Impl: Gebäudehierarchie (Service)			4.0	7.0			2.0	2.0								1.0
A502: Impl: Hardwarekontroller simulieren			3.0	2.5			8.0	5.0							9.0	16.5
A502: Impl: Node Setup (Service)				1.5				7.0			8.0	8.0		3.0	1.0	3.0
A502: Impl: HW-Kontroller Details (Service)															2.0	2.0
A502: Impl: Punkttest (Service)											8.0	7.3				4.0
A502: Impl: Authentifizierung (UI)	4.0				4.0	13.0				2.0						
A502: Impl: Gebäudehierarchie (UI)	12.0	17.5			4.0					7.0				2.5		
A502: Impl: Node Setup (UI)					6.0	6.0			6.0	8.0				2.0		
A502: Impl: RA-Kontroller verbinden (UI)																
A502: Impl: HW-Kontroller Details (UI)									4.0	5.0			2.0			
A502: Impl: Punkttest (UI)									10.0	2.0			9.0	22.5		
A503: Fehlerkorrektur																
A504: Codedokumentation																
Test	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	2.0	7.5	0.0	0.0	5.0	0.0
A601: System Test											2.0	2.0			2.0	
A602: Testdokument							3.0					5.5			3.0	
Dokumentation	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	3.0	0.0
A701: Kurzbericht																
A702: Technischer Bericht													3.0		3.0	
A703: Management Summary																
A704: Poster																
A705: Dokument Abschluss																
A706: Schlussbericht																
Studium Technologien	0.0	3.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0
A011: Einarbeiten in WP7 Entwicklung																
A012: Einarbeiten in Mobile UI Design Konzepte		3.0				2.0				3.0						
Sitzungen	2.5	1.0	2.5	1.0	2.5	4.0	2.5	4.0	2.5	1.5	2.5	1.5	2.5	4.5	2.5	4.5
A021: Teamsitzungen HSR	1.0		1.0		1.0	1.0	1.0	1.0	1.0		1.0		1.0	2.0	1.0	2.0
A022: Teamsitzungen Siemens (Reviews)	1.5	1.0	1.5	1.0	1.5	3.0	1.5	3.0	1.5	1.5	1.5	1.5	1.5	2.5	1.5	2.5
_																
Qualitätssicherung	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0
A031: Code Review		1.0		1.0	1.0		1.0						1.0		1.0	
A032: Reserve (100h)																
Wochen Subtotal	**32.5**	**33.5**	**31.5**	**35.0**	**31.5**	**33.0**	**31.5**	**34.0**	**29.5**	**33.5**	**25.5**	**33.8**	**29.5**	**37.0**	**30.5**	**35.0**
Wochentotal (inkl. Reserve)			**64.0**	**68.5**			**63.0**	**67.0**			**55.0**	**67.3**			**60.0**	**72.0**

	Woche 11				Woche 12				Woche 13				Woche 14			
Phasen	Construction (2. Iteration)								Transition							
Meilensteine	29.11								13.12							23.12
Woche (von - bis)	29.11		05.12		06.12		12.12		13.12		19.12		20.12		26.12	
	KF		RS		KF		RS		KF		RS		KF		RS	
Task	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I
Projekt Managment	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.5	0.0	0.0	0.0	0.0
A101: Projektplanung (initial)																
A102: Infrastruktur												1.5				
A103: Projektplanung (laufend)							1.0									
Requirements	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A201: Nichtfunktionale Anforderungen																
A202: Dokumentenerstellung und Use Cases 'brief'																
A203: Use Case Diagramm																
A204: Use Cases 'fully dressed'																
A205: Abklärungen (Anforderungen und UI)																
Analyse	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A301: Domain Analyse (OOA)																
A302: VA: Sicherheitskonzept																
A303: VA: SOA Web-Techn. (optional)																
Design	2.0	0.0	2.0	1.5	2.0	0.0	2.0	6.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A401: Software Architektur																
A402: Design (OOD)	2.0		2.0		2.0		2.0	6.0								
A403: Web-Service Interface Design				1.5												
A404: UI Design Studie für Mobile-Devices																
Implementation	19.5	21.5	14.5	10.5	7.5	15.5	2.0	3.5	2.0	2.5	2.5	0.0	0.0	0.0	0.0	0.0
A501: Architekturprototyp																
A502: Implementation, wird am Ende der Elaborationphase 2 aufgeteilt.																
A502: Impl: Refactoring Architekturprototyp																
A502: Impl: Gateway starten																
A502: Impl: Authentifizierung (Service)																
A502: Impl: Gebäudehierarchie (Service)																
A502: Impl: Hardwarekontroller simulieren			2.0	1.0												
A502: Impl: Node Setup (Service)			1.0													
A502: Impl: HW-Kontroller Details (Service)			2.0	1.5												
A502: Impl: Punkttest (Service)			8.0	8.0												
A502: Impl: Authentifizierung (UI)																
A502: Impl: Gebäudehierarchie (UI)		5.0														
A502: Impl: Node Setup (UI)		0.5														
A502: Impl: RA-Kontroller verbinden (UI)	8.0					2.0										
A502: Impl: HW-Kontroller Details (UI)																
A502: Impl: Punkttest (UI)	10.0	16.0			6.0	12.0		3.5								
A503: Fehlerkorrektur	1.0		1.0			1.5			2.0		2.0					
A504: Codedokumentation	0.5		0.5		1.5		2.0			2.5	0.5					
Test	0.0	0.0	4.0	4.0	2.0	0.0	2.0	0.0	0.0	0.0	2.0	2.5	0.0	0.0	0.0	0.0
A601: System Test				1.0	2.0											
A602: Testdokument			4.0	3.0			2.0				2.0	2.5				
Dokumentation	4.0	11.0	5.0	11.0	12.5	17.0	13.5	23.0	20.0	36.0	18.0	34.5	7.5	34.0	7.5	36.0
A701: Kurzbericht					2.5		2.5		2.0		2.0		0.5		0.5	
A702: Technischer Bericht	4.0	11.0	5.0	11.0	6.0	17.0	7.0	19.0	7.0	23.5	5.0	25.0		19.5		21.0
A703: Management Summary					3.0		3.0	4.0	3.0	8.5	3.0	6.5				
A704: Poster					1.0		1.0		4.0		4.0	3.0		5.0		5.0
A705: Dokument Abschluss									4.0	4.0	4.0		4.0	8.0	4.0	6.5
A706: Schlussbericht													3.0	1.5	3.0	3.5
Studium Technologien	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A011: Einarbeiten in WP7 Entwicklung																
A012: Einarbeiten in Mobile UI Design Konzepte																
Sitzungen	2.5	1.0	2.5	1.0	2.5	2.5	2.5	2.5	2.0	1.5	2.0	1.5	1.0	1.0	1.0	1.0
A021: Teamsitzungen HSR	1.0	1.0	1.0	1.0	1.0		1.0		1.0		1.0		1.0	1.0	1.0	1.0
A022: Teamsitzungen Siemens (Reviews)	1.5		1.5		1.5	2.5	1.5	2.5	1.0	1.5	1.0	1.5				
_																
Qualitätssicherung	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A031: Code Review					1.0	1.0	1.0	1.0								
A032: Reserve (100h)																
Wochen Subtotal	**28.0**	**33.5**	**28.0**	**28.0**	**27.5**	**36.0**	**24.0**	**36.0**	**24.0**	**40.0**	**24.5**	**40.0**	**8.5**	**35.0**	**8.5**	**37.0**
Wochentotal (inkl. Reserve)			**56.0**	**61.5**			**51.5**	**72.0**			**48.5**	**80.0**			**17.0**	**72.0**

Auswertung:	**Gesammt**		**KF**		**RS**		**Diffenenz (Ist-Soll)**			**Diffenenz (Akkumuliert)**		
	SOLL	**IST**	**SOLL**	**IST**	**SOLL**	**IST**	**KF**	**RS**	**Total**	**KF**	**RS**	**Total**
vor Projektstart	2.0	3.0	1.0	1.5	1.0	1.5	0.5	0.5	1.0	0.5	0.5	1.0
Inception	90.0	85.5	45.5	41.0	44.5	44.5	-4.5	0.0	-4.5	-4.0	0.5	-3.5
Elaboration (1. Iteration)	89.0	82.0	44.0	36.0	45.0	46.0	-8.0	1.0	-7.0	-12.0	1.5	-10.5
Elaboration (2. Iteration)	106.0	112.0	51.0	53.5	55.0	58.5	2.5	3.5	6.0	-9.5	5.0	-4.5
Elaboration (3. Iteration)	127.0	135.5	64.0	66.5	63.0	69.0	2.5	6.0	8.5	-7.0	11.0	4.0
Construction (1. Iteration)	115.0	139.3	59.0	70.5	56.0	68.8	11.5	12.8	24.3	4.5	23.8	28.3
Construction (2. Iteration)	107.5	133.5	55.5	69.5	52.0	64.0	14.0	12.0	26.0	18.5	35.8	54.3
Transition	65.5	152.0	32.5	75.0	33.0	77.0	42.5	44.0	86.5	61.0	79.8	140.8
Total	702.0	842.8	352.5	413.5	349.5	429.3	61.0	79.8	140.8			

Kontrolle

			SOLL	**IST**
Total SOLL	702.0	**Total KF**	352.5	413.5
Total IST	842.8	**Total RS**	349.5	429.3

Balkendiagram — Mobile Commissioning Tool for Room Automation

Phasen			vor Projektstart	Inception		Elaboration (1. Iteration)		Elaboration (2. Iteration)		Elaboration (3. Iteration)		Construction (1. Iteration)		Construction (2. Iteration)		Transition	
Task	**Total SOLL**	**IST**	**Woche 0**	**Woche 1**	**Woche 2**	**Woche 3**	**Woche 4**	**Woche 5**	**Woche 6**	**Woche 7**	**Woche 8**	**Woche 9**	**Woche 10**	**Woche 11**	**Woche 12**	**Woche 13**	**Woche 14**
Projekt Managment	68.0			37	14	4	3		3		4		2		1		
		80.5		37	17.5			1.5	4	3	10.5		5.5			1.5	
Requirements	55.0			2	32	12	5	2	2								
		64.5		2	24	17	19		2.5								
Analyse	24.0					12	4			3	5						
		24.5				9.5	12		3								
Design	111.0						12	3	21	21	17	12	17	4	4		
		65.0					1.5	1	18	14	7.5	14.5	1	1.5	6		
Implementation	229.0						4	36	20	35	27	36	23	34	9.5	4.5	
		299.3					4.5	38	24	44.5	39	39.3	56.5	32	19	2.5	
Test	20.0										3	2	5	4	4	2	
		14.0										7.5		4		2.5	
Dokumentation	94.0												6	9	26	38	15
		202.5												22	40		
Studium Technologien	30.0					9	14	7									
		28.5					7.5	10	3	3	2	3					
Sitzungen	63.0		2		5	5	5	5	5	5	5	5	5	5	5	4	2
		60.0	3		5	4	7	2	5	2	8	3	9	2	5	3	2
Qualitätssicherung	108.0								2		2		2		2		
		4.0								2					2		
Total	**802.0**	**842.8**															

Legende

SOLL:

< 8 Stunden	0.001	7.999
zwischen 8 und 16 Stunden	8	16
> 16 Stunden	16.001	64

IST:

< 8 Stunden	0.001	7.999
zwischen 8 und 16 Stunden	8	16
> 16 Stunden	16.001	64

8.4. Backup/Sicherung

Das konstante einchecken des Quellcodes sowie allen anderen Dokumenten in der SVN-Versionsverwaltung garantiert, dass alle Daten jederzeit und in allen eingecheckten Versionen wiederherstellbar sind.

8.5. Tests

Da es in diesem Projekt hauptsächlich um das Userinterface geht und fast keine Businesslogik existiert, wird mehr Wert auf einen ausführlichen Systemtest als auf Unittests gelegt. Bei jedem Release wird ein vollständiger manueller Systemtest durchgeführt. Dafür wird im Voraus eine Testspezifikation anhand der UseCases bzw. Anforderungsspezifikation erstellt.

8.6. Programmierrichtlinien

Für die Entwicklung mit C#. halten wir uns an die Siemens-Programmierrichtlinien, die weitgehend mit den von Microsoft veröffentlichten *.NET Naming Guidelines* (http://msdn.microsoft.com/en-us/library/ms229002.aspx) übereinstimmen. Es ist wichtig, dass der Programmierstil von allen Teammitgliedern in etwa gleich ist, damit die Verständlichkeit des ganzen Codes gewahrt bleibt. Daher wird auch die Namensgebung strikte in Englisch gehalten.

Zeiterfassung

Projekt: **Mobile Commissioning Tool for Room Automation**
Teammitglied: **Kaspar Fenner**

Durchschnittliche Soll-Arbeitszeit: **25.7**

Woche	Datum	Tätigkeit	Zeit	Zeit/Wo	Total	Soll
Woche 1	16.09.2010	Kick-off Meeting	1.5			
	21.09.2010	Infrastruktur: WP7 Developer Account	0.5			
	23.09.2010	Infrastruktur: SVN, Laptop	1.0			
	23.09.2010	Besprechung mit Reto: Tasks, Iterationen + Meilensteine festlegen	4.0			
	23.09.2010	Projektplan erarbeiten	3.0			
	24.09.2010	Projektplan erarbeiten	8.0			
	25.09.2010	Projektplan erarbeiten	3.5			
				21.5	**21.5**	25.7
Woche 2	27.09.2010	Projektplan review mit Reto	1.0			
	29.09.2010	Projektplan Risikoanalyse überarbeitet	1.0			
	29.09.2010	Weekly Meeting mit H. Huser	1.0			
	29.09.2010	Sitzungsprotokoll Template erstellt	1.0			
	30.09.2010	Infrastruktur: WP7 Developer Tools installiert	1.5			
	30.09.2010	Abklärungen zu Requirements, GUI, Systemdesign	2.0			
	30.09.2010	GUI Design Sketches	2.0			
	30.09.2010	Requirements erfassen "brief" + Layout	1.0			
	01.10.2010	Requirements document erarbeiten	3.0			
	01.10.2010	GUI Design Sketches Besprechung	1.0			
	01.10.2010	Meeting mit Dani Wyss "Diskussion Node Setup + DPT für Smartphone	2.5			
	03.10.2010	Requirements document erarbeiten	4.0			
				21.0	**42.5**	51.4
Woche 3	06.10.2010	Non-functional requirements	3.0			
	07.10.2010	Besprechung mit Reto: Domainmodell, Requirements	2.0			
	07.10.2010	Besprechung mit Roger Meier	1.0			
	07.10.2010	Siemens-Dokumentenstudie "Controller Web Architecture" + "User Stc	1.0			
	08.10.2010	Requirements / Use Cases "fully dressed"	2.0			
	10.10.2010	Requirements / Use Cases "fully dressed"	1.5			
				10.5	**53.0**	77.1
Woche 4	12.10.2010	Review Use Cases + Domain Modell mit H. Meier und T. Gollackner	1.5			
	13.10.2010	Requirements / Use Cases "fully dressed"	2.0			
	13.10.2010	Meeting mit H. Huser: Use Cases und Arbeitspackete	1.0			
	14.10.2010	Requirements Chapter 2+3 updated	5.0			
	14.10.2010	WP7 + REST Technologiestudium	1.0			
	15.10.2010	Requirements: Document updated, Use Cases "fully dressed"	4.0			
	15.10.2010	Besprechung mit Reto: Requirements, Domainmodell	1.5			
	15.10.2010	Review Use Cases + Domain Modell mit K. Panteleeva	1.0			
	17.10.2010	WP7 + REST Technologiestudium / Architekturprototyp	8.5			
				25.5	**78.5**	102.9

Zeiterfassung

Projekt: **Mobile Commissioning Tool for Room Automation**
Teammitglied: **Kaspar Fenner**

Durchschnittliche Soll-Arbeitszeit: **25.7**

Woche	Datum	Tätigkeit	Zeit	Zeit/Wo	Total	Soll
Woche 5	18.10.2010	Review/Einführung Architekturprototyp mit Reto	1.0			
	20.10.2010	Architekturprototyp	6.0			
	21.10.2010	Architekturprototyp	7.0			
	22.10.2010	Besprechung mit Reto: Architektur / Technologien / Zeitplanung	2.0			
	22.10.2010	Architekturprototyp	4.0			
	24.10.2010	Netzwerktesting (Windows Networkbridge / Phone / Controller-Simulation)	5.0			
				25.0	**103.5**	128.6
Woche 6	25.10.2010	Besprechung der Architektur mit K. Panteleeva	1.0			
	26.10.2010	Physical Architecture Diagramm anpassen	0.5			
	27.10.2010	Architekturdokument erarbeiten	1.0			
	27.10.2010	Architekturdiskussion mit Reto	1.0			
	27.10.2010	Device Discovery API verstehen/testen/integrieren	4.0			
	28.10.2010	Device Discovery Konzept erarbeiten, API austesten, besprechen mit Reto + A. Schlumberger	4.0			
	28.10.2010	Review von REST Serviceinfrastruktur mit Reto	1.0			
	28.10.2010	Dokumente aktualisiert nach Review-Comments (Use Cases, Analyse)	2.0			
	29.10.2010	Architekturdokument und Diagramme erarbeiten	5.0			
	29.10.2010	Meeting mit H. Huser in der Siemens	1.5			
	31.10.2010	Projektplanung aktualisiert	2.0			
	31.10.2010	Architekturdokument Kapitel 2 + 3	5.5			
				28.5	**132.0**	154.3
Woche 7	01.11.2010	Architekturdokument Kapitel 2 + 3	4.0			
	01.11.2010	WP7 UI: WP7Guide ausprobiert, TreeView Problem analysiert	3.0			
	03.11.2010	WP7 UI: Building Hierarchy view (nur visuell ohne Funktion)	7.0			
	04.11.2010	WP7 UI Design Guidelines + XAML verstehen + TreeView Lösungen für WP7	3.0			
	04.11.2010	WP7 UI: Building Hierarchy view (rekursive Struktur)	6.0			
	05.11.2010	WP7 Phone Hardwarebeschaffung	1.0			
	05.11.2010	Codereview mit Reto	1.0			
	05.11.2010	Besprechung mit Reto	1.0			
	05.11.2010	WP7 UI: Building Hierarchy View	4.5			
	07.11.2010	WP7 UI Konzepte: Page Navigation, Application Bar, Status Handling + Prism	3.0			
				33.5	**165.5**	180.0

Zeiterfassung

Projekt: **Mobile Commissioning Tool for Room Automation**
Teammitglied: **Kaspar Fenner**

Durchschnittliche Soll-Arbeitszeit: **25.7**

Woche	Datum	Tätigkeit	Zeit	Zeit/Wo	Total	Soll
Woche 8	08.11.2010	WP7 UI Konzepte: Page Navigation, Application Bar, Status Handling + Prism	2.0			
	10.11.2010	WP7 UI: Prism, Login Popup, ViewModelBase, ViewModelLocator	7.0			
	10.11.2010	Inbetriebnahme von Windows Phone Hardware Device	1.0			
	10.11.2010	Architekturbesprechung und Codereview mit K. Panteleeva	1.0			
	10.11.2010	PC und Phone für Development einrichten und Dev Account freischalten	2.0			
	11.11.2010	Beispiel App für App Hub Identitiy Validation submitted	1.0			
	11.11.2010	WP7: Login Dialog	3.0			
	12.11.2010	Meeting mit H. Huser	1.0			
	12.11.2010	Page Handling definiert mit Aufruf von Login Popup und Fehlerverhalten	6.0			
	12.11.2010	Besprechung mit Reto: Page Handling mit Login und Fehlerverhalten / Serviceinterface	2.0			
	14.11.2010	GeoTrust Identity Validation für App Hub Account	1.0			
	14.11.2010	WP7: Login Dialog + Device Discovery View	6.0			
				33.0	**198.5**	205.7
Woche 9	16.11.2010	Besprechungen mit A. Schlumberger (DeviceDiscovery API) + R. Meier (Controller Web Interface)	1.5			
	17.11.2010	WP7: XAML Verständnis DataTemplate, ControlTemplate, Styles	3.0			
	17.11.2010	WP7: DeviceDiscovery ListView Selection, BuildingHierarchy Tree Selection	6.0			
	17.11.2010	Besprechung mit Reto: NodeSetup Webinterface, Handling von Text-Ressourcen	1.5			
	18.11.2010	WP7: DeviceDetails View, DeviceDiscovery View, BuildingHierarchy View	11.0			
	19.11.2010	WP7: Statusanzeige der Controller, Devicedetails, Page Navigation verbessert, Login Page Statushandling	7.0			
	21.11.2010	Page Navigation Map erstellt	3.5			
				33.5	**232.0**	231.4

Zeiterfassung

Projekt: **Mobile Commissioning Tool for Room Automation**
Teammitglied: **Kaspar Fenner**

Durchschnittliche Soll-Arbeitszeit: **25.7**

Woche	Datum	Tätigkeit	Zeit	Zeit/Wo	Total	Soll
Woche 10	22.11.2010	Besprechung mit Reto: Meilensteinabschluss, weiteres Vorgehen	1.0			
	22.11.2010	Projektplan/Zeitplanung aktualisiert	1.0			
	22.11.2010	WP7: Receive Service Pin	2.0			
	23.11.2010	WP7: Receive Service Pin	1.0			
	23.11.2010	WP7: Analyse von langsamen Aufbau der Building Hierarchy	1.0			
	24.11.2010	WP7: ContextMenu Fehler korrigiert	0.5			
	24.11.2010	WP7: User Interface Konzept für Point Test erarbeitet	2.0			
	24.11.2010	Weekly Meeting mit H. Huser und A. Rinkel	2.0			
	24.11.2010	Statusbesprechung mit Reto, Technischer Bericht	1.5			
	25.11.2010	WP7: Point Selection View, viele Verbesserungen in Building Hierarchy View (ContextMenu, Lazy Loading, visuelle Darstellung)	6.5			
	25.11.2010	Kurzes Statusmeeting mit K. Panteleeva und R. Föhn	0.5			
	26.11.2010	WP7: Point Selection View, Aufruf und Gerüst von Point Test View	7.0			
	27.11.2010	WP7: Point Test View	3.0			
	28.11.2010	WP7: Point Test View	8.0			
				37.0	**269.0**	257.1
Woche 11	30.11.2010	WP7: Expand/Collaps Icons/Handling mit visuellem "Upgrade" von Building Hierarchy View + Kleine Verbesserungen in allen Views	5.0			
	30.11.2010	WP7: Laden/Speichern von Status bei Deaktivierung/Thombstoning	1.5			
	01.12.2010	WP7: Point Test View (Autorefresh, ListPickers, Change Values)	9.0			
	02.12.2010	WP7: ListPicker-Problem lösen / Point Test Details View	6.0			
	03.12.2010	Meeting mit H. Huser in der Siemens	1.0			
	03.12.2010	Technischer Bericht: Kapiteldefinitionen, Einführung	7.0			
	05.12.2010	Technischer Bericht: Einführung	4.0			
				33.5	**302.5**	282.9
Woche 12	08.12.2010	Technischer Bericht: Einführung	10.0			
	09.12.2010	Besprechung (Review) mit Reto: Technischer Bericht	2.5			
	09.12.2010	Technischer Bericht: Einführung	5.0			
	10.12.2010	Technischer Bericht: Einführung, Review	3.0			
	10.12.2010	WP7: Trending Graph	7.0			
	11.12.2010	WP7: Trending Graph	2.0			
	12.12.2010	WP7: Point Selection View (Collapse/Expand, Reliability/CommStatus display)	5.0			
	12.12.2010	WP7: Fehler gefixt und versch. Kleinigkeiten verbessert	1.5			
				36.0	**338.5**	308.6

Zeiterfassung

Projekt: **Mobile Commissioning Tool for Room Automation**
Teammitglied: **Kaspar Fenner**

Durchschnittliche Soll-Arbeitszeit: **25.7**

Woche	Datum	Tätigkeit	Zeit	Zeit/Wo	Total	Soll
Woche 13	14.12.2010	Abstract für Diplomarbeitsbroschüre	1.5			
	15.12.2010	Abstract für Diplomarbeitsbroschüre	7.5			
	15.12.2010	Technischer Bericht: Einführung WP7-Entwicklung	1.0			
	16.12.2010	Code-Comments überprüft und nachgeführt	2.5			
	16.12.2010	Technischer Bericht: Einführung WP7-Entwicklung	4.5			
	17.12.2010	Besprechung mit Reto	1.0			
	17.12.2010	Technischer Bericht: Einführung WP7-Entwicklung	4.5			
	17.12.2010	Projektlogo-Design	4.0			
	18.12.2010	WP7-App: Cleanup	2.5			
	18.12.2010	Technischer Bericht: WP7-App (SCT)	4.0			
	19.12.2010	Technischer Bericht: WP7-App (SCT)	7.0			
				40.0	**378.5**	334.3
Woche 14	20.12.2010	Technischer Bericht: WP7-App (SCT)	4.0			
	20.12.2010	Besprechung mit Reto Poster, Zeitplanung	1.0			
	20.12.2010	Poster: Mit Reto überarbeitet	4.0			
	20.12.2010	Meeting mit H. Huser: Abstract für Brochüre, Poster, Tech. Bericht	1.0			
	21.12.2010	Technischer Bericht: WP7-App (SCT)	9.0			
	22.12.2010	Technischer Bericht: Schlussfolgerungen, Korrekturen in Kapitel 1 + 9, Persönlicher Bericht	8.0			
	22.12.2010	Glossar/Literaturverzeichnis extrahiert, Allgemeines Cleanup von Dokumenten	3.0			
	23.12.2010	Allgemeines Cleanup von Dokumenten	3.0			
	23.12.2010	CDs, Drucken + Binden	2.0			
				35.0	**413.5**	360.0

Total Ist - Arbeitszeit: **413.5**
Total Soll - Arbeitszeit: **360.0**
Differenz: **53.5**

Requirement Specification

SMART COMMISSIONING

Windows Phone 7

Bachelorarbeit:	Mobile Commissioning Tool for Room Automation
Autoren:	Kaspar Fenner Reto Schneebeli
Betreuer:	Prof. Hansjörg Huser
Projektpartner:	Siemens Schweiz AG Building Technologies, Zug
Experte:	Stefan Zettel, Ascentiv AG, Zürich

0. Document Information

0.1. Revision History

Date	Revision	Remarks	Author
22.09.2010	0.1	Initial Version	Reto Schneebeli
23.09.2010	0.2	Format template changed and main Use Cases added	Reto Schneebeli
30.09.2010	0.3	Use Cases brief	RS, KF
06.10.2010	0.4	Non-functional requirements	Kaspar Fenner
10.10.2010	0.5	Overview texts, Use Case diagram Use Cases 3, 4, 6 “fully dressed”	Reto Schneebeli
10.10.2010	0.6	Use Cases 1, 2 “fully dressed”	Kaspar Fenner
14.10.2010	0.7	UI Sketches added Use Case 11,14,15,16,17 “fully dressed”	Reto Schneebeli
14.10.2010	0.8	Chapter 2+3 updated + Use cases 9, 10 “fully dressed”	Kaspar Fenner
28.10.2010	0.9	Updated after review comments	Kaspar Fenner
22.12.2010	1.0	Finalized	RS, KF

0.2. Table of Contents

0. Document Information 2
0.1. Revision History 2
0.2. Table of Contents 3
1. Introduction, general 5
1.1. Document purpose 5
1.2. Validity, scope of applicability 5
1.3. Glossary of new terms, abbreviations 5
1.4. Referenced documents 5
1.5. Document overview 5
2. General Description 6
2.1. Product perspective 6
2.2. Product functions 6
2.3. User characteristics 6
2.4. Limitations 6
2.5. Assumptions 7
2.6. Dependencies 7
3. Specific requirements 7
3.1. Functional requirements 7
3.1.1. Controller simulation 7
3.1.2. Laptop as gateway 7
3.1.3. Login (single sign-on) 7
3.2. Usability 7
3.3. Reliability 7
3.4. Performance 8
3.5. Privacy 8
3.6. Data security 8
3.7. Maintainability 8
3.8. Interfaces 8
3.8.1. Device discovery API 8
3.8.2. Simulation tool web service 8
3.8.3. Gateway web service 8
3.9. License requirements 8
3.10. Used Standards 8
4. Use Cases 9
4.1. Use Case Diagram 9
4.2. Actors 10
4.3. General error handling 10
4.4. UI Storyboard 10
4.5. General Preconditions 10
4.6. UC01: Start web service on gateway 11
4.7. UC02: Connect gateway 11
4.8. UC03: Browse RA controllers 12
4.9. UC04: Map RA controller to corresponding hardware 13
4.10. UC05: Remove controller - hardware mapping (optional) 13
4.11. UC06: Search RA controllers with filter 13
4.12. UC07: Configure RA controller (optional) 14
4.13. UC08: Unconfigure RA controller (optional) 14
4.14. UC09: Send wink to RA controller 14
4.15. UC10: Receive service pin 15
4.16. UC11: Show controller details 15
4.17. UC12: Automatically search corresponding hardware (optional) 16
4.18. UC13: Download Device- and Application Configuration (optional) 16
4.19. UC14: Browse data points of an RA controller (focus on TXIO) 16
4.20. UC15: Perform wiring test for a selection of points (focus on TXIO) 16
4.21. UC16: Set commissioning state 17

4.22. UC17: Add commissioning comment 18
4.23. UC18: Release all commanded objects (optional) 18
4.24. UC19: Generate commissioning report (optional) 18
4.25. UC20: Show alarm (optional) 18
4.26. UC21: Confirm alarm (optional) 18

1. Introduction, general

1.1. Document purpose

The requirements specification describes the basic requirements of the software. Especially the use cases build the foundation of the later software design.

1.2. Validity, scope of applicability

The validity of this document spans over the whole project duration of 14 weeks. The document will be updated regularly and changes will be added to the revision history.

1.3. Glossary of new terms, abbreviations

Term	Description
SystemONE	System ONE (Engineering-System for room automation of Siemens BT)
TIA Portal	Totally Integrated Automation Portal (Software framework used and extended by System ONE)
RA	Room Automation
WP7	Windows Phone 7 is the new mobile platform from Microsoft and replaces Windows Mobile (Release: late October 2010).
SCT	Smart Commissioning Tool is the name of the WP7 app.
SCG	Smart Commissioning Gateway is the name of the web service running on the laptop.
WCF	The Windows Communication Foundation is an application programming interface in the .NET Framework for building connected, service-oriented applications.
REST	Representational State Transfer is a style of software architecture for distributed hypermedia systems such as the World Wide Web.
AP	Wireless Access Point – a bridging device that makes the transition from LAN to WLAN
NIC	Network Interface Card
DP	Data Point – an input or output value of a peripheral device. It can be analog, digital or multistate.
BACnet	Building Automation and Control Networks (Communication protocol for building automation and control networks)
Building Hierarchy	A building can be divided into floors and rooms. The building hierarchy is the hierarchical structure of buildings, floors and rooms.
SARB, DALI, TXIO	SARB, DALI and TXIO are subsystems in the building and room automation of Siemens BT.

1.4. Referenced documents

Aufgabenstellung.pdf

1.5. Document overview

The next chapter describes the product from different perspectives, followed by a list of functional and non-functional requirements. The last chapter describes the use cases in detail.

2. General Description

2.1. Product perspective

The *Smart Commissioning* software consists of the *Smart Commissioning Tool* (SCT) representing the WP7 app and the *Smart Commissioning Gateway* (SCG) being the web service. The WP7 app is used for the commissioning of RA controllers which are installed in a building. What commissioning basically means is downloading the configuration files on the controller and validating the correct wiring of the field devices. Currently, this task can be accomplished through various *DESIGO* tools which run on a Windows laptop. With SystemONE the point test will be performed through a web browser accessing a website hosted directly on the RA controller. The need for a laptop has several drawbacks for the installer as he needs to walk around the various rooms in a building to do the data point test, while the laptop needs to stay connected to the RA controller IP network. He also needs to monitor the value changes on the laptop while he manipulates the data point inputs. Usually, this task requires two engineers because it was too complicated to handle for a single person. As a solution the *DESIGO Point Test* (DPT) was developed which provides a trending functionality to record the value changes over time of a selection of data points. This allows for a one-man commissioning because there's no need for an engineer to sit in front of the laptop all the time. Although this solution is not bad, there are still drawbacks like having to carry the laptop around or switching between manipulating data point inputs and the laptop to check what the outcome was or vice versa.

With SCT we go one step further: By putting the *Node Discovery/Setup* and DPT functionality in a WP7 app, the installer will be able to have the smartphone in one hand while he physically manipulates the data point inputs with his other hand. Sometimes the RA controllers are mounted to the ceiling or in the intermediate ceiling. In this case he could take the smartphone with him while he climbs the ladder. Because he can follow the value changes directly on the smartphone he doesn't need the trending chart anymore for a time-displaced verification.

2.2. Product functions

SCT will provide the following basic functionality:

- Filtered search for RA controllers
- Browsing for RA controllers via building hierarchy
- Configure/unconfigure RA controller
- Receive service pin from RA controller
- Send wink to RA controller
- Browse data points of RA controller
- Perform wiring test for selected data points
- Set commissioning status and comment
- Generate commissioning report

Because SCT will be developed as a prototype during the bachelor thesis, the software is not required to work with actual, physical controllers. The prototype should show what is possible on a smartphone and how it could help in reducing engineering costs and making the installer's life easier. To demonstrate the functionality of SCT in a dynamic environment, the RA controllers will be simulated through a very simple simulation tool.

2.3. User characteristics

SCT targets installers which normally have only standard computer experience. It cannot be expected that they get a deep introduction to the application. In the best case they know the existing DESIGO tools used to do the commissioning today. The application needs to be self-explanatory as far as possible. It will help the users when the usability of the application provides a similar experience on the smartphone and on the computer.

2.4. Limitations

WP7 is limited to HTTP/HTTPS communication over standard ports. It is not possible to create TCP/UDP sockets. But this is needed for the initial communication with the controllers as the device identification and configuration protocol (DICP) is based on UDP multicast. This has a major influence on system architecture and technology selection and led to the idea of having a laptop as a gateway between the smartphone and the controller network.

2.5. Assumptions

During engineering in TIA Portal the building hierarchy can be defined. This building hierarchy doesn't include the RA controllers. To discover which rooms are assigned to which controllers, the DESIGO Insight export data needs to be parsed.

For the Bachelor thesis, we assume that this data is already parsed and added to the building hierarchy and saved as XML file. We define this file as a must-have in order to work with SCT. The file contains all buildings, floors, rooms and the RA controllers which can be assigned to every level (floor is standard). This file is available to the Installer and must be placed on the gateway.

2.6. Dependencies

Component/Product	Responsible/Contact
Device Identification and Configuration Protocol (API)	Andreas Schlumberger (Tel. 3177)
RA controller web architecture and interfaces	Roger Meier (Tel. 4942)

3. Specific requirements

3.1. Functional requirements

Most functional requirements are described by the use cases in chapter 4. Other functional requirements that cannot be described from a user perspective are listed below.

3.1.1. Controller simulation

RA controllers should be simulated through software on a computer. A simple tool shall be developed which provides creation, configuration, commanding and monitoring of the simulated RA controller instances.

3.1.2. Laptop as gateway

A laptop shall be used as a gateway between the smartphone and the controller IP network. This has several benefits:

- Laptop can be used as a wireless access point via ad hoc network (physical media change from LAN to WLAN)
- Central access point for all mobile devices
- Bypass restrictions of WP7 OS allowing no UDP multicast sockets, which is required for device identification and configuration

3.1.3. Login (single sign-on)

The web site of the RA controllers is protected by a username/password. The same basic authentication should also be implemented by SCT on the smartphone, but the login should only be done once for the whole controller network (single sign-on).

3.2. Usability

Usability-wise the WP7 app should provide a similar experience as the commissioning tools on the PC including the new controller hosted website used for the wiring test. Generally, this means providing similar functionality and using well-known terms and graphics. Everything else will probably look and behave differently as the WP7 app will be touch-controlled and the screen resolution is only 480 x 800 pixels. In contrast to a normal PC application which is keyboard & mouse-controlled and the most common screen resolution in the year 2010 is 1280 x 1024.

The GUI should be fully localizable and provide support for multiple languages. During the bachelor thesis, only English localization will be available.

3.3. Reliability

n/a

3.4. Performance

The GUI should always be responsive to user interaction. Blocking or CPU intensive tasks should be run in the background. It has to be clearly visible (hour glass, progress bar, etc.) to the user in the case where the GUI may not be able to respond to any user commands.

On the client side no additional storage is needed aside from a small XML file that holds some application settings which can be customized by the user.

The number of smartphones which can be connected to a gateway at a time shall not be limited to an exact number. In the standard use case only a single smartphone will be used. In special cases it could be necessary to connect two or three smartphones at a time. Connecting more smartphones is definitely not required at the moment aside from demo purpose. Perhaps this could change in the future if additional use cases are found. During the bachelor thesis the case with a maximum of two concurrently connected phones will be tested (if enough hardware is available).

3.5. Privacy

The WP7 App shall be protected with a username/password combination. The connection between the smartphone and the gateway shall be secured to prevent sniffing-attacks. The details about the used standard/technology have to be further analyzed.

There exists a flaw regarding the connection between the gateway and the controllers which is only secured by an authentication token which is retrieved in exchange for a username/password combination, but the connection will not be encrypted which would allow for a sniffing-attack. This is a well-known problem as the whole BACnet standard used to communicate with the controllers provides no built-in security as of today.

An advanced security concept shall be developed during the BA thesis as a separate document.

3.6. Data security

Data security or back-up strategies are not relevant for this project as there is no important data which needs to be stored. Most of the information is retrieved/written online from/to the controllers or directly retrieved from the engineering data. The security of this data is handled in other workflows and is not covered by this project.

3.7. Maintainability

The software must be easily extensible with new functionality. Even though it is developed only as a prototype, it should be well-refactored and come in a clean and product-ready condition.

3.8. Interfaces

3.8.1. Device discovery API

The device discovery protocol is used to search and detect RA controllers with unknown IP addresses in a network by using UDP multicast. This discovery functionality is currently being implemented as a C# DLL which can be used by SCT. The DLL is planned to be available in November. Until then the functionality can be simulated by using the same interface definitions and later the DLL can just be replaced by the real thing.

3.8.2. Simulation tool web service

The simulation tool should simulate virtual RA controllers providing web services like they were running on real RA controllers. The web service interface can be drastically simplified and should only provide the functionality required to perform a basic point test for TXIO modules. It also doesn't matter which technology is used for the web service and the communication between the SCT running on the gateway and the simulation tool.

3.8.3. Gateway web service

The web service on the gateway defines the interface for the communication with the smartphone clients. It should be a well thought-out interface following the REST architecture style being open for any client platform.

3.9. License requirements

n/a

3.10. Used Standards

Standard	Usage

Silverlight for Windows Phone	Silverlight is the application development platform for Windows Phone 7. The SCT mobile application will be developed with Silverlight for Windows Phone.
.NET Framework 4	On server side (gateway) we use the .NET Framework 4. On client side (mobile phone) we use the light version of the .NET Framework 4 for WP7 development.
XML	Every configuration file will be in XML format.
JSON	To be evaluated as an alternative format to XML in order to transfer objects between mobile phone and gateway.
WCF	To implement the communication on mobile phone and gateway side we use the communication platform WCF (Windows Communication Foundation).
REST	The communication standard between mobile phone and gateway will be REST (Representational State Transfer).

4. Use Cases

4.1. Use Case Diagram

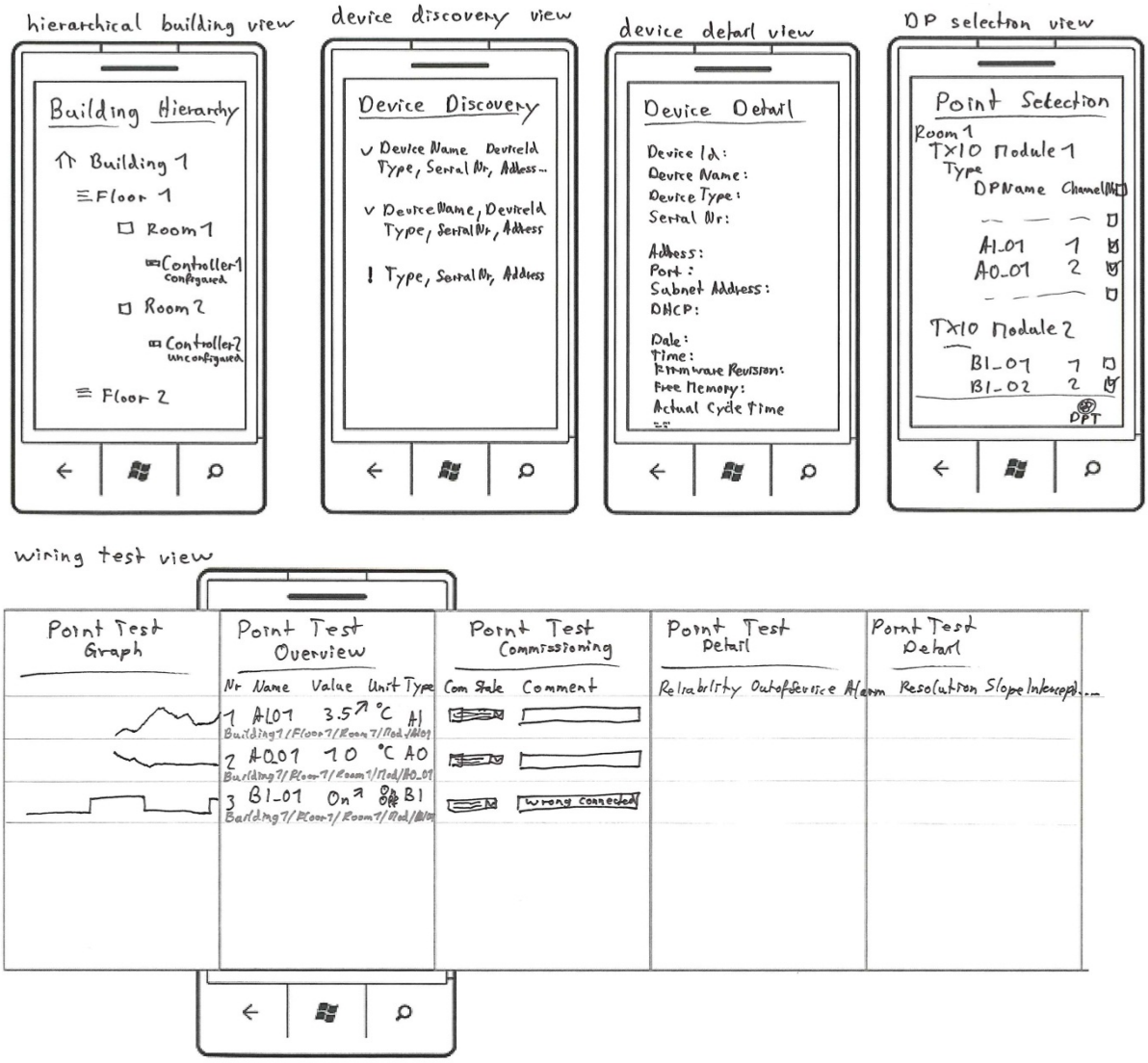
4.2. Actors

The Installer is the only actor. In the following use cases the “installer” is referred to as “user”.

4.3. General error handling

It is not yet decided, if and how the errors should be displayed on the user interface. This has to be further evaluated in the user interface design study. What is definitely required is having the option to write the errors to a log file. Because it is not very easy to read a log file on a small smartphone display, it might be a good idea if it could be sent back to the gateway, where it could be viewed on a big screen. The implementation details have to be clarified in the design specification.

4.4. UI Storyboard



4.5. General Preconditions

- A wireless access point or a configured ad hoc network. The network settings must be configured on both the smartphone and the laptop and basic communication must be possible between the two devices.
- The laptop which acts as a gateway must be connected to the RA controller IP network (can be simulated on the laptop).
- Access to the DESIGO Insight data export which includes XML files with the building hierarchy and the network, device and application configuration.

4.6. UC01: Start web service on gateway

Overview	The user starts the web service on the gateway (laptop).
Preconditions	Smart Commissioning Gateway (SCG) is installed on the laptop.
Postconditions	Web service is running and the smartphones are able to connect to the gateway.

Main Success Scenario	Extensions or Alternative Flows
1. User starts the SCG (executing an EXE file). 2. System shows a little configuration dialog. 3. System loads the selected NIC from config file. 4. User chooses the NIC of the RA controller network. 5. System starts the web service and responds to connecting smartphones.	2a. An instance of SCT is already running: 1. System recognizes this and opens the configuration dialog of the already running instance. 3a. Error: XML config file invalid or not available: 1. System selects the first NIC. 7. User starts the simulation tool to change and monitor the RA controllers (on the same or on another computer).

Special Requirements	- Only one instance of SCG can run at a time on a PC (singleton). - Allow to minimize the application to the system tray. - Configuration is saved to an XML config file as soon as a value changes.
Technology and Data Variations List	
Frequency of Occurrence	Not decisive.
Open Issues	

4.7. UC02: Connect gateway

Overview	The user connects with the gateway (laptop) and logs in with his credentials. After successful authentication the hierarchical building view is shown on screen.
Preconditions	The user has the URL/IP of the gateway and the username/password combination.
Postconditions	The hierarchical building view is displayed. The user is able to browse the RA controllers or switch to the device discovery view.

Main Success Scenario	Extensions or Alternative Flows
1. User starts SCT app on smartphone 2. System tries to automatically find an active gateway 3. User selects gateway 4. User enters username and password 5. System initiates connection with gateway (user gets authenticated) 6. System shows hierarchical building view	2-3a. System cannot discover an active gateway: 1. User enters URL/IP manually. 5-6a. Authentication fails: 1. User gets notification.

Special Requirements	
Technology and Data Variations List	

Frequency of Occurrence	Not decisive.
Open Issues	

4.8. UC03: Browse RA controllers

Overview	The user searches RA controllers by browsing through the building hierarchy. For each hierarchy element (building, floor, room) the name and type is visible, and for each RA controller the name, the type, and its status is visible. The status can either be configured, unconfigured or unmapped (means that the corresponding hardware is not yet mapped to the logical controller representation).
Preconditions	UC02
Postconditions	An RA controller might be selected and the application is ready for further operations.

Main Success Scenario	Extensions or Alternative Flows
1. System reads the XML file containing information about the building hierarchy. 2. System detects the status of each RA controller in the building hierarchy. 3. User navigates through the building hierarchy. 4. User selects an RA controller. 5. System shows the status dependent operations. *Configured:* - Controller Detail (UC11) - Wiring Test (UC13) - Send Wink (UC09) - Unconfigure (UC08) - Unmap (UC05) *Unconfigured:* - Controller Detail (UC11) - Send Wink (UC09) - Configure (UC07) - Unmap (UC05) *Unmapped:* - Controller Detail (UC11) - Wait for Service Pin (UC10) - Select manually (UC06) - Auto Map (UC12) 6. User might perform an operation or select another RA controller.	1a. Step 1 is just executed the first time, on following occasions it can be skipped. 1b. Error: XML file is invalid or unavailable: 1. User gets notification. 2. Device discovery view (UC06) will be displayed instead of the building hierarchy view.

Special Requirements	The status detection of each RA controller should run in background and not affect the responsiveness of the user interface.
Technology and Data Variations List	

Frequency of Occurrence	Not decisive.
Open Issues	

4.9. UC04: Map RA controller to corresponding hardware

Overview	The user maps an RA controller in the building hierarchy to its corresponding hardware.
Preconditions	An unmapped RA controller is selected in the building hierarchy (UC03).
Postconditions	The logical RA controller is mapped to its corresponding hardware.

Main Success Scenario	Extensions or Alternative Flows
1. User chooses 'Wait for Service Pin'. 2. User presses service pin on physical device. 3. System waits to receive the service pin signal. 4. System saves the mapping between the RA controller and the hardware. 5. System performs an RA controller status update.	1-3a. User chooses 'Select manually': ⇨ Execute UC06: Search RA controllers with filter. 1-3b. User chooses 'Auto Map': ⇨ Execute UC12: Automatically search corresponding hardware.

Special Requirements	When the user closes the SCT and stops the gateway and starts everything again, the mapping between logical RA controller and hardware should persist.
Technology and Data Variations List	
Frequency of Occurrence	Not decisive.
Open Issues	

4.10. UC05: Remove controller - hardware mapping (optional)

Overview	The user removes the mapping between logical RA controller and hardware.

4.11. UC06: Search RA controllers with filter

Overview	The user searches RA controllers in the network by specifying filter criteria. The system performs the device discovery and shows a flat list of controllers (device discovery view). This view is used to show controllers which are not in the building hierarchy or to manually map an RA controller in the building hierarchy to its corresponding hardware.
Preconditions	UC02
Postconditions	Flat list of controllers depending on specified filter criteria.

Main Success Scenario	Extensions or Alternative Flows
1. User switches to the device discovery view 2. User defines filter criteria: Filter by	1-2a. Use Case is started by UC04: Map RA controller to corresponding hardware: 1. System defines filter criteria: Filter by passed serial

controller status, by controller type or by serial number. 3. System performs device discovery with specified filter criteria. 4. System shows a flat list of controllers with the following properties depending on controller status: Status, Device Type, Serial Number *If configured:* Device Name, Device Id, Address, Port *If unconfigured:* Address (generated), Port (default)	number.
Special Requirements	Intelligent caching strategy for device discovery.
Technology and Data Variations List	
Frequency of Occurrence	Not decisive.
Open Issues	

4.12. UC07: Configure RA controller (optional)

Overview	The user configures an RA controller (download network, device and application configuration).

4.13. UC08: Unconfigure RA controller (optional)

Overview	The user performs an unconfigure of an RA controller (clear/reset controller).

4.14. UC09: Send wink to RA controller

Overview	Sometimes, there are multiple RA controllers right next to each other. In order to know, which is which, the user sends a wink via SCT to a particular controller. The physical controller will then flash an LED light so that it can be optically identified by the user.
Preconditions	A controller is selected in the building hierarchy view (UC3) or in the device discovery view (UC06). In case of the building hierarchy view the controller needs to be in “mapped” state.
Postconditions	The physical RA controller flashes its LED light.

Main Success Scenario	Extensions or Alternative Flows
1. User chooses ‘send wink’. 2. System sends wink to controller.	
Special Requirements	
Technology and Data Variations List	

Frequency of Occurrence	Not decisive.
Open Issues	

4.15. UC10: Receive service pin

Overview	When a service pin is pressed on a physical RA controller and the device discovery view is visible, the corresponding controller in the list should be selected/highlighted.
Preconditions	UC06
Postconditions	The controller is selected/highlighted in the device discovery view.

Main Success Scenario	Extensions or Alternative Flows
1. User presses service pin on physical device. 2. System selects/highlights controller in list.	

Special Requirements	
Technology and Data Variations List	
Frequency of Occurrence	Not decisive.
Open Issues	

4.16. UC11: Show controller details

Overview	The user has the possibility to get a detailed view of an RA controller with all available properties and operations (depending on controller status).
Preconditions	A controller is selected in the building hierarchy view (UC3) or in the device discovery view (UC6).
Postconditions	The device detail view is visible.

Main Success Scenario	Extensions or Alternative Flows
1. User chooses “Details” 2. System shows the device detail view for the RA controller with the following properties: Status, Device Name, Device Id, Device Type, Serial Number, Address, Port, Subnet, DHCP, Date, Time, Firmware Revision, Free Memory, Actual Cycle Time, Max Cycle Time, Last Download *Some properties might be empty depending on current status of the controller.*	

Special Requirements	

Technology and Data Variations List	
Frequency of Occurrence	Not decisive.
Open Issues	

4.17. UC12: Automatically search corresponding hardware (optional)

Overview	Map an RA controller in the building hierarchy to its corresponding hardware by automatically searching its serial number in the flat list of controllers.

4.18. UC13: Download Device- and Application Configuration (optional)

Overview	If the local configuration is newer than the configuration in the RA controller it is needed to download the device and application configuration again.

4.19. UC14: Browse data points of an RA controller (focus on TXIO)

Overview	The user browses/searches all the data points of an RA controller and selects one or more of them to perform the wiring test.
Preconditions	A configured controller is selected in the building hierarchy view (UC3) or in the device discovery view (UC6).
Postconditions	If one or more data points are selected, the system is ready to perform a wiring test.

Main Success Scenario	Extensions or Alternative Flows
1. System shows a list of all TXIO modules of an RA controller. 2. User navigates through modules. 3. System shows the data points of a module with channel information. 4. User choses the data points for the wiring test.	1a. Additionally, SARB and DALI devices are shown and all devices are grouped by their subsystem (SARB, DALI, TXIO).

Special Requirements	
Technology and Data Variations List	
Frequency of Occurrence	Not decisive.
Open Issues	

4.20. UC15: Perform wiring test for a selection of points (focus on TXIO)

Overview	The user performs the wiring test for a selection of points (usually, just for a single point at a time).
Preconditions	At least one data point is selected in the DP selection view (UC14)
Postconditions	The wiring test is done and each commissioning status and comment saved.

Main Success Scenario	Extensions or Alternative Flows
1. System shows the following attributes for each selected data point: Name, Description, Unit, Present Value, Reliability, Commissioning State, Commissioning Comment, Status Flag *Analog Input or Output:* Resolution, Correction Offset, Correction Factor, Change of Value, Min Present Value, Max Present Value *Binary Input or Output:* Polarity 2. System shows the trend of each selected data point 3. User performs the wiring test for one data point. *(Have a look on the extension flows for some wiring test examples)* 4. User sets the commissioning status. ⇨ Execute UC16: Set commissioning status 5. Go back to step 3 till every data point is tested. 6. User exits the wiring test view. 7. System shows the DP selection view.	3a. Analog Input: 1. User removes the wire of the device for this DP. 2. System should now show a value change. 3. User checks if the right data point has changed. 3b. Analog Output: 1. User increases or decreases the value (+, - button). 2. System increases or decreases the value by 10%. 3. User checks the reaction of the device. 3a. Binary or Multistate Input: 1. User changes something on the device which is responsible for this input value. 2. System shows the value change. 3. User checks if the right data point has changed. 3c. Binary or Multistate Output: 1. User increases or decreases the value (+, - button). 2. System increases or decreases the value by 1. 3. User checks the reaction of the device. 4a. The data point is wired wrong, the user wants to set the commissioning state and comment. ⇨ Execute UC16: Set commissioning state ⇨ Execute UC17: Set commissioning comment
Special Requirements	
Technology and Data Variations List	
Frequency of Occurrence	Not decisive.
Open Issues	

4.21. UC16: Set commissioning state

Overview	The user sets the commissioning state for a data point.
Preconditions	UC15
Postconditions	Commissioning state is saved for the data point.

Main Success Scenario	Extensions or Alternative Flows
1. User sets the commissioning state: Ok, Not checked, Defect. 2. System saves the commissioning state.	
Special Requirements	
Technology and Data Variations List	

Frequency of Occurrence	Not decisive.
Open Issues	

4.22. UC17: Add commissioning comment

Overview	The user adds a commissioning comment for a wrong wired data point.
Preconditions	UC15 and the data point is wired wrong so that a comment is needed.
Postconditions	Commissioning comment is saved for the data point.

Main Success Scenario	Extensions or Alternative Flows
1. User adds a commissioning comment. 2. System saves the commissioning comment.	

Special Requirements	
Technology and Data Variations List	
Frequency of Occurrence	Not decisive.
Open Issues	

4.23. UC18: Release all commanded objects (optional)

Overview	During wiring test some output values may be commanded with a higher priority than the attached sensor. After the wiring test a release for all these commanded values is needed.

4.24. UC19: Generate commissioning report (optional)

Overview	Generate a commissioning report containing the comments and statistics of the wiring test.

4.25. UC20: Show alarm (optional)

Overview	Show alarm state of each data point in wiring test.

4.26. UC21: Confirm alarm (optional)

Overview	Confirm pending alarms of data points in wiring test.

Domain Analysis

SMART COMMISSIONING

Windows Phone 7

Bachelorarbeit:	Mobile Commissioning Tool for Room Automation
Autoren:	Kaspar Fenner Reto Schneebeli
Betreuer:	Prof. Hansjörg Huser
Projektpartner:	Siemens Schweiz AG Building Technologies, Zug
Experte:	Stefan Zettel, Ascentiv AG, Zürich

0. Document Information

0.1. Revision History

Date	Revision	Remarks	Author
10.10.2010	0.1	Initial Version	Reto Schneebeli
13.10.2010	0.2	Worked in review comments	Reto Schneebeli
15.10.2010	0.3	Concept description added	Reto Schneebeli
18.10.2010	0.4	Review for MS2	KF, RS
28.10.2010	0.5	English language reviewed and edited	Kaspar Fenner
22.12.2010	1.0	Finalized	RS, KF

0.2. Table of Contents

0. Document Information 2
0.1. Revision History 2
0.2. Table of Contents 3
1. Introduction, general 4
1.1. Document purpose 4
1.2. Validity, scope of applicability 4
1.3. Glossary of new terms, abbreviations 4
1.4. Referenced documents 4
1.5. Document overview 4
2. Domain model 5
2.1. Diagram 5
2.2. Concept description 6
2.2.1. BuildingHierarchyNode 6
2.2.2. LogicalController 6
2.2.3. ControllerHW 7
2.2.4. FieldBus 7
2.2.5. Device 7
2.2.6. DataPoint 7
2.2.7. IoAddress 8

1. Introduction, general

1.1. Document purpose

This document contains the domain model with description and builds the foundation of the later software design.

1.2. Validity, scope of applicability

The validity of this document spans over the whole project duration of 14 weeks. The document will be updated regularly and changes will be added to the revision history.

1.3. Glossary of new terms, abbreviations

Term	Description
SystemONE	System ONE (Engineering-System for room automation of Siemens BT)
TIA Portal	Totally Integrated Automation Portal (Software framework used and extended by System ONE)
RA	Room Automation
SCT	Smart Commissioning Tool
BACnet	Building Automation and Control Networks (Communication protocol for building automation and control networks)
Building Hierarchy	A building can be divided into floors and rooms. The building hierarchy is the hierarchical structure of buildings, floors and rooms.
IO Address	Input Output Address

1.4. Referenced documents

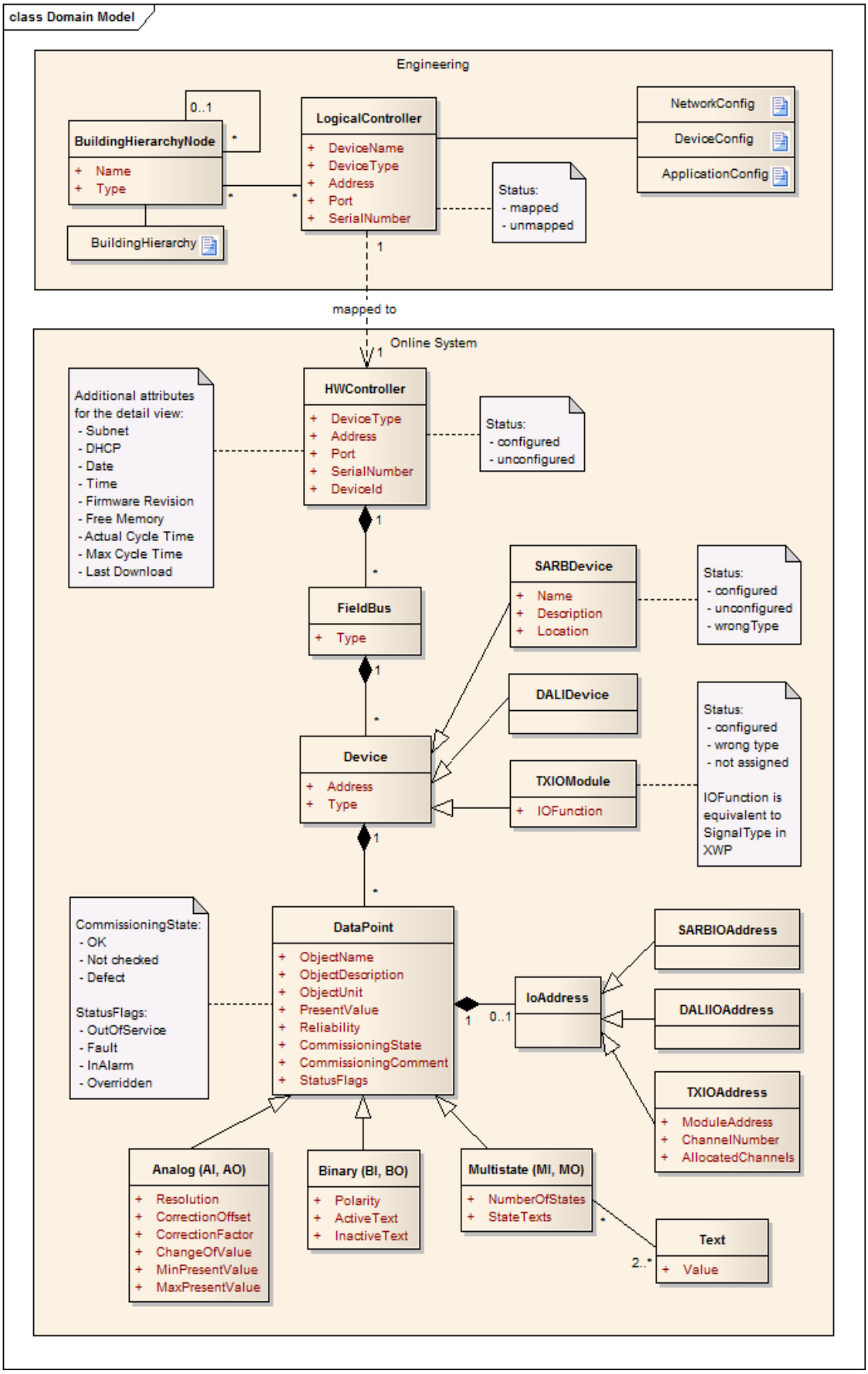
Aufgabenstellung.pdf

1.5. Document overview

The next chapter describes the domain model.

2. Domain model

2.1. Diagram



2.2. Concept description

The domain model is split into two parts, the *Engineering* and the *Online System*.

The part *Engineering* contains the concepts which are only available during engineering in TIA Portal.

- The *BuildingHierarchyNodes* form a tree-like structure representing the whole building hierarchy with buildings, floors and rooms. Every *LogicalController* is responsible for one or more rooms and a room can be controlled by one or more *LogicalControllers*. The building hierarchy and the mapping to the controllers is saved in the XML file *BuildingHierarchy*.
- The *LogicalController* has three configuration files:
  - *NetworkConfig*: Contains the node config data with information about the network.
  - *DeviceConfig*: Contain information about all the devices.
  - *ApplicationConfig*: Contains information about the control program.

  These three files are downloaded to the physical controller (*ControllerHW*) during 'Configure device' operation.

The part *Online System* covers all the information which is available on the physical RA controller.

- The *ControllerHW* represents the physical controller and must be mapped to the *LogicalController* which is set up during engineering.
- The *ControllerHW* has multiple *FieldBus* objects and the *FieldBus* has multiple *Device* objects.
- Each *Device* can be a SARB or DALI peripheral device (*SARBDevice*, *DALIDevice*) or a TXIO module (*TXIOModule*).
- The *Device* has multiple *DataPoints* which may have an *IoAddress*.
- There are six different kinds of *DataPoints:*
  - Analog inputs and analog outputs (*Analog*)
  - Binary inputs and binary outputs (*Binary*)
  - Multistate inputs and multistate outputs (*Multistate*)

2.2.1. BuildingHierarchyNode

Description	The *BuildingHierarchyNode* can be a building, a floor or a room. Together, they define the hierarchical structure of the buildings.
Attributes	*String* Name: Name of the hierarchy node. *Enum* Type: Type of the hierarchy node (building, floor, room).
Relations	0..1:n Relation with BuildingHierarchyNode: A hierarchy node may have multiple child hierarchy nodes. n:n Relation with LogicalController: A room may be controlled by multiple controllers and a controller may control multiple rooms.

2.2.2. LogicalController

Description	The *LogicalController* describes the controller defined during engineering in TIA Portal and not the physical controller.
Attributes	*String* DeviceName: Name of the controller, defined during engineering. *Enum* DeviceType: Device type of the controller. *String* Address: IP address of the controller. *Integer* Port: UDP port of the controller. *String* SerialNumber: Serial number of the controller (MAC address).
Relations	n:n Relation with BuildingHierarchyNode: A controller may control multiple rooms and a room may be controlled by multiple controllers. 1:1 Relation with ControllerHW: A logical controller is mapped to a physical controller.

2.2.3. ControllerHW

Description	The *ControllerHW* describes the physical controller in the BACnet network which controls the RA devices (light, blinds, heating and cooling devices, ...).
Attributes	*Integer* DeviceId: The ID of the controller. *Enum* DeviceType: Type of the controller (PXC3.E72, PXC3.E75A, ...). *String* Address: IP address of the controller (auto generated until configured). *Integer* Port: UDP port (default until configured). *String* SerialNumber: Serial number of the controller (MAC address).
Relations	1:1 Relation with ControllerHW: A logical controller is mapped to a physical controller. 1:n Relation with FieldBus: A controller has multiple field buses.

2.2.4. FieldBus

Description	The *FieldBus* is the physical or logical bus (rail) of a controller to get a connection with SARB or DALI devices or TXIO modules.
Attributes	*Enum* Type: The type of the field bus (SARB, DALI, TXIO);
Relations	1:n Relation with Device: On a field bus there are several devices of the same subsystem defined by the field bus. n:1 Relation with ControllerHW: A field bus belongs to a controller.

2.2.5. Device

Description	The *Device* is a SARB or DALI peripheral device or a TXIO module.
Attributes	*String* Address: The address of the device. *Enum* Type: The type of the device (TXM1.8D, TXM1.6R, PB_AX, RTS_AX, LB_AX, ...).
Relations	1:n Relation with Datapoint: A device may have multiple data points. n:1 Relation with FieldBus: A device belongs to a field bus.

2.2.6. DataPoint

Description	The *DataPoint* is an input or output value of a peripheral device or a TXIO module. It can be analog, digital or multistate.
Attributes	*String* ObjectName: The name of the data point. *String* ObjectDescription: The description of the data point. *Enum* ObjectUnit: The unit of the data point. *Integer/Real* PresentValue: The present value of the data point. *Enum* Reliability: The reliability of the data point (NoFault, NoSensor, ...) *Enum* CommissioningState: The commissioning state of the data point (OK, NotChecked, Defect). *String* CommissioningComment: The commissioning comment for wrong wired data points. *Enum* StatusFlags: The current status of the data point (OutOfService, Fault, InAlarm, Overriden).
Relations	1:0..1 Relation with IoAddress: A data point may have zero or one IO address. n:1 Relation with Device: A data point belongs to a device.

2.2.7. IoAddress

Description	The *IoAddress* is a logical address for an input or output data point. The syntax is subsystem dependent as SARB, DALI, and TXIO addresses have a different syntax.
Attributes	
Relations	0..1:1 Relation with DataPoint: An IO address belongs to a data point.

Software Architecture & Design

SMART COMMISSIONING

Windows Phone 7

Bachelorarbeit:	Mobile Commissioning Tool for Room Automation
Autoren:	Kaspar Fenner Reto Schneebeli
Betreuer:	Prof. Hansjörg Huser
Projektpartner:	Siemens Schweiz AG Building Technologies, Zug
Experte:	Stefan Zettel, Ascentiv AG, Zürich

0. Document Information

0.1. Revision History

Date	Revision	Remarks	Author
16.10.2010	0.1	Initial Version	Reto Schneebeli
29.10.2010	0.2	Physical Architecture, Figures added	Kaspar Fenner
31.10.2010	0.3	Chapter 2 + 3	Kaspar Fenner
01.11.2010	0.4	Web service interface description	Reto Schneebeli
07.11.2010	0.5	Changes on chapter 2 and 4	Reto Schneebeli
14.11.2010	0.6	Changes on chapter 4	Reto Schneebeli
21.11.2010	0.7	Chapter 11 “User Interface Design” added	Kaspar Fenner
01.12.2010	0.8	Web service interface description changed	Reto Schneebeli
22.12.2010	1.0	Finalized	RS, KF

0.2. Table of Contents

0. Document Information 2
0.1. Revision History 2
0.2. Table of Contents 3
0.3. List of Figures 4
1. Introduction, general 5
1.1. Document purpose 5
1.2. Validity, scope of applicability 5
1.3. Glossary of new terms, abbreviations 5
1.4. Referenced documents 5
2. Physical Architecture 6
3. Architectural Goals and Restrictions 7
3.1. Restrictions by Windows Phone 7 7
3.1.1. Socket communication 7
3.1.2. Differences between Silverlight and Silverlight for Windows Phone 7
3.2. Restrictions by service architecture 7
3.2.1. Polling 7
3.2.2. Long polling 7
3.3. Engineering laptop as wireless AP and network bridge 7
3.4. Engineering laptop as discovery server 8
3.5. Cross-platform web service architecture for mobile devices (REST) 8
4. Logical Architecture 10
4.1. Overview 10
4.2. Gateway (Smart Commissioning Gateway) 11
4.3. WP7 Client (Smart Commissioning Tool) 12
4.4. Service Interface Descriptions 13
4.4.1. Gateway (SCG) 13
4.4.2. RA controller (CST) 15
4.4.3. Data point attributes and BACnet Ids 16
4.4.4. Meta data enumerations 17
5. Implementation View 18
5.1. Common (Library) 18
5.1.1. Error handling 18
5.1.2. Logging 18
5.2. Gateway (SCG) 19
5.2.1. Business Logic - Authentication 19
5.2.2. Business Logic - Engineering Data 20
5.2.3. Business Logic - Node Setup 20
5.2.4. Sequence - Start device discovery 21
5.2.5. Sequence - Wait for service pin (long polling) 22
5.3. WP7 Client (SCT) 23
5.3.1. Service Layer 23
5.3.2. Service Event Arguments 24
5.3.3. Web Service Helper Classes 24
6. User Interface Design 25
6.1. Page Navigation Map 25

0.4. List of Figures

Figure 1: Physical architecture 6
Figure 2: Bridge two network adapters 8
Figure 3: Create a wireless ad hoc network 8
Figure 4: Logical architecture overview 10
Figure 5: Gateway architecture 11
Figure 6: WP7 client architecture 12
Figure 7: SCG web service interface 13
Figure 8: CST web service interface 15
Figure 9: Meta data enumerations 17
Figure 10: Common - Error handling 18
Figure 11: Common - Logging 18
Figure 12: Common - Logging Logger 19
Figure 13: Gateway - Authentication 19
Figure 14: Gateway - Engineering Data 20
Figure 15: Gateway - Node Setup 20
Figure 16: Gateway - Start device discovery 21
Figure 17: Gateway - Service Pin 22
Figure 18: Gateway - Wait for service pin 22
Figure 19: SCT - Service Layer 23
Figure 20: SCT - Service event arguments 24
Figure 21: SCT - Web Service Helper Classes 24
Figure 22: Page Navigation Map 25

1. Introduction, general

1.1. Document purpose

This document is only an addition to the 'Technical Report' and gives deeper information in some topics. It does not stand on its own and does not reflect a complete architecture and design document.

1.2. Validity, scope of applicability

The validity of this document spans over the whole project duration of 14 weeks. The document will be updated regularly and changes will be added to the revision history.

1.3. Glossary of new terms, abbreviations

Term	Description
System ONE	System ONE (Engineering-System for room automation of Siemens BT)
SCT	Smart Commissioning Tool is the name of the WP7 app.
SCG	Smart Commissioning Gateway is the name of the web service running on the laptop.
RA	Room Automation
CST	Controller Simulation Tool is the name of the software which simulates the RA controllers.
WP7	Windows Phone 7 is the new mobile platform from Microsoft and replaces Windows Mobile.
BACnet	Building Automation and Control Networks (Communication protocol for building automation and control networks)
DICP	Device Identification and Configuration Protocol (Discovery protocol for RA controllers)
Building Hierarchy	A building can be divided into floors and rooms. The building hierarchy is the hierarchical structure of buildings, floors and rooms.
Node Setup	The node setup is the general term for device discovery, configure and unconfigure of devices, send wink and receive service.

1.4. Referenced documents

RequirementSpecification.pdf

Domainanalysis.pdf

2. Physical Architecture

The SCT *Smart Commissioning Tool* is running on a smartphone and connects to the SCG *Smart Commissioning Gateway* running on the engineering laptop. The SCG hosts the building hierarchy and performs the device discovery over a UDP multicast protocol. This information is then used by the SCT to search and list RA controllers, which are directly connected to perform the data point test. The smartphone communicates via HTTP with the RESTful web services running on the engineering laptop and on the RA controllers. During the bachelor thesis, the RA controllers will be simulated on a PC since there are no hardware devices available because the firmware is still under development.

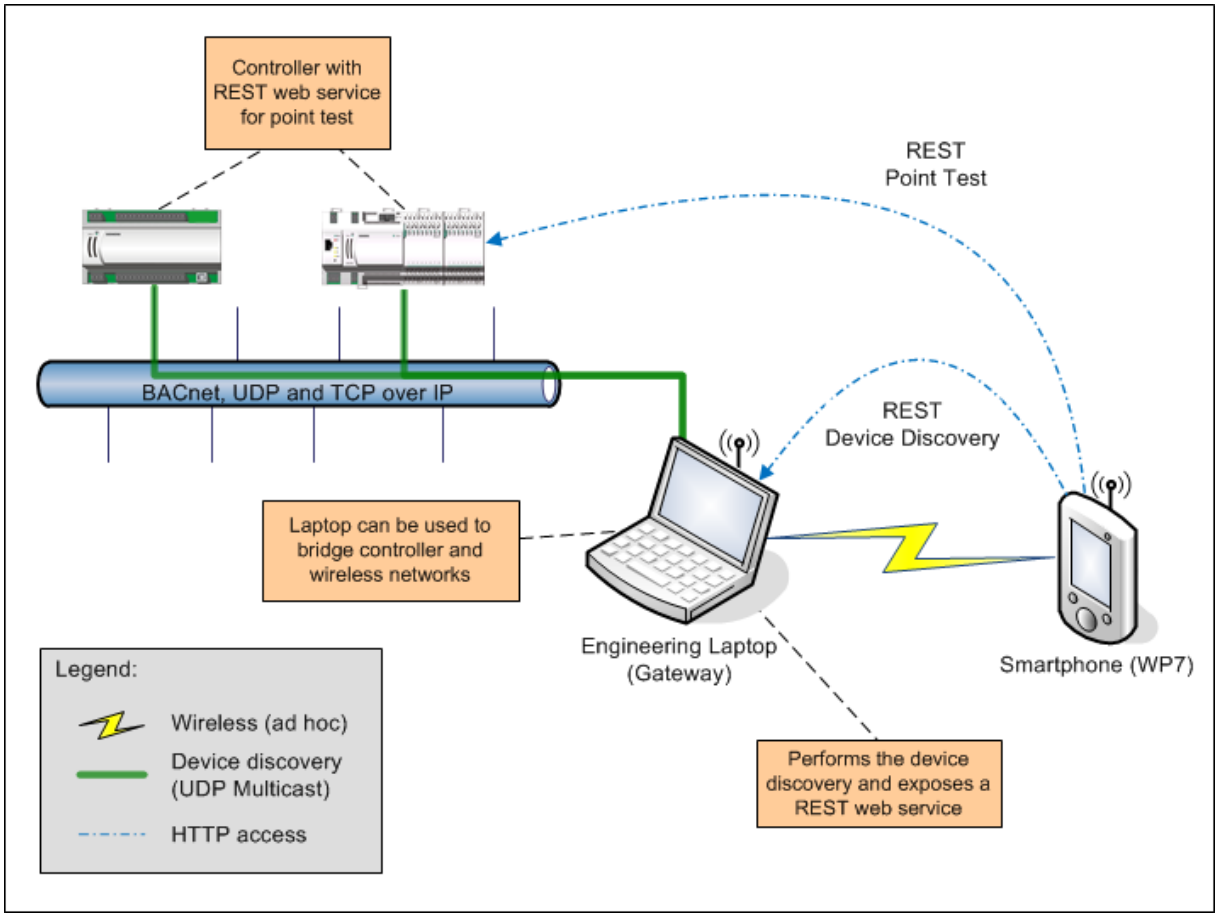


Figure 1: Physical architecture

The figure above shows the following components:

- IP network with RA controllers (sometimes referenced as 'devices')
- Engineering laptop used as discovery server and gateway between smartphones and RA controllers
- Smartphones used as clients

The engineering laptop (gateway) has the following three functions:

- It can be used as a network bridge and wireless access point which the smartphone clients can connect to and access the RA controllers behind the laptop. This removes the need for a separate wireless access point.
- It acts as a discovery server which the smartphone clients can use to find the RA controllers.
- It hosts the engineering data which is used to configure the RA controllers.

3. Architectural Goals and Restrictions

3.1. Restrictions by Windows Phone 7

3.1.1. Socket communication

Network communication is restricted to HTTP via `System.Net.WebClient` and there is no access to sockets in Silverlight for WP7. This means that it is not possible to communicate over UDP multicast from a WP7 phone what prevents using the device identification and configuration protocol (DICP) on WP7.

3.1.2. Differences between Silverlight and Silverlight for Windows Phone

Silverlight for Windows Phone is based on Silverlight 3. But there are several differences between Silverlight and Silverlight for Windows Phone which also supports some additional types specific to Windows Phone.

- A list of differences between Silverlight and Silverlight for Windows Phone: http://msdn.microsoft.com/en-us/library/ff426930(VS.95).aspx
- A list of unsupported types and a list of additional types supported in Windows Phone: http://msdn.microsoft.com/en-us/library/dd470087(v=VS.95).aspx

3.2. Restrictions by service architecture

In pure client server architecture based on services it is not possible to send a notification from server to the client. But when a user presses the service pin button on a controller we need to send a notification to the WP7 App to highlight this RA controller on the user interface.

3.2.1. Polling

One possibility to get informed on a ‘service pin’ is to poll the server and ask every time if a service pin was pressed. But that will cause a lot of unnecessary traffic.

3.2.2. Long polling

A better possibility is long polling. That means the client sends a request and the server doesn’t response immediately. The server holds the connection open and sleeps till he got a ‘service pin’. This strategy doesn’t scale well, because a lot of clients cause a lot of long open connections. But in our situation that doesn’t matter, we normally have one or maybe two clients. We decided to use this kind of polling.

3.3. Engineering laptop as wireless AP and network bridge

As an alternative to a separate wireless access point hooked-up to the RA controller network, the engineering laptop can be used to fulfill the same functionality. This can be achieved through an ad hoc wireless network and a layer 2 network bridge created in Windows. Both tasks can be accomplished through standard Windows tools on Windows XP and later.

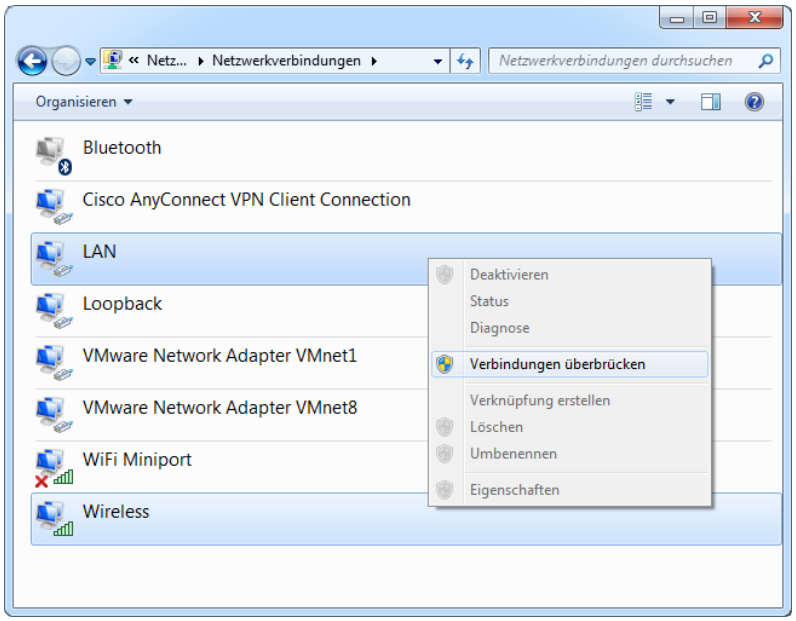


Figure 2: Bridge two network adapters

Figure 3: Create a wireless ad hoc network

3.4. Engineering laptop as discovery server

The fact that the DICP cannot be used on WP7 means that an intermediate node is required between the smartphone and the RA controllers, which performs the device discovery and sends the results back to the WP7 client. This led to the idea of having the SCG running on the engineering laptop and performing all tasks which involve communication via DICP. To make the device discovery controllable from the smartphone, the SCG exposes an API similar to the DICP API as a REST web service.

3.5. Cross-platform web service architecture for mobile devices (REST)

In search for a cross-platform web service architecture for mobile devices, REST was found to be the most appropriate. But REST isn't directly a software architecture, it's an architectural style. When REST is used in the context of HTTP, it describes some very basic principles of how a web service should look like and how clients and servers interact with each other by using concepts of the World Wide Web. Today, REST is used by almost all major web applications that make use of the client-server model where Facebook is probably one of the most prominent at the moment.

What sets REST apart of all other web service architectures like SOAP is that it works with the existing vocabulary and concepts of the World Wide Web like URIs, response codes, methods (GET, POST, …), etc. while SOAP encourages to create new concepts for every new application. That means, in SOAP we would define a new method 'getAllUsers()' where in REST we would use the existing HTTP URI concept and use 'GET Users/All' to accomplish the same task.

In order to use a RESTful web service, a client will only need to be able to communicate through HTTP. As this is possible on all current mobile and desktop platforms, it makes it very easy to create cross-platform compatible web services.

Developing the server side on the .NET platform allows using the WCF library also for hosting RESTful web services. This makes development and operation of RESTful web services very easy.

On the client side, however, there is no standard library that could be used to consume the web services. Luckily, this is no big deal as it is very easy to rebuild the services on the client side with help of standard HTTP –communication libraries on any possible platform.

4. Logical Architecture

4.1. Overview

The following illustration shows a simplified architectural overview of the *Smart Commissioning* software components. It shows which services are available and how the software components interact with each other. It also shows the different XML files used as information source and data storage.

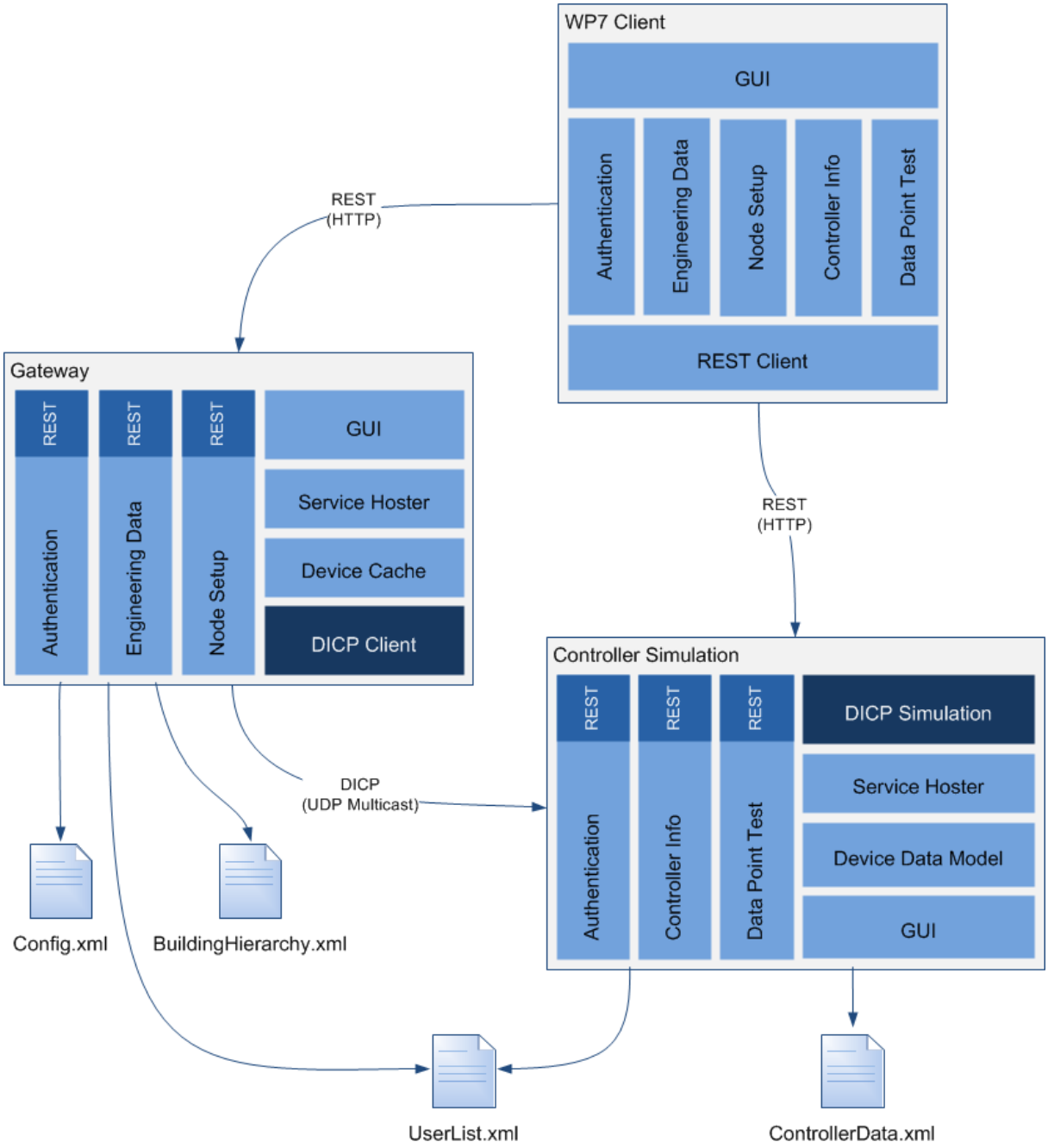


Figure 4: Logical architecture overview

WP7 Client

The *Smart Commissioning Tool* which runs on the windows phone 7 is the WP7 client. The WP7 client has no local storage. All information comes from the REST services on the *Gateway* and the controllers. It first connects the *Gateway* for a user authentication, to show the building hierarchy and to list all the RA controllers in the network. For detailed controller information or to perform the data point test, it has to access the REST services of a controller which run in the *Controller Simulation*.

Gateway

The *Gateway* is responsible for the user authentication, the provision of engineering data (building hierarchy) and the execution of the node setup (device discovery, configure/unconfigure, wink and service pin). It accesses the UserList.xml for user authentication, the BuildingHierarchy.xml for the engineering data and uses the DICP client to perform the node setup tasks.

Controller Simulation

The *Controller Simulation* has two responsibilities. The first responsibility is to simulate the controller part of the DICP that a real device discovery of simulated controllers is possible. The other responsibility is to simulate the REST service of each simulated controller that a point test is possible. The controller simulation tool provides a user interface to show, add and edit controllers, to start the simulation and to show, add and edit data points for point test simulation.

Config.xml

The file config.xml contains some general setting for the gateway.

BuildingHierarchy.xml

This file BuildingHierarchy.xml is exported from the engineering tool System One. It contains the whole building hierarchy with buildings and building elements (floor, room). Additionally it contains a list of logical RA controllers with some offline information to the controllers and the relation to one or more building element.

UserList.xml

The file UserList.xml contains some user information for a basic authentication (user name, first name, last name and password). It is used to simulate a basic authentication service.

ControllerData.xml

The file DeviceData.xml is the internal storage file for the device simulation. It contains all the information about modules and data points needed to simulate a point test.

4.2. Gateway (Smart Commissioning Gateway)

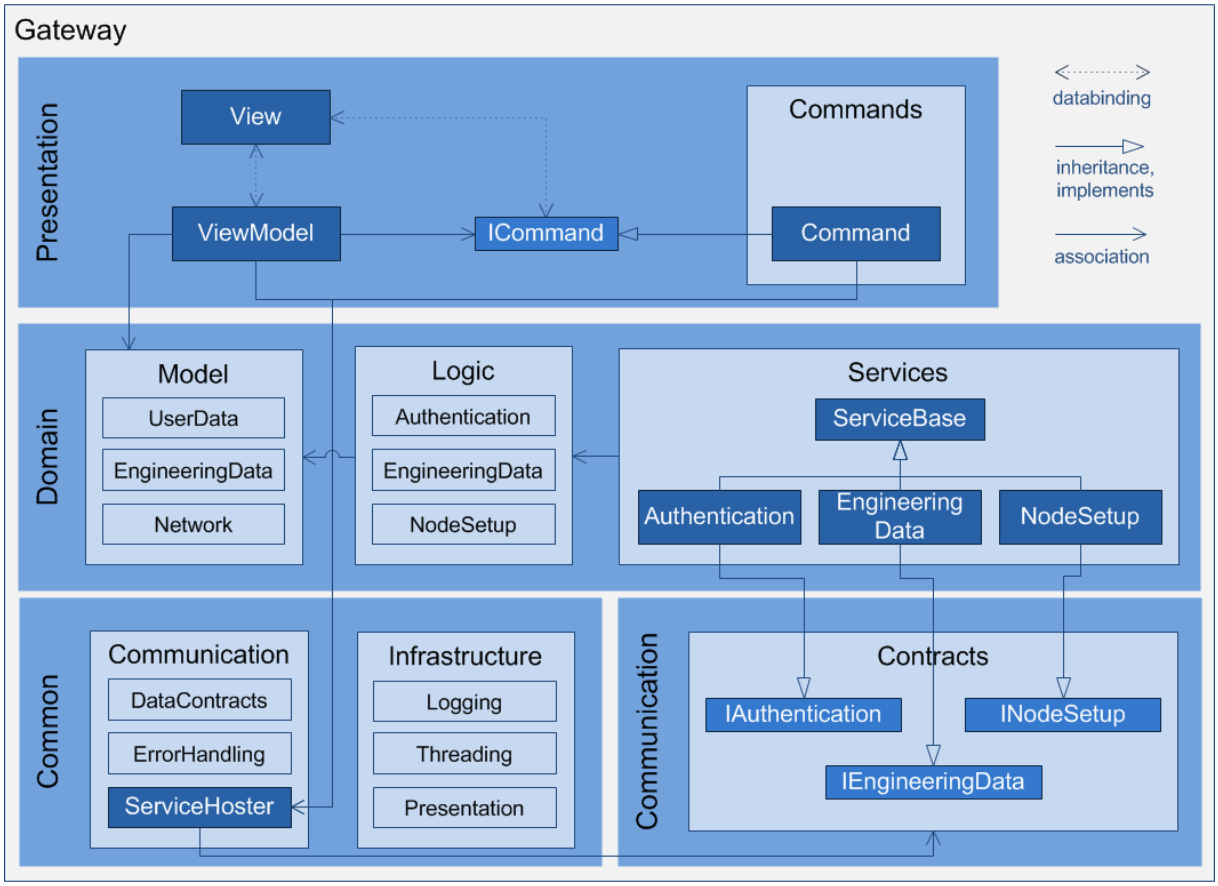


Figure 5: Gateway architecture

4.3. WP7 Client (Smart Commissioning Tool)

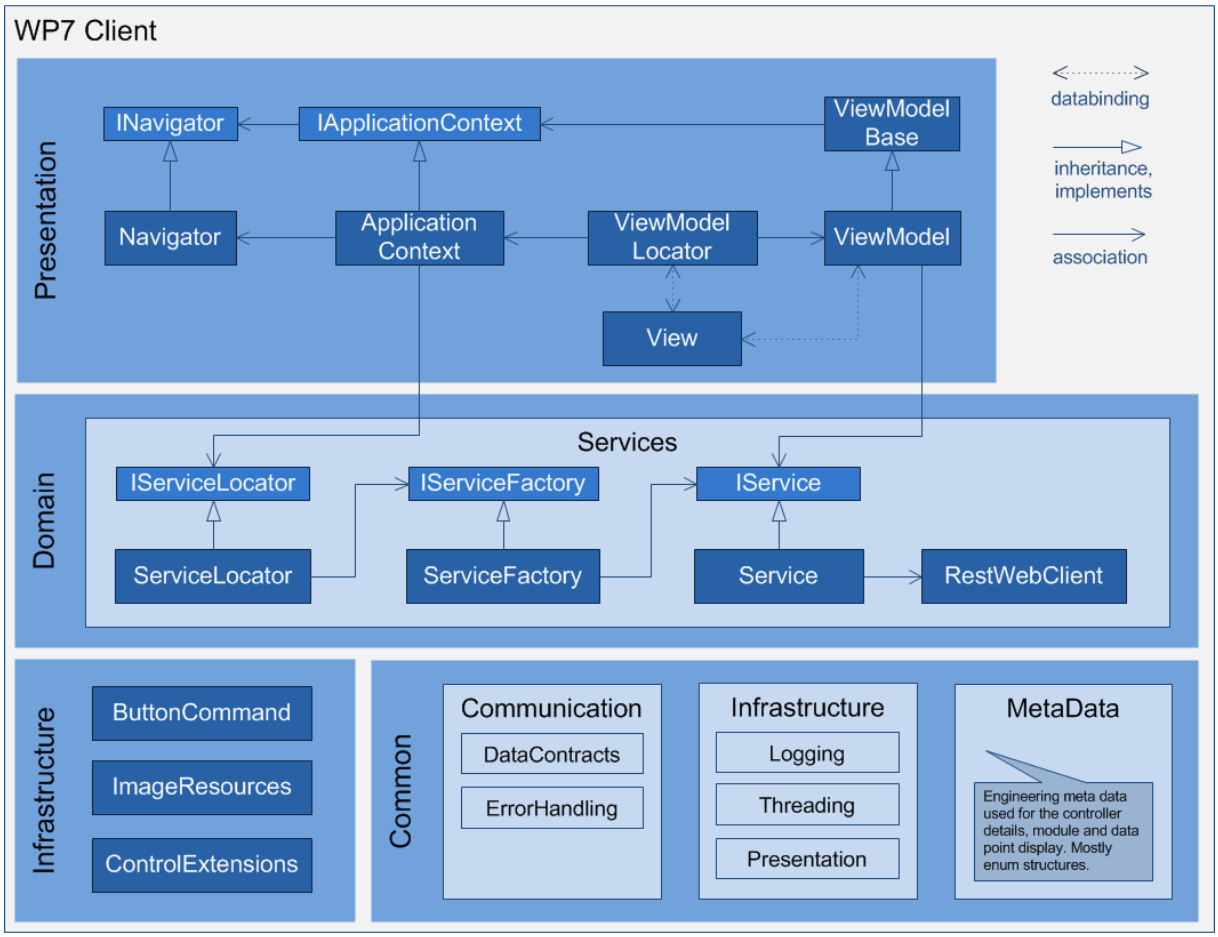


Figure 6: WP7 client architecture

4.4. Service Interface Descriptions

The following class diagrams show the web service interfaces and contract classes.

4.4.1. Gateway (SCG)

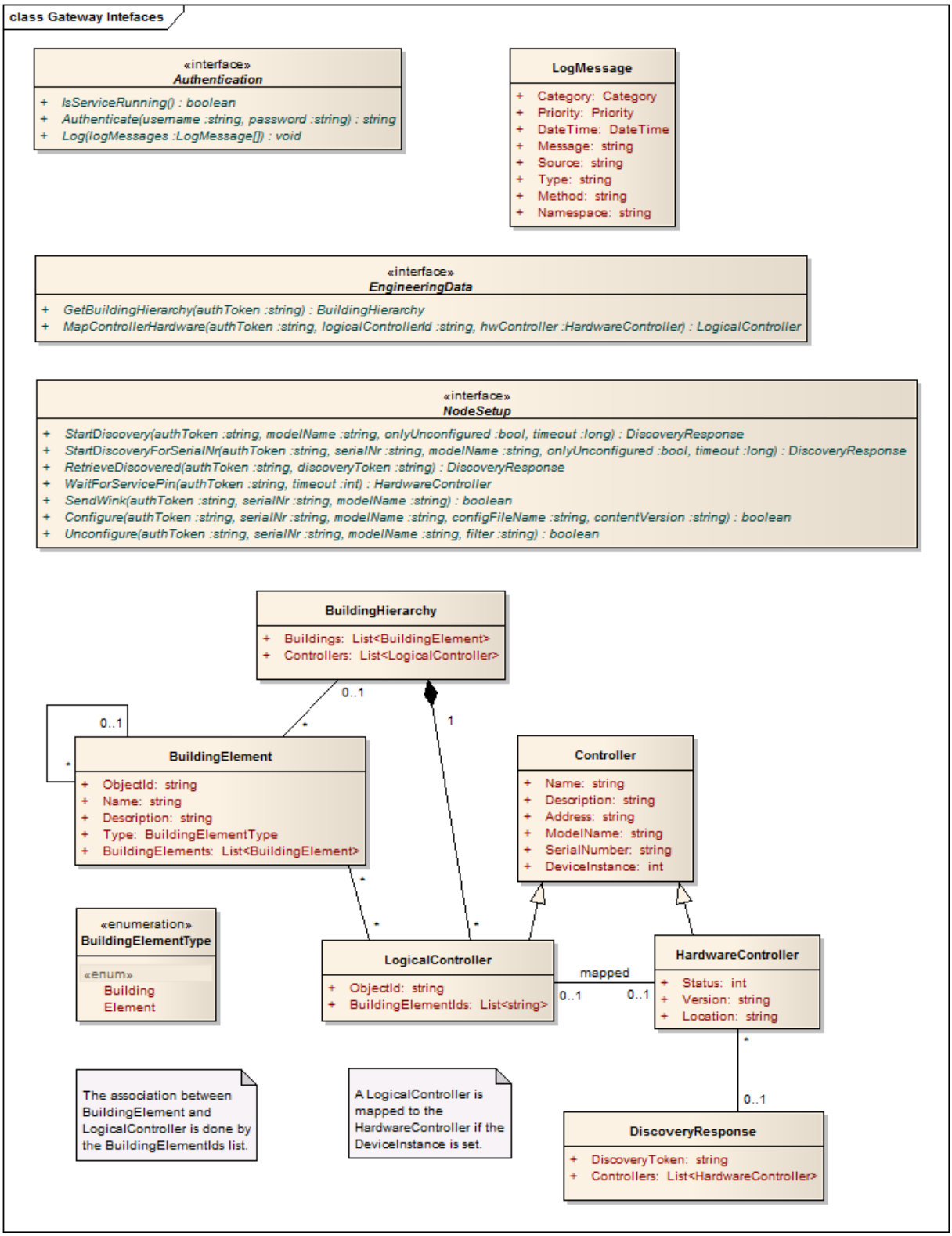


Figure 7: SCG web service interface

Example web service calls:

- Authenticate (HTTP GET):
  http://192.168.1.1/Authentication/Authenticate?user=Reto&pw=phone
- Log (HTTP POST):
  http://192.168.1.1/Authentication/Log
  HTTP POST content is an array of log messages (LogMessage[]) as a JSON serialized string.
- GetBuildingHierarchy (HTTP GET):
  http://192.168.1.1/EngineeringData/BuildingHierarchy?auth=c65b2a15-5b1b-4d7b-a487-81a9f0168047
- MapControllerHardware (HTTP PUT):
  http://192.168.1.1/EngineeringData/MapHardware?auth=c65b2a15-5b1b-4d7b-a487-81a9f0168047&id=Contr1
  HTTP PUT content is a HardwareController as a JSON serialized string.
- StartDiscovery (HTTP GET):
  http://192.168.1.1/NodeSetup/StartDiscovery?auth=c65b2a15-5b1b-4d7b-a487-81a9f0168047&model=PXC3.E72&uconf=False&timeout=1000
- RetrieveDiscovered (HTTP GET):
  http://192.168.1.1/NodeSetup/RetrieveDiscovered?auth=c65b2a15-5b1b-4d7b-a487-81a9f0168047&token=df10aa98-17cb-4572-bd0d-492d030f3116
- WaitForServicePin (HTTP GET):
  http://192.168.1.1/NodeSetup/ServicePin?auth=c65b2a15-5b1b-4d7b-a487-81a9f0168047&timeout=30000
- SendWink (HTTP PUT):
  http://192.168.1.1/NodeSetup/Wink?auth=c65b2a15-5b1b-4d7b-a487-81a9f0168047&serial=1234A09A55555555&model=PXC3.E72
  HTTP PUT content is empty.

4.4.2. RA controller (CST)

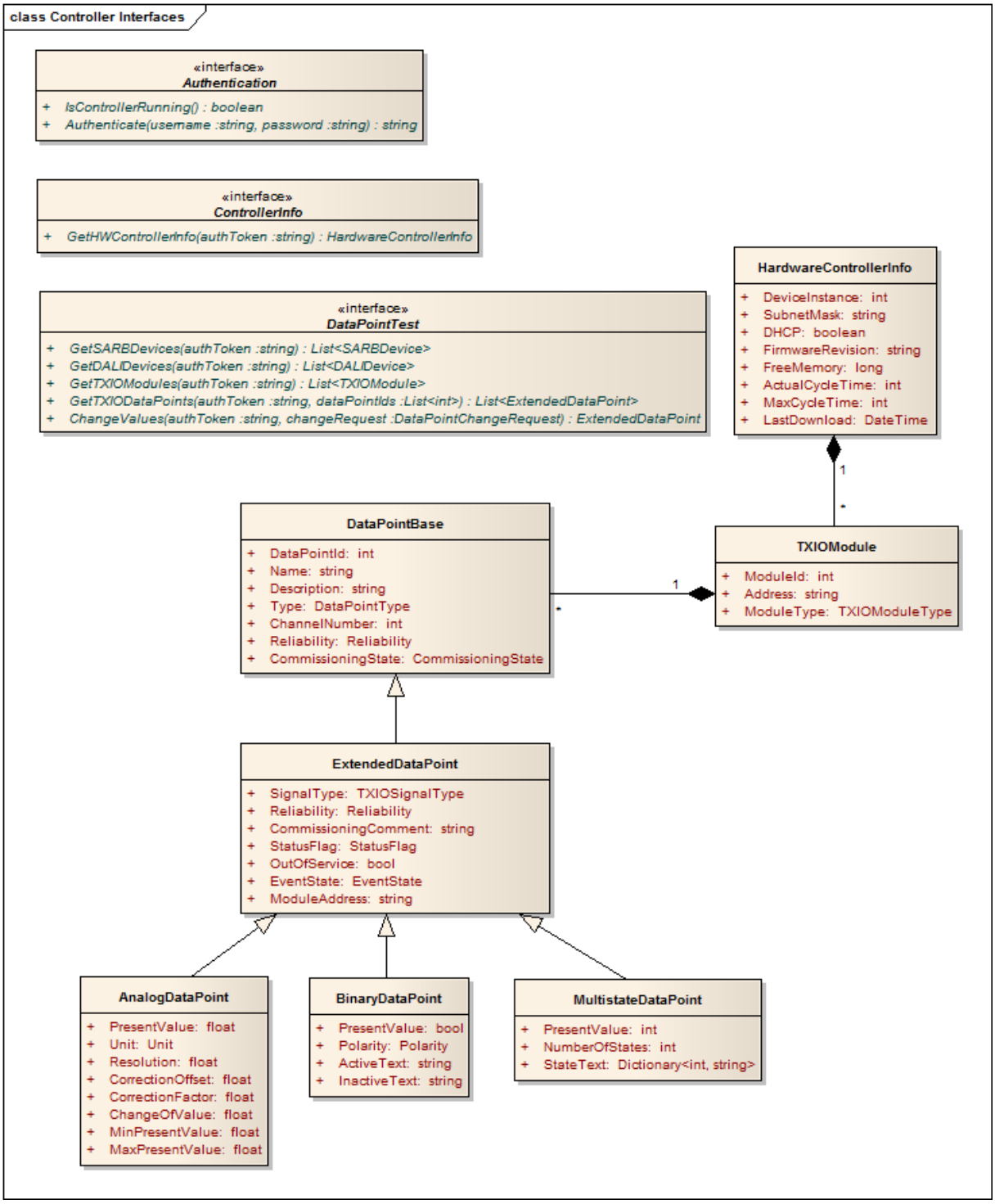


Figure 8: CST web service interface

Example web service calls:

- Authenticate (HTTP GET): http://192.168.1.1:11001/Authentication/Authenticate?user=Reto&pw=phone
- GetControllerInfo (HTTP GET): http://192.168.1.1:11001/ControllerInfo/Details?auth=6f1fb301-99e3-418b-8cb1-311813191b29
- GetTXIOModules (HTTP GET) http://192.168.1.1:11001/DataPointTest/TXIO/Modules?auth=6f1fb301-99e3-418b-8cb1-311813191b29
- GetTXIODataPoints (HTTP PUT)

http://192.168.1.1:11001/DataPointTest/TXIO/DataPoints?auth=6f1fb301-99e3-418b-8cb1-311813191b29
HTTP PUT content is a list of integers (List<int>) as a JSON serialized string.

- ChangeValues (HTTP PUT)
  http://192.168.1.1:11001/DataPointTest/TXIO/ChangeValues?auth=6f1fb301-99e3-418b-8cb1-311813191b29
  HTTP PUT content is a DataPointChangeRequest as a JSON serialized string.

4.4.3. Data point attributes and BACnet Ids

Attribute Name	BACnet Attribute Id	BACnet Attribute Name	Access (R, E, O)
DataPointId	75	object-identifier	R
Name	77	object-name	R
Description	28	description	O
Type	-	(not a BACnet attribute)	R
ChannelNumber	-	(not a BACnet attribute)	R
Unit	117	units	R
Reliability	103	reliability	O
CommissioningState	3606	ds1-commissioning-state	E
CommissioningComment	3607	ds1-commissioning-info	E
Status	111	status-flags	R
OutOfService	81	out-of-service	R
EventState	36	event-state	R
PresentValue	85	present-value	R
Resolution	106	resolution	O
CorrectionOffset	3604	ds1-correction-offset	E
CorrectionFactor	3605	ds1-correction-factor	E
ChangeOfValue	22	cov-increment	O
MinPresentValue	69	min-pres-value	O
MaxPresentValue	65	max-pres-value	O
Polarity	84	polarity	R
ActiveText	4	active-text	O
InactiveText	46	inactive-text	O
NumberOfStates	74	number-of-states	R
StateText	110	state-text	O

4.4.4. Meta data enumerations

Figure 9: Meta data enumerations

5. Implementation View

5.1. Common (Library)

5.1.1. Error handling

The *Error* class is used to define a specific error code *(ErrorCode)* with detailed error description. It is serialized as a JSON string to return a specific error from server to client. Every error code has an *EnumDescriptionAttribute* which contains the error text with placeholders.

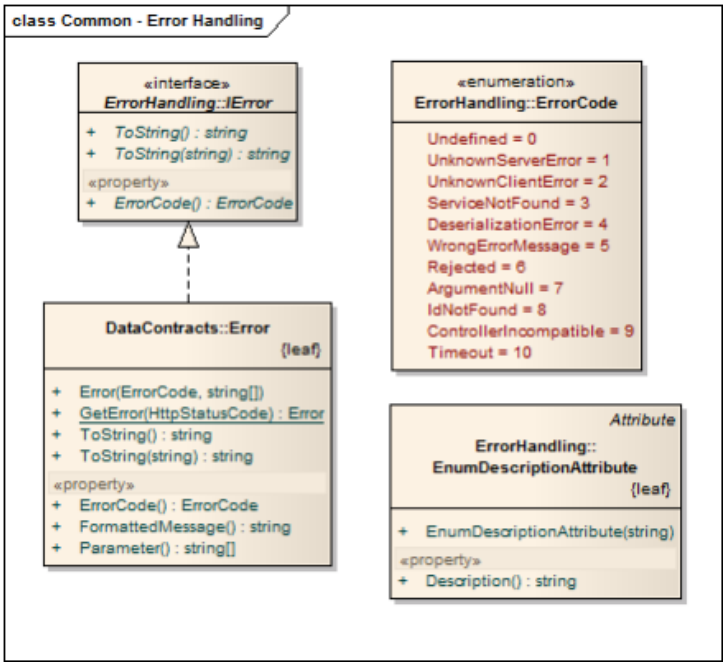


Figure 10: Common - Error handling

5.1.2. Logging

The class Logger has a static method *GetLogger* to get the instance of the *ILoggerFacade* interface. With this instance it is possible to write log messages, add logger targets and remove logger targets.

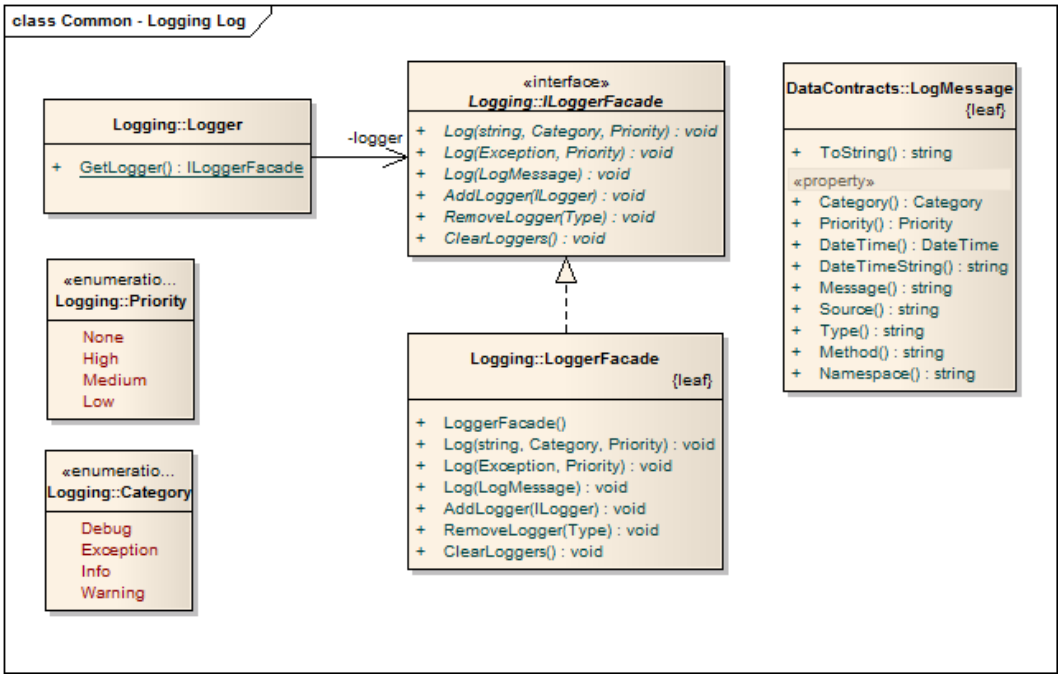


Figure 11: Common - Logging

Every logger target is derived from *LoggerBase* class, which implements the *ILogger* interface. The *TextLogger* writes log messages to a defined text file. The *ServiceLogger* is used in the SCT-App to send log messages to the gateway. The *ViewLogger* shows the log messages in a data grid view. The ViewLogger is used in the gateway and in the controller simulator.

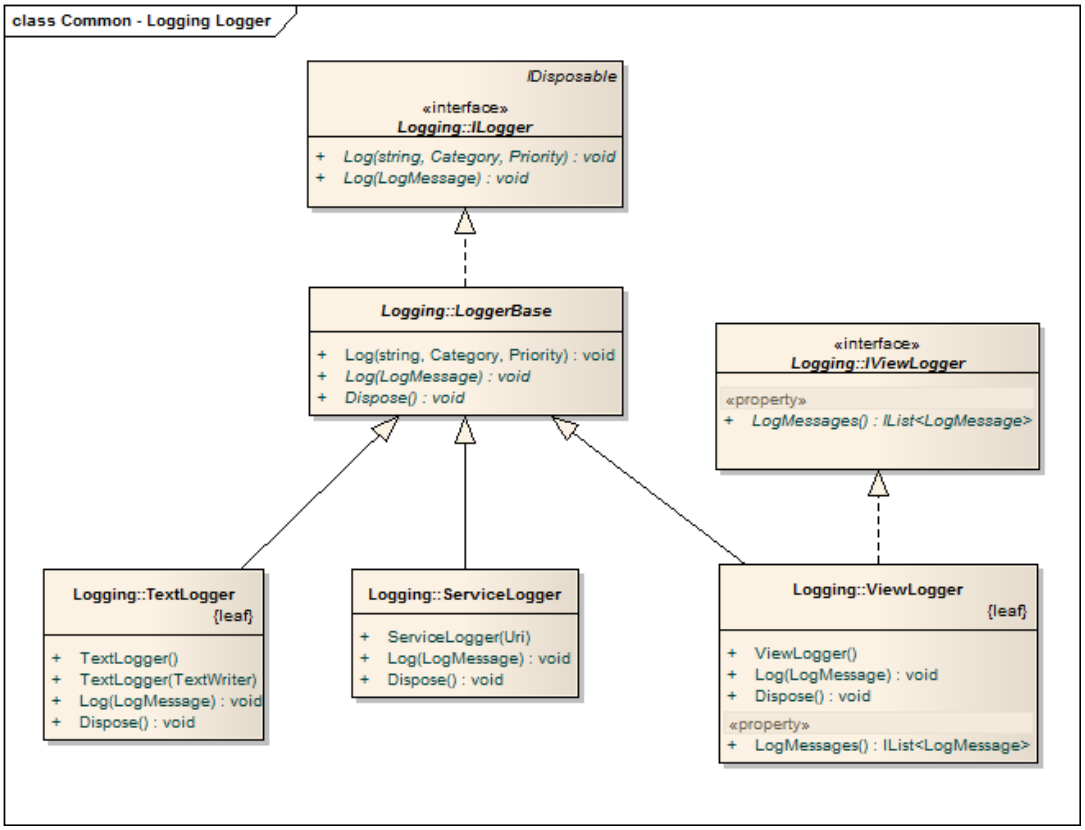


Figure 12: Common - Logging Logger

5.2. Gateway (SCG)

5.2.1. Business Logic - Authentication

The authenthentication interface in business logic *(IAuthenticationManager)* is used in service classes to authenticate a user or validate an authentication token. If the username/password combination is wrong or the authentication token is invalid, an *AuthenticationException* is thrown.

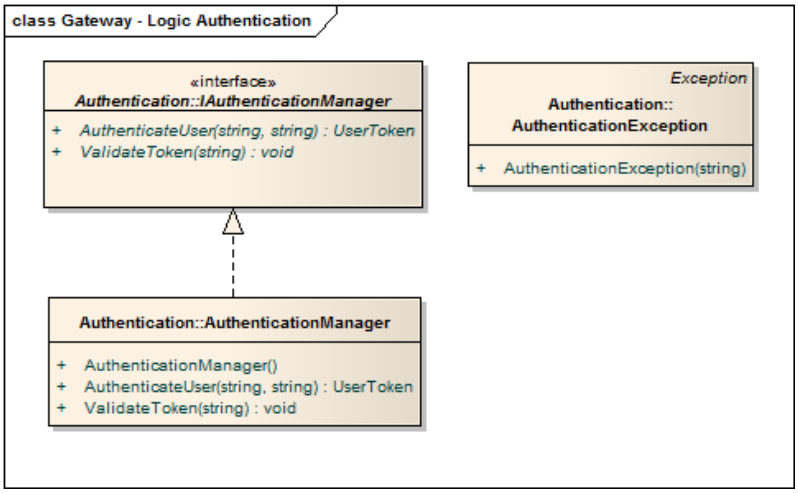


Figure 13: Gateway - Authentication

5.2.2. Business Logic - Engineering Data

The engineering data interface in business logic *(IEngineeringData)* is used in service classes to get the building hierarchy or to map hardware controllers to logical controller. If the hardware controller and the logical controller have a different type, an *IncompatibleControllerException* is thrown.

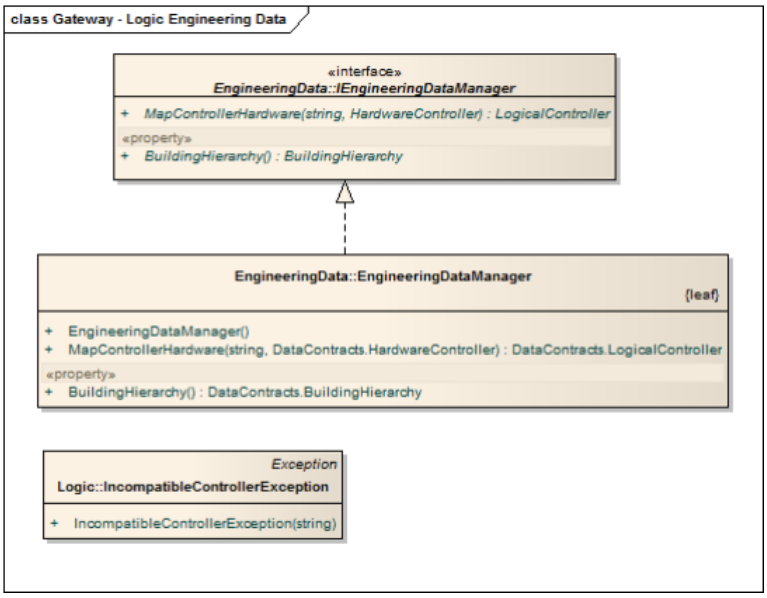


Figure 14: Gateway - Engineering Data

5.2.3. Business Logic - Node Setup

The node setup interface in business logic *(INodeSetupManager)* is used in service classes to perform node setup operations like discover devices, send wink, receive service pin... The *NodeSetupManager* class uses the *ConcurrentDeviceCache* to cache incoming response of discovery requests.

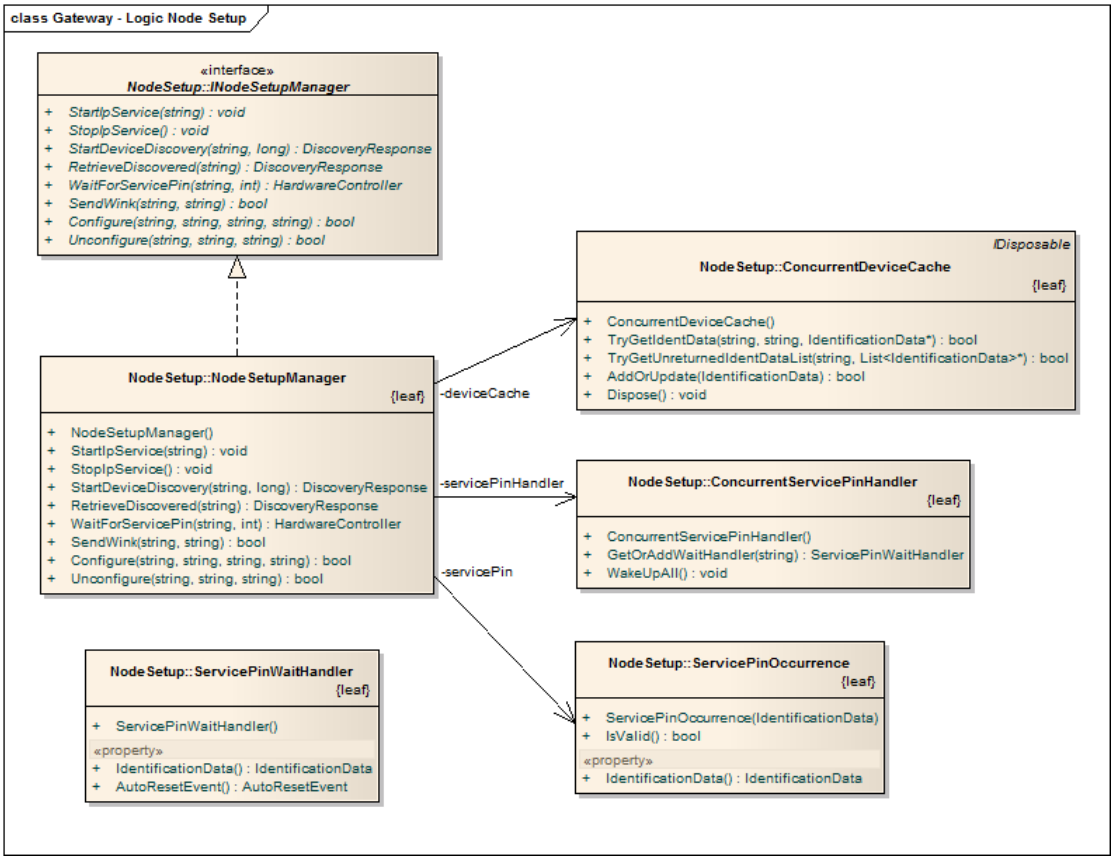


Figure 15: Gateway - Node Setup

5.2.4. Sequence - Start device discovery

The following sequence diagram shows how a service function looks like. Every service function looks similar. It has an overall try-catch block to handle errors in authentication and business functions. The first step is a call to the class *ServiceBase* to disable caching. The next step is a call to the authentication manager to validate the authentication token. If it's invalid an exception is thrown, which is cached by the overall try-catch block. After the validation of the token the real business function is called and then the result returned.

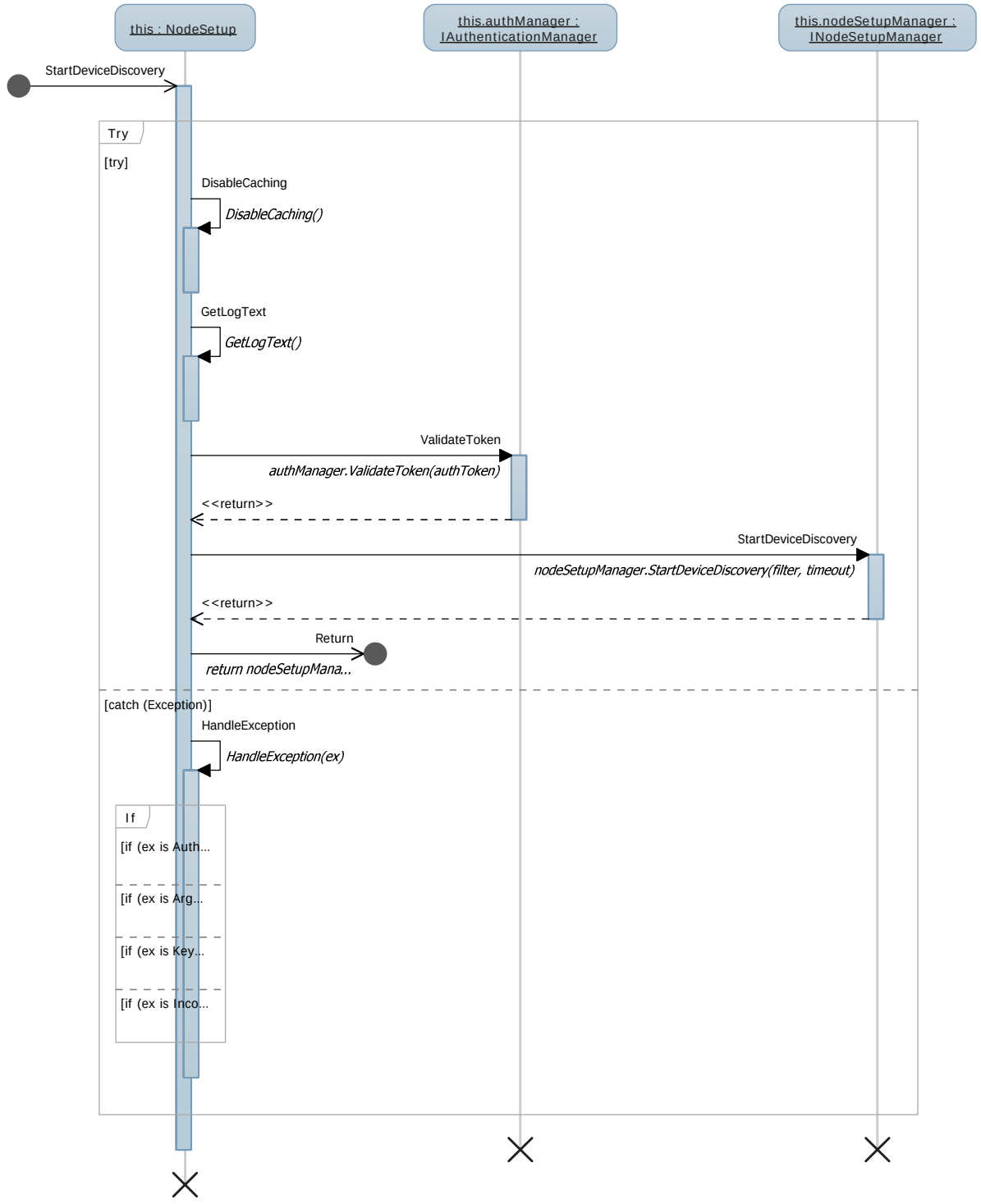


Figure 16: Gateway - Start device discovery

5.2.5. Sequence - Wait for service pin (long polling)

We implemented a long polling to notify the SCT-App when a service pin is received. The SCT-App starts a “wait for service pin” request. The gateway doesn’t response immediately. It holds the connection open till a service pin is received or the timeout is reached. Internally there is a *ConcurrentServicePinHandler* which wakes up every *ServicePinWaitHandler* when a service pin occurs.

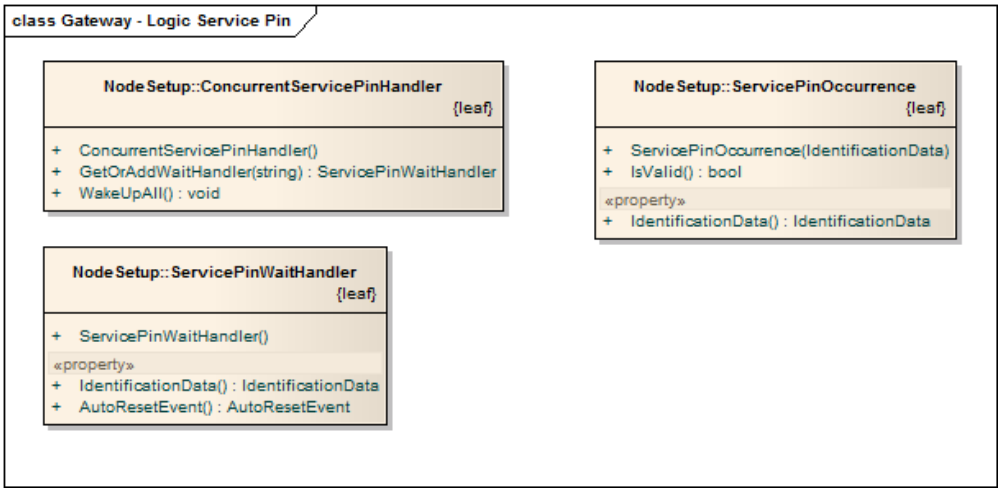


Figure 17: Gateway - Service Pin

The following sequence diagram shows how the long polling to wait for service pins is implemented. It is a simplified diagram for better understanding and some lock points and if statements are removed.

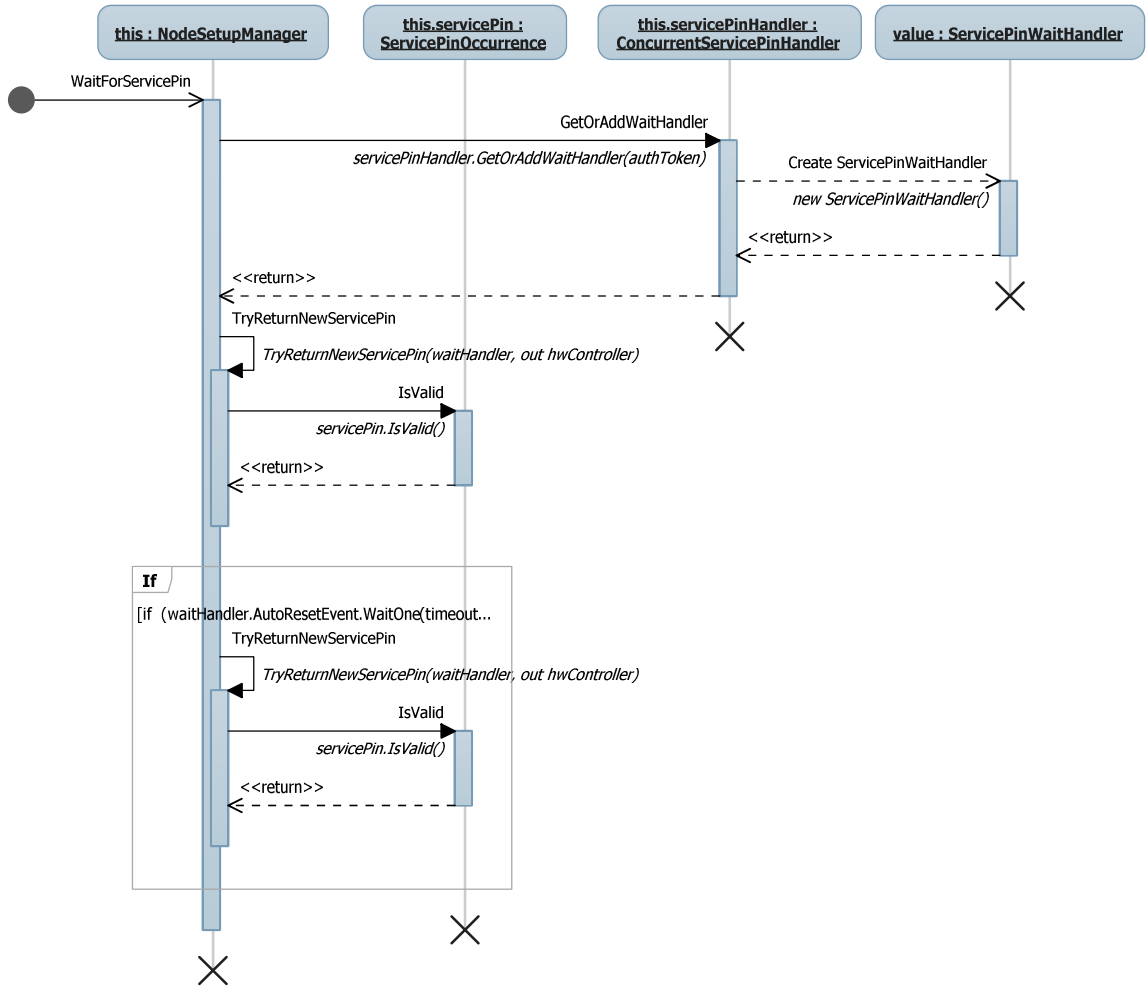


Figure 18: Gateway - Wait for service pin

5.3. WP7 Client (SCT)

5.3.1. Service Layer

The interfaces in the following class diagram describe the service layer methods.

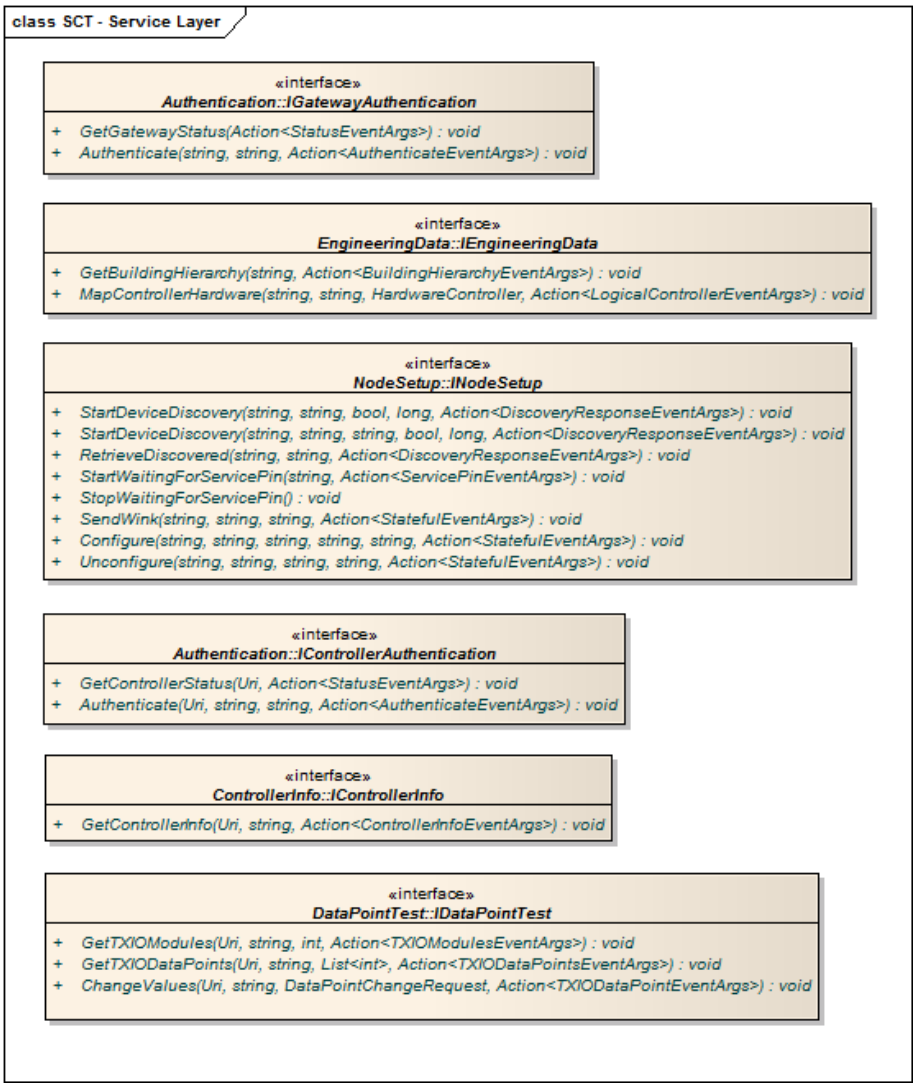


Figure 19: SCT - Service Layer

5.3.2. Service Event Arguments

Near every service function has its own event arguments to return the asked data via callback method.

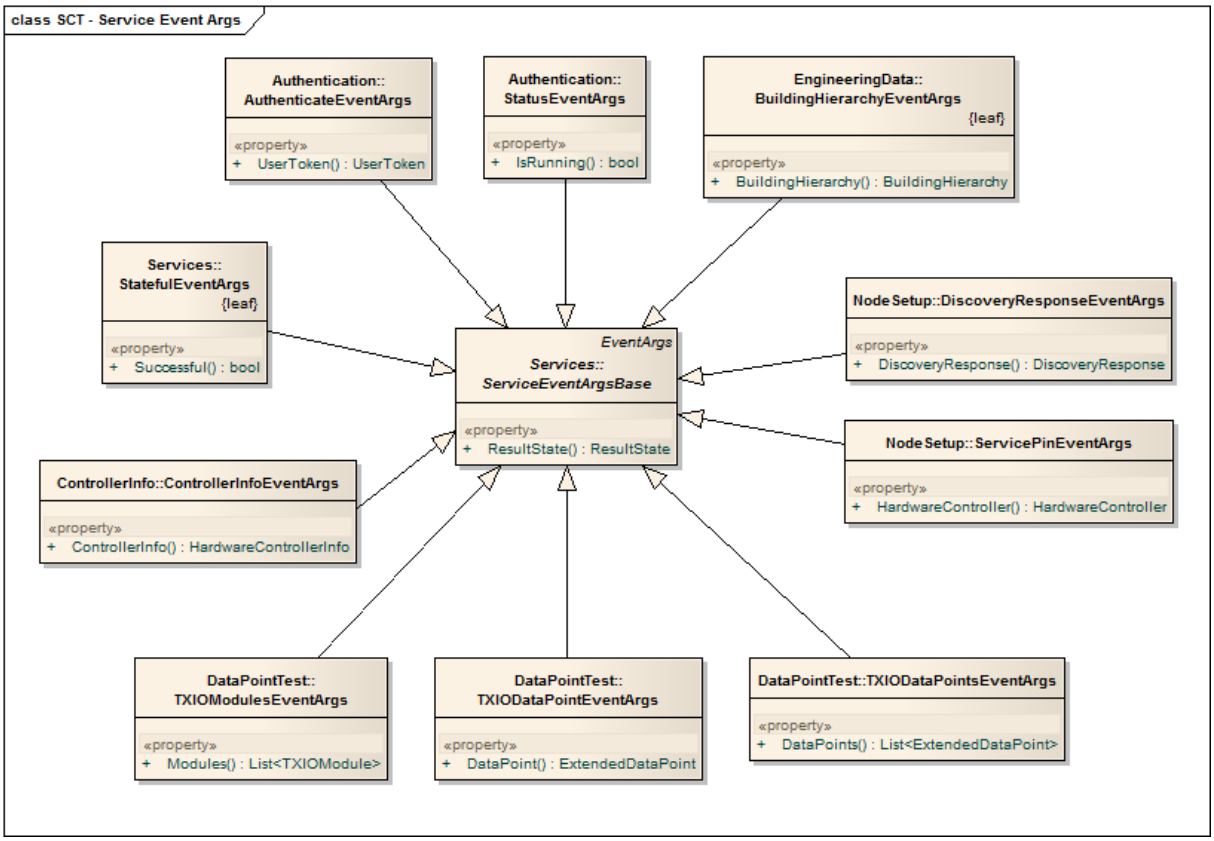


Figure 20: SCT - Service event arguments

5.3.3. Web Service Helper Classes

We implemented some generic helper classes to make the call of REST service functions easier and to have less duplicated code.

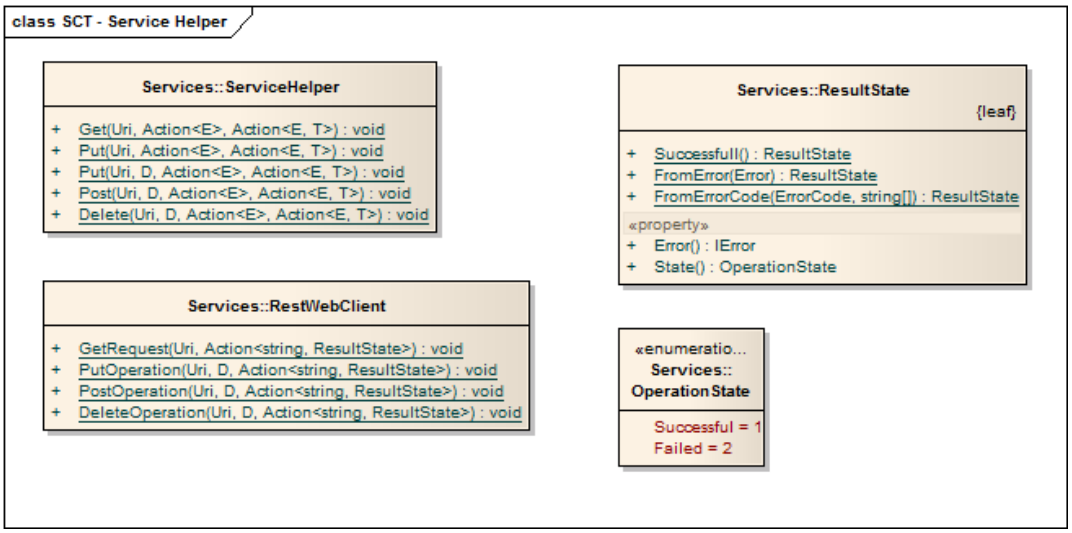


Figure 21: SCT - Web Service Helper Classes

6. User Interface Design

Please check the 'Technical Report' document for a detailed description of the user interface concepts and implementation. This document just provides an additional Page Navigation Map, which shows all pages including the navigation paths in-between.

6.1. Page Navigation Map

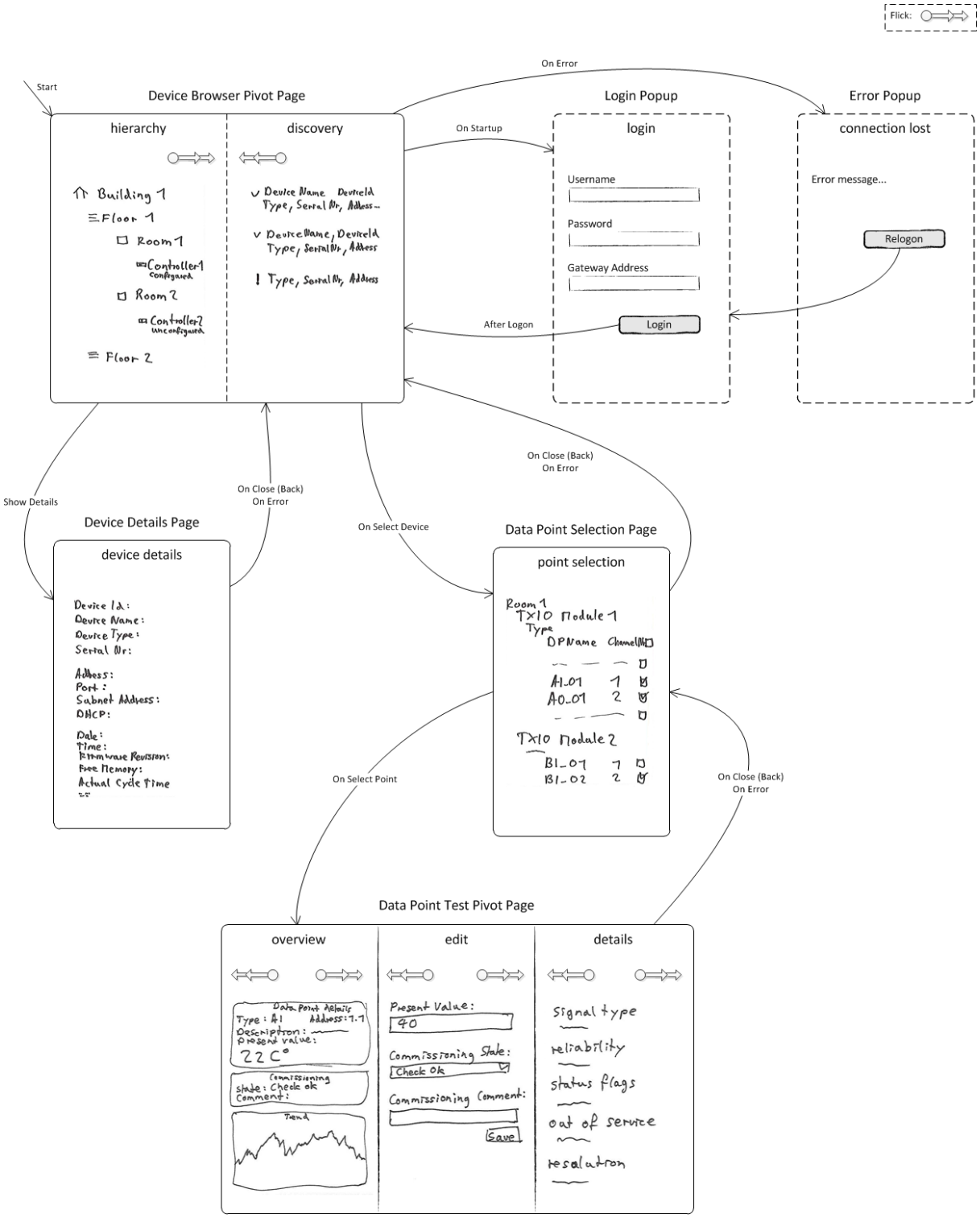


Figure 22: Page Navigation Map

Test Documentation
Alpha Release

SMART COMMISSIONING

Windows Phone 7

Bachelorarbeit:	Mobile Commissioning Tool for Room Automation
Autoren:	Kaspar Fenner Reto Schneebeli
Betreuer:	Prof. Hansjörg Huser
Projektpartner:	Siemens Schweiz AG Building Technologies, Zug
Experte:	Stefan Zettel, Ascentiv AG, Zürich

0. Document Information

0.1. Revision History

Date	Revision	Remarks	Author
18.11.2010	0.1	Initial document creation.	Reto Schneebeli
19.11.2010	0.2	Test cases for UC01, UC02, UC03, UC06 and UC11 added.	Reto Schneebeli
20.11.2010	0.3	Test cases for UC04, UC09 and UC10 added	Reto Schneebeli
20.11.2010	0.4	Test execution “Alpha Release”.	Reto Schneebeli
22.12.2010	1.0	Finalized	Reto Schneebeli

0.3. Table of Contents

0. Document Information 2
0.1. Revision History 2
0.3. Table of Contents 3
1. Requirements 4
2. Preparation 4
3. Information 4
4. Test execution 4
4.1. UC01: Start web service on gateway 4
4.2. UC02: Connect gateway 5
4.3. UC03: Browse RA controllers 5
4.4. UC04: Map RA controller to corresponding hardware 6
4.5. UC05: Remove controller - hardware mapping (optional) 6
4.6. UC06: Search RA controllers with filter 6
4.7. UC07: Configure RA controller (optional) 7
4.8. UC08: Unconfigure RA controller (optional) 7
4.9. UC09: Send wink to RA controller 7
4.10. UC10: Receive service pin 7
4.11. UC11: Show controller details 7
4.12. UC12: Automatically search corresponding hardware (optional) 8
4.13. UC13: Download Device- and Application Configuration (optional) 8
4.14. UC14: Browse data points of an RA controller (focus on TXIO) 8
4.15. UC15: Perform wiring test for a selection of points (focus on TXIO) 8
4.16. UC16: Set commissioning state 8
4.17. UC17: Add commissioning comment 8
4.18. UC18: Release all commanded objects (optional) 8
4.19. UC19: Generate commissioning report (optional) 8
4.20. UC20: Show alarm (optional) 8
4.21. UC21: Confirm alarm (optional) 8
5. Overview 9
6. Improvements 9
6.1. Known Limitations 9

1. Requirements

- Notebook with wireless connection and the SCG *Smart Commissioning Gateway* installed.
- Notebook with connection to the SCG and the CST *Controller Simulation Tool* installed.

2. Preparation

1. Start *Controller Simulation Tool*.
2. Start *Smart Commissioning Gateway*.
3. Install the *Smart Commissioning Tool* on the Windows Phone 7.

3. Information

Date	20.11.2010
SVN revision number	315
Tester	Reto Schneebeli
Hardware	Notebook (Core 2 Duo 2GHz, 4GB Memory)
Operating system	Windows 7 Professional, 64 Bit
WP7 version	7.0.7004.0

4. Test execution

We used the following common gestures in test descriptions:

Tap	A tap is a single, brief touch on the screen within a bounded area and back up again.
Flick	A flick is a single finger down moved rapidly in any direction and ends with the finger up.
Touch and hold	Touch and hold is a single finger down within a bounded area for a defined period of time.

4.1. UC01: Start web service on gateway

Test 1.1	Start the web services on the gateway.	ok
Description	Expectation	
1. Choose the RA controller network. 2. Press the “Start Gateway” button.	1. Every network interface on the notebook is listed and it’s possible to select one. The last used is preselected. 2. “Start Gateway” button is disabled and “Stop Gateway” button is enabled. No error message appears.	
Error, Aberration		
Comment		

Test 1.2	Stop the web services on the gateway.	ok
Precondition	Test 1.1: Start the web services on the gateway.	
Description	Expectation	

1. Press the “Stop Gateway” button.	1. “Stop Gateway” button is disabled and “Start Gateway” button is enabled. No error message appears.
Error, Aberration	
Comment	

4.2. UC02: Connect gateway

Test 2.1	Connect with gateway and login.	error
Precondition	Test 1.1: Start web service on gateway.	
Description	Expectation	
1. Start SCT on windows phone 7. 2. Enter User name, Password and Gateway Address. 3. Press the Login button	1. Login page is displayed. 2. Login button is enabled. 3. Login button is disabled and after less than 10 seconds the building hierarchy is shown.	
Error, Aberration	The Login button is not disabled after a click.	
Comment	A progress indication (blue points) would be nice.	

Test 2.2	Login data saved.	ok
Precondition	Test 2.1: Connect to gateway and login.	
Description	Expectation	
1. Close running SCT. 2. Start SCT on windows phone 7.	1. No error message or delay. 2. Login page is displayed and user name, password and gateway address is preset from last use.	
Error, Aberration		
Comment		

Test 2.3	Try to connect with wrong login information.	ok
Precondition		
Description	Expectation	
1. Start SCT on windows phone 7. 2. Enter a wrong password or user name. 3. Enter a wrong gateway address.	1. Login page is displayed. 2. An understandable error message will appear. 3. An understandable error message will appear.	
Error, Aberration		
Comment	The timeout when a wrong gateway address was entered is too long (40 seconds).	

4.3. UC03: Browse RA controllers

Test 3.1	Browse through the building hierarchy.	ok

Precondition	Test 2.1: Connect with gateway and login.
Description	**Expectation**
1. Precondition: Connect with gateway and login. 2. Tap on a building or room.	1. Building hierarchy view is displayed. It shows the buildings, the floors, the rooms and the controllers. A controller which controls more than one room is displayed more than once. 2. The element expands or collapses its child elements.
Error, Aberration	
Comment	An animation on expand and collapse would be nice.

4.4. UC04: Map RA controller to corresponding hardware

Test 4.1	Map RA controller to corresponding hardware in building hierarchy view	-
Precondition	Test 3.1: Browse through the building hierarchy.	
Description	**Expectation**	
1. Touch and hold on an unmapped (red) controller in building hierarchy view. 2. Press “map to hardware device”. 3. Touch and hold on a controller in device discovery view. 4. Press “accept mapping”	1. The context menu is displayed. 2. The device discovery view is displayed. 3. The context menu is displayed. 4. The building hierarchy view is displayed and the status of the controller is mapped (green).	
Error, Aberration		
Comment	Not yet implemented.	

4.5. UC05: Remove controller - hardware mapping (optional)

Will not be implemented.

4.6. UC06: Search RA controllers with filter

Test 6.1	Switch between building hierarchy and discovery view.	ok
Precondition	Test 3.1: Browse through the building hierarchy.	
Description	**Expectation**	
1. Flick the building hierarchy view away. 2. Flick the device discovery view away. 3. Press on the discovery text in title.	1. The device discovery view is displayed. 2. The building hierarchy view is displayed. 3. The device discovery view is displayed.	
Error, Aberration		
Comment		

Test 6.2	Search and display RA controllers.	error
Precondition	Test 6.1: Switch between building hierarchy and discovery view.	
Description	**Expectation**	

1. Enter the model type of a controller in the network and press the Search button. 2. Clear the model type filter and press the Search button. 3. Wait 10 seconds.	1. The Search button is disabled and after a short delay the first controllers are displayed. 2. The Search button is disabled and after a short delay the first controllers are displayed. 3. Every controller in the network is displayed.
Error, Aberration	The Search button is not disabled after a click.
Comment	

4.7. UC07: Configure RA controller (optional)

Will not be implemented.

4.8. UC08: Unconfigure RA controller (optional)

Will not be implemented.

4.9. UC09: Send wink to RA controller

Test 9.1	Send a flash LED operation to an RA controller	-
Precondition	Test 3.1: Browse through the building hierarchy.	
Description	Expectation	
1. Touch and hold on an unmapped (red) controller in building hierarchy view. 2. Press “flash LED”.	1. The context menu is displayed. 2. The LED on the hardware device is flashed.	
Error, Aberration		
Comment	Not yet implemented.	

4.10. UC10: Receive service pin

Test 10.1	Receive service pin from a hardware device	-
Precondition	Test 6.2: Search and display RA controllers.	
Description	Expectation	
1. Precondition: Search and display RA controllers. 2. Press the “Service Pin” button in the CST *Controller Simulation Tool* on a controller.	1. Every controller in the network is displayed but no controller is highlighted. 2. The controller is highlighted.	
Error, Aberration		
Comment	Not yet implemented.	

4.11. UC11: Show controller details

Test 11.1	Show controller details in building hierarchy view	ok
Precondition	Test 3.1: Browse through the building hierarchy.	
Description	Expectation	

1. Touch and hold on a controller in building hierarchy view. 2. Press “show details”.	1. The context menu is displayed. 2. The device details page of the selected controller is displayed.
Error, Aberration	
Comment	

Test 11.2	Show controller details in device discovery view	ok
Precondition	Test 6.1: Switch between building hierarchy and discovery view.	
Description	Expectation	
1. Touch and hold on a controller in device discovery view. 2. Press “show details”.	1. The context menu is displayed. 2. The device details page of the selected controller is displayed.	
Error, Aberration		
Comment		

4.12. UC12: Automatically search corresponding hardware (optional)

Will not be implemented.

4.13. UC13: Download Device- and Application Configuration (optional)

Will not be implemented.

4.14. UC14: Browse data points of an RA controller (focus on TXIO)

4.15. UC15: Perform wiring test for a selection of points (focus on TXIO)

4.16. UC16: Set commissioning state

4.17. UC17: Add commissioning comment

4.18. UC18: Release all commanded objects (optional)

Will not be implemented.

4.19. UC19: Generate commissioning report (optional)

Will not be implemented.

4.20. UC20: Show alarm (optional)

Will not be implemented.

4.21. UC21: Confirm alarm (optional)

Will not be implemented.

5. Overview

Use Case	*Implement.*	*Error/Aberration*	*Status*
UC01: Start web service on gateway	yes		ok
UC02: Connect gateway	yes	• The Login button is not disabled after a click. • A progress indication (blue points) would be nice. • The timeout when a wrong gateway address was entered is too long (40 seconds).	error
UC03: Browse RA controllers	yes	An animation on expand and collapse would be nice.	ok
UC04: Map RA controller to corresponding hardware	no	Not yet implemented.	-
UC05: Remove controller - hardware mapping	no, optional		-
UC06: Search RA controllers with filter	yes	The Search button is not disabled after a click.	error
UC07: Configure RA controller	no, optional		-
UC08: Unconfigure RA controller	no, optional		-
UC09: Send wink to RA controller	no	Not yet implemented.	-
UC10: Receive service pin	no	Not yet implemented.	-
UC11: Show controller details	yes		ok
UC12: Automatically search corresponding hardware	no, optional		
UC13: Download Device- and Application Configuration	no, optional		
UC14: Browse data points of an RA controller			
UC15: Perform wiring test for a selection of points			
UC16: Set commissioning state			
UC17: Add commissioning comment			
UC18: Release all commanded objects	optional		
UC19: Generate commissioning report	optional		
UC20: Show alarm	optional		
UC21: Confirm alarm	optional		

6. Improvements

6.1. Known Limitations

- It is possible to start the SCG more than once but then the start of the web service would fail.
- The gateway address cannot be auto discovered.

Test Documentation
Beta Release

SMART
COMMISSIONING

Windows Phone 7

Bachelorarbeit:	Mobile Commissioning Tool for Room Automation
Autoren:	Kaspar Fenner Reto Schneebeli
Betreuer:	Prof. Hansjörg Huser
Projektpartner:	Siemens Schweiz AG Building Technologies, Zug
Experte:	Stefan Zettel, Ascentiv AG, Zürich

0. Document Information

0.1. Revision History

Date	Revision	Remarks	Author
18.11.2010	0.1	Initial document creation.	Reto Schneebeli
19.11.2010	0.2	Test cases for UC01, UC02, UC03, UC06 and UC11 added.	Reto Schneebeli
20.11.2010	0.3	Test cases for UC04, UC09 and UC10 added	Reto Schneebeli
01.12.2010	0.4	Test cases for UC14, UC15, UC16 and UC17 added	Reto Schneebeli
01.12.2010	0.5	Test execution “Beta Release”.	Reto Schneebeli
22.12.2010	1.0	Finalized	Reto Schneebeli

0.2. Table of Contents

0. Document Information 2
0.1. Revision History 2
0.2. Table of Contents 3
1. Requirements 4
2. Preparation 4
3. Information 4
4. Test execution 4
4.1. UC01: Start web service on gateway 4
4.2. UC02: Connect gateway 5
4.3. UC03: Browse RA controllers 6
4.4. UC04: Map RA controller to corresponding hardware 6
4.5. UC05: Remove controller - hardware mapping (optional) 6
4.6. UC06: Search RA controllers with filter 6
4.7. UC07: Configure RA controller (optional) 7
4.8. UC08: Unconfigure RA controller (optional) 7
4.9. UC09: Send wink to RA controller 7
4.10. UC10: Receive service pin 7
4.11. UC11: Show controller details 7
4.12. UC12: Automatically search corresponding hardware (optional) 8
4.13. UC13: Download Device- and Application Configuration (optional) 8
4.14. UC14: Browse data points of an RA controller (focus on TXIO) 8
4.15. UC15: Perform wiring test for a selection of points (focus on TXIO) 9
4.16. UC16: Set commissioning state 10
4.17. UC17: Add commissioning comment 10
4.18. UC18: Release all commanded objects (optional) 10
4.19. UC19: Generate commissioning report (optional) 10
4.20. UC20: Show alarm (optional) 10
4.21. UC21: Confirm alarm (optional) 10
5. Overview 11
6. Improvements 12
6.1. Known Limitations 12

1. Requirements

- Notebook with wireless connection and the SCG *Smart Commissioning Gateway* installed.
- Notebook with connection to the SCG and the CST *Controller Simulation Tool* installed.

2. Preparation

1. Start *Controller Simulation Tool*.
2. Choose the RA controller network on the *Controller Simulation Tool.*
3. Start the DICP service and the web services on the *Controller Simulation Tool.*
4. Start *Smart Commissioning Gateway.*
5. Install the *Smart Commissioning Tool* on the Windows Phone 7.

3. Information

Date	01.12.2010
SVN revision number	382
Tester	Reto Schneebeli
Hardware	Notebook (Core 2 Duo 2GHz, 4GB Memory)
Operating system	Windows 7 Professional, 64 Bit
WP7 version	7.0.7004.0

4. Test execution

We used the following common gestures in test descriptions:

Tap	A tap is a single, brief touch on the screen within a bounded area and back up again.
Flick	A flick is a single finger down moved rapidly in any direction and ends with the finger up.
Touch and hold	Touch and hold is a single finger down within a bounded area for a defined period of time.

4.1. UC01: Start web service on gateway

Test 1.1	Start the web services on the gateway.	ok
Description	Expectation	
1. Choose the RA controller network. 2. Press the “Start Gateway” button.	1. Every network interface on the notebook is listed and it’s possible to select one. The last used is preselected. 2. “Start Gateway” button is disabled and “Stop Gateway” button is enabled. No error message appears.	
Error, Aberration		
Comment		

Test 1.2	Stop the web services on the gateway.	ok
Precondition	Test 1.1: Start the web services on the gateway.	
Description	Expectation	

1. Press the “Stop Gateway” button.	1. “Stop Gateway” button is disabled and “Start Gateway” button is enabled. No error message appears.
Error, Aberration	
Comment	

4.2. UC02: Connect gateway

Test 2.1	Connect with gateway and login.	ok
Precondition	Test 1.1: Start web service on gateway.	
Description	Expectation	
1. Start SCT on windows phone 7. 2. Enter User name, Password and Gateway Address. 3. Press the Login button	1. Login page is displayed. 2. Login button is enabled. 3. Login button is disabled and after less than 10 seconds the building hierarchy is shown.	
Error, Aberration		
Comment	A progress indication (blue points) would be nice.	

Test 2.2	Login data saved.	ok
Precondition	Test 2.1: Connect to gateway and login.	
Description	Expectation	
1. Close running SCT. 2. Start SCT on windows phone 7.	1. No error message or delay. 2. Login page is displayed and user name, password and gateway address is preset from last use.	
Error, Aberration		
Comment		

Test 2.3	Try to connect with wrong login information.	ok
Precondition		
Description	Expectation	
1. Start SCT on windows phone 7. 2. Enter a wrong password or user name and press the Login button. 3. Enter a wrong gateway address and press the Login button.	1. Login page is displayed. 2. An understandable error message will appear. 3. An understandable error message will appear.	
Error, Aberration		
Comment	The timeout when a wrong gateway address was entered is very long (40 seconds). An abort button would be nice.	

4.3. UC03: Browse RA controllers

Test 3.1	Browse through the building hierarchy.	ok
Precondition	Test 2.1: Connect with gateway and login.	

Description	Expectation
1. Precondition: Connect with gateway and login. 2. Tap on a building or room.	1. Building hierarchy view is displayed. It shows the buildings, the floors, the rooms and the controllers. A controller which controls more than one room is displayed more than once. 2. The element expands or collapses its child elements.

Error, Aberration	
Comment	

4.4. UC04: Map RA controller to corresponding hardware

Test 4.1	Map RA controller to corresponding hardware in building hierarchy view	-
Precondition	Test 3.1: Browse through the building hierarchy.	

Description	Expectation
1. Touch and hold on an unmapped (red) controller in building hierarchy view. 2. Press “map to hardware device”. 3. Touch and hold on a controller in device discovery view. 4. Press “accept mapping”	1. The context menu is displayed. 2. The device discovery view is displayed. 3. The context menu is displayed. 4. The building hierarchy view is displayed and the status of the controller is mapped (green).

Error, Aberration	
Comment	Not yet implemented.

4.5. UC05: Remove controller - hardware mapping (optional)

Will not be implemented.

4.6. UC06: Search RA controllers with filter

Test 6.1	Switch between building hierarchy and discovery view.	ok
Precondition	Test 3.1: Browse through the building hierarchy.	

Description	Expectation
1. Flick the building hierarchy view away. 2. Flick the device discovery view away. 3. Press on the discovery text in title.	1. The device discovery view is displayed. 2. The building hierarchy view is displayed. 3. The device discovery view is displayed.

Error, Aberration	
Comment	

Test 6.2	Search and display RA controllers.	ok

Precondition	Test 6.1: Switch between building hierarchy and discovery view.

Description	Expectation
1. Enter the model type of a controller in the network and press the Search button. 2. Clear the model type filter and press the Search button. 3. Wait 10 seconds.	1. After a short delay the first controllers are displayed. 2. After a short delay the first controllers are displayed. 3. Every controller in the network is displayed.

Error, Aberration	
Comment	

4.7. UC07: Configure RA controller (optional)

Will not be implemented.

4.8. UC08: Unconfigure RA controller (optional)

Will not be implemented.

4.9. UC09: Send wink to RA controller

Test 9.1	Send a flash LED operation to an RA controller	ok
Precondition	Test 6.2: Search and display RA controllers.	

Description	Expectation
1. Touch and hold on a controller in device discovery view. 2. Press “flash LED”.	1. The context menu is displayed. 2. The LED on the hardware device is flashed (in controller simulation tool the hardware device is selected).

Error, Aberration	
Comment	

4.10. UC10: Receive service pin

Test 10.1	Receive service pin from a hardware device	ok
Precondition	Test 6.2: Search and display RA controllers.	

Description	Expectation
1. Precondition: Search and display RA controllers. 2. Press the “Service Pin” button in the CST *Controller Simulation Tool* on a controller.	1. Every controller in the network is displayed but no controller is highlighted. 2. The controller is highlighted.

Error, Aberration	
Comment	

4.11. UC11: Show controller details

Test 11.1	Show controller details in building hierarchy view	ok
Precondition	Test 3.1: Browse through the building hierarchy.	

Description	Expectation
1. Touch and hold on a controller in building hierarchy view. 2. Press “show details”.	1. The context menu is displayed. 2. The device details page of the selected controller is displayed.
Error, Aberration	
Comment	

Test 11.2	Show controller details in device discovery view	ok
Precondition	Test 6.1: Switch between building hierarchy and discovery view.	

Description	Expectation
1. Touch and hold on a controller in device discovery view. 2. Press “show details”.	1. The context menu is displayed. 2. The device details page of the selected controller is displayed.
Error, Aberration	
Comment	

4.12. UC12: Automatically search corresponding hardware (optional)

Will not be implemented.

4.13. UC13: Download Device- and Application Configuration (optional)

Will not be implemented.

4.14. UC14: Browse data points of an RA controller (focus on TXIO)

Test 14.1	Go to point selection page in building hierarchy view	ok
Precondition	Test 3.1: Browse through the building hierarchy.	

Description	Expectation
1. Tab on an unmapped controller (red) in building hierarchy view. 2. Tab on an unconfigured controller (orange) in building hierarchy view. 3. Tab on a configured controller (green) in building hierarchy view.	1. An understandable error message will appear. 2. An understandable error message will appear. 3. The data point selection page of the selected controller is displayed. It shows the modules and the data points.
Error, Aberration	
Comment	

Test 14.2	Go to point selection page in device discovery view	ok
Precondition	Test 6.1: Switch between building hierarchy and discovery view.	

Description	Expectation

1. Tab on an unconfigured controller (orange) in device discovery view. 2. Tab on a configured controller (green) in device discovery view.	1. An understandable error message will appear. 2. The data point selection page of the selected controller is displayed. It shows the modules and the data points.
Error, Aberration	
Comment	

Test 14.3	Browse data points and start point test	error
Precondition	Test 14.2: Go to point selection page in device discovery view.	
Description	Expectation	
1. Tab on a module. 2. Tab on a data point.	1. The module expands or collapses its data points. 2. The point test view of the selected data point is displayed. It shows the overview page of the data point.	
Error, Aberration	It is not possible to expand or collapse modules.	
Comment		

4.15. UC15: Perform wiring test for a selection of points (focus on TXIO)

Test 15.1	Perform wiring test for a input data point	error
Precondition	Test 14.2: Go to point selection page in device discovery view.	
Description	Expectation	
1. Tab on an analog, binary or multistate input data point. 2. Simulate a change for this data point in *Controller Simulation Tool*.	1. The overview page of the data point is displayed with data point details, commissioning information and a trend graph. 2. The new present value is displayed and the trend graph is updated.	
Error, Aberration		
Comment	After tab on a data point the overview page first shows the old data point value and then changes to the new one.	

Test 15.2	Perform wiring test for an output data point	ok
Precondition	Test 14.2: Go to point selection page in device discovery view.	
Description	Expectation	
1. Tab on an analog, binary or multistate output data point. 2. Flick the overview page away. 3. Change the present value (command output).	1. The overview page of the data point is displayed with data point details, commissioning information and a trend graph. 2. The edit page of the data point is displayed. 3. The new present value is displayed in the *Controller Simulation Tool.*	
Error, Aberration		
Comment		

Test 15.3	Show data point details	ok
Precondition	Test 15.2: Perform wiring test for an output data point.	
Description	Expectation	
1. Precondition: Perform wiring test for an output data point. 2. Flick the edit page away.	1. The edit page of the data point is displayed. 2. The details page of the data point is displayed.	
Error, Aberration		
Comment		

4.16. UC16: Set commissioning state

Test 16.1	Show data point details	ok
Precondition	Test 15.2: Perform wiring test for an output data point.	
Description	Expectation	
1. Precondition: Perform wiring test for an output data point. 2. Change the commissioning state and press the Save button.	1. The edit page of the data point is displayed. 2. The new commissioning state is displayed in the *Controller Simulation Tool.*	
Error, Aberration		
Comment		

4.17. UC17: Add commissioning comment

Test 17.1	Show data point details	ok
Precondition	Test 16.1: Perform wiring test for an output data point.	
Description	Expectation	
1. Change the commissioning comment and press the Save button.	1. The new commissioning comment is displayed in the *Controller Simulation Tool.*	
Error, Aberration		
Comment		

4.18. UC18: Release all commanded objects (optional)

Will not be implemented.

4.19. UC19: Generate commissioning report (optional)

Will not be implemented.

4.20. UC20: Show alarm (optional)

Will not be implemented.

4.21. UC21: Confirm alarm (optional)

Will not be implemented.

5. Overview

Use Case	*Implement.*	*Error/Aberration*	*Status*
UC01: Start web service on gateway	yes		ok
UC02: Connect gateway	yes	A progress indication (blue points) would be nice. The timeout when a wrong gateway address was entered is very long (40 seconds). An abort button would be nice.	ok
UC03: Browse RA controllers	yes		ok
UC04: Map RA controller to corresponding hardware	no	Not yet implemented.	-
UC05: Remove controller - hardware mapping	no, optional		-
UC06: Search RA controllers with filter	yes		ok
UC07: Configure RA controller	no, optional		-
UC08: Unconfigure RA controller	no, optional		-
UC09: Send wink to RA controller	yes		ok
UC10: Receive service pin	yes		ok
UC11: Show controller details	yes		ok
UC12: Automatically search corresponding hardware	no, optional		-
UC13: Download Device- and Application Configuration	no, optional		-
UC14: Browse data points of an RA controller	yes	It is not possible to expand or collapse modules.	error
UC15: Perform wiring test for a selection of points	yes	Sometimes, an IndexOutOfRange exception is thrown: (System.Windows.Controls.Item.ItemContainerGenerator.RealizedItemBlock.ContainerAt(Int32 index), Microsoft.Phone.Controls.ListPicker.< OnItemsChanged>) After tab on a data point the overview page still shows the old data point value, before it changes to the new one a “second” later.	error
UC16: Set commissioning state	yes		ok
UC17: Add commissioning comment	yes		ok
UC18: Release all commanded objects	no, optional		-
UC19: Generate commissioning report	no, optional		-
UC20: Show alarm	no, optional		-
UC21: Confirm alarm	no, optional		-

6. Improvements

6.1. Known Limitations

- It is possible to start the SCG more than once but then the start of the web service would fail.
- The gateway address cannot be auto discovered.
- The mapped controllers in engineering data are not updated with new attributes (address) of hardware controllers.

Test Documentation
Final Release

SMART
COMMISSIONING

Windows Phone 7

Bachelorarbeit:	Mobile Commissioning Tool for Room Automation
Autoren:	Kaspar Fenner Reto Schneebeli
Betreuer:	Prof. Hansjörg Huser
Projektpartner:	Siemens Schweiz AG Building Technologies, Zug
Experte:	Stefan Zettel, Ascentiv AG, Zürich

0. Document Information

0.1. Revision History

Date	Revision	Remarks	Author
18.11.2010	0.1	Initial document creation.	Reto Schneebeli
19.11.2010	0.2	Test cases for UC01, UC02, UC03, UC06 and UC11 added.	Reto Schneebeli
20.11.2010	0.3	Test cases for UC04, UC09 and UC10 added	Reto Schneebeli
01.12.2010	0.4	Test cases for UC14, UC15, UC16 and UC17 added	Reto Schneebeli
13.12.2010	0.5	Test execution “Final Release”.	Reto Schneebeli
22.12.2010	1.0	Finalized	Reto Schneebeli

0.2. Table of Contents

0. Document Information 2
0.1. Revision History 2
0.2. Table of Contents 3
1. Requirements 4
2. Preparation 4
3. Information 4
4. Test execution 4
4.1. UC01: Start web service on gateway 4
4.2. UC02: Connect gateway 5
4.3. UC03: Browse RA controllers 6
4.4. UC04: Map RA controller to corresponding hardware 6
4.5. UC05: Remove controller - hardware mapping (optional) 6
4.6. UC06: Search RA controllers with filter 6
4.7. UC07: Configure RA controller (optional) 7
4.8. UC08: Unconfigure RA controller (optional) 7
4.9. UC09: Send wink to RA controller 7
4.10. UC10: Receive service pin 7
4.11. UC11: Show controller details 7
4.12. UC12: Automatically search corresponding hardware (optional) 8
4.13. UC13: Download Device- and Application Configuration (optional) 8
4.14. UC14: Browse data points of an RA controller (focus on TXIO) 8
4.15. UC15: Perform wiring test for a selection of points (focus on TXIO) 9
4.16. UC16: Set commissioning state 10
4.17. UC17: Add commissioning comment 10
4.18. UC18: Release all commanded objects (optional) 10
4.19. UC19: Generate commissioning report (optional) 10
4.20. UC20: Show alarm (optional) 10
4.21. UC21: Confirm alarm (optional) 10
5. Overview 11
6. Improvements 11
6.1. Known Limitations 11

1. Requirements

- Notebook with wireless connection and the SCG *Smart Commissioning Gateway* installed.
- Notebook with connection to the SCG and the CST *Controller Simulation Tool* installed.

2. Preparation

1. Start *Controller Simulation Tool*.
2. Choose the RA controller network on the *Controller Simulation Tool.*
3. Start the DICP service and the web services on the *Controller Simulation Tool.*
4. Start *Smart Commissioning Gateway.*
5. Install the *Smart Commissioning Tool* on the Windows Phone 7.

3. Information

Date	01.12.2010
SVN revision number	382
Tester	Reto Schneebeli
Hardware	Notebook (Core 2 Duo 2GHz, 4GB Memory)
Operating system	Windows 7 Professional, 64 Bit
WP7 version	7.0.7004.0

4. Test execution

We used the following common gestures in test descriptions:

Tap	A tap is a single, brief touch on the screen within a bounded area and back up again.
Flick	A flick is a single finger down moved rapidly in any direction and ends with the finger up.
Touch and hold	Touch and hold is a single finger down within a bounded area for a defined period of time.

4.1. UC01: Start web service on gateway

Test 1.1	Start the web services on the gateway.	ok
Description	Expectation	
1. Choose the RA controller network. 2. Press the “Start Gateway” button.	1. Every network interface on the notebook is listed and it’s possible to select one. The last used is preselected. 2. “Start Gateway” button is disabled and “Stop Gateway” button is enabled. No error message appears.	
Error, Aberration		
Comment		

Test 1.2	Stop the web services on the gateway.	ok
Precondition	Test 1.1: Start the web services on the gateway.	
Description	Expectation	

1. Press the “Stop Gateway” button.	1. “Stop Gateway” button is disabled and “Start Gateway” button is enabled. No error message appears.
Error, Aberration	
Comment	

4.2. UC02: Connect gateway

Test 2.1	Connect with gateway and login.	ok
Precondition	Test 1.1: Start web service on gateway.	
Description	Expectation	
1. Start SCT on windows phone 7. 2. Enter User name, Password and Gateway Address. 3. Press the Login button	1. Login page is displayed. 2. Login button is enabled. 3. Login button is disabled and after less than 10 seconds the building hierarchy is shown.	
Error, Aberration		
Comment	A progress indication (blue points) would be nice.	

Test 2.2	Login data saved.	ok
Precondition	Test 2.1: Connect to gateway and login.	
Description	Expectation	
1. Close running SCT. 2. Start SCT on windows phone 7.	1. No error message or delay. 2. Login page is displayed and user name, password and gateway address is preset from last use.	
Error, Aberration		
Comment		

Test 2.3	Try to connect with wrong login information.	ok
Precondition		
Description	Expectation	
1. Start SCT on windows phone 7. 2. Enter a wrong password or user name and press the Login button. 3. Enter a wrong gateway address and press the Login button.	1. Login page is displayed. 2. An understandable error message will appear. 3. An understandable error message will appear.	
Error, Aberration		
Comment	The timeout when a wrong gateway address was entered is very long (40 seconds). An abort button would be nice.	

4.3. UC03: Browse RA controllers

Test 3.1	Browse through the building hierarchy.	ok
Precondition	Test 2.1: Connect with gateway and login.	
Description	Expectation	
1. Precondition: Connect with gateway and login. 2. Tap on a building or room.	1. Building hierarchy view is displayed. It shows the buildings, the floors, the rooms and the controllers. A controller which controls more than one room is displayed more than once. 2. The element expands or collapses its child elements.	
Error, Aberration		
Comment		

4.4. UC04: Map RA controller to corresponding hardware

Test 4.1	Map RA controller to corresponding hardware in building hierarchy view	ok
Precondition	Test 3.1: Browse through the building hierarchy.	
Description	Expectation	
1. Touch and hold on an unmapped (red) controller in building hierarchy view. 2. Press “map to hardware device”. 3. Touch and hold on a controller in device discovery view. 4. Press “accept mapping”	1. The context menu is displayed. 2. The device discovery view is displayed. 3. The context menu is displayed. 4. The building hierarchy view is displayed and the status of the controller is mapped (green).	
Error, Aberration		
Comment		

4.5. UC05: Remove controller - hardware mapping (optional)

Will not be implemented.

4.6. UC06: Search RA controllers with filter

Test 6.1	Switch between building hierarchy and discovery view.	ok
Precondition	Test 3.1: Browse through the building hierarchy.	
Description	Expectation	
1. Flick the building hierarchy view away. 2. Flick the device discovery view away. 3. Press on the discovery text in title.	1. The device discovery view is displayed. 2. The building hierarchy view is displayed. 3. The device discovery view is displayed.	
Error, Aberration		
Comment		

Test 6.2	Search and display RA controllers.	ok

Precondition	Test 6.1: Switch between building hierarchy and discovery view.
Description	**Expectation**
1. Enter the model type of a controller in the network and press the Search button. 2. Clear the model type filter and press the Search button. 3. Wait 10 seconds.	1. After a short delay the first controllers are displayed. 2. After a short delay the first controllers are displayed. 3. Every controller in the network is displayed.
Error, Aberration	
Comment	

4.7. UC07: Configure RA controller (optional)

Will not be implemented.

4.8. UC08: Unconfigure RA controller (optional)

Will not be implemented.

4.9. UC09: Send wink to RA controller

Test 9.1	Send a flash LED operation to an RA controller	ok
Precondition	Test 6.2: Search and display RA controllers.	
Description	**Expectation**	
1. Touch and hold on a controller in device discovery view. 2. Press “flash LED”.	1. The context menu is displayed. 2. The LED on the hardware device is flashed (in controller simulation tool the hardware device is selected).	
Error, Aberration		
Comment		

4.10. UC10: Receive service pin

Test 10.1	Receive service pin from a hardware device	ok
Precondition	Test 6.2: Search and display RA controllers.	
Description	**Expectation**	
1. Precondition: Search and display RA controllers. 2. Press the “Service Pin” button in the CST *Controller Simulation Tool* on a controller.	1. Every controller in the network is displayed but no controller is highlighted. 2. The controller is highlighted.	
Error, Aberration		
Comment		

4.11. UC11: Show controller details

Test 11.1	Show controller details in building hierarchy view	ok
Precondition	Test 3.1: Browse through the building hierarchy.	

Description	Expectation
1. Touch and hold on a controller in building hierarchy view. 2. Press “show details”.	1. The context menu is displayed. 2. The device details page of the selected controller is displayed.
Error, Aberration	
Comment	

Test 11.2	Show controller details in device discovery view	ok
Precondition	Test 6.1: Switch between building hierarchy and discovery view.	

Description	Expectation
1. Touch and hold on a controller in device discovery view. 2. Press “show details”.	1. The context menu is displayed. 2. The device details page of the selected controller is displayed.
Error, Aberration	
Comment	

4.12. UC12: Automatically search corresponding hardware (optional)

Will not be implemented.

4.13. UC13: Download Device- and Application Configuration (optional)

Will not be implemented.

4.14. UC14: Browse data points of an RA controller (focus on TXIO)

Test 14.1	Go to point selection page in building hierarchy view	ok
Precondition	Test 3.1: Browse through the building hierarchy.	

Description	Expectation
1. Tab on an unmapped controller (red) in building hierarchy view. 2. Tab on an unconfigured controller (orange) in building hierarchy view. 3. Tab on a configured controller (green) in building hierarchy view.	1. An understandable error message will appear. 2. An understandable error message will appear. 3. The data point selection page of the selected controller is displayed. It shows the modules and the data points.
Error, Aberration	
Comment	

Test 14.2	Go to point selection page in device discovery view	ok
Precondition	Test 6.1: Switch between building hierarchy and discovery view.	

Description	Expectation

1. Tab on an unconfigured controller (orange) in device discovery view. 2. Tab on a configured controller (green) in device discovery view.	1. An understandable error message will appear. 2. The data point selection page of the selected controller is displayed. It shows the modules and the data points.
Error, Aberration	
Comment	

Test 14.3	Browse data points and start point test	ok
Precondition	Test 14.2: Go to point selection page in device discovery view.	
Description	Expectation	
1. Tab on a module. 2. Tab on a data point.	1. The module expands or collapses its data points. 2. The point test view of the selected data point is displayed. It shows the overview page of the data point.	
Error, Aberration		
Comment		

4.15. UC15: Perform wiring test for a selection of points (focus on TXIO)

Test 15.1	Perform wiring test for a input data point	ok
Precondition	Test 14.2: Go to point selection page in device discovery view.	
Description	Expectation	
1. Tab on an analog, binary or multistate input data point. 2. Simulate a change for this data point in *Controller Simulation Tool*.	1. The overview page of the data point is displayed with data point details, commissioning information and a trend graph. 2. The new present value is displayed and the trend graph is updated.	
Error, Aberration		
Comment		

Test 15.2	Perform wiring test for an output data point	ok
Precondition	Test 14.2: Go to point selection page in device discovery view.	
Description	Expectation	
1. Tab on an analog, binary or multistate output data point. 2. Flick the overview page away. 3. Change the present value (command output).	1. The overview page of the data point is displayed with data point details, commissioning information and a trend graph. 2. The edit page of the data point is displayed. 3. The new present value is displayed in the *Controller Simulation Tool.*	
Error, Aberration		
Comment		

Test 15.3	Show data point details	ok
Precondition	Test 15.2: Perform wiring test for an output data point.	
Description	Expectation	
1. Precondition: Perform wiring test for an output data point. 2. Flick the edit page away.	1. The edit page of the data point is displayed. 2. The details page of the data point is displayed.	
Error, Aberration		
Comment		

4.16. UC16: Set commissioning state

Test 16.1	Show data point details	ok
Precondition	Test 15.2: Perform wiring test for an output data point.	
Description	Expectation	
1. Precondition: Perform wiring test for an output data point. 2. Change the commissioning state and press the Save button.	1. The edit page of the data point is displayed. 2. The new commissioning state is displayed in the *Controller Simulation Tool.*	
Error, Aberration		
Comment		

4.17. UC17: Add commissioning comment

Test 17.1	Show data point details	ok
Precondition	Test 16.1: Perform wiring test for an output data point.	
Description	Expectation	
1. Change the commissioning comment and press the Save button.	1. The new commissioning comment is displayed in the *Controller Simulation Tool.*	
Error, Aberration		
Comment		

4.18. UC18: Release all commanded objects (optional)

Will not be implemented.

4.19. UC19: Generate commissioning report (optional)

Will not be implemented.

4.20. UC20: Show alarm (optional)

Will not be implemented.

4.21. UC21: Confirm alarm (optional)

Will not be implemented.

5. Overview

Use Case	*Implement.*	*Error/Aberration*	*Status*
UC01: Start web service on gateway	yes		ok
UC02: Connect gateway	yes	A progress indication (blue points) would be nice. The timeout when a wrong gateway address was entered is very long (40 seconds). An abort button would be nice.	ok
UC03: Browse RA controllers	yes		ok
UC04: Map RA controller to corresponding hardware	yes		ok
UC05: Remove controller - hardware mapping	no, optional		-
UC06: Search RA controllers with filter	yes		ok
UC07: Configure RA controller	no, optional		-
UC08: Unconfigure RA controller	no, optional		-
UC09: Send wink to RA controller	yes		ok
UC10: Receive service pin	yes		ok
UC11: Show controller details	yes		ok
UC12: Automatically search corresponding hardware	no, optional		-
UC13: Download Device- and Application Configuration	no, optional		-
UC14: Browse data points of an RA controller	yes		ok
UC15: Perform wiring test for a selection of points	yes		ok
UC16: Set commissioning state	yes		ok
UC17: Add commissioning comment	yes		ok
UC18: Release all commanded objects	no, optional		-
UC19: Generate commissioning report	no, optional		-
UC20: Show alarm	no, optional		-
UC21: Confirm alarm	no, optional		-

6. Improvements

6.1. Known Limitations

- The gateway address cannot be auto discovered.

Erfahrungsberichte

SMART COMMISSIONING

Windows Phone 7

Bachelorarbeit:	Mobile Commissioning Tool for Room Automation
Autoren:	Kaspar Fenner Reto Schneebeli
Betreuer:	Prof. Hansjörg Huser
Projektpartner:	Siemens Schweiz AG Building Technologies, Zug
Experte:	Stefan Zettel, Ascentiv AG, Zürich

1. Erfahrungsbericht von Reto Schneebeli

Ziemlich bald stand für mich fest, dass ich als Industriepartner wieder den eigenen Arbeitgeber (Siemens Schweiz AG, Zug) anfragen werde. Mein Studienkollege Kaspar Fenner, der sogar in derselben Abteilung arbeitet, war als Projektpartner prädestiniert. Wir haben schon einige Projekte gemeinsam zum Erfolg gebracht, unteranderem die Studienarbeit und das SE2-Projekt. Schon lange hatten wir die Idee, einmal eine Applikation für ein mobiles Gerät zu entwickeln. Interessanterweise ist die Siemens zurzeit ebenfalls dabei, in vielen Bereichen durch Prototypen die Machbarkeit und den Nutzen von mobilen Geräten abzuklären. Deshalb haben wir ein passendes Thema im Bereich der Inbetriebnahme von Raumkontrollern gefunden. Da unsere Abteilung fast ausschliesslich mit Microsoft Technologien entwickelt, war es naheliegen, als mobile Plattform das neue Windows Phone 7 zu wählen. Als Betreuer konnten wir glücklicherweise wieder Professor Hansjörg Huser gewinnen.

In allen vorgängigen Projekten mit Kaspar war die Arbeitsaufteilung immer dieselbe. Kaspar war vor allem für die unteren Layer (Infrastruktur, Kommunikation, Services und Businesslogik) zuständig und ich für die oberen Layer (Benutzeroberfläche und ein wenig Businesslogik). Dieses Mal wollten wir beide etwas Neues ausprobieren und haben somit die Rollen getauscht. Ich habe mich grundsätzlich um den Gateway, den Kontrollersimulator und den Service-Layer auf dem *Smart Commissioning Tool* (SCT-App) gekümmert und Kaspar um die Benutzerschnittstelle auf dem SCT-App und die ganzen Usability-Themen. Es war sehr interessant, mich in einem anderen Themenbereich zu vertiefen. Im Bereich der Webservices und der Businesslogik hat man mit ganz anderen Problemstellungen zu tun als beim Userinterface. Es geht mehr um saubere und verständliche Schnittstellen, Threadsicherheit bei Nebenläufigkeiten, Authentifizierung, intelligentes Zwischenspeichern (Caching), Logging und vieles mehr. Ein verteiltes System basierend auf REST-Webservices mit JSON als Datenübertragungsformat habe ich bis anhin noch nie umgesetzt. Ich konnte viel Erfahrung in diesem Bereich sowie in der Ganzen Problematik der Threadsicherheit bei Nebenläufigkeiten sammeln.

Eine Schwierigkeit bei diesem Projekt war einerseits die Doppelbelastung, denn Kaspar und ich haben beide nebenbei noch zwei Tage pro Woche für die Firma gearbeitet. Andererseits war die Aufgabenstellung ziemlich wage und das ganze Umfeld befand sich noch in Entwicklung. Zudem war unsicher, ab wann die Windows Phone 7 Hardware wirklich zur Verfügung steht. Ein grosser Teil der Arbeit war auf jeden Fall das Aufnehmen und Analysieren des Umfelds sowie das Austüfteln eines geeigneten Systemaufbaus. Die Idee mit dem Kontrollersimulator, welcher wirklich das DICP-Protokoll und die REST-Webservices simuliert, kam erst in der Projektmitte auf. Zuvor war das DICP-Protokoll der Siemens noch gar nicht fertig spezifiziert, geschweige denn implementiert.

Aus Projektmanagement-Sicht waren wir grundsätzlich gut im Zeitplan und konnten fast alle Meilensteine termingerecht einhalten. Unser Entscheid, mehr in den Technischen Bericht zu investieren, hat die gute Planung am Ende ein wenig durcheinandergebracht und resultierte in einem beträchtlichen Mehraufwand. Der Mehraufwand hat sich jedoch gelohnt, denn das Resultat ist ein umfangreicher Technischer Bericht, der alle wichtigen Themen behandelt. Interessant ist auch, dass die geleisteten Arbeitsstunden kontinuierlich von anfänglich 20h auf 40h gegen Projektende gestiegen sind. Es stellt sich natürlich die Frage, wie man diesen Anstieg in einem zukünftigen Projekt verhindern könnte. Eine Möglichkeit ist, früher mit den Abgabedokumenten zu beginnen. Eine weitere wäre vielleicht, sich zu Beginn höhere Ziele zu stecken, damit bereits am Anfang mehr Druck vorhanden ist.

Mit dem Resultat bin ich mehr als zufrieden. Es ist uns gelungen, eine saubere, benutzerfreundliche Applikation für das Windows Phone 7 zu schreiben, welche die Mobilität bei der Durchführung des Datenpunkttests erheblich erhöht und die Arbeit vereinfacht. Mit dem Kontrollersimulator konnten wir ebenfalls zeigen, wie eine einfache und saubere REST-Schnittstelle in einem Kontroller aussehen könnte. Das Feedback vom Industriepartner ist durchwegs positiv und ich würde mich freuen, wenn aus unserem Prototyp ein Produkt entstehen würde, welches den Inbetriebnahme-Ingenieuren das Arbeiten erleichtert.

2. Erfahrungsbericht von Kaspar Fenner

Für diese Bachelorarbeit kam eine Traumkombination zustande: Als Industriepartner der eigene Arbeitgeber und als Projektpartner der Studienkollege, welcher in derselben Abteilung arbeitet. Damit waren die Grundvoraussetzungen für eine Erfolgreiche Arbeit auf jeden Fall gegeben, da die Motivation natürlich enorm war.

Der Projektstart war jedoch schwierig. Die Anforderungen waren noch etwas schwammig und die Abhängigkeiten zum restlichen Umfeld aus seidenen Fäden, welche jeden Moment zu zerreissen drohten. Der Grund dafür war, dass sich die neue Generation der Raumautomation zurzeit bei Siemens in Entwicklung befindet. Das heisst, das ganze Umfeld, in dem unsere Bachelorarbeit verankert ist, war zu Beginn der Arbeit im Wandel. Zum Teil waren noch nicht einmal genaue Spezifikationen verfügbar, geschweige denn implementierte Software, auf der man hätte aufbauen können. Diese Situation verbesserte sich zwar allmählich zur Projektmitte hin. Die letzten Steine fielen aber erst in Woche neun, als wir unsere Windows Phone 7 Geräte für die Entwicklung freischalten konnten. Von da an durften wir uns für die restlichen fünf bis sechs Wochen in einem mehr oder weniger definierten Umfeld bewegen.

Der Start eines Projektes, das in ein fixes Umfeld eingebettet ist, wie dies z.B. bei der Studienarbeit der Fall war, ist einiges einfacher. In so einem Umfeld kann schneller und leichter Handfestes geschaffen werden. Bei dieser Bachelorarbeit floss hingegen zuerst viel Zeit in den Wissensaufbau, da bei diesem Projekt relativ viel Domain-Know-How erforderlich war. Das meiste von diesem Wissen war aber nicht auf Papier vorhanden, sondern musste bei verschiedenen Personen erfragt werden. Dieses Projekt ist schon fast als Interdisziplinär zu betrachten, da mit Personen aus vielen verschiedenen Tätigkeitsbereichen zusammengearbeitet werden musste: Personen aus der Softwareentwicklung, der Firmware-Entwicklung, der Systemarchitektur, des Produktedesigns sowie einem „Benutzer-Vertreter“. Das alles ist bei einem innovativen Projekt im Umfeld eines Grossunternehmens natürlich normal. Für eine relativ kleine Bachelorarbeit mit engem Zeitrahmen aber doch sehr anspruchsvoll.

Bei allen vorhergehenden Projekten, die ich zusammen mit Reto Schneebeli, meinem Studienkollegen, durchgeführt hatte, war ich nie für die Benutzeroberfläche zuständig. Die Domain- und Infrastructure-Layers mit Services und Kommunikation waren meine präferierten Gebiete. Dieses Mal wollten wir die Aufgabenbereiche austauschen, um neue Erfahrungen sammeln zu können. Diese Entscheidung hat sich als goldrichtig herausgestellt, denn es war wirklich sehr interessant und lehrreich für mich, eine Benutzeroberfläche für ein Smartphone umzusetzen. Insbesondere, weil ich zuvor noch nie etwas auf einer mobilen Plattform entwickelt hatte. Ich musste (wieder einmal) erfahren, wie mühsam GUI-Entwicklung sein kann, und wie viele schwierige Probleme es zu lösen gibt, die von aussen betrachtet einfach erscheinen.

Ein kleiner Vorteil hatten wir aber beide, denn die verwendeten Microsoft Technologien waren uns nicht völlig neu. Ich hatte zuvor einfach noch fast nichts mit XAML gemacht und die Windows Phone-spezifischen Konzepte waren natürlich völlig neu für mich.

Das Projektmanagement fand ich diesmal eigentlich recht gelungen. Wir haben auch einige Risiken richtig erkannt und genügend Zeit eingeplant. Da die Projektdauer von 360h aber sowieso sehr knapp ist, haben wir von Anfang an eingeplant, möglichen Mehraufwand zum Teil durch Mehrarbeit aufzuholen. Dies war dann für den Technischen Bericht auch nötig, dessen Umfang wir zu Beginn anders eingeplant hatten.

Das durchwegs positive Feedback zur geleisteten Arbeit hat mich sehr gefreut. Ich hoffe, dass unsere Arbeit einen nützlichen Beitrag für zukünftige Entscheidungen und Lösungen im Bereich von mobilen Plattformen und der Inbetriebnahme im spezifischen, zu leisten vermag.

Glossar

Begriff	Beschreibung
ABT	Automation Building Tool ist der Name des Engineering-Tools für System ONE.
BACnet	Building Automation and Control Networks ist ein standardisiertes Netzwerkprotokoll für die Gebäudeautomation (ISO-Norm 16484-5).
CST	Controller Simulation Tool ist der Name der Software, welche die Kontroller simuliert.
DALI	Siehe TXIO
DESIGO	Gebäudeautomatisierungssystem von Siemens BT.
DICP	Device Identification and Configuration Protocol ist ein Protokoll für das Identifizieren und Konfigurieren von Kontrollern.
Gestures	Bewegungen, die auf einem Touch-Display mit einem oder zwei Fingern ausgeführt werden und bestimmten Mustern entsprechen.
HW	Hardware (physikalisches Gerät)
IO	Input / Output (Signaleingang / Signalausgang).
JSON	JavaScript Object Notation ist kompaktes Dateiformat, welches für Menschen wie auch für Maschinen einfach lesbar ist und für den Datenaustauch zwischen Applikationen benutzt wird.
MVVM	Model View ViewModel ist ein Architekturpattern von Microsoft für die UI Entwicklung in WPF oder Silverlight. Es basiert auf dem Presentation-Model Pattern von Martin Fowler.
Node-Setup	Das Node-Setup beinhaltet alle Aufgaben, welche für das Auffinden und Konfigurieren der Kontroller notwendig sind wie z.B. Kontroller suchen, Kontroller konfigurieren, *Flash LED* und *Service Pin*.
RA	Room Automation (Raumautomation)
REST	Representational State Transfer ist ein Softwarearchitekturstil für verteilte Hypermedia-Informationssysteme wie das World Wide Web.
SARB	Siehe TXIO
SOA	Service Oriented Architecture steht für eine Serviceorientierte Architektur.
SCG	Smart Commissioning Gateway ist der Name des Webservices, welcher auf dem Engineering Laptop läuft.
SCT	Smart Commissioning Tool ist der Name für die WP7 Applikation.
System ONE	System ONE ist das neue Engineering-System für die Raumautomation von Siemens BT.
Systemhaus	Siehe VAP
TIA Portal	Totally Integrated Automation Portal ist ein Softwaresystem von Siemens, welches vom ABT benutzt und erweitert wurde.
TXIO	SARB, DALI und TXIO sind Untersysteme von Siemens für die Gebäude und Raumautomation.
UI	User Interface (Benutzeroberfläche)
VAP	Mit Value Added Partners, kurz VAPs, werden Firmen bezeichnet, welche Siemensprodukte verkaufen. Diese Firmen werden auch als Systemhäuser bezeichnet.
WP7	Windows Phone 7 ist die neue Mobile-Plattform von Microsoft und ersetzt Windows Mobile (Release: Ende Oktober 2010).

WCF	Windows Communication Foundation ist eine dienstorientierte Kommunikationsplattform von Microsoft für verteilte Anwendungen.
WPF	Windows Presentation Foundation ist ein Grafik-Framework von Microsoft basierend auf XAML-Dateien. In XAML werden Oberflächen deklarativ in XML beschrieben.
XWORKS	Engineering-System für die Gebäudeautomation von Siemens BT.

Literaturverzeichnis

[1] Siemens Building Technologies - Products & Systems
http://w1.siemens.ch/web/bt_ch/de/products_systems/Pages/products_systems.aspx
letzter Zugriff am 03.12.2010

[2] App Hub – Develop for Windows Phone & Xbox 360
http://create.msdn.com/en-US
letzter Zugriff am 15.12.2010

[3] WPF-Anwendungen mit dem Model-View-ViewModel-Entwurfsmuster
http://msdn.microsoft.com/de-de/magazine/dd419663.aspx
letzter Zugriff am 18.12.2010

[4] Execution Model for Windows Phone
http://msdn.microsoft.com/en-us/library/ff769557(v=VS.92).aspx
letzter Zugriff am 20.12.2010

[5] Windows Phone Developer Forum - Treeview?
http://social.msdn.microsoft.com/Forums/en-US/windowsphone7series/thread/db91d4b2-5d20-456e-8a4a-b000e81f4ca3
letzter Zugriff am 21.12.2010

[6] App Hub Forums - TreeView control
http://forums.create.msdn.com/forums/p/66701/407847.aspx
letzter Zugriff am 21.12.2010

[7] A Developer's Guide to the WCF REST Starter Kit
http://msdn.microsoft.com/en-us/library/ee391967.aspx
letzter Zugriff am 15.10.2010

[8] Eine Einführung in REST-Dienste mit WCF
http://msdn.microsoft.com/de-de/magazine/dd315413.aspx
letzter Zugriff am 20.10.2010