HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

SIEMENS

XWORKS ProjectManager

Webclient mit Silverlight

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2009

Autoren:	Kaspar Fenner
	Reto Schneebeli
Betreuer:	Prof. Hansjörg Huser
Projektpartner:	Siemens BT, Zug

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	Abstract 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

XWORKS ProjectManager - Webclient mit Silverlight

Aufgabenstellung für Kaspar Fenner, Reto Schneebeli

Einführung

DESIGO XWORKS *plus* ist das neue Engineering Tool um das Siemens Gebäudeautomationssystem DESIGO in Betrieb nehmen zu können. DESIGO XWORKS plus basiert wie bereits das Vorgänger Tool DTS auf der Siemens Simatic Programmierumgebung Step7 und CFC.
Sämtliche neuen Applikationen – vor allem Frontend - von XWORKS plus wurden innerhalb dem Projekt DESIGO V4.0 mit Microsoft .net und C# Technologie entwickelt. Die XWORKS plus wurde mit der Systemversion DESIGO V4.0 Anfangs April 2009 auf den Markt gebracht.

Eine der zentralen Applikationen von XWORKS plus ist der **XwpProjectManager**. Es ist das zentrale Tool wo ein Projektleiter Projekte lokal aufsetzen kann und zentral auf einem sogenannten **Branch Office Server** (BOS-Server) verwalten kann.
Der BOS Server wird typischerweise in den Siemens Ländervertretungen zentral verwaltet und beinhaltet (hosted) sämtliche Projektdaten von DESIGO Projekten, welche entweder von Siemens Ingenieuren oder sogar von Partnerfirmen (Value Added Partners, VAP) verwendet werden können.

Aufgabenstellung

1.) Entwicklung eines Web-Clients für Administratoren (mit Silverlight 3)

Die Benutzung des XwpProjectManagers Clients erfordert zurzeit eine Vollinstallation von XWORKS plus. Der XwpProjectManagers ist einerseits die Client-Applikation für den DESIGO Ingenieur, als auch das Admin Tool für den BOS Administrator (sofern mit Administratoren Rechten versehen).

Es wird gewünscht, dass der BOS Administrator einen Web-Client zur Verfügung hat um die wichtigsten Administratoren Aufgaben (bsp User Recht Verwaltung, Browse oder Undo Check-out von beliebigen Projekten) mit einem Web-Client tätigen kann.

Der XwpProjectManager Web-Client soll als Prototyp mit Microsoft Silverlight 3 Technologie auf MS Internet Explorer bzw. Firefox laufen können. Es gilt vor allem darum die Technolgie Silverlight einzusetzen und die Möglichkeiten und Erfahrungen in das Software Team einzubringen.

Der Web-Client wird (noch) nicht als verkaufbare Lösung benötigt, dh: es gibt Spielraum die Möglichkeiten an einem praktischen Beispiel zu erforschen.

Es ist darauf zu achten, dass es sich bei BOS Daten um sensible Daten handelt, dh. der WebClient muss so ausgelegt werden, dass die Sicherheitskonzepte (Login / Authentifizierung, gesicherte Datenübertragung (https)) gewährleistet werden.

Mögliche Use Cases:

- Login (Authentizierung)
- Anzeige von Folder / Projekt-Struktur

Arbeitsaufwand

Für die erfolgreich abgeschlossene Arbeit werden 8 ECTS angerechnet. Dies entspricht einer Arbeitsleistung von 240 Stunden pro Student. Bei einer 14-wöchigen Laufzeit sind dies ca. 2 Arbeitstage pro Woche.

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.).

- Anzeigen von Log Funktionalitäten
- Ausführen von Browse und Search Funktionen
- Anlegen von User, Gruppen und User-Gruppen Zuweisung
- Vergabe von Rechten von User an bestimmte Folder / Projekte
- Undo Check-out (als Admin)

Einschränkung:

- Der ***BOS-WebAdmin*** Client muss nicht Check-Out Funktionen ausführen können.
- Der BOS Server sollte möglichst nicht erweitert werden müssen; dh. es sollten wenn möglich die aktuellen Services verwendet werden.

Resultate

- Ausführbare Software mit Dokumentation
- Präsentation der Arbeit bei Siemens I BT (zB R&D Forum oder Abteilungs – Meeting)

Projektpartner

Auftraggeber

René Föhn
Project Manager DESIGO V5 Tools

Ekaterina Panteleeva

Senior SW Architect
Project Manager DESIGO V4.1 Tools

Studenten

Reto Schneebeli

Kaspar Fenner

Betreuung HSR

Hansjörg Huser

Projektabwicklung

Termine:

- Beginn der Arbeit: M**o., 14. Sept. 2009**
- Abgabetermin Kurzfassung zum Review: Fr. **11.12.2009**
- Abgabetermin (ink. Poster): **Fr. 18.12.2009**, 17.00 Uhr
- Zwischenbesprechung/Review mit Auftraggeber nach Projektplan

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, 14. Sept.09
Hansjörg Huser

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

Kurzfassung der Studienarbeit

Abteilung	Informatik
Namen der Studierenden	Kaspar Fenner Reto Schneebeli
Studienjahr	HS 2009
Titel der Studienarbeit	XWORKS ProjectManager – Webclient mit Silverlight
Examinatorin / Examinator	Prof. Hansjörg Huser
Themengebiet	Software
Projektpartner	Siemens Schweiz AG, Zug
Institut	INS, Institute for Networked Solutions

Kurzfassung der Studienarbeit

DESIGO XWORKS plus ist das neue Engineering-Tool für die Inbetriebnahme des Siemens Gebäudeautomationssystem *DESIGO*. Eine der zentralen Applikationen von *XWORKS plus* ist der *XwpProjectManager*. Dieser erlaubt es einem Projektleiter, *DESIGO*-Projekte lokal aufzusetzen und zentral auf einem *Branch Office Server (BOS)* zu verwalten.

Inhalt dieser Studienarbeit ist die Konzeption und Realisierung eines RIA-Webclients mit *Microsoft Silverlight 3* für die Administration des BOS-Servers. Dabei gibt es zwei Hauptproblembereiche: Einerseits sollen die typischen Administrations-UseCases analysiert und für den Webclient umgesetzt werden. Andererseits muss eine Lösung gefunden werden, wie der Webclient in die bestehende Client-Server-Architektur integriert werden kann. Dabei verlangt eine Anforderung, dass die bestehende WCF-Serviceschnittstelle auf dem Server möglichst nicht erweitert werden muss. Während der Problemanalyse stellte sich heraus, dass die Serverschnittstelle jedoch zwingender Änderung bedarf, da *Silverlight 3* nicht den vollen WCF-Funktionsumfang unterstützt. Insbesondere für die Umsetzung des geforderten Sicherheitskonzeptes (Zertifikate, verschlüsselte Datenübertragung, etc.) mussten *Silverlight*-spezifische Anpassungen vorgenommen werden.

Bei der Entwicklung des Webclients lag der Fokus auf der Ausarbeitung eines sauberen Designs. Bei der Umsetzung der gewählten Dreischichtenarchitektur kamen verschiedene Patterns und Technologien zum Einsatz. Im Presentation-Layer werden u.a. Dependency-Injection, View-Injection und Binding verwendet. Im Domain-Layer wurde das Prinzip einer asynchronen SOA mit Service-Locator umgesetzt. Dadurch kann der Webclient sehr einfach mit neuen Services erweitert werden. Da konsequent Interface und Implementation getrennt wurden, können alle drei Schichten mit Unit-Tests getestet werden.

Für die Umsetzung des User Interfaces wurde der existierende *XwpProjectManager* als Vorlage genommen, da die User-Experience vergleichbar sein sollte. Dies stellte sich aber als schwierig heraus, da sich Microsoft bei der Umsetzung des RIA-Konzeptes in gewissen Bereichen mehr für Plattformunabhängigkeit als für die volle Unterstützung einer Desktopumgebung entschieden hat. Da das *Silverlight*-Framework sehr schlank ausgefallen ist, mussten auch einige GUI-Komponenten selbst implementiert werden.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

Management Summary

XWORKS ProjectManager

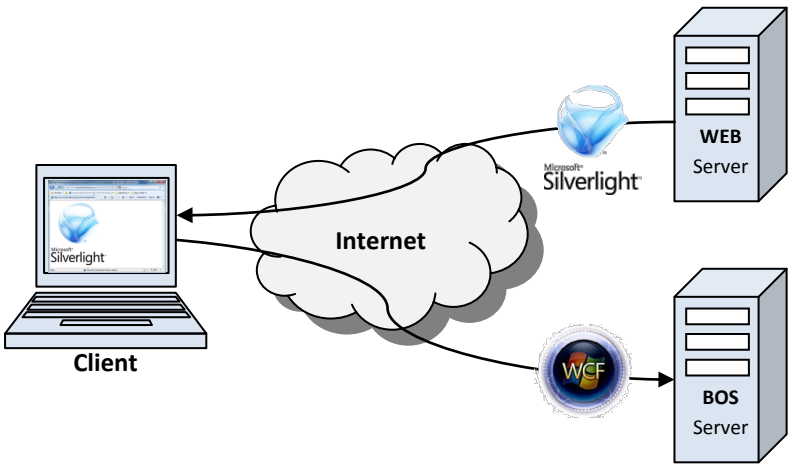
Webclient mit Silverlight

Diplomanden	Kaspar Fenner Reto Schneebeli
Examinator	Prof. Hansjörg Huser
Experte	
Themengebiet	Software
Industriepartner	Siemens Schweiz AG, Zug

Ausgangslage:

DESIGO XWORKS plus ist das neue Engineering-Tool, um das Siemens Gebäudeautomationssystem *DESIGO* in Betrieb nehmen zu können. Eine der zentralen Applikationen von *XWORKS plus* ist der *XwpProjectManager*, mit dem ein Projektleiter Projekte lokal aufsetzen und zentral auf einem sogenannten *Branch Office Server (BOS)* verwalten kann. Die Benutzung des *XwpProjectManagers*-Clients erfordert zurzeit eine Vollinstallation von *XWORKS plus*. Der *XwpProjectManager* ist einerseits die Client-Applikation für den *DESIGO*-Ingenieur, als auch das Admin-Tool für den BOS-Administrator.

Inhalt dieser Studienarbeit ist die Konzeption und Realisierung eines Webclients mit *Microsoft Silverlight 3* für die Administration des BOS-Servers.



Der *Silverlight*-Webclient wird auf einem Webserver gehosted. Dies kann auch derselbe Server sein, auf dem BOS läuft. Beim Aufruf des Webclients via Internet-Browser wird die *Silverlight*-Applikation auf das Client-System heruntergeladen und dort lokal im Browser ausgeführt. Mit dem Webclient kann sich der Benutzer dann auf einem beliebigen BOS-Server anmelden.

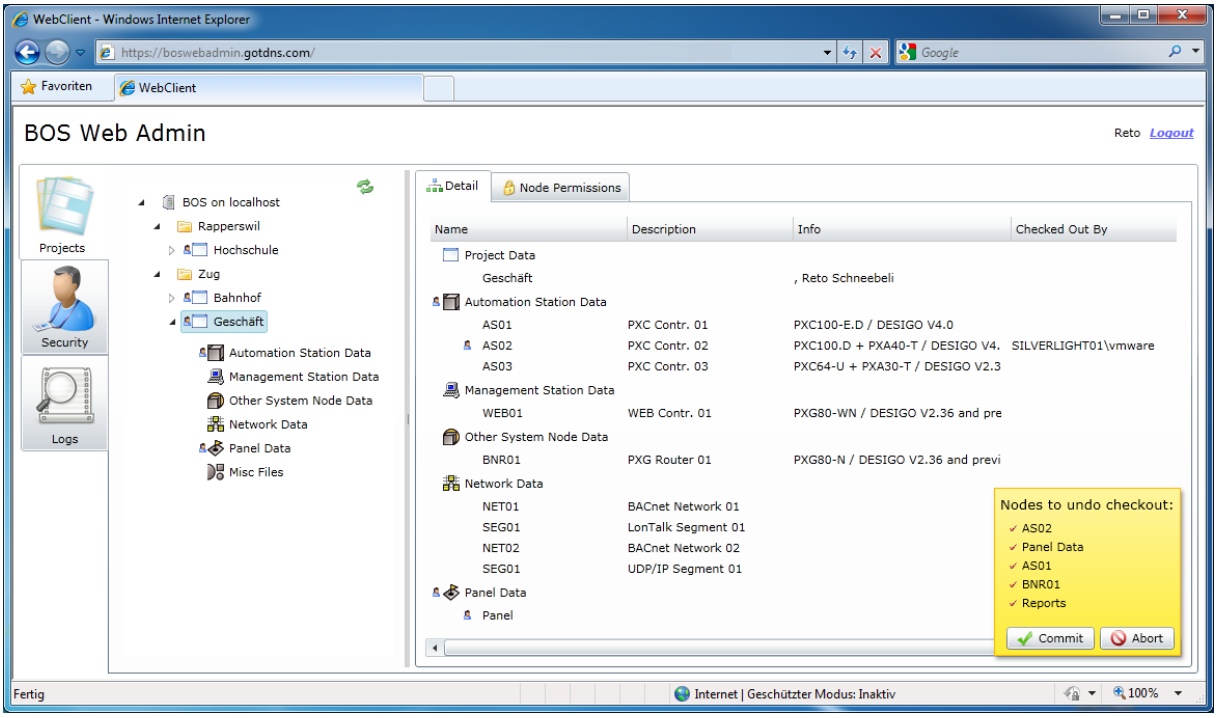
Vorgehen/Technologien:

Das Entwicklungsprojekt wurde nach den standardisierten Phasen von RUP durchgeführt. Entwickelt wurde in einem Iterativen-Prozess mit einer Inception-Iteration, zwei Elaborations-Iterationen, zwei Construction-Iterationen und einer Transition-Iteration. Um die Architektur zu verifizieren wurde während der Designphase bereits ein Prototyp über alle Layer hinweg entwickelt und zur Halbzeit der 14-Wöchigen Projektdauer fertiggestellt.

Der Webclient wurde mit *Silverlight 3* und C# 3.5 entwickelt. Die Benutzeroberfläche wird bei *Silverlight* in XAML geschrieben, einer Beschreibungssprache für die Oberflächengestaltung von Anwendungen. *Silverlight* erlaubt die Entwicklung von *Rich Internet Applications (RIA)* und unterstützt auch *Windows Communication Foundation (WCF)*. Da der *XwpProjectManager* bereits über WCF mit dem BOS-Server kommuniziert, konnte dieselbe Technologie auch beim Webclient eingesetzt werden.

Ergebnisse:

Ergebnis der Studienarbeit ist der *BOS-WebAdmin* Client und die dazugehörige Dokumentation. Zur Dokumentation gehört der Projektplan, die Domainanalyse, Architektur- und Designdokument, Technischer-Bericht, Systemtestdokument.



Beim *BOS-WebAdmin* wurde grossen Wert auf eine einfach bedienbare Benutzeroberfläche gelegt. Ein Beispiel dafür ist das kleine Fenster, um die bevorstehenden *„Undo Check-out“* Operationen zu bestätigen oder abzubrechen. Damit die Benutzeroberfläche reaktionsfähig bleibt und nie einfriert, werden die gerade benötigten Daten inkrementell im Hintergrund geladen. Dies wird durch die asynchrone Serviceschnittstelle ermöglicht.

Ausblick :

Der in der Studienarbeit erstellte Webclient wird der Entwicklungsabteilung und auch einigen *DESIGO* Ländervertretungen vorgestellt. Bei grossem Anklang wird er möglicherweise Bestandteil der Version 5 von *DESIGO XWORKS plus*. Dafür müsste aber der Funktionsumfang noch um einige wichtige Funktionen wie z.B. „Suchen“ oder „History ansehen“ erweitert werden.

Projekt: BOS-WebAdmin

Technischer Bericht

0. Dokumentinformationen

0.1. Änderungsgeschichte

Datum	*Version*	*Änderung*	*Autor*
29.11.2009	0.1	Dokumentstruktur definiert	Kaspar Fenner
12.12.2009	0.2	Kapitel zum Domainmodell, Design und User Interface	Reto Schneebeli
12.12.2009	0.3	Kapitel 1 + 2 überarbeitet	Kaspar Fenner
13.12.2009	0.4	Kapitel 5, 6 und 8 überarbeitet	Reto Schneebeli
13.12.2009	0.5	Kapitel 4.5	Kaspar Fenner
16.12.2009	0.6	Kapitel 7 und 8 überarbeitet	Reto Schneebeli
16.12.2009	0.7	Kapitel 5 und 8 überarbeitet	Kaspar Fenner
17.12.2009	1.0	Kapitel 9	Kaspar Fenner

0.2. Inhalt

0. Dokumentinformationen 2
0.1. Änderungsgeschichte 2
0.2. Inhalt 3
1. Einleitung 5
1.1. Definitionen und Abkürzungen 5
1.2. Referenzen 5
2. Projektauftrag 5
2.1. Allgemeine Übersicht 5
2.2. Aufgabenstellung 5
3. Anforderungen 6
3.1. Generelle Anforderungen 6
3.1.1. BOS-Server vor unberechtigtem Zugriff schützen (Login/Logout) 6
3.1.2. Aktionen protokollieren 6
3.1.3. Bedienbarkeit 6
3.1.4. Leistung 6
3.1.5. Datenschutz 6
3.1.6. Wartbarkeit 6
3.1.7. Schnittstellen 6
3.2. Use Cases 7
3.2.1. Diagramm 7
3.2.2. UC01: BOS-Server verbinden 7
3.2.3. UC02: XWP-Projekte browsen 7
3.2.4. UC03: BOS-Server-Verbindung trennen 7
3.2.5. UC04: Auschecken rückgängig machen (Undo check-out) 8
3.2.6. UC05: History ansehen (nicht umgesetzt) 8
3.2.7. UC06: Suchen (nicht umgesetzt) 8
3.2.8. UC07: Benutzer verwalten (CRUD) 8
3.2.9. UC08: Benutzergruppen verwalten (CRUD) 8
3.2.10. UC09: Rollen verwalten (CRUD) 8
3.2.11. UC10: Generelle Berechtigungen verwalten 8
3.2.12. UC11: Projekt/Ordner-spezifische Berechtigungen verwalten 8
3.2.13. UC12: Rolle zu Benutzer zuweisen 8
3.2.14. UC13: Rolle zu Benutzergruppe zuweisen 8
3.2.15. UC14: History bereinigen (nicht umgesetzt) 8
4. Systemarchitektur 9
4.1. Übersicht 9
4.2. Domainmodell 10
4.2.1. Diagramm 10
4.2.2. Beschreibung 11
4.3. Rechtemodell 11
4.4. Client-Server Architektur 13
4.5. Kommunikation mit WCF 14
4.5.1. Ausgangslage 14
4.5.2. Einschränkungen durch Silverlight 14
4.5.3. Anpassungen der Serverschnittstelle 14
5. Design 16
5.1. Übersicht 16
5.2. Verwendung der Composite Application Library 17
5.2.1. View-Injection (Named-Locations) 17
5.2.2. Low Coupling (Event-Aggregation) 17
5.2.3. Dependency-Injection (Constructor-Injection) 17
5.3. Separation of concerns 17

5.4. Asynchrone Servicearchitektur 17
5.5. Fehlerbehandlung 18
5.6. Sicherheit 18
6. User Interface 18
6.1. Übersicht 18
6.2. Einschränkungen durch Silverlight 18
6.3. Umsetzungs-Highlights 19
6.3.1. Custom ComboBox für Server-Auswahl 19
6.3.2. Intelligenter Browser-Tree mit Detail-View 20
6.3.3. „Prepare and Commit“ für Undo-Checkout 21
7. Qualitätssicherung 21
7.1. Unit Tests 21
7.2. Systemtests 21
7.3. Codequalität 22
8. Technologiestudium 22
8.1. Silverlight 3 22
8.2. Composite Application Guidance 22
8.2.1. Unity Container (Dependency-Injection) 22
8.2.2. Event Aggregation 23
8.2.3. Regions (View-Injection) 23
8.3. Moq (Mocking Library for .NET 3.5) 23
9. Schlussfolgerungen 24
9.1. Zusammenfassung 24
9.2. Beurteilung der Resultate 24
9.3. Ausblick 25

1. Einleitung

1.1. Definitionen und Abkürzungen

Begriff	Beschreibung
XWP	DESIGO **XW**ORKS **p**lus (Engineering-Umgebung von Siemens BT)
BOS	**B**ranch **O**ffice **S**erver (Zentrale Ablage der XWORKS Projekte)
CAG	**C**omposite **A**pplication **G**uidance (Designpatterns und eine Library welche einem unterstützen beim erstellen von modulare WPF oder Silverlight Applikationen.)
CAL	**C**omposite **A**pplication **L**ibrary (Eine Library welche einem unterstützen beim Erstellen von modularen WPF oder Silverlight Applikationen.)
RIA	**R**ich **I**nternet **A**pplications
WCF	**W**indows **C**ommunication **F**oundation
WPF	**W**indows **P**resentation **F**oundation

1.2. Referenzen

Composite Application Guidance for WPF and Silverlight
Anforderungsspezifikation.pdf
Domainanalyse.pdf
Studie_CompositeApplicationGuidance.pdf
Softwarearchitektur.pdf
Systemtest - Final Release.pdf

2. Projektauftrag

2.1. Allgemeine Übersicht

DESIGO XWORKS plus ist das neue Engineering-Tool, um das Siemens Gebäudeautomationssystem *DESIGO* in Betrieb nehmen zu können. Eine der zentralen Applikationen von *XWORKS plus* ist der *XwpProjectManager*, mit dem ein Projektleiter Projekte lokal aufsetzen und zentral auf einem sogenannten *Branch Office Server (BOS)* verwalten kann. Der BOS-Server wird typischerweise in den Siemens Ländervertretungen zentral verwaltet und beinhaltet (hosted) sämtliche Projektdaten von *DESIGO*-Projekten. Diese Projekte können entweder von Siemens Ingenieuren oder auch von Partnerfirmen (Value Added Partners, VAP) verwendet werden.

2.2. Aufgabenstellung

Die Benutzung des *XwpProjectManager*-Clients erfordert zurzeit eine Vollinstallation von *XWORKS plus*. Der *XwpProjectManager* ist einerseits die Client-Applikation für den *DESIGO*-Ingenieur, als auch das Admin-Tool für den BOS-Administrator (sofern mit Administratoren Rechten versehen).

Es wird gewünscht, dass der BOS-Administrator einen Webclient zur Verfügung hat, um die wichtigsten Administrationsaufgaben, wie z.B. Benutzerrechte verwalten oder einen Check-out rückgängig machen, direkt über den Internet-Browser tätigen kann.

Der *BOS-WebAdmin* soll als Prototyp mit Microsoft Silverlight 3 Technologie auf MS Internet Explorer bzw. Firefox laufen können. Es geht vor allem darum, die Technologie Silverlight einzusetzen und die Möglichkeiten und Erfahrungen in das Software-Team einzubringen.

Es ist darauf zu achten, dass es sich bei BOS-Daten um sensible Daten handelt, d.h. der Webclient muss so ausgelegt werden, dass die Sicherheitskonzepte (Login / Authentifizierung, gesicherte Datenübertragung (https)) gewährleistet sind.

3. Anforderungen

3.1. Generelle Anforderungen

Nachfolgend sind die wichtigsten generellen Anforderungen, die nicht durch UseCases abgedeckt werden, in stark gekürzter Form beschrieben.

3.1.1. BOS-Server vor unberechtigtem Zugriff schützen (Login/Logout)

Da mit dem BOS-WebAdmin keine XWP-Installation mehr nötig ist und der BOS-Server möglicherweise über das Internet erreichbar ist, muss der BOS-WebAdmin vor unberechtigtem Zugriff auf den BOS-Server schützen. Der Benutzer muss sich bei jedem Zugriff mit seinen Credentials (Benutzernamen, Passwort) authentifizieren. Der Benutzer muss sich auch jederzeit wieder vom BOS-Server abmelden können.

3.1.2. Aktionen protokollieren

Da die Verbindung zwischen Client und BOS-Server anfällig für Unterbrüche ist, können Aktionen unangekündigt unterbrochen werden. Dabei dürfen im Fehlerfall keine Inkonsistenzen entstehen.

3.1.3. Bedienbarkeit

Die Bedienung soll sich soweit möglich am bestehenden XWP-ProjectManager orientieren, damit sich die Benutzer nicht völlig neu orientieren müssen.

3.1.4. Leistung

Die Ansprechbarkeit des Userinterfaces soll ohne menschlich wahrnehmbare Verzögerung erfolgen und zu jedem Zeitpunkt gegeben sein.
Die Anzahl Benutzer und Projekte, die mit dem BOS-WebAdmin verwaltet werden können soll nicht limitiert sein. Genaue Angaben müssen nicht gemacht werden, da diese Werte in einem relativ überschaubaren Bereichen liegen. So hostet z.B. der Schweizer BOS-Server ca. 2000 Projekte und hat knapp 200 Benutzer. Für grössere Länder sind diese Werte maximal um das 5 bis 10-fache höher.

3.1.5. Datenschutz

Benutzer müssen vom BOS-Server mit ihren Credentials (Benutzername, Passwort) authentifiziert werden. Benutzer dürfen nur die Aktionen durchführen, für die sie die Rechte besitzen (Autorisierung). Die gesamte Kommunikation zwischen Client und Server erfolgt über HTTPS wozu ein serverseitiges Zertifikat verwendet wird.

3.1.6. Wartbarkeit

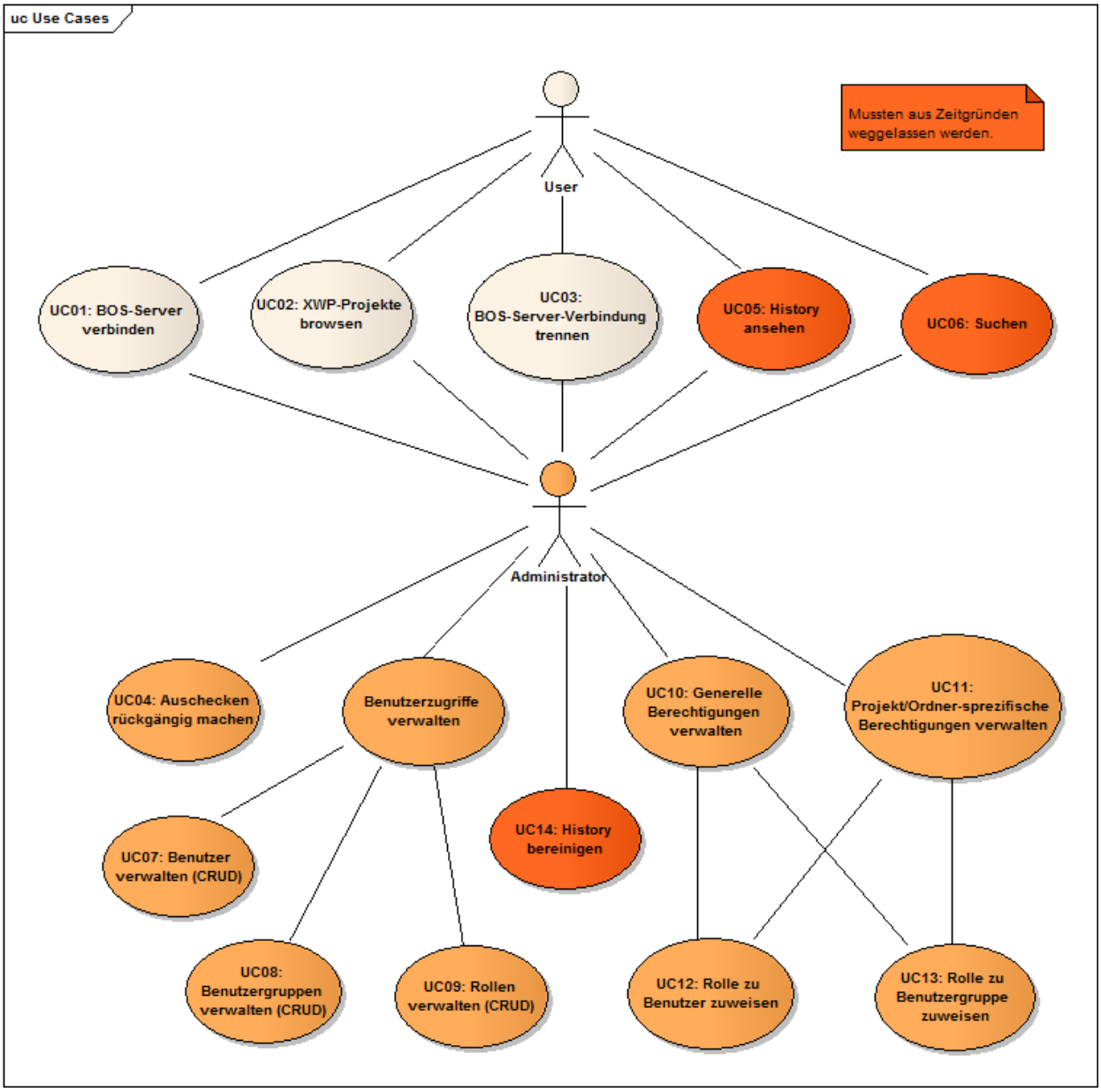
Die gewählte Softwarearchitektur soll einfach erweiterbar sein und so das Hinzufügen von zukünftigen Features so einfach als möglich gestalten. Das Produkt muss über mehrere Zyklen wartbar sein.

3.1.7. Schnittstellen

Die Kommunikation mit dem BOS-Server erfolgt via WCF und wsHttps-Binding. Um die Verbindung mit dem BOS-Server herzustellen, werden Hostname, Benutzername und Passwort benötigt.

3.2. Use Cases

3.2.1. Diagramm



3.2.2. UC01: BOS-Server verbinden

Der Benutzer verbindet mit einem BOS-Server (Login). Nach erfolgreicher Authentifizierung des Users wird der Projektbaum geladen und angezeigt.

3.2.3. UC02: XWP-Projekte browsen

Der Benutzer kann durch den Projektbaum navigieren, ähnlich wie beim Windows Explorer, und so den gewünschten Ordner, Projekt oder Sub-Node (AS, MS, Network, etc.) auswählen. Für einen angewählten Node werden jeweils die entsprechenden Detailinformationen angezeigt. Dieser Use Case ist die Voraussetzung für fast alle weiteren Use Cases.

3.2.4. UC03: BOS-Server-Verbindung trennen

Der Benutzer trennt die Verbindung zum BOS-Server (Abmelden). Nachdem die Verbindung getrennt wurde, wird der Login-Bildschirm angezeigt.

3.2.5. UC04: Auschecken rückgängig machen (Undo check-out)

Der Benutzer selektiert ein ausgechecktes Projekt oder ein oder mehrere ausgecheckte Sub-Nodes (AS, MS, Network, etc.) unterhalb eines Projektes und macht das Auschecken rückgängig. Der „Check-out"-Status wird von den gewählten Nodes entfernt. Damit werden die Projekte für andere Benutzer wieder bearbeitbar.

3.2.6. UC05: History ansehen (nicht umgesetzt)

Der Benutzer kann sich die Vergangenheit (History) eines Projektes oder des kompletten BOS-Servers ansehen. Zur Einschränkung stehen drei Kriterien zur Verfügung: User, Operation oder Zeitperiode. Alle drei können miteinander verknüpft werden. Als Resultat wird eine Liste mit den gefundenen History-Einträgen angezeigt. (Die History auf einem Ordner wird momentan noch nicht unterstützt.)

3.2.7. UC06: Suchen (nicht umgesetzt)

Der Benutzer selektiert einen Ordner oder ein Projekt und kann in allen darunter liegenden Nodes nach einem Text suchen. Es kann in mehreren Textelementen (ProjectName, ProjectVersion, NodeName, ...) kombiniert gesucht werden. Als Resultat werden alle Nodes aufgelistet die den Suchkriterien entsprechen.

3.2.8. UC07: Benutzer verwalten (CRUD)

Der Benutzer kann einen User erstellen, bearbeiten oder löschen und Benutzergruppen zuweisen. Ein User hat einen Benutzernamen, ein Passwort, eine Email-Adresse, einen Vornamen, einen Nachnamen und einen Kommentar.

3.2.9. UC08: Benutzergruppen verwalten (CRUD)

Der Benutzer kann eine Benutzergruppe erstellen oder löschen und User zuweisen. Eine Benutzergruppe hat nur einen Namen.

3.2.10. UC09: Rollen verwalten (CRUD)

Der Benutzer kann eine Rolle erstellen, bearbeiten oder löschen und Rechte zuweisen. Ein Rolle hat einen Namen und eine Beschreibung.

3.2.11. UC10: Generelle Berechtigungen verwalten

Der Benutzer weist Rollen an Benutzer oder Benutzergruppen zu, die dann die applikationsweiten Berechtigungen wie z.B. das Verwalten der Benutzerrechte definieren.

3.2.12. UC11: Projekt/Ordner-spezifische Berechtigungen verwalten

Der Benutzer weist Rollen an Benutzer oder Benutzergruppen zu, die nur für ein bestimmtes Projekt oder Ordner gelten.

3.2.13. UC12: Rolle zu Benutzer zuweisen

Der Benutzer weist eine Rolle einem User zu.

3.2.14. UC13: Rolle zu Benutzergruppe zuweisen

Der Benutzer weist eine Rolle einer Benutzergruppe zu.

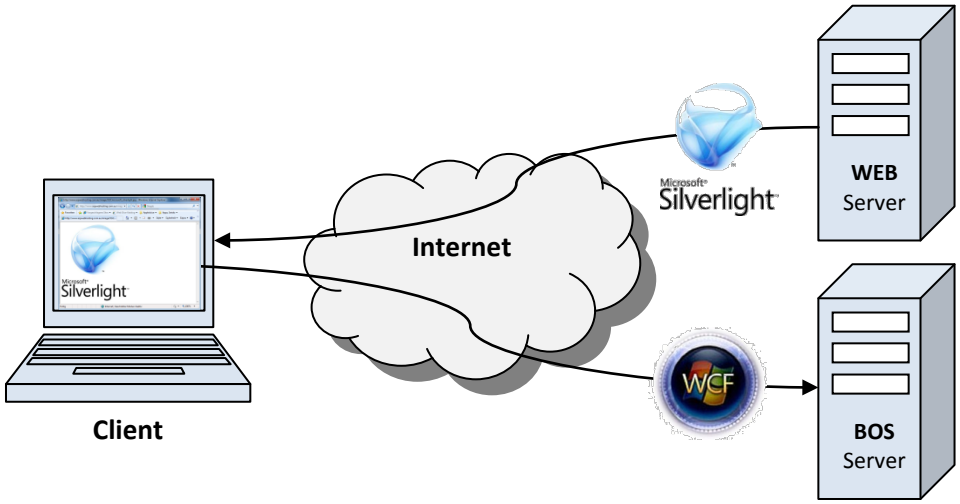
3.2.15. UC14: History bereinigen (nicht umgesetzt)

Der Benutzer bereinigt die Server-Logs.

4. Systemarchitektur

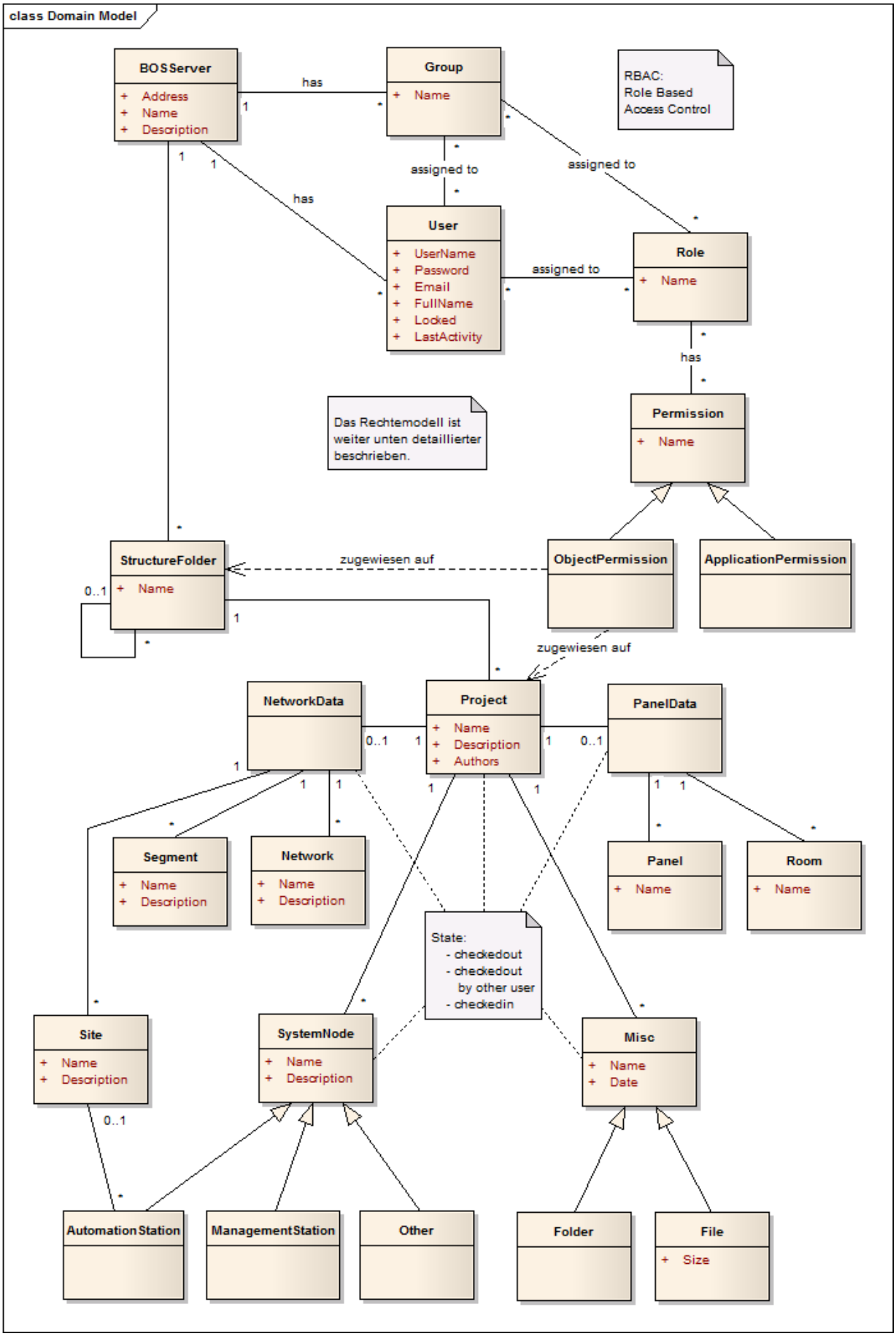
4.1. Übersicht

Die Silverlight Applikation *BOS-WebAdmin* wird auf einem ganz normalen Webserver gehostet. Durch Eingabe der URL im Browser des Clients wird die Applikation auf den Client Rechner heruntergeladen und gestartet. Nach der Auswahl eines BOS wird über WCF eine Verbindung zum BOS hergestellt. BOS und Webserver können auf demselben oder auf verschiedenen Rechnern installiert sein. Es findet keine direkte Kommunikation zwischen BOS und Webserver statt. Nur der Client stellt eine Verbindung zum BOS her.



4.2. Domainmodell

4.2.1. Diagramm



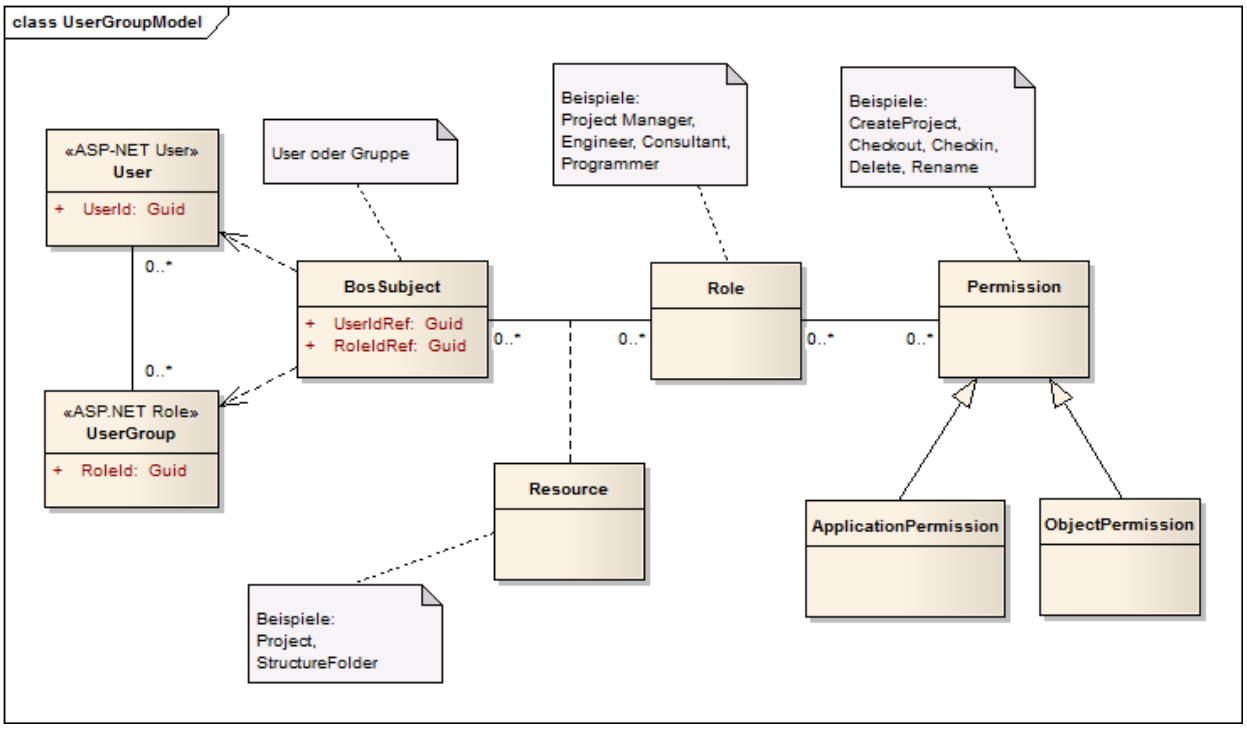
4.2.2. Beschreibung

Das Domainmodell kurz in Worten beschrieben.

- Das Security Modell entspricht in etwa dem RBAC, Role Based Access Control, mit dem Unterschied, dass als Subjekt entweder User oder Gruppen verwendet werden.
- Der *BOSServer* hat Benutzergruppen (*Group*) und *User*. Der *User* ist gewissen Gruppen zugewiesen. *Group* und *User* haben Rollen (*Role*) welche wiederum Berechtigungen (*Permission*) haben.
- Es gibt zwei Typen von Berechtigungen, *ObjectPermission* und *ApplicationPermission*. *ApplicationPermission* sind allgemeine Berechtigungen und *ObjectPermission* sind Berechtigungen auf *StructureFolder* oder *Project*. Solange für ein User oder eine Gruppe auf ein Objekt keine *ObjectPermission* vorhanden ist, gilt die *ObjectPermission* des Parents.
- Ein *BOSServer* hat mehrere Verzeichnisse (*StructureFolder*) welche wiederum Unterverzeichnisse (*StructureFolder*) oder Projekte (*Project*) haben.
- Das *Project* hat vier verschiedene Arten von Daten, Netzwerkdaten (*NetworkData*), Paneldaten (*PanelData*), Systemnode-Daten (*SystemNode*) und diverse Verzeichnisse oder Dateien (*Misc*).
- *NetworkData* und *PanelData* gibt es ein oder keinmal in einem Projekt, *SystemNode* oder *Misc* kann es beliebig viele geben. *Misc* ist entweder ein *Folder* oder ein *File*.
- *SystemNode* ist eine Automationsstation (*AutomationStation*) eine Managementstation (*ManagementStation*) oder ein anderer Systemnode (*Other*), z.B. Router.
- Das *Project*, *NetworkData*, *PanelData*, *SystemNode* und *Misc* können ausgecheckt sein und haben somit den Status "checkedout" oder "checkedin".
- Enitäten die individuell ein- oder ausgechecked werden können sind: *ProjectData*, *NetworkData*, *PanelData*, *SystemNodes* (Controller, Mgmt Station etc) oder *Misc* Daten.
- Ein Projekt kann mit all seinen zugehörigen Nodes zusammen ein- oder ausgecheckt werden.

4.3. Rechtemodell

Grundsätzlich wird einem User oder einer Gruppe für eine Ressource eine Rolle zugewiesen. Die Rolle beschreibt welche Berechtigungen auf dieses Objekt gewährt werden.



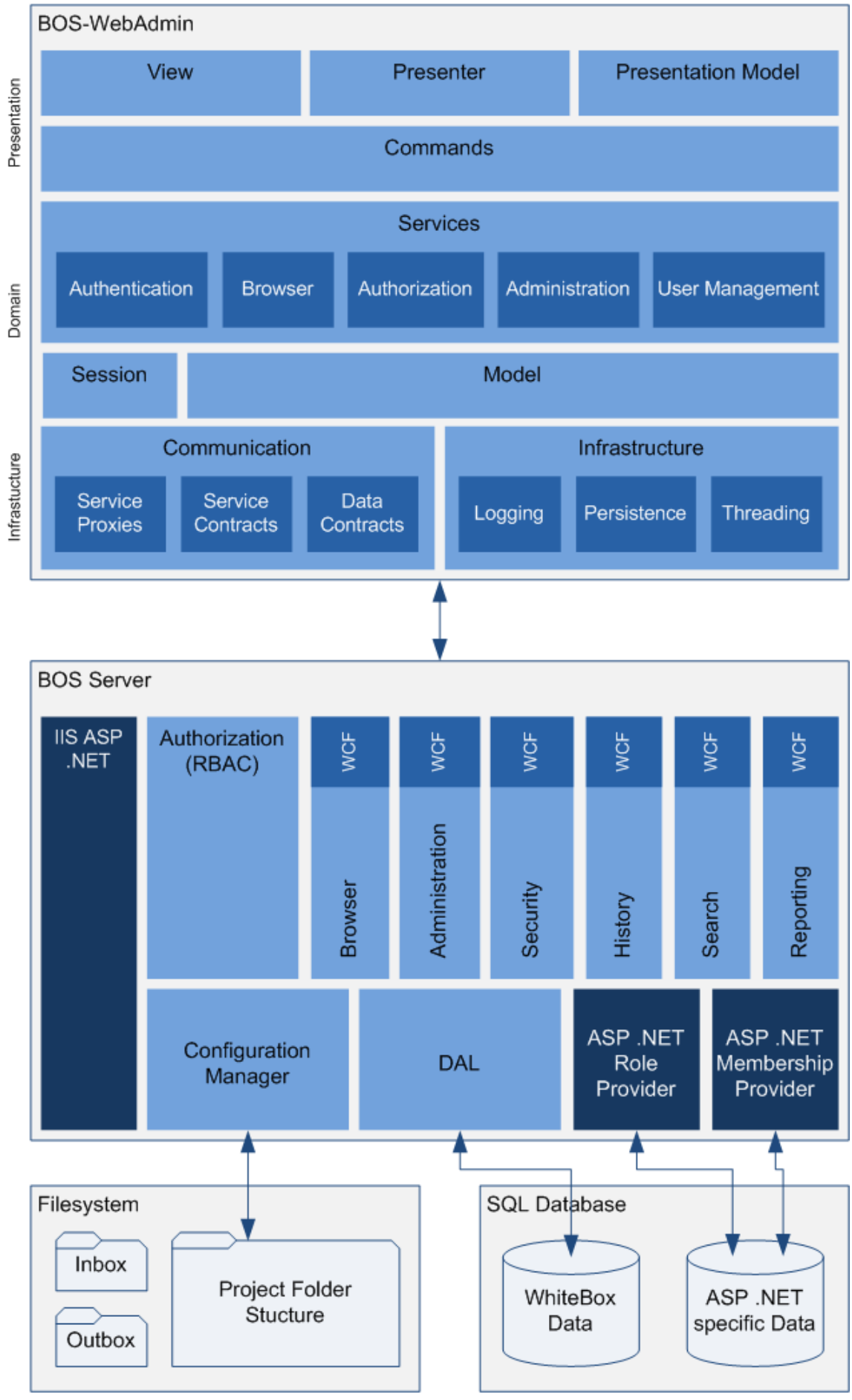
Es gibt zwei Arten von Berechtigungen:

- ApplicationPermission
  Allgemeine Berechtigungen welche die Applikation und nicht einen Node betreffen. Zum Beispiel: ManageSecurity, UseHistory oder UseSearch

- ObjectPermission
  Berechtigungen welche auf einen Node (Ordner oder Projekt) gesetzt werden. Zum Beispiel: CheckIn, CheckOut, GetLatest oder UndoCheckoutOther.

Die Resource im Diagramm entspricht einem Node (Ordner oder Projekt) und wird nur für die ObjectPermission benötigt.

4.4. Client-Server Architektur



4.5. Kommunikation mit WCF

4.5.1. Ausgangslage

Die Kommunikation zwischen XwpProjectManager als Client und dem BOS als Server, erfolgt ausschliesslich via WCF. Als Binding wird wsHttpBinding verwendet. Der BOS erlaubt die Kommunikation über HTTP und HTTPS. Bei HTTP wird nur Message-Security verwendet. Bei HTTPS wird dann zusätzlich auch noch Transport-Security hinzugeschaltet. Für letzteres wird aber ein gültiges Serverzertifikat benötigt.

4.5.2. Einschränkungen durch Silverlight

- **Silverlight unterstützt nicht den vollen WCF-Funktionsumfang**
  Bei Silverlight ist nur eine abgespeckte Version des .NET Frameworks verfügbar, in der unter anderem auch entscheidende Features bei der WCF-Implementierung weggelassen wurden. So wird z.B. nur basicHttpBinding als Verbindungsart unterstützt. Falls die Authentifizierungs-Features von basicHttpBinding benutzt werden, verlangt Silverlight 3 zwingend die Verwendung von HTTPS, damit die Credentials (Benutzername/Passwort) verschlüsselt übertragen werden.

- **Die Kommunikation in Silverlight funktioniert asynchron**
  Silverlight forciert bei der Kommunikation via WCF die Verwendung des Async-Patterns. D.h. es werden nur als asynchron definierte Servicefunktionen unterstützt. Das merkt man spätestens beim Versuch, die existierenden synchronen ServiceContracts zu verwenden. Denn in diesem Fall wirft Silverlight sofort eine Exception. Das bedeutet, dass die ServiceContracts, die serverseitig zur Verfügung gestellt werden, von Silverlight nicht ohne weiteres benutzt werden können. Clientseitig muss zuerst ein neuer ServiceContract erstellt werden, der die Serviceoperationen asynchron (Async-Pattern) verfügbar macht. Das Async-Pattern beschreibt, dass jede Serviceoperation durch eine Begin- und eine End-Methode ersetzt wird. D.h. serverseitig gibt es z.B. die Methode GetChildNodes(). Diese wird clientseitig in BeginGetChildNodes() und EndGetChildNodes() aufgeteilt. Dabei gibt die Begin-Methode ein IAsyncResult-Objekt zurück, auf dem geprüft werden kann, wann die Methode vollständig abgearbeitet wurde und das Resultat via End-Methode abgeholt werden kann.

4.5.3. Anpassungen der Serverschnittstelle

Damit der BOS-WebAdmin auf denselben BOS verbinden kann, wie der bestehende XwpProjectManager, braucht es einige serverseitige Anpassungen. Diese können in die vier folgenden Bereiche unterteilt werden:

- **WCF-Konfiguration**
  Die Service-Endpunkte des BOS-Servers verwendeten bisher ausschliesslich wsHttpBinding, welches aber von Silverlight 3 bekanntlich nicht unterstützt wird. Da die Verwendung von normalen Clients (XwpProjectManager) und Silverlight-Clients (BosWebAdmin) parallel funktionieren muss, müssen somit auch zwei verschiedene Endpunkte aufgesetzt werden. Da aber die meisten Sicherheitseinstellungen nicht für jeden Endpunkt, sondern nur pro Service gemacht werden können, und sich diese Einstellungen zwischen normalen Clients und Silverlight-Clients unterscheiden, müssen zwingend zwei Services definiert werden. Somit bleibt nichts anderes übrig, als die Service-Implementierung in eine separate Klasse auszulagern, und jeweils zwei Wrapper-Klassen zu erstellen, die je für einen Client (XwpProjectManager/BosWebAdmin) zuständig ist.
  In der Web.config Datei wird es so gelöst, dass je ein Service für den wsHttpBinding-Zugriff mit zwei Endpunkten für HTTP und HTTPS und ein Service für den basicHttpBinding-Zugriff mit einem Endpunkt für HTTPS konfiguriert werden.

- **Serviceschnittstellen**
  Einige der bestehenden Serviceschnittstellen des BOS haben den Instanzierungsmodus auf `InstanceContextMode.PerSession` gesetzt. Das heisst, dass die erstellte Instanz des Service für nachfolgende Methodenaufrufe dieselbe bleibt. Das basicHttpBinding unterstützt

aber keine Sessions und da Silverlight kein wsHttpBinding unterstützt, müssen die Serviceschnittstellen für Silverlight kopiert und auf `InstanceContextMode.PerCall` gesetzt werden. Dies bedeutet, dass z.B. die Browser-Nodes nicht mehr stückchenweise vom Service zurückgegeben werden können, sondern immer direkt alle in einem Methodenaufruf zurückgeben werden müssen.

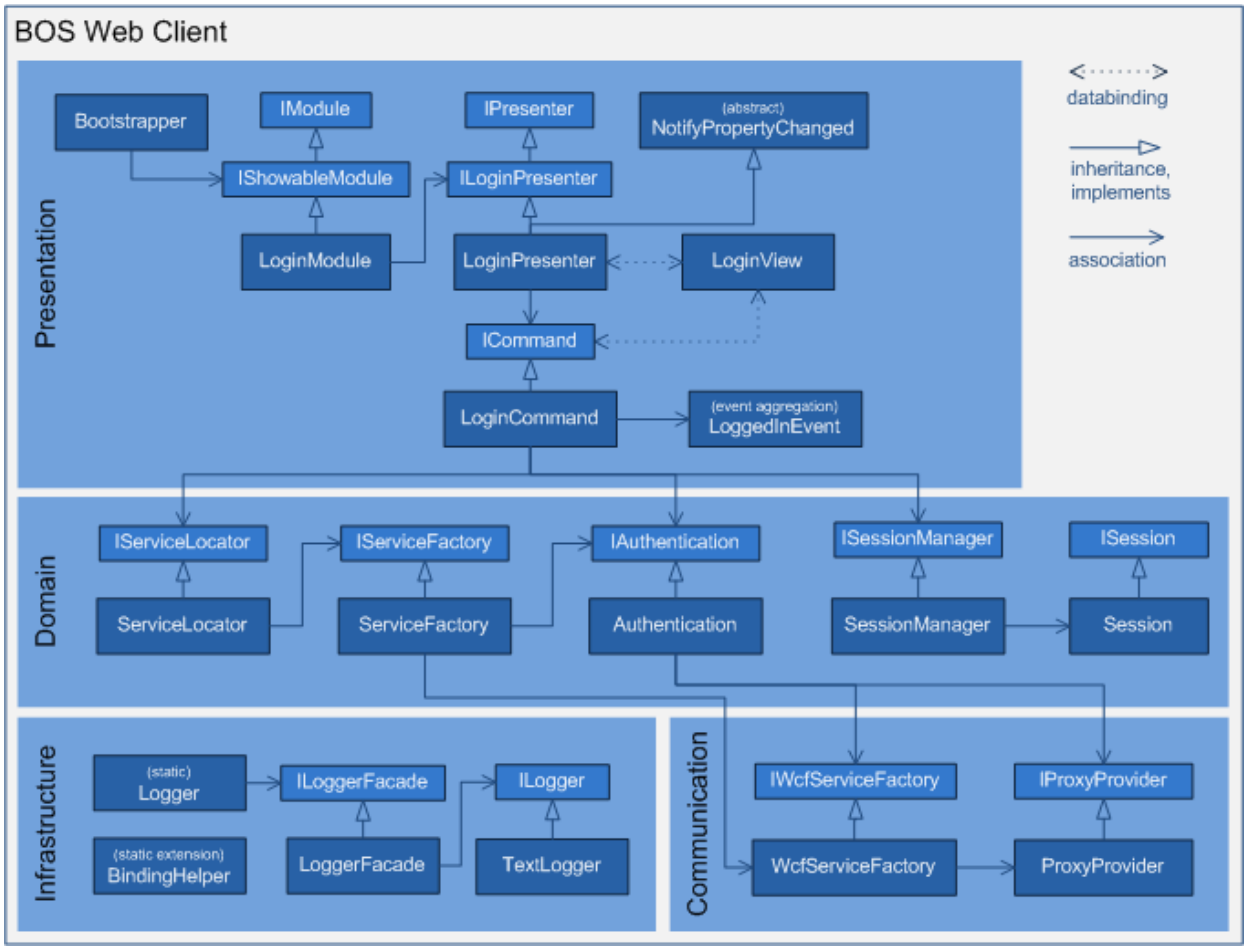
- **Dynamisches Service-Hosting**
  Die ServiceHostFactory-Klasse wird dazu benutzt, um darüber zu entscheiden, wie ein Service gehostet wird. Sie ist die erste Klasse, die vom IIS bzw. ASP.NET-Prozess aufgerufen wird, sobald eine Anfrage vom Client eintrifft. Die ServiceHostFactory wird vom bestehenden BOS genutzt, um dynamisch zwischen HTTP und HTTPS zu umzuschalten. Dies würde mit einer statischen Konfigurationsdatei nämlich nicht funktionieren, weil sich die Sicherheitseinstellungen zwischen HTTP und HTTPS unterscheiden, aber diese für einen Service nur einmal definiert werden können (im Gegensatz zu den Service-Endpunkten, von denen es pro Service mehrere geben kann).
  Mit Silverlight kommt jetzt ein dritter Verbindungstyp hinzu, der zwar auch über HTTPS verbindet, aber nicht wsHttpBinding sondern basicHttpBinding verwendet. Die ServiceHostFactory wurde nun so konfiguriert, dass sie beim Zugriff eines Silverlight-Clients nur HTTPS Verbindungen erlaubt. Zusätzlich kann in der Web.config Konfigurationsdatei ein Flag gesetzt werden, ob der Zugriff via Silverlight überhaupt erlaubt ist. Dies wird dann ebenfalls in der ServiceHostFactory überprüft.

- **Fehlerverhalten**
  Da die gesamte WCF-Kommunikation bei Silverlight über den Browser-Network-Stack läuft, werden FaultExceptions, die mit dem HTTP Code 500 (INTERNAL SERVER ERROR) gesendet werden, vom Internet-Browser unterschlagen. Um zu erreichen, dass die FaultExceptions vom Browser an Silverlight weitergegeben werden, muss der Server so konfiguriert werden, dass die Exceptions nicht mit dem HTTP Code 500, sondern mit dem HTTP Code 200 (OK) gesendet werden. Der BOS ist so konfiguriert, dass nur die Silverlight-spezifischen Service-Endpunkte von dieser Änderung beeinflusst werden. Der BOS wurde genau nach der Anleitung konfiguriert, die man in der MSDN unter folgendem Link findet:
  http://msdn.microsoft.com/en-us/library/dd470096(VS.96).aspx

5. Design

5.1. Übersicht

Wir verwenden eine Layer-Architektur mit Infrastructure-, Domain- und Presentation-Layer. Eine wie in der Composite Application Guidance beschriebene komplett modulare Architektur ist bei dieser eher kleinen Applikation nicht sinnvoll. Gewisse Ansätze davon werden jedoch verwendet.
Die gesamte BOS-WebAdmin-Klassenhierarchie wäre viel zu umfangreich, um sie übersichtlich darzustellen. Deshalb zeigt das folgende Diagramm nur die Klassenhierarchie für den Login-UseCase. Bei einem anderen UseCase würden sich zwar einige Klassen im Presentation-Layer ändern, aber sonst ändert sich jeweils nur der verwendete Service im Domain-Layer (beim Login wird der Authentication-Service benutzt).



Der Bootstrapper initialisiert die Module, welche wiederum die benötigten Presenter und Views initialisieren. Das Login-Modul initialisiert zusätzlich noch den *ServiceLocator*. Die Views bekommen ihren Presenter im Konstruktor und setze ihn gleich als *DataContext*. Presenter und View haben keine weiteren Beziehungen, sie sind nur durch Databinding verbunden. Die View Elemente sind gebunden an Properties im Presenter. Der Login-Button ist beispielsweise an den *LoginCommand* gebunden, welcher vom Presenter initialisiert wurde. Der *LoginCommand* bekommt über den *ServiceLocator* eine Instanz des *Authentication* Services und ruft darauf die benötigten Funktionen mit der Übergabe einer Callback-Funktion auf. Der Service führt dann die WCF-Funktionen asynchron aus und gibt danach das Ergebnis über die Callback-Funktion an den *LoginCommand* zurück. Im Beispiel des *LoginCommands* wird in der Callback-Funktion mit Event-Aggregation der *LoggedInEvent* gefeuert und alle die sich darauf registriert haben, werden notifiziert. Nach erfolgreichem Login werden eingeloggter User sowie der verbundene Server in einer Session gespeichert, die zu jeder Zeit über den *SessionManager* abgerufen werden kann.

5.2. Verwendung der Composite Application Library

Kurze Beschreibung der verwendeten Komponenten aus der *Composite Application Library (CAL)*.

5.2.1. View-Injection (Named-Locations)

Die *Named-Locations* vereinfachen das Aufteilen und Darstellen der Views stark. *Named-Locations* sind Platzhalter mit einem Namen im XAML und teilen das GUI in Bereiche auf. Im Code kann eine View in einer gewählten *Named-Location* dargestellt werden. Die View selbst muss somit nicht wissen, wo sie angezeigt wird.

5.2.2. Low Coupling (Event-Aggregation)

Um die Abhängigkeiten zu minimieren wird im Presentation-Layer die *Dependency-Injection* sowie *Event-Aggregation* verwendet.

5.2.3. Dependency-Injection (Constructor-Injection)

Damit möglichst viel mit Unittests getestet werden kann, verwenden wir nur beim Logger das *Singleton-Pattern*. Alle restlichen Objekte, welche global Verfügbar sein müssen, werden jeweils im Konstruktor übergeben. Constructor-Injection ist eine einfache Möglichkeit um das Singleton-Pattern elegant zu umgehen und wird im Presentation-Layer eingesetzt. Um im Domain- und Infrastructure-Layer keine Abhängigkeit zur CAL zu generieren, wird dort weder Constructor -Injection noch Event-Aggregation verwendet.

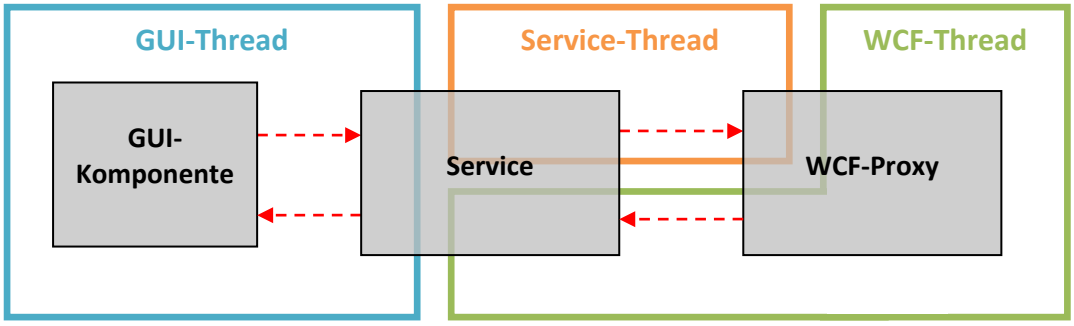
5.3. Separation of concerns

Damit nun die Presentation-Layer-Klassen nicht wissen müssen, wie man z.B. einen Service erstellen muss, wird das Erstellen der Services in eine *ServiceFactory* ausgelagert. So muss nur die *ServiceFactory* wissen, wie man einen Service instanziert.

5.4. Asynchrone Servicearchitektur

Silverlight unterstützt nur asynchrone WCF-Serviceaufrufe. Da diese Aufrufe über das Netzwerk gehen und daher auch relativ viel Zeit benötigen, macht es auch durchaus Sinn, dass während dieser Zeit nicht das gesamte GUI blockiert ist. Das Async-Pattern, welches bei den WCF-Services eingesetzt wird, ist aber etwas umständlich in der Benutzung. Zuerst muss immer die Begin-Methode aufgerufen werden. Danach muss gewartet werden, bis die Methode abgearbeitet wurde, um dann mit der End-Methode das Resultat abzurufen. Dabei darf das Warten nicht auf dem GUI-Thread ausgeführt werden, da dieser sonst blockiert wird.
Aus diesem Grund haben die Servicefunktionen, welche die WCF-Services kapseln, einen zusätzlichen Parameter, mit dem eine Callback-Funktion übergeben werden kann. Mit dieser Callback-Funktion wird dann jeweils am Ende der Servicefunktion das Resultat an den Aufrufer zurückgegeben. Die Callback-Funktion wird auch gleich wieder auf dem GUI-Thread aufgerufen, was den Vorteil hat, dass sich der Aufrufer in keinster Weise um irgendwelche Threads kümmern muss.

Folgendes Diagramm zeigt die schematische Darstellung der Komponenten und wie die Teile ihres Codes in den verschiedenen Threads ablaufen. Die Aufrufe zwischen den Komponenten sind mit roten Pfeilen dargestellt. Die Thread-Wechsel finden innerhalb der Komponenten statt.



5.5. Fehlerbehandlung

Serviceoperationen werfen nicht direkt Exceptions. Auftretende Exceptions werden in einem *ResultState*-Objekt gekapselt und via EventArgs-Objekt der Callback-Funktion übergeben. Dadurch erhält der Aufrufer der Serviceoperation im Erfolgs- wie im Fehlerfall ein Resultat zurück. Dies birgt den Vorteil, dass auf jeden Fehler individuell reagiert werden kann, ohne eine komplizierte globale Fehlerbehandlung. Um wichtige Fehler dem Benutzer mitzuteilen, werden diese einem „globalen“ *StatusHandler* übergeben. Der StatusHandler besitzt aber keine Fehlerbehandlung, sondern ist nur für die Darstellung der Fehlermeldungen verantwortlich. Neben Fehlermeldungen können auch ganz normale Informations- oder Statusmeldungen angezeigt werden.
Von der WCF-Schnittstelle werden verschiedene Exceptions geworfen, denen aber eigentlich immer ein Kommunikations-Problem zugrunde liegt. Daher werden alle diese Exceptions als ServiceUnavailableExceptions an den Serviceaufrufer zurückgegeben. Dies erlaubt eine einfachere Fehlerbehandlung im Presentation-Layer.

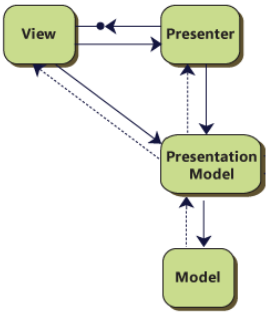
5.6. Sicherheit

Authentication- und Authorization-Service stellen Funktionalität zur Gewährleistung der Sicherheit bereit. Mit dem Authentication-Service wird beim Login überprüft, ob der Benutzer beim gewählten BOS-Server bekannt ist. Mit dem Authorization-Service wird überprüft, ob der Benutzer berechtigt ist, eine spezifische Funktion auf dem Server auszuführen. Diese Überprüfung wird vor allem dazu benutzt, um gewisse GUI-Bereiche für nicht berechtigte Benutzer auszublenden bzw. zu deaktivieren. Beim Aufruf der sicherheitskritischen Funktionen wird die Berechtigung auf dem Client aber nicht noch einmal überprüft, denn auf dem Server findet sowieso bei jedem Zugriff eine Berechtigungsprüfung statt.

6. User Interface

6.1. Übersicht

Von Silverlight wird man stark unterstützt bei der Separierung von Datenmodell und User Interface. Das User Interface wird rein deklarative in XAML definiert und ist durch Data-Binding mit den Presentern verbunden. Der *Presenter* stellt alle von der *View* benötigten Daten oder *Commands* als Properties zur Verfügung. Falls das *Model* nicht genau den darzustellenden Daten entspricht, wird zwischen *View* und *Model* noch ein *Presentation Model* verwendet. Entspricht in etwa dem *Model-View-ViewModel* (MVVM) Pattern.



Das GUI selbst muss auch anders konzipiert werden als beim *XwpProjectManager*.

- ***Keine Kontextmenüs:*** Alle Operationen können mit einem normalen Mausklick auf einen Button oder ein kleines Icon, wie z.B. beim „Undo Check-out“, ausgeführt werden.

- ***Keine Unterfenster***: Die einzelnen Views sind in eigenen Tabs und können so gewechselt werden. Es gibt die drei Haupttabs: Projects, Security und Logs, welche Untertabs für die jeweiligen Aufgabenbereiche haben.

6.2. Einschränkungen durch Silverlight

Weak-References
Die Event-Aggregation-Klassen aus der CAL (Composite Application Library) unterstützen weak references. Es wäre möglich sich auf einen Event zu registrieren und anzugeben, dass diese Beziehung alleine den Objektgrafen nicht am Leben erhalten soll. In Silverlight werden die 'weak references' nicht unterstützt und deshalb können wir diese Möglichkeit auch nicht benutzen.

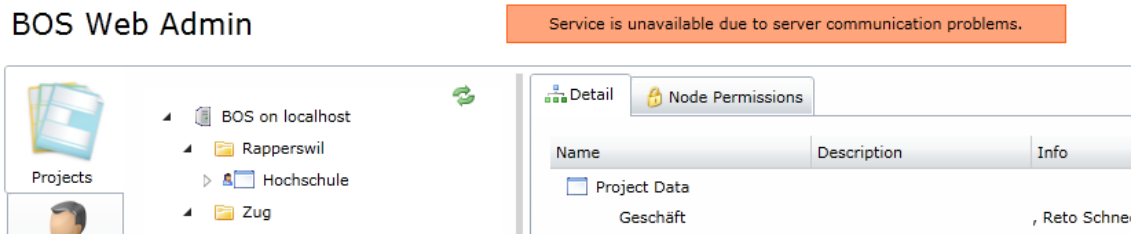
Buttons auf Commands binden
In WPF können Buttons direkt im XAML an Commands gebunden werden. Das CanExecute() wird dabei mit den Enabled/Disabled verknüpft und der Klick-Event mit dem Execute(). Diese Möglichkeit wurde im Silverlight weg gelassen. Jedoch gibt es in der CAL eine Klasse, die genau diese Möglichkeit wieder anbietet. Das war eigentlich schon Grund genug um gewisse DLL's der CAL zu verwenden.

Databinding
Silverlight unterstützt in vielen Bereichen weniger als WPF, so existiert zum Beispiel bei der TreeView kein Property für das selektierte Element. Das bedeutet, dass mit einfachem Databinding vom Presenter her nicht herausgefunden werden kann, welches Element selektiert ist. Damit der Presenter jetzt nicht komplett von der View abhängig ist und sich direkt für den *OnSelectedItemChanged*-Event registriert, verwenden wir in solchen Fällen ein Interface mit genau den Events und Properties, welche der Presenter von der View benötigt. Der Presenter bekommt in diesem Fall mit dem Konstruktor eine View, welche das Interface implementiert und setzt sich selbst über das Property *DataContext* der View als Datenkontext.

6.3. Umsetzungs-Highlights

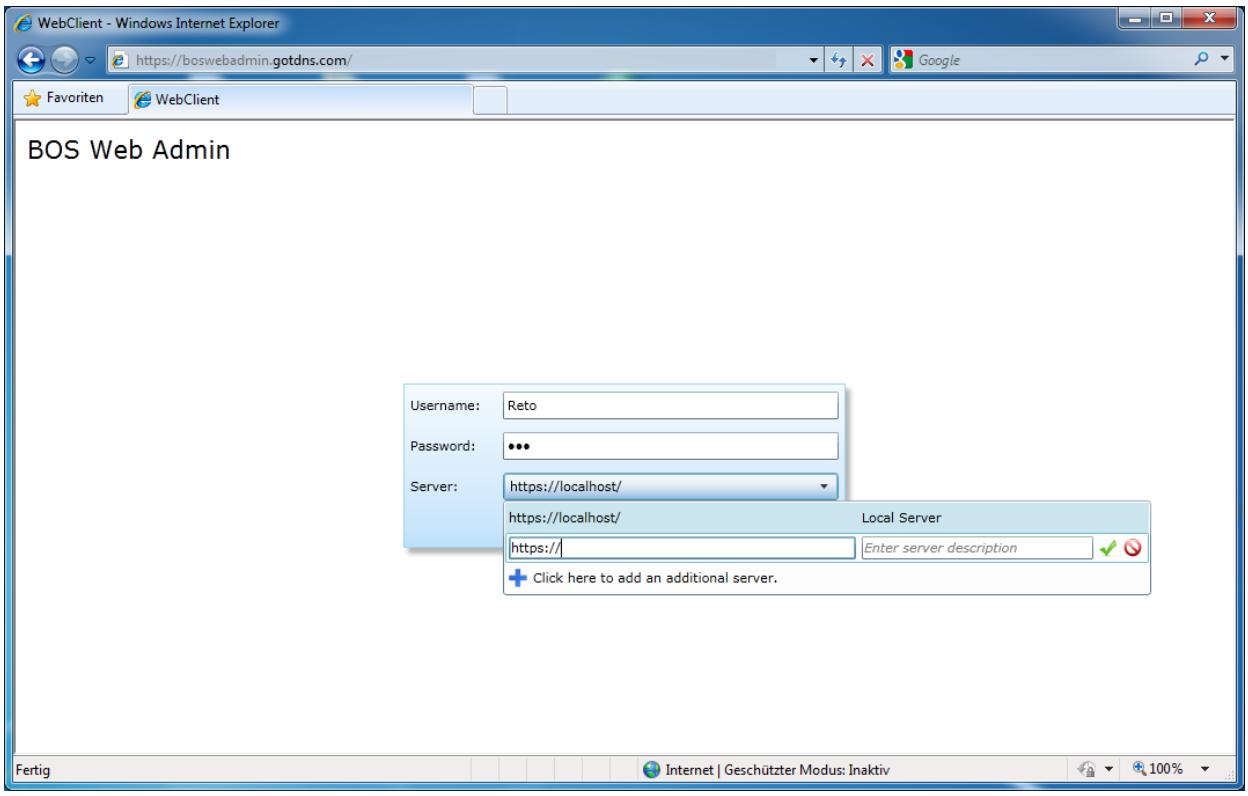
Allgemeine Highlights zur Umsetzung des User-Interfaces.

- Alle benötigten Daten werden im Hintergrund inkrementell, bzw. on-demand geladen.
- Alle Operationen werden asynchron ausgeführt, damit das User-Interface jederzeit reaktionsfähig bleibt.
- Damit der Benutzer nicht durch Popup-Dialoge behindert wird, werden Status- und Fehlermeldungen direkt im Menübereich auf der Hauptseite angezeigt. Um wichtige von weniger wichtigen Meldungen zu unterscheiden, werden sie in unterschiedlichen Farben angezeigt, deren Bedeutung vom Benutzer intuitiv verstanden wird.



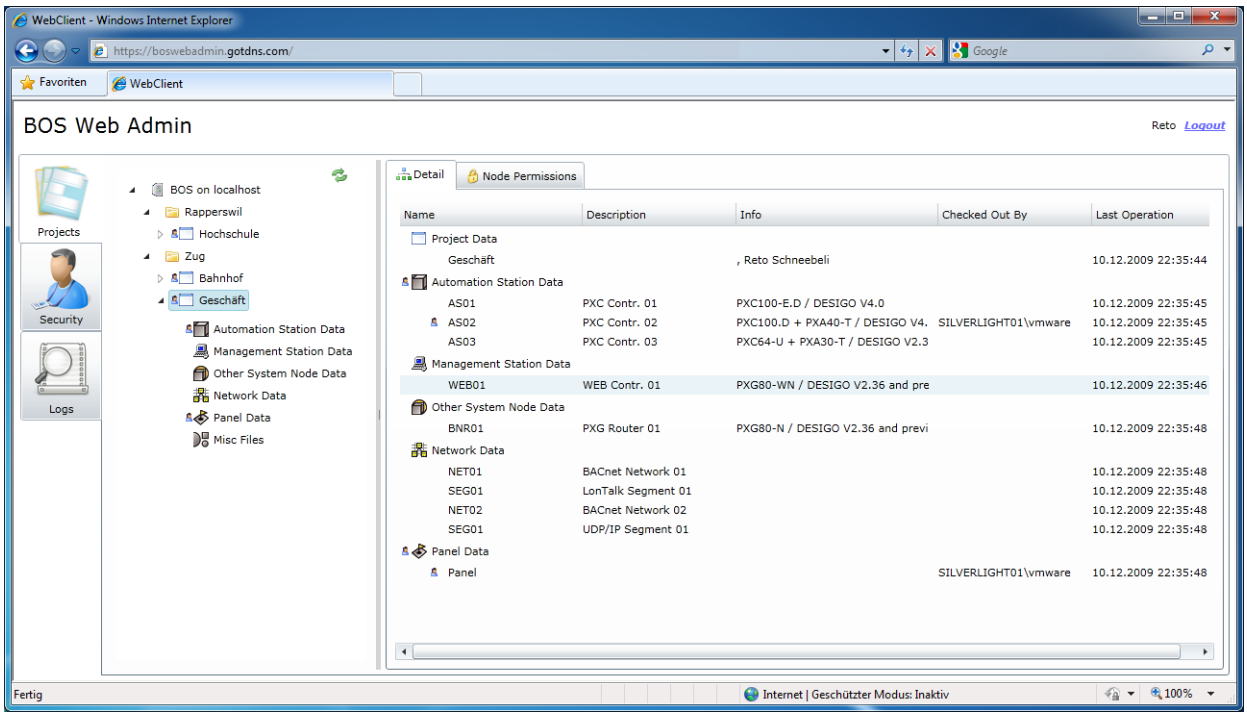
6.3.1. Custom ComboBox für Server-Auswahl

Im Login-Bereich müssen Server hinzugefügt, bearbeitet oder gelöscht werden können. Um keine Unterfenster zu benötigen, wurde eine bearbeitbare *ComboBox* umgesetzt. Beim Öffnen der *ComboBox* erscheint zuunterst immer ein *Button*, um einen neuen Server hinzuzufügen. Wenn man mit der Maus auf einen bestehenden Server kommt, erscheinen auf der rechten Seite zwei *Buttons* um den Server zu bearbeiten oder zu löschen. Sobald man in den editier-Modus wechselt, werden anstatt *Labels* zwei bearbeitbare *Textboxen* für die Adresse und die Beschreibung angezeigt. Die vorgenommene Änderung kann gespeichert oder verworfen werden.



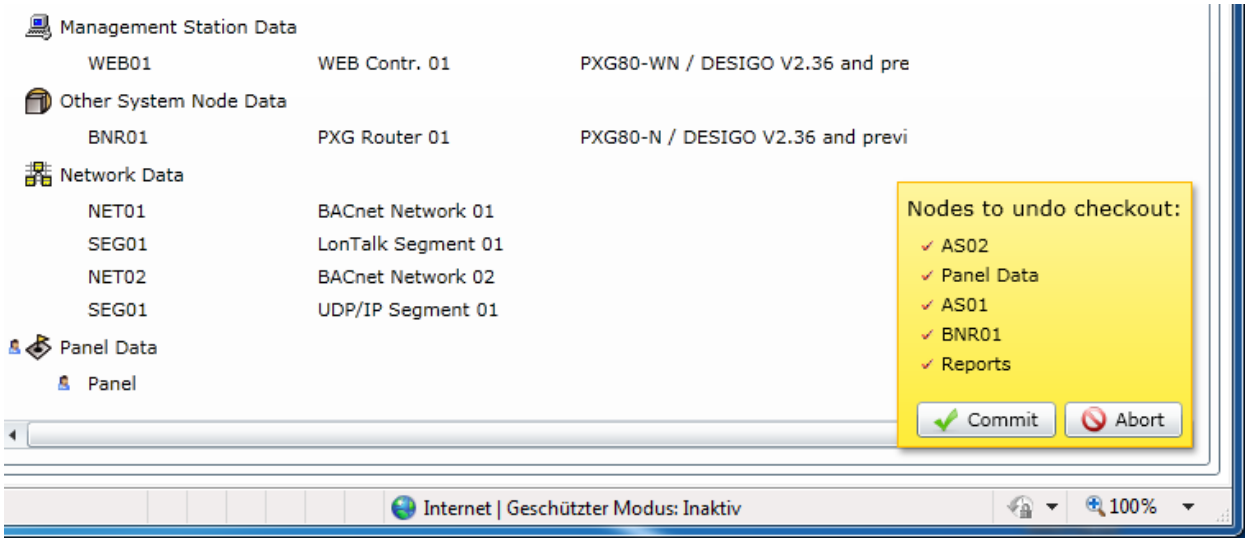
6.3.2. Intelligenter Browser-Tree mit Detail-View

Um den Projektbaum übersichtlich browsen zu können und trotzdem immer alle wichtigen Informationen sichtbar zu haben, wurde ein *Master-Detail* Dialog realisiert. Auf der linken Seite ist der komplette Projektbaum, welcher mit *Lazy-Loading* immer gerade die zusätzlich benötigten Daten lädt. Auf der rechten Seite werden jeweils zum selektierten Node alle wichtigen Detailinformationen angezeigt. Die Projektbaum-Ansicht ist mit dem Datei-Explorer vergleichbar.



6.3.3. „Prepare and Commit“ für Undo-Checkout

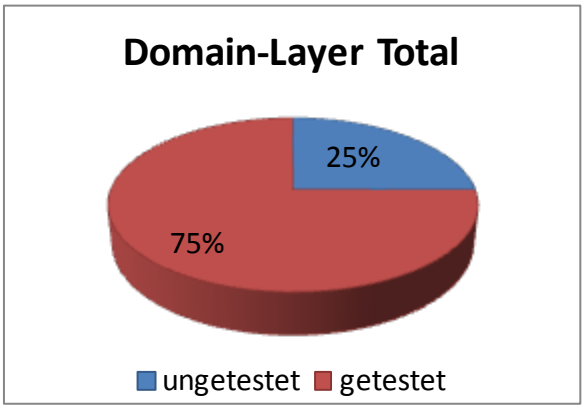
Heiklere Operationen wie das forcierte *„Undo Check-out“* können nicht einfach nach einem simplen Mausklick ohne Vorwarnung durchgeführt werden. Damit nicht bei jedem Node eine „Möchten Sie wirklich..“-Meldung angezeigt werden muss, wurde ein *Overlay-Window* realisiert. Es sammelt zuerst alle Nodes für die ein *„Undo Check-out“* ausgeführt werden soll und stellt sie übersichtlich in einem kleinen Fenster dar. Der Benutzer hat dann die Möglichkeit noch einmal zu kontrollieren, abzubrechen oder auszuführen.



7. Qualitätssicherung

7.1. Unit Tests

Als Testing-Framework wird NUnit 2.5 verwendet und für die Analyse der Testabdeckung NCover. Um eine hohe Qualität zu erreichen, wurde der Domain-Layer zu einem grossen Teil durch Unittests abgedeckt. Beim kompletten Domain-Layer konnte eine Testabdeckung von 75% erreicht werden. Wenn man nur die Serviceoperationen betrachtet, sind es sogar 97%.



Da alle Daten für das User-Interface von den *Presentern* kommen und nur durch Data-Binding mit den *Views* verbunden sind, kann auch der Presentation-Layer zu einem grossen Teil mit Unittests abgedeckt werden. Aus Zeitgründen musste leider auf die Implementierung dieser Tests verzichtet werden.

7.2. Systemtests

Beim Prototyp, Alpha-Release, Beta-Release und Final-Release wurde jeweils ein kompletter manueller Systemtest durchgeführt. Der Systemtest beinhaltete immer die bis dahin implementierten UseCases. Allfällige Abweichungen oder Fehler werden im Systemtest-Dokument festgehalten.

7.3. Codequalität

Für die Entwicklung mit C# halten wir uns an die Siemens-Programmierrichtlinien, welche weitgehend mit den von Microsoft veröffentlichten *.NET Naming Guidelines* (http://msdn2.microsoft.com/en-us/library/xzf533w0(vs.71).aspx) übereinstimmen. Die Namensgebung im Code wird strikt in Englisch gehalten. Alle Klassen, Interfaces, public Methoden und Properties müssen mit Kommentaren versehen werden. Damit lässt sich mit Doc-O-Matic eine Interface-Dokumentation generieren.

8. Technologiestudium

8.1. Silverlight 3

Silverlight ist eine Erweiterung für Webbrowser, welche die Ausführung von *Rich Internet Applications (RIA)* ermöglicht. Die Applikationen werden wie bei *WPF (Windows Presentation Foundation)* im *Visual Studio* mit C# entwickelt. Das User-Interface ist gut vom restlichen Code entkoppelt und wird mit *XAML (Extensible Application Markup Language)* z.B. im *Expression Blend* erstellt. Microsoft hat sehr darauf geachtet, das Silverlight-Framework möglichst schlank zu halten. Die im Vergleich zu WPF fehlenden Klassen oder Funktionen sind unter 'Einschränkungen durch Silverlight' in den Kapiteln 4.5.2 und 6.2 beschrieben. Viele Limitierungen werden jedoch durch das *Silverlight Toolkit* von Codeplex behoben.

Im Vergleich zur GUI-Programmierung mit *WinForms* stösst man beim Erstellen des User-Interfaces mit *XAML* auf ganz andere Probleme. Viele Fehler treten erst zur Laufzeit auf und nicht bereits beim Kompilieren. Ein Beispiel dafür ist das Binding auf Property-Namen, diese werden beim Kompilieren nicht überprüft. Da praktisch alles mit Data-Binding gemacht wird, können ziemlich viele Probleme auftreten, wenn z.B. Properties umbenennt werden. Im grafischen Bereich bietet Silverlight viel mehr Möglichkeit. Effekte wie Farbverläufe, Transparenz, Schatten und noch viele mehr, können sehr einfach auf alle GUI-Elemente angewendet werden, genauso wie jegliche Transformationen (Rotieren, Skalieren, ...).

8.2. Composite Application Guidance

Die *Composite Application Guidance* für WPF und Silverlight gehört zu den Microsoft *patterns & practices* und bietet viel Unterstützung beim erstellen von modularen Applikationen. Sie beinhaltet eine Vielzahl von Design-Patterns, eine Library *(Composite Application Library CAL)*, einige *hands-on labs* und komplette Beispiel-Implementierungen. Weitere Informationen dazu sind im Dokument Studie_CompositeApplicationGuidance.doc zu finden, hier nur eine knappe Zusammenfassung.

8.2.1. Unity Container (Dependency-Injection)

Abhängigkeiten werden nicht explizit instanziert, sondern nur in den Klassen über Parameter im Konstruktor, oder Properties definiert. Die Instanzierung selbst wird vom *UnityContainer* der CAL übernommen.

Dafür müssen die Klassen zu den jeweiligen Interfaces registriert werden:

```
unityContainer.RegisterType<ILoginPresenter, LoginPresenter>();
```

Hier besteht auch gerade die Möglichkeit zu definieren, dass immer nur eine Instanz existieren soll:

```
unityContainer.RegisterType<IServiceLocator, ServiceLocator>(new
ContainerControlledLifetimeManager());
```

Eine neue Instanz eines Interfaces erhält man nun auf foldende Weise:

```
ILoginPresenter loginPresenter = unityContainer.Resolve<ILoginPresenter>();
```

Alle Parameter der nun instanzierten Klasse werden mit *dependency-injection* durch den *UnityContainer* injiziert. Falls die zu injizierenden Typen so definiert sind, dass immer nur eine Instanz existieren soll, wird diese Instanz injiziert, sonst immer eine neue.

8.2.2. Event Aggregation

Der *EventAggregator* ist primär ein Container für Events, welcher das entkoppeln von *Publisher* und *Subscriber* erlaubt. Bei der Event-Aggregation müssen alle Beteiligten nur denselben Event, abgeleitet von *CompositePresentationEvent* und den *EventAggregator* kennen. Eine weitere Abhängigkeit besteht nicht.

Abonnieren eines Events (subscribe):
```
LoggedOutEvent loggedOutEvent = eventAggregator.GetEvent<LoggedOutEvent>();
loggedOutEvent.Subscribe(LoggedOutEventHandler, ThreadOption.UIThread, true);
```

Feuern eines Events (publish):
```
eventAggregator.GetEvent<LoggedOutEvent>().Publish(new EmptyEventArgs());
```

8.2.3. Regions (View-Injection)

Das Layout des GUI's kann grob mit benannten Regionen *(Named-Locations)* definiert werden. Die einzelnen Views können vom Code aus zur Laufzeit in diesen Regionen angezeigt oder entfernt werden.

Definieren einer Region im XAML:
```
<ItemsControl cal:RegionManager.RegionName="MainRegion" />
```

Anzeigen einer View in der definierten Region:
```
regionManager.Regions["MainRegion"].Add(loginView);
```

8.3. Moq (Mocking Library for .NET 3.5)

Moq (gesprochen „Mock-you“ oder einfach „Mock“) ist eine Mocking-Library für .NET und nutzt die neuen Features von .NET 3.5 und C# 3.0. Moq hebt sich gegenüber den meisten anderen Mocking-Libraries durch folgende Punkte hervor:

- Typensicher und dadurch Refactoring-freundlich
- Nutzt Extension Methods und Lambda Expressions von C# 3.0 / .NET 3.5
- Kein Record/Replay
- Unterstützt Silverlight

Wir haben Moq ausgewählt, weil es durch die Nutzung der neuen C# 3.0-Features einfach die intuitivste Mocking-Library ist. Die Einarbeitungszeit betrug durch die sehr schlanke API gerade mal eine Stunde. Ein weiteres Kriterium war natürlich auch die Kompatibilität mit Silverlight, welche Moq vollständig erfüllt. Moq folgt nicht dem Record/Replay-Modell anderer Mocking-Libraries. Dadurch ist die Benutzung „straightforward“ und benötigt kein grösseres Verständnis des Mocking-Konzeptes.

Moq wurde mit Rhino.Mocks und NMock verglichen. NMock ist leider nicht typensicher und die Kompatibilität mit Silverlight konnte nicht vollständig geklärt werden. Rhino.Mocks ist eine state-of-the-art Mocking-Library mit Silverlight-Unterstützung aber durch das Record/Replay-Modell nicht so intuitiv in der Benutzung wie Moq.

Mehr Informationen zu Moq gibt es auf http://code.google.com/p/moq.

9. Schlussfolgerungen

9.1. Zusammenfassung

Während der Studienarbeit wurde der BOS-WebAdmin zur Administration eines BOS-Servers entwickelt. Die Applikation wurde ausgiebig getestet und läuft stabil in den gängigsten Browsern. Die wichtigsten und gebräuchlichsten Administrationsfunktionen wurden im BOS-WebAdmin implementiert und funktionieren Fehlerfrei.
Aus Sicht des Designs konnte eine saubere und stabile Architektur ausgearbeitet werden. Bei der Umsetzung der gewählten Dreischichtenarchitektur kamen verschiedene Patterns und Technologien zum Einsatz. So werden im Presentation-Layer u.a. Lösungen der Composite Application Guidance eingesetzt. Im Domain-Layer wurde das Prinzip einer asynchronen SOA mit Service-Locator umgesetzt. Dadurch kann der BOS-WebAdmin sehr einfach mit neuen Services erweitert werden. Da konsequent Interface und Implementation getrennt wurden, können alle drei Schichten mit Unit-Tests getestet werden.

Für die Umsetzung des User Interfaces wurde der existierende *XwpProjectManager* als Vorlage genommen, da die User-Experience vergleichbar sein sollte. Dies wurde im BOS-WebAdmin auch grösstenteils erreicht, obwohl in vielen Bereichen neue Lösungen gefunden werden mussten, die speziell auf Silverlight zugeschnitten sind. Denn Microsoft hat sich bei der Umsetzung des RIA-Konzeptes in gewissen Bereichen mehr für Plattformunabhängigkeit als für die volle Unterstützung einer Desktopumgebung entschieden. Da das *Silverlight*-Framework sehr schlank ausgefallen ist, mussten auch einige GUI-Komponenten selbst implementiert werden. Ein Beispiel dafür ist die ComboBox für die Serverauswahl im Login-Screen. Aber auch die Detail-View, die beim Browsen der Projekte eingesetzt wird, musste selbst implementiert werden. Ebenfalls sehr elegant gelöst, ist die Anzeige der Statusmeldungen, wie sie für Fehler, Warnungen, Bestätigungen oder sonstige Informationen erscheint. Anders als bei Desktopapplikationen üblich, erscheinen wichtige Informationen nicht als Popup, sondern werden immer am selben Ort direkt im Hauptfenster angezeigt.

Zu Beginn des Projektes gab es die Anforderung, dass die bestehende WCF-Serviceschnittstelle auf dem Server möglichst nicht erweitert werden muss. Während der Problemanalyse stellte sich heraus, dass die Serverschnittstelle jedoch zwingender Änderung bedarf, da *Silverlight 3* nicht den vollen WCF-Funktionsumfang unterstützt. Insbesondere für die Umsetzung des geforderten Sicherheitskonzeptes (Zertifikate, verschlüsselte Datenübertragung, etc.) mussten *Silverlight*-spezifische Anpassungen vorgenommen werden. Aber auch der parallele Betrieb von BOS-WebAdmin und XwpProjectManager erforderte Änderungen am BOS-Server.

Bei den Qualitätssicherungsmassnahmen wurde ganz auf Unit-Testing mit NUnit gesetzt. Das Testen wurde aber durch die asynchrone Servicearchitektur und die vielen Remotezugriffe via WCF stark erschwert. Um trotzdem vernünftige Unit-Tests schreiben zu können, war die Verwendung eines Mocking-Frameworks unerlässlich. Moq war hierfür die perfekte Lösung, da sie sehr einfach zu benutzen und mit Silverlight 3 vollständig kompatibel ist.

9.2. Beurteilung der Resultate

Der wichtigste Punkt während der gesamten Projektdauer war ohne Zweifel die Fertigstellung des Prototypen zur Projekt-Halbzeit. Mit diesem Prototypen, der über alle Layer funktionsfähig war, wurde der Grundstein für das Gelingen des Projektes gelegt. Denn in den ersten sieben Wochen gab es einige grosse Probleme, mit denen wir zu kämpfen hatten. Ursprünglich war man davon ausgegangen, dass Silverlight das komplette WCF-Framework enthält. Während der Analyse stellte sich aber heraus, dass Silverlight nur einen Teil der WCF-Plattform unterstützt und mit der bestehenden Serverschnittstelle auf dem BOS nicht kompatibel ist. Viele Arbeitsstunden flossen dann in die Lösung dieser WCF-Kompatibilitätsprobleme auf Client wie auf Serverseite. Im Nachhinein muss man sagen, dass für solche Probleme definitiv zu wenig Zeit eingeplant wurde. Zum Glück konnten einige weniger wichtige UseCases weggelassen werden, aber ohne die geleisteten „Überstunden“ hätte dieser Mehraufwand

nicht bewältigt werden können. Dennoch können solche Risiken bei komplexen Projekten, auch bei sorgfältiger Planung, nicht völlig ausgeschlossen werden. Bei einer so kurzen Projektdauer bedeutet jeder Mehraufwand schnell auch eine Verzögerung des Projektes. Da wir aber bei den zu implementierenden UseCases recht flexible Vorgaben hatten, konnten wir dort gut Abstriche machen.

Abstriche bei den UseCases haben wir aber nicht nur wegen den WCF-Problemen gemacht, sondern auch um die allgemeine Qualität der Software zu steigern. Zuerst wurde viel Zeit in die Ausarbeitung eines sauberen Designs gesteckt. Danach ist ebenso viel Zeit in die Detailarbeiten am User-Interface geflossen. Beides hat sich unserer Meinung mehr als ausbezahlt. Was nützt es, so viel Funktionalität wie nur möglich einzubauen, wenn die Anwendung danach nicht bedienbar ist. So sind doch gerade Präsentation und Benutzerfreundlichkeit entscheidende Gründe dafür, wieso es überhaupt Silverlight gibt. Und weil das Studium der Technologie Silverlight ein wesentlicher Bestandteil dieser Studienarbeit ist, wollten wir auch zeigen, wie man diese neue Technologie in der Praxis sinnvoll nutzen kann.

Mit einer serviceorientieren Architektur in Kombination mit einem Service-Locator wurde die Anforderung eines einfach erweiterbaren Designs erfüllt. Das sauber ausgearbeitete Design erlaubt es in Zukunft, die verschiedensten Services aller Art hinzuzufügen, ohne die bestehende Architektur ändern zu müssen. Und weil die WCF-Schnittstelle vollständig „gewrapt“ ist, müssen neue Services nicht WCF-basiert sein. Für die bestehenden Services könnte die jetzige Kommunikationsmethode auch einfach durch eine andere ersetzt werden.

Die Codequalität wurde durch einhalten der Siemens Coding-Guidlines, durch extensive Nutzung von XML-Kommentaren und nicht zuletzt auch durch Code-Reviews hoch gehalten. Ein Tool zur automatischen Überprüfung der Richtlinien wie z.B. FxCop wurde nicht eingesetzt. Dies wäre noch eine zusätzliche Verbesserung für die Zukunft.
Qualitätssicherungs-Massnahmen wie das Durchführen von Unit- und Systemtests wurden erfolgreich eingesetzt. Die Fehlerrate bei den Systemtests war durchgängig sehr tief. Meist waren es nur ein paar Schönheitsfehler im User-Interface, die sich meist erst durch vermehrte Benutzung feststellen liessen. Im Domain-Layer gab es fast keine Fehler, den dort konnten die meisten bereits bei den Unit-Tests erkannt und behoben werden.

Beim Projektstart hatten wir noch die Idee, ein Reporting-Service im BOS-WebAdmin zu implementieren, mit dem verschieden Reports generiert werden könnten. Leider blieb dafür überhaupt keine Zeit, solch ein Reporting-Feature zu analysieren. Das ist aber auch nicht weiter schlimm, denn bis in die Projektplanung schaffte es das Reporting sowieso nie. Nichtsdestotrotz bleibt es eine spannende Idee für die Zukunft, denn Reporting-Funktionen werden immer wieder gefordert und da würde sich Silverlight sicher gut dafür eignen.

Während dem Testen mit den echten Projektdaten des Schweizer BOS-Servers haben wir festgestellt, dass wegen der grossen Menge an Projekten, sehr viele WCF-Anfragen fast gleichzeitig an den Server geschickt werden. Dies führt auf der einen Seite zu Timeouts auf der Clientseite, weil die Anfragen automatisch gepuffert werden und das Timeout z.T. noch abläuft, bevor die Anfragen überhaupt an den Server geschickt werden. Andererseits kann durch die vielen gleichzeitigen Anfragen von potentiell duzenden von Clients auch den Server überlasten. Daher wäre es sinnvoll, nur die Daten zu laden, die auch wirklich benötigt werden. Darüber hinaus könnten Anfragen, die gar nicht mehr benötigt werden, weil der Benutzer z.B. gleich auf den nächsten Ordner oder das nächste Projekt springt, abgebrochen werden, bevor sie an den Server geschickt werden. Dadurch liessen sich die Effizienz sowie die Performance des gesamten Systems nochmals beträchtlich erhöhen.

9.3. Ausblick

Der BOS-WebAdmin wird neben dem Management auch der Entwicklungsabteilung und einigen *DESIGO* Ländervertretungen vorgestellt. Bei grossem Anklang wird er möglicherweise Bestandteil der Version 5 von *DESIGO XWORKS plus*. Für einen produktiven Einsatz müsste aber noch eine Lösung für die Überflutung des Servers mit WCF-Anfragen implementiert werden. Auch der Funktionsumfang

müsste möglicherweise noch um einige wichtige Funktionen wie z.B. „Suchen“ oder „History ansehen“ erweitert werden, um den realen Anforderungen im produktiven Einsatz gerecht zu werden.

Projekt: BOS-WebAdmin

Projektplan

[Dokumentstruktur basiert auf RUP „Software Development Plan“]

0. Dokumentinformationen

0.1. Änderungsgeschichte

Datum	*Version*	*Änderung*	*Autor*
16.09.2009	0.1	Initial Version	Reto Schneebeli
18.09.2009	0.2	Einführung, Projekt Übersicht, Meilensteine, Arbeitspakete, Risikoanalyse	Reto Schneebeli
20.09.2009	0.3	Texte überarbeitet: Projekt Übersicht, Infrastruktur, Qualitätssicherung	Kaspar Fenner
21.09.2009	0.4	Arbeitspakete überarbeitet/erweitert, Texte überarbeitet	Reto Schneebeli
22.09.2009	0.5	Projektorganisation , für Review überarbeitet	Kaspar Fenner
23.09.2009	0.6	Review-Kommentare einarbeitet	Reto Schneebeli
03.11.2009	0.7	Implementation in Themen aufgeteilt und Zeitschätzung angepasst	Kaspar Fenner

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 4
2.2. Annahmen und Einschränkungen 4
3. Projektorganisation 5
3.1. Organisationsstruktur 5
3.2. Externe Schnittstellen 5
4. Management Abläufe 5
4.1. Projekt Kostenvoranschlag 5
4.2. Projektplan 5
4.2.1. Zeitplan 5
4.2.2. Iterationsplanung / Meilensteine 5
4.2.3. Besprechungen 6
4.2.4. Software Releases 6
5. Risiko Management 6
6. Zeitschätzungen der Arbeiten 7
6.1. Projekt Management 7
6.2. Requirements 7
6.3. Analyse 7
6.4. Design 7
6.5. Implementation 8
6.6. Test 8
6.7. Dokumentation 9
6.8. Studium Technologien (über ganze Projektdauer) 9
6.9. Sitzungen (über ganze Projektdauer) 9
6.10. Qualitätssicherung (über ganze Projektdauer) 9
7. Infrastruktur 10
7.1. Räumlichkeiten 10
7.2. Hardware 10
7.3. Software 10
7.3.1. Entwicklungstools 10
7.3.2. Dokumentation 10
7.3.3. Versionsverwaltung 10
7.3.4. Sonstiges 10
8. Qualitätsmassnahmen 10
8.1. Teamarbeit und Kommunikation 10
8.2. Dokumentation und Planung 10
8.3. Reviews 11
8.4. Backup/Sicherung 11
8.5. Tests 11
8.6. Programmierrichtlinien 11

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
BOS	**B**ranch **O**ffice **S**erver (Zentrale Ablage der XWORKS Projekte)
.NET	.NET ist eine von Microsoft entwickelte Softwareplattform. Mit dem .NET-Framework wird eine gut ausgestattete Library für die Entwicklung mit .NET bereitgestellt.
OOA	**O**bjekt**o**rientierte **A**nalyse
OOD	**O**bjekt**o**rientiertes **D**esign
XWP	DESIGO **XW**ORKS **p**lus ist eine Engineering-Umgebung im Bereich Gebäudeautomation der Siemens BT.
XWORKS plus	Siehe XWP.

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

BOS-WebAdmin ist ein Silverlight basierter Webclient welcher gewisse Admin-Funktionen zum BOS zur Verfügung stellt. Bisher ist eine Vollinstallation von *XWORKS plus* notwendig, um mit Hilfe des *XWP Project Managers* auf den BOS zuzugreifen. Durch den *BOS-WebAdmin* hat ein Administrator nun die Möglichkeit, von irgendeinem Computer mit Internetzugang und einem Browser (IE, Firefox, Safari) sich mit dem BOS zu verbinden, ohne zusätzliche Software installieren zu müssen. Der BOS-WebAdmin stellt die gebräuchlichsten Admin-Funktionen zur Verfügung, wie z.B. die Verwaltung der Benutzer und deren Rechte auf XWP Projekte. Ebenfalls kann überprüft werden, wer welche Projekte ausgecheckt hat und gegebenenfalls ein „Undo Check-out“ ausgeführt werden.

2.1. Zweck und Ziel

Mit diesem Projekt wollen wir das bisher angeeignete Wissen in Softwareengineering und objektorientierter Entwicklung anwenden und vertiefen. Wir wollen uns der Herausforderung stellen, ein Projekt mit einer für uns neuen Technologie (Silverlight) durchzuführen. Nebenbei möchten wir auch unsere Skills im Team- und Projektmanagement ausbauen.

2.2. Annahmen und Einschränkungen

Pro Teammitglied wird ein Arbeitsaufwand von ca. 17 Stunden pro Woche festgelegt. Falls Verzögerungen eintreten, die nicht mehr in sinnvollem Masse durch Mehrarbeit aufgeholt werden können, müssen einzelne weniger kritische Features 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	FK	• Software: Server-Schnittstelle (WCF), Asynchrone Servicearchitektur, Grafiken/Icons • Dokumentation
Reto Schneebeli	RS	• Software: User Interface, MVVM • Testing (Unit/System) • Versionsmanagement/Backup

3.2. Externe Schnittstellen

Hansjörg Huser	Prof. für Informatik, Dozent an der HSR, Leiter INS Beratung, Betreuung, Kontrolle (Reviews), Bewertung
Katja Panteleeva	Project Manager DESIGO V4.1 Tools, Siemens Schweiz AG Fachliche Betreuung seitens Industriepartner
René Föhn	Manager DESIGO V5.0 Tools, Siemens Schweiz AG Beratung und Administratorisches seitens Industriepartner

4. Management Abläufe

4.1. Projekt Kostenvoranschlag

Das Projekt startet am 14. September 2009 und dauert 14 Wochen bis spätestens am 18. Dezember 2009. Der Projektumfang beträgt ca. 480 Arbeitsstunden. Das entspricht ca. 17 Stunden pro Woche und Person.

4.2. Projektplan

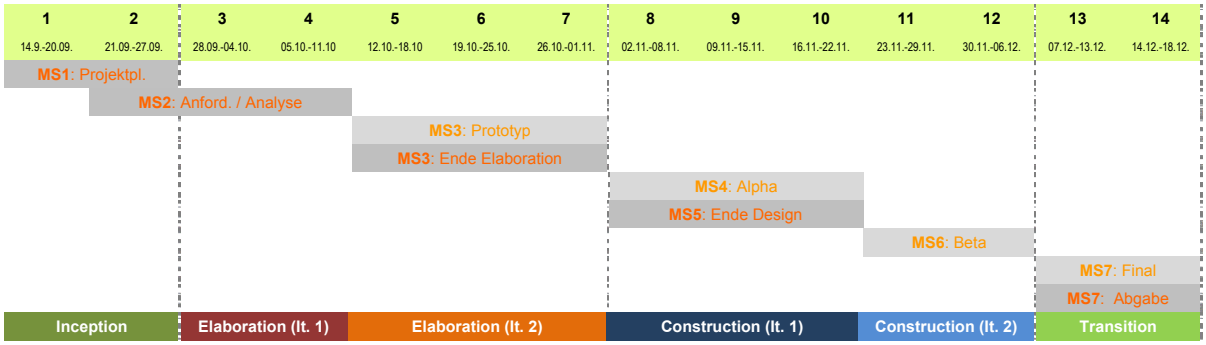
4.2.1. Zeitplan

Siehe separates Dokument: Projektplan_Zeitplanung.pdf

4.2.2. Iterationsplanung / Meilensteine

Meilenstein	Geplant	Erreicht	Artefakte
MS1: Projektplan	28.09.2009		• Projektplan • Zeitplan
MS2: Anforderungen und Analyse	12.10.2009		• Anforderungspezifikation (nichtfunktionale Anforderungen, UseCases) • Domain Modell
MS3: Ende Elaboration	02.11.2009		• Architekturprototyp • Design Entwurf (Klassendiagramme, Sequenzdiagramme)
MS4: Alpha	23.11.2009		• Alpha Release
MS5: Design	23.11.2009		• Software Architektur Spezifikation

			• Klassendiagramme • Sequenzdiagramme
MS6: Beta	07.12.2009		• Beta Release
MS7: Abgabe	18.12.2009		• Final Release • Testdokument • Anleitung? • Technischer Bericht



4.2.3. Besprechungen

Das Team trifft sich wöchentlich am Mittwoch zwischen 13:30 und 14:30 zur Besprechung, dabei werden die wichtigsten Fragen und Entscheide in einem Sitzungsprotokoll festgehalten. Zusätzlich wird bei Bedarf in der Siemens eine weitere Teambesprechung durchgeführt.

4.2.4. Software Releases

Release	Datum	Beschreibung
Prototyp	02.11.2009	Architekturprototyp über alle Layer
Alpha	23.11.2009	Erster Teil der Grundfunktionalitäten
Beta	07.12.2009	Voller Funktionsumfang
Final	18.12.2009	Funktion der Beta mit marginalen Anpassungen, Systemtest erfolgreich durchlaufen

5. Risiko Management

ID	Risiko	Auswirkung	Massnahmen	S	W	G	P
R01	Ausfall von Mitarbeiter (Zweierteam)	Ein Teil der definierten Arbeit wird nicht geleistet	Zeitreserve einplanen	3	3	9	1
R02	Datenverlust	Teil der Arbeit geht verloren	SVN einsetzen	4	1	4	4
R03	Zuwenig Silverlight Kenntnis	Unschöne Lösungen, falscher Einsatz der Technologie	Technologiestudie möglichst früh beginnen	3	2	6	2
R04	Fehler in der Zeitplanung	Meilensteine können nicht gehalten werden	Zeitreserve einplanen, Funktionen streichen.	2	3	6	2
R05	Technische Machbarkeit falsch eingeschätzt	Funktionen nicht umsetzbar	- Prototyp über alle Layer - Funktionen, deren Realisierbarkeit unsicher ist, als P2 bzw. P3 planen	2	2	4	3

S: Schaden [1 = gering, 2 = deutlich, 3 = hoch, 4 = sehr hoch]

W: Wahrscheinlichkeit [1 = gering, 2 = deutlich, 3 = hoch, 4 = sehr hoch]
G: Gewichteter Schaden (G=W*S)
P: Priorität [1 bis *]

Aus der Tabelle wird ersichtlich, dass auf jeden Fall eine Zeitreserve von 10% (50 h) eingeplant werden muss. Wurde so gelöst, dass bei jeder Zeitschätzung 10% addiert wurden. Aufgrund des Zweierteams hat der Ausfall einer Person eine Halbierung der geleisteten Arbeit zufolge und somit wäre auch das Einhalten der Meilensteine gefährdet. Zudem werden wir alle Funktionen die nicht essentiell sind als Priorität zwei kennzeichnen, um als Release auf jeden Fall eine lauffähige wenn auch noch ausbaufähige Applikation zu liefern. Da der Endtermin fix ist und unter keinen Umständen verschoben werden kann, ist bei Verzögerungen die einzige Möglichkeit gewisse Funktionen zu streichen.

6. Zeitschätzungen der Arbeiten

6.1. Projekt Management

ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeiten / Risiken
A101	Projektplanung (initial)	Projektplanungs-Dokument verfassen und übergeben.	RS, KF	18	
A102	Infrastruktur Aufbau	SVN-Zugang organisieren und Ablagestruktur definieren, Persönliche Infrastruktur.	RS, KF	11	
A103	Projektplanung (laufend)	Arbeitspakete und Zeitplanung weiterführen/verfeinern	RS	14	A101

6.2. Requirements

ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeiten / Risiken
A201	Anforderungen nichtfunktional	Definition aller nichtfunktionalen Anforderungen.		3	
A202	Dokument-erstellung + Use Cases ‚brief'	Dokumenterstellung + Definition aller funktionalen Anforderungen im ‚brief' Format.		6	A201
A203	Use Case Diagramm	UseCase Diagramm erstellen.		3	A202
A204	Use Cases ‚fully dressed'	Für die wichtigsten funktionalen Anforderungen die UseCases im ‚fully dressed' Format erstellen.		20	A202, A203

6.3. Analyse

ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeiten / Risiken
A301	Domain Analyse (OOA)	Domain Modell erarbeiten.		14	A202

6.4. Design

ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeiten / Risiken

A401	Software Architektur	Beschreiben der Software Architektur.		30	
A402	Design (OOD)	Klassendiagramm erstellen.		36	A2*, A3*
A403	Threads Kommunikation	Beschreiben der Threads und deren Kommunikation (Sequenzdiagramme).		10	

6.5. Implementation

ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeiten / Risiken
A501	Implementation	Umfasst die komplette Implementation. Wird am Ende der Elaborationsphase 2 genauer aufgeteilt.	KF, RS	124	
	Server-Verwaltung	Server-History-Verwaltung im Login-Dialog	RS	4	
	XWP-Projekte browsen	- Browser-Tree und Detail-View - Browser Service vollständig	RS	24	
	Undo-Checkout	- Auschecken rückgängig machen	KF	14	
	Benutzer-zugriffe verwalten	- Benutzer verwalten - Benutzergruppen verwalten - Benutzer zu Gruppe zuweisen - Rollen verwalten - Berechtigung zu Rolle zuweisen	KF	42	
	Generelle Berechtigungen verwalten	Generelle Berechtigungen verwalten	KF	9	
	Projektspezifische Berechtigungen verwalten	Projekt-/Ordnerspezifische Berechtigungen verwalten	KF	9	
	Permission	Rechte-Überprüfung	RS	5	
	Refactoring	Refactoring des Codes.	KF	7	
A502	Codedokumentation	Generieren der Codedokumentation		2	
A503	Prototyp	Architekturprototyp		41	A3*, A401, A402, A403
A504	Fehlerkorrektur	Korrektur allfälliger Fehler	KF, RS	6	
A505	Implementation BOS	Anpassungen am BOS.	KF	wurde nichts geplant	

6.6. Test

ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeiten / Risiken
A601	Systemtest	Beinhaltet den manuellen, auf die Auslieferung ausgerichteten Test der gesamten Applikation vor jedem Release.		6	A2*, A5* / Falls Fehler vorhanden Mehraufwand

					bei A501.
A602	Testdokumen-tation	Erstellen der Testdokumentation.		10	

6.7. Dokumentation

ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeiten / Risiken
A701	Poster	Erstellen des Posters.	RS	5	
A702	Technischer Bericht	Verfassen des Technischen Berichts.	RS	30	
A703	Dokument Abschluss	Abschluss und Bereinigung aller Dokumente.	KS	16	
A704	Management Summary	Management Summary erstellen	KS	12	
A705	Schlussbericht	Schreiben des persönlichen Schlussberichtes	RS, KF	2	

6.8. Studium Technologien (über ganze Projektdauer)

ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeiten / Risiken
A011	Einarbeiten in Silverlight 3	Wissen zur Verwendung von Silverlight 3 aneignen.	RS, KF	14	
A012	Einarbeiten in XWP BOS Architektur	Wissen über intelligentes Threading im .NET Framework, Synchronisations-Prinzipien sowie nützliche Design Patterns aneignen.	RS, KF	5	

6.9. Sitzungen (über ganze Projektdauer)

ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeiten / Risiken
A021	Teamsitzungen	Teamsitzungen zur Orientierung über den aktuellen Projektstand.	KF	30	
A022	Reviews	Reviews zur Überprüfung der Meilensteine.	KF	14	

6.10. Qualitätssicherung (über ganze Projektdauer)

ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeiten / Risiken
A031	Code Review	Gegenseitiges Code-Review	RS	8	
A032	Infrastruktur Betrieb	Versionsverwaltung, Backup, …	RS	2	

7. Infrastruktur

7.1. Räumlichkeiten

Ca. 8 Stunden pro Woche werden die Räume für Semesterarbeiten in der HSR benutzt. Die restliche Arbeit erfolgt zu Hause sowie direkt beim Industriepartner.

7.2. Hardware

- Private Notebooks der einzelnen Teammitglieder
- Rechner in den Computerräumen der HSR
- SVN-Server der HSR
- Private Drucker
- Drucker und Kopierer der HSR

7.3. Software

7.3.1. Entwicklungstools

- Microsoft Visual Studio 2008 (Entwicklungsumgebung)
- Microsoft Expression Blend 3 (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 2007 (Dokumentation)
- Microsoft Excel 2007 (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.

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

Während dem Programmieren werden laufend Unit-Tests erstellt und bei jedem Build ausgeführt. Soweit möglich wird Test Driven Development (TDD) eingesetzt. Für den GUI-Teil ist dies allerdings nicht möglich. Als Unit-Test-Framework wird NUnit 2.5 eingesetzt.
Bei jedem Release wird ein 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://msdn2.microsoft.com/en-us/library/xzf533w0(vs.71).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: **BOS-WebAdmin**
Teammitglied: **Kaspar Fenner**

Durchschnittliche Soll-Arbeitszeit: **17.0**

Woche	Datum	Tätigkeit	Zeit	Zeit/Wo	Total	Soll
Woche 1	14.09.2009	Kick-Off Meeting	1.5			
	16.09.2009	Infrastruktur: Einrichten von Notebook, Dokumentablagestruktur	3.0			
	17.09.2009	Infrastruktur: Einrichten von Notebook	3.0			
	17.09.2009	Meeting mit René Föhn über weiteres Vorgehen, Benötigte Infrastruktur, Diskussion über Requirements	1.0			
	17.09.2009	Technologiestudium: Silverlight 3	0.5			
	20.09.2009	Projektplan erarbeiten	4.0			
	20.09.2009	Requirements: Dokumenterstellung, Use Cases erarbeiten "brief"	2.0			
				15.0	**15.0**	17.0
Woche 2	21.09.2009	Requirements: Use Cases erarbeiten "fully dressed"	3.0			
	21.09.2009	Einarbeiten in XWP BOS Architektur (User Security Management)	1.0			
	22.09.2009	Requirements: Use Cases erarbeiten "fully dressed"	4.0			
	22.09.2009	Projektplan erarbeiten, für erstes Review vorbereiten	2.0			
	23.09.2009	Meeting mit H. Huser: Besprechung bezüglich Arbeitspakete und Zeitplanung	1.0			
	24.09.2009	Bestehende Projektmanager-Architektur studiert, mögliche neue Architektur überlegt	1.5			
	24.09.2009	Architektur-Besprechung mit Katja Panteleeva	2.0			
	24.09.2009	Technologiestudium: Silverlight 3	2.0			
	25.09.2009	Grober Architektur-Vorschlag ausarbeiten	2.5			
	25.09.2009	Technologiestudium: Silverlight 3	1.0			
				20.0	**35.0**	34.0
Woche 3	28.09.2009	Requirements: Use Cases erarbeiten "brief" + Diagramm	2.5			
	28.09.2009	Weekly Meeting bei Siemens mit R, Föhn und K. Panteleeva	1.0			
	29.09.2009	Infrastruktur: VMWare mit XWP + BOS einrichten	2.0			
	29.09.2009	Feasibility Study: Composite Application Guidance	2.0			
	29.09.2009	Weekly Meeting mit H. Huser	1.0			
	01.10.2009	Infrastruktur: VMWare mit XWP + BOS einrichten	2.0			
	02.10.2009	Infrastruktur: Remotezugriff von einer VMWare zur anderen mit XWP-ProjectManager	0.5			
	02.10.2009	Technologiestudium: Silverlight 3 / Architektur MVVM / CAG / ICommand	2.5			
	04.10.2009	Requirements: Kapitel 2 + 3 + "Finishing"	4.0			
				17.5	**52.5**	51.0
Woche 4	05.10.2009	Weekly Meeting mit H. Huser	1.0			
	06.10.2009	Weekly Meeting bei Siemens (R. Föhn + K. Panteleeva): Review UseCases/DomainModell	1.0			
	07.10.2009	Requirements: Reviewkommentare einarbeiten	1.5			
	07.10.2009	Prototyp: Visual Studio Projekt aufsetzen	1.0			
	08.10.2009	Prototyp: WCF Connection mit BOS	7.0			
	09.10.2009	Prototyp: WCF Connection mit BOS	1.0			
	09.10.2009	Technologiestudium: Silverlight 3 und WCF	3.0			
	11.10.2009	Prototyp: WCF Connection mit BOS / SSL Zertifikate	4.0			
				19.5	**72.0**	68.0

Zeiterfassung

Projekt: **BOS-WebAdmin**
Teammitglied: **Kaspar Fenner**

Durchschnittliche Soll-Arbeitszeit: **17.0**

Woche	Datum	Tätigkeit	Zeit	Zeit/Wo	Total	Soll
Woche 5	12.10.2009	Weekly Meeting mit H. Huser	1.0			
	12.10.2009	Besprechung mit K. Panteleeva bez. WCF Connection mit BOS	1.0			
	12.10.2009	Vorbereitung von Debug Version von BOS (Convertierung auf VS2008 + .NET Framework 3.5)	2.0			
	13.10.2009	Prototyp: WCF Connection mit BOS	8.0			
	14.10.2009	Prototyp: WCF Connection mit BOS	2.5			
	14.10.2008	BOS: Fehler behoben bei Verbindung mit SQL Server (SQL-Server + .NET 3.5 Problem)	3.0			
	15.10.2009	BOS: Remote Debugging aufsetzen	3.0			
	15.10.2009	Architektur-Besprechung mit Reto	3.0			
	15.10.2009	BOS: Allow parallel connections from Silverlight & Desktop Applications	1.0			
	16.10.2009	BOS: Allow parallel connections from Silverlight & Desktop Applications	3.5			
				28.0	**100.0**	85.0
Woche 6	19.10.2009	Weekly Meeting mit H. Huser	1.0			
	19.10.2009	Prototyp: WCF Connection mit BOS (Serverseitige Anpassungen)	3.5			
	20.10.2009	Prototyp: WCF Connection mit BOS (Client + Serverseitige Anpassungen)	6.0			
	21.10.2009	Prototyp: Implement advanced WCF service proxy handling + Service async callback handling	5.0			
	22.10.2009	BOS: Silverlight Fault Behavior im Server aktiviert / Prototyp: Browser Service	6.0			
	23.10.2009	Prototyp: Browser + Security Services	3.5			
				25.0	**125.0**	102.0
Woche 7	26.10.2009	Weekly Meeting bei Siemens mit R. Föhn und K. Panteleeva	1.0			
	26.10.2009	Prototyp: Singletons entfernt -> neu ServiceLocator / ApplicationFactory / Authentication, Login funktioniert	4.5			
	27.10.2009	Code- + Architektur-Review mit K. Panteleeva	1.0			
	27.10.2009	Prototyp: SessionManager eingebaut	3.0			
	28.10.2009	Weekly Meeting mit H. Huser	1.0			
	28.10.2009	Prototyp: Improve Service Implementation (Async Pattern)	3.0			
	29.10.2009	Prototyp: WcfServiceFactory / Exception Handling / Service Implementation	6.0			
	29.10.2009	Code- + Architektur-Review mit K. Panteleeva	1.0			
	30.10.2009	Prototyp: Authorization Service	3.0			
	31.10.2009	Architekturdokument: Asynchrone Servicearchitektur, Einschränkungen Async-Pattern, Klassenübersicht erweitert	4.0			
	01.11.2009	BOS-Server Code im SVN eingecheckt (benötigte Libraries und Projekte aus XWP-Tree extrahiert)	0.5			
	01.11.2009	Architekturdokument: Serverseitige Anpassungen, Übersicht Application Core, weitere Überarbeitung und Ergänzungen von Diagrammen und Texten	4.0			
				32.0	**157.0**	119.0

Zeiterfassung

Projekt: **BOS-WebAdmin**
Teammitglied: **Kaspar Fenner**

Durchschnittliche Soll-Arbeitszeit: **17.0**

Woche	Datum	Tätigkeit	Zeit	Zeit/Wo	Total	Soll
Woche 8	02.11.2009	Introduce Moq Framework for Unit Testing / Switch completely to Nunit	4.0			
	02.11.2009	UseCases und Planung weiterer Iterationen	2.0			
	03.11.2009	Weekly Meeting mit H. Huser, R. Föhn, K. Panteleeva bei Siemens	1.0			
	03.11.2009	Unit Tests mit Moq-Framework (Browser Service, ServiceFactory, ServiceLocator)	6.0			
	03.11.2009	Planung/Zeitschätzung UseCases, Nachführung Projektplan	1.0			
	05.11.2009	Browser Service + Model erweitert für Browse-Detail-View	2.0			
	06.11.2009	Browser Service + Model erweitert für Browse-Detail-View	3.0			
				19.0	**176.0**	136.0
Woche 9	10.11.2009	Browser Service + Model erweitert für Browse-Detail-View	2.0			
	11.11.2009	Browser Service + Model erweitert für Browse-Detail-View	2.0			
	11.11.2009	Administration Service (Änderungen auf BOS Server)	1.0			
	11.11.2009	Administration Service	1.0			
	12.11.2009	BOS-WebAdmin auf boswebadmin.gotdns.com eingerichtet (Domain, VMWare, Zertifikate, Portforwarding, etc.)	3.0			
	12.11.2009	Administration Service	2.0			
	13.11.2009	Administration Service	3.0			
	14.11.2009	Moq Unit Test Framework gepatcht, um mehr als 4 Methodenparameter zu unterstützten in Func<> Delegates	1.0			
	15.11.2009	Administration Service	1.0			
				16.0	**192.0**	153.0
Woche 10	16.11.2009	User Management Service	2.0			
	17.11.2009	User Management Service	5.0			
	18.11.2009	User Management Service	3.0			
	18.11.2009	Weekly Meeting mit H. Huser	1.0			
	19.11.2009	User Management Service	5.5			
	20.11.2009	User Management Service	3.0			
	22.11.2009	Systemtest	2.0			
				21.5	**213.5**	170.0
Woche 11	23.11.2009	Design Dokument Kapitel 9 verfasst / Dokumente zum Review verschickt	1.0			
	23.11.2009	User Management Service	1.0			
	23.11.2009	Besprechung mit Reto: Design von Permission-Dialogen	1.0			
	23.11.2009	Planung/Zeitschätzung UseCases, Nachführung Projektplan	1.0			
	25.11.2009	User Management Service	4.0			
	25.11.2009	Weekly Meeting mit H. Huser	1.0			
	27.11.2009	User Management Service	2.5			
	29.11.2009	Technischer Bericht	3.0			
				14.5	**228.0**	187.0

Zeiterfassung

Projekt: **BOS-WebAdmin**
Teammitglied: **Kaspar Fenner**

Durchschnittliche Soll-Arbeitszeit: **17.0**

Woche	Datum	Tätigkeit	Zeit	Zeit/Wo	Total	Soll
Woche 12	02.12.2009	Replanning UseCases, Nachführung Projektplan	1.0			
	02.12.2009	Weekly Meeting mit H. Huser	1.0			
	02.12.2009	Authorization Service / UserManagement Service / Refactoring	4.5			
	04.12.2009	User Management Service / Cleanup & Refactoring	3.0			
	06.12.2009	Code-Kommentare überarbeitet	6.0			
				15.5	**243.5**	204.0
Woche 13	07.12.2009	Verschlüsselung von Benutzerpasswort bei XML-Serialisierung	2.0			
	07.12.2009	Versch. Kleinigkeiten gefixt, Verschönerungen, Refactoring etc.	2.0			
	07.12.2009	Design Dokument Aktualisiert	0.5			
	08.12.2009	Design Dokument Aktualisiert	6.5			
	09.12.2009	Weekly Meeting mit H. Huser	1.0			
	09.12.2009	Abstract, Management Summary, Poster, Titelblatt von Templates erstellt / Abstract angefangen	2.5			
	10.12.2009	Codereview mit K. Panteleeva	0.5			
	11.12.2009	Abstract und Management Summary	4.5			
	12.12.2009	Technischer Bericht	1.5			
	13.12.2009	Abstract überarbeitet / Technischer Bericht	3.5			
				24.5	**268.0**	221.0
Woche 14	14.12.2009	Poster / Technischer Bericht	2.0			
	15.12.2009	Vorbereitung Abschlusspräsentation	3.0			
	15.12.2009	Weekly Meeting mit H. Huser / Abschlusspräsentation des Projektes bei Siemens	1.0			
	16.12.2009	Technischer Bericht	4.5			
	17.12.2009	Technischer Bericht	5.0			
	17.12.2009	Dokumentabschluss	1.5			
	17.12.2009	Persönlicher Bericht	2.0			
	18.12.2009	Dokumentabschluss	2.0			
				21.0	**289.0**	238.0

Total Ist - Arbeitszeit: **289.0**
Total Soll - Arbeitszeit: **238.0**
Differenz: **51.0**

Zeiterfassung

Projekt: **BOS-WebAdmin**
Teammitglied: **Reto Schneebeli**

Durchschnittliche Soll-Arbeitszeit: **17.0**

Woche	Datum	Tätigkeit	Zeit	Zeit/Wo	Total	Soll
Woche 1	14.09.2009	Kick-Off Meeting	1.5			
	16.09.2009	Infrastruktur: Dokumentstruktur	1.0			
	16-09-2009	Zeitplanung	2.0			
	17.09.2009	Meeting mit René Föhn über weiteres Vorgehen, Benötigte Infrastruktur, Diskussion über Requirements	1.0			
	18.09.2009	Infrastruktur: SVN Repository	0.5			
	18.09.2009	Projektplan / Zeitplanung	5.0			
	20.09.2009	Infrastruktur: Einrichten von Notebook	2.0			
				13.0	**13.0**	17.0
Woche 2	21.09.2009	Projektplan (Arbeitspakete) / Zeitplanung	2.0			
	22.09.2009	Besprechung bezüglich Arbeitspakete und Zeitplanung	1.0			
	23.09.2009	Meeting mit H. Huser: Besprechung bezüglich Arbeitspakete und Zeitplanung	1.0			
	23.09.2009	Review-Kommentare einarbeitet	1.0			
	24.09.2009	Architektur Ideen besprochen	2.0			
	25.09.2009	Silverlight, Domainmodell und Architektur	4.0			
	26.09.2009	Domainmodell, Architektur Schema	4.0			
	26.09.2009	Silverlight Einarbeiten	2.0			
				17.0	**30.0**	34.0
Woche 3	28.09.2009	Weekly Meeting in der Siemens	1.0			
	29.09.2009	Domainmodell fertiggestellt und mit Raphael verifiziert	4.0			
	30.09.2009	Domainmodelldokument erstellt und Beschreibungen hinzugefügt	1.5			
	30.09.2009	Meeting mit H. Huser: Domainmodell, UseCases und Architektur	1.0			
	30.09.2009	Architektur Ideen besprochen	1.5			
	30.09.2009	Silverlight Tests mit Composite Application Guidance	2.0			
	01.10.2009	Silverlight MVVM Master/Detail Dialog mit Binding	3.0			
	02.10.2009	Infrastruktur BOS und Test Projekte	1.0			
	02.10.2009	UseCases und Architektur	2.0			
	04.10.2009	Weitere UseCases "fully dressed" erstellt + UseCase Diagramm	3.0			
				20.0	**50.0**	51.0
Woche 4	05.10.2009	Meeting mit H. Huser: UseCases und Architektur	1.0			
	06.10.2009	Weekly Meeting in der Siemens: UseCase und Domainmodell	1.0			
	06.10.2009	Review-Kommentare einarbeitet (Domainmodell)	0.5			
	07.10.2009	Domainmodell: Weitere Beschreibungen	1.5			
	07.10.2009	UseCases: Review Kommentare einarbeitet + UseCase Diagramm	1.0			
	07.10.2009	Architektur: CAG - View Injection (Named Locations, …)	1.0			
	09.10.2009	UseCases: Review Kommentare einarbeitet	1.0			
	09.10.2009	Architektur: CAG - View Injection (Named Locations, …)	3.0			
	10.10.2009	Architektur: CAG - View Injection (Named Locations, …)	5.3			
	11.10.2009	Zeitplanung, Dokumente Meilenstein 2	1.7			
				17.0	**67.0**	68.0

Zeiterfassung

Projekt: **BOS-WebAdmin**
Teammitglied: **Reto Schneebeli**

Durchschnittliche Soll-Arbeitszeit: **17.0**

Woche	Datum	Tätigkeit	Zeit	Zeit/Wo	Total	Soll
Woche 5	12.10.2009	Meeting mit H. Huser: UseCases, Domainmodell, CAG und WCF	1.0			
	14.10.2009	Domainmodell (User/Gruppen - Berechtigungen)	1.5			
	14.10.2009	Architektur: CAG - DependencyInjection, EventAggregation und Logger	6.5			
	15.10.2009	Architektur Übersicht: Diskussion und Diagramm erstellt	4.0			
	16.10.2009	Review Kommentare von René + Entwurf User Interface Storyboard	1.5			
	16.10.2009	Architektur: Dokument erstellt und zwei Varianten aufgezeigt	2.5			
	17.10.2009	Arch: Namespace/Folder Struktur und View - Command - Service Verbindungen	2.0			
	17.10.2009	Prototyp: Bootstrapper, Authentication Handling	3.0			
				22.0	**89.0**	85.0
Woche 6	19.10.2009	Meeting mit H. Huser: Domainmodell, CAG und WCF	1.0			
	19.10.2009	Prototyp: Main View, Named Locations, Browse Module	5.5			
	20.10.2009	Prototyp: Main View, Browse Module/View	5.0			
	21.10.2009	Prototyp: Logger, Authentication	4.0			
	22.10.2009	Prototyp: Browse über WCF	2.0			
	23.10.2009	Prototyp: Wechsel zwischen der Login View und der Main View	5.5			
	24.10.2009	Prototyp: Login View	1.5			
	25.10.2009	Prototyp: Alle ViewRegion Namen in eine Klasse	0.5			
	25.10.2009	Prototyp: Alle Texte in Resourcen und Titel auch in der Login View	2.0			
				27.0	**116.0**	102.0
Woche 7	26.10.2009	Weekly Meeting in der Siemens: Prototyp und Architektur	1.0			
	26.10.2009	Prototyp: Localization und TextBox Binding	1.0			
	27.10.2009	Code - Architekturreview mit Katja	1.0			
	27.10.2009	Prototyp: Browser Icons, Resource Management + Besprechung	3.0			
	28.10.2009	Meeting mit H. Huser: Prototyp und Architektur	1.0			
	28.10.2009	Architektur: Klassendiagramme Presentation + Unittest	5.5			
	29.10.2009	Codereview + Prototyp: StatusPresenter	2.0			
	30.10.2009	Prototyp: StatusPresenter, Presenter-View Beziehung entfernt.+ Architektur	8.0			
	31.10.2009	Architekturdokument: Einschränkungen / Beschreibungen / Diagramme	3.5			
	01.11.2009	Architekturdokument + Systemtestdokument erstellt und Test durchgeführt	5.0			
				31.0	**147.0**	119.0

Zeiterfassung

Projekt: **BOS-WebAdmin**
Teammitglied: **Reto Schneebeli**

Durchschnittliche Soll-Arbeitszeit: **17.0**

Woche	Datum	Tätigkeit	Zeit	Zeit/Wo	Total	Soll
Woche 8	02.11.2009	Besprechung bezüglich Arbeitspakete	1.5			
	02.11.2009	Zeitplanung für die Construction-Phase	1.0			
	03.11.2009	Review in der Siemens inkl. H. Huser	1.0			
	04.11.2009	Usecases für die Benutzerverwaltung UC07, UC08, UC09 "fully dressed"	2.0			
	05.11.2009	Usecases für die Rechtezuweisung UC10, UC11, UC12, UC13 "fully dressed"	2.0			
	06.11.2009	Impl: Animationen in Silverlight und Browser Detail	8.0			
	07.11.2009	Impl: XWP-Projekte browsen und kleine Verbesserungen	2.5			
	08.11.2009	Impl: Server-Verwaltung	1.0			
				19.0	**166.0**	136.0
Woche 9	09.11.2009	Impl: XWP-Projekte browsen	3.0			
	10.11.2009	Impl: Server-Verwaltung	2.0			
	11.11.2009	Impl: Server-Verwaltung und Einarbeiten in Silverlight User Controls	5.0			
	11.11.2009	Impl: XWP-Projekte browsen	2.0			
	12.11.2009	Impl: Server-Verwaltung (Info Text) + Session anzeigen	1.0			
	13.11.2009	Impl: XWP-Projekte browsen (Load on Click, DetailView, Scrolling)	3.0			
	14.11.2009	Impl: UndoCheckout	2.0			
	15.11.2009	Impl: UndoCheckout	1.0			
				19.0	**185.0**	153.0
Woche 10	16.11.2009	Impl: UndoCheckout	2.0			
	17.11.2009	Impl: UndoCheckout (Refresh nach undo checkout)	2.0			
	18.11.2009	Meeting mit H. Huser: Implementationsfortschritt	1.0			
	18.11.2009	Impl: Benutzer verwalten - User	4.0			
	19.11.2009	Impl: Benutzer verwalten - User	3.0			
	20.11.2009	Impl: Benutzer verwalten - User + User Groups	4.0			
	21.11.2009	Impl: Benutzer verwalten + Vereinfachung des Bindings an Textressourcen	5.0			
	22.11.2009	Impl: Benutzer verwalten - User Groups	1.0			
	22.11.2009	Testdokument erweitert	1.0			
				23.0	**208.0**	170.0
Woche 11	23.11.2009	Impl: Benutzer verwalten - Roles	1.0			
	23.11.2009	Besprechung mit Kaspar: Design von Permission-Dialogen	1.0			
	23.11.2009	Planung/Zeitschätzung UseCases, Nachführung Projektplan	1.0			
	25.11.2009	Meeting mit H. Huser: Dokumente, …	1.5			
	25.11.2009	Impl: Benutzer verwalten - Roles + GUI überall einwenig verändert	4.0			
	26.11.2009	Impl: Benutzer verwalten - Roles	3.5			
	27.11.2009	Impl: Benutzer verwalten - Roles	1.0			
	27.11.2009	Impl: Projektspezifische Berechtigungen	2.0			
	28.11.2009	Impl: Generelle Berechtigungen	4.0			
	29.11.2009	Impl: Projektspezifische Berechtigungen + Kommentare	2.0			
				21.0	**229.0**	187.0

Zeiterfassung

Projekt: **BOS-WebAdmin**
Teammitglied: **Reto Schneebeli**

Durchschnittliche Soll-Arbeitszeit: **17.0**

Woche	Datum	Tätigkeit	Zeit	Zeit/Wo	Total	Soll
Woche 12	30.11.2009	Besprechung mit Kaspar: Replaning	1.0			
	02.12.2009	Meeting mit H. Huser: Technischer Bericht...	1.0			
	02.12.2009	Impl: Icon für CheckedOutByOther anzeigen	2.5			
	02.12.2009	Impl: Permissions überprüfen	2.5			
	04.12.2009	Impl: Permissions überprüfen	1.5			
	04.12.2009	Impl: Refactoring	2.5			
	04.12.2009	Impl: Refactoring	3.5			
	05.12.2009	Impl: Refactoring	1.0			
	05.12.2009	Impl: Undo Checkout (prepaire-commit)	3.0			
				18.5	**247.5**	204.0
Woche 13	07.12.2009	Fehlerkorrektur und Gui-Styling	3.5			
	07.12.2009	Testdokument erweitert	3.0			
	08.12.2009	Designdokument - Klassendiagramme aktualisiert	6.0			
	09.12.2009	Impl: Fehlerkorrektur	1.0			
	09.12.2009	Meeting mit H. Huser: Demo des Beta-Release	1.0			
	09.12.2009	Dokummentabschluss (Architekturdokumment)	2.0			
	10.12.2009	Interfacedoku generiert + Code-Review mit Katja Panteleeva	1.5			
	11.12.2009	Impl: Fehlerkorrektur + Management-Summary + Technischer-Bericht	6.5			
	12.12.2009	Technischer Bericht (Design, User-Interface, ...)	1.0			
	13.12.2009	Management-Summary, Technischer-Bericht und Poster	3.0			
				28.5	**276.0**	221.0
Woche 14	14.12.2009	Poster	1.0			
	15.12.2009	Präsentation für Abschluss-Meeting	1.0			
	15.12.2009	Abschluss-Meeting in der Siemens	1.0			
	16.12.2009	Technischer-Bericht (Qualitätssicherung, Technologie-Studium, …)	4.0			
	16.12.2009	Systemtest Final-Release	1.0			
	17.12.2009	Dokummentabschluss und Technischer-Bericht	2.0			
	17.12.2009	Persönlicher Erfahrungsbericht.	3.0			
	18.12.2009	Dokumentabschluss (Register, Literaturverzeichnis, …)	1.0			
				14.0	**290.0**	238.0

Total Ist - Arbeitszeit: **290.0**
Total Soll - Arbeitszeit: **238.0**
Differenz: **52.0**

Projekt: BOS-WebAdmin

Anforderungsspezifikation

[Dokumentstruktur basiert auf RUP „Requirement Specification“]

0. Dokumentinformationen

0.1. Änderungsgeschichte

Datum	*Version*	*Änderung*	*Autor*
20.09.2009	0.1	Initial Version, UC01, UC02 „brief“	Kaspar Fenner
21.09.2009	0.2	UC01, UC02 „fully dressed“, UC03 “brief”	Kaspar Fenner
22.09.2009	0.3	UC01, UC02, UC03 „fully dressed“	Kaspar Fenner
28.09.2009	0.4	Weitere Use Cases “brief” hinzugefügt + Diagramm	Kaspar Fenner
04.10.2009	0.5	UC04, UC05 und UC06 "fully dressed"	Reto Schneebeli
04.10.2009	0.6	Kapitel 2 + 3	Kaspar Fenner
07.10.2009	0.7	Reviewcomments eingearbeitet	Kaspar Fenner
09.10.2009	0.8	Review Kommentare einarbeitet	Reto Schneebeli
16.10.2009	0.9	Review Kommentare von René einarbeitet (Storyboard)	Reto Schneebeli
04.11.2009	1.0	UC07, UC08, UC09 "fully dressed" + bereinigt	Reto Schneebeli
05.11.2009	1.1	UC10, UC11, UC12, UC13 "fully dressed"	Reto Schneebeli

0.2. Inhalt

0. Dokumentinformationen 2
0.1. Änderungsgeschichte 2
0.2. Inhalt 3
1. Einführung 5
1.1. Zweck 5
1.2. Gültigkeitsbereich 5
1.3. Definitionen und Abkürzungen 5
1.4. Referenzen 5
1.5. Übersicht 5
2. Allgemeine Beschreibung 6
2.1. Produkt Perspektive 6
2.2. Produkt Funktion 6
2.3. Benutzer Charakteristik 6
2.4. Einschränkungen 6
2.5. Annahmen 7
2.6. Abhängigkeiten 7
3. Spezifische Anforderungen 7
3.1. Funktionale Anforderungen 7
3.1.1. BOS-Server vor unberechtigtem Zugriff schützen (Login/Logout) 7
3.1.2. Aktionen protokollieren 7
3.2. Bedienbarkeit 7
3.3. Zuverlässigkeit 7
3.4. Leistung 7
3.5. Datenschutz 8
3.6. Datensicherheit 8
3.7. Wartbarkeit 8
3.8. Schnittstellen 8
3.9. Lizenzanforderungen 8
3.10. Verwendete Standards 8
4. Use Cases 9
4.1. Use Case Diagramm 9
4.2. Aktoren 9
4.3. Use Case Notation 9
4.4. Generelle Fehlerbehandlung 10
4.5. UI Storyboard 10
4.6. UC01: BOS-Server verbinden 10
4.7. UC02: XWP-Projekte browsen 11
4.8. UC03: BOS-Server-Verbindung trennen 12
4.9. UC04: Auschecken rückgängig machen (Undo check-out) 12
4.10. UC05: History ansehen (wird nicht umgesetzt) 13
4.11. UC06: Suchen (wird nicht umgesetzt) 13
4.12. UC07: Benutzer verwalten (CRUD) 14
4.13. UC08: Benutzergruppen verwalten (CRUD) 15
4.14. UC09: Rollen verwalten (CRUD) 15
4.15. UC10: Generelle Berechtigungen verwalten 16

4.16. UC11: Projekt/Ordner-spezifische Berechtigungen verwalten 16
4.17. UC12: Rolle zu Benutzer zuweisen 17
4.18. UC13: Rolle zu Benutzergruppe zuweisen 17
4.19. UC14: History bereinigen (wird nicht umgesetzt) 17

1. Einführung

1.1. Zweck

Die Anforderungsspezifikation beschreibt die grundlegenden Anforderungen, die an die Software gestellt werden. Dabei setzen vor allem die Use Cases einen wichtigen Grundstein für das spätere Design.

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**
XWP	DESIGO **XW**ORKS **p**lus (Engineering-Umgebung von Siemens BT)
BOS	**B**ranch **O**ffice **S**erver (Zentrale Ablage der XWORKS Projekte)
AS	**A**utomation **S**tation
MS	**M**anagement **S**tation

1.4. Referenzen

Aufgabenstellung.pdf

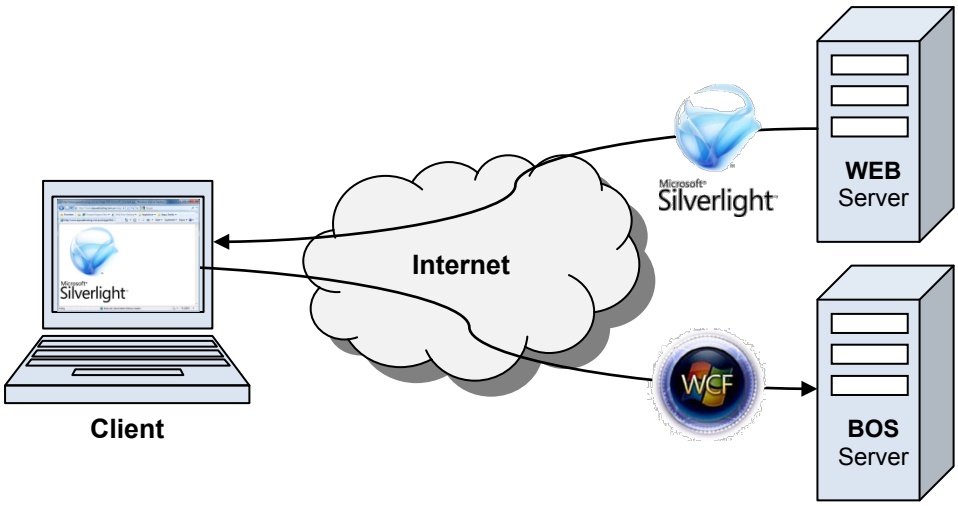
1.5. Übersicht

Im nächsten Kapitel wird das Produkt aus verschiedenen Perspektiven beschrieben. Darauf folgt eine Auflistung von funktionalen sowie nicht-funktionalen Anforderungen. Im letzten Kapitel werden die Use Cases detailiert beschrieben.

2. Allgemeine Beschreibung

2.1. Produkt Perspektive

Das BOS-WebAdmin ist ein Silverlight basierter Webclient, welcher gewisse Admin- und Reporting-Funktionen zum BOS-Server zur Verfügung stellt. Es ist somit kein Ersatz für den bestehenden XWP-ProjectManager, da keine Projektdaten bearbeitet werden können. D.h. die Kernfunktionen werden nach wie vor nur vom XWP-ProjectManager zur Verfügung gestellt. Das BOS-WebAdmin ist nur ein zusätzliches „Lightweight"-Tool, das insbesondere von Projektmanagern dazu genutzt werden kann, um möglichst schnell und einfach Verwaltungsoperationen auf dem BOS-Server auszuführen. Dadurch, dass das BOS-WebAdmin über den Internetbrowser aufgerufen wird, kann aus der ganzen Welt darauf zugegriffen werden. Dies natürlich unter der Voraussetzung, dass Web- und BOS-Server über das Internet erreichbar sind.



2.2. Produkt Funktion

Während des Zeitraumes der Semesterarbeit werden mindestens die Grundlegendsten Administrationsfunktionen sowie ausgewählte History- und Reporting-Funktionen implementiert. Das heisst, der Benutzer kann:

- mit einem BOS-Server verbinden
- XWP-Projekte browsen
- das Auschecken eines Nodes rückgängig machen (undo checkout)
- Benutzer verwalten
- Benutzerrechte und Zugriffsrechte auf Ordner/Projekte verwalten
- die History ansehen (gefiltert)
- nach Projekten, Benutzern etc. suchen

Als Erweiterung ist geplant, Reporting-Funktionalitäten einzubinden wie z.B. eine Benutzungsstatistik oder einen Report über alle auf dem BOS vorhandenen Projekte. Diese Reports sollten dann in druckbarer Form ausgegeben werden. Details müssen aber noch genauer spezifiziert werden.

2.3. Benutzer Charakteristik

Die Hauptzielgruppe sind Projektmanager, die zu jeder Zeit und ohne grossen Aufwand Administrationstätigkeiten wahrnehmen müssen. Diese Benutzer sind normalerweise mit dem bestehenden XWP-ProjectManager vertraut, weshalb das BOS-WebAdmin in der Bedienung ähnlich aufgebaut sein sollte, soweit dies möglich ist. Es kann nicht unbedingt von sehr erfahrenen Windows-Benutzern ausgegangen werden, aber die Grundlagen sind auf jeden Fall vorhanden.

2.4. Einschränkungen

Da das Projekt in sehr kurzer Zeit und mit beschränktem Aufwand realisiert werden muss, muss nicht zwingend die ganze Benutzer- und Rechteverwaltung vom XWP-ProjectManager übernommen werden, weil dies ein relativ hoher Teil an Fleissarbeit enthält, um die verschiedenen Dialoge etc. zu erstellen. Die Grundfunktionalität der Benutzerverwaltung sollte implementiert werden.

Die Serverschnittstelle sollte wenn möglich nicht geändert werden. Auf jeden Fall muss die Rückwärtskompatibilität mit dem bestehenden XWP-ProjectManager gewahrt bleiben. Für neue Funktionen wie das Reporting sollte eine neue Serverschnittstelle definiert werden.

2.5. Annahmen

Da es bei diesem Projekt primär um die Client-Entwicklung geht, muss eine einwandfreie Installation des BOS-Server vorausgesetzt werden. Das Netzwerk und die Firewall müssen ebenfalls richtig konfiguriert sein. Dies schliesst selbstverständlich eine entsprechende Fehlerbehandlung, wie z.B. bei Nichterreichbarkeit des Servers, nicht aus.

2.6. Abhängigkeiten

BOS-WebAdmin wird in C# und Silverlight 3 (.NET-Framework 3.5) entwickelt. Deshalb muss zur Ausführung Silverlight 3 oder neuer auf dem Client installiert sein. Die Silverlight-Installation wird normalerweise automatisch vom Browser initiiert (Bestätigung vom Benutzer wird verlangt), sobald das BOS-WebAdmin aufgerufen wird. Die WCF-Contracts der Serverschnittstelle müssen von der XWP-ProjectManager implementation übernommen werden. Es wird der aktuelle Stand von XWP V4.1 verwendet. Laufende Änderungen müssen nicht übernommen werden.

3. Spezifische Anforderungen

3.1. Funktionale Anforderungen

Die funktionalen Anforderungen werden durch die Use Cases in Kapitel 4 beschrieben. Hier werden aber zusätzliche funktionale Anforderungen definiert, die nicht direkt aus der Anwendersicht beschrieben werden können.

3.1.1. BOS-Server vor unberechtigtem Zugriff schützen (Login/Logout)

Da mit dem BOS-WebAdmin keine XWP-Installation mehr nötig ist und der BOS-Server möglicherweise über das Internet erreichbar ist, muss das BOS-WebAdmin vor unberechtigtem Zugriff auf den BOS-Server schützen. Der Benutzer muss sich bei jedem Zugriff mit seinen Credentials (Benutzernamen, Passwort) authentifizieren. Die Authentifizierung wird immer vom BOS-Server vorgenommen. Eine Autologin-Funktion ist aber trotzdem möglich, wenn diese nicht der Standardeinstellung entspricht. Der Benutzer muss aber auch in diesem Fall noch vom BOS-Server authentifiziert werden. Der Benutzer muss sich auch jederzeit wieder vom BOS-Server abmelden können.

3.1.2. Aktionen protokollieren

Da die Verbindung zwischen Client und BOS-Server anfällig für Unterbrüche ist, können Aktionen unangekündigt unterbrochen werden. Dabei dürfen im Fehlerfall keine Inkonsistenzen entstehen. Serverseitig wird dies bereits vom BOS gehandhabt. Zum jetzigen Zeitpunkt wird dabei aber keine richtige Transaktionsverwaltung eingesetzt, die Client und Server überspannen würde. Daher können Inkonsistenzen nicht ganz ausgeschlossen werden, was allein durch die Client-Implementation aber nicht wesentlich verbessert werden kann. Auf jeden Fall müssen alle Aktionen protokolliert werden, damit später nachvollzogen werden kann, was geschehen ist.

3.2. Bedienbarkeit

Die Bedienung soll sich soweit möglich am bestehenden XWP-ProjectManager orientieren, damit sich die Benutzer nicht völlig neu orientieren müssen. Natürlich ergeben sich allein durch die unterschiedlichen GUI-Technologien (Windows Forms und Silverlight WPF) bereits Unterschiede. So gibt es bei Silverlight z.B. kein Kontextmenu bei einem Klick mit der rechten Maustaste.

Das GUI soll vollständig Lokalisierbar sein, d.h. mehrere Sprachen unterstützen. Während der Semesterarbeit muss das GUI aber nur in englischer Sprache vorhanden sein.

3.3. Zuverlässigkeit

n/a

3.4. Leistung

Die Ansprechbarkeit des Userinterfaces soll ohne menschlich wahrnehmbare Verzögerung erfolgen und zu jedem Zeitpunkt gegeben sein. Falls das Userinterface gerade nicht auf Benutzeraktionen antworten kann, muss dies dem Benutzer klar ersichtlich gemacht werden (Sanduhr, Progressbar, etc.). Da die Dauer der einzelnen Aktionen massgeblich vom Datendurchsatz/Latenzzeiten der

Verbindung mit dem BOS-Server abhängt, können keine genaueren Zeitangaben für die Ausführung einer Standardaktion (z.b. Browsen, Undo-checkout) gemacht werden. Clientseitig wird fast kein Speicherplatz benötigt. Lediglich eine XML-Konfigurationsdatei zum Speichern der Optionen/Serverliste etc. wird lokal abgelegt in einem speziellen, von Silverlight zur Verfügung gestellten Verzeichnis. Die Anzahl Benutzer und Projekte, die mit dem BOS-WebAdmin verwaltet werden können soll nicht limitiert sein. Genaue Angaben müssen nicht gemacht werden, da diese Werte in einem relativ überschaubaren Bereichen liegen. So hostet z.B. der Schweizer BOS-Server ca. 2000 Projekte und hat knapp 200 Benutzer. Für grössere Länder sind diese Werte maximal um das 5 bis 10-fache höher. Es gibt also mit Sicherheit keinen Server, der 10 oder 100-tausende von Projekten hosten muss.

Ein wichtiger Punkt ist auch die parallele Benutzung von XWP-ProjectManager und BOS-WebAdmin und dies mit mehreren Benutzern gleichzeitig. Die Concurrency wird grundsätzlich vom Server gehandelt, aber der Client muss zumindest davon ausgehen, dass etwas „schief“ gehen könnte und dem entsprechend Rechnung tragen. So können sich bei vielen gleichzeitigen Benutzern u.a. auch die Antwortzeiten des Servers vergrössern.

3.5. Datenschutz

Benutzer müssen vom BOS-Server mit ihren Credentials (Benutzername, Passwort) authentifiziert werden. Benutzer dürfen nur diese Aktionen durchführen, für die sie die Rechte besitzen (Autorisierung). Diese Rechte werden serverseitig geprüft, aber um den Benutzer nicht zu verwirren, sollten die Rechte auch schon clientseitig geprüft werden, damit nur die Aktionen ausgeführt werden können, zu denen der Benutzer berechtigt ist.

Die gesamte Kommunikation zwischen Client und Server erfolgt über HTTPS. Es werden client- und serverseitige Zertifikate verwendet.

3.6. Datensicherheit

Clientseitig nicht besonders relevant. Zur Nachverfolgbarkeit soll aber ein Audit erstellt werden (Protokollieren von Aktionen).

3.7. Wartbarkeit

Die gewählte Softwarearchitektur soll einfach erweiterbar sein und so das Hinzufügen von zukünftigen Features so einfach als möglichst gestalten. Das Produkt muss über mehrere Zyklen wartbar sein.

3.8. Schnittstellen

Die Kommunikation mit dem BOS-Server erfolgt via WCF und wsHttps-Binding. Um die Verbindung mit dem BOS-Server herzustellen, werden Hostname, Benutzername und Passwort benötigt.

3.9. Lizenzanforderungen

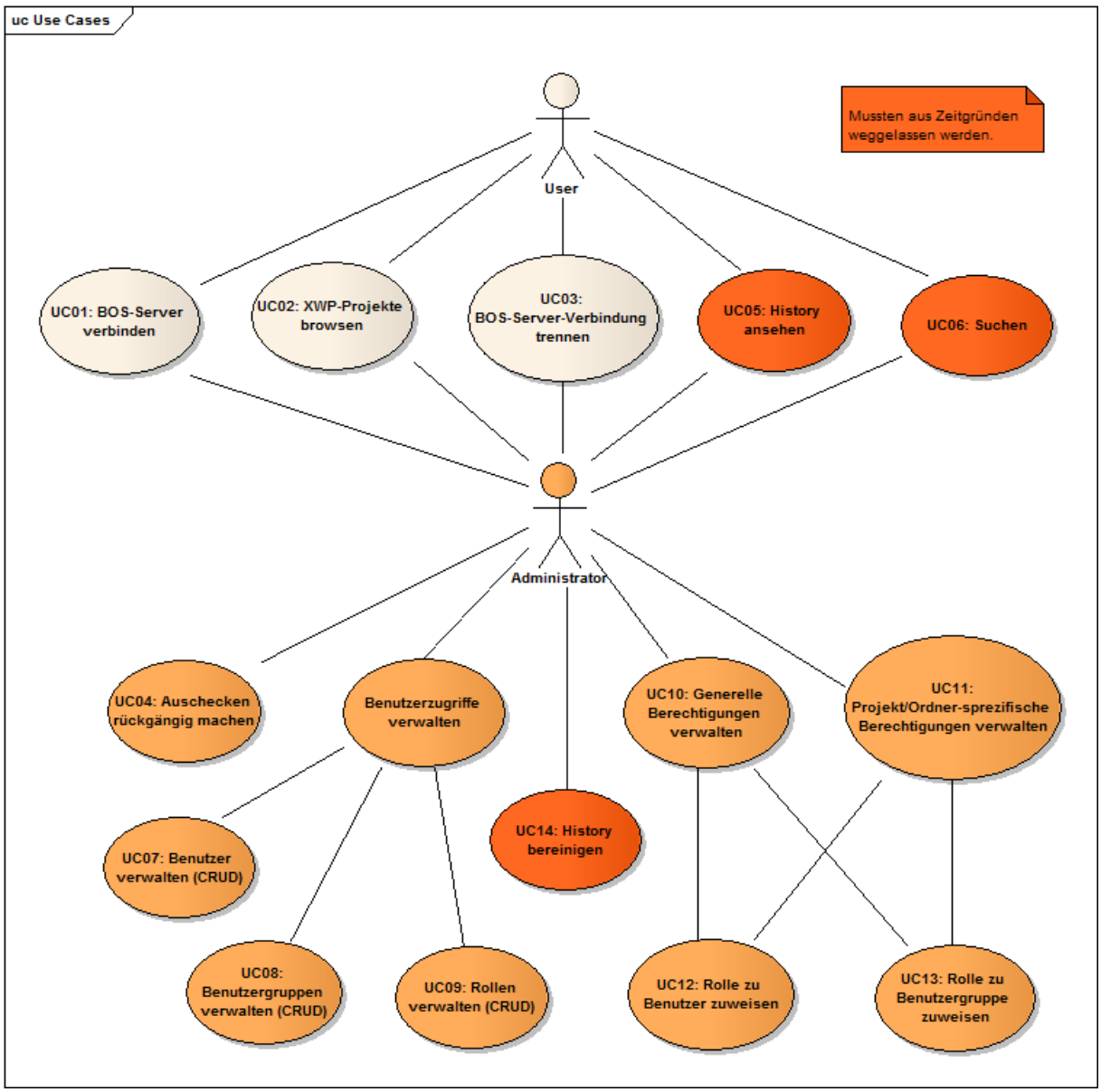
Es werden keine zusätzlichen Lizenzen benötigt, da alle benötigten Zusatzkomponenten als Open-Source verfügbar sind. Das Produkt selbst untersteht den Lizenzbedingungen der Siemens Schweiz AG.

3.10. Verwendete Standards

Standard	Verwendung
Silverlight 3.0	BOS-WebAdmin wird in C# und Silverlight 3.0 entwickelt.
.NET Framework 3.5	Serverseitig wird das .NET Framework 3.5 verwendet.
XML	Alle Konfigurations- und Textdaten werden clientseitig in XML-Dateien abgelegt.
WCF	Die Kommunikation zwischen Client und Server erfolgt über die Kommunikationsplattform WCF (Windows Communication Foundation).
HTTPS	Die Kommunikation zwischen Client und Server wird über HTTPS gesichert.

4. Use Cases

4.1. Use Case Diagramm



4.2. Aktoren

Aktoren sind der Administrator sowie der normale User. Als „User“ gilt jeder Benutzer, der auf dem BOS Server bekannt ist, jedoch keine Administratorrechte hat. D.h. ein User hat grundsätzlich nur Lese-Berechtigung abgesehen von den Daten, von denen er der Ersteller bzw. Besitzer ist. Als „Administrator“ gilt hingegen jeder Benutzer, der neben den gewöhnlichen User-Rechten mindestens ein Admin-Recht besitzt.

4.3. Use Case Notation

Um die Startprozedur des BOS-WebAdmins abzukürzen, wird folgende Notation verwendet:

Benutzer startet den BOS-WebAdmin.

Das beinhaltet das starten eines von Silverlight 3 unterstützten Webbrowsers (IE, Firefox, Safari), das Installieren des Silverlight 3 Plugins (falls nicht bereits installiert) sowie das eigentliche Starten des BOS-WebAdmin (mit dem Webbrowser zur richtigen URL navigieren).

Damit es keine Verwechslung zwischen dem Benutzer vom UseCase her gesehen und dem Benutzer in der BOS Datenhaltung gibt, wird der Begriff User für den BOS-Benutzer verwendet.

4.4. Generelle Fehlerbehandlung

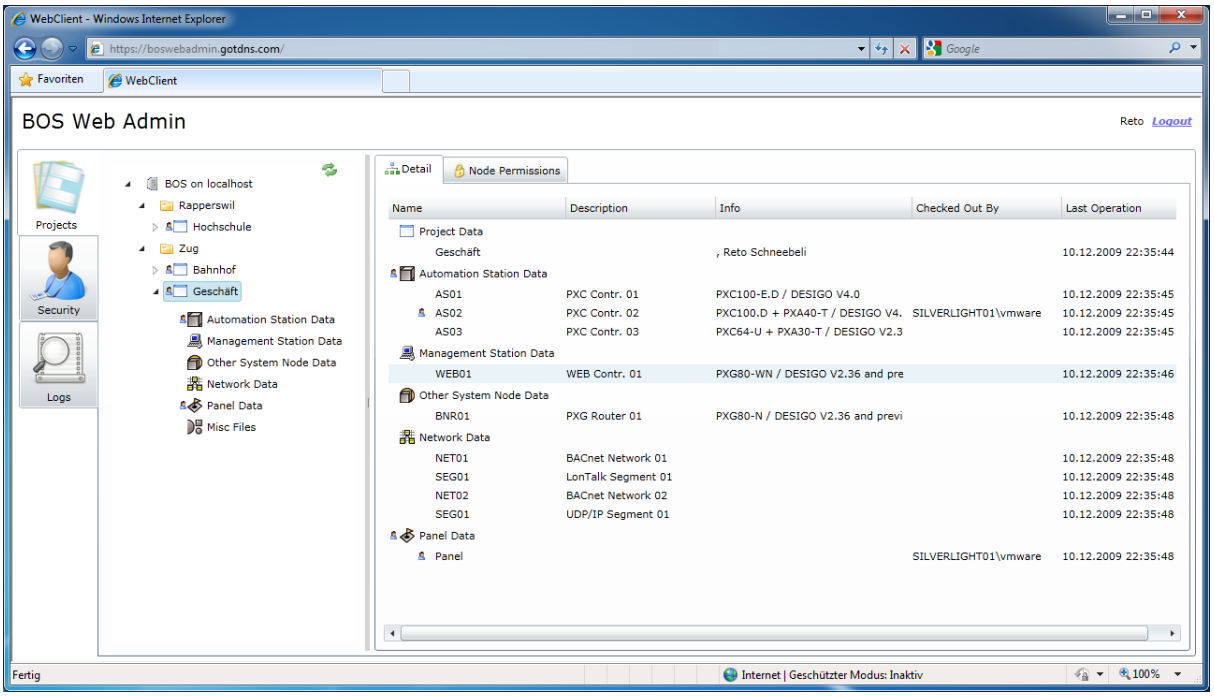
Folgende Fehlertypen können jederzeit auftreten und gelten somit für alle UseCases.

- **Network (Connection Error)**
  Fehler die mit der Netzwerkverbindung zu tun haben (TCP, IP).
- **WCF (Communication Error)**
  Fehler die mit der WCF Kommunikation zu tun haben.
- **Application Error**
  Fehler welche die Client Applikation selbst betreffen.

Bei allen Fehlertypen wird direkt eine Meldung angezeigt.

Wenn ein User für eine Operation (UseCase) nicht genügend Rechte hat, steht die Operation gar nicht erst zur Verfügung, oder es erscheint eine Fehlermeldung. Die 'Alternative Flows' bei zu wenig Recht, werden in den UseCases weggelassen.

4.5. UI Storyboard



4.6. UC01: BOS-Server verbinden

Overview	Der Benutzer verbindet mit einem BOS-Server (Login). Nach erfolgreicher Authentifizierung des Users wird der Projektbaum geladen und angezeigt.
Preconditions	Die URL zum BOS-Server sowie Benutzername und Passwort sind bekannt.
Postconditions	Verbindung zum BOS ist hergestellt und der Projektbaum wird angezeigt (nur oberster Level). Programm ist bereit für weitere Operationen.

Main Success Scenario	**Extensions or Alternative Flows**

1. Benutzer startet den BOS-WebAdmin. 2. Benutzer wählt einen BOS-Server. 3. Benutzer gibt Benutzername und Passwort ein. 4. Benutzer wählt, ob die Anmeldeinformationen gespeichert werden sollen. 5. System initialisiert die Verbindung mit dem BOS-Server (User wird authentifiziert). 6. System zeigt den Projektbaum an.	2a. BOS-Server ist nicht in Liste. 1. Benutzer erstellt neuen BOS-Server. 2. Benutzer gibt BOS-Servernamen, URL und Beschreibung ein. 3. System fügt BOS-Server der Auswahl-Liste hinzu und wählt diesen als aktuellen Server. 2a. BOS-Server ist nicht mehr aktuell. 1. Benutzer wählt einen BOS-Server. 2. Benutzer aktualisiert BOS-Servernamen, URL und Beschreibung. 3. System übernimmt die Daten und wählt den Server automatisch als aktuellen Server aus. 2a. BOS-Server muss gelöscht werden. 1. Benutzer wählt den zu löschenden BOS-Server. 2. System entfernt BOS-Server aus Liste. 5-6b. Authentifizierung nicht erfolgreich. 1. Benutzer erhält eine Meldung.
Special Requirements	Vom Benutzer vorgenommene Änderungen werden laufend im Hintergrund gespeichert.
Technology and Data Variations List	BOS-Serverliste sowie Anmeldeinformationen werden lokal im XML Format gespeichert.
Frequency of Occurrence	Nicht massgebend.
Open Issues	

4.7. UC02: XWP-Projekte browsen

Overview	Der Benutzer kann durch den Projektbaum navigieren, ähnlich wie beim Windows Explorer, und so den gewünschten Ordner, Projekt oder Sub-Node (AS, MS, Network, etc.) auswählen. Für einen angewählten Node werden jeweils die entsprechenden Detailinformationen angezeigt. Dieser Use Case ist die Voraussetzung für fast alle weiteren Use Cases.
Preconditions	UC01
Postconditions	Letzer angewählte Node im Projektbaum ist markiert und die entsprechenden Detailinformationen werden in der Content-Area angezeigt. Programm ist bereit für weitere Operationen.

Main Success Scenario	**Extensions or Alternative Flows**
1. Benutzer navigiert durch den Projektbaum.	1a. Verbindung zum BOS wird unterbrochen. 1. Benutzer erhält eine Meldung. 2. Falls vom Benutzer gewünscht, kann das System versuchen, die Verbindung wieder aufzubauen.

Special Requirements	
Technology and Data Variations List	
Frequency of Occurrence	Nicht massgebend.

Open Issues	

4.8. UC03: BOS-Server-Verbindung trennen

Overview	Der Benutzer trennt die Verbindung zum BOS-Server (Abmelden). Nachdem die Verbindung getrennt wurde, wird der Login-Bildschirm angezeigt.	
Preconditions	UC01	
Postconditions	Login-Bildschirm wird angezeigt. Benutzer hat die Möglichkeit, erneut mit demselben oder einem anderen BOS-Server zu verbinden.	
Main Success Scenario		**Extensions or Alternative Flows**
1. Benutzer meldet sich vom BOS-Server ab. 2. System trennt die Verbindung mit dem BOS-Server 3. System zeigt Login-Bildschirm an.		
Special Requirements		
Technology and Data Variations List		
Frequency of Occurrence	Nicht massgebend.	
Open Issues		

4.9. UC04: Auschecken rückgängig machen (Undo check-out)

Overview	Der Benutzer selektiert ein ausgechecktes Projekt oder ein oder mehrere ausgecheckte Sub-Nodes (AS, MS, Network, etc.) unterhalb eines Projektes und macht das Auschecken rückgängig. Der „Check-out"-Status wird von den gewählten Nodes entfernt. Damit werden die Projekte für andere Benutzer wieder bearbeitbar.	
Preconditions	UC01, Entsprechende Rechte	
Postconditions	Die selektierten Projekte oder Sub-Nodes sind nicht mehr ausgecheckt und können nun von anderen Benutzern ausgecheckt und bearbeitet werden.	
Main Success Scenario		**Extensions or Alternative Flows**
1. Benutzer selektiert ein Projekt, oder ein oder mehrere Sub-Nodes des selben Projektes. 2. Benutzer wählt „Undo check-out". 3. System entfernt den „Check-out"-Status der gewählten Nodes.		1a. Das selektierte Projekt oder ein selektierter Sub-Node ist nicht ausgecheckt. 1. Die „Undo check-out" Funktion steht nicht zur Verfügung (Button disabled). 1b. Die selektierten Sub-Nodes sind nicht vom selben Projekt. 1. Die „Undo check-out" Funktion steht nicht zur Verfügung (Button disabled). 3a. Das Projekt oder ein Node ist nicht mehr ausgecheckt. Wurde bereits von einem anderen Benutzer eingecheckt, oder ein undo checkout ausgeführt. 1. Benutzer erhält eine Meldung.

Special Requirements	Muss funktionieren, auch wenn gleichzeitig ein anderer User ein „Check-in oder „Undo check-out“ durchführt.
Technology and Data Variations List	
Frequency of Occurrence	Nicht massgebend.
Open Issues	

4.10. UC05: History ansehen (wird nicht umgesetzt)

Overview	Der Benutzer kann sich die Vergangenheit (History) eines Projektes oder des kompletten BOS-Servers ansehen. Zur Einschränkung stehen drei Kriterien zur Verfügung: User, Operation oder Zeitperiode. Alle drei können miteinander verknüpft werden. Als Resultat wird eine Liste mit den gefundenen History-Einträgen angezeigt. (Die History auf einem Ordner wird momentan noch nicht unterstützt.)
Preconditions	UC01
Postconditions	Eine Liste mit History-Einträgen wird angezeigt und beinhaltet folgende Informationen: Zeitpunkt, Operation, Benutzername, Node-Name und Projektname.

Main Success Scenario	**Extensions or Alternative Flows**
1. Benutzer wählt "Show History". 2. Benutzer schränkt die Kriterien nicht weiter ein. 3. System zeigt alle History-Einträge.	4a. Benutzer möchte die History nur von einem Projekt. 1. Benutzer selektiert ein Projekt. 2. Jetzt folgt Schritt 1-3 des 'Main Success Scenario'. 4a. Benutzer wählt nur einen gewissen User aus und/oder Benutzer wählt nur eine bestimmte Operation und/oder Benutzer wählt nur eine bestimmte Zeitperiode. a. System zeigt nur die entsprechenden History-Einträge an.

Special Requirements	
Technology and Data Variations List	
Frequency of Occurrence	Nicht massgebend.
Open Issues	

4.11. UC06: Suchen (wird nicht umgesetzt)

Overview	Der Benutzer selektiert einen Ordner oder ein Projekt und kann in allen darunter liegenden Nodes nach einem Text suchen. Es kann in mehreren Textelementen (ProjectName, ProjectVersion, NodeName, ...) kombiniert gesucht werden. Als Resultat werden alle Nodes aufgelistet die den Suchkriterien entsprechen.
Preconditions	UC01

Postconditions	Eine Liste mit alle Nodes die den Suchkriterien entsprechen wird angezeigt und beinhaltet folgende Informationen: Status, Node-Name, Node-Typ, Projektname und Zeitpunkt der letzten Operation.

Main Success Scenario	**Extensions or Alternative Flows**
1. Benutzer selektiert einen Ordner oder ein Projekt. 2. Benutzer wählt ein Suchkriterium und gibt einen Suchtext ein. 3. System sucht nach Nodes die dem Suchkriterium entsprechen. 4. System zeigt eine Liste mit den entsprechenden Suchresultate.	2a. Benutzer definiert weitere Suchkriterien. 1. System sucht nach Nodes die den Suchkriterien entsprechen. 2. System zeigt eine Liste mit den entsprechenden Suchresultate 3a. Keine Suchresultate. 1. Benutzer erhält eine Meldung.

Special Requirements	Die Suchkriterien werden mit einem logischen UND verknüpft.
Technology and Data Variations List	
Frequency of Occurrence	Nicht massgebend.
Open Issues	

4.12. UC07: Benutzer verwalten (CRUD)

Overview	Der Benutzer kann einen User erstellen, bearbeiten oder löschen und Benutzergruppen zuweisen. Ein User hat einen Benutzernamen, ein Passwort, eine Email-Adresse, einen Vornamen, einen Nachnamen und einen Kommentar.
Preconditions	UC01
Postconditions	Der Benutzer ist erstellt, verändert oder gelöscht. Gruppen sind gegebenenfalls zugewiesen.

Main Success Scenario	**Extensions or Alternative Flows**
1. Benutzer wechselt ins Tab Security. 2. System zeigt die drei Tabs User, User, Groups und Roles an. 3. Benutzer wechselt ins Tab Users. 4. Benutzer fügt einen User hinzu, indem er Benutzernamen, Passwort, Email-Adresse, Vornamen, Nachnamen und Kommentar eingibt. 5. Benutzer/System speichert alle Änderungen.	4a. Benutzer möchte einen User bearbeiten. a. Benutzer selektiert den User. b. Benutzer bearbeitet den User. 4b. Benutzer möchte einen User löschen. a. Benutzer selektiert den User. b. Benutzer löscht den User. 4c. Benutzer möchte zugewiesene Benutzergruppen hinzufügen/entfernen. a. Benutzer selektiert den User. b. Benutzer fügt hinzu oder entfernt zugewiesene Benutzergruppen.

Special Requirements	
Technology and Data Variations List	
Frequency of Occurrence	Nicht massgebend.

Open Issues	

4.13. UC08: Benutzergruppen verwalten (CRUD)

Overview	Der Benutzer kann eine Benutzergruppe erstellen oder löschen und User zuweisen. Ein Benutzergruppe hat nur einen Namen.
Preconditions	UC01
Postconditions	Die Benutzergruppe ist erstellt oder gelöscht. User sind gegebenenfalls zugewiesen.

Main Success Scenario	**Extensions or Alternative Flows**
1. Benutzer wechselt ins Tab Security. 2. System zeigt die drei Tabs User, User Groups und Roles an. 3. Benutzer wechselt ins Tab Groups. 4. Benutzer fügt eine Benutzergruppe hinzu, indem er einen Namen eingibt. 5. Benutzer/System speichert alle Änderungen.	4a. Benutzer möchte eine Benutzergruppe löschen. a. Benutzer selektiert die Benutzergruppe. b. Benutzer löscht die Benutzergruppe. 4b. Benutzer möchte zugewiesene User hinzufügen/entfernen. a. Benutzer selektiert die Benutzergruppe. b. Benutzer fügt hinzu oder entfernt zugewiesene User.

Special Requirements	
Technology and Data Variations List	
Frequency of Occurrence	Nicht massgebend.
Open Issues	

4.14. UC09: Rollen verwalten (CRUD)

Overview	Der Benutzer kann eine Rolle erstellen, bearbeiten oder löschen und Rechte zuweisen. Ein Rolle hat einen Namen und eine Beschreibung.
Preconditions	UC01
Postconditions	Die Rolle ist erstellt, verändert oder gelöscht. Rechte sind gegebenenfalls zugewiesen.

Main Success Scenario	**Extensions or Alternative Flows**
1. Benutzer wechselt ins Tab Security. 2. System zeigt die drei Tabs User, User Groups und Roles. 3. Benutzer wechselt ins Tab Roles. 4. Benutzer fügt eine Rolle hinzu, indem er einen Namen und eine Beschreibung eingibt. 5. Benutzer/System speichert alle Änderungen.	4a. Benutzer möchte eine Rolle bearbeiten. a. Benutzer selektiert die Rolle. b. Benutzer bearbeitet die Rolle. 4b. Benutzer möchte eine Rolle löschen. a. Benutzer selektiert die Rolle. b. Benutzer löscht die Rolle. 4c. Benutzer möchte zugewiesene Rechte hinzufügen/entfernen. a. Benutzer selektiert die Rolle. b. Benutzer fügt hinzu oder entfernt zugewiesene Rechte.

Special Requirements	
Technology and Data Variations List	
Frequency of Occurrence	Nicht massgebend.
Open Issues	

4.15. UC10: Generelle Berechtigungen verwalten

Overview	Der Benutzer weist Rollen an Benutzer oder Benutzergruppen zu, die dann die applikationsweiten Berechtigungen wie z.B. das Verwalten der Benutzerrechte definieren.	
Preconditions	UC01	
Postconditions	Die applikationsweiten Berechtigungen sind geändert.	
Main Success Scenario		**Extensions or Alternative Flows**
1. Benutzer wechselt zu den 'Application Permissions'. 2. UC13 Rolle zu Benutzergruppe zuweisen. 3. Benutzer/System speichert alle Änderungen.		2a. Benutzer möchte Rolle auf einen User zuweisen. a. UC12 Rolle zu Benutzer zuweisen
Special Requirements		
Technology and Data Variations List		
Frequency of Occurrence	Nicht massgebend.	
Open Issues		

4.16. UC11: Projekt/Ordner-spezifische Berechtigungen verwalten

Overview	Der Benutzer weist Rollen an Benutzer oder Benutzergruppen zu, die nur für ein bestimmtes Projekt oder Ordner gelten.	
Preconditions	UC02 (Ein Projekt oder ein Ordner ist selektiert)	
Postconditions	Die Berechtigungen für das selektierte Projekt oder den Ordner sind geändert.	
Main Success Scenario		**Extensions or Alternative Flows**
4. Benutzer wählt 'Permissions'. 5. UC13 Rolle zu Benutzergruppe zuweisen. 6. Benutzer/System speichert alle Änderungen.		2b. Benutzer möchte Rolle auf einen User zuweisen. a. UC12 Rolle zu Benutzer zuweisen
Special Requirements		
Technology and Data Variations List		

Frequency of Occurrence	Nicht massgebend.
Open Issues	

4.17. UC12: Rolle zu Benutzer zuweisen

Overview	Der Benutzer weist eine Rolle einem User zu.	
Preconditions	UC10 oder UC11	
Postconditions	Dem User ist die Rolle zugewiesen.	
Main Success Scenario		**Extensions or Alternative Flows**
1. Benutzer wechselt ins Tab Users. 2. Benutzer wählt einen User aus. 3. Benutzer weist eine Rolle hinzu.		3a. Benutzer möchte Rolle für einen User entfernen. a. Benutzer entfernt die Zuweisung zur Rolle.
Special Requirements		
Technology and Data Variations List		
Frequency of Occurrence	Nicht massgebend.	
Open Issues		

4.18. UC13: Rolle zu Benutzergruppe zuweisen

Overview	Der Benutzer weist eine Rolle einer Benutzergruppe zu.	
Preconditions	UC10 oder UC11	
Postconditions	Der Benutzergruppe ist die Rolle zugewiesen.	
Main Success Scenario		**Extensions or Alternative Flows**
1. Benutzer wechselt ins Tab User Groups. 2. Benutzer wählt eine Benutzergruppe aus. 3. Benutzer weist eine Rolle hinzu.		3b. Benutzer möchte Rolle für eine Benutzergruppe entfernen. a. Benutzer entfernt die Zuweisung zur Rolle.
Special Requirements		
Technology and Data Variations List		
Frequency of Occurrence	Nicht massgebend.	
Open Issues		

4.19. UC14: History bereinigen (wird nicht umgesetzt)

Overview	Der Benutzer bereinigt die Server-Logs.

Projekt: BOS-WebAdmin

Domainanalyse

[Dokumentstruktur basiert auf RUP „Domainanalyse“ und Studentenvorlage]

0. Dokumentinformationen

0.1. Änderungsgeschichte

Datum	*Version*	*Änderung*	*Autor*
30.09.2009	0.1	Domainmodell erstellt + Beschreibungen	Reto Schneebeli
07.10.2009	0.2	Review Kommentare einarbeitet + Beschreibungen	Reto Schneebeli
13.10.2009	0.3	Genauere Beschreibung der Beziehung zwischen User/Gruppen, Rollen und Berechtigungen.	Reto Schneebeli
16.10.2009	0.4	Review Kommentare von René einarbeitet	Reto Schneebeli

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. Domain Modell 5
2.1. Strukturdiagramm 5
2.2. Konzeptbeschreibung 6
2.2.1. BOSServer 6
2.2.2. StructureFolder 6
2.2.3. Project 7
2.2.3.1. NetworkData 7
2.2.3.2. PanelData 7
2.2.4. SystemNode 7
2.2.4.1. Misc 7
2.3. Rechtemodell 8

1. Einführung

1.1. Zweck

Dieses Dokument beinhaltet das Domainmodell und deren Beschreibung und bildet die Grundlage für das spätere Design.

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**
BOS	**B**ranch **O**ffice **S**erver (Zentrale Ablage der XWORKS Projekte)

1.4. Referenzen

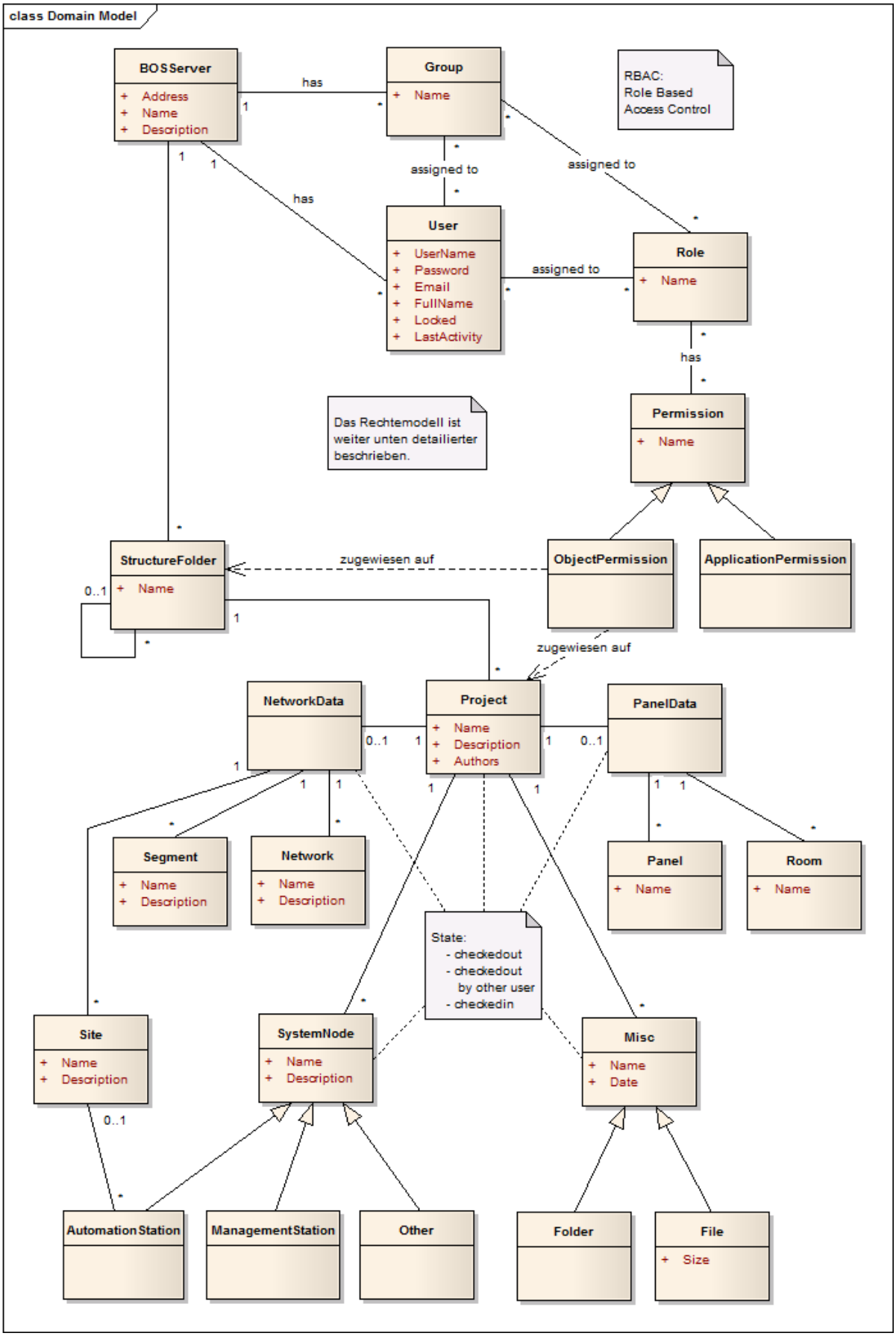
Aufgabenstellung.pdf

1.5. Übersicht

Im nächsten Kapitel wird das Domainmodell beschrieben.

2. Domain Modell

2.1. Strukturdiagramm



2.2. Konzeptbeschreibung

Domainmodell kurz in Worten beschrieben.

- Das Security Modell entspricht in etwa dem RBAC, Role Based Access Control, mit dem Unterschied, dass als Subjekt entweder User oder Gruppen verwendet werden.
- Der *BOSServer* hat Benutzergruppen (*Group*) und *User*. Der *User* ist gewissen Gruppen zugewiesen. *Group* und *User* haben Rollen (*Role*) welche wiederum Berechtigungen (*Permission*) haben.
- Es gibt zwei Typen von Berechtigungen, *ObjectPermission* und *ApplicationPermission*. *ApplicationPermission* sind allgemeine Berechtigungen und *ObjectPermission* sind Berechtigungen auf *StructureFolder* oder *Project*. Solange für ein User oder eine Gruppe auf ein Objekt keine *ObjectPermission* vorhanden ist, gilt die *ObjectPermission* des Parents.
- Ein *BOSServer* hat mehrere Verzeichnisse (*StructureFolder*) welche wiederum Unterverzeichnisse (*StructureFolder*) oder Projekte (*Project*) haben.
- Das *Project* hat vier verschiedene Arten von Daten, Netzwerkdaten (*NetworkData*), Paneldaten (*PanelData*), Systemnode-Daten (*SystemNode*) und diverse Verzeichnisse oder Dateien (*Misc*).
- *NetworkData* und *PanelData* gibt es ein oder keinmal in einem Projekt, *SystemNode* oder *Misc* kann es beliebig viele geben. *Misc* ist entweder ein *Folder* oder ein *File*.
- *SystemNode* ist eine Automationsstation (*AutomationStation*) eine Managementstation (*ManagementStation*) oder ein anderer Systemnode (*Other*), z.B. Router.
- Das *Project*, *NetworkData*, *PanelData*, *SystemNode* und *Misc* können ausgecheckt sein und haben somit den Status "checkedout" oder "checkedin".
- Enitäten die individuell ein- oder ausgechecked werden können sind: *ProjectData*, *NetworkData*, *PanelData*, *SystemNodes* (Controller, Mgmt Station etc) oder *Misc* Daten.
- Ein Projekt kann mit all seinen zugehörigen Nodes zusammen ein- oder ausgecheckt werden.

2.2.1. BOSServer

Beschreibung	Der BOSServer definiert den Server auf dem die Projekte und User verwaltet sind.
Attribute	• *String* Address: URL des Servers. • *String* Name: Name des Servers. • *String* Description: Beschreibung des Servers.
Beziehungen	• 1:n Beziehung mit Group: Der BOS hat mehrere Benutzergruppen, welche aber nur zu einem Server gehören. • 1:n Beziehung mit User: Der BOS hat mehrere Benutzer, welche aber nur zu einem Server gehören. • 1:n Beziehung mit StructureFolder: Der BOS hat zur Strukturierung einen Root Ordner welcher wiederum mehrere Unterordner haben kann.

2.2.2. StructureFolder

Beschreibung	Der StructureFolder bezeichnet einen Ordner zur Strukturierung der Projekte.
Attribute	• *String* Name: Name des Ordners.
Beziehungen	• 0..1:n Beziehung mit StructureFolder: Ein Ordner kann mehrere Unterordner haben, hat jedoch nur ein oder kein Parent Ordner. • 1:n Beziehung mit Project: Der Ordner hat mehrere Projekte, welche aber immer zu genau einem Ordner gehören. • n:1 Beziehung mit BOSServer: Ein Ordner gehören zu einem BOS.

2.2.3. Project

Beschreibung	Das Project beinhaltet die Netzwerkdaten, Paneldaten, Systemnodedaten und sonstigen Daten (Ordner, Dateien).
Attribute	• *String* Name: Name des Projektes. • *String* Description: Beschreibung des Projektes. • *String* Authors: Liste der Autoren (Ist in der lokalen Datenhaltung als eine Liste von Autoren abgebildet.
Beziehungen	• 1:0..1 Beziehung mit NetworkData: Ein Projekt kann Netzwerkdaten enthalten, muss jedoch nicht. • 1:0..1 Beziehung mit PanelData: Ein Projekt kann Paneldaten enthalten, muss jedoch nicht. • 1:n Beziehung mit NetworkData: Ein Projekt kann mehrere Systemnodes haben, welche wiederum nur zu einem Projekt gehören. • 1:n Beziehung mit NetworkData: Ein Projekt kann mehrere sonstige Daten enthalten (Ordner, Dateien), welche jedoch genau zu einem Projekt gehören. • n:1 Beziehung mit StructureFolder: Ein Projekt gehören zu einem Ordner.

2.2.3.1. NetworkData

Beschreibung	Die NetworkData beschreiben die Netzwerke sowie die Segmente (LON, IP) in einem Projekt.
Attribute	Keine, da es vor allem der Strukturierung dient.
Beziehungen	• 1:n Beziehung mit Network: Die Netzwerkdaten können mehrere Netzwerke haben. • 1:n Beziehung mit Segment: Die Netzwerkdaten können mehrere Segmente haben. • n:1 Beziehung mit Project: Die Netzwerkdaten gehören zu einem Projekt.

2.2.3.2. PanelData

Beschreibung	Die PanelData beschreiben die Räume und Schaltschränke in einem Projekt.
Attribute	Keine, da es vor allem der Strukturierung dient.
Beziehungen	• 1:n Beziehung mit Room: Die Paneldaten können mehrere Räume haben. • 1:n Beziehung mit Panel: Die Paneldaten können mehrere Panel haben. • n:1 Beziehung mit Project: Die Paneldaten gehören zu einem Projekt.

2.2.4. SystemNode

Beschreibung	Ein SystemNode ist eine Automationsstation eine Managementstation oder ein anderer Node, z.B. ein Router.
Attribute	• *String* Name: Name des Systemnodes. • *String* Description: Beschreibung des Systemnodes.
Beziehungen	• n:1 Beziehung mit Project: Ein Systemnode gehört zu einem Projekt.

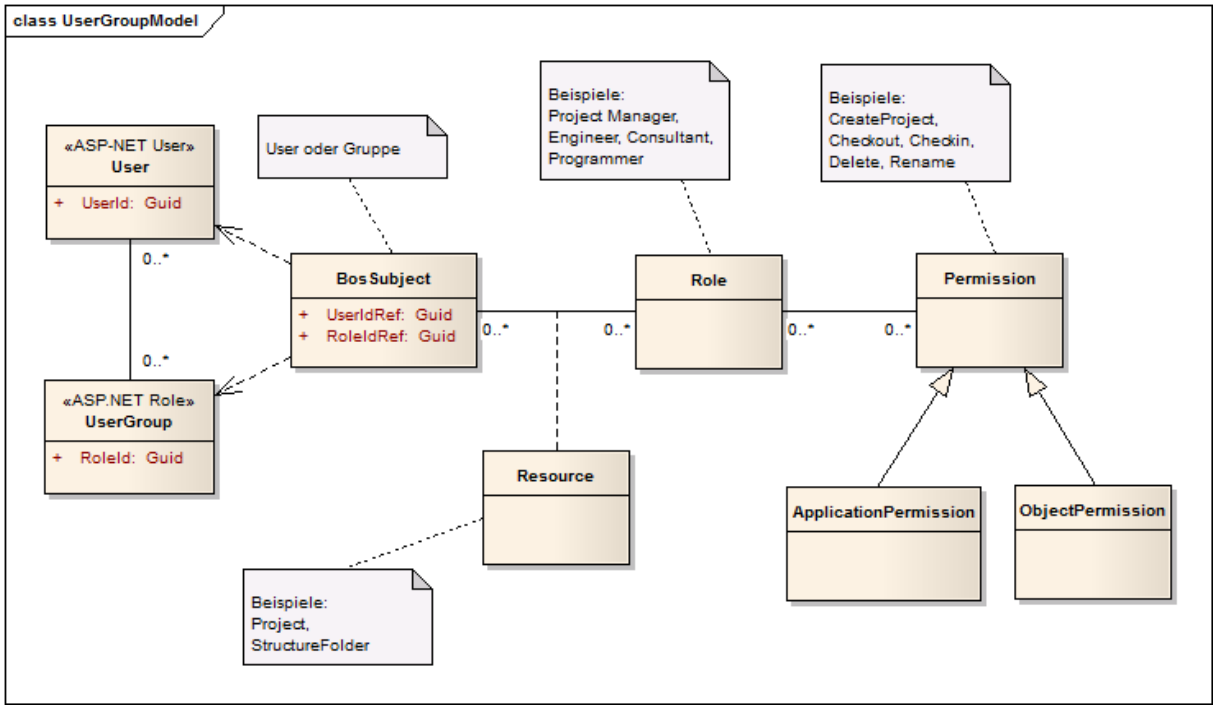
2.2.4.1. Misc

Beschreibung	Misc beschreibt entweder eine Datei oder ein Ordner und wird vom System nicht weiter behandelt (Blackbox Daten).
Attribute	• *String* Name: Name der Datei oder des Ordners.

	• *DateTime* Date: Datum und Zeit der letzten Änderung.
Beziehungen	• n:1 Beziehung mit Project: Die Ordner/Dateien gehören zu einem Projekt.

2.3. Rechtemodell

Grundsätzlich wird einem User oder einer Gruppe für eine Ressource eine Rolle zugewiesen. Die Rolle beschreibt welche Berechtigungen auf dieses Objekt gewährt werden.



Es gibt zwei Arten von Berechtigungen:

- ApplicationPermission

  Allgemeine Berechtigungen welche die Applikation und nicht einen Node betreffen. Zum Beispiel: ManageSecurity, UseHistory oder UseSearch

- ObjectPermission

  Berechtigungen welche auf einen Node (Ordner oder Projekt) gesetzt werden. Zum Beispiel: CheckIn, CheckOut, GetLatest oder UndoCheckoutOther.

Die Resource im Diagramm entspricht einem Node (Ordner oder Projekt) und wird nur für die ObjectPermission benötigt.

Projekt: BOS-WebAdmin

Software Architektur Spezifikation

[Dokumentstruktur basiert auf RUP „Software Architecture Document“]

0 Dokumentinformationen

0.1 Änderungsgeschichte

Datum	*Version*	*Änderung*	*Autor*
16.10.2009	0.1	Initial Version + Erste Architektur Diagramme	Reto Schneebeli
17.10.2009	0.2	Namespace/Ordner Struktur und View - Comand - Service Diagramm hinzugefügt	Reto Schneebeli
	0.3	Einschränkungen WCF+Silverlight	Kaspar Fenner
31.10.2009	0.4	Einschränkungen, Klassenübersichts-Diagramm und Beschreibungen	Reto Schneebeli
01.11.2009	0.5	Klassendiagramme und Beschreibungen des Presentation und Application Layers hinzugefügt und Error Handling	Reto Schneebeli
01.11.2009	0.6	Asynchrone Servicearchitektur, Serverseitige Anpassungen, Übersicht Application Core, Einschränkungen Async-Pattern, Klassenübersicht erweitert, weitere Diagramme und Texte angepasst	Kaspar Fenner
23.11.2009	0.7	Kapitel 8: Diagramm hinzugefügt	Reto Schneebeli
23.11.2009	0.8	Kapitel 9 hinzugefügt	Kaspar Fenner
07.12.2009	0.9	Kapitel 4.9 Domain Layer hinzugefügt, Klassendiagramme aktualisiert	Kaspar Fenner
08.12.2009	1.0	Kommentare eingearbeitet und Klassendiagramme aktualisiert	Reto Schneebeli

0.2 Inhalt

0 Dokumentinformationen 2
0.1 Änderungsgeschichte 2
0.2 Inhalt 3
0.3 Abbildungen 4
1 Einführung 6
1.1 Zweck 6
1.2 Gültigkeitsbereich 6
1.3 Definitionen und Abkürzungen 6
1.4 Referenzen 6
1.5 Übersicht 6
2 Architektonische Darstellung 7
3 Architektonische Ziele & Einschränkungen 7
3.1 Einschränkung durch Silverlight 7
3.1.1 Weak-References 7
3.1.2 Buttons auf Commands binden 7
3.1.3 Databinding 7
3.1.4 WCF (Binding und Security) 8
3.1.5 WCF (Async-Pattern) 8
4 Logische Architektur 9
4.1 Übersicht (Layer) 9
4.1.1 View-Injection (Named-Locations) 9
4.1.2 Low Coupling (Event-Aggregation) 9
4.1.3 Dependency-Injection (Constructor-Injection) 9
4.1.4 Separation of concerns 9
4.2 Namespaces und Ordnerstruktur 11
4.3 Aufrufe: View - Command - Service 12
4.4 Interface Beschreibung 13
4.5 Klassenübersicht 14
4.6 Global Verfügbare Objekte 15
4.6.1 Service Locator 15
4.6.2 Logger 15
4.6.3 Decoupling durch Event-Aggregation 15
4.7 Presentation Layer 15
4.7.1 Bootstrapper 15
4.7.1.1 Beschreibung 15
4.7.1.2 Diagramme 16
4.7.2 Login Modul 16
4.7.2.1 Beschreibung 16
4.7.2.2 Diagramme 17
4.7.3 Browser Modul 18
4.7.3.1 Beschreibung 18
4.7.3.2 Diagramme 19
4.7.4 Benutzer-Management 20
4.7.4.1 Beschreibung 20
4.7.4.2 Diagramme 21
4.7.5 Benutzergruppen-Management 21
4.7.5.1 Beschreibung 21
4.7.5.2 Diagramme 22
4.7.6 Rollen-Management 22
4.7.6.1 Beschreibung 22
4.7.6.2 Diagramme 23
4.7.7 Berechtigungen-Management 23

4.7.7.1 Beschreibung ... 23
4.7.7.2 Diagramme ... 24
4.8 Übersicht Application Core ... 25
4.8.1 Domain, Communication, Infrastructure Packages ... 25
4.8.1.1 Beschreibung ... 25
4.8.1.2 Diagramme ... 25
4.9 Domain Layer ... 26
4.9.1 Browser Service ... 26
4.9.1.1 Beschreibung ... 26
4.9.1.2 Diagramme ... 27
4.9.2 Administration Service ... 28
4.9.2.1 Beschreibung ... 28
4.9.2.2 Diagramme ... 29
4.9.3 Authentication Service ... 29
4.9.3.1 Beschreibung ... 29
4.9.3.2 Diagramme ... 29
4.9.4 Authorization Service ... 30
4.9.4.1 Beschreibung ... 30
4.9.4.2 Diagramme ... 30
4.9.5 UserManagement Service ... 31
4.9.5.1 Beschreibung ... 31
4.9.5.2 Diagramme ... 31
4.9.6 Ergebnisbehandlung von Serviceoperationen ... 32
4.9.6.1 Beschreibung ... 32
4.9.6.2 Diagramme ... 32
5 Serverseitige Anpassungen ... 33
5.1 Hosting und WCF-Konfiguration ... 33
5.2 Service Contracts ... 33
5.3 Fehlerverhalten (Fault Behavior) ... 34
6 Asynchrone Servicearchitektur ... 35
7 Error Handling ... 36
8 Datenspeicherung ... 37
9 Grössen und Leistung ... 37

0.3 Abbildungen

Abbildung 1: Systemübersicht ... 7
Abbildung 2: Übersicht (Layer) ... 10
Abbildung 3: Ordnerstruktur, Namespaces ... 11
Abbildung 4: View - Command - Service ... 12
Abbildung 5: Klassenübersicht ... 14
Abbildung 6: Bootstrapper und Module ... 16
Abbildung 7: Login Modul ... 17
Abbildung 8: Browser Modul ... 19
Abbildung 9 Browser - Undocheckout ... 20
Abbildung 10 Benutzer-Management ... 21
Abbildung 11 Benutzergruppen-Management ... 22
Abbildung 12 Rollen-Management ... 23
Abbildung 13 Berechtigungen-Management ... 24
Abbildung 14: Übersicht Application Core ... 25
Abbildung 15: Browser Service ... 27
Abbildung 16: Browser Service Model ... 28
Abbildung 17: Administration Service ... 29

Abbildung 18: Authentication Service 29
Abbildung 19: Authorization Service 30
Abbildung 20: UserManagement Service 31
Abbildung 21: Ergebnisbehandlung von Serviceoperationen 32
Abbildung 22: Asynchroner Service-Aufruf 35
Abbildung 23 Fehlerbehandlung 36
Abbildung 24 Datenspeicherung 37

1 Einführung

1.1 Zweck

Dieses Dokument beschreibt die physikalische und logische Architektur vom BOS Web Admin Client. Das Dokument bildet die Grundlage für die Implementation.

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**
BOS	**B**ranch **O**ffice **S**erver (Zentrale Ablage der XWORKS Projekte)
XWP	DESIGO **XW**ORKS **p**lus ist eine Engineering-Umgebung im Bereich Gebäudeautomation der Siemens BT.
WCF	**W**indows **C**ommunication **F**oundation
CAL	**C**omposite **A**pplication **L**ibrary (Eine Library welche einem unterstützen beim Erstellen von modularen WPF oder Silverlight Applikationen.)
XAML	E**x**tensible **A**pplication **M**arkup **L**anguage

1.4 Referenzen

Anforderungsspezifikation.pdf
Domainanalyse.pdf
Studie_CompositeApplicationGuidance.pdf

1.5 Übersicht

Zuerst wird die physikalische Architektur beschrieben, darauf folgen die Architektonischen Ziele & Einschränkungen, die logische Architektur, Prozesse und Threads, Datenhaltung und zuletzt noch Angaben zu Grössen und Leistung.

2 Architektonische Darstellung

Die Silverlight Applikation *BOS-WebAdmin* wird auf einem ganz normalen Webserver gehostet. Durch Eingabe der URL im Browser des Clients wird die Applikation auf den Client Rechner heruntergeladen und gestartet. Nach der Auswahl eines BOS wird über WCF eine Verbindung zum BOS hergestellt. BOS und Webserver können auf dem selben oder auf verschiedenen Rechnern installiert sein. Es findet keine direkte Kommunikation zwischen BOS und Webserver statt. Nur der Client stellt eine Verbindung zum BOS her.

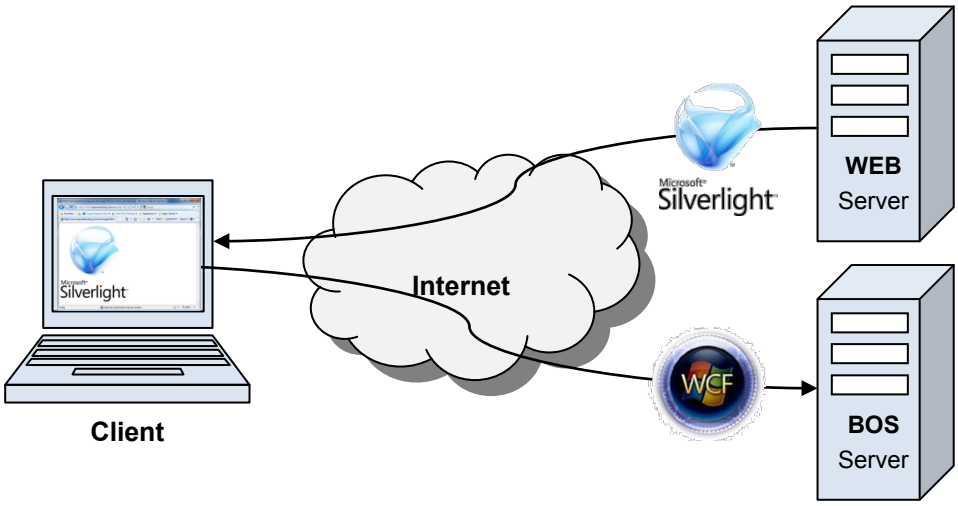


Abbildung 1: Systemübersicht

3 Architektonische Ziele & Einschränkungen

Um gewisse Einschränkungen oder Entscheide besser verstehen zu können, ist es sinnvoll einen Blick in die Studie zur Composite Application Guidance zu werfen. Der Link ist unter Referenzen aufgeführt.

3.1 Einschränkung durch Silverlight

3.1.1 Weak-References

Die Event-Aggregation-Klassen aus der CAL (Composite Application Library) unterstützen weak references. Es wäre möglich sich auf einen Event zu registrieren und anzugeben, dass diese Beziehung alleine den Objektgrafen nicht am Leben erhalten soll. In Silverlight werden die 'weak references' nicht unterstützt und deshalb können wir diese Möglichkeit auch nicht benutzen.

3.1.2 Buttons auf Commands binden

In WPF können Buttons direkt im xaml an Commands gebunden werden. Das CanExecute() wird dabei mit den Enabled/Disabled verknüpft und der Klick Event mit dem Execute(). Diese Möglichkeit wurde im Silverlight weg gelassen. Jedoch gibt es in der CAL eine Klasse, die genau diese Möglichkeit wieder anbietet. Das war eigentlich schon Grund genug um gewisse DLL's der CAL zu verwenden.

3.1.3 Databinding

Silverlight unterstützt in vielen Bereichen weniger als WPF, so existiert zum Beispiel bei der TreeView kein Property für das selektierte Element. Das bedeutet, dass mit einfachem Databinding vom Presenter her nicht herausgefunden werden kann, welches Element selektiert ist. Damit der Presenter jetzt nicht komplett von der View abhängig ist und sich direkt für den *OnSelectedItemChanged*-Event registriert, verwenden wir in solchen Fällen ein Interface mit genau den Events und Properties, welche der Presenter von der View benötigt. Der Presenter bekommt in diesem Fall mit dem Konstruktor eine View,

welche das Interface implementiert und setzt sich selbst über das Property *DataContext* der View als Datenkontext.

3.1.4 WCF (Binding und Security)

Bei Silverlight ist nur eine abgespeckte Version des .NET Frameworks verfügbar, in der unter anderem auch entscheidende Features bei der WCF-Implementierung weggelassen wurden. So wird z.B. nur basicHttpBinding als Verbindungsart unterstützt. Falls die Authentifizierungs-Features von basicHttpBinding benutzt werden, verlangt Silverlight 3 zwingend die Verwendung von HTTPS, damit die Credentials (Username/Passwort) verschlüsselt übertragen werden. Der BOS-Server ist zurzeit aber mit wsHttpBinding konfiguriert, welches eine zuverlässige, gesicherte Verbindung mit Transaktionsmanagement bietet (WS-* Spezifikation). Um Silverlight 3 zu unterstützen, sind also auch serverseitige Anpassungen nötig.

3.1.5 WCF (Async-Pattern)

Die WCF-Kommunikation in Silverlight funktioniert asynchron. Beim Versuch, die existierenden synchronen ServiceContracts zu verwenden, wirft Silverlight sofort eine Exception. D.h. die ServiceContracts, die serverseitig zur Verfügung gestellt werden, können von Silverlight nicht ohne weiteres benutzt werden. Clientseitig muss zuerst ein neuer ServiceContract erstellt werden, der die Serviceoperationen asynchron (Async-Pattern) verfügbar macht. Das Async-Pattern beschreibt, dass jede Serviceoperation durch eine Begin- und eine End-Methode ersetzt wird. D.h. serverseitig gibt es z.B. die Methode GetChildNodes(). Diese wird clientseitig in BeginGetChildNodes() und EndGetChildNodes() aufgeteilt. Dabei gibt die Begin-Methode ein IAsyncResult-Objekt zurück, auf dem geprüft werden kann, wann die Methode vollständig abgearbeitet wurde und das Resultat via End-Methode abgeholt werden kann.

4 Logische Architektur

4.1 Übersicht (Layer)

Wir verwenden eine Layer-Architektur mit Infrastructure-, Domain- und Presentation-Layer. Eine wie in der Composite Application Guidance beschriebene komplett modulare Architektur ist bei dieser eher kleinen Applikation nicht sinnvoll. Gewisse Ansätze davon werden jedoch verwendet.

4.1.1 View-Injection (Named-Locations)

Die *Named-Locations* vereinfachen das Aufteilen und Darstellen der Views stark. *Named-Locations* sind Platzhalter mit einem Namen im XAML und teilen das GUI in Bereiche auf. Im Code kann eine View in einer gewählten *Named-Location* dargestellt werden. Die View selbst muss somit nicht wissen, wo sie angezeigt wird.

4.1.2 Low Coupling (Event-Aggregation)

Um die Abhängigkeiten zu minimieren wird im Presentation-Layer die *Dependency-Injection* sowie *Event-Aggregation* verwendet.

4.1.3 Dependency-Injection (Constructor-Injection)

Damit möglichst viel mit Unittests getestet werden kann, verwenden wir nur beim Logger das *Singleton-Pattern*. Alle restlichen Objekte, welche global Verfügbar sein müssen, werden jeweils im Konstruktor übergeben. Constructor-Injection ist eine einfache Möglichkeit um das Singleton-Pattern elegant zu umgehen und wird im Presentation-Layer eingesetzt. Um im Domain- und Infrastructure-Layer keine Abhängigkeit zur CAL zu generieren, wird dort weder Constructor -Injection noch Event-Aggregation verwendet.

4.1.4 Separation of concerns

Damit nun die Presentation-Layer-Klassen nicht wissen müssen, wie man z.B. einen Service erstellen muss, wird das Erstellen der Services in eine ServiceFactory ausgelagert. So muss nur die ServiceFactory wissen, wie man einen Service instanziert.

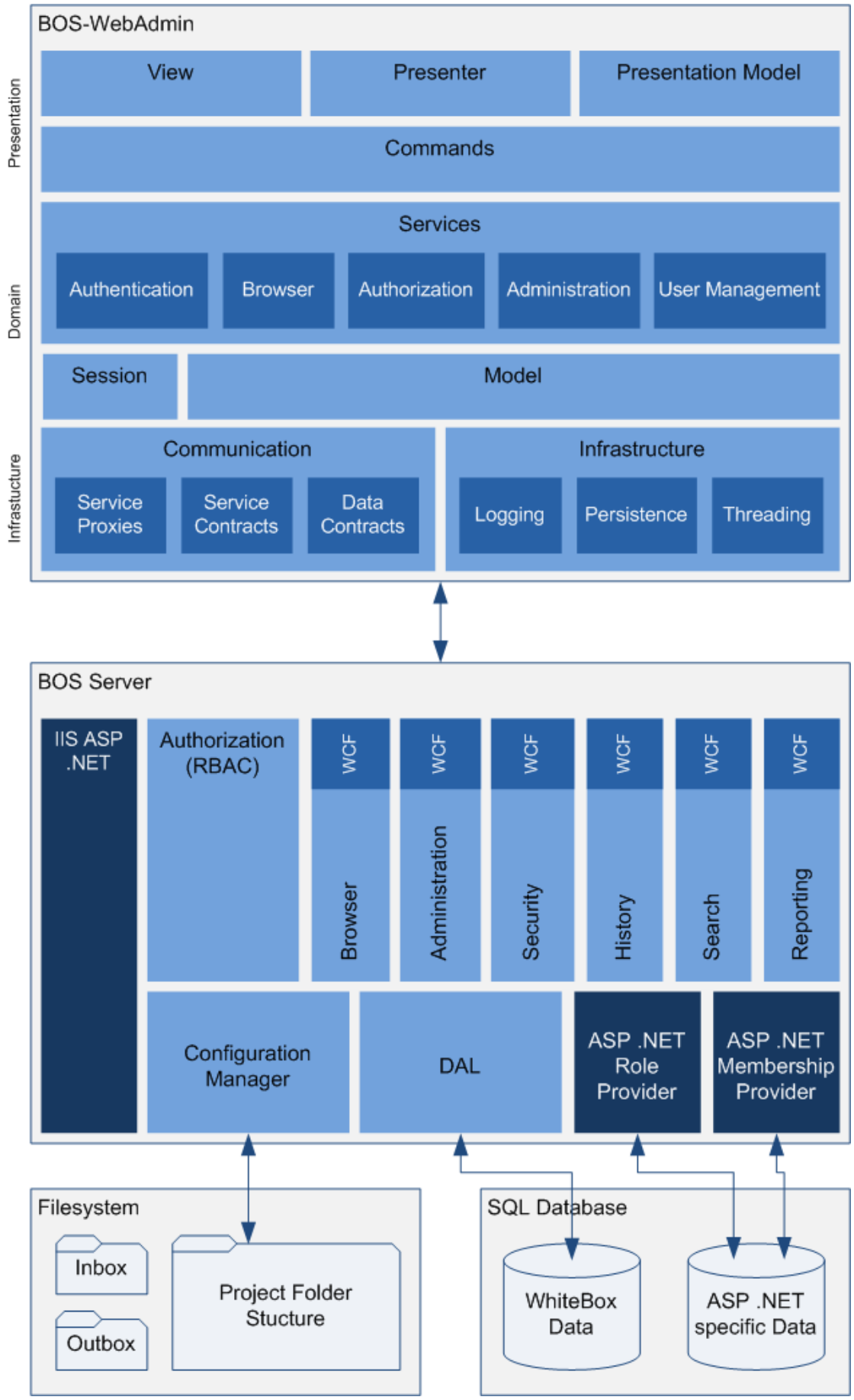


Abbildung 2: Übersicht (Layer)

4.2 Namespaces und Ordnerstruktur

Die Namespaces entsprechen auch genau der Ordnerstruktur. Der Root-Namespace *Siemens.DesigoToolset.XwpProjectManager* ist bereits gegeben. Dazu kommt für den BOS Web Client *WebClient* und danach der entsprechende Layer Namen. Weiter kommen dann die Namen, der zur Strukturierung verwendeten Ordner.

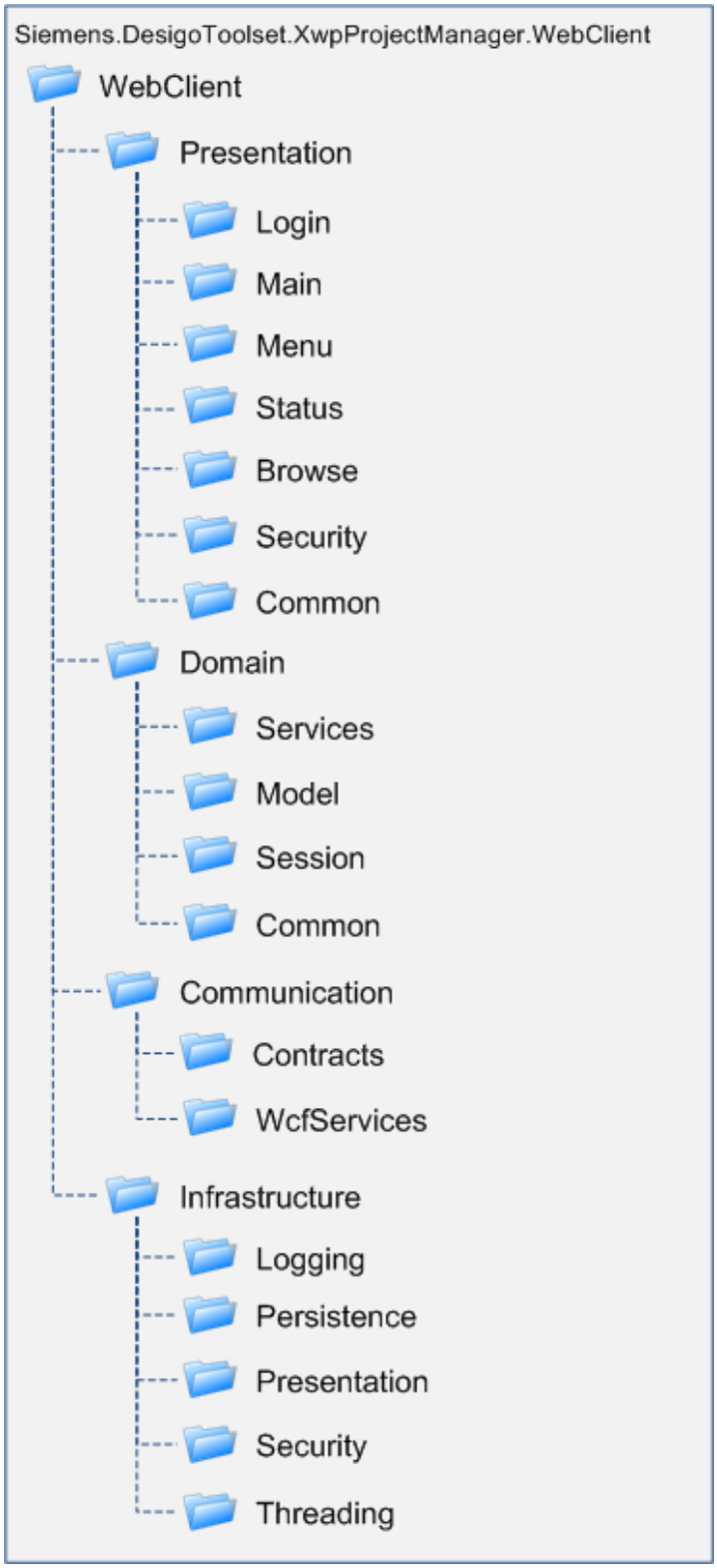


Abbildung 3: Ordnerstruktur, Namespaces

4.3 Aufrufe: View - Command - Service

Dieses Diagramm zeigt rein schematisch eine Übersicht der Views, Commands und Services. Grundsätzlich gibt es für jeden UseCase einen Command welcher die Service-Funktionen aufruft. Die hier gezeigten Commands entsprechen nicht genau den Commands im Code. Aus Zeitmangel wurden Search, History und Reporting nicht implementiert.

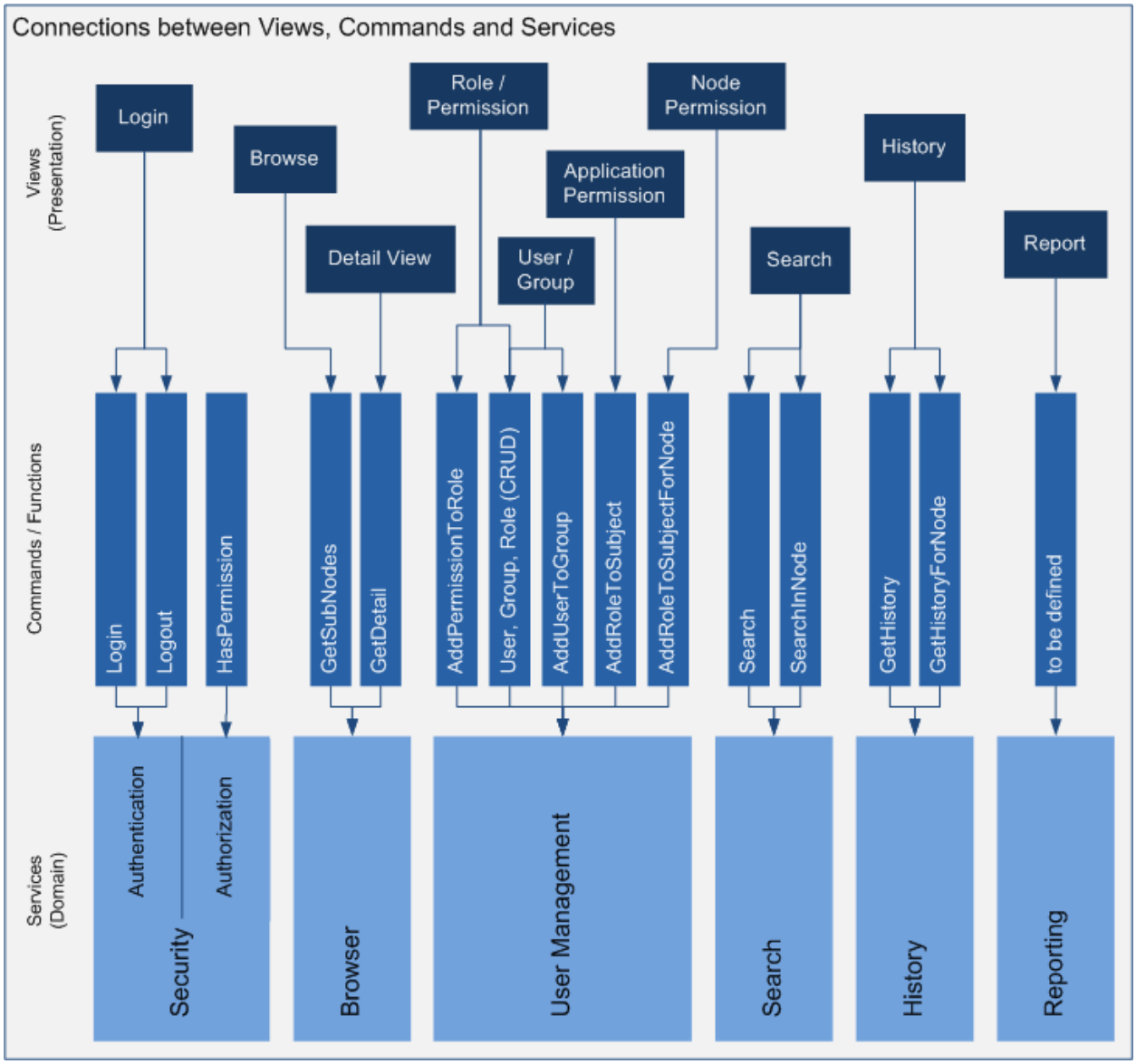
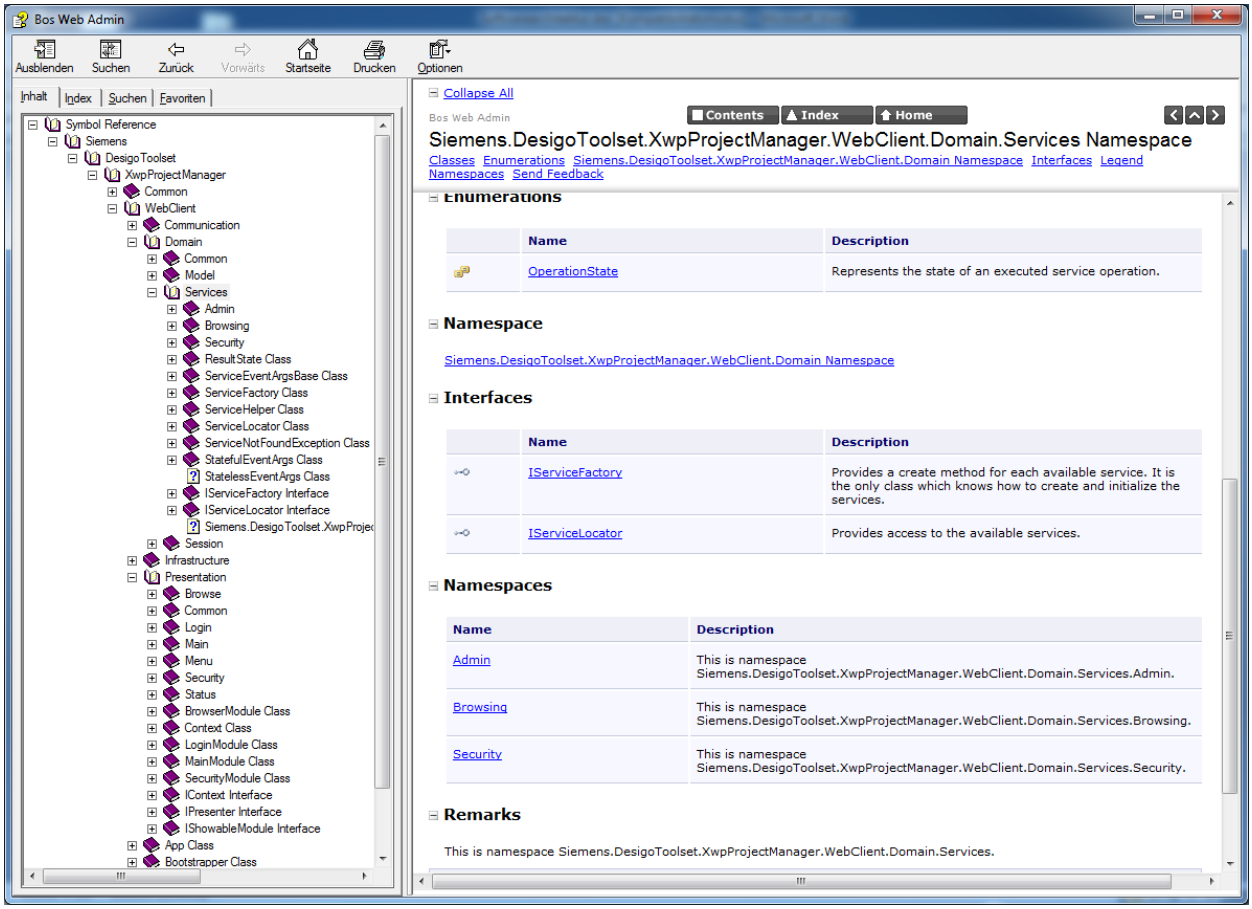


Abbildung 4: View - Command - Service

4.4 Interface Beschreibung

Aus den XML-Kommentaren generierte Interface-Dokumentation.



4.5 Klassenübersicht

Dieses Diagramm soll die Architektur möglichst übersichtlich darstellen und ist deshalb nicht vollständig. Es zeigt anhand des Login-UseCases die Klassen und deren Beziehungen über alle Layer hinweg. Alle weiteren UseCases sind vom Prinzip her gleich. Es gibt einfach entsprechend weitere Module, Presenter, Views, Commands und Services.

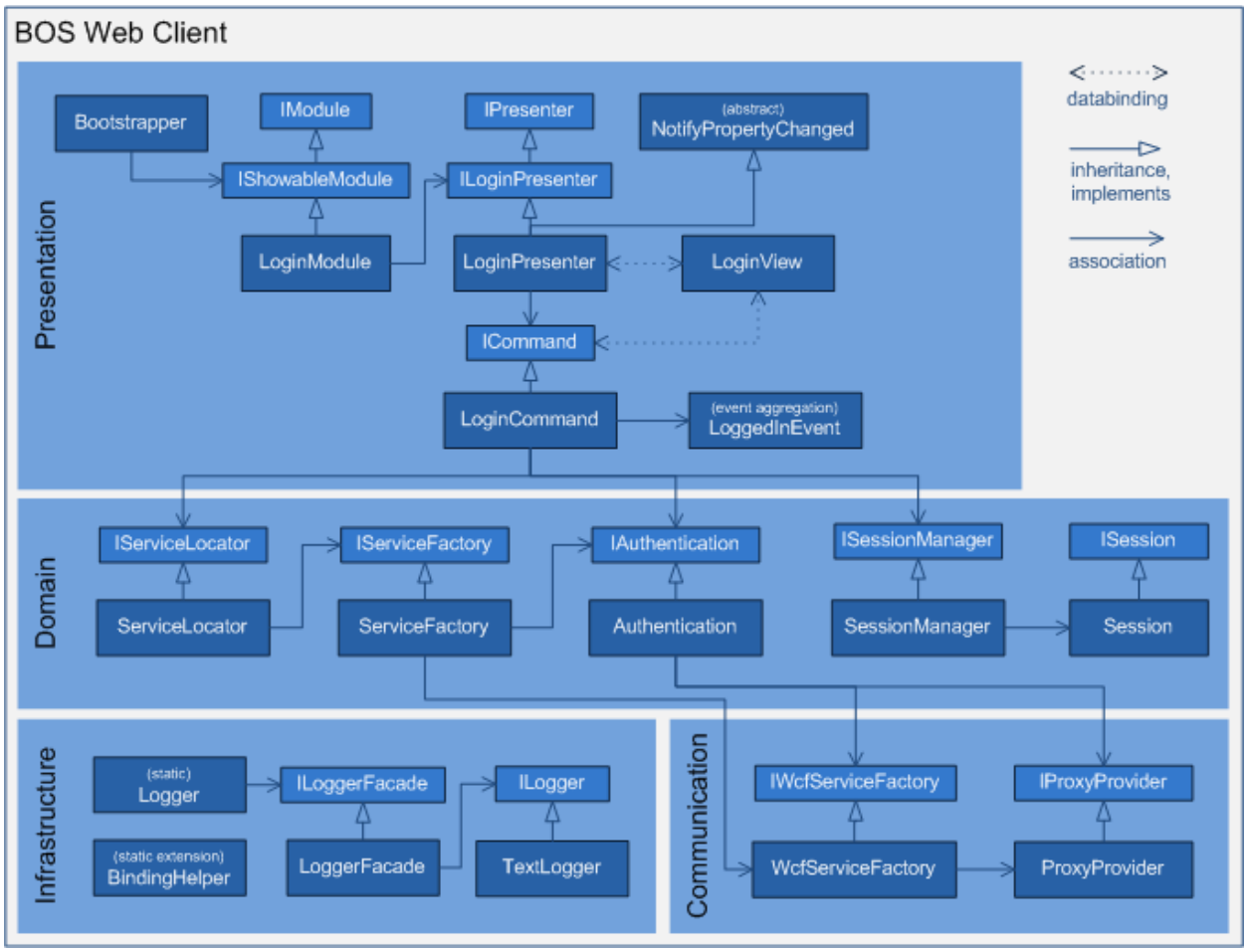


Abbildung 5: Klassenübersicht

Der Bootstrapper initialisiert die Module, welche wiederum die benötigten Presenter und Views initialisieren. Das Login-Modul initialisiert zusätzlich noch den *ServiceLocator*. Die Views bekommen ihren Presenter im Konstruktor und setze ihn gleich als *DataContext*. Presenter und View haben keine weiteren Beziehungen, sie sind nur durch Databinding verbunden. Die View Elemente sind gebunden an Properties im Presenter. Der Login-Button ist beispielsweise an den *LoginCommand* gebunden, welcher vom Presenter initialisiert wurde. Der *LoginCommand* bekommt über den *ServiceLocator* eine Instanz des *Authentication* Services und ruft darauf die benötigten Funktionen mit der Übergabe einer Callback-Funktion auf. Der Service führt dann die WCF-Funktionen asynchron aus und gibt danach das Ergebnis über die Callback-Funktion an den *LoginCommand* zurück. Im Beispiel des *LoginCommands* wird in der Callback-Funktion mit Event-Aggregation der *LoggedInEvent* gefeuert und alle die sich darauf registriert haben, werden notifiziert. Nach erfolgreichem Login werden eingeloggter User sowie der verbundene Server in einer Session gespeichert, die zu jeder Zeit über den *SessionManager* abgerufen werden kann.

4.6 Global Verfügbare Objekte

Objekte wie der *SessionManager* oder *ServiceLocator* müssen von vielen Orten verfügbar sein. Damit sie jedoch mit Unit-Tests getestet werden, bzw. durch ein Mock-Objekt ersetzt werden können, dürfen keine Singletons verwendet werden. Die CAL stellt gute Klassen für die Dependency-Injection zur Verfügung, welche wir auch verwenden. Die Klassen für die Interfaces *IStatusManager* und *IServiceLocator* werden mit der Option registriert, dass nur eine Instanz existieren darf.

```
//Register and initialize service locator
this.container.RegisterType<IServiceLocator, ServiceLocator>(new
ContainerControlledLifetimeManager());
IServiceLocator serviceLocator = this.container.Resolve<IServiceLocator>();
```

Überall wo über den *IUnityContainer* von CAL ein Resolve auf den *IServiceLocator* ausgeführt wird, wird dieselbe Instanz zurückgegeben. Zusätzlich ist es auch möglich, im Konstruktor der Presenter-Klassen z.B. den *IServiceLocator* zu definieren, welcher beim initialisieren des Presenters automatisch mit der einen Instanz des *IServiceLocators* injiziert wird. Gleiches gilt für den SessionManager, der ebenfalls so übergeben wird.

```
public Presenter(IServiceLocator serviceLocator)
{
      this.serviceLocator = serviceLocator;
}
```

4.6.1 Service Locator

Um nun z.B. den *Authentication*-Service zu benutzen, muss er zuerst über den ServiceLocator geholt werden. Dies geschieht wie folgt:

```
IAuthentication authService = serviceLocator.FindService<IAuthentication>();
```

Die `IServiceLocator.FindService<T>()` Methode gibt dabei immer dieselbe Serviceinstanz zurück, die intern im *ServiceLocator* zwischengespeichert wird. Beim ersten Aufruf von FindService<T>() wird eine neue Instanz des gesuchten Services initialisiert.

4.6.2 Logger

Nur der Logger ist trozdem ein Singleton, weil sonst jede Klasse die loggen will, zum Beispiel über den Konstruktor den *IUnityConainer* oder den Logger direkt bekommen muss. Geloggt wird bei einem Debug-Build in das Tab LogView und bei einem Release-Build in eine Datei.

4.6.3 Decoupling durch Event-Aggregation

Damit keine unnötigen Abhängigkeiten zwischen den Modulen entstehen, wird z.B. das Einloggen oder Ausloggen mit Event-Aggregation von CAL den anderen Modulen und Presentern mitgeteilt. Bei der Event-Aggregation müssen alle Beteiligten nur denselben Event, abgeleitet von *CompositePresentationEvent* und den *IEventAggregator* kennen. Eine weitere Abhängigkeit besteht nicht.

4.7 Presentation Layer

4.7.1 Bootstrapper

4.7.1.1 Beschreibung

Der *Bootstrapper* initialisiert zu Beginn das *LoginModule* und das *MainModule*. Das *LoginModule* erstellt Instanzen vom *LoginPresenter* und *StatusPresenter*. Das *MainModule* initialisiert weiter den

MainPresenter und den *MenuPresenter*, sowie das *SecurityModule* und *BrowserModule*, welches wiederum den *BrowserPresenter* und *DetailPresenter* initialisiert. Alle Module haben eine Initialize- und eine Show-Methode. Die Initialize-Methode wird vom Bootstrapper aufgerufen und initialisiert jeweils die benötigten Klassen. Die Show-Methode zeigt das Modul an, indem es in den Named-Locations die korrekten Views hinzufügt. Jeder Presenter verfügt auch über eine Initialize-Methode, welche den Presenter auf den Initialzustand zurück setzt. Die Initialize-Methoden werden vom Modul in der Show-Methode aufgerufen. Der *Bootstrapper* ruft nach dem initialisieren der Module die Show-Methode des *LoginModule* auf, damit es angezeigt wird.

4.7.1.2 Diagramme

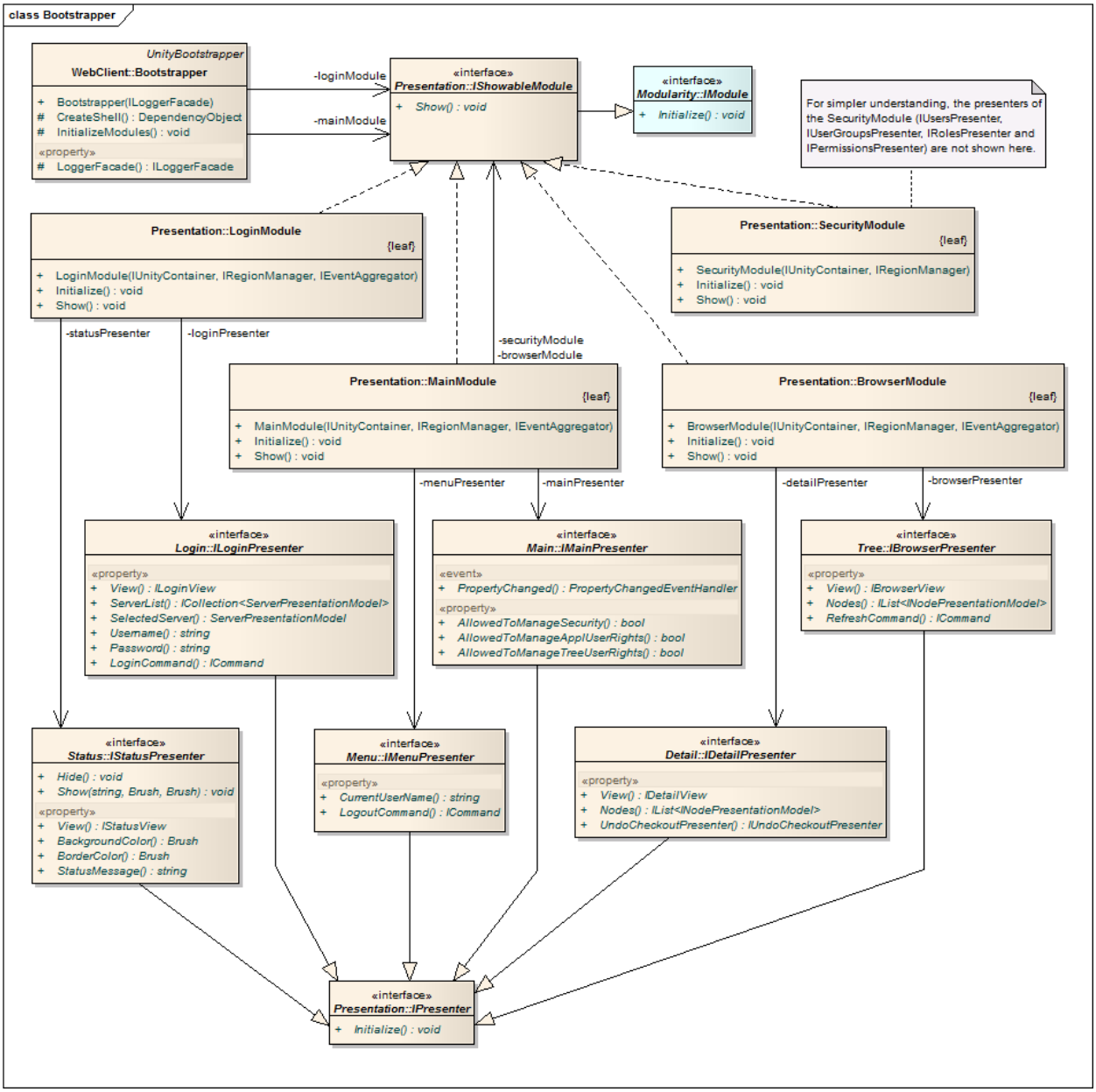


Abbildung 6: Bootstrapper und Module

4.7.2 Login Modul

4.7.2.1 Beschreibung

In der Initialize-Methode registriert sich das *LoginModul* für den *LoggedOutEvent* und das *MainModule* für den *LoggedInEvent*. Die Event-Handler rufen jeweils die Show-Methode auf um das Modul anzuzeigen.

Das *LoginModul* wird als erstes angezeigt und fügt die *LoginView* der Named-Location mit dem Namen MainRegion hinzu. Die *LoginView* ist durch Databinding mit den *LoginPresenter* verbunden, welcher den *LoginCommand* initialisiert und zur Verfügung stellt. Der *LoginCommand* holt in der Execute-Methode über den *ServiceLocator* den Authentication-Service. Auf dem Authentication-Service wird die ValidateUser-Methode aufgerufen. In der übergebenen Callback-Funktion wird dann der LoggedInEvent (Event-Aggregation) gefeuert und dadurch das MainModule notifiziert. Das *MainModule* ruft, wenn die Authentifizierung erfolgreich war, die eigene Show-Methode auf. Darin wird die *MainView* der Named-Location MainRegion hinzugefügt und die Initialize-Methoden der Presenter aufgerufen.

4.7.2.2 Diagramme

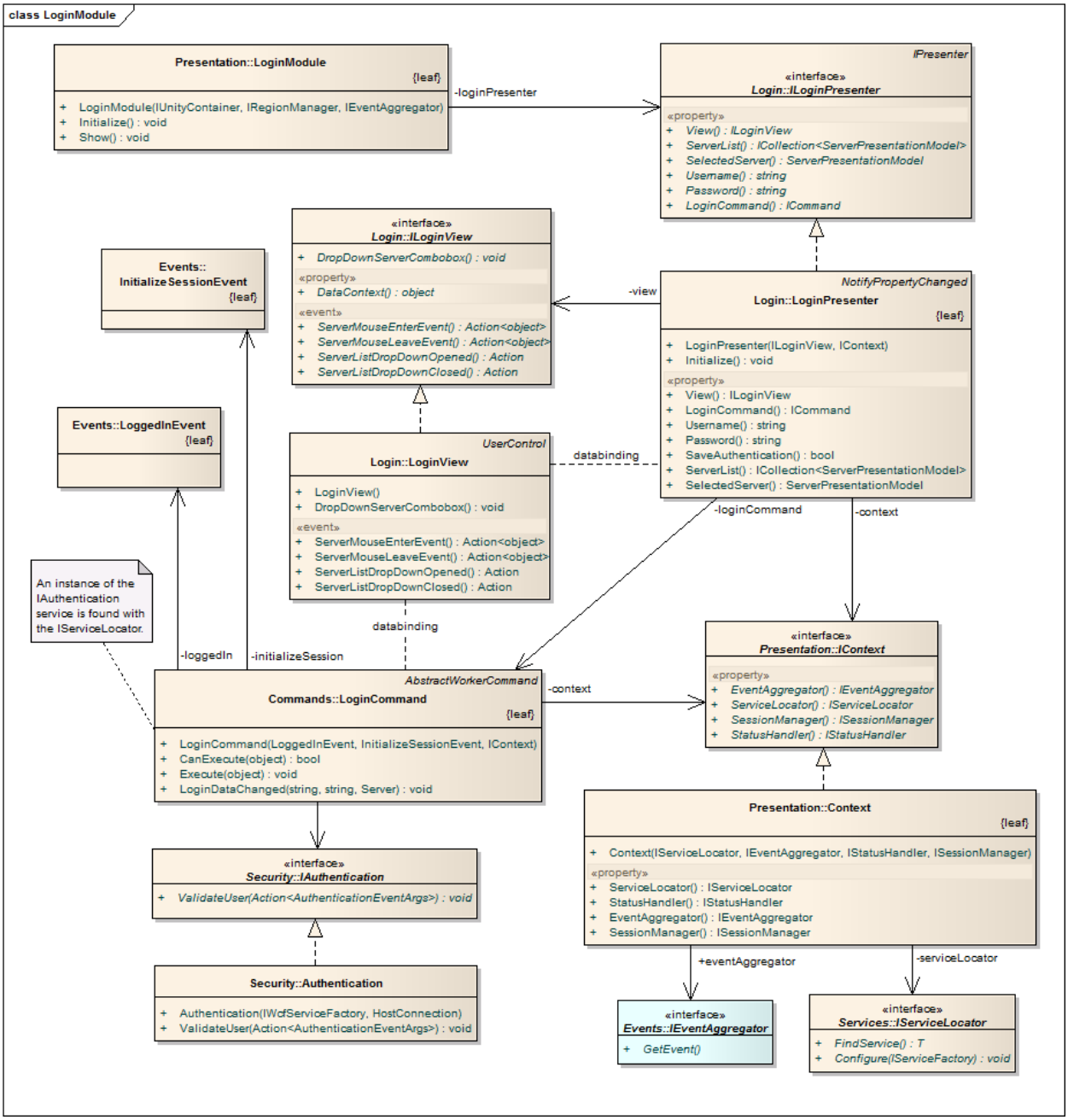


Abbildung 7: Login Modul

4.7.3 Browser Modul

4.7.3.1 Beschreibung

Das *BrowserModule* gehört zum *MainModule*, weil es auch vom User-Interface her ein Teil davon ist. Das *MainModule* ruft in der eigenen Initialize-Methode die Initialize-Methode des *BrowserModule* auf, sowie in der eigenen Show-Methode, die Show-Methode des *BrowserModule*. Das *BrowserModule* selbst muss dadurch nicht auf den *LoggedInEvent* hören. Das *BrowserModule* erstellt einen *BrowsePresenter*, welcher die Daten zur Verfügung stellt um den Projektbaum anzuzeigen. Der *BrowserPresenter* muss sich bei der *BrowserView* für den *SelectedNodeChanged* Event registrieren und eine Methode aufrufen um den Root-Node aufzuklappen. Damit er nicht direkt von der *BrowserView* abhängig ist, muss die *BrowserView* das *IBrowserView* Interface implementieren und *BrowserPresenter* bekommte eine Instanz dieses Interfaces im Konstruktor. Der *BrowserPresenter* hat eine Observable-Collection von *INodePresentationModel* und jedes *INodePresentationModel* hat wiederum eine Observable-Collection von *INodePresentationModel*. Dadurch kann mit Databinding der Projektbaum angezeigt werden. Für den Browser relevant sind drei Browser-Node-Klassen welche das Interface *INodePresentationModel* implementieren, *StructureNodePresentationModel*, *ProjectNodePresentationModel* und *VirtualBrowserNodePresentationModel*. Das *BaseNodePresentationModel* ruft im Getter des Properties *Children* die Methode *EvaluateChildNodes* auf, welche von den Browser-Node-Klassen entsprechend implementiert wird. Im *StructureNodePresentationModel* wird dort die Methode *GetChildNodes* auf dem Browser-Service aufgerufen. In der übergebenen Callback-Methode werden die Sub-Nodes der *nodes* Liste hinzugefügt und da es eine *ObservableCollection* ist, zeigt die *BrowserView* gleich die neuen Nodes an. Das Property *Children* selbst ist vom Typ *ICollection*, denn es reicht wenn die private Variabel *nodes* dahinter vom Typ *ObservableCollection* ist.

4.7.3.2 Diagramme

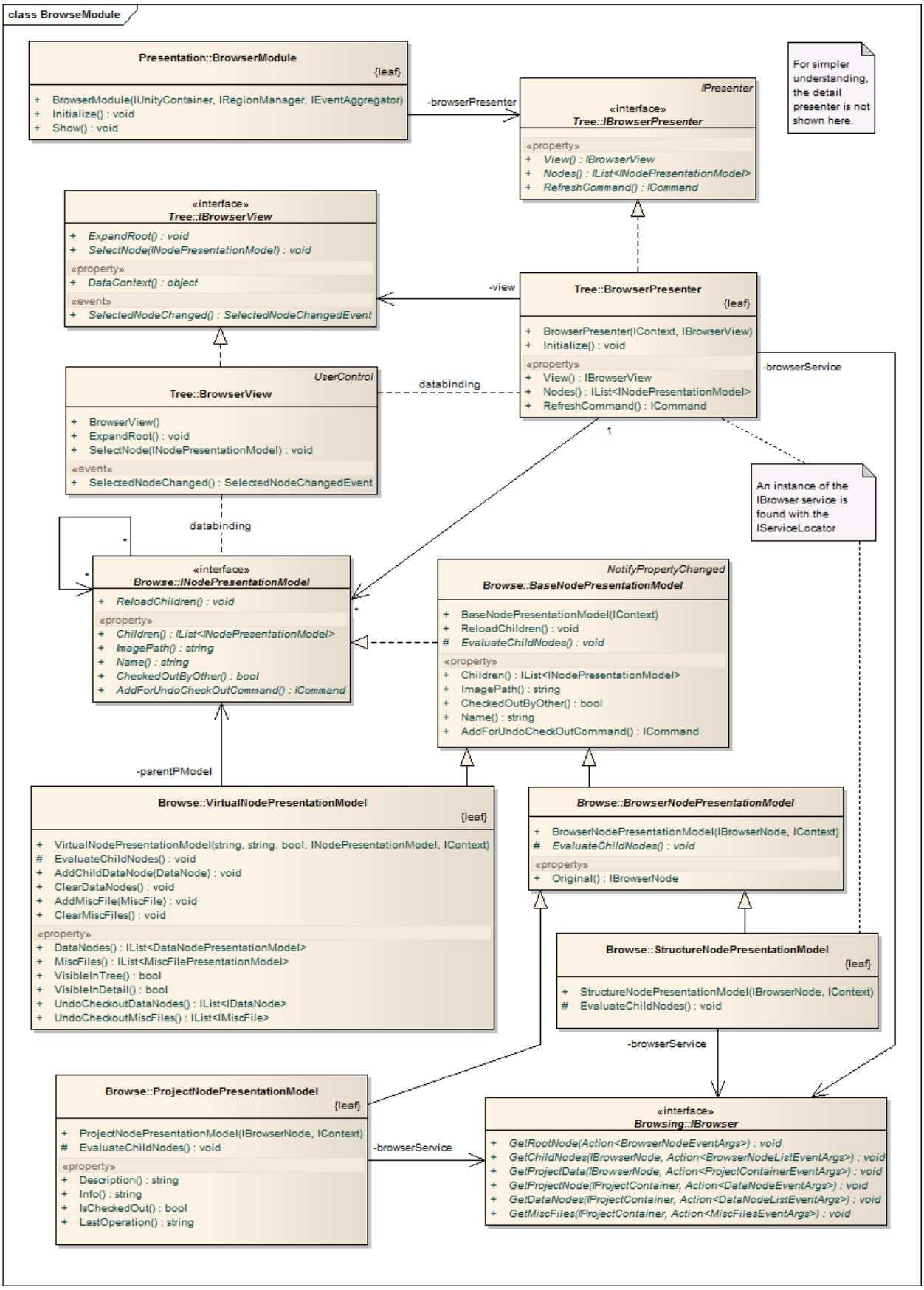


Abbildung 8: Browser Modul

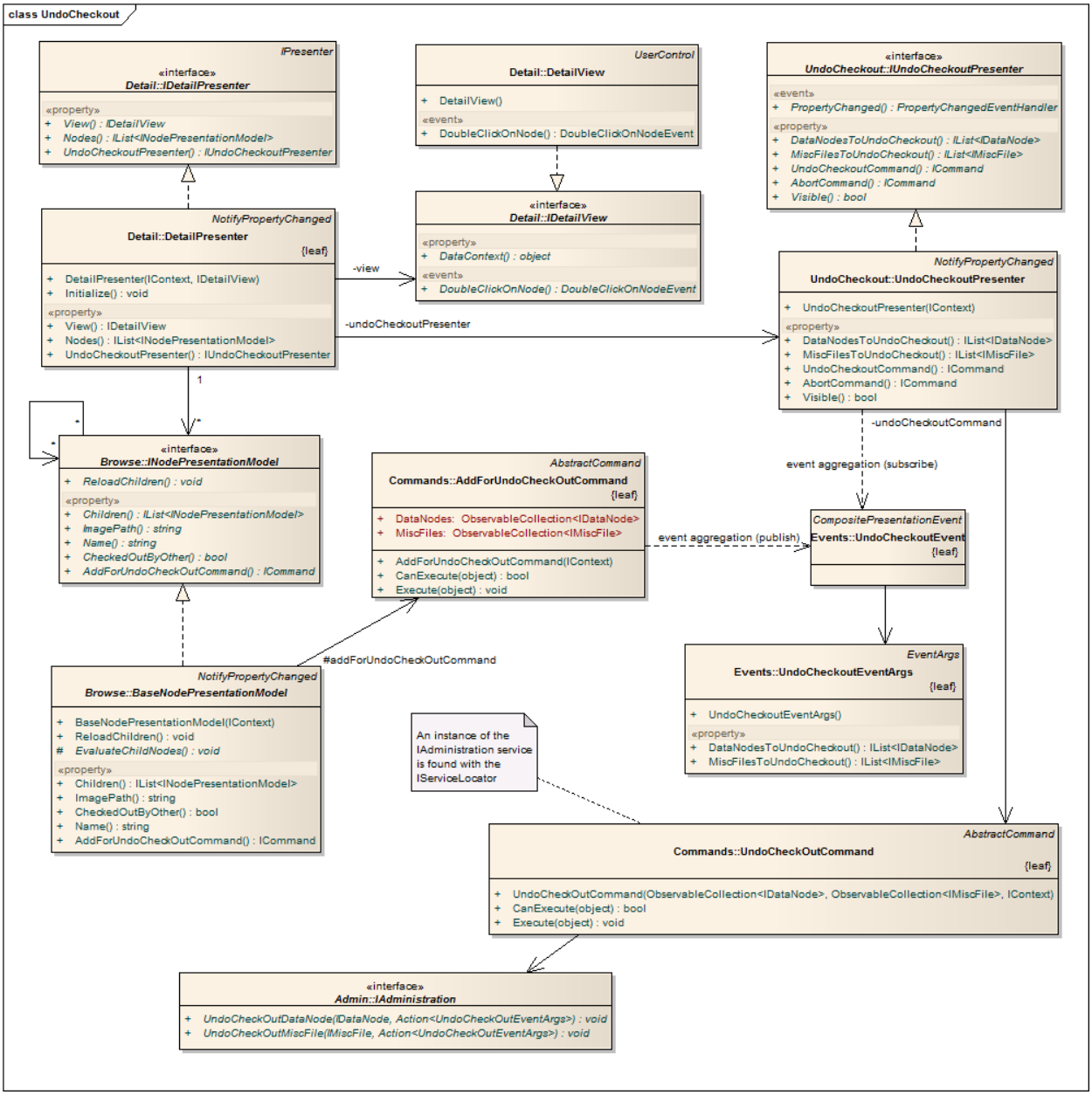


Abbildung 9 Browser - Undocheckout

4.7.4 Benutzer-Management

4.7.4.1 Beschreibung

Zeigt die einzelnen Klassen welche im Presentation-Layer für die Benutzer-Verwaltung zuständig sind. Das *IUsersPresenter* Interface stellt alle benötigten Daten und Commands für die *UserView* zur Verfügung. Die *UserView* ist durch Databinding mit dem *UserPresenter* verbunden und für die Darstellung verantwortlich. Die Klasse *TranscendentUser* wird benötigt um einen neuer Benutzer gleich behandeln zu können, wie einen der im System bereits existiert. Sie implementiert auch das *IUser* Interface, benötigt jedoch keinen *DCUser* (Data Contract).

4.7.4.2 Diagramme

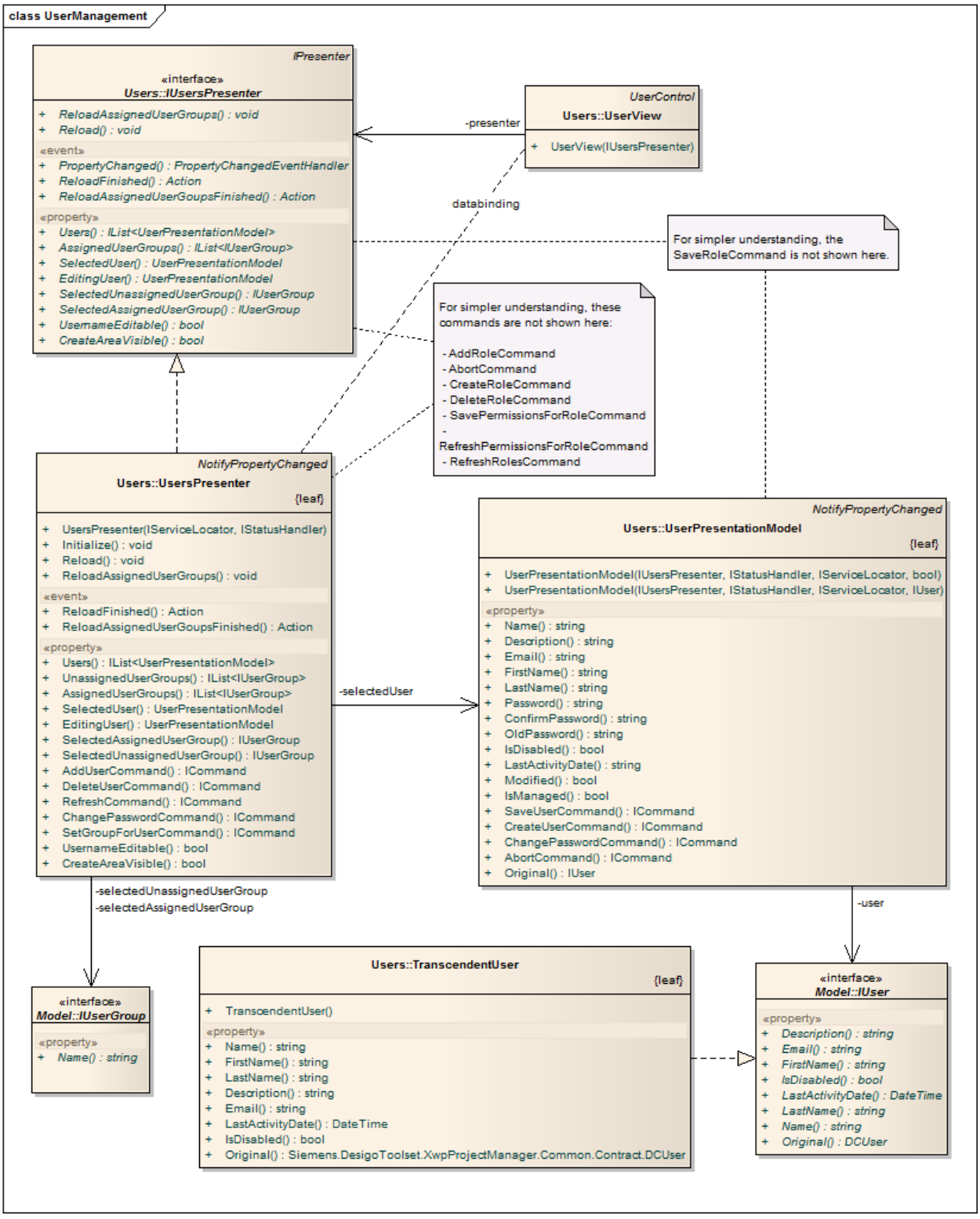


Abbildung 10 Benutzer-Management

4.7.5 Benutzergruppen-Management

4.7.5.1 Beschreibung

Zeigt die einzelnen Klassen welche im Presentation-Layer für die Benutzergruppen-Verwaltung zuständig sind. Das *IUserGroupsPresenter* Interface stellt alle benötigten Daten und Commands für die

UserGroupsView zur Verfügung. Die *UserGroupsView* ist durch Databinding mit dem *UserGroupsPresenter* verbunden und für die Darstellung verantwortlich.

4.7.5.2 Diagramme

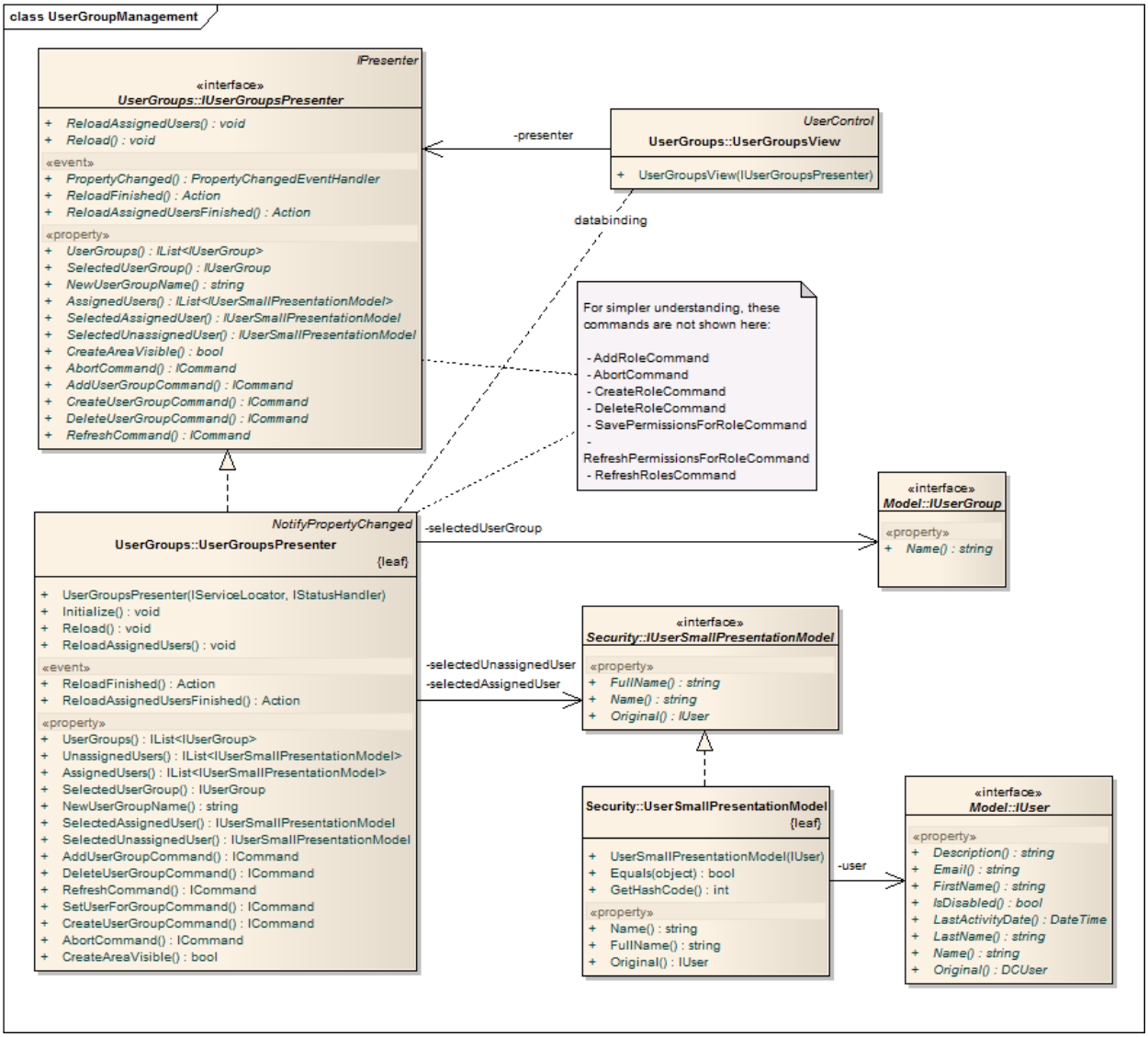


Abbildung 11 Benutzergruppen-Management

4.7.6 Rollen-Management

4.7.6.1 Beschreibung

Zeigt die einzelnen Klassen welche im Presentation-Layer für die Rollen-Verwaltung zuständig sind. Das *IRolesPresenter* Interface stellt alle benötigten Daten und Commands für die *RolesView* zur Verfügung. Die *RolesView* ist durch Databinding mit dem *RolesPresenter* verbunden und für die Darstellung verantwortlich.

4.7.6.2 Diagramme

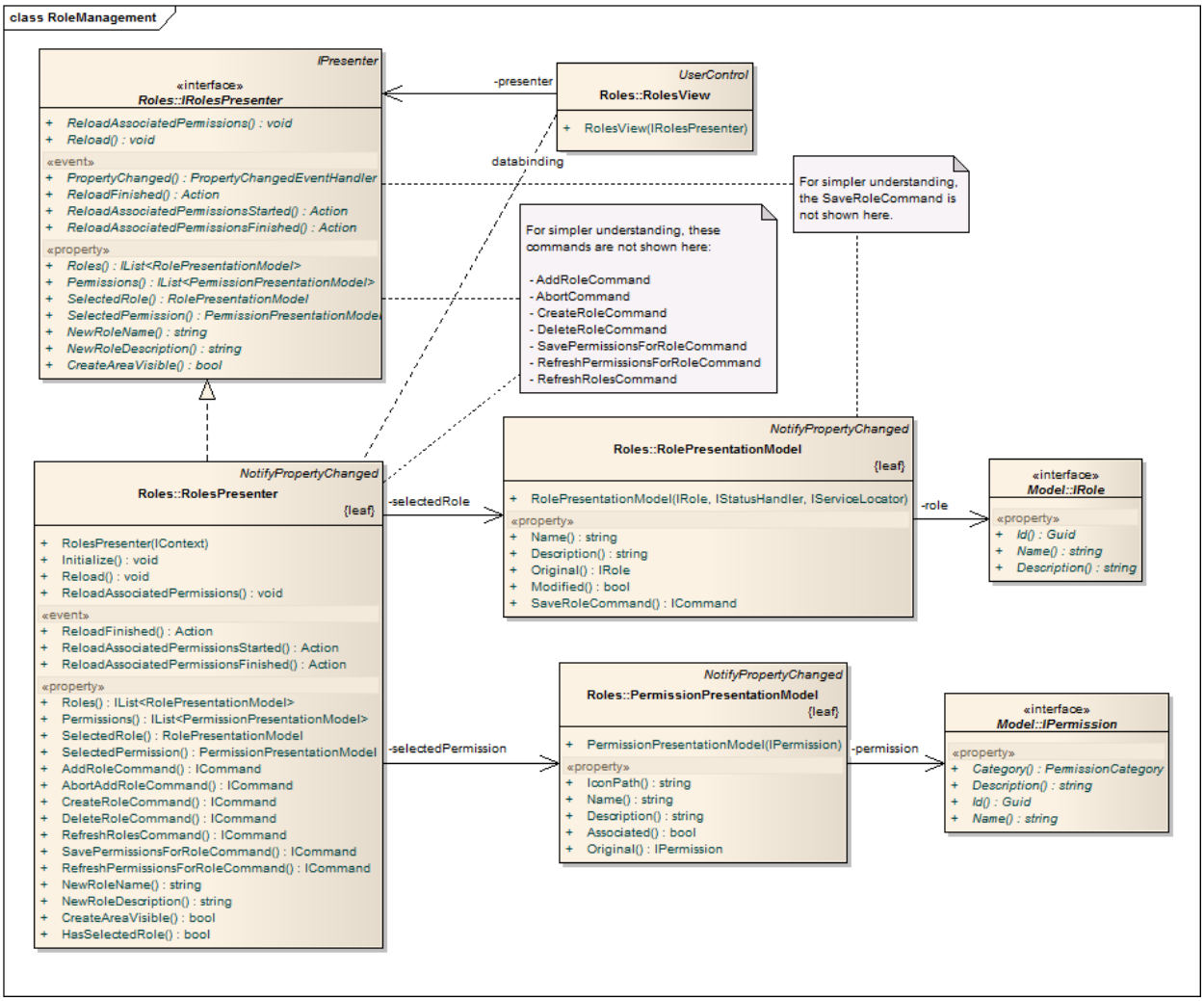


Abbildung 12 Rollen-Management

4.7.7 Berechtigungen-Management

4.7.7.1 Beschreibung

Zeigt die einzelnen Klassen welche im Presentation-Layer für die Berechtigungen-Verwaltung zuständig sind. Das *IPermissionsPresenter* Interface stellt alle benötigten Daten und Commands für die *PermissionsView* zur Verfügung. Die *PermissionsView* ist durch Databinding mit dem *PermissionsPresenter* verbunden und für die Darstellung verantwortlich. Der *ApplicationPermissionsPresenter* ist zuständig für das Setzen von Applikationsweiten Berechtigungen und der *NodePermissionsPresenter* für die Projekt/Ordnerspezifischen Berechtigungen.

4.7.7.2 Diagramme

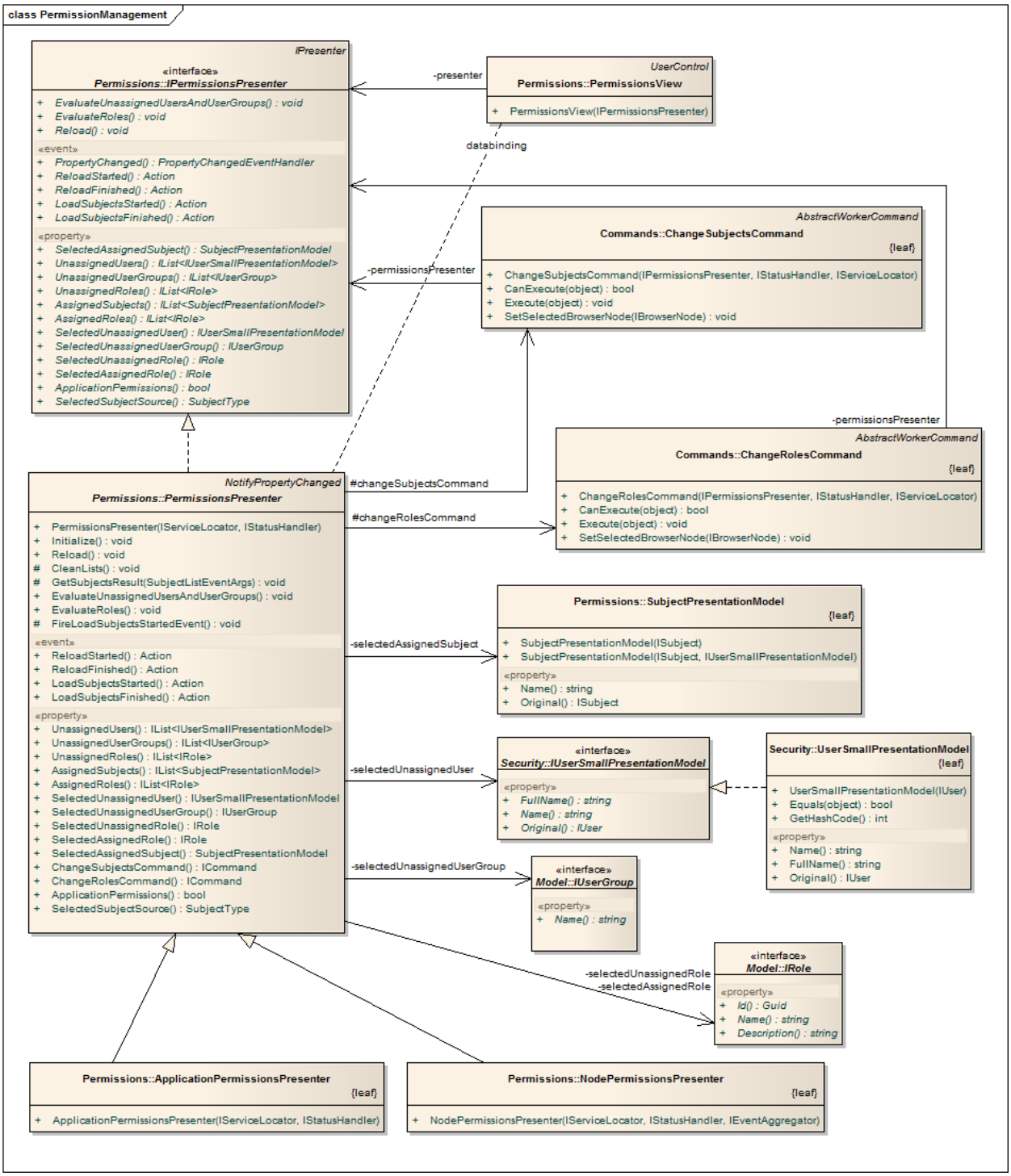


Abbildung 13 Berechtigungen-Management

4.8 Übersicht Application Core

4.8.1 Domain, Communication, Infrastructure Packages

4.8.1.1 Beschreibung

Dies ist eine Übersicht über die Hauptklassen, wie z.B. den *ServiceLocator*, den *SessionManager* sowie die *ServiceFactory* und die *WcfServiceFactory*.

- Der ***ServiceLocator*** stellt eine *FindService*-Methode zur Verfügung, mit Hilfe derer ein spezifischer Service gefunden werden kann. Dabei ist der *ServiceLocator* auch gleich für die Initialisierung der einzelnen Services verantwortlich, wobei für die eigentliche Instanzierung eine ServiceFactory benötigt wird, die dem ServiceLocator via *Configure*-Methode mitgegeben wird.
- Die ***ServiceFactory*** enthält das Know-How über die Instanzierung der einzelnen Services. Dafür stellt sie für jeden Service eine *Create*-Methode bereit.
- Die ***WcfServiceFactory*** stellt die Verbindung zum BOS her, validiert die Service-Contracts und enthält das Know-How für die Erstellung der verschiedenen WCF-Proxies. Für jeden WCF-Proxy stellt sie eine *Create*-Methode bereit.
- Der ***SessionManager*** speichert Werte in einem Session-Objekt, welche nur solang gültig sind, wie der Benutzer auf einem BOS-Server angemeldet ist bzw. bis er sich vom BOS-Server wieder abmeldet. Dies sind z.B. der angemeldete Benutzer sowie der verbundene Server.
- Der ***ProxyProvider*** kapselt den eigentlichen WCF-Proxy. Dies ermöglicht ein kontrolliertes Disposing des eigentlichen WCF-Proxies, da die WCF-Proxies die Eigenschaft haben, dass sie beim Dispose() unter gewissen Umständen eine Exception werfen können.

4.8.1.2 Diagramme

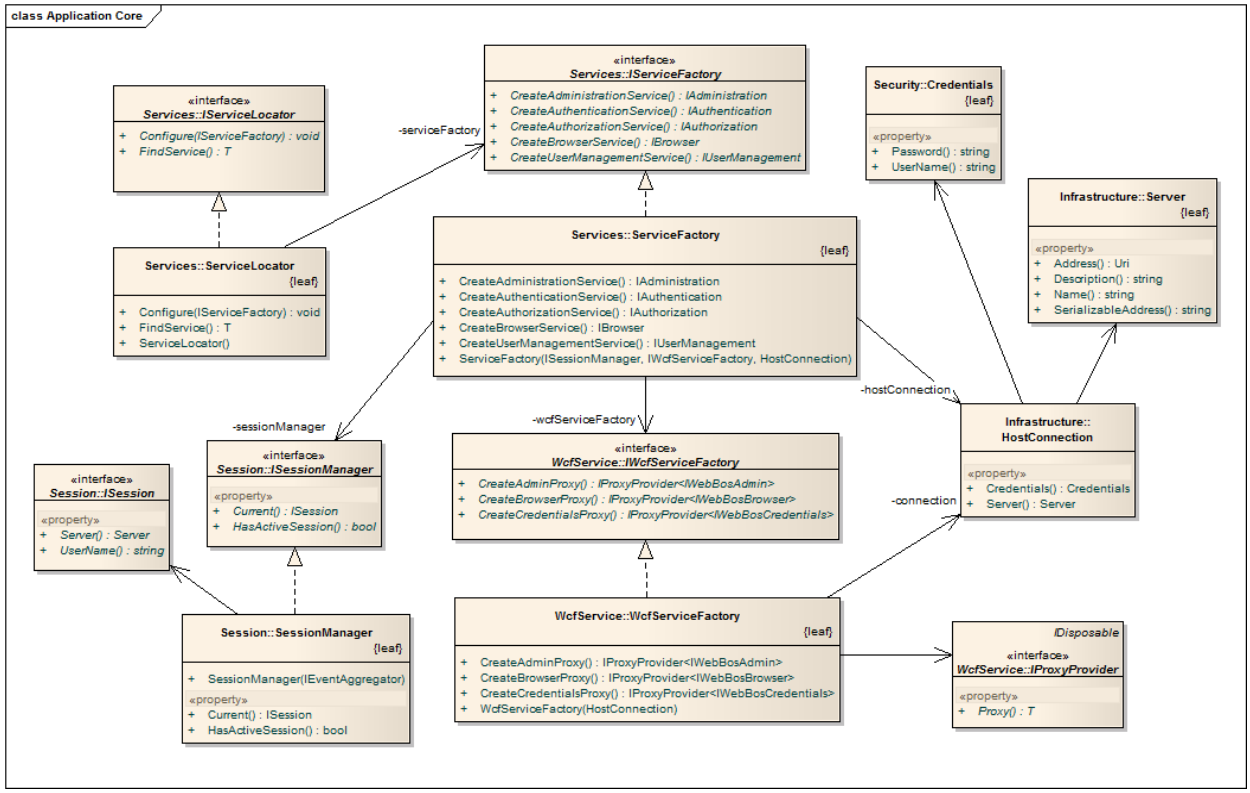


Abbildung 14: Übersicht Application Core

4.9 Domain Layer

Die verschiedenen Services gehören zu den wichtigsten Klassen im Domain-Layer. Dabei sind alle Services gleich aufgebaut. Zu jedem Service gibt es ein Interface, das über den ServiceLocator aufgelöst werden kann, um eine Instanz der Service-Implementierung zu erhalten. Es wird also immer über das jeweilige Interface auf den Service zugegriffen und nie direkt die implementierende Klasse instanziert. Alle Objekte, die den Servicemethoden übergeben bzw. als Resultat zurückgegeben werden, werden ausschliesslich über ein entsprechendes Interface angesprochen. So bleiben die Servicemethoden testbar und auch die Flexibilität der eigentlichen Implementation bleibt gewahrt.
Fast alle Servicemethoden sind asynchron, d.h. sie laufen auf einem separaten Thread ab und sobald das Resultat bereit steht, wird die übergebene Callback-Methode aufgerufen, bei der dann das Resultat übergeben wird. So wird das GUI auch bei länger dauernden Operationen nie blockiert. Normalerweise kann die Callback-Methode via Action<T>-Delegate als jeweils letzten Parameter einer Servicemethode übergeben werden.

4.9.1 Browser Service

4.9.1.1 Beschreibung

Der *Browser*-Service stellt Funktionen zur Verfügung, die für das Navigieren sowie zur Darstellung des Projektbaumes benötigt werden. Dabei ist das *IBrowser*-Interface die zentrale Schnittstelle für die Benutzung des Services. Im Diagramm der Abbildung 15 sind auch alle verwendeten *EventArgs* ersichtlich, die für die Rückgabe des Resultates via Callback-Methode benutzt werden. Im zweiten Diagramm der Abbildung 16 sind alle im *Browser*-Service verwendeten Datenklassen mit der gesamten Vererbungsstruktur ersichtlich.

- Der ***BrowserNode*** kann entweder ein Verzeichnis- oder ein Projekt-Knoten darstellen und wird für die Darstellung des Projektbaumes verwendet.
- Der ***ProjectContainer*** ist ein virtuelles Konstrukt, welches die Funktion einer Projekt-Klasse übernimmt, für welche aber eigentlich keine effektiven Daten hinterlegt sind und einfach dafür da ist, die verschiedenen *DataNode*s, welche zu einem gemeinsamen Projekt gehören, zusammenzuhalten.
- Der ***DataNode*** wird für die Darstellung der Detailansicht eines Projektes verwendet. Ein *DataNode* kann die verschiedensten Typen von Projektdaten enthalten, wie z.B. die Daten einer Automation Station (AS), einer Management Station (MS) oder auch die gesamten Panel- oder Netzwerkdaten.
- Ein ***MiscFile*** kann entweder eine Datei oder ein Verzeichnis (mit darin enthaltenen Dateien) repräsentieren.
- Der ***VirtualDataNode*** wird für *DataNode*s verwendet, die selbst wieder Children haben. Dies ist bei den Panel- und Netzwerkdaten der Fall. Die Children sind dann wieder vom Typ *DataNode*.
- Der *ControlledNode* ist die gemeinsame Basisklasse für DataNode und MiscFile und enthält vor allem Informationen zum Checkout-Status. Ein ControlledNode-Objekt ist ein Objekt, das unter der Kontrolle des BOS-Servers steht.

4.9.1.2 Diagramme

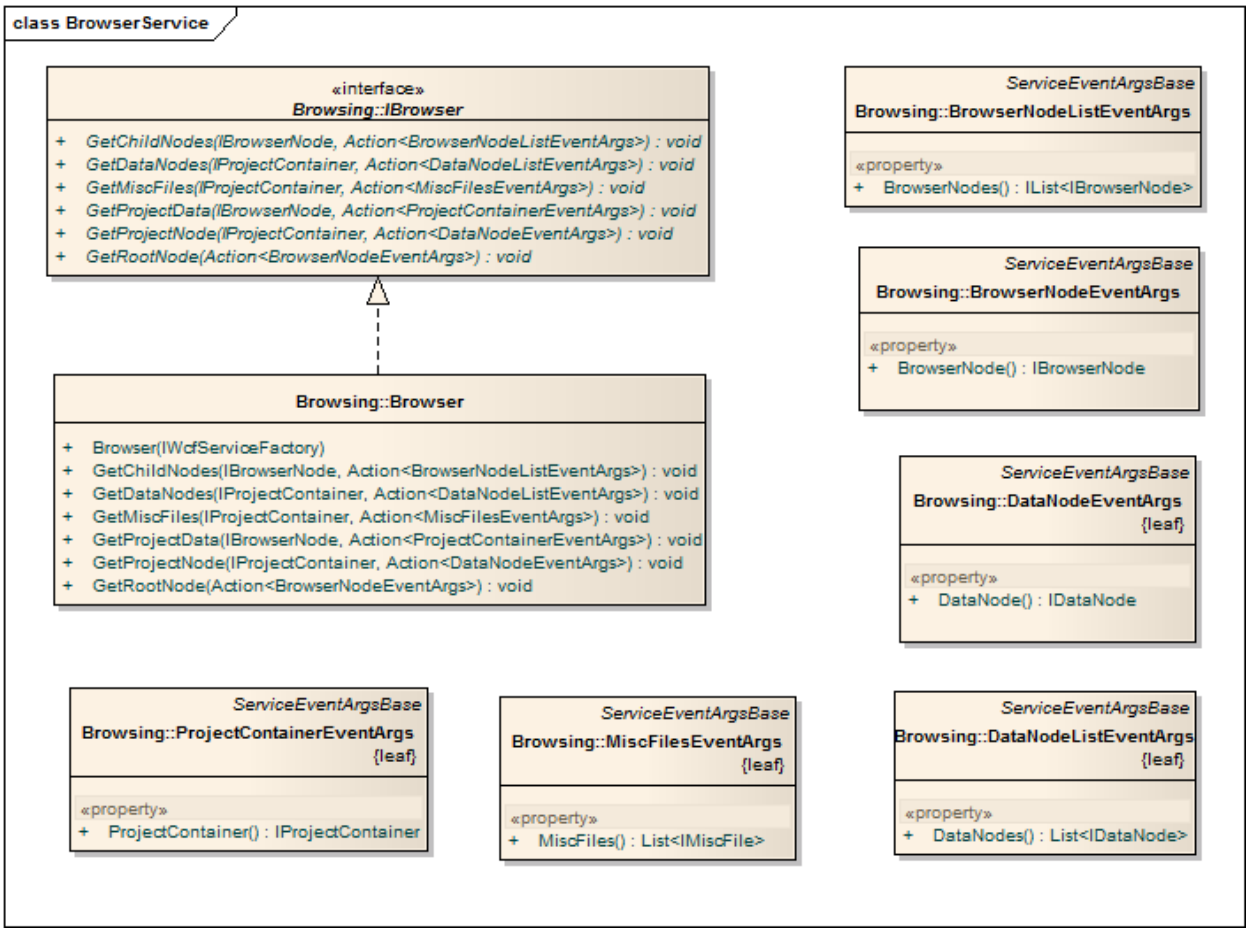


Abbildung 15: Browser Service

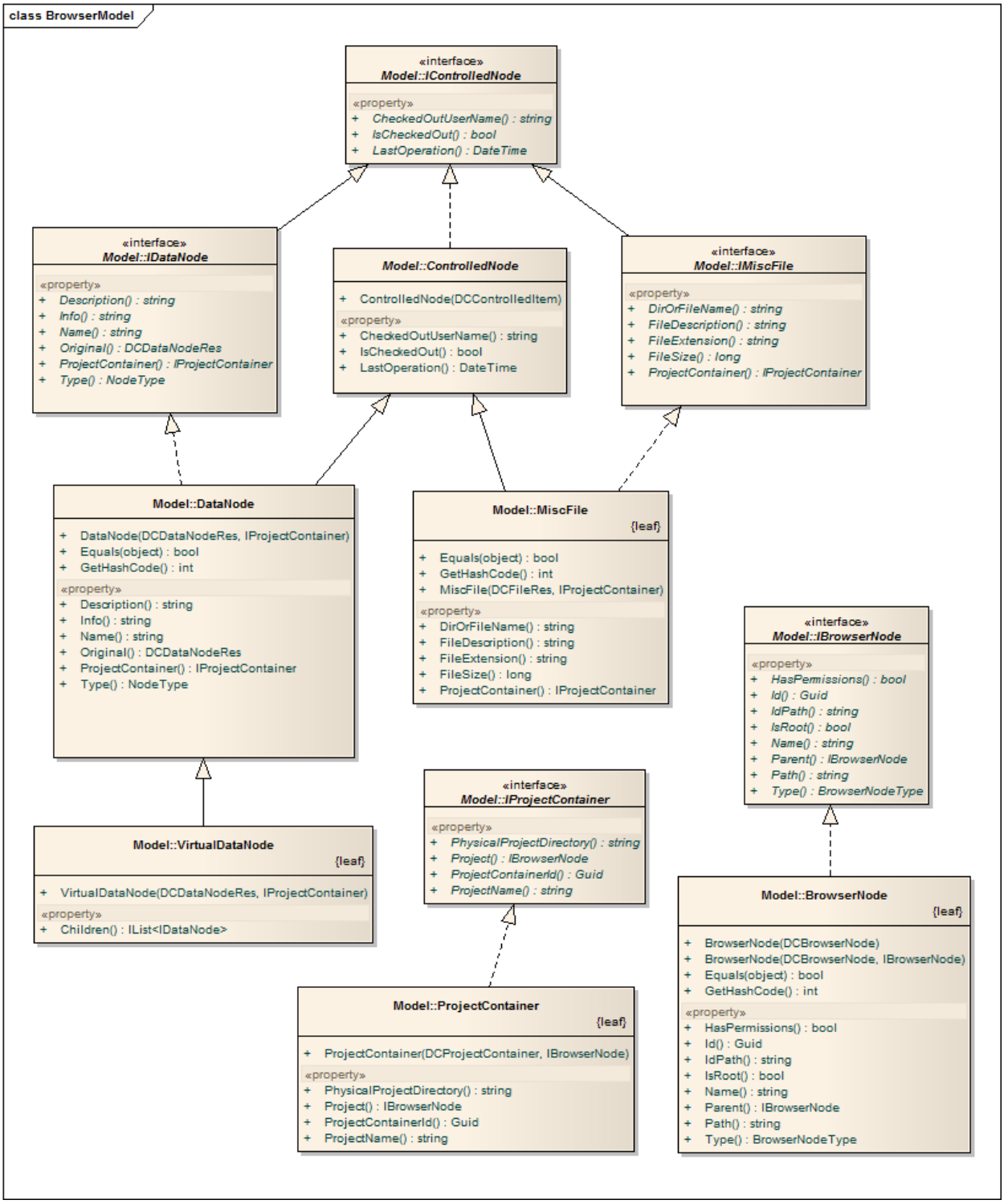


Abbildung 16: Browser Service Model

4.9.2 Administration Service

4.9.2.1 Beschreibung

Der Administration-Service stellt Funktionen zur Verfügung, die für verschiedene administrative Tätigkeiten benötigt werden. Zurzeit wird nur das „Rückgängigmachen“ von ausgecheckten DataNodes und MiscFiles unterstützt. Die UndoCheckOutEventArgs enthalten jeweils den betreffenden ControlledNode, damit der Servicebenutzer bei wiederholten Aufrufen der UndoCheckOut-Funktion erkennen kann, für welche Nodes die Funktion bereits abgearbeitet wurde.

4.9.2.2 Diagramme

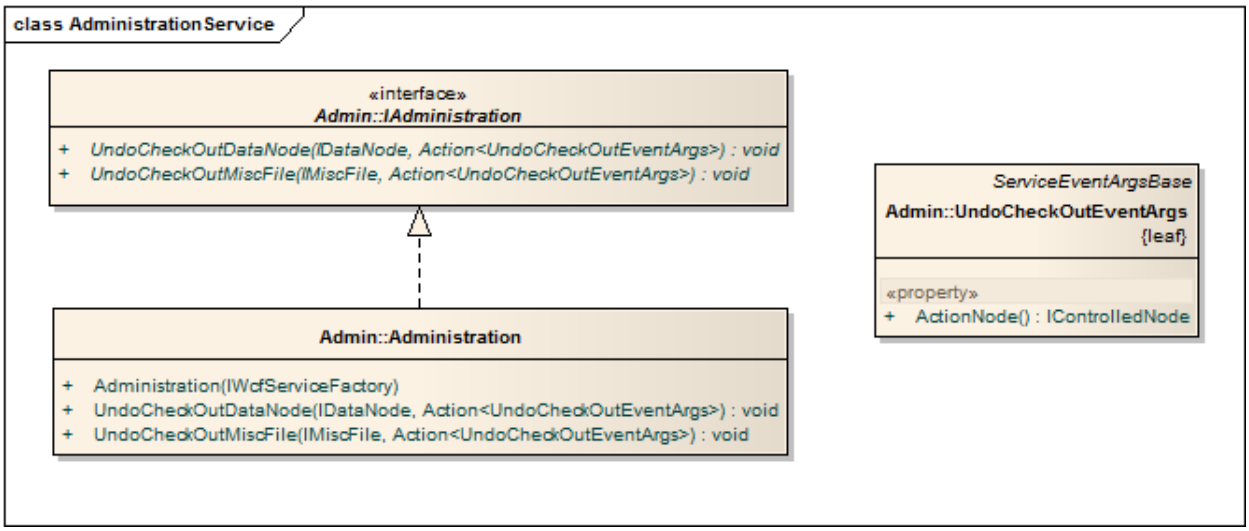


Abbildung 17: Administration Service

4.9.3 Authentication Service

4.9.3.1 Beschreibung

Der Authentication-Service stellt eine ValidateUser-Funktion zur Verfügung, mit derer überprüft werden kann, ob der eingeloggte Benutzer auf den gewählten BOS-Server zugreifen darf. Bei dieser Methode muss ein kleiner Trick angewendet werden, da es nicht möglich ist, eine Servicefunktion via WCF auf dem Server auszuführen, wenn die Credentials (Username/Passwort) auf diesem Server nicht gültig sind. Somit ist es auch nicht möglich, eine Validate-Funktion mit Parametern für Username/Passwort auf dem Server aufzurufen, um die Gültigkeit zu überprüfen, da es bei nicht Gültigkeit einen Fehler geben würde. Genau dieser Sachverhalt wird nun benutzt, um zu überprüfen, ob ein Benutzer zugriff hat, indem einfach eine GetVersion-Funktion auf dem Server aufgerufen wird, und wenn es eine Exception gibt, wird angenommen, dass der Benutzer auf dem Server keinen Zugriff hat.

4.9.3.2 Diagramme

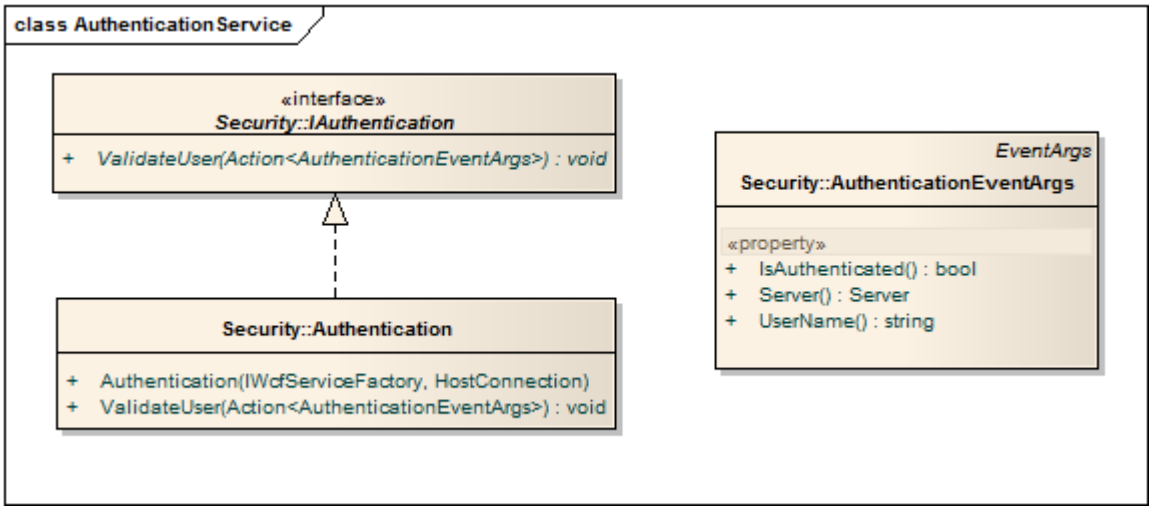


Abbildung 18: Authentication Service

4.9.4 Authorization Service

4.9.4.1 Beschreibung

Der Authorization-Service wird benutzt, um die Autorisierung für verschiedene Tätigkeiten zu überprüfen. Die Tätigkeit kann mit dem *ClaimType* angegeben werden. Der ClaimType ist eine Enumeration von allen verfügbaren Tätigkeiten/Funktionen, die vom BOS-Server unterstützt werden. Ob ein Benutzer für eine Tätigkeit berechtigt ist, wird auch nochmals auf dem Server selbst überprüft. Aber für ein Intuitives, selbsterklärendes User Interface müssen diese Berechtigungen im Voraus evaluiert werden können.

4.9.4.2 Diagramme

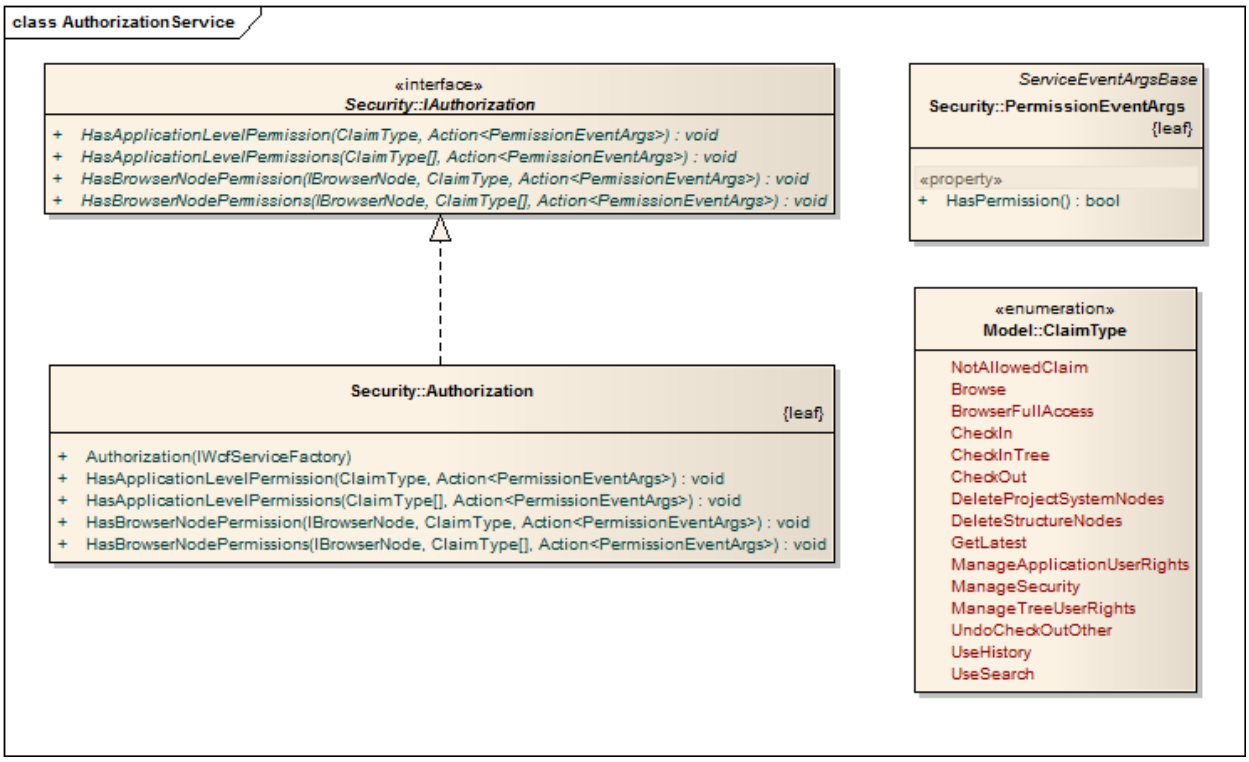


Abbildung 19: Authorization Service

4.9.5 UserManagement Service

4.9.5.1 Beschreibung

Der UserManagement-Service stellt eine grosse Auswahl an Benutzer- und Zugriffsberechtigungsverwaltungs-Operationen zur Verfügung. Wie beim Browser-Service haben alle benötigten Datenklassen ein Interface, über welches auch ausschliesslich auf diese Objekte zugegriffen wird. Im folgenden Diagramm ist der Übersichtlichkeit halber nur das Interface des UserManagement-Services aufgeführt, und nicht die implementierende UserManagement-Klasse. Auch von den Datenklassen ist ebenfalls immer nur ihr Interface aufgeführt.

4.9.5.2 Diagramme

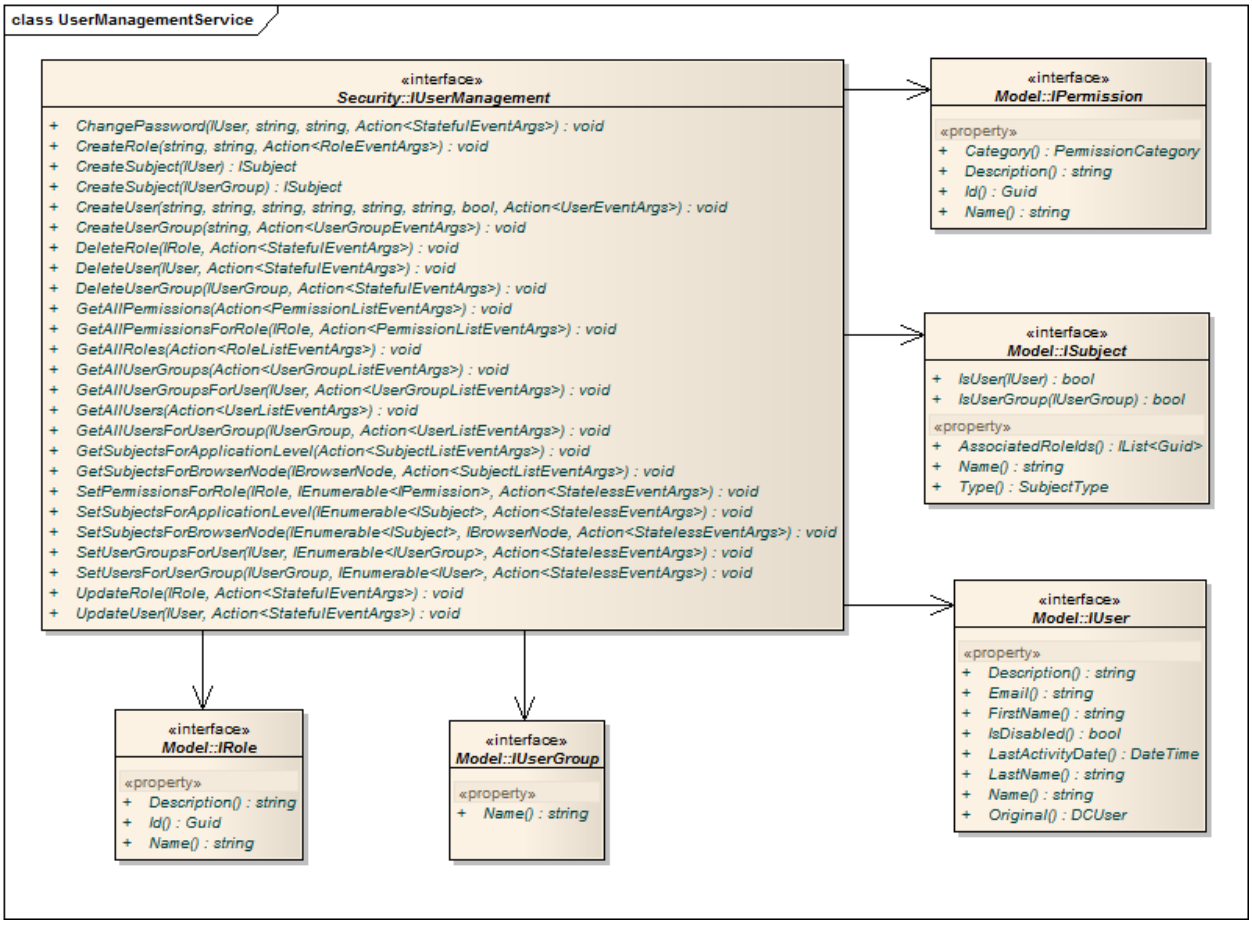


Abbildung 20: UserManagement Service

4.9.6 Ergebnisbehandlung von Serviceoperationen

4.9.6.1 Beschreibung

Die Auswertung der Resultate von asynchronen Serviceoperationen geschieht immer auf die gleiche Art und Weise. Das beim Aufruf der Callback-Methode übergebene EventArgs-Objekt, welches von der *ServiceEventArgsBase*-Klasse ableitetet, enthält neben möglichen weiteren Attributen immer ein *ResultState*-Attribut, welches wiederum ein *Success*-Attribut hat, auf welchem geprüft werden kann, ob die Servicemethode erfolgreich war, oder ob es einen Fehler gegeben hat. Im Erfolgsfall ist das *ExceptionDetails*-Attribut immer null, hingegen im Fehlerfall enthält es immer die aufgetretene Exception. Im folgenden Diagramm sind die drei dargestellten EventArgs-Klassen, die von der *ServiceEventArgsBase*-Klasse ableiten, nur als Beispiele gewählt. Es gibt noch viele weitere EventArgs-Klassen, die von *ServiceEventArgsBase* ableiten.

4.9.6.2 Diagramme

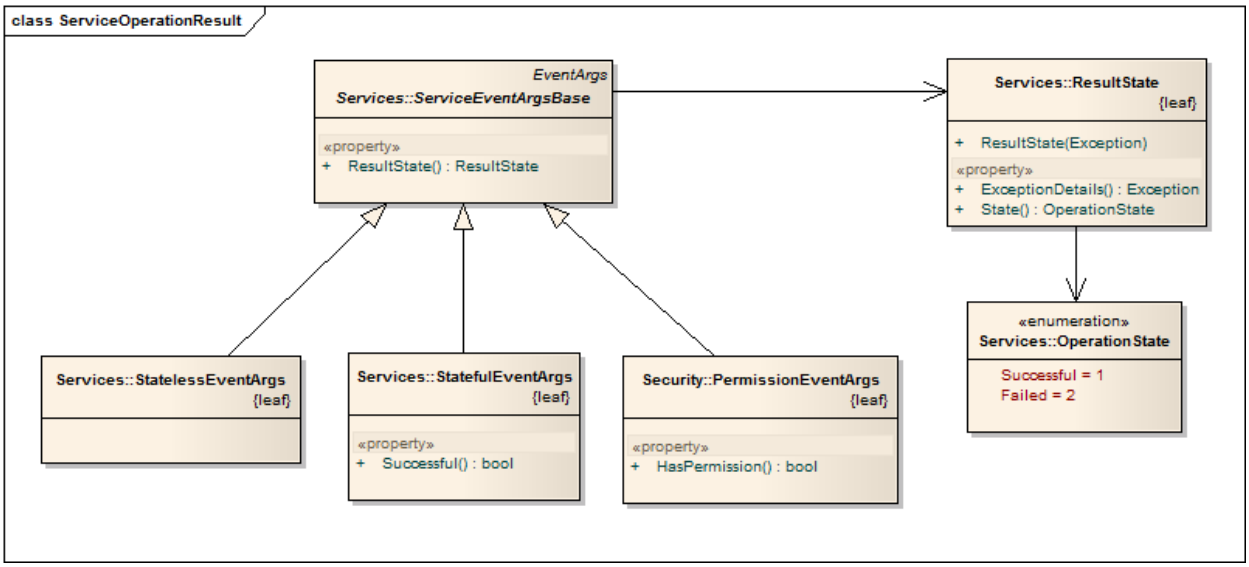


Abbildung 21: Ergebnisbehandlung von Serviceoperationen

5 Serverseitige Anpassungen

Da die Verwendung von Silverlight mit seinen Einschränkungen auch eine Anpassung der bestehenden serverseitigen Implementation verlangt, sind nachfolgend die Details dieser Änderungen beschrieben.

5.1 Hosting und WCF-Konfiguration

Die Service-Endpunkte des BOS-Servers verwendeten bisher ausschliesslich wsHttpBinding, welches aber von Silverlight 3 bekanntlich nicht unterstützt wird. Darüber hinaus wird im BOS eine eigene ServiceHostFactory verwendet, mit welcher die Erstellung der Service-Endpunkte gesteuert werden kann. Diese ServiceHostFactory wird direkt vom IIS bzw. ASP.NET-Prozess aufgerufen, sobald ein Client die erste Verbindungsanfrage an einen Service schickt. So wird im BOS diese Factory benutzt, um zwischen HTTP und HTTPS umzuschalten. HTTP wird nur im Entwicklungs- und Testumfeld benutzt, da ansonsten immer Server- und Client-Zertifikate installiert und konfiguriert werden müssten. Da Silverlight bei eingeschalteter Authentifizierung sowieso nur Verbindungen über HTTPS erlaubt, fällt dieser HTTP-Modus weg. Somit muss die ServiceHostFactory angepasst werden, um zwischen Verbindungsanfragen von normalen Clients und Silverlight-Clients zu unterscheiden, um die Endpunkte entsprechend richtig zu konfigurieren. Da die Verwendung von normalen Clients (XwpProjectManager) und Silverlight-Clients (BosWebAdmin) parallel funktionieren muss, müssen somit auch zwei verschiedene Endpunkte aufgesetzt werden. Da aber die meisten Sicherheitseinstellungen nicht für jeden Endpunkt, sondern nur pro Service gemacht werden können, und sich diese Einstellungen zwischen normalen Clients und Silverlight-Clients unterscheiden, müssen zwingend zwei Services definiert werden. Somit bleibt nichts anderes übrig, als die Service-Implementierung in eine separate Klasse auszulagern, und jeweils zwei Wrapper-Klassen zu erstellen, die je für einen Client (XwpProjectManager/BosWebAdmin) zuständig ist. In der Web.config Datei wird es so gelöst, dass je ein Service für den wsHttpBinding-Zugriff mit zwei Endpunkten für HTTP und HTTPS und ein Service für den basicHttpBinding-Zugriff mit einem Endpunkt für HTTPS konfiguriert werden. Die ServiceHostFactory entscheidet jetzt nur noch, welcher Endpunkt (HTTP oder HTTPS) für den Zugriff vom XwpProjectManager gültig ist, und ob man überhaupt mit dem BosWebAdmin auf den BOS zugreifen darf oder nicht.

5.2 Service Contracts

Da Silverlight keine WCF-Sessions (`InstanceContextMode.PerSession`) für die Instanzierung der ServiceContracts unterstützt, müssen die ServiceContracts, die bisher eine PerSession-Instanzierung hatten, für Silverlight eine PerCall-Instanzierung (`InstanceContextMode.PerCall`) anbieten.
Dies ist bei den folgenden ServiceContracts der Fall: BosBrowser, BosSearch, BosControlledItemLog (History).
Diese ServiceContracts haben bei einigen Methoden einen out-Parameter, der angibt, ob noch mehr Items vorhanden sind, die bei einem weiteren Aufruf geholt werden können. Dies ist z.B. bei der folgenden Browser-Methode der Fall:

```
DCBrowserNode[] IBosBrowser.GetChildren(DCBrowserNode parent, out bool
moreDataAvailable) { ... }
```

Hier werden pro Aufruf immer nur maximal 20 (oder je nach Einstellung) Nodes zurückgegeben und wenn es mehr hat, wird der moreDataAvailable-Parameter auf `true` gesetzt und man muss die Funktion erneut aufrufen, um die nächsten 20 Nodes zu erhalten.
Da dieses Prozedere aber nur bei Session-basierter Instanzierung der Service-Klasse funktioniert, werden beim Silverlight-spezifischen Service immer gleich alle verfügbaren Nodes beim ersten Aufruf zurückgegeben. Dies ist etwas unschön, da so ein einzelner Aufruf viel länger dauert, falls viele Nodes verfügbar sein sollten. Da dies unter normalen Umständen (bei < 1000 Nodes) vernachlässigbar ist, ist dieser Nachteil aber eher unbedeutend. Bei dem für Silverlight angepassten ServiceContract sieht die GetChildren-Methode nun wie folgt aus:

```
DCBrowserNode[] IWebBosBrowser.GetChildren(DCBrowserNode parent) { ... }
```

5.3 Fehlerverhalten (Fault Behavior)

Da die gesamte WCF-Kommunikation bei Silverlight über den Browser-Network-Stack läuft, werden FaultExceptions, die mit dem HTTP Code 500 (INTERNAL SERVER ERROR) gesendet werden, vom Internet-Browser unterschlagen. Um zu erreichen, dass die FaultExceptions vom Browser an Silverlight weitergegeben werden, muss der Server so konfiguriert werden, dass die Exceptions nicht mit dem HTTP Code 500 sondern mit dem HTTP Code 200 (OK) gesendet werden. Der BOS ist so konfiguriert, dass nur die Silverlight-spezifischen Service-Endpunkte von dieser Änderung beeinflusst werden. Der BOS wurde genau nach der Anleitung konfiguriert, die man in der MSDN unter folgendem Link findet:
http://msdn.microsoft.com/en-us/library/dd470096(VS.96).aspx

6 Asynchrone Servicearchitektur

Silverlight unterstützt nur asynchrone WCF-Serviceaufrufe, die dem Async-Pattern folgen. Da diese Aufrufe über das Netzwerk gehen und daher auch relativ viel Zeit benötigen, macht es auch durchaus Sinn, dass während dieser Zeit nicht das gesamte GUI blockiert ist. Aus diesem Grund würde es auch wenig Sinn machen, den Service-Layer synchron zu machen, da man dann wieder das GUI blockieren würde und den Vorteil der asynchronen WCF-Schnittstelle wieder zunichtemachen würde. Das Async-Pattern, welches bei den WCF-Services eingesetzt wird, ist aber etwas umständlich in der Benutzung. Zuerst muss immer die Begin-Methode aufgerufen werden. Danach muss gewartet werden, bis die Methode abgearbeitet wurde, um dann mit der End-Methode das Resultat abzurufen. Dabei darf das Warten nicht auf dem GUI-Thread ausgeführt werden, da dieser sonst blockiert wird.
Aus diesem Grund haben die Servicefunktionen (nicht WCF-Services) einen zusätzlichen Parameter, mit dem eine Callback-Funktion übergeben werden kann. Mit dieser Callback-Funktion wird dann jeweils am Ende der Servicefunktion das Resultat an den Aufrufer zurückgegeben. Die Callback-Funktion wird aber wieder auf dem GUI-Thread aufgerufen, was den Vorteil hat, dass sich der Aufrufer in keinster Weise um irgendwelche Threads kümmern muss.

Das folgende Sequenzdiagramm zeigt anhand der Funktion GetRootNode() den gesamten Ablauf vom Button-Klick bis zum Funktionsaufruf des WCF-Service-Proxy und wieder zurück. Zu beachten ist, dass der Übersichtlichkeit halber einige Objekte und Aufrufe weggelassen wurden.

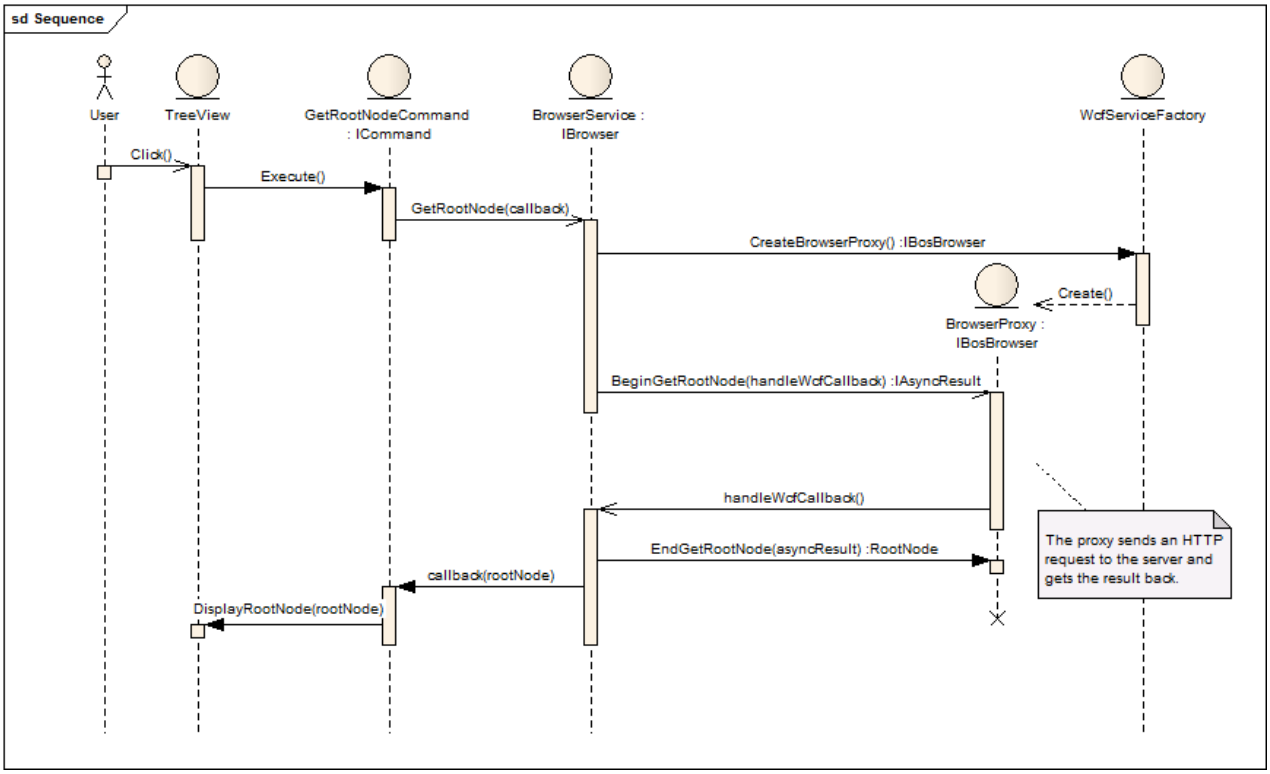


Abbildung 22: Asynchroner Service-Aufruf

Wichtig ist auch, dass die Erzeugung des Service-Proxy nie im GUI-Thread geschieht. Der Grund ist, dass beim ersten Aufruf der Create-Methode für die Proxy-Erzeugung jeweils die Interface-Version des Servers überprüft wird. Falls nun der Server aus irgendeinem Grund nicht erreichbar sein sollte, wird dieser Aufruf erst vom konfigurierten WCF-Timeout beendet. Dieses Timeout kann in der ServiceReferences.ClientConfig Konfigurationsdatei gesetzt werden. So ein Timeout kann unter Umständen mehrere Minuten betragen, während dessen der komplette Browser blockiert wäre.
Wie weiter oben beschrieben, wird das Problem durch die Benutzung der Services aber vollständig gelöst, da jede Serviceoperation auf ihrem eigenen Thread abläuft.

7 Error Handling

Anhand dem Service-Aufruf im *LoginCommand* wird das Error-Handling und die Status- und Fehleranzeige veranschaulicht. Alle Presenter welche Status- oder Fehlermeldungen anzeigen müssen, bekommen eine Instanz des *IStatusHandler* direkt oder des *IContext* durch Dependency-Injection. Der *StatusHandler* ruft auf dem *IStatusPresenter* die Show- oder Hide-Methode auf. Der *StatusPresenter* setzt in der Show-Methode die Properties *BackgroundColor*, *BorderColor* und *StatusMessage*. Durch Databinding wird dadurch die Meldung oder der Fehler in der *StatusView* angezeigt. Die *StatusView* implementiert das Interface *IStatusView* damit der *StatusPresenter* die View ein- und ausblenden kann. Damit sich die Presenter nicht mit den WCF Verbindungsexceptions beschäftigen müssen, wird durch den *ServiceHelper* die *CommunicationExceptionFactory* aufgerufen. Sie gibt bei allen Exceptions welche mit der Serververbindung zu tun haben eine *ServiceUnavailableException* zurück.

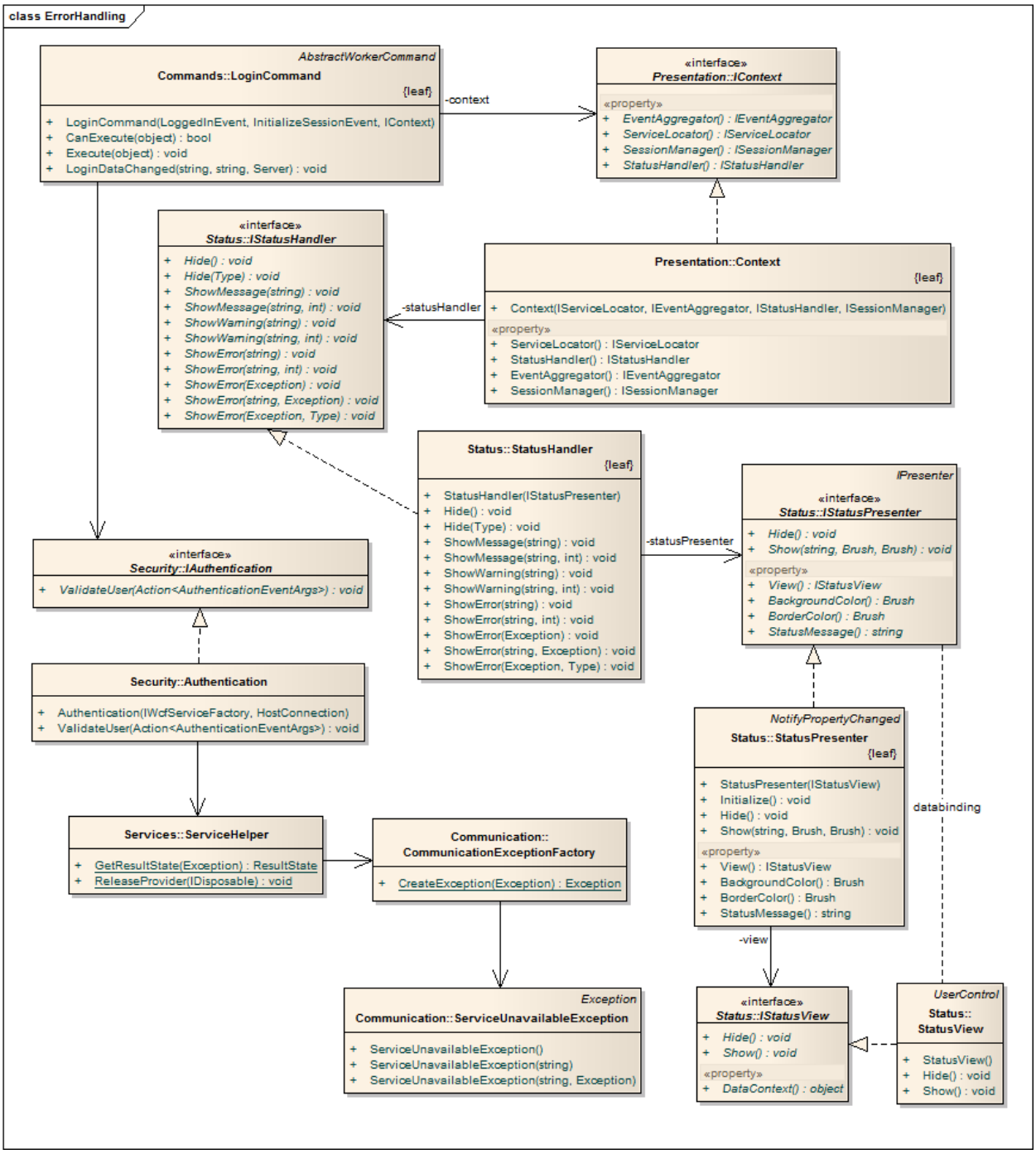


Abbildung 23 Fehlerbehandlung

8 Datenspeicherung

Grundsätzlich sind alle Daten auf dem BOS Server gespeichert und werden im BOS-WebAdmin nur angezeigt und eventuell über WCF Services verändert (Undo-Checkout). Somit spielt die Datenspeicherung in diesem Projekt nur eine kleine Rolle. Im BOS-WebAdmin wird nur eine Serverliste und die Anmeldeinformationen des jeweils letzten Benutzers lokal gespeichert. Im Silverlight steht auf dem Clientsystem für jede Applikation ein kleiner abgetrennter Speicherbereich zur Verfügung. Die Serverliste und die Anmeldeinformationen werden via Serializierung in eine XML-Datei gespeichert.

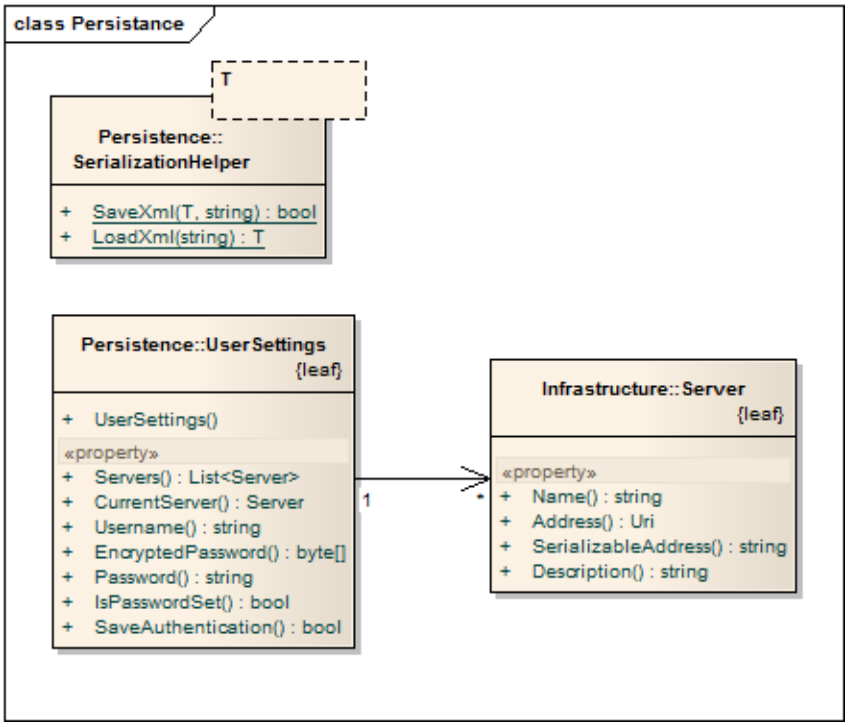


Abbildung 24 Datenspeicherung

9 Grössen und Leistung

Eine der wichtigsten Grössen beim BOS-Server ist die Anzahl der gleichzeitig verbundenen Clients. Diese Grösse wird aber auf dem Server festgelegt und kann vom Client nicht beeinflusst werden. Ein Client muss aber damit rechnen, dass die selben Daten, die er abruft oder sogar bearbeitet, zur gleichen Zeit von einem anderen Client modifiziert werden können. Solch ein gleichzeitiger Zugriff geschieht aber fast ausschliesslich beim lesenden Zugriff. Um einen gleichzeitigen Schreibzugriff zu verhindern, werden die Projektdaten aus- und eingecheckt (ausgecheckte Projekte können von anderen Benutzern nicht geändert werden). Bei der User- und Permission-Verwaltung gibt es zwar keinen solchen Schutz, jedoch wird es kaum vorkommen, dass zwei Benutzer gleichzeitig diese Daten modifizieren, weil dies eine Funktion ist, die nur Administratoren zugänglich ist.

Die Anzahl Benutzer und Projekte, die mit dem BOS-WebAdmin verwaltet werden können soll nicht limitiert sein. Genaue Angaben müssen nicht gemacht werden, da diese Werte in einem relativ überschaubaren Bereichen liegen. So hostet z.B. der Schweizer BOS-Server ca. 2000 Projekte und hat knapp 200 Benutzer. Für grössere Länder sind diese Werte maximal um das 5 bis 10-fache höher. Es gibt also mit Sicherheit keinen Server, der 10 oder 100-tausende von Projekten hosten muss.

Aus zeitlichen Gründen können keine spezifischen Belastungstests für das BOS-WebAdmin durchgeführt werden. Um aber trotzdem das Verhalten bei grösseren Datenmengen testen zu können, wird das BOS-WebAdmin beim Systemtest der Beta-Version mit einer Kopie der echten Projektdaten des Schweizer BOS-Servers getestet.

Projekt: BOS-WebAdmin

Prototyp - Testdokumentation

[Dokumentstruktur basiert auf RUP „Dokument Test Evaluation Summary“]

0 Dokumentinformationen

0.1 Änderungsgeschichte

Datum	Version	Änderung	Autor
01.10.2009	0.1	Dokument erstellt und Test für UC1, UC2 und UC3 beschrieben.	Reto Schneebeli
01.10.2009	1.0	Testausführung für Prototyp	Reto Schneebeli

0.2 Inhalt

0 Dokumentinformationen 2
0.1 Änderungsgeschichte 2
0.2 Inhalt 2
1 Voraussetzungen 3
2 Vorbereitungen 3
3 Testdurchführung 3
4 Systemtest 3
4.1 UC01: BOS-Server verbinden 3
4.2 UC02: XWP-Projekte browsen 4
4.3 UC03: BOS-Server-Verbindung trennen 5
4.4 Übersicht 5
5 Verbesserungsmöglichkeiten 6
5.1 Bekannte Einschränkungen 6

1 Voraussetzungen

Ein laufender BOS-Server und die entsprechenden Zugangsdaten (Adresse, Username, Passwort). Eine Verbindung mit dem BOS-Server kann aufgebaut werden, z.B. mit dem XWP Projekt Manager.

2 Vorbereitungen

Keine notwendig.

3 Testdurchführung

Datum: 01.11.2009
SVN Revisionnummer: 173
Tester: Reto Schneebeli
Hardware: Notebook (Core 2 Duo 2GHz, 4GB Ram)
Betriebssystem: Windows 7 Professional, 64 Bit
Browser: Firefox 3.5.4
BOS Server: WmWare localhost

4 Systemtest

Den BOS-WebAdmin starten, bedeutet immer den Browser mit der BOS-WebAdmin Seite öffnen.

4.1 UC01: BOS-Server verbinden

Test 1: BOS Server verbinden (einfach)	**Resultat**	**OK**
Beschreibung:	**Erwartung:**	
1. BOS-WebAdmin starten. 2. BOS-Server auswählen. 3. Korrekten Benutzernamen und Passwort eingeben. 4. Button „Login“ betätigen.	1. Die Login-Seite wird angezeigt. 2. Keine Fehlermeldung oder Verzögerung. 3. Login Button wird enabled. 4. Projektbaum wird in weniger als 4 Sekunden angezeigt. Und die 'Successfull logged in" Statusmeldung verschwindet nach einigen Sekunden.	
Fehler, Abweichung:		
Kommentar:	Es ist nirgends ersichtlich wenn der Projektbaum geladen wird. Ein loading Text wäre hilfreich, vor allem wenn es länger geht.	

Test 2: BOS Server verbinden (falsche Benutzerdaten)	**Resultat**	**OK**
Beschreibung:	**Erwartung:**	
1. BOS-WebAdmin starten. 2. BOS-Server auswählen. 3. Falschen Benutzernamen oder falsches Passwort eingeben. 4. Button „Login“ betätigen.	1. Die Login-Seite wird angezeigt. 2. Keine Fehlermeldung oder Verzögerung. 3. Login-Button wird enabled. 4. Fehlermeldung wird angezeigt und die Login-Seite bleibt.	
Fehler, Abweichung:		
Kommentar:	Bessere Fehlermeldung.	

Test 3: BOS Server verbinden (erweitert)	**Resultat**	**na**
Beschreibung:	**Erwartung:**	
1. BOS-WebAdmin starten. 2. BOS-Server auswählen. 3. Benutzernamen und Passwort eingeben. 4. Wählen dass die Anmeldeinformationen gespeichert werden sollen. 5. Button „Login“ betätigen. 6. Refresh Button im Browser betätigen.	1. Die Login-Seite wird angezeigt. 2. Keine Fehlermeldung oder Verzögerung. 3. Login Button wird enabled. 4. Keine Fehlermeldung oder Verzögerung. 5. Projektbaum wird angezeigt. 6. Die Login-Seite wird angezeigt und die Anmeldeinformationen (Benutzername und Passwort) sind bereits eingegeben.	
Fehler, Abweichung:	Noch nicht implementiert.	
Kommentar:		

Test 4: BOS Server hinzufügen/löschen/ändern	**Resultat**	**na**
Beschreibung:	**Erwartung:**	
Im Prototyp noch nicht implementiert.		
Fehler, Abweichung:	Noch nicht implementiert.	
Kommentar:		

4.2 UC02: XWP-Projekte browsen

Test 1: Projektbaum navigieren.	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	
Beschreibung:	**Erwartung:**	
1. Vorbedingung: Mit dem BOS Server verbinden (einfach) 2. Ordner und Projekte im Baum öffnen, schliessen. 3. Ordner im Baum selektieren. 4. Projekt im Baum selektieren.	1. Projektbaum wird angezeigt und die erste Ebene ist bereits aufgeklappt. 2. Die Unterordner oder Projekte werden angezeigt. Die Unterelemente werden immer eine Ebene tiefer vorgeladen, das GUI darf jedoch nie einfrieren. 3. Entsprechende Detaildaten werden im rechts angezeigt. 4. Entsprechende Detaildaten werden im rechts angezeigt.	
Fehler, Abweichung:		
Kommentar:	Detaildaten sind im Prototyp einfach der Ordner- oder Projektname.	

Test 2: Verbindung beim Navigieren trennen	**Resultat**	**OK**
Vorbedingung:	Test 1: Projektbaum navigieren.	
Beschreibung:	**Erwartung:**	

1. Die Verbindung zum BOS Server trennen (Netzwerkverbindung unterbrechen, WmWare pausieren oder IIS stoppen) . 2. Einen weiteren Ordner öffnen, welcher Unterelemente hat und noch nicht geöffnet wurde. 3. Die Verbindung zum BOS Server wieder herstellen. 4. Einen weiteren Ordner öffnen, welcher Unterelemente hat und noch nicht geöffnet wurde.	1. Nichts passiert. 2. Eine entsprechende Fehlermeldung wird angezeigt. 3. Nichts passiert. 4. Die Fehlermeldung verschwindet und das navigieren funktioniert wieder.
Fehler, Abweichung:	
Kommentar:	Damit die Unterelemente vom selben Ordner angezeigt werden, ist ein Refresh nötig. Exceptions werden nicht geloggt.

4.3 UC03: BOS-Server-Verbindung trennen

Test 1: Verbindung trennen	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	
Beschreibung:	**Erwartung:**	
1. Button „Logout“ betätigen.	1. Die Login-Seite wird angezeigt.	
Fehler, Abweichung:		
Kommentar:		

4.4 Übersicht

Use Case	*Implementiert*	*Fehler/Unschönheit*	*Status*
UC01: BOS-Server verbinden	teilweise	Test 1: Es ist nirgends ersichtlich wenn der Projektbaum geladen wird. Ein loading Text wäre hilfreich, vor allem wenn es länger geht. Test 2: Bessere Fehlermeldung.	**OK**
UC02: XWP-Projekte browsen	teilweise	Test 1: Detaildaten sind im Prototyp einfach der Ordner- oder Projektname. Test 2: Damit die Unterelemente vom selben Ordner angezeigt werden, ist ein Refresh nötig. Exceptions werden nicht geloggt.	**OK**
UC03: BOS-Server-Verbindung trennen	ja		**OK**
UC04: Auschecken rückgängig machen (Undo check-out)	nein		
UC05: History ansehen	nein		
UC06: Suchen	nein		
UC07: Benutzer verwalten (CRUD)	nein		

UC08: Benutzergruppen verwalten (CRUD)	nein		
UC09: Rollen verwalten (CRUD)	nein		

5 Verbesserungsmöglichkeiten

5.1 Bekannte Einschränkungen

Im Prototyp ist nur das 'Main Success Scenario' implementiert und nur dieses wird getestet. Alle bestehenden Tests werden später noch erweitert.

Projekt: BOS-WebAdmin

Alpha Release - Testdokumentation

[Dokumentstruktur basiert auf RUP „Dokument Test Evaluation Summary“]

0 Dokumentinformationen

0.1 Änderungsgeschichte

Datum	Version	Änderung	Autor
01.10.2009	0.1	Dokument erstellt und Test für UC1, UC2 und UC3 beschrieben.	Reto Schneebeli
22.11.2009	0.2	Dokument mit dem Gerüst der restlichen UseCases erweitert. Test für UC07 beschrieben.	Reto Schneebeli
22.11.2009	0.3	Test für UC01, UC07, UC08 angepasst	Kaspar Fenner
22.11.2009	1.0	Testausführung für Alpha Release	Kaspar Fenner

0.2 Inhalt

0 Dokumentinformationen 2
0.1 Änderungsgeschichte 2
0.2 Inhalt 2
1 Voraussetzungen 3
2 Vorbereitungen 3
3 Testdurchführung 3
4 Systemtest 3
4.1 UC01: BOS-Server verbinden 3
4.2 UC02: XWP-Projekte browsen 4
4.3 UC03: BOS-Server-Verbindung trennen 5
4.4 UC04: Auschecken rückgängig machen (Undo check-out) 5
4.5 UC05: History ansehen 6
4.6 UC06: Suchen 6
4.7 UC07: Benutzer verwalten (CRUD) 6
4.8 UC08: Benutzergruppen verwalten (CRUD) 7
4.9 UC09: Rollen verwalten (CRUD) 8
4.10 UC10: Generelle Berechtigungen verwalten 8
4.11 UC11: Projekt/Ordner-spezifische Berechtigungen verwalten 8
4.12 UC12: Rolle zu Benutzer zuweisen 8
4.13 UC13: Rolle zu Benutzergruppe zuweisen 8
4.14 UC14: History bereinigen 8
4.15 Übersicht 8
5 Verbesserungsmöglichkeiten 9
5.1 Bekannte Einschränkungen 9

1 Voraussetzungen

Ein laufender BOS-Server und die entsprechenden Zugangsdaten (Adresse, Username, Passwort). Eine Verbindung mit dem BOS-Server kann aufgebaut werden, z.B. mit dem XWP Projekt Manager.

2 Vorbereitungen

Keine notwendig.

3 Testdurchführung

Datum: 22.11.2009
SVN Revisionnummer: 293
Tester: Kaspar Fenner
Hardware: Notebook (Core 2 Duo 1.8GHz, 3GB Ram)
Betriebssystem: Windows 7 Professional, 64 Bit
Browser: Internet Explorer 8
BOS Server: WmWare localhost

4 Systemtest

Den BOS-WebAdmin starten, bedeutet immer den Browser mit der BOS-WebAdmin Seite öffnen.

4.1 UC01: BOS-Server verbinden

Test 1: BOS Server verbinden (einfach)	**Resultat**	**OK**
Beschreibung:	**Erwartung:**	
1. BOS-WebAdmin starten. 2. BOS-Server auswählen. 3. Korrekten Benutzernamen und Passwort eingeben. 4. Button „Login“ betätigen.	1. Die Login-Seite wird angezeigt. 2. Keine Fehlermeldung oder Verzögerung. 3. Login Button wird enabled. 4. Projektbaum wird in weniger als 4 Sekunden angezeigt. Und die 'Successfull loged in" Statusmeldung verschwindet nach einigen Sekunden.	
Fehler, Abweichung:		
Kommentar:		

Test 2: BOS Server verbinden (falsche Benutzerdaten)	**Resultat**	**OK**
Beschreibung:	**Erwartung:**	
1. BOS-WebAdmin starten. 2. BOS-Server auswählen. 3. Falschen Benutzernamen oder falsches Passwort eingeben. 4. Button „Login“ betätigen.	1. Die Login-Seite wird angezeigt. 2. Keine Fehlermeldung oder Verzögerung. 3. Login Button wird enabled. 4. Fehlermeldung wird angezeigt und Login-Seite bleibt.	
Fehler, Abweichung:		
Kommentar:		

Test 3: BOS Server verbinden (erweitert)	**Resultat**	**n/a**
Beschreibung:	**Erwartung:**	
1. BOS-WebAdmin starten. 2. BOS-Server auswählen. 3. Benutzernamen und Passwort eingeben. 4. Wählen dass die Anmeldeinformationen gespeichert werden sollen. 5. Button „Login“ betätigen.	1. Die Login-Seite wird angezeigt. 2. Keine Fehlermeldung oder Verzögerung. 3. Login Button wird enabled. 4. Keine Fehlermeldung oder Verzögerung. 5. Projektbaum wird in weniger als 4 Sekunden angezeigt. Und die 'Successfull loged in" Statusmeldung verschwindet nach einigen Sekunden.	
Fehler, Abweichung:	Noch nicht implementiert.	
Kommentar:		

Test 4: BOS Server hinzufügen/löschen/ändern	**Resultat**	**OK**
Beschreibung:	**Erwartung:**	
1. BOS-WebAdmin starten. 2. BOS-Server Liste anklicken. 3. Einen neuen Server hinzufügen. 4. Einen bestehenden Server löschen. 5. Einen Server bearbeiten. 6. Refresh Button im Browser betätigen.	1. Die Login-Seite wird angezeigt. 2. Keine Fehlermeldung oder Verzögerung. 3. Server erscheint in der Server-Liste. 4. Server wird aus der Server-Liste gelöscht. 5. Geänderte Serverdaten werden in der Server-Liste angezeigt. 6. Server-Liste beinhaltet den hinzugefügten und den veränderten Server mit den korrekten Angaben.	
Fehler, Abweichung:		
Kommentar:	Unschön: Zu lange URL sowie Beschreibung wird beim Anzeigen abgetrennt. Unschön: Editier-Modus kann nicht durch „Abort“ abgebrochen werden. Verbesserung: ENTER als Bestätigung beim Eingeben eines Servers.	

4.2 UC02: XWP-Projekte browsen

Test 1: Projektbaum navigieren.	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	
Beschreibung:	**Erwartung:**	

1. Vorbedingung: Mit dem BOS Server verbinden (einfach) 2. Ordner und Projekte im Baum öffnen, schliessen. 3. Ordner im Baum selektieren. 4. Projekt im Baum selektieren.	1. Projektbaum wird angezeigt und die erste Ebene ist bereits aufgeklappt. 2. Die Unterordner oder Projekte werden angezeigt. Die Unterelemente werden immer eine Ebene tiefer vorgeladen, das GUI darf jedoch nie einfrieren. 3. Entsprechende Detaildaten werden rechts angezeigt. 4. Entsprechende Detaildaten werden rechts angezeigt.
Fehler, Abweichung:	
Kommentar:	

Test 2: Verbindung beim Navigieren trennen	**Resultat**	**OK**
Vorbedingung:	Test 1: Projektbaum navigieren.	
Beschreibung:	**Erwartung:**	
1. Die Verbindung zum BOS Server trennen (Netzwerkverbindung unterbrechen, WmWare pausieren oder IIS stoppen) . 2. Einen weiteren Ordner öffnen, welcher Unterelemente hat und noch nicht geöffnet wurde. 3. Die Verbindung zum BOS Server wieder herstellen. 4. Einen weiteren Ordner öffnen, welcher Unterelemente hat und noch nicht geöffnet wurde.	1. Nichts passiert. 2. Eine entsprechende Fehlermeldung wird angezeigt. 3. Nichts passiert. 4. Die Fehlermeldung verschwindet und das navigieren funktioniert wieder.	
Fehler, Abweichung:		
Kommentar:		

4.3 UC03: BOS-Server-Verbindung trennen

Test 1: Verbindung trennen	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	
Beschreibung:	**Erwartung:**	
1. Button „Logout“ betätigen.	1. Die Login-Seite wird angezeigt.	
Fehler, Abweichung:		
Kommentar:		

4.4 UC04: Auschecken rückgängig machen (Undo check-out)

Test 1: Undo-Checkout Node	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	

Beschreibung:	**Erwartung:**
1. Undo-Checkout auf einem einzelnen Node durchführen.	1. Das Undo-Checkout Icon verschwindet und der Node ist nicht mehr ausgecheckt. Falls es ein virtueller Node war, wie z.B. die Netzwerkdaten oder Paneldaten wird das Undo-Checkout auf dem ganzen virtuellen Node durchgeführt.
Fehler, Abweichung:	
Kommentar:	

Test 1: Undo-Checkout Projekt	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	
Beschreibung:	**Erwartung:**	
1. Undo-Checkout auf einem ganzen Projekt durchführen.	1. Das Undo-Checkout Icon verschwindet auf allen Nodes des Projektes und alle Nodes sind nicht mehr ausgecheckt.	
Fehler, Abweichung:		
Kommentar:	Unschön: Beim Undo-checkout auf einem Projekt-Node im Tree wird danach der „unsichtbare“ Projekt-Data-Node selektiert, der als kleiner runder Punkt erscheint. -> Es sollte wieder der Projekt-Node selektiert werden	

4.5 UC05: History ansehen

Noch nicht implementiert.

4.6 UC06: Suchen

Noch nicht implementiert.

4.7 UC07: Benutzer verwalten (CRUD)

Test 1: Benutzer hinzufügen, bearbeiten und löschen	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	
Beschreibung:	**Erwartung:**	
1. Ins Tab Security wechseln. 2. Ins Tab Users wechseln. 3. Einen Benutzer hinzufügen. 4. Den Benutzer auswählen und den Vornamen, den Nachnamen, die Email-Adresse und die Beschreibung verändern. 5. Die Veränderung speichern. 6. Einen anderen Benutzer auswählen und löschen. 7. Den Refresh-Button betätigen.	1. Das Tab Security wird angezeigt. 2. Eine Liste mit allen Benutzern ist sichtbar. 3. Der hinzugefügte Benutzer erscheint in der Benutzerliste. 4. Ein Speichern-Button wird angezeigt. 5. Die Änderungen wurden gespeichert. 6. Der Benutzer wird aus der Liste gelöscht. 7. Es verändert sich höchstens die Reihenfolge der Benutzer, alle anderen Daten bleiben erhalten.	
Fehler, Abweichung:	Schritt 3: Falls ein neuer Benutzer mit einem Namen, den es bereits gibt, erstellt wird, erscheint die Meldung „Operation failed.“, aber in Grün (also wie wenn es erfolgreich wäre!).	

Kommentar:	

Test 2: Benutzergruppen-Zuweisung verändern	**Resultat**	**OK**
Vorbedingung:	UC7 Benutzer verwalten CRUD: Test 1: Benutzer hinzufügen, bearbeiten und löschen	

Beschreibung:	**Erwartung:**
1. Einen Benutzer auswählen. 2. Einen nicht zugewiesene Benutzergruppe zuweisen. 3. Eine zugewiesenen Benutzergruppe entfernen.	1. Die zugewiesenen Gruppen und alle noch nicht zugewiesenen Gruppen werden angezeigt. 2. Die Benutzergruppe verschwindet und erscheint neu in der Liste der zugewiesenen Benutzergruppen. 3. Die Benutzergruppe verschwindet und erscheint neu in der Liste der nicht zugewiesenen Benutzergruppen.
Fehler, Abweichung:	
Kommentar:	

Test 3: Benutzer-Passwort verändern	**Resultat**	**OK**
Vorbedingung:	UC7 Benutzer verwalten CRUD: Test 1: Benutzer hinzufügen, bearbeiten und löschen	

Beschreibung:	**Erwartung:**
1. Einen Benutzer auswählen. 2. Den Change-Password-Button drücken. 3. Das alte, sowie neue Passwort eingeben und abbrechen. 4. Das alte und neue Passwort eingeben und bestätigen.	1. Der Change-Password-Button wird aktiv. 2. Die Textboxen für die Eingabe des alten und neuen Passwortes werden eingeblendet. 3. Die Textboxen von Schritt 2 werden ausgeblendet. 4. Die Textboxen werden ausgeblendet und das Passwort wurde geändert (überprüfen mit einem Re-Login).
Fehler, Abweichung:	
Kommentar:	

4.8 UC08: Benutzergruppen verwalten (CRUD)

Test 1: Benutzergruppe hinzufügen, bearbeiten und löschen	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	

Beschreibung:	**Erwartung:**
1. Ins Tab Security wechseln. 2. Ins Tab 'User Groups' wechseln. 3. Eine Benutzergruppe hinzufügen. 4. Eine andere Benutzergruppe auswählen und löschen. 5. Den Refresh-Button betätigen.	1. Das Tab Security wird angezeigt. 2. Eine Liste mit allen Benutzergruppen ist sichtbar. 3. Die hinzugefügte Benutzergruppe erscheint in der Liste. 4. Die Benutzergruppe wird aus der Liste gelöscht. 5. Es verändert sich höchstens die Reihenfolge der Benutzergruppen.

Fehler, Abweichung:	Schritt 3: Falls eine neue Benutzergruppe mit einem Namen, den es bereits gibt, erstellt wird, erscheint die Meldung „Operation failed.“, aber in Grün (also wie wenn es erfolgreich wäre!).
Kommentar:	

Test 2: Benutzer-Zuweisung verändern		**Resultat**	**OK**
Vorbedingung:	UC7 Benutzergruppen verwalten CRUD: Test 1: Benutzergruppen hinzufügen, bearbeiten und löschen		
Beschreibung:		**Erwartung:**	
1. Eine Benutzergruppe auswählen. 2. Einen nicht zugewiesenen Benutzer zuweisen. 3. Einen zugewiesenn Benutzer entfernen.		1. Die zugewiesenen Benutzer und alle noch nicht zugewiesenen Benutzer werden angezeigt. 2. Der Benutzer verschwindet und erscheint neu in der Liste der zugewiesenen Benutzer. 3. Der Benutzer verschwindet und erscheint neu in der Liste der nicht zugewiesenen Benutzer.	
Fehler, Abweichung:			
Kommentar:			

4.9 UC09: Rollen verwalten (CRUD)

Noch nicht implementiert.

4.10 UC10: Generelle Berechtigungen verwalten

Noch nicht implementiert.

4.11 UC11: Projekt/Ordner-spezifische Berechtigungen verwalten

Noch nicht implementiert.

4.12 UC12: Rolle zu Benutzer zuweisen

Sub-UseCase von UC10 und UC11.

4.13 UC13: Rolle zu Benutzergruppe zuweisen

Sub-UseCase von UC10 und UC11.

4.14 UC14: History bereinigen

Wird nicht umgesetzt.

4.15 Übersicht

Use Case	*Implementiert*	*Fehler/Unschönheit*	*Status*
UC01: BOS-Server verbinden	teilweise	Unschön: Zu lange URL sowie Beschreibung wird beim Anzeigen abgetrennt. Unschön: Editier-Modus kann nicht durch „Abort“ abgebrochen werden. Verbesserung: ENTER als Bestätigung beim Eingeben eines Servers.	**OK**
UC02: XWP-Projekte browsen	ja		**OK**
UC03: BOS-Server-	ja		**OK**

Verbindung trennen			
UC04: Auschecken rückgängig machen (Undo check-out)	ja	Unschön: Beim Undo-checkout auf einem Projekt-Node im Tree wird danach der „unsichtbare“ Projekt-Data-Node selektiert, der als kleiner runder Punkt erscheint. -> Es sollte wieder der Projekt-Node selektiert werden	**OK**
UC05: History ansehen	nein		
UC06: Suchen	nein		
UC07: Benutzer verwalten (CRUD)	ja	Schritt 3: Falls ein neuer Benutzer mit einem Namen, den es bereits gibt, erstellt wird, erscheint die Meldung „Operation failed.“, aber in Grün (also wie wenn es erfolgreich wäre!).	**OK**
UC08: Benutzergruppen verwalten (CRUD)	ja	Schritt 3: Falls eine neue Benutzergruppe mit einem Namen, den es bereits gibt, erstellt wird, erscheint die Meldung „Operation failed.“, aber in Grün (also wie wenn es erfolgreich wäre!).	**OK**
UC09: Rollen verwalten (CRUD)	nein		
UC10: Generelle Berechtigungen verwalten	nein		
UC11: Projekt/Ordner-spezifische Berechtigungen verwalten	nein		
UC12: Rolle zu Benutzer zuweisen	nein		
UC13: Rolle zu Benutzergruppe zuweisen	nein		
UC14: History bereinigen	wird nicht		

5 Verbesserungsmöglichkeiten

5.1 Bekannte Einschränkungen

- Ob ein Element vom eigenen oder von einem anderen Benutzer ausgecheckt wurde, ist noch nicht sichtbar.
- Undo-Checkout- und Delete-Funktionen haben noch keinen Schutz, bzw. werden direkt ausgeführt.
- Rechte werden noch nicht im vorherein überprüft, es gibt bei zu wenig Recht einfach eine Fehlermeldung.
- Ordner und Projekte werden beim Browser noch durchmischt dargestellt, die Liste ist falsch sortiert.

Projekt: BOS-WebAdmin

Beta Release - Testdokumentation

[Dokumentstruktur basiert auf RUP „Dokument Test Evaluation Summary“]

0 Dokumentinformationen

0.1 Änderungsgeschichte

Datum	Version	Änderung	Autor
01.10.2009	0.1	Dokument erstellt und Test für UC1, UC2 und UC3 beschrieben.	Reto Schneebeli
22.11.2009	0.2	Dokument mit dem Gerüst der restlichen UseCases erweitert. Test für UC07 beschrieben.	Reto Schneebeli
22.11.2009	0.3	Test für UC01, UC07, UC08 angepasst	Kaspar Fenner
07.12.2009	0.4	Test für UC9, UC10, UC11, UC12 und UC13 beschrieben.	Reto Schneebeli
07.12.2009	1.0	Testausführung für Beta-Release	Reto Schneebeli

0.2 Inhalt

0 Dokumentinformationen 2
0.1 Änderungsgeschichte 2
0.2 Inhalt 2
1 Voraussetzungen 3
2 Vorbereitungen 3
3 Testdurchführung 3
4 Systemtest 3
4.1 UC01: BOS-Server verbinden 3
4.2 UC02: XWP-Projekte browsen 4
4.3 UC03: BOS-Server-Verbindung trennen 5
4.4 UC04: Auschecken rückgängig machen (Undo check-out) 5
4.5 UC05: History ansehen 6
4.6 UC06: Suchen 6
4.7 UC07: Benutzer verwalten (CRUD) 6
4.8 UC08: Benutzergruppen verwalten (CRUD) 7
4.9 UC09: Rollen verwalten (CRUD) 8
4.10 UC10: Generelle Berechtigungen verwalten 9
4.11 UC11: Projekt/Ordner-spezifische Berechtigungen verwalten 9
4.12 UC12: Rolle zu Benutzer zuweisen 10
4.13 UC13: Rolle zu Benutzergruppe zuweisen 10
4.14 UC14: History bereinigen 11
4.15 Übersicht 11
5 Verbesserungsmöglichkeiten 11
5.1 Bekannte Einschränkungen 11

1 Voraussetzungen

Ein laufender BOS-Server und die entsprechenden Zugangsdaten (Adresse, Username, Passwort). Eine Verbindung mit dem BOS-Server kann aufgebaut werden, z.B. mit dem XWP Projekt Manager.

2 Vorbereitungen

Keine notwendig.

3 Testdurchführung

Datum: 07.12.2009
SVN Revisionnummer: 390
Tester: Reto Schneebeli
Hardware: Notebook (Core 2 Duo 2GHz, 4GB Ram)
Betriebssystem: Windows 7 Professional, 64 Bit
Browser: Firefox 3.5.4
BOS Server: WmWare localhost

4 Systemtest

Den BOS-WebAdmin starten, bedeutet immer den Browser mit der BOS-WebAdmin Seite öffnen.

4.1 UC01: BOS-Server verbinden

Test 1: BOS Server verbinden (einfach)	**Resultat**	**OK**
Beschreibung:	**Erwartung:**	
1. BOS-WebAdmin starten. 2. BOS-Server auswählen. 3. Korrekten Benutzernamen und Passwort eingeben. 4. Button „Login“ betätigen.	1. Die Login-Seite wird angezeigt. 2. Keine Fehlermeldung oder Verzögerung. 3. Login Button wird enabled. 4. Projektbaum wird in weniger als 4 Sekunden angezeigt. Und die 'Successfull loged in" Statusmeldung verschwindet nach einigen Sekunden.	
Fehler, Abweichung:		
Kommentar:		

Test 2: BOS Server verbinden (falsche Benutzerdaten)	**Resultat**	**OK**
Beschreibung:	**Erwartung:**	
1. BOS-WebAdmin starten. 2. BOS-Server auswählen. 3. Falschen Benutzernamen oder falsches Passwort eingeben. 4. Button „Login“ betätigen.	1. Die Login-Seite wird angezeigt. 2. Keine Fehlermeldung oder Verzögerung. 3. Login Button wird enabled. 4. Fehlermeldung wird angezeigt und Login-Seite bleibt.	
Fehler, Abweichung:		
Kommentar:		

Test 3: BOS Server verbinden (erweitert)	**Resultat**	**OK**
Beschreibung:	**Erwartung:**	
1. BOS-WebAdmin starten. 2. BOS-Server auswählen. 3. Benutzernamen und Passwort eingeben. 4. Wählen dass die Anmeldeinformationen gespeichert werden sollen. 5. Button „Login“ betätigen. 6. Logout-Button betätigen. 7. Refresh Button im Browser betätigen. 8. Abwählen, so dass die Anmeldeinformationen nicht gespeichert werden sollen. 9. Refresh Button im Browser betätigen.	1. Die Login-Seite wird angezeigt. 2. Keine Fehlermeldung oder Verzögerung. 3. Login Button wird enabled. 4. Keine Fehlermeldung oder Verzögerung. 5. Projektbaum wird angezeigt. 6. Die Login-Seite wird angezeigt mit den vorhin eingegebenen Anmeldeinformationen. 7. Die Login-Seite wird angezeigt mit den vorhin eingegebenen Anmeldeinformationen. 8. Keine Fehlermeldung oder Verzögerung. 9. Die Login-Seite wird angezeigt, aber der Benutzername und das Passwort ist leer.	
Fehler, Abweichung:		
Kommentar:		

Test 4: BOS Server hinzufügen/löschen/ändern	**Resultat**	**OK**
Beschreibung:	**Erwartung:**	
1. BOS-WebAdmin starten. 2. BOS-Server Liste anklicken. 3. Einen neuen Server hinzufügen. 4. Einen bestehenden Server löschen. 5. Einen Server bearbeiten. 6. Refresh Button im Browser betätigen.	1. Die Login-Seite wird angezeigt. 2. Keine Fehlermeldung oder Verzögerung. 3. Server erscheint in der Server-Liste. 4. Server wird aus der Server-Liste gelöscht. 5. Geänderte Serverdaten werden in der Server-Liste angezeigt. 6. Server-Liste beinhaltet den hinzugefügten und den veränderten Server mit den korrekten Angaben.	
Fehler, Abweichung:		
Kommentar:		

4.2 UC02: XWP-Projekte browsen

Test 1: Projektbaum navigieren.	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	
Beschreibung:	**Erwartung:**	

1. Vorbedingung: Mit dem BOS Server verbinden (einfach) 2. Ordner und Projekte im Baum öffnen, schliessen. 3. Ordner im Baum selektieren. 4. Projekt im Baum selektieren.	1. Projektbaum wird angezeigt und die erste Ebene ist bereits aufgeklappt. 2. Die Unterordner oder Projekte werden angezeigt. Die Unterelemente werden immer eine Ebene tiefer vorgeladen, das GUI darf jedoch nie einfrieren. 3. Entsprechende Detaildaten werden rechts angezeigt. 4. Entsprechende Detaildaten werden rechts angezeigt.
Fehler, Abweichung:	
Kommentar:	

Test 2: Verbindung beim Navigieren trennen	**Resultat**	**OK**
Vorbedingung:	Test 1: Projektbaum navigieren.	
Beschreibung:	**Erwartung:**	
1. Die Verbindung zum BOS Server trennen (Netzwerkverbindung unterbrechen, WmWare pausieren oder IIS stoppen) . 2. Einen weiteren Ordner öffnen, welcher Unterelemente hat und noch nicht geöffnet wurde. 3. Die Verbindung zum BOS Server wieder herstellen. 4. Einen weiteren Ordner öffnen, welcher Unterelemente hat und noch nicht geöffnet wurde.	1. Nichts passiert. 2. Eine entsprechende Fehlermeldung wird angezeigt. 3. Nichts passiert. 4. Die Fehlermeldung verschwindet und das navigieren funktioniert wieder.	
Fehler, Abweichung:		
Kommentar:	Beim Element bei dem beim Laden die Verbindung getrennt wurde, werden auch nach dem Wiederaufbau der Verbindung keine Unterelemente geladen.	

4.3 UC03: BOS-Server-Verbindung trennen

Test 1: Verbindung trennen	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	
Beschreibung:	**Erwartung:**	
1. Button „Logout“ betätigen.	1. Die Login-Seite wird angezeigt.	
Fehler, Abweichung:		
Kommentar:		

4.4 UC04: Auschecken rückgängig machen (Undo check-out)

Test 1: Undo-Checkout Node	**Resultat**	**OK**

Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)
Beschreibung:	**Erwartung:**
1. Undo-Checkout auf einem einzelnen Node starten. 2. Commit-Button betätigen.	1. Ein kleines Fenster mit dem Node für das Undo-Checkout wird angezeigt. 2. Das Undo-Checkout Icon verschwindet und der Node ist nicht mehr ausgecheckt. Falls es ein virtueller Node war, wie z.B. die Netzwerkdaten oder Paneldaten wird das Undo-Checkout auf dem ganzen virtuellen Node durchgeführt.
Fehler, Abweichung:	
Kommentar:	

Test 1: Undo-Checkout Projekt	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	
Beschreibung:	**Erwartung:**	
1. Undo-Checkout auf einem ganzen Projekt starten. 2. Commit-Button betätigen.	1. Ein kleines Fenster mit allen Nodes des Projektes für das Undo-Checkout wird angezeigt. 2. Das Undo-Checkout Icon verschwindet auf allen Nodes des Projektes und alle Nodes sind nicht mehr ausgecheckt.	
Fehler, Abweichung:		
Kommentar:		

4.5 UC05: History ansehen

Wird nicht umgesetzt.

4.6 UC06: Suchen

Wird nicht umgesetzt.

4.7 UC07: Benutzer verwalten (CRUD)

Test 1: Benutzer hinzufügen, bearbeiten und löschen	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	
Beschreibung:	**Erwartung:**	
1. Ins Tab Security wechseln. 2. Ins Tab Users wechseln. 3. Einen Benutzer hinzufügen. 4. Den Benutzer auswählen und den Vornamen, den Nachnamen, die Email-Adresse und die Beschreibung verändern. 5. Die Veränderung speichern. 6. Einen anderen Benutzer auswählen und löschen. 7. Den Refresh-Button betätigen.	1. Das Tab Security wird angezeigt. 2. Eine Liste mit allen Benutzern ist sichtbar. 3. Der hinzugefügte Benutzer erscheint in der Benutzerliste. 4. Ein Speichern-Button wird angezeigt. 5. Die Änderungen wurden gespeichert. 6. Der Benutzer wird aus der Liste gelöscht. 7. Es verändert sich höchstens die Reihenfolge der Benutzer, alle anderen Daten bleiben erhalten.	

Fehler, Abweichung:	
Kommentar:	

Test 2: Benutzergruppen-Zuweisung verändern		**Resultat**	**OK**
Vorbedingung:	UC7 Benutzer verwalten CRUD: Test 1: Benutzer hinzufügen, bearbeiten und löschen		
Beschreibung:	**Erwartung:**		
1. Einen Benutzer auswählen. 2. Einen nicht zugewiesene Benutzergruppe zuweisen. 3. Eine zugewiesenen Benutzergruppe entfernen.	1. Die zugewiesenen Gruppen und alle noch nicht zugewiesenen Gruppen werden angezeigt. 2. Die Benutzergruppe verschwindet und erscheint neu in der Liste der zugewiesenen Benutzergruppen. 3. Die Benutzergruppe verschwindet und erscheint neu in der Liste der nicht zugewiesenen Benutzergruppen.		
Fehler, Abweichung:			
Kommentar:			

Test 3: Benutzer-Passwort verändern		**Resultat**	**OK**
Vorbedingung:	UC7 Benutzer verwalten CRUD: Test 1: Benutzer hinzufügen, bearbeiten und löschen		
Beschreibung:	**Erwartung:**		
1. Einen Benutzer auswählen. 2. Den Change-Password-Button drücken. 3. Das alte, sowie neue Passwort eingeben und abbrechen. 4. Das alte und neue Passwort eingeben und bestätigen.	1. Der Change-Password-Button wird aktiv. 2. Die Textboxen für die Eingabe des alten und neuen Passwortes werden eingeblendet. 3. Die Textboxen von Schritt 2 werden ausgeblendet. 4. Die Textboxen werden ausgeblendet und das Passwort wurde geändert (überprüfen mit einem Re-Login).		
Fehler, Abweichung:			
Kommentar:			

4.8 UC08: Benutzergruppen verwalten (CRUD)

Test 1: Benutzergruppe hinzufügen und löschen		**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)		
Beschreibung:	**Erwartung:**		

1. Ins Tab Security wechseln. 2. Ins Tab 'User Groups' wechseln. 3. Eine Benutzergruppe hinzufügen. 4. Eine andere Benutzergruppe auswählen und löschen. 5. Den Refresh-Button betätigen.	1. Das Tab Security wird angezeigt. 2. Eine Liste mit allen Benutzergruppen ist sichtbar. 3. Die hinzugefügte Benutzergruppe erscheint in der Liste. 4. Die Benutzergruppe wird aus der Liste gelöscht. 5. Es verändert sich höchstens die Reihenfolge der Benutzergruppen.
Fehler, Abweichung:	
Kommentar:	

Test 2: Benutzer-Zuweisung verändern	**Resultat**	**OK**
Vorbedingung:	UC8 Benutzergruppen verwalten CRUD: Test 1: Benutzergruppen hinzufügen und löschen	
Beschreibung:	**Erwartung:**	
1. Eine Benutzergruppe auswählen. 2. Einen nicht zugewiesenen Benutzer zuweisen. 3. Einen zugewiesenn Benutzer entfernen.	1. Die zugewiesenen Benutzer und alle noch nicht zugewiesenen Benutzer werden angezeigt. 2. Der Benutzer verschwindet und erscheint neu in der Liste der zugewiesenen Benutzer. 3. Der Benutzer verschwindet und erscheint neu in der Liste der nicht zugewiesenen Benutzer.	
Fehler, Abweichung:		
Kommentar:		

4.9 UC09: Rollen verwalten (CRUD)

Test 1: Rolle hinzufügen, bearbeiten und löschen	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	
Beschreibung:	**Erwartung:**	
1. Ins Tab Security wechseln. 2. Ins Tab 'Roles' wechseln. 3. Eine Rolle hinzufügen. 4. Eine andere Rolle auswählen und löschen. 5. Eine dritte Rolle auswählen und den Rollennamen sowie die Beschreibung verändern. 6. Den Refresh-Button betätigen.	1. Das Tab Security wird angezeigt. 2. Eine Liste mit allen Rollen ist sichtbar. 3. Die hinzugefügte Rolle erscheint in der Liste. 4. Die Rolle wird aus der Liste gelöscht. 5. Die Änderungen sind auch beim Verlassen der Zelle sichtbar. 6. Es verändert sich höchstens die Reihenfolge der Benutzergruppen.	
Fehler, Abweichung:		
Kommentar:		

Test 2: Berechtigungen verändern	**Resultat**	**OK**

Vorbedingung:	UC9 Rollen verwalten CRUD: Test 1: Rolle hinzufügen, bearbeiten und löschen
Beschreibung:	**Erwartung:**
1. Eine Rolle auswählen. 2. Einen nicht gesetzte Berechtigung zuweisen. 3. Speichern-Button betätigen. 4. Einen gesetzte Berechtigung entfernen. 5. Speichern-Button betätigen. 6. Eine Änderung vornehmen und den Refresh-Button betätigen.	1. Alle Berechtigungen werden angezeigt, die gesetzten mit einem Häkchen. 2. Die Berechtigung bekommt ein Häkchen und der Speichern-Button ist enabled. 3. Die Liste verändert sich nicht und der Speichern-Button ist disabled. 4. Das Häkchen der Berechtigung wird entfernt und der Speichern-Button ist enabled. 5. Die Liste verändert sich nicht und der Speichern-Button ist disabled. 6. Die Änderung wird wieder zurückgesetzt.
Fehler, Abweichung:	
Kommentar:	

4.10 UC10: Generelle Berechtigungen verwalten

Test 1: Vorbereitung um generelle Berechtigungen zu verwalten.	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	
Beschreibung:	**Erwartung:**	
1. Ins Tab Security wechseln. 2. Ins Tab 'Application Permissions' wechseln. 3. Tests zu UC12 und UC13 durchführen	1. Das Tab Security wird angezeigt. 2. Alle User und Gruppen ohne zugewiesenen Rechte werden links und alle anderen rechts angezeigt. 3. Siehe Erwartung.	
Fehler, Abweichung:		
Kommentar:		

4.11 UC11: Projekt/Ordner-spezifische Berechtigungen verwalten

Test 1: Vorbereitung um Projekt/Ordner-spezifische Berechtigungen zu verwalten.	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	
Beschreibung:	**Erwartung:**	
1. Beim Browser ins Tab 'Node Permissions' wechseln. 2. Im Browser-Tree den Root-Node selektieren. 3. Tests zu UC12 und UC13 durchführen	1. Falls der Root-Node bereits selektiert ist, stimmt die Erwartung mit Punkt 2 überein. 2. Alle User und Gruppen ohne zugewiesenen Rechte werden links und alle anderen rechts angezeigt. 3. Siehe Erwartung.	
Fehler, Abweichung:		
Kommentar:		

4.12 UC12: Rolle zu Benutzer zuweisen

Test 1: Einem Benutzer eine Rolle zuweisen und entfernen.	**Resultat**	-
Vorbedingung:	Kann nicht einzeln ausgeführt werden, ist ein Sub-UseCase von UC10 und UC11.	

Beschreibung:	**Erwartung:**
1. Ins Tab 'Users' wechseln. 2. Einen noch nicht zugewiesenen Benutzer auswählen und den Assign-Button betätigen. 3. Den zugewiesenen Benutzer auswählen. 4. Eine noch nicht zugewiesene Rolle auswählen und den Assign-Button bei den Rollen betätigen. 5. Eine bereits zugewiesene Rolle auswählen und den Unassign-Button betätigen. 6. Einen anderen Node im Tree selektieren, wieder den Root-Node selektieren und den vorhin hinzugefügten Benutzer.	1. Alle verfügbaren (noch nicht zugewiesenen) Benutzer werden angezeigt. 2. Der Benutzer verschwindet links und wird bei den zugewiesenen Benutzern angezeigt. 3. Die verfügbaren (noch nicht zugewiesenen) Rollen werden angezeigt. 4. Die Rolle verschwindet links und wird bei den zugewiesenen Rollen angezeigt. 5. Die Rolle verschwindet rechts und wird bei den noch nicht zugewiesenen Rollen angezeigt. 6. Es sind immer noch dieselben Benutzer und Rollen zugewiesen.

Fehler, Abweichung:	
Kommentar:	

4.13 UC13: Rolle zu Benutzergruppe zuweisen

Test 1: Einer Benutzergruppe eine Rolle zuweisen und entfernen.	**Resultat**	-
Vorbedingung:	Kann nicht einzeln ausgeführt werden, ist ein Sub-UseCase von UC10 und UC11.	

Beschreibung:	**Erwartung:**
7. Ins Tab 'Users Groups' wechseln. 8. Eine noch nicht zugewiesene Benutzergruppe auswählen und den Assign-Button betätigen. 9. Die zugewiesene Benutzergruppe auswählen. 10. Eine noch nicht zugewiesene Rolle auswählen und den Assign-Button bei den Rollen betätigen. 11. Eine bereits zugewiesene Rolle auswählen und den Unassign-Button betätigen. 12. Einen anderen Node im Tree selektieren, wieder den Root-Node selektieren und die vorhin hinzugefügte Benutzergruppe auswählen.	7. Alle verfügbaren (noch nicht zugewiesenen) Benutzergruppen werden angezeigt. 8. Die Benutzergruppe verschwindet links und wird bei den zugewiesenen Benutzergruppen angezeigt. 9. Die verfügbaren (noch nicht zugewiesenen) Rollen werden angezeigt. 10. Die Rolle verschwindet links und wird bei den zugewiesenen Rollen angezeigt. 11. Die Rolle verschwindet rechts und wird bei den noch nicht zugewiesenen Rollen angezeigt. 12. Es sind immer noch dieselben Benutzergruppen und Rollen zugewiesen.

Fehler, Abweichung:	
Kommentar:	

4.14 UC14: History bereinigen

Wird nicht umgesetzt.

4.15 Übersicht

Use Case	*Implementiert*	*Fehler/Unschönheit*	*Status*
UC01: BOS-Server verbinden	ja		**OK**
UC02: XWP-Projekte browsen	ja	Beim Element bei dem beim Laden die Verbindung getrennt wurde, werden auch nach dem Wiederaufbau der Verbindung keine Unterelemente geladen.	**OK**
UC03: BOS-Server-Verbindung trennen	ja		**OK**
UC04: Auschecken rückgängig machen (Undo check-out)	ja		**OK**
UC05: History ansehen	gestrichen		-
UC06: Suchen	gestrichen		-
UC07: Benutzer verwalten (CRUD)	ja		**OK**
UC08: Benutzergruppen verwalten (CRUD)	ja		**OK**
UC09: Rollen verwalten (CRUD)	ja		**OK**
UC10: Generelle Berechtigungen verwalten	ja		**OK**
UC11: Projekt/Ordner-spezifische Berechtigungen verwalten	ja		**OK**
UC12: Rolle zu Benutzer zuweisen	ja		**OK**
UC13: Rolle zu Benutzergruppe zuweisen	ja		**OK**
UC14: History bereinigen	gestrichen		-

5 Verbesserungsmöglichkeiten

5.1 Bekannte Einschränkungen

- Beim Undo-checkout auf einem Projekt-Node im Tree wird danach der „unsichtbare“ Projekt-Data-Node selektiert, der als kleiner runder Punkt erscheint.

Projekt: BOS-WebAdmin

Final Release - Testdokumentation

[Dokumentstruktur basiert auf RUP „Dokument Test Evaluation Summary“]

0 Dokumentinformationen

0.1 Änderungsgeschichte

Datum	Version	Änderung	Autor
01.10.2009	0.1	Dokument erstellt und Test für UC1, UC2 und UC3 beschrieben.	Reto Schneebeli
22.11.2009	0.2	Dokument mit dem Gerüst der restlichen UseCases erweitert. Test für UC07 beschrieben.	Reto Schneebeli
22.11.2009	0.3	Test für UC01, UC07, UC08 angepasst	Kaspar Fenner
07.12.2009	0.4	Test für UC9, UC10, UC11, UC12 und UC13 beschrieben.	Reto Schneebeli
16.12.2009	1.0	Testausführung für Final-Release	Reto Schneebeli

0.2 Inhalt

0 Dokumentinformationen 2
0.1 Änderungsgeschichte 2
0.2 Inhalt 2
1 Voraussetzungen 3
2 Vorbereitungen 3
3 Testdurchführung 3
4 Systemtest 3
4.1 UC01: BOS-Server verbinden 3
4.2 UC02: XWP-Projekte browsen 4
4.3 UC03: BOS-Server-Verbindung trennen 5
4.4 UC04: Auschecken rückgängig machen (Undo check-out) 5
4.5 UC05: History ansehen 6
4.6 UC06: Suchen 6
4.7 UC07: Benutzer verwalten (CRUD) 6
4.8 UC08: Benutzergruppen verwalten (CRUD) 7
4.9 UC09: Rollen verwalten (CRUD) 8
4.10 UC10: Generelle Berechtigungen verwalten 9
4.11 UC11: Projekt/Ordner-spezifische Berechtigungen verwalten 9
4.12 UC12: Rolle zu Benutzer zuweisen 9
4.13 UC13: Rolle zu Benutzergruppe zuweisen 10
4.14 UC14: History bereinigen 11
4.15 Übersicht 11
5 Verbesserungsmöglichkeiten 11
5.1 Bekannte Einschränkungen 11

1 Voraussetzungen

Ein laufender BOS-Server und die entsprechenden Zugangsdaten (Adresse, Username, Passwort). Eine Verbindung mit dem BOS-Server kann aufgebaut werden, z.B. mit dem XWP Projekt Manager.

2 Vorbereitungen

Keine notwendig.

3 Testdurchführung

Datum: 07.12.2009
SVN Revisionnummer: 476
Tester: Reto Schneebeli
Hardware: Notebook (Core 2 Duo 2GHz, 4GB Ram)
Betriebssystem: Windows 7 Professional, 64 Bit
Browser: Firefox 3.5.4
BOS Server: WmWare localhost

4 Systemtest

Den BOS-WebAdmin starten, bedeutet immer den Browser mit der BOS-WebAdmin Seite öffnen.

4.1 UC01: BOS-Server verbinden

Test 1: BOS Server verbinden (einfach)	**Resultat**	**OK**
Beschreibung:	**Erwartung:**	
1. BOS-WebAdmin starten. 2. BOS-Server auswählen. 3. Korrekten Benutzernamen und Passwort eingeben. 4. Button „Login“ betätigen.	1. Die Login-Seite wird angezeigt. 2. Keine Fehlermeldung oder Verzögerung. 3. Login Button wird enabled. 4. Projektbaum wird in weniger als 4 Sekunden angezeigt. Und die 'Successfull loged in" Statusmeldung verschwindet nach einigen Sekunden.	
Fehler, Abweichung:		
Kommentar:		

Test 2: BOS Server verbinden (falsche Benutzerdaten)	**Resultat**	**OK**
Beschreibung:	**Erwartung:**	
1. BOS-WebAdmin starten. 2. BOS-Server auswählen. 3. Falschen Benutzernamen oder falsches Passwort eingeben. 4. Button „Login“ betätigen.	1. Die Login-Seite wird angezeigt. 2. Keine Fehlermeldung oder Verzögerung. 3. Login Button wird enabled. 4. Fehlermeldung wird angezeigt und Login-Seite bleibt.	
Fehler, Abweichung:		
Kommentar:		

Test 3: BOS Server verbinden (erweitert)	**Resultat**	**OK**
Beschreibung:	**Erwartung:**	
1. BOS-WebAdmin starten. 2. BOS-Server auswählen. 3. Benutzernamen und Passwort eingeben. 4. Wählen dass die Anmeldeinformationen gespeichert werden sollen. 5. Button „Login“ betätigen. 6. Logout-Button betätigen. 7. Refresh Button im Browser betätigen. 8. Abwählen, so dass die Anmeldeinformationen nicht gespeichert werden sollen. 9. Refresh Button im Browser betätigen.	1. Die Login-Seite wird angezeigt. 2. Keine Fehlermeldung oder Verzögerung. 3. Login Button wird enabled. 4. Keine Fehlermeldung oder Verzögerung. 5. Projektbaum wird angezeigt. 6. Die Login-Seite wird angezeigt mit den vorhin eingegebenen Anmeldeinformationen. 7. Die Login-Seite wird angezeigt mit den vorhin eingegebenen Anmeldeinformationen. 8. Keine Fehlermeldung oder Verzögerung. 9. Die Login-Seite wird angezeigt, aber der Benutzername und das Passwort ist leer.	
Fehler, Abweichung:		
Kommentar:		

Test 4: BOS Server hinzufügen/löschen/ändern	**Resultat**	**OK**
Beschreibung:	**Erwartung:**	
1. BOS-WebAdmin starten. 2. BOS-Server Liste anklicken. 3. Einen neuen Server hinzufügen. 4. Einen bestehenden Server löschen. 5. Einen Server bearbeiten. 6. Refresh Button im Browser betätigen.	1. Die Login-Seite wird angezeigt. 2. Keine Fehlermeldung oder Verzögerung. 3. Server erscheint in der Server-Liste. 4. Server wird aus der Server-Liste gelöscht. 5. Geänderte Serverdaten werden in der Server-Liste angezeigt. 6. Server-Liste beinhaltet den hinzugefügten und den veränderten Server mit den korrekten Angaben.	
Fehler, Abweichung:		
Kommentar:		

4.2 UC02: XWP-Projekte browsen

Test 1: Projektbaum navigieren.	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	
Beschreibung:	**Erwartung:**	

1. Vorbedingung: Mit dem BOS Server verbinden (einfach) 2. Ordner und Projekte im Baum öffnen, schliessen. 3. Ordner im Baum selektieren. 4. Projekt im Baum selektieren.	1. Projektbaum wird angezeigt und die erste Ebene ist bereits aufgeklappt. 2. Die Unterordner oder Projekte werden angezeigt. Die Unterelemente werden immer eine Ebene tiefer vorgeladen, das GUI darf jedoch nie einfrieren. 3. Entsprechende Detaildaten werden rechts angezeigt. 4. Entsprechende Detaildaten werden rechts angezeigt.
Fehler, Abweichung:	
Kommentar:	

Test 2: Verbindung beim Navigieren trennen	**Resultat**	**OK**
Vorbedingung:	Test 1: Projektbaum navigieren.	
Beschreibung:	**Erwartung:**	
1. Die Verbindung zum BOS Server trennen (Netzwerkverbindung unterbrechen, WmWare pausieren oder IIS stoppen) . 2. Einen weiteren Ordner öffnen, welcher Unterelemente hat und noch nicht geöffnet wurde. 3. Die Verbindung zum BOS Server wieder herstellen. 4. Einen weiteren Ordner öffnen, welcher Unterelemente hat und noch nicht geöffnet wurde.	1. Nichts passiert. 2. Eine entsprechende Fehlermeldung wird angezeigt. 3. Nichts passiert. 4. Die Fehlermeldung verschwindet und das navigieren funktioniert wieder.	
Fehler, Abweichung:		
Kommentar:		

4.3 UC03: BOS-Server-Verbindung trennen

Test 1: Verbindung trennen	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	
Beschreibung:	**Erwartung:**	
1. Button „Logout“ betätigen.	1. Die Login-Seite wird angezeigt.	
Fehler, Abweichung:		
Kommentar:		

4.4 UC04: Auschecken rückgängig machen (Undo check-out)

Test 1: Undo-Checkout Node	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	
Beschreibung:	**Erwartung:**	

1. Undo-Checkout auf einem einzelnen Node starten. 2. Commit-Button betätigen.	1. Ein kleines Fenster mit dem Node für das Undo-Checkout wird angezeigt. 2. Das Undo-Checkout Icon verschwindet und der Node ist nicht mehr ausgecheckt. Falls es ein virtueller Node war, wie z.B. die Netzwerkdaten oder Paneldaten wird das Undo-Checkout auf dem ganzen virtuellen Node durchgeführt.
Fehler, Abweichung:	
Kommentar:	

Test 1: Undo-Checkout Projekt		**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)		
Beschreibung:	**Erwartung:**		
1. Undo-Checkout auf einem ganzen Projekt starten. 2. Commit-Button betätigen.	1. Ein kleines Fenster mit allen Nodes des Projektes für das Undo-Checkout wird angezeigt. 2. Das Undo-Checkout Icon verschwindet auf allen Nodes des Projektes und alle Nodes sind nicht mehr ausgecheckt.		
Fehler, Abweichung:			
Kommentar:			

4.5 UC05: History ansehen

Wird nicht umgesetzt.

4.6 UC06: Suchen

Wird nicht umgesetzt.

4.7 UC07: Benutzer verwalten (CRUD)

Test 1: Benutzer hinzufügen, bearbeiten und löschen		**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)		
Beschreibung:	**Erwartung:**		
1. Ins Tab Security wechseln. 2. Ins Tab Users wechseln. 3. Einen Benutzer hinzufügen. 4. Den Benutzer auswählen und den Vornamen, den Nachnamen, die Email-Adresse und die Beschreibung verändern. 5. Die Veränderung speichern. 6. Einen anderen Benutzer auswählen und löschen. 7. Den Refresh-Button betätigen.	1. Das Tab Security wird angezeigt. 2. Eine Liste mit allen Benutzern ist sichtbar. 3. Der hinzugefügte Benutzer erscheint in der Benutzerliste. 4. Ein Speichern-Button wird angezeigt. 5. Die Änderungen wurden gespeichert. 6. Der Benutzer wird aus der Liste gelöscht. 7. Es verändert sich höchstens die Reihenfolge der Benutzer, alle anderen Daten bleiben erhalten.		
Fehler, Abweichung:			
Kommentar:			

Test 2: Benutzergruppen-Zuweisung verändern	**Resultat**	**OK**
Vorbedingung:	UC7 Benutzer verwalten CRUD: Test 1: Benutzer hinzufügen, bearbeiten und löschen	
Beschreibung:	**Erwartung:**	
1. Einen Benutzer auswählen. 2. Einen nicht zugewiesene Benutzergruppe zuweisen. 3. Eine zugewiesenen Benutzergruppe entfernen.	1. Die zugewiesenen Gruppen und alle noch nicht zugewiesenen Gruppen werden angezeigt. 2. Die Benutzergruppe verschwindet und erscheint neu in der Liste der zugewiesenen Benutzergruppen. 3. Die Benutzergruppe verschwindet und erscheint neu in der Liste der nicht zugewiesenen Benutzergruppen.	
Fehler, Abweichung:		
Kommentar:		

Test 3: Benutzer-Passwort verändern	**Resultat**	**OK**
Vorbedingung:	UC7 Benutzer verwalten CRUD: Test 1: Benutzer hinzufügen, bearbeiten und löschen	
Beschreibung:	**Erwartung:**	
1. Einen Benutzer auswählen. 2. Den Change-Password-Button drücken. 3. Das alte, sowie neue Passwort eingeben und abbrechen. 4. Das alte und neue Passwort eingeben und bestätigen.	1. Der Change-Password-Button wird aktiv. 2. Die Textboxen für die Eingabe des alten und neuen Passwortes werden eingeblendet. 3. Die Textboxen von Schritt 2 werden ausgeblendet. 4. Die Textboxen werden ausgeblendet und das Passwort wurde geändert (überprüfen mit einem Re-Login).	
Fehler, Abweichung:		
Kommentar:		

4.8 UC08: Benutzergruppen verwalten (CRUD)

Test 1: Benutzergruppe hinzufügen und löschen	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	
Beschreibung:	**Erwartung:**	
1. Ins Tab Security wechseln. 2. Ins Tab 'User Groups' wechseln. 3. Eine Benutzergruppe hinzufügen. 4. Eine andere Benutzergruppe auswählen und löschen. 5. Den Refresh-Button betätigen.	1. Das Tab Security wird angezeigt. 2. Eine Liste mit allen Benutzergruppen ist sichtbar. 3. Die hinzugefügte Benutzergruppe erscheint in der Liste. 4. Die Benutzergruppe wird aus der Liste gelöscht. 5. Es verändert sich höchstens die Reihenfolge der Benutzergruppen.	
Fehler, Abweichung:		
Kommentar:		

Test 2: Benutzer-Zuweisung verändern		**Resultat**	**OK**
Vorbedingung:	UC8 Benutzergruppen verwalten CRUD: Test 1: Benutzergruppen hinzufügen und löschen		
Beschreibung:		**Erwartung:**	
1. Eine Benutzergruppe auswählen. 2. Einen nicht zugewiesenen Benutzer zuweisen. 3. Einen zugewiesenn Benutzer entfernen.		1. Die zugewiesenen Benutzer und alle noch nicht zugewiesenen Benutzer werden angezeigt. 2. Der Benutzer verschwindet und erscheint neu in der Liste der zugewiesenen Benutzer. 3. Der Benutzer verschwindet und erscheint neu in der Liste der nicht zugewiesenen Benutzer.	
Fehler, Abweichung:			
Kommentar:			

4.9 UC09: Rollen verwalten (CRUD)

Test 1: Rolle hinzufügen, bearbeiten und löschen		**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)		
Beschreibung:		**Erwartung:**	
1. Ins Tab Security wechseln. 2. Ins Tab 'Roles' wechseln. 3. Eine Rolle hinzufügen. 4. Eine andere Rolle auswählen und löschen. 5. Eine dritte Rolle auswählen und den Rollennamen sowie die Beschreibung verändern. 6. Den Refresh-Button betätigen.		1. Das Tab Security wird angezeigt. 2. Eine Liste mit allen Rollen ist sichtbar. 3. Die hinzugefügte Rolle erscheint in der Liste. 4. Die Rolle wird aus der Liste gelöscht. 5. Die Änderungen sind auch beim Verlassen der Zelle sichtbar. 6. Es verändert sich höchstens die Reihenfolge der Benutzergruppen.	
Fehler, Abweichung:			
Kommentar:			

Test 2: Berechtigungen verändern		**Resultat**	**OK**
Vorbedingung:	UC9 Rollen verwalten CRUD: Test 1: Rolle hinzufügen, bearbeiten und löschen		
Beschreibung:		**Erwartung:**	

1. Eine Rolle auswählen. 2. Einen nicht gesetzte Berechtigung zuweisen. 3. Speichern-Button betätigen. 4. Einen gesetzte Berechtigung entfernen. 5. Speichern-Button betätigen. 6. Eine Änderung vornehmen und den Refresh-Button betätigen.	1. Alle Berechtigungen werden angezeigt, die gesetzten mit einem Häkchen. 2. Die Berechtigung bekommt ein Häkchen und der Speichern-Button ist enabled. 3. Die Liste verändert sich nicht und der Speichern-Button ist disabled. 4. Das Häkchen der Berechtigung wird entfernt und der Speichern-Button ist enabled. 5. Die Liste verändert sich nicht und der Speichern-Button ist disabled. 6. Die Änderung wird wieder zurückgesetzt.
Fehler, Abweichung:	
Kommentar:	

4.10 UC10: Generelle Berechtigungen verwalten

Test 1: Vorbereitung um generelle Berechtigungen zu verwalten.	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	
Beschreibung:	**Erwartung:**	
1. Ins Tab Security wechseln. 2. Ins Tab 'Application Permissions' wechseln. 3. Tests zu UC12 und UC13 durchführen	1. Das Tab Security wird angezeigt. 2. Alle User und Gruppen ohne zugewiesenen Rechte werden links und alle anderen rechts angezeigt. 3. Siehe Erwartung.	
Fehler, Abweichung:		
Kommentar:		

4.11 UC11: Projekt/Ordner-spezifische Berechtigungen verwalten

Test 1: Vorbereitung um Projekt/Ordner-spezifische Berechtigungen zu verwalten.	**Resultat**	**OK**
Vorbedingung:	UC1 BOS Server verbinden: Test 1: BOS Server verbinden (einfach)	
Beschreibung:	**Erwartung:**	
1. Beim Browser ins Tab 'Node Permissions' wechseln. 2. Im Browser-Tree den Root-Node selektieren. 3. Tests zu UC12 und UC13 durchführen	1. Falls der Root-Node bereits selektiert ist, stimmt die Erwartung mit Punkt 2 überein. 2. Alle User und Gruppen ohne zugewiesenen Rechte werden links und alle anderen rechts angezeigt. 3. Siehe Erwartung.	
Fehler, Abweichung:		
Kommentar:		

4.12 UC12: Rolle zu Benutzer zuweisen

Test 1: Einem Benutzer eine Rolle zuweisen und entfernen.		**Resultat**	-
Vorbedingung:	Kann nicht einzeln ausgeführt werden, ist ein Sub-UseCase von UC10 und UC11.		
Beschreibung:		**Erwartung:**	
1. Ins Tab 'Users' wechseln. 2. Einen noch nicht zugewiesenen Benutzer auswählen und den Assign-Button betätigen. 3. Den zugewiesenen Benutzer auswählen. 4. Eine noch nicht zugewiesene Rolle auswählen und den Assign-Button bei den Rollen betätigen. 5. Eine bereits zugewiesene Rolle auswählen und den Unassign-Button betätigen. 6. Einen anderen Node im Tree selektieren, wieder den Root-Node selektieren und den vorhin hinzugefügten Benutzer.		1. Alle verfügbaren (noch nicht zugewiesenen) Benutzer werden angezeigt. 2. Der Benutzer verschwindet links und wird bei den zugewiesenen Benutzern angezeigt. 3. Die verfügbaren (noch nicht zugewiesenen) Rollen werden angezeigt. 4. Die Rolle verschwindet links und wird bei den zugewiesenen Rollen angezeigt. 5. Die Rolle verschwindet rechts und wird bei den noch nicht zugewiesenen Rollen angezeigt. 6. Es sind immer noch dieselben Benutzer und Rollen zugewiesen.	
Fehler, Abweichung:			
Kommentar:			

4.13 UC13: Rolle zu Benutzergruppe zuweisen

Test 1: Einer Benutzergruppe eine Rolle zuweisen und entfernen.		**Resultat**	-
Vorbedingung:	Kann nicht einzeln ausgeführt werden, ist ein Sub-UseCase von UC10 und UC11.		
Beschreibung:		**Erwartung:**	
7. Ins Tab 'Users Groups' wechseln. 8. Eine noch nicht zugewiesene Benutzergruppe auswählen und den Assign-Button betätigen. 9. Die zugewiesene Benutzergruppe auswählen. 10. Eine noch nicht zugewiesene Rolle auswählen und den Assign-Button bei den Rollen betätigen. 11. Eine bereits zugewiesene Rolle auswählen und den Unassign-Button betätigen. 12. Einen anderen Node im Tree selektieren, wieder den Root-Node selektieren und die vorhin hinzugefügte Benutzergruppe auswählen.		7. Alle verfügbaren (noch nicht zugewiesenen) Benutzergruppen werden angezeigt. 8. Die Benutzergruppe verschwindet links und wird bei den zugewiesenen Benutzergruppen angezeigt. 9. Die verfügbaren (noch nicht zugewiesenen) Rollen werden angezeigt. 10. Die Rolle verschwindet links und wird bei den zugewiesenen Rollen angezeigt. 11. Die Rolle verschwindet rechts und wird bei den noch nicht zugewiesenen Rollen angezeigt. 12. Es sind immer noch dieselben Benutzergruppen und Rollen zugewiesen.	
Fehler, Abweichung:			
Kommentar:			

4.14 UC14: History bereinigen

Wird nicht umgesetzt.

4.15 Übersicht

Use Case	*Implementiert*	*Fehler/Unschönheit*	*Status*
UC01: BOS-Server verbinden	ja		**OK**
UC02: XWP-Projekte browsen	ja		**OK**
UC03: BOS-Server-Verbindung trennen	ja		**OK**
UC04: Auschecken rückgängig machen (Undo check-out)	ja		**OK**
UC05: History ansehen	gestrichen		-
UC06: Suchen	gestrichen		-
UC07: Benutzer verwalten (CRUD)	ja		**OK**
UC08: Benutzergruppen verwalten (CRUD)	ja		**OK**
UC09: Rollen verwalten (CRUD)	ja		**OK**
UC10: Generelle Berechtigungen verwalten	ja		**OK**
UC11: Projekt/Ordner-spezifische Berechtigungen verwalten	ja		**OK**
UC12: Rolle zu Benutzer zuweisen	ja		**OK**
UC13: Rolle zu Benutzergruppe zuweisen	ja		**OK**
UC14: History bereinigen	gestrichen		-

5 Verbesserungsmöglichkeiten

5.1 Bekannte Einschränkungen

- Im Browser-Tree existiert ein „unsichtbarer“ Projekt-Data-Node, welcher selektiert werden kann und dann als kleiner runder Punkt erscheint.

Projekt: BOS-WebAdmin

Erfahrungsberichte

1 Erfahrungsbericht von Reto Schneebeli

Da ich berufsbegleitend an der HSR studiere, spielte ich schon lange mit dem Gedanken, als Industriepartner für die Studienarbeit den eigenen Arbeitsgeber (Siemens Schweiz AG, Zug) zu fragen. Zufälligerweise hat Kaspar Fenner, welcher auch schon mein Partner beim SE2-Projekt war, erst vor Kurzem eine Stelle in derselben Abteilung angenommen. Somit waren wir zwei natürlich prädestiniert, um zusammen ein Projekt für die eigene Abteilung als Studienarbeit durchzuführen. Deshalb haben wir bei unserem Teamchef und den Projektleitern angefragt, ob sie ein geeignetes Thema hätten. Glücklicherweise konnten wir ein gutes Thema finden, welches den Anforderungen einer Studienarbeit genügt. Die Aufgabenstellung lautet, einen Webclient in Silverlight zu entwickeln, um gewisse Administrations-Funktionen auf dem BOS (Branch Office Server) auszuführen. Als Betreuer hatten wir schon lange Professor Hansjörg Huser im Auge, da er als Dozent der Microsoft Technologien-Vorlesung über viel Wissen in diesem Themengebiet verfügt.

Vor der Studienarbeit hatte ich noch nie etwas in Silverlight oder WPF programmiert. Erfahrung in C# konnte ich an der HSR und vor allem in der Siemens schon viel sammeln. Neu an Silverlight waren für mich somit eigentlich nur die GUI-Erstellung mit XAML und das abgespeckte .Net Framework. Um uns möglichst schnell das nötige Wissen über Silverlight anzueignen, haben wir dafür einen zwei-Tägigen Kurs an der HSR besucht. Den Rest haben wir uns währen den ersten Wochen parallel zur Projektplanung und Anforderungsplanung angeeignet. Ich war von Anfang an vor allem für das User-Interface verantwortlich. Deshalb habe ich auch schnell bemerkt, dass Silverlight im GUI-Bereich einiges weniger bietet als WPF. Einige fehlende Controls stehen jedoch durch das *Silverlight Toolkit* von Codeplex trotzdem zur Verfügung. Wir haben von Anfang an einen sehr hohen Wert auf eine saubere und einfach erweiterbare Architektur gesetzt. Dadurch sind wir auch auf die *Composite Application Guidance* für WPF und Silverlight von den Microsoft *patterns & practices* gestossen. Ich habe darauf ihre Ansätze, Patterns, Beispiele und die Library dazu studiert und eine Technologie-Studie verfasst. Wir haben aus ihrer Library bereits im Prototyp die *Named-Locations (View-Injection)*, *die Dependency-Injection* und die *Event-Aggregation* verwendet, weil damit viel unnötige Kopplung vermieden werden kann und die GUI-Erstellung vereinfacht wird.

Wir haben uns zum Ziel gesetzt, in der Mitte des Projektes einen lauffähigen Prototypen über alle Layer hinweg zu haben, um das Design zu verifizieren und zu beweisen, dass der WCF Zugriff auf den BOS auch über Silverlight funktioniert. Das war im Nachhinein ein sehr guter Entscheid, weil wir dadurch viele Probleme mit Silverlight schon frühzeitig erkannt haben und auch lösen konnten. Die Fertigstellung des Prototypen hat aber für uns beide kurz vor dem Meilenstein „End of Elaboration“ einen enormen Arbeitseinsatz abverlangt. Ich habe in den beiden Woche davor 27 bzw. 31 Stunden gearbeitet, neben den drei Modulen an der HSR und jeweils etwa zwei Tage arbeiten. Das ging auch nur weil das Thema sehr Interessant war und das Entwickeln mit Silverlight wirklich Spass gemacht hat.

Es war ein sehr Interessantes und auch überaus lehrreiches Projekt. Ich habe wirklich einiges in Sachen sauberer Architektur und natürlich Silverlight dazu gelernt. Zum Beispiel die asynchrone Service-Schnittstelle, damit das User-Interface reaktionsfähig bleibt, das ganze Data-Binding und alle benutzten Komponenten der *Composite Application Library*. Natürlich war es auch anstrengend, aber das ist bei solch einem Projekt eigentlich meistens der Fall. Schlussendlich habe ich mit 290 Stunden auch deutlich mehr als die vorgegebenen 240 Stunden investiert.

Der BOS-WebAdmin ist noch lange nicht abgeschlossen und wird uns noch länger beschäftigen. Der Abteilungsleiter ist sehr Interessiert und wir dürfen den Webclient sogar dem Management vorstellen und später auch dem Marketing sowie den *DESIGO* Ländervertretern. Höchstwahrscheinlich wird er Bestandteil der Version 5 von XWORS plus, was mich natürlich sehr freut.

2 Erfahrungsbericht von Kaspar Fenner

Es ist nicht das erste Projekt, welches ich gemeinsam mit meinem Studien- und Arbeitskollegen Reto Schneebeli durchgeführt habe. Dabei hat sich wieder einmal gezeigt, wie gut die Zusammenarbeit zwischen uns funktioniert. Vieles läuft mittlerweile schon fast automatisch ab. Dafür gibt es halt auch die üblichen Differenzen, die sich aber zum Glück meist relativ einfach ausdiskutieren lassen. Ich empfinde die Zusammenarbeit jedenfalls immer als sehr positiv, da sich unsere Stärken und Schwächen in vielen Bereich sehr gut ergänzen. Dokumentation schreiben macht zwar keiner von uns besonders gerne, aber sonst sind die Arbeiten immer sehr schnell unter uns aufgeteilt.

In diesem Projekt habe ich mich vor allem um die Client-Server-Kommunikation mit WCF sowie Design und Implementation des Domain-Layers gekümmert. Dadurch habe ich leider eher weniger Erfahrungen mit dem UI-Design und insbesondere auch mit XAML machen können. Da ich aber sonst auch schon viel mit GUIs gemacht habe und auch die XAML-Grundlagen aus dem Silverlight-Kurs bereits kenne, stört mich das eigentlich auch nicht gross. Die wichtigsten Erfahrungen haben wir auch immer unter uns ausgetauscht, sodass beide davon profitieren konnten. Ich konnte jedenfalls wieder viel Neues dazulernen während dieser Studienarbeit. Angefangen hat es wie bereits erwähnt mit dem Silverlight-Kurs an der HSR, den wir kurz vor Beginn der Studienarbeit besucht haben. Dabei wurden vor allem die Grundlagen des GUI-Designs mit XAML und Silverlight vermittelt. Während der Studienarbeit habe ich dann viel mit WCF gemacht und dabei auch einiges dazugelernt, obwohl ich die WCF-Grundlagen bereits von der Schule und Arbeit her kenne. Ich musste recht tief in die Thematik einsteigen, um eine geeignete Lösung für die eher mässige WCF-Unterstützung von Silverlight zu finden. Auch die gesamte Palette an Sicherheitskonfigurationen mit Zertifikaten, Verschlüsselung und Authentifizierung habe ich durchgespielt, bis alles so funktioniert hat, wie es sollte.

Das Interessanteste für mich ist aber immer die Softwarearchitektur selbst. Dort besonders schöne Lösungen zu finden, gehört zu meinen liebsten Beschäftigungen. Bei diesem Projekt konnte ich besonders viel über asynchrone und serviceorientierte Architekturen lernen. Dabei konnte ich auch eine Vielzahl von Patterns und Technologien einsetzten.

Besonders gefallen hat mir auch das Unit-Testing und insbesondere das Mocking mit Moq, denn das war für mich absolut neu. Ich habe zuvor schon einiges mit Unit-Testing gemacht, aber meistens ist mir das ziemlich mühsam vorgekommen. Aber durch das konsequente Design mit Interfaces und der Verwendung von Moq bei den Uni-Tests, machte es richtig Spass. Sinnvolle Unit-Tests zu schreiben ist definitiv nicht einfach, aber ich konnte doch einiges an Erfahrung sammeln und fühle mich jetzt viel sicherer beim Unit-Testing. Und Moq ist für mich, nicht übertrieben, das absolute Highlight der ganzen Projektarbeit, jedenfalls aus technologischer Sicht. Ich bin überzeugt, dass mir diese Library in Zukunft noch viel Freude bereiten wird.

Aus Projektmanagement-Sicht war die beste Entscheidung, einige weniger wichtige UseCases wegzulassen und dafür die Qualität und die Präsentation des User-Interfaces zu steigern. Nach dem bisher erhaltenen Feedback scheint dies genau die richtige Strategie gewesen zu sein. Kritik muss ich mir aber gerade bei der Zeitplanung eingestehen, denn diese war schon nicht immer gerade optimal. Praktisch alles benötigte zum Schluss mehr Zeit als angenommen. Da hätte man sicher etwas besser machen können und noch etwas mehr Reserven einplanen. Die Meilensteine haben wir aber immer eingehalten und auch die geforderten Resultate geliefert. Der wichtigste Durchbruch war dabei sicher der Meilenstein „End of Elaboration“, bei dem wir den Prototypen abgeschlossen hatten. Dieser Prototyp hat wesentlich zum guten Gelingen dieses Projektes beigetragen. Diese Erfahrung werde ich bestimmt auch bei zukünftigen Projekten wieder einsetzten können.

Es würde mich freuen, wenn der BOS-WebAdmin bald auch produktiv eingesetzt werden könnte.

Glossar

Begriff	**Beschreibung**
AS	**A**utomation **S**tation
BOS	**B**ranch **O**ffice **S**erver (Zentrale Ablage der XWORKS Projekte)
CAG	**C**omposite **A**pplication **G**uidance (Designpatterns und eine Library welche einem unterstützen beim erstellen von modulare WPF oder Silverlight Applikationen.)
CAL	**C**omposite **A**pplication **L**ibrary (Eine Library welche einem unterstützen beim Erstellen von modularen WPF oder Silverlight Applikationen.)
MS	**M**anagement **S**tation
MSDN	**M**icro**s**oft **D**eveloper **N**etwork (Informations- und Software-Angebot für Programmierer und Software-Architekten, die sich mit Microsoft-Technologien beschäftigen.)
.NET	.NET ist eine von Microsoft entwickelte Softwareplattform. Mit dem .NET-Framework wird eine gut ausgestattete Library für die Entwicklung mit .NET bereitgestellt.
OOA	**O**bjekt**o**rientierte **A**nalyse
OOD	**O**bjekt**o**rientiertes **D**esign
RIA	**R**ich **I**nternet **A**pplications
WCF	**W**indows **C**ommunication **F**oundation
WPF	**W**indows **P**resentation **F**oundation
XAML	E**x**tensible **A**pplication **M**arkup **L**anguage
XWP	DESIGO **XW**ORKS **p**lus ist eine Engineering-Umgebung im Bereich Gebäudeautomation der Siemens BT.
XWORKS plus	Siehe XWP.

Literaturverzeichnis

- Composite Application Guidance für WPF und Silverlight von Microsoft, http://msdn.microsoft.com/en-us/library/dd458809.aspx, Stand 14.10.2009
- Composite Application Guidance Technical Concepts, http://msdn.microsoft.com/en-us/library/dd458876.aspx, Stand 15.10.2009
- Lösung für SQL-Verbindungsprobleme nach Upgrade von ASP.NET 2.0 auf .NET 3.5, http://stackoverflow.com/questions/1236923/upgraded-net-webapp-cant-open-connection-to-database/1566676, Stand 14.10.2009
- Lösung für die Benutzung von NUnit 2.5.1 mit Silverlight 3 (gepatchte DLLs), http://wesmcclure.tumblr.com/post/152727000, Stand 02.11.2009
- Moq Mocking-Library auf Google Code, http://code.google.com/p/moq/, Stand 17.12.2009
- Rhino.Mocks Mocking-Framework, http://www.ayende.com/projects/rhino-mocks.aspx, Stand 17.12.2009
- NMock dynamic mock object library for .NET, http://www.nmock.org, Stand 17.12.2009
- Binding in Silverlight von DarksideCookie, http://chris.59north.com/post/Bindings-the-glue-that-connects-the-view-to-the-viewmodel-part-1.aspx, Stand 20.11.2009
- Lösung für den fehlenden PropertyChanged Trigger von .NET Rocker, http://www.thomasclaudiushuber.com/blog/2009/07/17/here-it-is-the-updatesourcetrigger-for-propertychanged-in-silverlight, Stand 20.11.2009
- BoolToVisiblility Binding-Konverter von Stefan Cruysberghs, http://www.scip.be/index.php?Page=ArticlesNET22#VisibilityConverter, Stand 25.11.2009